Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

# Machine Learning Based Vulnerability Detection in Internet of Things Operating System's Codes

Babangida Zachariah[1], Philip O. Odion[2], Peter Ayuba[3], Isaac Samson[4], Solomon Matthew Karma[5]
[1,4]Computer Science Department, Kaduna State University, Tafawa Balewa Way, Kaduna, Nigeria.
[2]Computer Science Department, Nigerian Defence Academy, Afaka, Kaduna, Nigeria.
[3,5]Mathematical Sciences Department, Kaduna State University, Tafawa Balewa Way, Kaduna, Nigeria.
babangida.zachariah@kasu.edu.ng[1], poodion@nda.edu.ng[2], ayubng@kasu.edu.ng[3],
samson.isaac@kasu.edu.ng[4], solomon.karma@kasu.edu.ng[5]

## Abstract

***Background:*** *the proliferation of IoT devices has had a significant impact on our daily lives. However, the adoption and use of these devices is still faced with challenges that border around security and privacy of the users. Research efforts have focused on detecting security issues in transmission of packets in the wireless sensor networks, which these IoT devices operate optimally to achieve their mission. One of the major security and privacy concerns in such systems that is yet to receive significant attention is the vulnerabilities in the devices' operating systems. These vulnerabilities, once exploited, security and privacy are compromised, and the cost associated with this is high.* ***Aim:*** *this research aimed to explore the use of machine learning in the detection of vulnerabilities in IoT devices' operating systems source code.* ***Method****: Modelled as a supervised learning problem, Al-Boghdady's labelled dataset was used to train and test Linear Support Vector Classifier (LinearSVC), Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD), Adaptive Boosting (AdaBoost) and Extreme Trees Classifier (Extra-Trees).* ***Results****: LinearSVC achieved the best performance in the Binary classification tasks with test accuracy of 97.83%, weighted F1-Score of 0.98 and macro F1-Score of 0.96. In the multiclass classification, Extra-Trees achieved the best performance compared to other stated classifiers by achieving 97.52% classification accuracy, weighted F1-Score of 0.98 and macro F1-Score of 0.99. Compared to the benchmark classifier (Random Forest), it achieved comparable performance. Therefore, it is recommended for deployment following that it has the capability of minimizing overfitting problems. The implication of these results is that it helps in the identification of source code level vulnerabilities in order that they might be fixed and thereby, enhancing the security and privacy of IoT devices and networks.*

**Keywords:** *detection, security and privacy, vulnerability, internet of things*

## 1. Introduction

The world of Internet of Things (IoT) is continuously experiencing increasing devices miniaturization, computing, network, and energy harvesting capabilities. Thus, their widespread adoption and deployments explains their ubiquity; since they are more useful and usable in diverse fields from monitoring/surveillance, health, aquatic, etc. (Balsamo et. al., 2019; Tayus, et. al., 2022). Although IoT systems are ubiquitous, they are faced with several challenges such as constrained energy supply, compute infrastructure, memory and most importantly, the security and privacy of the systems and their associated networks (Joe, 2020; Al-Boghdady, Wassif & El-Ramly, 2021; Gaouzi & Chougdali, 2022; Yang, 2022). These challenges have attracted a lot of research efforts that are aimed at addressing one or more of these challenges. Particularly, the challenge of security and privacy of the IoT systems and their associated networks affects their acceptability and deployments, especially when it is to be used in area where data security and privacy is critical (Zachariah et. al., 2016; Mahari, 2022; Tyagi & Kumar, 2022). Therefore, the spurring interest from the

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

research community and the industry.

In addressing the security and privacy challenges of IoT systems, solutions have focused on device level (Malina et. al. 2019), cloud level (Priyantharsini et. al. 2022), network level packet transmission and reception (Xiong, Sarwate, & Mandayam, 2022), conditional power dissipation (Myridakis et. al. 2020), and only a few have considered the source code level (Al-Boghdady, El-Ramly, & Wassif, 2022; Bier et. al., 2021; Zhang, 2020). The source code level of the IoT operating systems as well as the application specific source code usually contains vulnerabilities that can be exploited by malicious agents (And et. al., 2020). These vulnerabilities may be memory based such as buffer overflows, string based such as strcpy; others may include cross-site scripting, inadequate encryption, and authentication mechanisms, as well as firmware/source code updates (Zhang, 2020). Source codes vulnerabilities may be as a result of weaknesses in programming languages such as C/C++ or insecure coding practices by developers and pose threats, since it may result in denial-of-service attacks, data leakages, and general compromise of the IoT systems.

It has been identified that IoT systems, particularly devices, are vulnerable to Remote Code Execution (RCE) as well as Denial of Service (DoS) attacks or Distributed Denial of Service (DDoS) attacks. It has also been reported that the exploitation of these vulnerabilities results in huge monetary costs and otherwise. For example, (Andra, 2023) presented a report that in United State of America alone, an estimate of about $623.7 million was lost as cost to system compromise; and it was predicted that over $10.5 trillion is expected to be the annual cost by 2025. These compromises are not unconnected to device security, which essentially involves implementation techniques that may be considered unsecured.

One of the most recent solution paradigms that is gaining momentum in the attempts to uncover and solve IoT systems related vulnerabilities is the machine learning paradigm. This involves the application of various machine leaning techniques for the development of intelligent models that detect vulnerabilities in IoT systems networks and system codes (Injadat, Moubayed, & Shami, 2020). However, due to the numerous dimensions of the vulnerabilities arising from networks, access controls, and systems source codes, there is no one-size-fits-all solution to the problems. Therefore, this research focuses on the application of linear and non-linear machine learning techniques for the detection of IoT systems source code vulnerabilities. Particularly, this research examines Linear Support Vector Classification (LinearSVC), Support Vector Classier (SVC), Stochastic Gradient Descent (SGD) Classifier, Adaptive Boosting (AdaBoost), and Extra-Trees algorithms in source code vulnerability detection. The source code implementation of these algorithms is provided in the Github Repository with universal resource locator (URL): https://github.com/babangidazachariah/IoTOSCodesVulnerability.

The remaining part of this paper is organized as follows: section 2 presents back-ground of the machine learning algorithms considered in this work, as well as the related works, section 3 presents the description of datasets and methodology, section 4 presents the results and comparison between the models as well as with the benchmark model on the used dataset and section 5 presents the conclusion and future work.

## 2. Related Works

All first level section headings must be number as above, in sentence case, in Times New Roman and font Following that most problems can be modelled using simple linear models, this paper considered a number of linear models for the detection of IoT source codes vulnerabilities. Also, noting that linear models may be limited in their capacity to model the complexities that may be inherent in real world problems, necessitating the need for nonlinear models with better capacities. This research considered ensemble learning techniques, which are known to have better capacities at handling non-linear relationships that may exist in a system. This section briefly presents the considered models.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

### 2.1 Support Vector Classifier

Support vector machine (SVM), LinearSVC and SVC are often used for both binary and multiclass classification tasks. LinearSVC is a linear model whose goal is to determine the best hyperplane that optimally demarcates between the different classes of the task at hand. The capacity of LinearSVC has provided it a place in many classification tasks, where is performed competitively (Yildirim et. al., 2019; Zambang, Jiang, & Wahab, 2021). LinearSVC models a task as a linear model shown in equation 1 and learn the parameters w and b by minimizing the objective function shown in equation 2 while trying to maximize the linear hyperplane's margin between classes of the task.

$$wX + b = 0 \tag{1}$$

$$L(w,b) = \min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(max(0, y_i(wx_i + b))\right) \tag{2}$$

Where $L$ is the objective function, $C$ is regularization parameter trade-off between maximizing the hyperplane and training error; and $\|w\|^2$ is the norm of the weight vector.
The SVC model transforms data in higher-dimensional space to effectively handle non-linear decision boundaries in a task. This transformation is achieved using kernel functions such as linear, radial, polynomial or sigmoid basis functions.

### 2.2 Stochastic Gradient Descent Classifier

The SGD classifier is a linear classifier with capabilities to handle non-linearity in a task. It uses stochastic gradient technique in minimizing the loss function while learning the parameters of the hypothesis function. One of the major advantages of SGD classifier is that it has the capacity for online learning. Therefore, it has been applied in many classification problems (Elshoush & Dinar, 2019; Kabir et. al., 2015). Stochastic gradient technique learns the parameters of the by updating the parameters as in equation 3.

$$\theta_{i+1} = \theta_i - \eta\nabla l(\theta) \quad \text{for } i = 1, 2, \ldots \tag{3}$$

Where $\theta_i$ is the parameter vector, $\eta$ is the learning rate and $l(\theta)$ depends on the whether a logistic, hinge, Huber, perceptron, etc.

### 2.3 Adaptive Boost Classifier

AdaBoost is an ensemble learning algorithm that takes advantage of weak learners through their combination to formulate a stronger classifier. Given the dataset $(x_i, y_i)$ for $i = 1, 2, \ldots, m$ AdaBoost computes weighted error using equation 4 and chooses weight of weak learners using equation 5 and then updates parameters of training sample using equation 6.

$$\epsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i] \tag{4}$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{5}$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp\left(-\alpha_t y_i h_t(x_i)\right) \tag{6}$$

Where $h_t(x_i)$ is the hypothesis function, $D_t(i)$ is the distribution and $Z_t$ is the normalization factor chosen or estimated for the distribution $D_t(i)$. With these, the hypothesis function then calculated using equation 7.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

$$H(x) = -\left(\sum_{i=1}^{T} \alpha_t h_t(x_i)\right) \quad (7)$$

With the capacity of AdaBoost as a machine learning model, having the capacity to model linear and non-linear relationships in data, it has found application in various task and have been reported to have performed competitively (Guo & Wang, 2023; Mehta et. al., 2023).

### 2.4 Extra-Trees Classifier

Extra-Trees algorithm is an ensemble learning technique that uses a top-down approach in creating unpruned decision trees while splitting nodes of the trees using randomized attributes and cut point selection. That is, it builds several decision trees and then ensemble their individual predictions to arrive at the final prediction decision. Extra-Trees essentially start with the user-defined number of decisions trees and growing the trees using random subset of the training data and a random subset of features as well as a random threshold for each feature. To determine the feature that would result in the best split, either information gain or Gini impurity is used as a metric. The splitting of the nodes of the trees continues recursively until a stopping criterion is met. At the combination phase, averages or voting technique is used as a combination approach for the final prediction.

Extra-Trees have been found to perform better than their decision trees and random forest algorithm counterparts since they are able to minimize overfitting problems that are often associated with decision trees and random forest. Also, Extra-Trees can achieve higher performance since they increase diversity of the trees in the ensemble. With their capacities and abilities of achieving less generalization errors, they have been applied in different prediction problems and reported to have produced competitive results (Sharma et. al., 2019; Zafari et.al., 2019). Extra-Trees uses a cross-entropy loss function for the classification tasks and mean squared error for the regression tasks. These are often not a major concern as Extra-Trees are considered as black-box models, where emphasis is on performance and not choice of loss function.

With the performances of these machine learning algorithms as recorded in diverse fields where machine learning is applicable, it motivates this research effort to assess their performances in the detection of vulnerabilities in IoT systems source codes. One important algorithm in the class of ensemble learning is random forest. However, its inclusion of highly correlated or redundant features often affects its performance, especially for high-dimensional data with a lot of noise. Therefore, this research did not consider it for implementation as it has been implemented in Al-Boghdady, El-Ramly, & Wassif (2022) and would be used as a benchmark for assessing the considered algorithms.

Following that security and privacy of IoT systems is of utmost importance, a lot of research has been done to provide new and enhanced solutions to the problem. Generally, research in IoT systems may be categorized into device level security, network and cloud level security, or firmware/source code level security (Zainuddin et. al. 2021). Also, for each category, various techniques have been proposed towards enhancing the security and privacy of IoT systems. However, this review focuses on the application of machine learning and static analysis of IoT systems source codes for identification of vulnerabilities.

In Oser et. al. (2022), a security assessment framework for embedded-device risks was presented. It was developed to provide a partially automated assessment of IoT device and network risks by considering past vulnerabilities history and patches of devices based on models. It was reported that the system achieved an accuracy of 92.83% at device identification, and successful identification of possible risks. Also, in (Gaouzi & Chougdadli, 2022), focus was directed to the security of communication channels and configurations of IoT devices operating with cloud-based end.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

In Alnaeli et. al. (2016) and Alnaeli et. al. (2017), static analysis techniques and tools were performed on C/C++ program codes to identify potential vulnerable statements. This was done on different open-source program codes that consisted of varying number of program files. It was reported that most of the vulnerable statements in program codes included the memcpy, and strlen.

In He et al. (2022), a homology-based vulnerability detection in IoT devices source code was proposed. Using clonal algorithm for selection numerical and structural features source code vulnerabilities were generated from limited dataset and used in the training of the proposed detector model. The advantage of this approach was that the performance of the model depended on objective function and not on the limited dataset used. The results reported showed that the clonal algorithm for selection achieved better performances in terms of accuracy and recall in all the experiments performed compared to the neural network model used as benchmark. The limitation of this approach is that the choice of objective function for most problems has proven to be a tough task. Also, numerical and structural features may not be adaptable and results in lower efficiency.

In Zhang (2020), the limitation of numerical and structural features in the detection of vulnerabilities in IoT systems programs were highlighted and an approach that uses vulnerability and target function was proposed. Both functions receive the output of a reverse binary analysis of the IoT system program to generate string and numerical features, which are used by a locality sensitivity hashing technique called MinHash. The results of the MinHash LSH fore and screening are then fed into a kNearest Neighbor (kNN), which performs optimization and detection of vulnerabilities in the program based on previously learnt vulnerabilities in the used datasets. The limitation of this approach is in the high sensitivity of MinHash LSH. It is known to be too sensitive to thresholds, data skewness and does not provide adequate flexibility for dealing with multiple constraints.

In Liu (2022), a vulnerability detection solution of IoT software supply chain was proposed. A multi-model network that combined graph matching networks and pseudocode graphing to build customized features from structural and behavioral characteristics of a software. The graph matching networks then uses the customized features in the detection process of vulnerabilities in the software. Reported results showed that the approach achieved better performance at the detection of vulnerabilities than other approaches such as Asteria, VulSeeker, etc. The limitation of this approach is that graph matching networks are known to have scalability challenges due to memory and computational constraints. They are also sensitive to graph structure and do not handle noisy data effectively.

In Zhao (2022), vulnerabilities associated with third-party components of IoT binary code for Tsmart were considered and a large-scale empirical analysis tool was proposed. The aim was to detect vulnerabilities in source codes and to curate dataset for further research. Binwalk and Iterative Disassembler (IDA) plugins were customized to support unpacking and disassembling of interesting binary codes. Also, binwalk and IDA generated syntactical and control flow graph features of the binary code and then a database of vulnerabilities is used for the detection of vulnerabilities in the program code. A comparison of the performance of the proposed approach to Gemini, and Context-Free Grammar based techniques, the results show that the proposed technique achieved better performance in terms of recall. The limitation of this approach is in the use of IDA, which has limited support for obfuscated code as well as high false positive rates when there are high complex structures in the code.

In Al-Boghdady, El-Ramly, & Wassif, (2022), Random Forest (RF), Convolutional Neural Networks (CNN) and Recurrent Neural network (RNN) models were used in detection of vulnerabilities in IoT systems source codes. It was reported that RF had the best classification performance in terms of accuracy, macro and weighted F1-scores. However, seeing the high dimensionality in dataset, and with the limited ability of RF to adequately manage such, as well the high-dimensionality of data, it presents an opportunity to assess the possible use of a model that is known to have capability to handle imbalance and high-dimensionality in

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

datasets.

From the reviewed related literatures, it clear that although machine learning techniques have been proposed for the detection of IoT systems codes vulnerabilities, there are challenges that relates to the volume of datasets used in the training of the models. This limitation usually has an impact in the generalizability of machine learning models. Also, another challenge is that although models have achieved high accuracy in the detection of vulnerabilities of IoT systems codes, they are yet to attain the desired level of performance. Thus, research efforts aimed at improving the detection of most of the vulnerabilities becomes a necessity. This research is an attempt to contribute to this direction.

## 3. Methodology

As many other machine-learning based research, the first step in this research was the acquisition of dataset; the second was data preprocessing; third step was the choice of machine learning algorithms; the fourth was training, validation, and testing of the models. This section provides the description of the methodology taken towards achieving the goal of this research.

### 3.1 Data Acquisition

Most researchers have identified limited available datasets is one of the major challenges in the development of machine learning models for the prediction of vulnerabilities in IoT systems OS codes. However, the goal of this research was not to curate new datasets but to use an existing dataset for the development of the machine learning model considered in this research. The dataset used in this research was curated by Al-Boghdady, El-Ramly, & Wassif, (2022).

The dataset is source in part from the C/C++ SARD database and the other part from actual IoT operating systems code snippets of various IoT devices. The operating systems considered for the identification of vulnerabilities included RIOT, Contiki, FreeRTOs, and Amazon FreeRTOs. Different versions of these operating systems, ranging from year 2015 to year 2020 were considered as shown in Figure 1.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

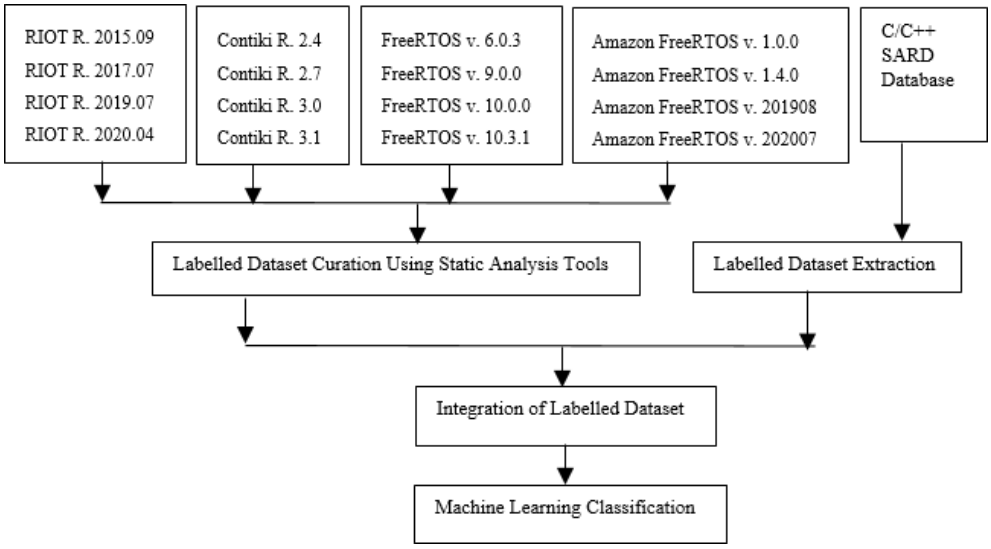



Figure 1: Dataset Curation Process

To identify the vulnerabilities in the source codes, Common Weakness Enumeration (CWE) from Common Vulnerability Exposure (CVE) was used as benchmark. A total of 2626 vulnerable codes were identified in the IoT systems OS based on the CWE benchmark and 2491 benign codes were acquired from SARD. This brought the labeled dataset to a total of 5117 used for training and validation of machine learning models. The training examples count based on labels in presented in Figure 2. Another 322 codes were provided differently and used for testing the performance of the developed model on unseen dataset. The testing dataset consisted of 48 benign source codes and 274 vulnerable source codes from TinyOS for IoT systems. The test examples count based on labels is presented in Figure 3. The detailed description of dataset is provided in Al-Boghdady, El-Ramly, & Wassif, (2022).

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

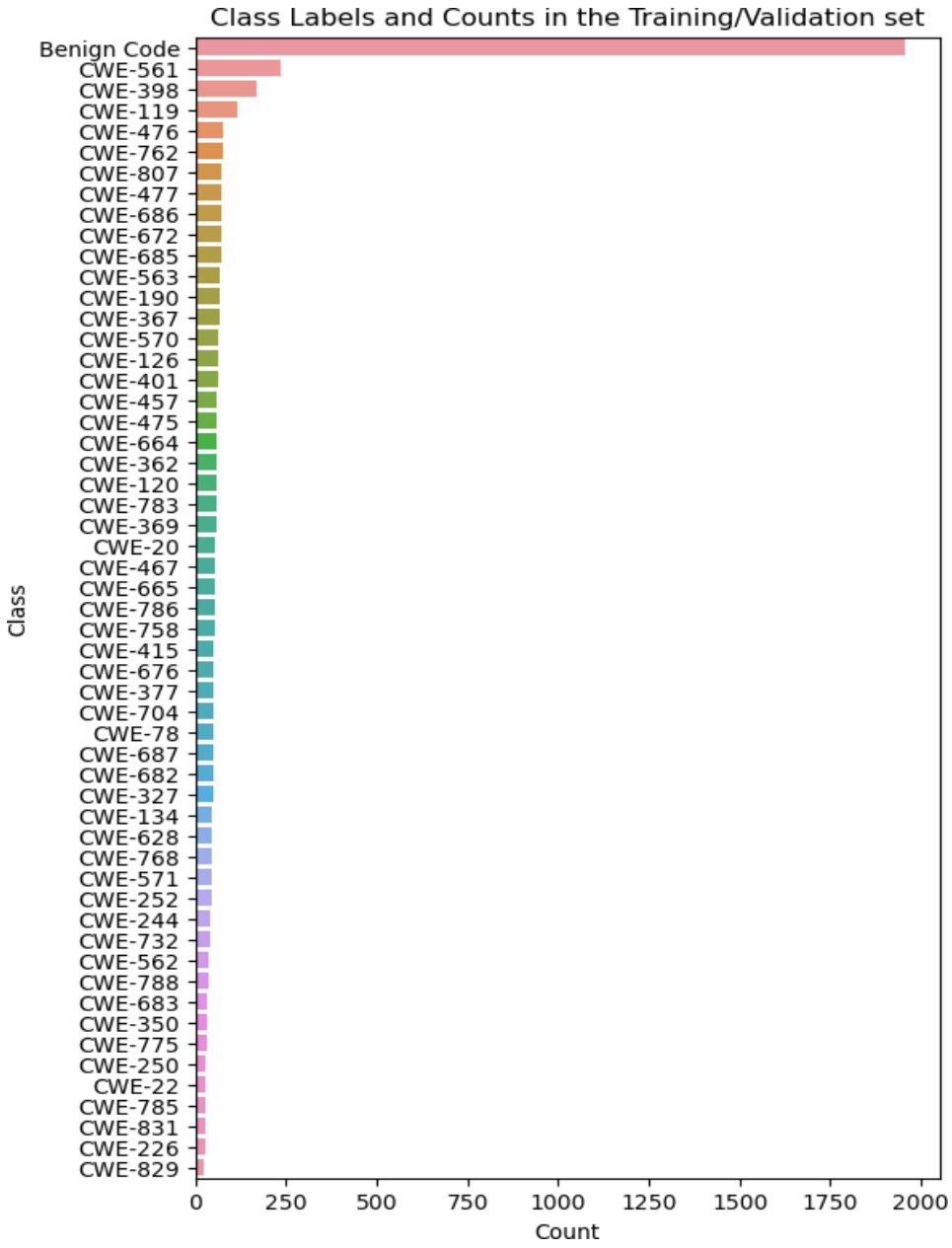



Figure 2: Class Labels and Counts in the Training/Validation set

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

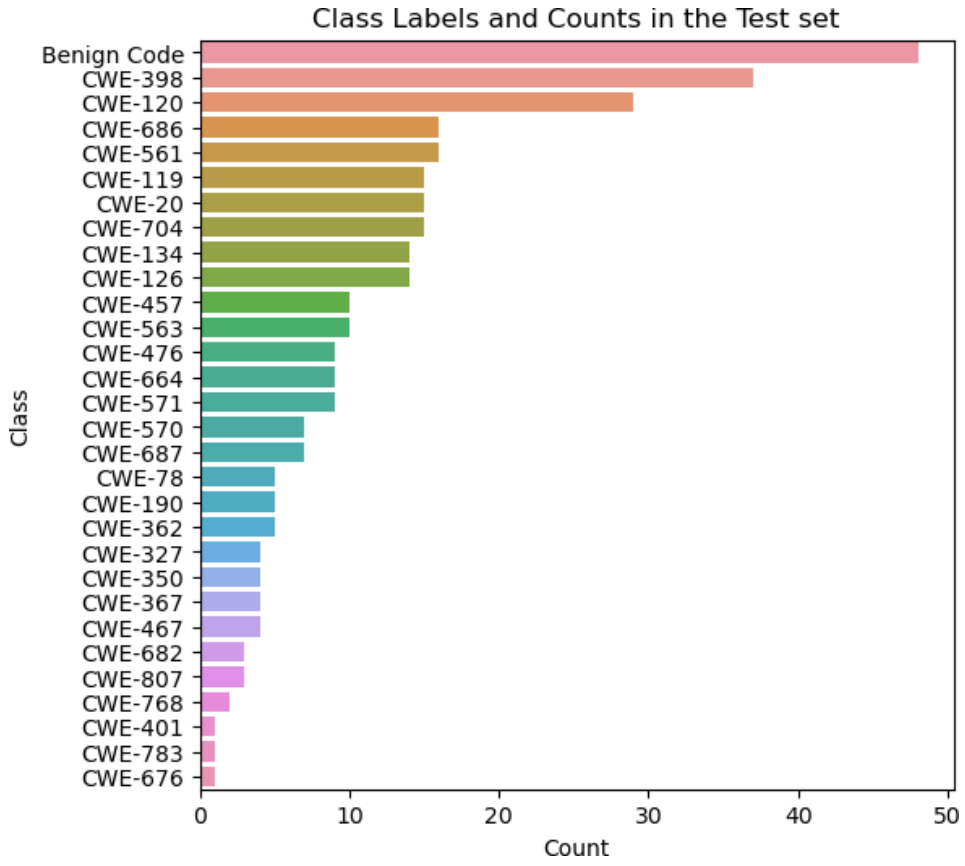



Figure 3: Class Labels and Counts in the Test set

### 3.2 Data Preprocessing

With the dataset acquired, the next step was to preprocess the data. In this research, the first preprocessing step involved the removal of training examples with null/empty feature(s) as well as duplicates. After this step, the training dataset of 5117 dropped to 4810 training examples.

The next step in the data preprocessing was the tokenization of the source code. For this, tokens were created using regular expressions, where multiple whitespaces in the source code were removed. After tokenization, vectorization was performed using word statistics-based feature extraction algorithm known as Term Frequency-Inverse Document Frequency (TF-IDF) (Liu et. al. 2018).

### 3.3 Machine Learning Algorithms

For this research, LinearSVC, kernel-based SVC, SGD, AdaBoost, and Extra-Trees were chosen as machine learning algorithms for the classification vulnerability detection in IoT operating systems source codes. The choice was motivated by their reported performances in most classification tasks.

In terms of model parameters, LinearSVC model was implemented with regularization strength (C) value of 120, balanced class weight, squared-hinge loss, 15000 maximum iterations, and was trained using 25-fold cross-validation; for SVC model, the C parameter was set to 100, scaled gamma, kernel was poly, and the training used 25-fold cross-validation; for SGD model, the regularization parameter (alpha) was set to 1e-05, hinge loss, 1150 maximum iterations, and 10-fold cross-validation; for AdaBoost, the base estimator

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

was set to Random Forest classifier, with "gini" criterion, max features was set to "None", minimum sample split was set to 2, and number or estimators for the random forest was set to 100 and 10 for AdaBoost model, the model was trained using 20-fold cross-validation; and for Extra-Trees model was implemented with 300 estimators, gini as criterion, maximum depth was set to "None", minimum sample split was set to 2, random state was 42, and was trained using 30-fold cross-validation. It should be noted that these parameters were arrived at after several hyper parameter tuning. Thus, these may not be the best of the parameters.

### 3.4 Training, Testing and Validation of Models

For the training of the models chosen for this research, the training dataset was divided into training and validation in the ratio of 70:30. That is, 3367 training examples of the 4810 training datasets were used for the training and the remaining 1443 training examples were used for the validation. As previously stated, the testing phase had 322 test examples.

In terms of model parameters and hyper-parameter tuning, Grid Search Cross Validation (GSCV) was used to determine the most appropriate hyper-parameters for each model.

After preliminary training of the models with randomly chosen model parameters, k-fold cross-validation training approach was used for the training of the models. To determine the best k-fold cross-validation, training was done in a sequence of steps of 5 to 25. Based on the observed results, the k-fold cross-validation was chosen for each model for the final training phase.

For the testing phase of the models, this research used accuracy, and F1-score was used to evaluate the performance of the models. Particularly, weighted F1-score, and Macro F1-score were used. These F1-scores were considered for the evaluation of the models because they dataset has class/target variable imbalance, and each class is equally important.

## 4. Results and discussion

For all machine learning models, there is the training and validation phase that seeks to determine suitable parameters of the model given the training set. These parameters or training phase aims at minimizing the difference between the training and test (or generalization) error. This section presents the evaluation metrics used for evaluating the machine learning models developed in this research as well as the models' performances in training and testing in this research. The evaluation metrics used in this research are Precision and Recall, which are used to compute the F1-Scores for each model. Therefore, a confusion matrix is computed for each model. Contextually, correct classification of an IoT OS source code as vulnerable is define as True Positive (TP), the misclassification of any of the source code snippet without vulnerability as vulnerable is False Positive (FP), the correct classification of code snippet without vulnerability as non-vulnerable is True Negative (TN), and the misclassification of vulnerable code snippet as non-vulnerable is False Negative (FN). With these parameters other necessary parameters are computed, False Positive Rate, FPR $= \frac{FP}{(TN+FP)}$; the False Negative Rate, FNR $= \frac{FN}{(FN+TP)}$ ; the accuracy, Acc $= \frac{(TP+TN)}{(TP+TN+FP+FN)} * 100\%$; Precision, P $= \frac{TP}{(TP+FP)}$; Recall, R $= \frac{TP}{(TN+FN)}$; and F1 $= 2 * \frac{P*R}{(P+R)}$. Extending the Precision, Recall and F1-Scores to the multiclass classification, their averages are computed using equations 8, 9, 10 and 11.

$$Average\ Precision = \frac{1}{N}\sum_{n=1}^{N} Precision_n \quad (8)$$

$$Average\ Recall = \frac{1}{N}\sum_{n=1}^{N} Recall_n \quad (9)$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

$$mF1 = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{2 * Precision_n * Recall_n}{Precision_n + Recall_n}\right] \quad (10)$$

$$wF1 = \frac{1}{\sum_{n=1}^{N} X_n}\sum_{n=1}^{N} X_n\left[\frac{2 * Precision_n * Recall_n}{Precision_n + Recall_n}\right] \quad (11)$$

Based on the stated evaluation metrics, the evaluation of the models developed in this research is done for the binary and multi-class classification tasks. The models used for binary classification were the same models, in terms of parameters, used for multi-class classification. The goal of binary classifiers in this research was to classify a given IoT OS source code as either (or containing) a vulnerability or not. The goal of the multiclass classifier was to classify IoT OS source code as being or containing vulnerability or not, and in case of vulnerability, it should show what kind of vulnerability is it. This is important as vulnerabilities are not all at the same scale of threats. They are often classed as either low, high or critical risks vulnerabilities. The training and testing performances of the classifiers is presented in Table 1.

Table 1. Binary-Class Classification Results

| | Training Accuracy (%) | Test Accuracy (%) | Weighted F1 | Macro F1 |
|---|---|---|---|---|
| LinearSVC | 98.75 | 97.83 | 0.98 | 0.96 |
| SVC | 98.50 | 97.21 | 0.97 | 0.95 |
| SGD | 98.84 | 94.72 | 0.94 | 0.87 |
| AdaBoost | 99.02 | 96.90 | 0.97 | 0.93 |
| Extra-Trees | 98.71 | 96.58 | 0.96 | 0.93 |

### 4.1 LinearSVC

The goal of binary classifiers in this research was to classify a given IoT OS source code as either (or containing) a vulnerability or not. The training and testing performances of the classifiers is presented in Table 1. From Table 1, the training of LinearSVC model for binary classification achieved an accuracy of 98.75%. The k-fold training performance is as shown in Figure 4(a).

The test accuracy of the LnearSVC was 97.83% with a weighted F1-Score of 0.98, which shows that the model performed well in terms of precision and recall across all the classes of the test set; and Macro F1-Score of 0.96, which shows that has high precision and recall for each of the class.

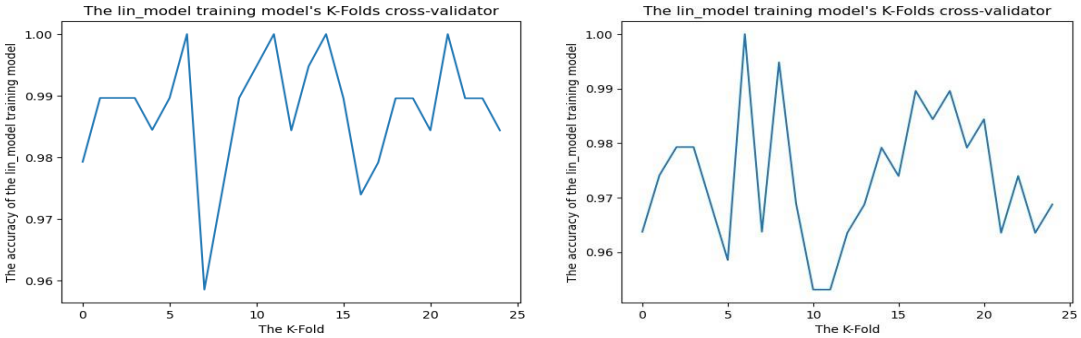
Also, from Table 2, the performance of the LinearSVC classifier in the multiclass classification task shows that it achieved 97.36% accuracy in the training and validation set shown in Figure 4(b), 93.79% in the test set, weighted F1-Score of 0.93 and macro F1-Score of 0.87. This shows that the model performed well in terms of precision and recall across all classes, however, it did not perform well for each class.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Table 2. Multi-Class Classification Results

| | Training Accuracy (%) | Test Accuracy (%) | Weighted F1 | Macro F1 |
|---|---|---|---|---|
| LinearSVC | 97.36 | 93.79 | 0.93 | 0.87 |
| SVC | 96.70 | 94.72 | 0.94 | 0.95 |
| SGD | 95.22 | 93.48 | 0.93 | 0.91 |
| AdaBoost | 97.42 | 96.89 | 0.97 | 0.97 |
| Extra-Trees | 97.24 | 97.52 | 0.98 | 0.99 |




(a) Binary (b) Multi-Class

Figure 4: LinearSVC Classifier Training Performance

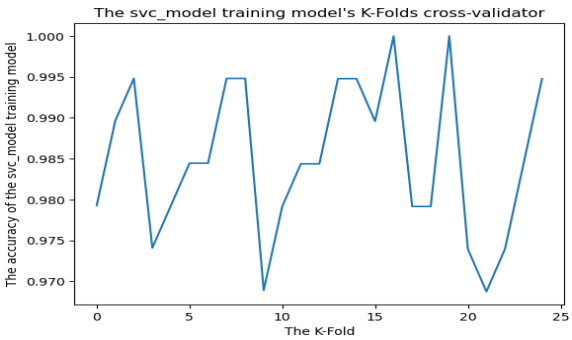
## 4.2 SVC Model

From Table 1, the training of SVC model for binary classification achieved an accuracy of 98.50%. The k-fold training performance is as shown in Figure 5(a). The test accuracy of the SVC was 97.21% with a weighted F1-Score of 0.97, which shows that the model performed well in terms of precision and recall across all the classes of the test set; and Macro F1-Score of 0.95, which shows that the model had high precision and recall for each of the class.

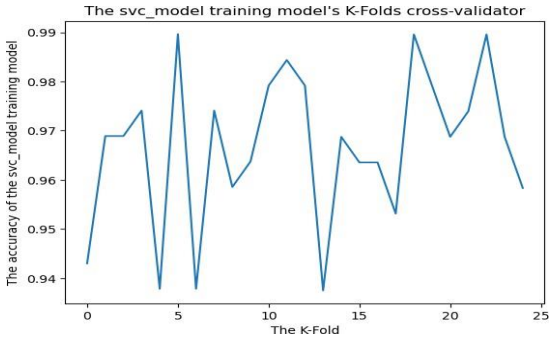
Also, from Table 2, the performance of the SVC classifier in the multi-class classification task shows that it achieved 96.70% accuracy in the training and validation set shown in Figure 5(b), 94.72% in the test set, weighted F1-Score of 0.94 and macro F1-Score of 0.95. This shows that the model performed well in terms of precision and recall across all classes; also, it did perform acceptably well for each class.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng




(a) Binary



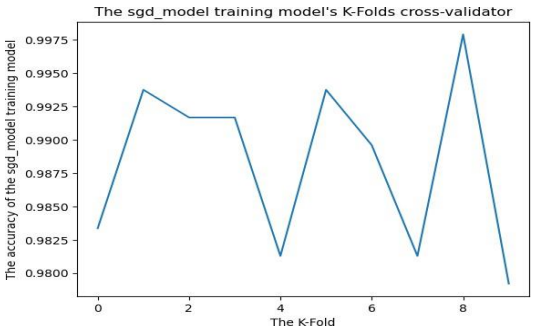

(b) Multi-Class

Figure 5: SVC Classifier Training Performance

### 4.3 SGD Model

From Table 1, the training of SGD model for binary classification achieved an accuracy of 98.84%. The k-fold training performance is as shown in Figure 6(a). The test accuracy of the SVC was 94.72% with a weighted F1-Score of 0.94, which shows that the model performed well in terms of precision and recall across all the classes of the test set; and Macro F1-Score of 0.87, which shows that the model did not had high precision and recall for each of the classes.

Also, from Table 2, the performance of the SGD classifier in the multi-class classification task shows that it achieved 95.22% accuracy in the training and validation set shown in Figure 6(b), 93.48% in the test set, weighted F1-Score of 0.93 and macro F1-Score of 0.91. This shows that model performed well in terms of precision and recall across all classes; also, it did perform acceptably well for each class.




(a) Binary


(b) Multi-Class

Figure 6: SDG Classifier Training Performance

### 4.4 AdaBoost Model

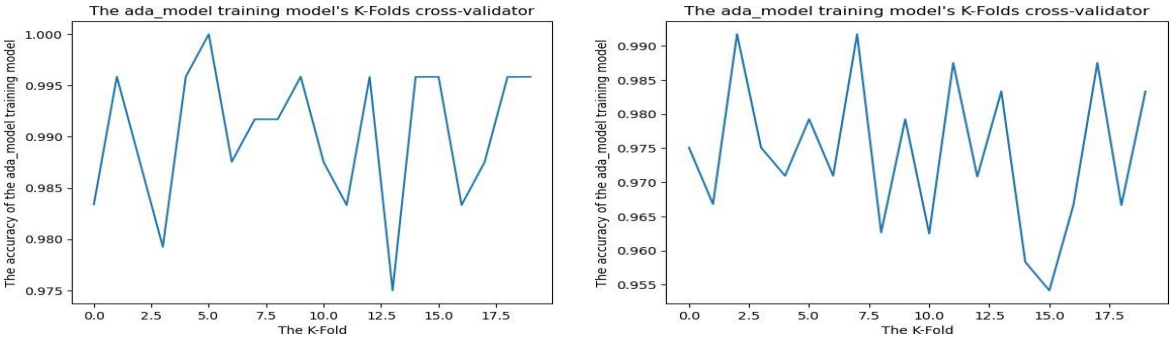
From Table 1, the training of AdaBoost model for binary classification achieved an accuracy of 99.02%. The k-fold training performance is as shown in Figure 7(a). The test accuracy of the SVC was 96.90% with a weighted F1-Score of 0.97, which shows that the model performed well in terms of precision and recall

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

across all the classes of the test set; and Macro F1-Score of 0.93, which shows that it had acceptable level of precision and recall for each of the class.

Also, from Table 2, the performance of the AdaBoost classifier in the multi-class classification task shows that it achieved 97.42% accuracy in the training and validation set shown in Figure 7(b), 96.89% in the test set, weighted F1-Score of 0.97 and macro F1Score of 0.97. This shows that model performed well in terms of precision and recall across all classes; also, it did performed very high in terms of precision and recall for each class.




(a) Binary (b) Multi-Class

**Figure 7** AdaBoost Classifier Training Performance

### 4.5 Extra-Trees Model

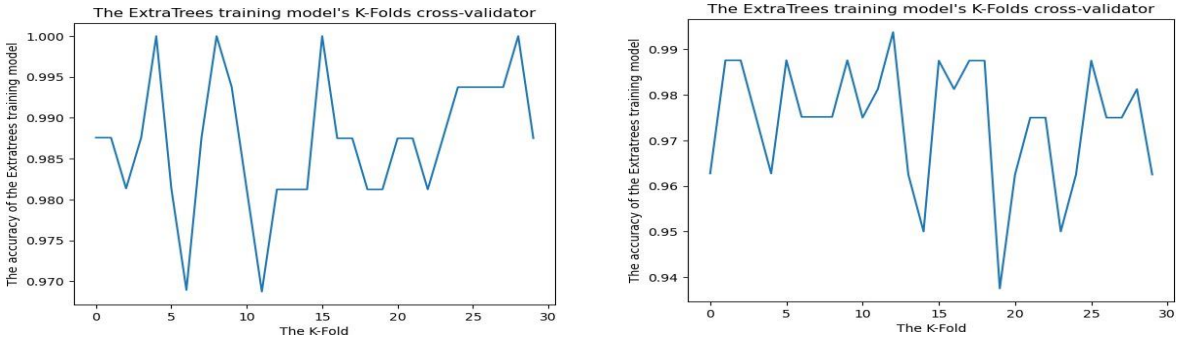
From Table 1, the training of Extra-Trees model for binary classification achieved an accuracy of 98.71%. The k-fold training performance is as shown in Figure 8(a). The test accuracy of the SVC was 96.58% with a weighted F1-Score of 0.96, which shows that the model performed well in terms of precision and recall across all the classes of the test set; and Macro F1-Score of 0.93, which shows that it had acceptable level of precision and recall for each of the class.

Also, from Table 2, the performance of the Extra-Trees classifier in the multi-class classification task shows that it achieved 97.24% accuracy in the training and validation set shown in Figure 8(b), 97.52% in the test set, weighted F1-Score of 0.98 and macro F1-Score of 0.99. This shows that model performed well in terms of precision and recall across all classes; also, it did perform extremely high in terms of precision and recall for each class.

From the presented results of the developed models' performances, the SGD Classifier performed poorly compared to other considered classifiers in the binary classification tasks. AdaBoost Classifier achieved highest training accuracy, but lower test accuracy compared to LinearSVC and SVC. Therefore, for the binary classification task, LinearSVC achieved overall best performance with 98.75% training accuracy, 97.83% test accuracy, 0.98 weighted F1-Score, and 0.96 macro F1-Score. However, LinearSVC underperformed compared to the Random Forest Classifier model in Al-Boghdady, A., El-Ramly, M., & Wassif, K. (2022), which achieved 99% in test accuracy, weighted F1-Score of 0.98 and macro F1-Score of 0.99.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 297-315
https://doi.org//10.47514/kjcs/2024.1.2.0011

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng




(a) Binary (b) Multi-Class

**Figure 8** Extra-Trees Classifier Training Performance

In the multi-class classification task, the results show that LinearSVC achieved the best accuracy in training but poor test accuracy and can be interpreted for a case of overfitting. While AdaBoost performed better than both the LinearSVC and SVC classifiers, it underperformed compared to the Extreme Trees (Extra-Trees) Classifier, which achieved 97.24% in training accuracy and 97.58% in test accuracy, 0.98 in weighted F1-Score and 0.99 in Macro F1-Score. Comparing the Extreme Trees Classification to the Random Forest Classifier in Al-Boghdady, El-Ramly, & Wassif, (2022), which achieved 96.78% in training accuracy, 98.14% in test accuracy, 0.98 in weighted F1-Score and 0.99 in macro F1Score; it can be said that Extra-Tress achieved comparable performance especially, when using weighted and macro F1-Scores. While taking into consideration the properties of the Random Forest and Extreme Trees Algorithms, Extreme Trees is the most preferred classifier for this task based on the results obtained. This is because Extra-Trees algorithms are known to have the ability to minimize chances of overfitting in a model.

## 5. Conclusion

This research aimed at predicting vulnerability in IoT OS Source codes. Machine learning algorithms, which included LinearSVC, SVC, SGD, AdaBoost, and Extra-Trees were considered in this research. Developed models were trained and tested on Al-Boghdady Binary and Al-Boghdady Multi Class datasets. Test results showed that LinearSVC classifier performed better in the binary classification task, while Extra-Trees classifier performed better in the multi-class classification task. While comparing the developed Extra-Trees classifier in this research to the chosen benchmark model, it achieves comparable performance; and may be considered a better classifier with bias to its ability to minimize overfitting problems.

With the results achieved in this research, it can be concluded that Extra-Trees classifier may be deployed an alternative vulnerability tool detection for supporting IoT systems developers towards developing secured IoT applications with reduced and possibly no vulnerabilities.

In the future, the effects of word embedding models on classification performances of these, and other algorithms may be considered to determine the most efficient combination of word embedding and machine learning algorithm for the detection of vulnerabilities in IoT systems

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

source codes. The used dataset only considered C/C++ source codes. However, IoT systems may also be implemented in a variety of languages such Java, python, Perl, etc. Therefore, a labelled dataset that includes most languages used for IoT systems development would help in the development of a more general machine learning classifier and instead of having different classifiers for different languages. This is necessary as IoT systems are now more heterogeneous in terms of underlying programming languages that are used to program them. Since the IoT system may be made of these heterogeneous devices, compromising one could lead to compromising others. Thus, security and privacy remain a major concern in the IoT domain.

## Reference

Al-Boghdady, A., El-Ramly, M., & Wassif, K. (2022). iDetect for vulnerability detection in Internet of Things operating systems using machine learning. *Scientific Reports, 12*(1), 1-12. doi:10.1038/s41598-022-12345-6

Al-Boghdady, A., Wassif, K., & El-Ramly, M. (2021). The presence, trends, and causes of security vulnerabilities in operating systems of IoT's low-end devices. *Sensors, 21*(7). doi:10.3390/s21072345

Alnaeli, S. M., Sarnowski, M., Aman, M. S., Abdelgawad, A., & Yelamarthi, K. (2016). Vulnerable C/C++ code usage in IoT software systems. In *Proceedings of the International Conference on Advances in Science, Engineering and Technology* (ICASET), pp. 30-33.

Alnaeli, S. M., Sarnowski, M., Aman, M. S., Abdelgawad, A., & Yelamarthi, K. (2017). Source code vulnerabilities in IoT software systems. *Advances in Science, Technology & Engineering Systems Journal, 2*(3), 1502-1507.

Anand, P., Singh, Y., Selwal, A., Alazab, M., Tanwar, S., & Kumar, N. (2020). IoT vulnerability assessment for sustainable computing: Threats, current solutions, and open challenges. *IEEE Access, 8*, 168825-168853. doi:10.1109/ACCESS.2020.3032980

Andra, Z. (2023). 300+ terrifying cybercrime and cybersecurity statistics (2023 edition). *Comparitech*. Retrieved from https://www.comparitech.com/blog/vpn-privacy/terrifying-cybercrime-statistics/

Balsamo, D., Magno, M., Kubara, K., Lazarescu, B., & Merrett, G. V. (2019). Energy harvesting meets IoT: Fuelling adoption of transient computing in embedded systems. In *Proceedings of the IEEE 5th World Forum on Internet of Things* (WF-IoT), pp. 413-417. doi:10.1109/WF-IoT.2019.8767351

Bier, S., Fajardo, B., Ezeadum, O., Guzman, G., Sultana, K. Z., & Anu, V. (2021). Mitigating remote code execution vulnerabilities: A study on Tomcat and Android security updates. In *Proceedings of the 2021 IEEE International IoT, Electronics and Mechatronics Conference* (IEMTRONICS), pp. 1-6. doi:10.1109/IEMTRONICS53260.2021.9394436

Elshoush, H. T., & Dinar, E. A. (2019). Using AdaBoost and stochastic gradient descent (SGD) algorithms with R and Orange software for filtering e-mail spam. In *Proceedings of the 11th Computer Science and Electronic Engineering Conference* (CEEC), pp. 41-46. doi:10.1109/CEEC47704.2019.8974702

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Gaouzi, Z., & Chougdali, K. (2022). Impact of security and privacy risks on the adoption of IoT: A state of the art. In *Proceedings of the 2022 9th International Conference on Wireless Networks and Mobile Communications* (WINCOM), pp. 1-5. doi:10.1109/WINCOM.2022.9725728

Guo, J., & Wang, X. (2023). Interpreting deep networks: By ensemble learning and visualizing the perturbation. In *Proceedings of the 6th International Conference on Electronic Information Technology and Computer Engineering* (pp. 777-781).

He, D., Yu, X., Li, T., Chan, S., & Guizani, M. (2022). Firmware vulnerabilities homology detection based on clonal selection algorithm for IoT devices. *IEEE Internet of Things Journal, 9*(17), 16438-16445. doi:10.1109/JIOT.2021.3157045

Injadat, M. N., Moubayed, A., & Shami, A. (2020). Detecting botnet attacks in IoT environments: An optimized machine learning approach. In *Proceedings of the International Conference on Microelectronics* (pp. 30-33).

Joe, O. (2020). Security concerns remain a major barrier to IoT adoption. *Computer Weekly*. Retrieved from https://www.computerweekly.com/security-concerns-iot-adoption

Kabir, F., Siddique, S., Kotwal, M. R. A., & Huda, M. N. (2015). Bangla text document categorization using stochastic gradient descent (SGD) classifier. In *2015 International Conference on Cognitive Computing and Information Processing* (CCIP) (pp. 1-4).

Liu, Q., Wang, J., Zhang, D., Yang, Y., & Wang, N. (2018). Text features extraction based on TF-IDF associating semantic. In *IEEE 4th International Conference on Computer and Communications* (ICCC) (pp. 2338-2343).

Liu, X. (2022). Pg-vulnet: Detect supply chain vulnerabilities in IoT devices using pseudo-code and graphs. In *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 205-215).

Mahari, M. (2022). Anatomy of attacks on IoT systems: Review of attacks, impacts and countermeasures. *Journal of Surveillance, Security and Safety, 3*, 150-173.

Malina, L., Srivastava, G., Dzurenda, P., Hajny, J., & Ricci, S. (2019). A privacy-enhancing framework for Internet of Things services. *Lecture Notes in Computer Science, 11928*, 77-97. doi:10.1007/978-3-030-32255-4_6

Mehta, J., Richard, G., Lugosch, L., Yu, D., & Meyer, B. H. (2023). Dt-ds: Can intrusion detection with decision tree ensembles. *ACM Transactions on Cyber-Physical Systems, 7*(1), Article 202303. doi:10.1145/3512554

Myridakis, D., Papafotikas, S., Kalovrektis, K., & Kakarountas, A. (2020). Enhancing security on IoT devices via machine learning on conditional power dissipation. *Electron, 9*(11), 1-19. doi:10.3390/electronics9111903

Oser, P., Heijden, R. W., Lu¨ders, S., & Kargl, F. (2022). Risk prediction of IoT devices based on vulnerability analysis. *ACM Transactions on Privacy and Security, 25*(2). doi:10.1145/3592581

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Priyatharsini, G. S., Babu, A. J., Kiran, M. G., Kumar, P. J. Sathish, Babu, C. Nelson Kennedy, & Ali, A. (2022). Self-secured model for cloud based IoT systems. *Measurement and Sensors, 24*.

Sharma, J., Giri, C., Granmo, O.-C., & Goodwin, M. (2019). Multi-layer intrusion detection system with extratrees feature selection, extreme learning machine ensemble, and softmax aggregation. *EURASIP Journal on Information Security, 2019*(1), Article 1-16. doi:10.1186/s13635-019-0104-7

Tayus, S. N., Alam Kakon, A. K. M. Kamrul, & Ullah, M. (2022). IoT based web controlled multiple home automation and monitoring with Raspberry Pi. In *Proceedings of the Conference on Emerging Technologies* (INCET) (pp. 3-6).

Thapaliya, B., Mursi, K. T., & Zhuang, Y. (2021). Machine learning-based vulnerability study of interpose PUFs as security primitives for IoT networks. In *2021 IEEE International Conference on Networking, Architecture, and Storage* (NAS) (pp. 1-7). doi:10.1109/NAS53273.2021.9533367

Tyagi, H., & Kumar, R. (2021). Analyzing security approaches for threats, vulnerabilities, and attacks in an IoT environment. In *Conference on Computer Performance Evaluation and Computation* (ComPE) (pp. 227-233).

Uprety, A., & Rawat, D. B. (2021). Reinforcement learning for IoT security: A comprehensive survey. *IEEE Internet of Things Journal, 8*(11), 8693-8706. doi:10.1109/JIOT.2021.3102626

Xiong, S., Sarwate, A. D., & Mandayam, N. B. (2022). Network traffic shaping for enhancing privacy in IoT systems. *IEEE/ACM Transactions on Networking, 30*(3), 1162-1177. doi:10.1109/TNET.2021.3126843

Yang, G. (2022). An overview of current solutions for privacy in the Internet of Things. *Frontiers in Artificial Intelligence, 5*. doi:10.3389/frai.2022.891491

Yıldırım, F. M., Kaya, A., O¨ ztu¨rk, S. N., & Kılın¸c, D. (2019). A real-world text classification application for an e-commerce platform. In *2019 Innovations in Intelligent Systems and Applications Conference* (ASYU) (pp. 1-5).

Zachariah B., Ndang P. Y., & Bernard E. (2016) Application of Steganography and Cryptography for Secured Data Communication – A Review. *International Journal of Engineering Research & Technology (IJERT)*, vol. 5, no. 4, pp. 186 - 190.

Zafari, A., Zurita-Milla, R., & Izquierdo-Verdiguier, E. (2019). Land cover classification using extremely randomized trees: A kernel perspective. *IEEE Geoscience and Remote Sensing Letters, 17*(10), 1702-1706. doi:10.1109/LGRS.2019.2904530

Zainuddin, N., Daud, M., Ahmad, S., Maslizan, M., & Abdullah, S. A. L. (2021). A study on privacy issues in Internet of Things (IoT). In *2021 IEEE 5th International Conference on Cryptography, Security, and Privacy* (CSP) (pp. 96-100).

Zambang, M. A. M., Jiang, H., & Wahab, L. (2021). Modeling vehicle ownership with machine learning techniques in the Greater Tamale Area, Ghana. *PLoS One, 16*(2), Article e0246044. doi:10.1371/journal.pone.0246044

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Zhang, Z. (2020). VSFBS: Vulnerability search in firmware based on string. In *Proceedings of the 2020 7th International Conference on Dependable Systems and Their Applications* (DSA) (pp. 555-563).

Zhao, B. (2022). A large-scale empirical analysis of the vulnerabilities introduced by third-party components in IoT firmware. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 442-454).