IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), Bandung, Indonesia, 2024

# Android Malware Detection Using Machine Learning

Shaikha Al Ali
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
g00084315@aus.edu

Ali Suleiman
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00083443@aus.edu

Ghina Hallal
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
g00086253@aus.edu

Sultan Alseiari
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00085157@aus.edu

Yiguang Ma
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00083281@aus.edu

Salam Dhou
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
sdhou@aus.edu

Fadi Aloul
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
faloul@aus.edu

***Abstract*— Malware, or malicious software, poses a significant threat to systems and networks. Malware attacks are becoming extremely sophisticated, and the ability to detect and prevent them is becoming more challenging. Detecting and preventing malware is crucial for several reasons, including the security of personal information, data loss and tampering, system disruptions, financial losses, and reputation damage. This paper presents a machine learning approach for Android malware detection. In this work, several machine learning algorithms were utilized, namely k-Nearest neighbor (KNN), Decision Trees (DT), Naive Bayes (NB), Support Vector Machine (SVM) and other ensemble classifiers including Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LGBM) and CatBoost. It was found that SVM using radial basis function (RBF) kernel achieved the highest performance with an accuracy of 99.5%. This work proved the feasibility of using machine learning in detecting malware and improving the security of mobile devices. The results of this work can be used to build more robust systems to protect devices and networks from malicious attacks.**

***Keywords*— *Android, malware detection, goodware, machine learning, cybersecurity.***

## I. Introduction

Malware (Malicious software) is a form of software that is designed to cause harm to a computer or electronic device. Malware can be spread through digital communications like email attachments or other forms of communication, including social engineering and infected websites [1]. In addition, malware can be downloaded by accident onto a computer and is mainly used for malicious purposes like gaining access to people's data and networks, preventing computers from working correctly, and causing crashes [2]. Malware can take many forms, including viruses, worms, Trojans, ransomware, and spyware, each with different methods of attacking and spreading [1]. The main goal of malware is to steal sensitive information, hijack systems for malicious purposes, or cause a breach of data and systems [2]. Additionally, detecting such software is highly predominant because it can cause harm to devices, compromises personal and sensitive information, and lead to financial losses or identity theft. Therefore, by detecting malware, individuals can protect their privacy, security and data integrity, and ensure their devices operate smoothly and efficiently.

To combat malware, it is essential to distinguish it from goodware. Goodware is software designed to benefit users, is safe for use, and does not cause harm or exploit computer systems. Emerging machine learning techniques were proved to be useful in significant cybersecurity applications such malicious website detection, intrusion detection [3] and botnet attack detection [4].

The objective of this work is to propose a machine learning framework for the purpose of Android malware detection. A dataset with several permission-based and API-based features is used to train the model and test its feasibility in distinguishing malware from goodware.

## II. Literature Review

Various studies proposed different approaches to Android malware detection using machine learning algorithms. Li et al. [5] introduced SIGPID, a malware detection system based on permission usage analysis, to address the growing number of Android malware. The study utilized machine learning models, including Support Vector Machine (SVM), and focused on 22 critical permissions, achieving over 90% accuracy, recall, F-measure, and detection rate. Suleiman et al. [6] similarly explored malware detection using a parallel classification approach leveraging algorithms such as Decision Tree (DT), Simple Logistic, Naive Bayes (NB), PART and RIpple-DOwn Rule learner (RIDOR). Their results showed that the PART algorithm performed best, with a detection rate of 95.8%. Another related study by Hanqing et al. [7] proposed a classification method based on abstracted API calls, using Random Fores (RF). Their method achieved high detection accuracies of 96% and 98% on the Drebin and AMD datasets, respectively.

Several other studies used RF as part of their malware detection systems. Hanqing et al. [7] used RF alongside abstracted API calls, achieving high detection results. Wei-Ling et al. [8] also used RF, focusing on dynamic behavior such as network activity. Using the Robotium program to collect network and app activity, RF was the best-performing algorithm, achieving an accuracy of 97%, a true positive rate

(TPR) of 97.1%, and a false positive rate (FPR) of 0.036. Similarly, Fauzi et al. [9] implemented RF, k-nearest neighbor (KNN), and DT for malware detection using grayscale images generated from Android APK files. RF achieved the highest accuracy of 84.14%. Lastly, Koli [10] proposed RanDroid, a system based on both static and dynamic features of Android apps, with RF achieving a classification accuracy of 97.7%.

Other studies focused on different features and classifiers. Hyo-Sik et al. [11] used a linear SVM technique for Android malware detection in IoT services, achieving a precision of 95.7% with a dataset containing 90% normal and 10% malicious applications. Ravi et al. [12] used permissions as features for detection, experimenting with multiple classifiers, including NB, DT-based method (J48), and RF. Their study, which used Google Play Store app data from 2015 and 2016, found that the multi-class classifier to be the most effective, with an average accuracy of 99.81%. N. Peiravian and X. Zhu [13] also focused on permissions and API calls as features. Using a dataset of 2,510 APK files with 1,260 malicious apps and 1,250 benign ones, they found that the Bagging classifier outperformed others, achieving 96.39% accuracy, 94.9% precision, and 94.1% recall. Like several works discussed above, the proposed solution utilizes machine learning algorithms to detect Android malware. Table I below summarizes the previous work that was discussed above.

TABLE I. SUMMARY OF PREVIOUS WORK

| Reference | Classifier | Performance |
|---|---|---|
| J. Li *et al.* [5] | SVM | Accuracy = 91.36%<br>Detection rate (for known malware) = 93.62%<br>Detection rate (for unknown malware) = 91.4% |
| Suleiman *et al.* [6] | PART | TPR = 95.8%<br>TNR = 97.7%<br>FPR = 3.3%<br>FNR=4.2%<br>Accuracy = 96.3%<br>AUC = 97.0% |
| Hanqing *et al.* [7] | RF | Accuracy (on Drebin dataset) = 96%<br>Accuracy (on AMD dataset) = 98% |
| Wei-Ling *et al.* [8] | RF | Accuracy = 97%<br>TPR = 97.1%<br>FPR = 0.036% |
| Fauzi *et al.* [9] | RF | Accuracy = 84.14% |
| Koli [10] | RanDroid | Accuracy = 97.7% |
| Hyo-Sik *et al.* [11] | SVM (linear) | Precision = 95.7% |
| Ravi *et al.* [12] | Multi-class classifier | Accuracy = 99.81% |
| N. Peiravian and X. Zhu [13] | Bagging | Accuracy = 96.39%<br>Precision = 94.9%<br>Recall = 94.1% |

## III. METHODOLOGY

### A. Dataset Description

The TUANDROMD dataset used in this work [14] comprises around 4470 samples classified into two classes. The first class is malware which can be defined as a software that harms a device or gains unauthorized access. There are around 2535 samples of malware. The second class is goodware, which does not yield to harming devices. However, it does ask for numerous permissions. There are around 1935 samples of goodware. As can be noticed, the dataset is not perfectly balanced as the ratio of malware vs. goodware is around 56:44. Data balancing techniques were used to balance the dataset as can be seen in Section III. B.

The samples of the dataset have around 240 binary features. Most of them are permission-based features. The rest of the features are API-based features. Examples of permissions-based features include access to the Wi-Fi state, all downloads, cache filesystem, and billing. The API-based features mainly involve Java API and Android API.

### B. Data Pre-processing

It is essential to preprocess the dataset before feeding it to the machine learning pipeline. Numerous preprocessing techniques have been used such as data cleaning and feature selection. Data cleaning is a data preprocessing technique that removes errors, inconsistencies, or inaccuracies in the dataset. To ensure that the samples have as minimum error as possible, records with empty feature values were removed. Moreover, class weights were applied to handle the class imbalance in the dataset. Using this approach, the minority class was given a higher weight than the majority class. Finally, dimensionality reduction using principal components analysis (PCA) was applied to ensure that the most relevant features are used to improve the model's performance.

### C. Machine Learning Models

Several machine learning algorithms were trained in this work. A description of each one of these algorithms is provided as follows.

*1) k-Nearest Neighbors (KNN):* KNN is a non-parametric machine learning model that determines distances between new input points and their nearest neighbors in the training dataset, where *k* represents the number of nearest neighbors to consider. As KNN can easily be incorporated into distance calculations. It is best suited to numerical data, especially continuous data. KNN is not recommended for high-dimensional and sparse datasets [15]. Even though KNN is simple, it can perform well on small to medium-sized datasets with low dimensionality. On the other hand, larger datasets may require more computational resources [16].

*2) Decision Tree (DT):* Using recursive splitting of data based on the values of the features, a DT constructs a tree-like model, selecting the feature that provides the most information [17]. Once the data has been split, the algorithm continues to split it until a stopping criterion has been reached, such as a maximum tree depth or a minimum information gain threshold [18]. After constructing the tree, it can predict new instances by traversing the tree from the root to the leaf nodes. As a result of their interpretability, ability to handle categorical and numerical data, and ease of use, DTs are viral models [19]. It should be noted, however,

that they are highly susceptible to overfitting if the tree is overly complex.

*3) Naive Bayes (NB):* Bayes theorem is used to compute the probability of each class given input features in the NB model [20]. In practice, it is not always true that each feature is independent of all other features, which makes the algorithm regarded as "naive." Despite its simplicity, NB is an effective algorithm and capable of handling classification and regression problems [21]. The model can handle categorical and numerical data using techniques such as binning or discretization. In addition, it is effective in dealing with large, multidimensional datasets [22]. It should be noted that NB is a probabilistic algorithm that works well with categorical data. However, it can also process numerical data with the appropriate preprocessing.

*4) Support Vector Machine (SVM):* SVMs are supervised machine learning models that operate by identifying a hyperplane that maximizes the separation of classes in input data [23]. A hyperplane can be constructed by maximizing the margin between the two nearest points from different classes, known as support vectors. In addition to being versatile, SVM can be used for both binary classification and regression tasks, and by utilizing kernel functions, it can also handle non-linearly separable data [24]. In the case of our binary dataset, which is mostly malware or goodware, SVM is an excellent approach, especially with its two types, linear and nonlinear kernels. The linear kernel is utilized for linearly separable data, while for non-linearly separable data, the nonlinear kernel is utilized. Due to its reduced sensitivity to overfitting, SVM is particularly suitable for datasets containing many features. Generally, SVM is a robust algorithm that works well with numerical and continuous data [25]. However, it can also be applied to categorical data if appropriate preprocessing techniques are applied.

*5) Extreme Gradient Boosting (XGBoost):* XGBoost consists of iteratively building an ensemble of decision trees while minimizing the loss function [26]. As a result of gradient boosting, the model increases the accuracy of weak learners by adding new trees sequentially to the model that can fit the residual errors of the previous trees. In addition, XGBoost facilitates parallel computation, resulting in a significant speedup during the training process [27]. As a result of the model's outstanding performance in Kaggle competitions, it has been widely used for both classification and regression tasks. It is most effective when dealing with numerical data, specifically continuous data. XGBoost can also handle categorical data with the appropriate preprocessing [28]. As a result, it is a preferred option for high-dimensional datasets due to its speed and accuracy.

*6) Light Gradient Boosting Machine (LGBM):* LGBM consists of an ensemble of decision trees built iteratively using gradient boosting [29]. As a result, it minimizes the loss function, utilizes gradient-based techniques, and handles imbalanced data [30]. Due to its ability to split datasets by a leaf-wise rather than level-wise modes and use histogram-based algorithms to compute split points, LGBM is faster and more accurate than other gradient boosting algorithms, such as XGBoost. While LGBM can handle categorical and numerical data, it performs optimally with numerical data, specifically continuous data [31]. In addition, it is capable of handling high-dimensional data with a large number of features.

*CatBoost:* In CatBoost, gradient-boosting techniques are utilized to build an ensemble of decision trees based on supervised machine learning [32]. The platform is designed to handle categorical data efficiently and effectively without requiring any prior encoding or preprocessing. CatBoost utilizes techniques such as gradient-based optimization, learning-to-rank, and ordered boosting to handle categorical features [33]. Additionally, it is capable of handling numerical features as well as missing values. In both classification and regression tasks, CatBoost is widely used due to its high performance, accuracy and speed. It is noteworthy that CatBoost is an efficient and robust algorithm that is particularly suited to datasets that contain categorical features [34]. Therefore, CatBoost is a popular choice for data scientists and machine learning practitioners because of its speed, accuracy, and ability to handle categorical data effectively.

## IV. Experimental Results

In this section, the results of the machine learning algorithms are presented. For testing the machine learning models, 5-fold cross validation was considered. The results presented in this section represent the average performance of the model using testing data across several folds.

For the KNN model, it achieved its highest accuracy when k was set to 1, with an accuracy of 99.2%. As we increase the number of neighbors, the overall accuracy tends to decrease slightly. For example, setting k as 50 resulted in an accuracy of 96.1%, which still indicates a good accuracy overall. The precision, recall, and f1-score of classifying malware were 99%, 97% and 98%, respectively. Whereas for classifying goodware, the precision, recall, and f1-score were 99%, 100% and 100%, respectively. Fig. 1 and Fig. 2 show the learning curve for KNN models at K=1 and K=50, respectively.

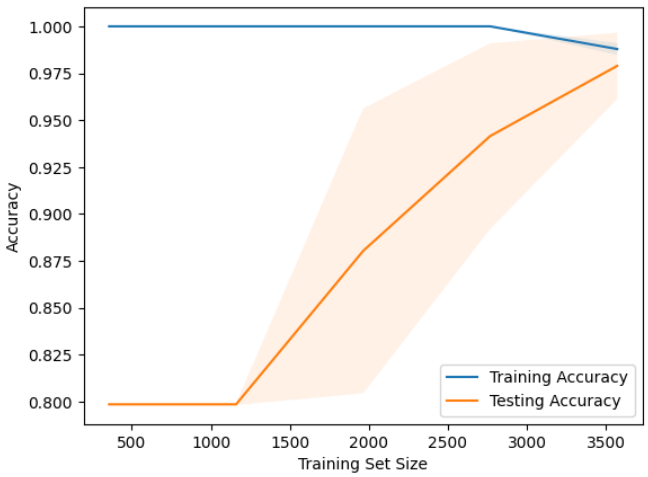


Fig. 1. Learning curve for KNN model at K = 1

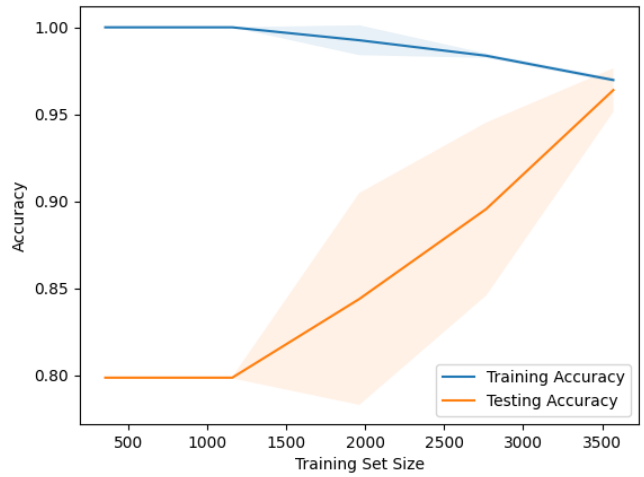


Fig. 2. Learning curve for KNN model at K = 50

As for the DT, the maximum depth was hypertuned. Initially at depth 1, the accuracy has reached 91.5%. As the maximum depth increases, the accuracy tends to increase as well, reaching maximum accuracy of 99.3% at depth 11. The precision, recall, and f1-score of classifying malware were 99%, 98% and 98%, respectively. In comparison, they were 99%, 100% and 100% when classifying goodware. Fig. 3 and Fig. 4 show the learning curve for DT models at maximum depth =1 and maximum depth = 11, respectively.

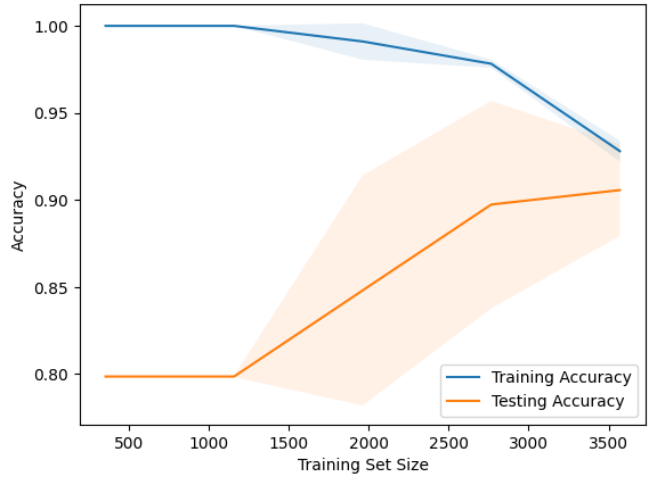


Fig. 3. Learning curve for Decision Tree model at max depth = 1

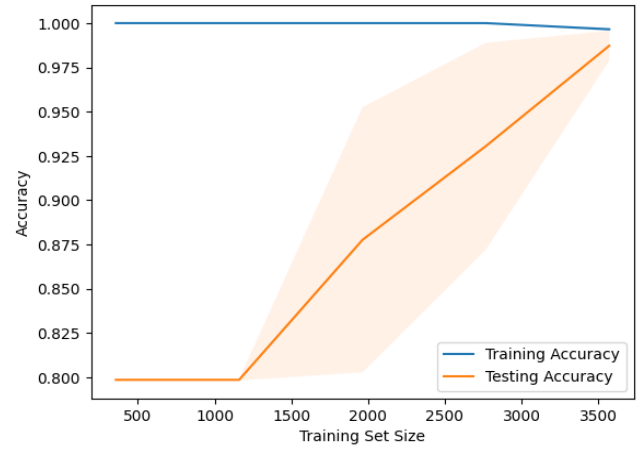


Fig. 4. Learning curve for Decision Tree model at max depth = 11

For the SVM with a linear kernel, the model achieved an accuracy of 98.1%. The precision, recall, and f1-score for classifying malware were 94%, 96% and 95%, respectively. At the same time, they were 99%, 98% and 99%, respectively for classifying goodware. For the SVM with a nonlinear kernel, Radial Basis Function (RBF), the model achieved the highest accuracy of 99.5% when C = 2 and gamma = 0.3. Changing values of both gamma and C does not lead to significant changes as it converged to 99% accuracy for most of the cases. The precision, recall and f1-score of classifying malware were 99%, 99% and 99%, respectively, whereas for classifying goodware, the precision, recall and f1-score were 100%, 100% and 100%, respectively. Fig. 5 and Fig. 6 show the learning curve for SVM models with linear and non-linear (RBF) kernels, respectively.

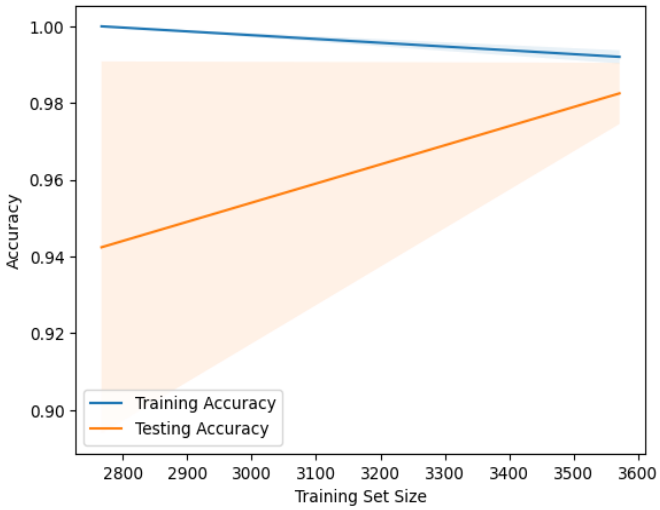


Fig. 5. Learning curve for SVM model with linear kernel

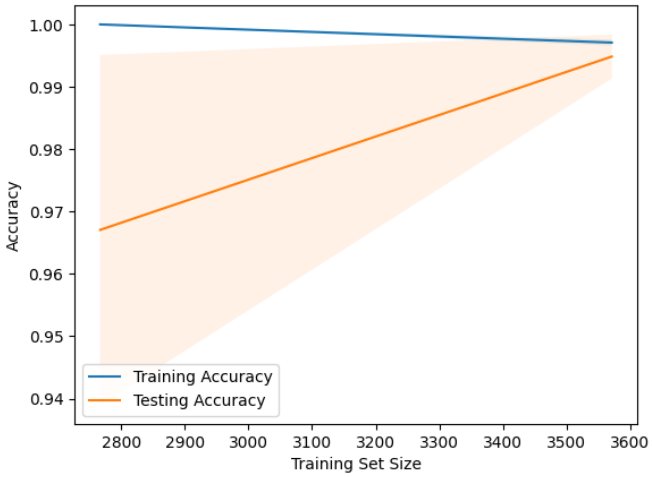


Fig. 6. Learning curve for SVM model with non-linear kernel (RBF)

When using NB on the features after preprocessing (total of 199 features), the resulting accuracy was 24%. The performance is considered low compared to the rest of the machine learning models used. To handle this, dimensionality reduction using PCA was applied. PCA transformed the dataset into a new space consisting of 100 principal component. The NB model was trained on these 100 component as input features and the testing result increased to 92%. Even with PCA being used, NB showed the lowest performance results among the rest of the models. The precision, recall, and f1-score of classifying malware were

82%, 79% and 81%, respectively whereas for classifying goodware, the precision, recall, and f1-score were 95%, 96% and 95%, respectively. Fig. 7 shows the learning curve for NB model used considering the resulting 100 principal components as the input features.

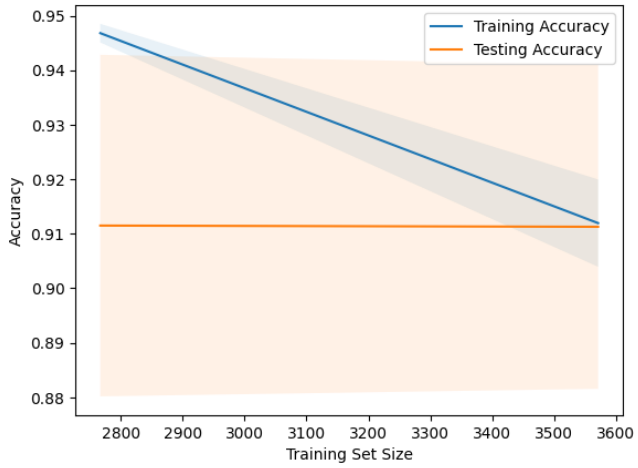


Fig. 7. Learning curve for Naïve Bayes

Moreover, it was found that all ensemble classifiers performed well in this work. For example, XGBoost achieved an accuracy of 99.3%. The precision, recall, and f1-score of classifying malware were 99%, 98%, and 98%, respectively. They were 99%, 100%, and 100%, respectively, when classifying goodware. The LGBM model achieved an accuracy of 99.2%. The precision, recall, and f1-score of classifying malware were 99%, 97%, and 98%, respectively. They were 99%, 100%, and 100%, respectively, when classifying goodware. Additionally, the CatBoost model achieved an accuracy of 99.3%. The precisions, recall, and f1-score for classifying malware were 99%, 98%, and 98%, respectively. In comparison, they were 99%, 100%, and 100% for classifying goodware.

Table II provides a summary of the results for classifying malware and goodware using the different machine learning algorithms used in this work. As can be seen from the table, SVM with the RBF kernel had the best overall performance with an average accuracy of 99.5%.

TABLE II. SUMMARY OF THE CLASSIFICATION RESULTS

| Model | Class | Precision | Recall | F1-score | Overall Accuracy |
|---|---|---|---|---|---|
| KNN | Malware | 99% | 97% | 98% | 99.2% |
| | Goodware | 99% | 100% | 100% | |
| DT | Malware | 99% | 98% | 98% | 99.3% |
| | Goodware | 99% | 100% | 100% | |
| SVM (linear) | Malware | 94% | 96% | 95% | 98.1% |
| | Goodware | 99% | 98% | 99% | |
| SVM (RBF) | Malware | **99%** | **99%** | **99%** | **99.5%** |
| | Goodware | **100%** | **100%** | **100%** | |
| NB | Malware | 82% | 79% | 81% | 92.6% |
| | Goodware | 95% | 96% | 95% | |
| XGBoost | Malware | 99% | 98% | 98% | 99.3% |
| | Goodware | 99% | 100% | 100% | |
| LGBM | Malware | 99% | 97% | 98% | 99.2% |
| | Goodware | 99% | 100% | 100% | |
| CatBoost | Malware | 99% | 98% | 98% | 99.3% |
| | Goodware | 99% | 100% | 100% | |

## V. CONCLUSION

In this work, a machine learning based malware detection method was proposed. Several machine learning algorithms were used for this task, including DT, SVM with linear and non-linear kernels, KNN, NB, and other ensemble methods. These machine learning models were trained and evaluated based on the TUANDROMD dataset. Experimental results demonstrated that SVM with RBF kernel had the best overall performance among all other classifiers with an average accuracy of 99.5%.

This paper showed that machine learning is an effective approach for detecting Android malware and improving the security of mobile devices. The results of this work can be used to build robust systems that can protect devices and networks from malicious attacks. As a future work, researchers can explore using machine learning for multiclass classification problems where the models can classify samples into different categories within goodware and malware classes. Multiclass classification can provide a higher level of protection in mobile devices as sophisticated protection measures can be designed for each of the different types of attacks.

## REFERENCES


[1] T. Alsmadi and N. Alqudah, "A survey on malware detection techniques," *2021 International Conference on Information Technology (ICIT)*, 2021.

[2] M.S. Akhtar and T. Feng, "Malware analysis and detection using machine learning algorithms," *Symmetry*, vol. 14, no. 11, p. 2304, 2022.

[3] A. Hamza, F. Hammam, M. Abouzeid, M.A. Ahmed, S. Dhou and F. Aloul, "Malicious URL and Intrusion Detection using Machine Learning," *2024 International Conference on Information Networking (ICOIN)*, Ho Chi Minh City, Vietnam, 2024, pp. 795-800, doi: 10.1109/ICOIN59985.2024.10572207.

[4] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou and F. Aloul, "Botnet Attack Detection using Machine Learning," *2020 14th International Conference on Innovations in Information Technology (IIT)*, Al Ain, United Arab Emirates, 2020, pp. 203-208, doi: 10.1109/IIT50501.2020.9299061

[5] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[6] S.Y. Yerima, S. Sezer, and I. Muttik, "Android malware detection using Parallel Machine Learning Classifiers," *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, 2014.

[7] H. Zhang, S. Luo, Y. Zhang, and L. Pan, "An efficient Android malware detection system based on method-level behavioral semantic analysis," *IEEE Access*, vol. 7, pp. 69246–69256, 2019.

[8] W.-L. Chang, H.-M. Sun, and W. Wu, "An Android Behavior-Based Malware Detection Method using Machine Learning," in *2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Aug. 2016, pp. 1–4. doi: 10.1109/ICSPCC.2016.7753624.

[9] F.M. Darus, N.A. Salleh, and A.F. Mohd Ariffin, "Android malware detection using machine learning on image patterns," *2018 Cyber Resilience Conference (CRC)*, 2018.

[10] J.D. Koli, "RanDroid: Android malware detection using random machine learning classifiers," in *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*, Mar. 2018, pp. 1–6.

[11] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear SVM-based Android malware detection for reliable IOT services," *Journal of Applied Mathematics*, vol. 2014, pp. 1–10, 2014.

[12] P.R. Varma, K.P. Raj, and K.V.S. Raju, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017.

[13] N. Peiravian and X. Zhu, "Machine learning for Android malware detection using permission and API calls," *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, 2013.

[14] *UCI Machine Learning Repository: TUANDROMD (Tezpur University Android Malware Dataset) data set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/TUANDROMD+%28+Tezpur+University+Android+Malware+Dataset%29.

[15] A.R. Guzman, "*Image steganography using Deep Learning Techniques*, *figshare*." Purdue University Graduate School, 2022. Available at: https://hammer.purdue.edu/articles/thesis/Image_Steganography_Using_Deep_Learning_Techniques/19666473 (Accessed: April 26, 2023).

[16] O. Harrison, "*Machine learning basics with the K-nearest neighbors algorithm*, *Medium*," Towards Data Science, 2019, Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (Accessed: April 29, 2023).

[17] "*Cloud Data Intelligence detection based on decision tree algorithm,*" *Machine Learning Theory and Practice*, 1(1), 2020.

[18] "*Creating a decision tree classifier,*" *Machine Learning for iOS Developers*, pp. 175–202., 2022.

[19] A. Murphy, "Decision tree (machine learning)," *Radiopaedia.org* [Preprint], 2017.

[20] D. Lowd and P. Domingos, "Naive Bayes models for probability estimation," *Proceedings of the 22nd international conference on Machine learning - ICML '05,* 2005 [Preprint].

[21] "Naive Bayes" *Bayesian Reasoning and Machine Learning*, pp. 243–255, 2012.

[22] "*Text classification and naive Bayes*" *Introduction to Information Retrieval*, pp. 234–265, 2008.

[23] A.D. Cahyani and W. Budiharto, "Modeling Intelligent Human Resources Systems (IRHS) using big data and support vector machine (SVM)," *Proceedings of the 9th International Conference on Machine Learning and Computing,* 2017 [Preprint].

[24] "Support Vector Machine (SVM)" *Encyclopedic Dictionary of Genetics, Genomics and Proteomics,* 2004 [Preprint].

[25] "Support Vector Machine" *Applications of Machine Learning and Data Analytics Models in Maritime Transportation*, pp. 79–92, 2022, Available at: https://doi.org/10.1049/pbtr038e_ch7.

[26] "21 extreme gradient boosting" *Data Science for Supply Chain Forecasting*, pp. 189–199, 2021.

[27] T.C. Nokeri "Big Data, machine learning, and Deep Learning Frameworks," *Data Science Solutions with Python*, pp. 7–14, 2021. Available at: https://doi.org/10.1007/978-1-4842-7762-1_2.

[28] Y. Zheng, "A default prediction method using XGBoost and lightgbm," *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)* [Preprint], 2022, Available at: https://doi.org/10.1109/icicml57342.2022.10009823.

[29] "Classical machine-learning paradigms for Data Mining" (2016) *Data Mining and Machine Learning in Cybersecurity*, pp. 45–78.

[30] M. Osman *et al.*, "ML-LGBM: A machine learning model based on light gradient boosting machine for the detection of version number attacks in RPL-based networks," *IEEE Access*, 9, pp. 83654–83665, 2021.

[31] A. Shukla, *et al.* "Integrating comparison of malware detection classification using LGBM and XGB machine learning algorithms," *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS),* 2022 [Preprint].

[32] J.T. Hancock and T.M. Khoshgoftaar, "CatBoost for Big Data: An interdisciplinary review," *Journal of Big Data*, 7(1), 2020.

[33] A.A. Ibrahim *et al.,* "Comparison of the CatBoost classifier with other machine learning methods," *International Journal of Advanced Computer Science and Applications*, 11(11), 2020.

[34] J. Moubarak and T. Feghali, "Comparing machine learning techniques for malware detection," *Proceedings of the 6th International Conference on Information Systems Security and Privacy,* 2020 [Preprint].