WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Dynamic and Static Malware Detection: Evaluating Machine Learning Algorithms

P. Somasekhar[1], Mr. N. L. V. Venu Gopal[2]

[1]Computer Science Student, GMR Institute of Technology, Rajam
[2]Assistant Professor, GMR Institute of Technology, Rajam

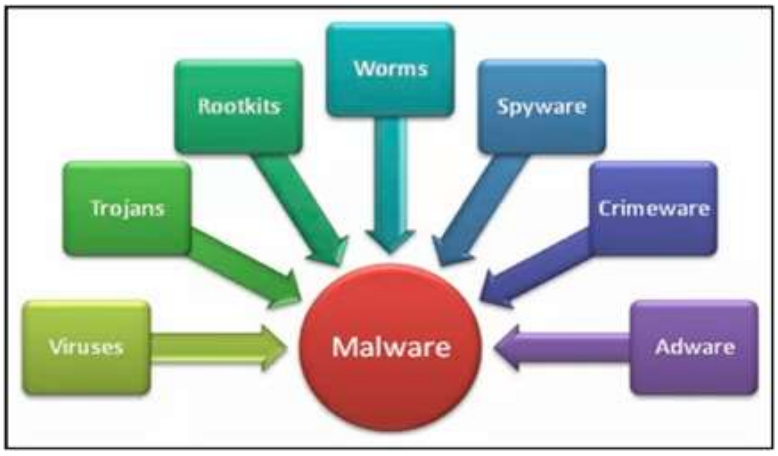
ABSTRACT :

The increasing sophistication of malware has rendered traditional detection methods, such as signature-based techniques, less effective against emerging threats. This paper investigates the application of machine learning algorithms for malware detection, focusing on their ability to analyze and classify malicious behavior dynamically. By using features extracted from behaviorbased analysis, this study evaluates various machine learning classifiers, including Random Forest, Support Vector Machines, and Neural Networks. The experimental results demonstrate that these algorithms achieve superior accuracy and efficiency in detecting malware compared to conventional methods, with significantly reduced false positive rates. The findings underscore the potential of machine learning as a powerful tool in cybersecurity, offering a more adaptive and proactive approach to identifying and mitigating malware threats. This research provides valuable insights into the implementation of machine learning for real-time malware detection and suggests future directions for improving the robustness and accuracy of these systems.

Keywords: ***malware detection , machine learning algorithms, dynamic analysis,accuracy improvement ,realtime detection***

1. INTRODUCTION

Malware detection is one of the chief issues in cybersecurity, especially with the nature of malware continually evolving into higher sophistication. Machine learning algorithms provide a dynamic solution for automatic malware detection and its classification based on behavior patterns. However, machine learning algorithms have an edge over traditional signature-based detection because polymorphic malware, new threats, etc., are often catered for in such algorithms. Accordingly, analysis of sandbox environments implies that machine learning models like Random Forest, k-Nearest Neighbors (kNN), Support Vector Machines (SVM), can perform a good job in classifying malicious software hence very important tools in boosting the efficacy of malware detection and generally enhancing the overall cybersecurity.



Traditional methods of malware detection, such as signature-based techniques, rely on predefined patterns to identify malicious activities. While effective against known threats, these approaches often struggle to detect novel and evolving malware variants. This limitation has driven the cybersecurity community to explore more sophisticated and adaptive techniques, with machine learning emerging as a promising solution.

Machine learning-based malware detection leverages algorithms that can learn patterns, behaviors, and anomalies from vast datasets to classify files as benign or malicious. Techniques such as supervised learning, unsupervised learning, and ensemble methods have demonstrated significant potential in

identifying threats with higher accuracy and speed compared to traditional methods. Moreover, the ability to generalize across unseen threats makes machine learning an indispensable tool in the fight against modern malware.

2. LITERATURE SURVEY

The objective of the article "Malware Analysis and Detection Using Machine Learning Algorithms" is to investigate the application of machine learning techniques in the analysis and detection of malware. The authors aim to identify the most effective algorithms and approaches for enhancing malware detection capabilities. Through this research, they seek to provide a comprehensive overview of machine learning methodologies that can be utilized to combat the evolving threats posed by malware.[1]

The objective of the paper titled "An in-depth review of machine learning based Android malware detection" is to critically analyze and summarize existing research on the use of machine learning techniques for detecting Android malware. The authors aim to categorize various approaches, including supervised, unsupervised, and deep learning methods, while discussing their effectiveness and limitations. This review is intended to highlight gaps in the current literature and suggest future directions for research in the field of Android malware detection.[2]

The primary goal of this article is to assess the architectural frameworks used in systems design and their implications for performance and security. It aims to provide a comprehensive overview of current architectural practices and their effectiveness in addressing the challenges posed by modern computing environments. The document seeks to highlight best practices that can lead to improved system resilience and security.[3]

The objective of this study is to investigate the various aspects of information security applications, emphasizing the importance of secure systems in the digital landscape. The paper aims to identify key vulnerabilities in current practices and propose solutions that can enhance the security posture of organizations. By examining existing literature, the study seeks to contribute to the development of more robust security frameworks.[4]

This document seeks to achieve an understanding of organizational effectiveness through the lens of data and metadata management. Its objective is to analyze how effective use of data can drive organizational success and improve decision-making processes. The paper aims to present a framework that organizations can adopt to enhance their data management practices and ultimately achieve better outcomes.[5]

This systematic review aims to consolidate existing research on Android mobile malware detection using machine learning techniques. The objective is to critically evaluate 106 selected studies, assessing their strengths and weaknesses while identifying gaps in the current knowledge base. The paper seeks to provide a comprehensive overview of methodologies used in malware detection and to propose future directions for research in this critical area of cybersecurity.[6]

In the "Journal of Computer Science Research," the objective is to present original research findings that contribute to the field of computer science. The journal aims to provide a platform for researchers to share innovative ideas, methodologies, and technological advancements. By encouraging submissions that cover a broad range of topics, the journal seeks to foster knowledge exchange and promote further research in computer science.[7]

The objective of the paper published in "Expert Systems With Applications" is to explore the role of expert systems in various applications, particularly focusing on their effectiveness in solving complex problems. The authors aim to review existing literature and methodologies to identify best practices and potential improvements in the design and implementation of expert systems. This work is intended to provide insights that can lead to better decision-making processes in diverse fields, enhancing the overall utility of expert systems.[8]

The objective of the paper titled "Learning features from enhanced function call graphs for Android malware detection" is to improve the detection of Android malware through the use of enhanced function call graphs. The authors aim to extract and analyze various features from these graphs to develop a more effective malware detection system. By leveraging advanced graph-based techniques, the study seeks to enhance the accuracy and efficiency of existing detection methods, thereby addressing the increasing threat of malware to Android devices.[9]

The primary objective of this document is to explore advancements in computer security, particularly focusing on methods and technologies that enhance the protection of computer systems against various types of cyber threats. The paper aims to provide insights into emerging security practices, evaluate their effectiveness, and contribute to the ongoing discourse in the field of cybersecurity [10]

The primary objective of this article is to introduce a novel malware detection approach that utilizes memory forensics, manifold learning, and computer vision. The study aims to enhance the detection of sophisticated malware, particularly fileless variants, by employing innovative techniques that analyze memory dumps and visual patterns, ultimately providing a robust defense against emerging cyber threats.[11]

The objective of this study is to propose a hybrid approach for detecting Android malware and classifying its families. By integrating various detection techniques, the research seeks to improve the accuracy and reliability of malware identification on Android devices, addressing the growing concerns of mobile security vulnerabilities.[12]

This paper aims to investigate new approaches in information security applications with a focus on enhancing malware detection techniques. The objective is to contribute to the existing body of knowledge by presenting novel methodologies that improve the identification and classification of malicious software, thereby reinforcing information security measures.[13]

The primary objective of this research is to explore advancements in computer security, particularly focusing on innovative methodologies for detecting and mitigating security threats. The study aims to provide insights and comprehensive analyses that contribute to the field of cybersecurity, emphasizing the need for effective protective measures against evolving cyber threats[14].

The objective of this article is to develop an automated malware detection system for mobile app stores by leveraging robust feature generation techniques. This study aims to enhance the accuracy and efficiency of malware detection mechanisms, thereby ensuring better security for mobile applications and protecting users from malicious threats.[15]

3. METHODOLOGY

The methodology for conducting a comparative analysis of machine learning techniques for malware detection is designed to ensure a comprehensive and systematic evaluation of different algorithms. This involves multiple stages, beginning with dataset selection. The selection of a reliable and representative dataset is critical for effective analysis. Publicly available malware datasets, such as those from Kaggle or VirusTotal, often serve as good starting points. These datasets typically include a mix of malware and benign samples, with features extracted through static analysis (e.g., examining file structures or opcode sequences) and dynamic analysis (e.g., monitoring system calls, process behavior, and network activities). Ensuring the dataset is diverse and up-to-date is essential to account for evolving malware variants.

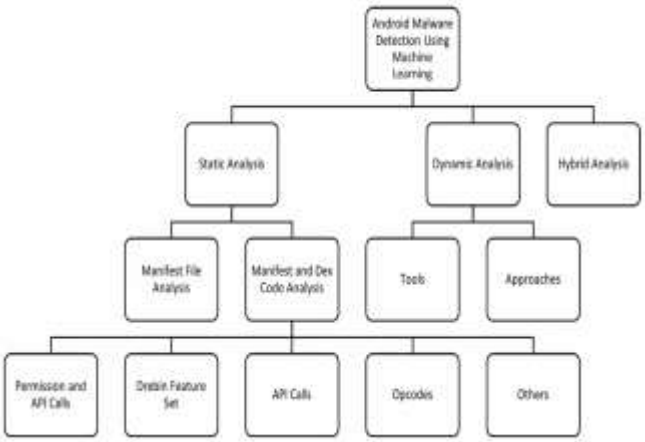
The next phase involves feature engineering, which focuses on extracting and selecting the most relevant features for classification. Features can be derived through static analysis, where the malware's code structure is analyzed without execution, or dynamic analysis, which observes the program's behavior during execution. A hybrid approach that combines these methods may also be employed. To enhance efficiency, feature selection techniques, such as correlation analysis or mutual information, are applied to reduce dimensionality while retaining the features most predictive of malware.

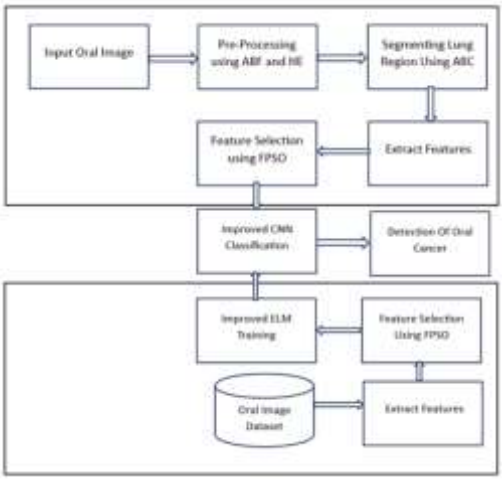
In the model selection phase, various machine learning techniques are chosen to ensure a robust comparison. Supervised learning methods, including decision trees, support vector machines (SVM), random forests, and gradient boosting algorithms like XGBoost, are commonly used due to their strong classification capabilities. Unsupervised learning approaches, such as k-means clustering, are often employed for anomaly detection, while deep learning models, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are explored for their ability to learn complex patterns. Traditional signature-based methods are also included as baseline models for benchmarking.

The model training and evaluation phase involves splitting the dataset into training, validation, and testing subsets, often in proportions such as 70%-20%-10%. Models are evaluated using standard performance metrics, including accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC). Techniques like k-fold cross-validation are employed to ensure that the results are reliable and generalizable. Additionally, attention is paid to computational efficiency and the models' ability to handle real-time malware detection scenarios.

Following training and evaluation, a comparative analysis is conducted to assess the performance of the selected models. The strengths and weaknesses of each algorithm are compared based on their detection accuracy, computational efficiency, and robustness against adversarial malware. Trade-offs, such as balancing high detection rates with low false-positive rates and resource constraints, are analyzed to identify the most practical solutions.

A. machine learning-based malware detection





DATASET:

Drebin Dataset: Contains over 5,000 Android malware samples and is frequently used for analyzing permissions and API calls.

AMD Dataset: Provides a diverse set of malware categorized by families, helpful in evaluating model robustness across malware variants.

VirusShare: A comprehensive collection of malware samples released annually, frequently utilized to benchmark machine learning models against new malware types.

PROCEDURE:

Data Collection: Data is gathered from various Android marketplaces (e.g., Google Play, APKMirror) and labeled as benign or malware using tools like VirusTotal.

Feature Extraction and Processing:

Static Analysis Features: Permission requests, API calls, manifest files, and network addresses.

Dynamic Analysis Features: System calls, file modifications, and network packets captured during execution in a sandbox environment.

Detection and Classification:

Feature Selection and Model Training: Machine learning models, such as Random Forests or SVMs, classify applications as benign or malicious using selected features.

Evaluation Metrics: Models are evaluated using metrics such as accuracy, F1-score, precision, and recall, especially crucial for imbalanced datasets with low malware occurrence.

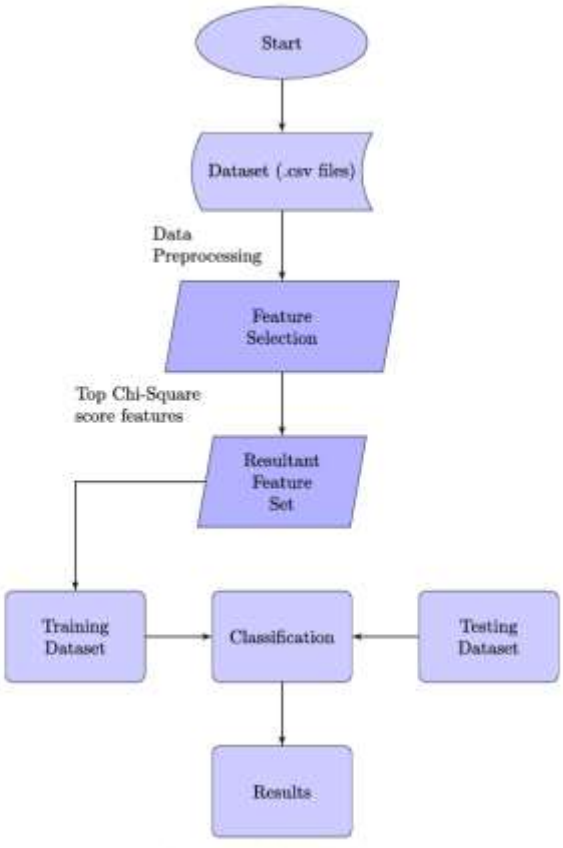


Fig. 1. Flowchart of proposed methodology

DATASET:

CICMalDroid2020 Dataset:

A comprehensive dataset containing 11,598 benign and malicious APK files across categories like adware, banking, SMS malware, and riskware. Verified with VirusTotal, it is used for both training and testing ensemble models.

PROCEDURE:

Dynamic Feature Extraction and Pre-processing:

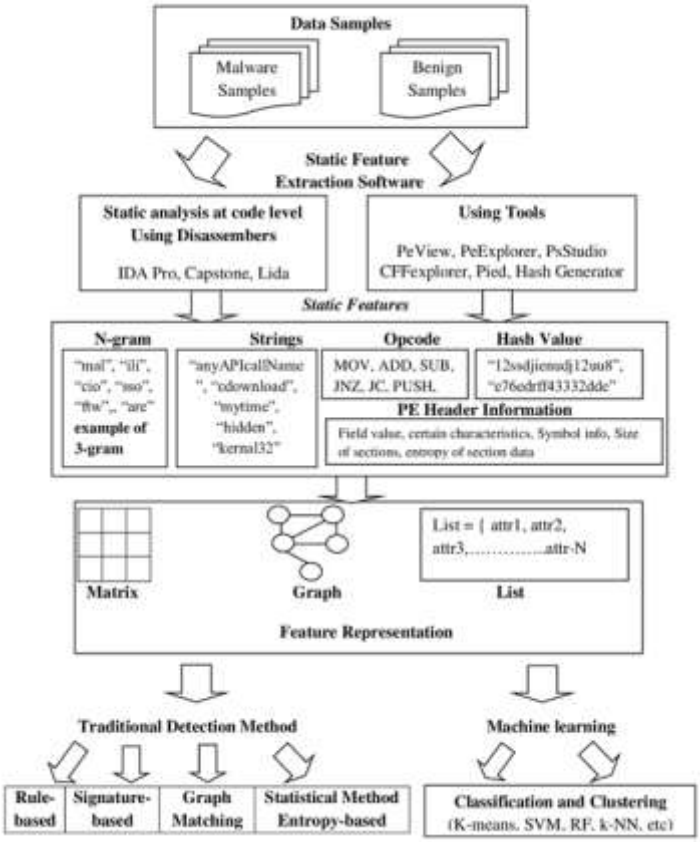
Features are extracted using CopperDroid, logging behaviors in JSON format and creating feature vectors based on system call frequency.

Chi-Square Feature Selection: The chi-square test is applied to select high-impact features, reducing dimensionality and training time.

Classification and Evaluation:

Classification: Models are trained on selected features, and ensemble techniques classify applications as benign or malicious. Stacking combines predictions from SVM, Decision Tree, and other classifiers to increase accuracy.

Evaluation Metrics: Models are evaluated using accuracy, precision, recall, and MCC to ensure robust performance across malware types.



DATASET:

VxHeaven and VirusShare: Malware repositories that provide samples across multiple malware types, including Trojans, ransomware, and rootkits.

Custom Datasets: Some researchers build datasets with malware samples labeled through antivirus programs like VirusTotal for model training and validation.

PROCEDURE:

Feature Extraction and Selection:

Static Analysis Tools: Uses tools such as Yara and SSDeep to capture PE headers, opcodes, and strings for malware signatures.

Dynamic Analysis Tools: Tools like Cuckoo Sandbox capture runtime behavior, while network traffic analyzers like Wireshark log malicious activities.

Machine Learning Models and Training:

Classifier Training: Employs algorithms such as SVM, Random Forest, and ensemble methods, trained on feature sets extracted from malware samples.

Evaluation: Models are evaluated using metrics like accuracy, F1-score, and false-positive rate

4. CASE STUDY

Case Study – 1:

Implementation of Dynamic Android Malware Detection in a Banking Application

Overview : To detect and mitigate malware targeting Android-based banking applications by leveraging a dynamic analysis model that employs both homogeneous and heterogeneous ensemble machine learning approaches. This study examines the efficiency and accuracy of detecting malware such as banking trojans, SMS malware, and adware that target sensitive financial data in mobile banking applications.

Dataset Selection: Using a public malware dataset (e.g., CICMalDroid2020), which includes samples of banking malware, adware, SMS malware, and benign applications to train the model. The dataset should be validated with VirusTotal to confirm the authenticity of benign and malicious samples.

Dynamic Analysis Framework:

The malware detection model extracts system calls and binder information from Android applications to identify specific behaviors associated with malware, including unauthorized access, data exfiltration, and sensitive data interception.

For this case study, the system call logs were collected using CopperDroid, a tool for capturing behavioral features from Android applications in a virtualized environment.

Feature Selection:

The model employs chi-square and recursive feature elimination (RFE) to retain significant features that contribute to detecting malicious behaviors while reducing noise from unnecessary features.

Classification Model:

Homogeneous ensemble approaches, like bagging with Extra Trees and Random Forest, are applied to ensure robust classification results.

For comparison, a heterogeneous stacking ensemble combines classifiers like Naive Bayes, SVM, and Decision Tree to build a comprehensive model capable of identifying diverse malware types.

1. Testing and Evaluation:

The model's performance is measured through precision, recall, accuracy, and F1-score. For instance, in the document's findings, the stacking model achieved an accuracy rate of 98.08%, showing effectiveness for high-stakes applications such as banking.

Conclusion: The stacking ensemble model provided the best classification results, effectively distinguishing between benign and malicious applications with a high recall rate. This robust dynamic analysis approach is capable of countering complex, evasive malware techniques, making it suitable for secure environments like banking applications where accuracy is critical.

Case Study -2

Enhancing Security in Mobile Financial Applications with Machine Learning-Based Malware Detection

Overview: To implement an advanced Android malware detection system for a financial app that can effectively identify both known and zero-day malware. The system is designed using machine learning (ML) models to capture patterns and behaviors that distinguish malicious apps from benign ones.

Data Collection:

A labeled dataset comprising benign and malicious Android apps was obtained from sources like the Google Play Store for benign samples and VirusShare or Drebin for malware samples. These datasets included permissions, API calls, and system calls to provide a broad basis for feature extraction.

Feature Engineering:

The study focused on extracting static features (e.g., permissions, intents) and dynamic features (e.g., API call sequences) from apps to understand both the declared permissions and runtime behavior.

Permissions analysis revealed patterns specific to financial malware, like accessing SMS or using GPS, which are often exploited for phishing attacks and surveillance.

Implementation: The model's performance was assessed using metrics such as accuracy, F1-score, recall, and precision, with results indicating an accuracy above 95% for known malware and 89% for new, zero-day samples.

Cross-validation techniques, such as k-fold, were used to test model consistency and guard against overfitting.

Conclusion: Incorporating machine learning for Android malware detection enables organizations to stay ahead of sophisticated, evolving threats. By combining static and dynamic analysis, the framework effectively identifies and mitigates zero-day vulnerabilities and evasive malware. The case study highlights the value of ensemble machine learning models in achieving high detection rates with minimal false positives, ensuring security without compromising user experience.

Case Study -3

Financial services provider

Overview : A major financial services provider needs to secure its network and devices from malware that targets sensitive financial information. Given the limitations of traditional antivirus software against complex and obfuscated malware, the company seeks to implement a machine learning-based malware detection framework combining static and dynamic analysis techniques.

Solution : This ensemble approach yields a 97% detection accuracy, demonstrating the effectiveness of a combined static-dynamic analysis for malware detection in financial services. The deployment substantially improves security posture, reducing the likelihood of data breaches from malware attacks.

Implementation :

Dataset Collection: Malware samples are obtained from reputable repositories (e.g., VirusTotal, Drebin) and validated to include various classes of malware such as banking trojans, ransomware, and spyware. Clean and verified benign samples are also included to balance the dataset.

Feature Extraction and Selection: Static features like API calls and file metadata are extracted without executing the files. Dynamic analysis captures runtime behaviors, including system calls, registry modifications, and network activities. Using CopperDroid and Cuckoo sandbox, the system captures behavioral signatures that static analysis alone might miss. Recursive feature elimination (RFE) is applied to retain only discriminative features.

Machine Learning Models: The model uses a hybrid ensemble approach with Random Forest for static features and Support Vector Machine (SVM) for dynamic analysis. By blending ensemble classifiers, the system effectively distinguishes between benign and malicious samples.

Conclusion: The case study demonstrates that machine learning-based malware detection combining static and dynamic analysis techniques offers robust security in environments with high risks, such as financial applications. By employing a hybrid, ensemble model, the framework not only detects known malware but also adapts to new, obfuscated threats. This approach allows for better accuracy, scalability, and adaptability than traditional malware detection methods, showcasing the potential of machine learning in managing evolving cybersecurity threats.

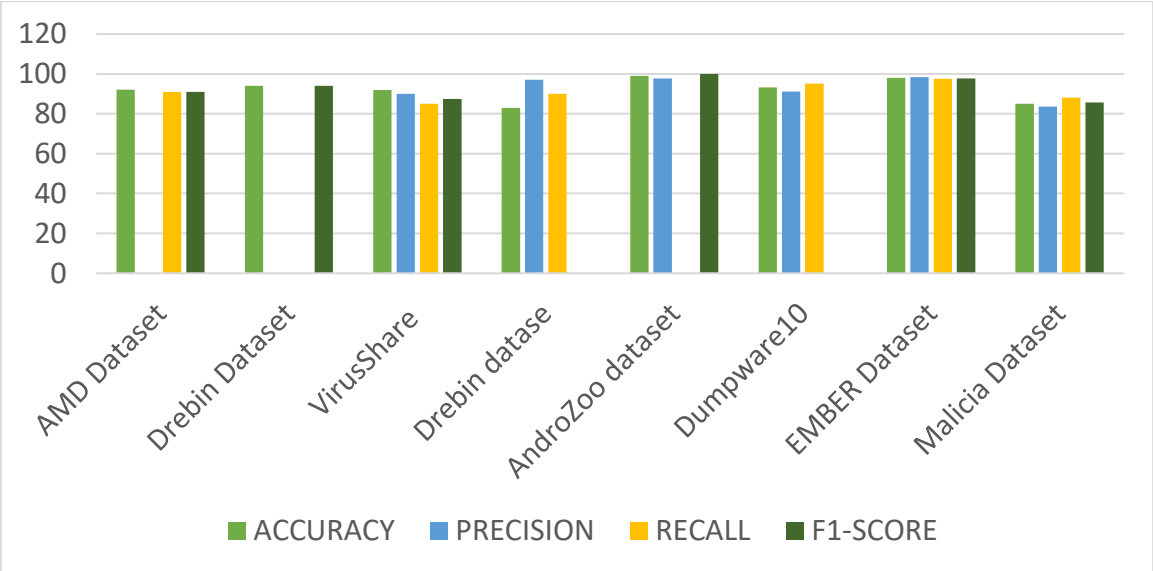
5. RESULTS & DISCUSSIONS

In the field of malware detection, machine learning techniques offer a powerful alternative to traditional signature-based methods, which are limited by their dependence on known malware patterns. By using models like Random Forest, Support Vector Machines (SVM), Neural Networks, and Gradient Boosting, researchers can classify malicious software based on patterns derived from both static and dynamic features of malware. Each model has its strengths and weaknesses; for instance, Random Forest is efficient for structured data and interpretable, while Neural Networks handle complex, high-dimensional data but may require extensive tuning and computational resources. Studies show that ensemble models like Gradient Boosting and Random Forest often outperform simpler classifiers by leveraging diverse decision boundaries, resulting in higher accuracy, precision, recall, and F1-scores. A comparative analysis of these techniques provides insights into their adaptability, scalability, and effectiveness in evolving cyber threat landscapes, highlighting the trade-offs between interpretability, computational cost, and predictive accuracy essential for real-world deployment.

values in all categories. Such performance clearly explains how adaptive AI can be in dealing with variability in imaging conditions, which is one major problem posed by resource-limited settings.

While DenseNet169 was confident on both accuracy of 96.5 % and fair metrics on oral cavity datasets, the HRNet-W18 was also powerful in delivering an F1-score of 83.6 %. The simpler methods like BPANN was found performing poorly in color lesion datasets, but showed potent architecture. Also, LightGBM showed a nearly perfect performance on online sources with an F1-score of 98.38%.

Availability of smartphone data authenticated the feasibility of real-time diagnostics. Models including HRNet-W18, with this domain specificity, processed the variability in image quality and camera angles through preprocessing techniques, resampling, and augmentation. This makes AI-driven mobile solutions a game changer for the early detection of oral cancer in resource-deprived regions, where accessibility would compromise the diagnostic accuracy achieved here.



DATASET	MODEL	ACCURACY	PRECISION	RECALL	F1-SCORE
AMD Dataset	Random Forest	92.01%	-	95.9%	91%
	Decision Tree	99%	-	99.07%	-
Drebin Dataset	Support Vector Machine	94%	-	92.04%	99.02%
	K-NN	82.95%	97%	90%	-
VirusShare	Decision Tree	87%	85%	80%	82.5%
	SVM	92%	90%	85%	87.5%
Drebin datase	K-Nearest Neighbors	82.95%	97%	90%	-
	K-NN	91.01%	92.02%	-	-
AndroZoo dataset	XGBoost	99%	97.70%	-	99.9%
	svm	96.6%	-	88.01%	-
Dumpware10	Random Forest	93.14%	91.14%	95.14%	-
	SVM	93%	89%	87%	88%
EMBER Dataset	Logistic Regression	98.00%	98.30%	97.50%	97.90%
	Random Forest	98.50%	98.70%	98.10%	98.40%
Malicia Dataset	Decision Trees	85.00%	83.50%	88.00%	85.70%
	KNN	87.00%	85.00%	89.50%	87.20%

6. CONCLUSION

In conclusion, the comparative analysis of machine learning techniques for malware detection highlights the critical role of model selection in achieving high accuracy, precision, recall, and F1-scores. Ensemble methods such as Random Forest and Gradient Boosting generally demonstrate superior performance due to their robustness and ability to handle complex patterns in malware behavior. While Neural Networks offer promising results for large and intricate datasets, they come with higher computational costs and require careful tuning to avoid overfitting. Simpler models like SVM and Decision Trees provide interpretability and efficiency, making them suitable for smaller or less

complex datasets. Ultimately, the choice of machine learning technique depends on the specific requirements of the detection environment, such as computational resources, interpretability needs, and the ability to generalize to new malware types, underscoring the importance of tailored approaches for effective malware detection in diverse cybersecurity contexts.

REFERENCES

1. Akhtar, M. S., & Feng, T. (2022). Malware analysis and detection using machine learning algorithms. *Symmetry*, *14*(11), 2304.

2. Muzaffar, A., Hassen, H. R., Lones, M. A., & Zantout, H. (2022). An in-depth review of machine learning based Android malware detection. *Computers & Security*, *121*, 102833.

3. Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, *112*, 101861.

4. Syrris, V., & Geneiatakis, D. (2021). On machine learning effectiveness for malware detection in Android OS using static analysis data. *Journal of Information Security and Applications*, *59*, 102794.

5. Haque, M. A., Ahmad, S., Sonal, D., Abdeljaber, H. A., Mishra, B. K., Eljialy, A. E. M., ... & Nazeer, J. (2023). Achieving organizational effectiveness through machine learning based approaches for malware analysis and detection. *Data and Metadata*, *2*, 139-139.

6. Senanayake, J., Kalutarage, H., & Al-Kadri, M. O. (2021). Android mobile malware detection using machine learning :Asystematic review. Electronics, 10(13), 1606.

7. Xiong, S., & Zhang, H. (2024). A Multi-model Fusion Strategy for Android Malware Detection Based on Machine Learning Algorithms. *Journal of Computer Science Research*, *6*(2), 1-11.

8. Kabakus, A. T. (2022). DroidMalwareDetector: A novel Android malware detection framework based on convolutional neural network. *Expert Systems with Applications*, *206*, 117833.

9. Cai, M., Jiang, Y., Gao, C., Li, H., & Yuan, W. (2021). Learning features from enhanced function call graphs for Android malware detection. *Neurocomputing*, *423*, 301-307**.**

10. Bhat, P., Behal, S., & Dutta, K. (2023). A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. *Computers & Security*, *130*, 103277.

11. Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, *103*, 102166.

12.Dhalaria, M., & Gandotra, E. (2020). A hybrid approach for android malware detection and family classification.

13.Surendran, R., Thomas, T., & Emmanuel, S. (2020). A TAN based hybrid model for android malware detection. *Journal of Information Security and Applications*, *54*, 102483.

14.Almashhadani, A. O., Kaiiali, M., Carlin, D., & Sezer, S. (2020). MaldomDetector: A system for detecting algorithmically generated domain names with machine learning. *Computers & Security*, *93*, 101787.

15.Alazab, M. (2020). Automated malware detection in mobile app stores based on robust feature generation. *Electronics*, *9*(3), 435.