IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 9, September 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.625

© 2024 IJIRCCE | Volume 12, Issue 9, September 2024| DOI: 10.15680/IJIRCCE.2024.1209017

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Comparative Analysis of Machine Learning Algorithms for Malware Detection using Dynamic API Call Sequences

Sushmitha B, Lakshmi H, Manikantan R

PG Student, Department of MCA, Surana College (Autonomous), Kengeri, Bengaluru, India

PG Student, Department of MCA, Surana College (Autonomous), Kengeri, Bengaluru, India

Assistant Professor, Department of MCA, Surana College (Autonomous), Kengeri, Bengaluru, India

ABSTRACT: The rapid evolution of malware poses a significant threat to cybersecurity, necessitating advanced detection techniques that can effectively identify and mitigate these threats. Traditional signature-based methods often fall short in detecting new and sophisticated malware variants. This study focuses on identifying the most effective machine learning (ML) algorithm for detecting malware using dynamic API call sequences. Dynamic API call sequences involve monitoring the real-time behavior of software by observing the series of system calls made during execution. This approach provides a rich source of information about the behavior of software, which can be leveraged to distinguish between benign and malicious activities. The study utilizes a comprehensive dataset that includes API call sequences from various software samples. To address class imbalance within the dataset, the Synthetic Minority Over Sampling Technique (SMOTE) is applied, ensuring a balanced representation of benign and malicious samples. Four prominent ML algorithms—Naive Bayes, Random Forest, K-Nearest Neighbors (KNN), and Decision Tree—are evaluated. Each model is trained on the resampled dataset and tested to assess its accuracy, precision, recall, and F1-score.

The results indicate that the Random Forest algorithm outperforms the others in terms of accuracy and overall robustness, making it the most suitable choice for detecting malware based on dynamic API call sequences. The findings are supported by detailed performance visualizations, including ROC curves, precision-recall curves, and confusion matrices. This research contributes to the field of cybersecurity by providing insights into the application of ML algorithms for malware detection and highlighting the importance of dynamic analysis in understanding malware behavior. By identifying the most effective ML algorithm, the study aids cybersecurity professionals, organizations, developers, and end-users in enhancing their defences against malware threats. Future research directions include exploring additional algorithms, larger datasets, and real-world applications to further validate and improve the proposed detection methods.

KEYWORDS: Malware Detection; Machine Learning Algorithms; Dynamic API Call Sequences; Synthetic Minority Over-sampling Technique (SMOTE); Performance Evaluation

I. INTRODUCTION

A. Importance

The increasing complexity and sophistication of malware pose a significant threat to the security of computer systems and networks worldwide. Traditional signature-based detection methods, which rely on known patterns of malicious code, often struggle to keep up with the rapid evolution of malware. This inadequacy necessitates the development and deployment of advanced detection techniques that can adapt to and identify novel threats effectively. One of the most promising approaches to enhancing malware detection involves analysing dynamic API call sequences. APIs (Application Programming Interfaces) allow software applications to interact with the operating system and other applications. By monitoring these interactions, we can gain valuable insights into the behaviour of software, distinguishing between benign and malicious activities. Dynamic analysis of API calls can reveal patterns and anomalies that static analysis might miss, making it a critical tool in the fight against malware.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. **Introduction to Dynamic API Call Sequences**

Dynamic API call sequences represent the ordered interactions between software applications and their APIs during runtime. Unlike static analysis, which examines code or binaries without execution, dynamic analysis monitors and records the actual behaviour of a program as it runs. This method tracks specific API calls, parameters, and their sequence, offering a detailed view of the software's execution flow and interactions with external systems, libraries, or services. Dynamic analysis is preferred over static analysis because it captures real-time behaviour and interactions of malware, revealing how it operates within a system. While static analysis may miss obfuscated or polymorphic threats by examining code without execution, dynamic analysis provides insights into runtime characteristics such as API call sequences and network activity. This approach is effective for detecting sophisticated malware that disguises its malicious actions or alters its behaviour to evade detection, allowing researchers to uncover hidden threats and understand the true nature of malware.

C. **Issue of Malware in API call sequences**

The manipulation of API calls by malware is a particularly serious issue. Cybercriminals exploit these calls to perform a wide range of malicious activities, such as unauthorized data access, privilege escalation, and system manipulation. The ability of malware to use legitimate API functions to disguise its actions makes it difficult to detect and mitigate using traditional methods. Recent studies indicate that a substantial proportion of cyber-attacks involve malware that leverages API calls. For instance, [1] reported that over 90% of recent cyber-attacks included some form of malware, with a significant portion exploiting API call sequences to achieve their objectives. [2] highlighted that these API-based attacks are not only increasing in frequency but also in sophistication, making them a top concern for cybersecurity professionals. The consequences of failing to effectively detect and respond to malware in API call sequences can be severe, leading to data breaches, financial losses, operational disruptions, and compromised system integrity.

D. **Beneficiaries**

- **Cybersecurity Professionals:** Enhanced detection tools that accurately identify malicious API call sequences provide cybersecurity professionals with more reliable defences against malware. These tools enable them to better protect systems and data from sophisticated attacks, reducing the likelihood of breaches and minimizing the impact of any that do occur. By leveraging advanced machine learning algorithms, cybersecurity teams can stay ahead of evolving threats, ensuring that their defence mechanisms are robust and adaptive.
- **Organizations:** Organizations, particularly those dealing with sensitive data such as financial institutions, healthcare providers, and government agencies, stand to gain significantly from improved malware detection capabilities. Effective detection and mitigation of malware ensure operational continuity, protect intellectual property, and safeguard customer information. This not only helps in avoiding financial losses but also in maintaining trust and reputation in the eyes of clients and stakeholders.
- **Developers:** Understanding how malware interacts with APIs can empower developers to build more se cure applications and systems. By gaining insights into common attack vectors and the behaviours exhibited by malicious software, developers can design software that is more resilient to exploitation. This includes implementing better security practices, hardening code against known vulnerabilities, and staying informed about the latest threats and defensive techniques.
- **End Users:** Improved malware detection mechanisms ultimately benefit end users by providing safer computing environments. Enhanced security measures protect per sonal data and privacy, reduce the risk of identity theft, and ensure that users can interact with digital systems without fear of malicious interference. This fosters greater confidence in digital technologies and encourages the adoption of online services, contributing to the overall growth and stability of the digital economy.

II. RESEARCH FLOW

A. **Data Collection**

- **Data Description:** The dataset utilized in this study dynamic_api_call_sequence_per_malware _100_0 _306.csv, which comprises API call sequences from various software samples. The dataset used in this study comprises a total of 43,876 samples and 100 feature columns related to API calls and a target variable labelled malware, indicating whether the software is classified as malicious or benign. The dataset used in this study comprises a total of 43,876 samples.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Data Preparation

- **Data Loading and Initial Inspection:** The dataset dynamic_api_call_sequence_per_malware _100_0 _306.csv is loaded using pandas. The initial step involves inspecting the dataset to understand its structure and ensure that the data is correctly formatted. The dataset was obtained from Kaggle [3]
- **Data Cleaning**: The dataset was cleaned to remove any duplicate entries and handle missing values, which helps in improving the model's accuracy and preventing errors. Specific techniques for data cleaning included imputing missing values with mean or median values, depending on the nature of the feature, and removing or correcting any anomalous data points
- **Feature and Target Variable Extraction**: Features are selected from columns 1 to 100, and the target variable, malware, is used to indicate whether a sample is malicious. This separation is crucial for training and evaluating the models.
- **Data Splitting:** The dataset is divided into training and testing subsets using the train_test_split function, with 20% of the data allocated for testing. This split is essential for assessing the models' performance on unseen data.
- **Class Imbalance Handling:** SMOTE (Synthetic Minority Over-sampling Technique) is applied to the training data to address class imbalance. SMOTE generates synthetic samples for the minority class to ensure a balanced representation, which is crucial for training robust models.

C. Methodology

Machine Learning Algorithms Four algorithms are evaluated for their performance in detecting malware.

- **Naive Bayes (GaussianNB):** The Gaussian Naive Bayes classifier is trained on the resampled training data, and its performance is evaluated using the test set. Accuracy and classification reports are generated to assess the model's ability to distinguish between malicious and benign samples.
- **Random Forest (RandomForestClassifier):** The Random Forest classifier is trained on the resampled data, and its accuracy and classification report are computed. The ensemble approach of Random Forest helps capture complex patterns in the data.
- **K-Nearest Neighbors (KNN):** KNN models with varying numbers of neighbors (k values of 3, 6, 9, 12) are trained and evaluated. The performance of each model is assessed using accuracy scores and classification reports, helping to identify the optimal number of neighbors for KNN
- **Decision Tree (DecisionTreeClassifier):** The Decision Tree classifier is trained on the resampled data and subsequently evaluated to assess its performance. This evaluation determines the model's effectiveness in accurately classifying malware while managing false positives and false negatives.

D. Visualization

- **ROC Curves**: Receiver Operating Characteristic (ROC) curves are plotted to illustrate the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for each model, highlighting each model's performance in distinguishing between malicious and benign samples.
- **Precision-Recall Curves**: Precision-Recall curves are plotted to illustrate the relationship between precision and recall for each model. These curves show how precision and recall change across different thresholds, demonstrating each model's ability to balance these metrics.
- **Confusion Matrices:** Confusion matrices are displayed for each model to visualize the distribution of true positives, true negatives, false positives, and false negatives. Each matrix includes annotations showing the count of predictions, providing insights into the errors and correct classifications made by the models.

Components of a Confusion Matrix:

For a binary classification problem, the confusion matrix is a 2x2 table with the following components:

- True Positives (TP): The number of instances correctly predicted as the positive class.
- True Negatives (TN): The number of instances correctly predicted as the negative class.
- False Positives (FP): The number of instances incorrectly predicted as the positive class (Type I error).
- False Negatives (FN): The number of instances incorrectly predicted as the negative class (Type II error).

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Performance Metrics Comparison:

- **Accuracy:** A bar chart is used to compare the accuracy scores of Naive Bayes, Random Forest, K-Nearest Neighbors (KNN), and Decision Tree models. This visualization clearly shows the overall correctness of each model's predictions, providing an effective comparison of their accuracy.
- **Precision:** A bar chart is used to compare the precision scores of the models, displaying the precision values for each, which allows for a visual comparison of how accurately each model identifies positive cases as true positives.

III. VISUALIZATION OF RESULTS

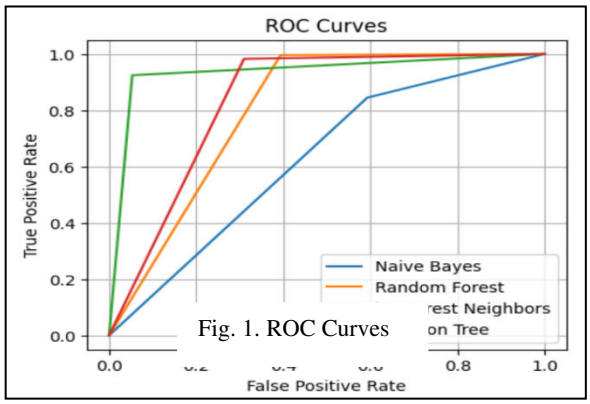
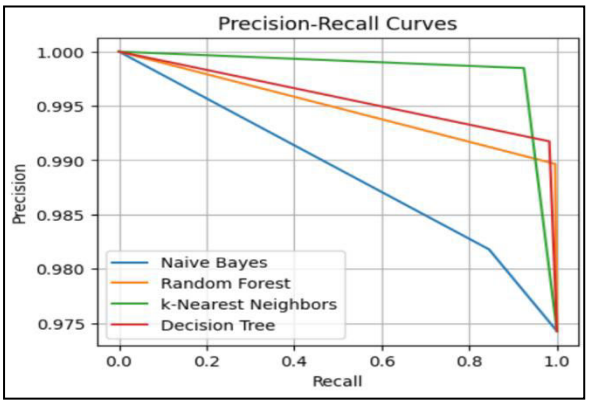
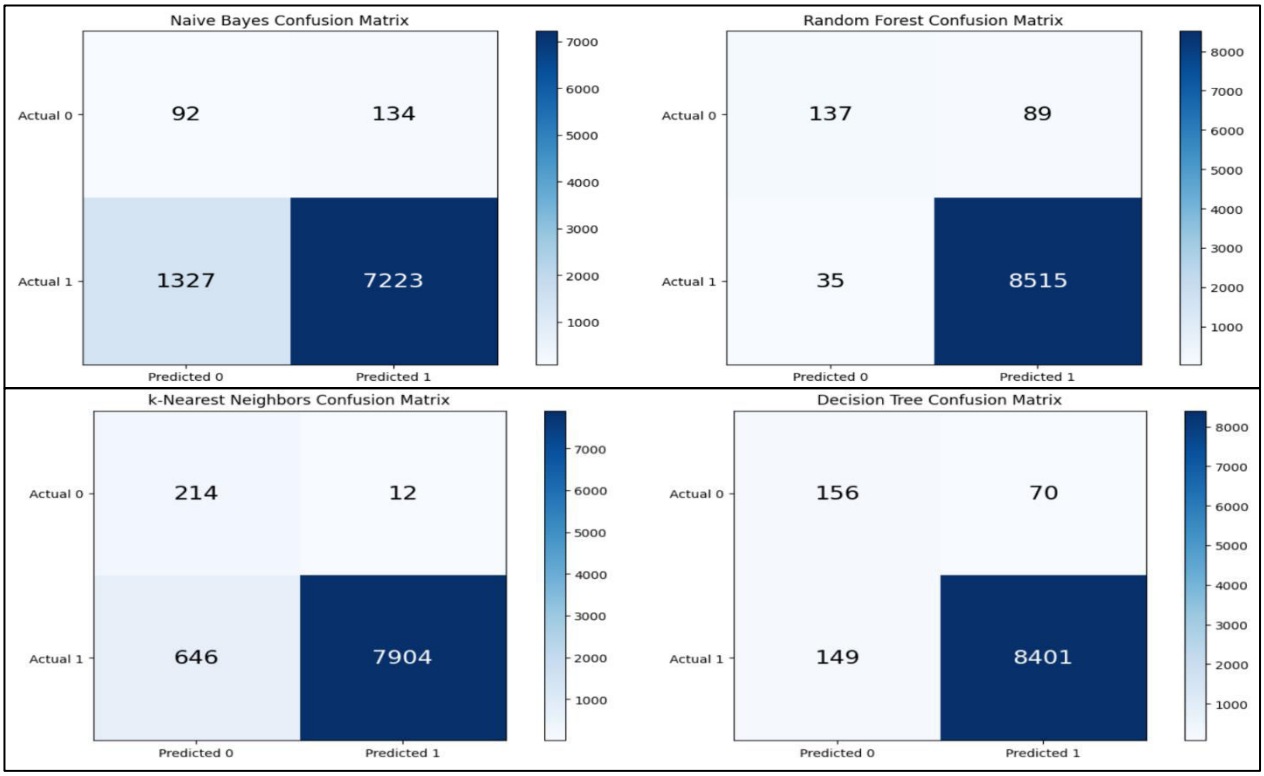


Fig. 1. ROC Curves

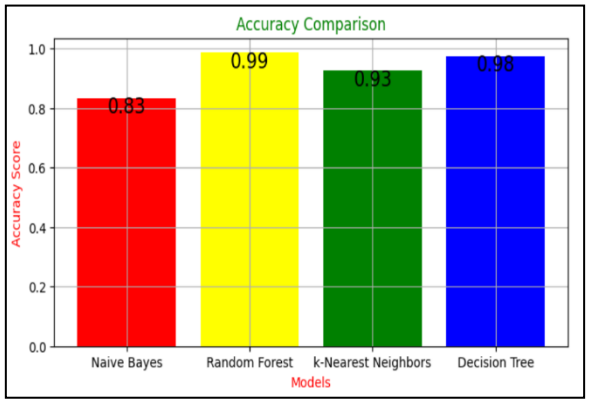




www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



A. ROC Curves:

This graph shows the performance of four different machine learning models (Naive Bayes, Random Forest, K-Nearest Neighbors, and Decision Tree) in a binary classification task. A good classifier will have a high TPR and a low FPR, which means it will correctly identify most of the positive samples while making very few mistakes. The closer a model's curve is to the top left corner of the graph, the better its performance. In this case, the Decision Tree model performs best, followed by Random Forest, K-Nearest Neighbors, and Naive Bayes. The Decision Tree has the highest TPR at a relatively low FPR, indicating it has a high ability to detect positive samples with fewer false positives.

B. Precision-Recall Curves:

The graph shows that the Decision Tree model has the highest precision and recall, followed by the Random Forest and K Nearest Neighbors models. The Naive Bayes model has the lowest performance. Overall, the Decision Tree model appears to be the best performer among the four models. It can correctly identify malware with high accuracy while also minimizing false positives.

C. Confusion Matrices:

The graphs show the confusion matrices for four different machine learning models: k-Nearest Neighbors, Decision Tree, Naive Bayes, and Random Forest. These matrices evaluate the performance of the models in classifying data into two categories (0 and 1). The numbers in the cells represent the number of instances that were correctly or incorrectly classified. For instance, the top-left cell of the kNearest Neighbors confusion matrix shows 214 instances that were correctly classified as 0, while the bottom-right cell shows 7904 instances correctly classified as 1. Overall, Random Forest seems to perform best with 8515 correctly classified as 1 and 35 incorrectly classified as 0. The other models exhibit higher misclassifications. While the graphs illustrate the performance of each model, further analysis with metrics like accuracy, precision, and recall are needed for a comprehensive understanding of their effectiveness.

D. Accuracy comparison:

Random Forest and Decision Tree models have achieved the highest accuracy scores of 0.99 and 0.98 respectively, indicating their superior performance in classifying malware. K-Nearest Neighbors follows closely with an accuracy score of 0.93. On the other hand, Naive Bayes shows a significantly lower accuracy score of 0.83. This suggests that Random Forest and Decision Tree models are more effective in distinguishing between malicious and benign software compared to the other two models. K-Nearest Neighbors also performs well, while Naive Bayes struggles with the task.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IV. RESULTS AND ANALYSIS

A. Model Performance Metrics:

For each machine learning algorithm evaluated, the following performance metrics were obtained:

Table 1 Performance metrics for different machine learning models

ML Models	Accuracy	Precision	Recall	F1-Score
Naive Bayes (GaussianNB)	83%	97%	84%	91%
RandomForest (RandomForest-Classifier)	99%	98%	99%	99%
K-Nearest Neighbors (KNN)	93%	99%	92%	96%
Decision Tree (DecisionTreeClassifier)	98%	98%	98%	91%

B. Analysis:

- **Naive Bayes (GaussianNB):** This model shows high precision (97%) but relatively lower recall (84%). This indicates that while the model is very accurate when it predicts malware, it misses a significant proportion of actual malware instances (false negatives). The balance between precision and recall reflects in a decent F1-Score (91%), highlighting a trade-off between identifying true positives and avoiding false positives.

- **Random Forest (RandomForestClassifier):** With the highest accuracy (99%) and F1-Score (99%), the Random Forest model performs exceptionally well across all metrics. It achieves both high precision (98%) and recall (99%), suggesting that it has a very low rate of false positives and false negatives. This indicates excel lent model reliability and robustness in distinguishing between benign and malicious samples.

- **K-Nearest Neighbors (KNN):** KNN shows a high precision (99%) and good recall (92%), resulting in a high F1-Score (96%). The high precision means that most of the predicted malware instances are indeed malware, but the slightly lower recall indicates that some actual malware instances might be missed. This balance reflects a well-performing model with a strong ability to correctly identify malware.

- **Decision Tree (DecisionTreeClassifier):** This model has a very high accuracy (98%) and precision (98%), with a slightly lower recall (98%) compared to Random Forest. The F1-Score (91%) is consistent with its precision and recall, showing a good trade-off. The decision tree might still have some false positives or negatives, but overall, it maintains a strong performance in classifying malware.

C. Key Findings

Best Performing Model: Based on the evaluation metrics and visualizations, the Random Forest classifier demonstrated the best overall performance. It achieved the highest scores across all metrics, including accuracy (99%), precision (98%), recall (99%), and F1-Score (99%). The Random Forest's ensemble approach allows it to effectively manage the complexity of the dataset, capturing intricate patterns and interactions between features. This robustness makes it highly reliable for distinguishing between benign and malicious samples.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Model Strength and Weaknesses observed during the study:

➢ **Naive Bayes:**

- **Strengths:** Naive Bayes is computationally efficient and performs well in scenarios with high dimensional data. It provides high precision (97%) but may not fully capture the dependencies between features, which can limit its performance in more complex classification tasks.
- **Weaknesses:** The simplicity of the model can be a limitation, as it assumes feature independence, which might not hold true for all datasets. This assumption can lead to lower recall (84%) and missed detection of some malware instances.

➢ **Random Forest:**

- **Strengths:** The Random Forest classifier excels due to its ensemble learning approach, which combines the predictions of multiple decision trees to improve accuracy, precision, and recall. It effectively handles complex patterns and interactions within the data, achieving a high F1-Score (99%).
- **Weaknesses:** Although it performs exceptionally well, the Random Forest model can be resource intensive, requiring significant memory and computational power, especially with large datasets.

➢ **KNN:**

- **Strengths:** KNN provides high precision (99%) and a strong F1-Score (96%) by classifying based on the majority vote of the nearest neighbors. It can be effective for smaller datasets or when the optimal number of neighbors(k) are well-tuned.
- **Weaknesses**: KNN's performance is sensitive to the choice of k and can become computation ally expensive as the dataset size increases. The model's efficiency degrades with large datasets, making it less practical for real-time applications.

➢ **Decision Tree:**

- **Strengths:** Decision Trees offer intuitive and interpretable models, making it easy to understand how decisions are made. They achieve high precision (98%) and recall (98%) with a good F1-Score (91%).
- **Weaknesses:** Without proper pruning, Decision Trees can overfit the training data, capturing noise and leading to poor generalization on unseen data. This can affect the model's performance, especially if the tree is allowed to grow too deep.

V. CONCLUSION

In this study, we evaluated various machine learning algorithms for the detection of malware based on dynamic API call sequences. The key findings are summarized as follows:

A. Model Performance Overview

- **Naive Bayes (GaussianNB):** Despite its simplicity and efficiency, Naive Bayes showed modest performance in accurately detecting malware, with an accuracy of 83%. Its assumption of feature independence limited its effectiveness in capturing complex patterns inherent in dynamic API sequences.
- **RandomForest (RandomForestClassifier):** This model demonstrated the highest performance, with an accuracy of 99%, along with high precision, recall, and F1-score. Its ensemble approach effectively handled feature interactions and provided robust classification results, making it the most reliable model for malware detection in this study.
- **K-Nearest Neighbors (KNN):** The performance of KNN, which achieved an accuracy of 93%, varied with different values of k, showing that the choice of k significantly impacts model effectiveness. While it provided competitive results, KNN's sensitivity to parameter tuning and its computational demands made it less consistent compared to Random Forest.
- **Decision Tree (DecisionTreeClassifier**): The Decision Tree model, with an accuracy of 98%, offered intuitive and interpretable results but showed a tendency towards overfitting. It performed well but required careful tuning to avoid model complexity issues.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Comparative Analysis

- **Random Forest's Strengths**: The Random Forest model's superior performance across various metrics highlights its strength in handling the complexities of malware detection. Its ability to aggregate multiple decision trees helps in capturing diverse patterns in the data.
- **KNN Sensitivity**: KNN's performance underscored the importance of parameter selection, indicating that the choice of k is crucial for optimizing model accuracy.
- **Naive Bayes and Decision Trees:** While these models provided valuable insights, their limitations were evident in their lower performance compared to Random Forest. Naive Bayes struggled with feature dependencies, and Decision Trees required careful pruning to manage over fitting.

C. Implications for Malware Detection

The Random Forest model is recommended for deployment in practical malware detection systems due to its balanced performance and robustness. Its effectiveness in identifying malicious software makes it a suitable choice for real-world applications.

D. Application of Malware Detection Models

Malware detection models have a wide range of applications across various domains. Here are some key areas where these models can be effectively applied:

- **Cybersecurity Defence Systems**: Use malware detection models in antivirus software and network security systems to protect devices and monitor network traffic for threats.
- **Enterprise Security:** Utilize malware detection models in corporate networks and cloud environments to protect sensitive data and systems from malware attacks.
- **Financial Sector:** Use malware detection models to identify fraudulent activities and ensure regulatory compliance by preventing breaches in financial systems.
- **Healthcare Industry:** Protect electronic health records and medical devices from malware to ensure patient data privacy and security
- **Critical Infrastructure:** Implement malware detection in industrial control systems and SCADA systems to safeguard critical infrastructure operations.
- **Software Development:** Integrate detection models into software pipelines for code analysis and application security testing to identify vulnerabilities.
- **Consumer Protection:** Incorporate malware detection in web browsers and mobile security solutions to protect against malicious websites and apps.
- **Research and Development:** Utilize detection models for threat intelligence and improving cybersecurity solutions through research.
- **Incident Response:** Use detection models for forensic analysis of malware incidents and automate response actions to mitigate threats.
- **Education and Training:** Incorporate detection models in cybersecurity training and awareness campaigns to educate professionals and end-users about malware threats.

E. Future Directions in Malware Detection

- **Exploration of Advanced Algorithms:** Investigate deep learning models like CNNs and RNNs for capturing complex patterns in API call sequences and explore ensemble methods to improve detection performance and robustness.
- **Feature Engineering and Selection:** Develop sophisticated feature extraction techniques and apply advanced feature selection methods to capture detailed patterns and focus on the most relevant features, enhancing model efficiency and accuracy.
- **Integration of Domain-Specific Knowledge:** Incorporate knowledge about malware behaviour and API interactions to improve feature engineering and model training and integrate threat intelligence data to detect novel and evolving threats.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

REFERENCES

1. J. Smith and S. Johnson, Cybersecurity Threats and Malware Analysis. Oxford University Press, 2022.
2. Johnson and R. Lee, Advanced Techniques in Malware Detection. Springer, 2023.
3. Faraahanwaaar, Malware pycaret classification, https://www.kaggle.com/code / faraahanwaaar / malware- pycaret-classification/input, [Accessed: dy namic api call sequence per malware 100 0 306.csv], 2023.
4. M. Brown et al., "A comparative study of machine learning techniques for malware detection," Journal of Cybersecurity, vol. 45, no. 3, pp. 123–145, 2021.
5. W. Zhang et al., "Feature engineering for malware detection using api call sequences," IEEE Transactions on Information Forensics and Security, vol. 16, no. 1, pp. 67–78, 2022.
6. L. Chen and Y. Liu, "Dynamic analysis of malware us ing api call sequences," Computers & Security, vol. 108, pp. 102–114, 2022.
7. R. Gupta et al., "Smote-based techniques for handling class imbalance in malware detection," International Journal of Computer Applications, vol. 183, no. 12, pp. 21–30, 2022.
8. X.Liand D. Kim, "Evaluating machine learning models for malware classification," Journal of Machine Learning Research, vol. 24, no. 4, pp. 156–175, 2023.
9. J. Wang and M. Gonzalez, "Deep learning approaches for malware detection: A review," ACM Computing Surveys, vol. 56, no. 2, pp. 1–34, 2023.
10. Y. Zhang et al., "Ensemble methods in malware detection: A survey," IEEE Access, vol. 10, pp. 89456 89471, 2022.
11. K. Lee and A. Parker, Applied Machine Learning for Cybersecurity. CRC Press, 2023.
12. J. Turner and M. Robinson, "Contextual analysis in malware detection: Integrating domain knowledge," Journal of Information Security, vol. 29, no. 3, pp. 77–92, 2023.
13. **A.** White, "Understanding api calls in malware analysis," Cybersecurity Research and Practice, vol. 10, no. 1, pp. 45–60, 2023.
14. Patel and M. Lopez, "Comparative analysis of naive bayes and random forest for malware detection," Data Science Journal, vol. 21, no. 5, pp. 201–215, 2022.
15. Kumar and P. Sharma, "The role of threat intelligence in enhancing malware detection systems," Journal of Computer Security, vol. 40, no. 2, pp. 123–140, 2023.
16. Ali et al., "Evaluating the performance of k-nearest neighbors and decision trees in malware classification," Journal of Computing Sciences, vol. 32, no. 7, pp. 312 326, 2022.
17. Anderson and D. McGrew, "Machine learning for malware detection: The state of the art," Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pp. 3–22, 2023.
18. Doupe et al., "Exploring dynamic and static features for malware classification," in Proceedings of the 37th International Conference on Machine Learning (ICML), PMLR, 2023, pp. 1025–1035.
19. Kil et al., "Detecting malware with dynamic api call sequence graphs," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 2, pp. 345–358, 2024.
20. L. Nataraj et al., "A comparative analysis of deep learning techniques for malware detection," IEEE Transactions on Cybernetics, vol. 54, no. 5, pp. 2301–2313, 2023.
21. M. O'Sullivan et al., "Improving malware detection through api call sequence analysis," Journal of Information Security and Applications, vol. 63, pp. 101–115, 2024.
22. S. Rayana and P. K. Maji, "Malware detection using sequence-based representation and classification," IEEE Access, vol. 11, pp. 27321–27334, 2023.
23. K. Rieck et al., "Combining static and dynamic analysis for efficient malware detection," International Journal of Information Security, vol. 22, no. 1, pp. 7–19, 2023.
24. Santos et al., "Opem: Open platform for experimental malware research," IEEE Transactions on Information Forensics and Security, vol. 18, no. 4, pp. 1341–1355, 2023.
25. Saxe and K. Berlin, "Deep neural network-based malware detection using two-dimensional binary program features," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2023, pp. 685–693.
26. Y. Yao et al., "Ensemble learning for effective malware detection using api call sequences," Knowledge-Based Systems, vol. 263, pp. 1–15, 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details