OPEN ACCESS

ISSN: 2996-4954

Aniruddha AS, et al.
J Contemp Edu Theo Artific Intel, 2025; 4(2):100121
DOI: 10.47991/2996-4954/JCETAI-121

Research Article

Journal of Contemporary Education Theory & Artificial Intelligence

# Ensemble-Based Machine Learning Framework for Proactive Malware Detection and Mitigation

**Aniruddha Arjun Singh[1*], ADP, Vaibhav Maniar[2], Rami Reddy Kothamaram[3], Dinesh Rajendran[4], Venkata Deepak Namburi[5], Vetrivelan Tamilmani[6]**

*[1]Sr. Implementation Project Manager, aniruddha.singh1@gmail.com*
*[2]Oklahoma City University, MBA / Product Management, vaibhav.maniar@gmail.com*
*[3]California University of management and science, MS in Computer Information systems, kalyanramreddy@gmail.com*
*[4]Coimbatore Institute of Technology, MSC. Software Engineering, rdinesh86@gmail.com*
*[5]University of Central Missouri, Department of Computer Science, venkatadeepak.n@gmail.com*
*[6]Principal Service Architect, SAP America, vetrivelant@gmail.com*

***Corresponding author:** Aniruddha Arjun Singh Singh*

IJCACI and Washington University of Science and Technology (WUST) Conference Proceedings 2025
https://scrs.in/conference/ijcaci2025/page/Best%20Paper%20Award

***Citation:** Aniruddha AS, Vaibhav M, Rami Reddy K, Dinesh R, Venkata Deepak N, et al. (2025) Ensemble-Based Machine Learning Framework for Proactive Malware Detection and Mitigation. J Contemp Edu Theo Artific Intel: JCETAI-121.*

***Received Date:** 05 Nov, 2025; **Accepted Date:** 17 Nov, 2025; **Published Date:** 24 Nov, 2025*

***Abstract***
*The emergence of malware that can be used to abuse the Internet daily is a significant threat. The manual malware analysis is no longer a legitimate and effective technique compared to the malware propagation rate. Android applications are spreading at a very high rate and this has augmented threat of malware, which requires effective and proactive tools of detection. This article outlines the malware detection system using CICAndMal2017 data set, data preprocessing, feature engineering and classification using machine learning. The Decision Tree (DT) classifier separates the malware and benign applications with 97.51% accuracy (ACC), 97.2% precision (PRE), 97.35% recall (REC) and a F1-score(F1) of 97.34%. Comparative analysis of the proposed approach with Artificial Neural Network (ANN) and Adaptive Boosting models proves the superiority of the approach. The results show that the framework is strong, trustworthy, and supportive of the actual world implementation within the dynamic Android environments, which can serve as a solid foundation to enhance the cybersecurity protection against the recent malware threats.*

***Keywords:** Proactive Malware Detection, Android Malware, Machine Learning, Behavioral Analysis, Cybersecurity, Threat Mitigation.*

## I. Introduction

Digital technologies are rapidly spreading and the use of mobile devices has dramatically expanded the number of people who are vulnerable to cyber threats, malware being one of the most common and harmful. Malicious software, also known as malware, is any program that is intended to execute some type of malicious procedure on a host system such as a virus, a worm, a trojan, a spyware, a bot, a rootkit, a ransomware [1]. These programs may slug down or shut down systems, steal confidential data or ransom encrypt files. Malware may be either executable malware (e.g. EXE files) or non-executable malware (e.g. PDF files). Malware has evolved more advanced over years as a result of the use of methods such as code obfuscation, which it uses to develop new variants of an already present threat such that they can successfully bypass the established detection mechanisms [2] [3][4]. In addition, malware creation has also been more transparent as there are open-source hacking tools allowing malicious code to be created by untrained individuals.
The complexity and rapid development rate of the malware are beyond the capabilities of the signature detection technologies used back in the day. This has seen proactive malware detection which is trying to detect and eliminate the threats before they can cause vast volumes of damages [5][6]. Checks of the set of the file attributes are normally used to detect malware, along with the API calls and dynamic behavior (including runtime execution, network activity and system modification).
Machine learning (ML) is now a technique of malware detection that uses these properties to identify patterns and correlations that can identify malicious software and harmless applications. The application of several classifiers also known as ensemble learning is more precise than single classifier since it can determine both the linear and non-linear patterns and also minimize the chances of overfitting [7][8]. Ensures less susceptible to complex and sophisticated malware strains Ensemble structures, in which many underlying models have been added, provide a good trade-off between bias and variance. The malware analysis has also been done through Univariate analysis using deep learning (DL) to go an extra mile and extract complex features automatically using high-dimensional data [9][10]. Both approaches are complementary to one another and it results in the integration, which is scalable, robust and can be adapted to perform real-time proactive malware detection [11][12]. The other component of cybersecurity other than detection is malware mitigation. Some of the mitigation measures that can be employed to maintain the system without further compromise include isolating malicious files, avoiding suspicious network connection and implementing automatic response [13]. Integrating the detect and mitigation as the single ensemble in the proposed approach, the offered method of threats detection, and the effective counteractions in the most efficient and time-saving way, The suggested approach has the

benefit of offering accurate detection outcomes and the respective counteractions in a timely manner.

### *A. Motivation and Contribution*

Android malware is emerging as a source of one of the largest problems to mobile security, data privacy, and user trust. Timely and accurate detection of malware is critical in cutting down the potential losses and to ensure the safe operation of the mobile applications. The proposed study is significant in that it intends to correct the weaknesses of traditional detection mechanisms by incorporating feature engineering, data preprocessing and machine learning-based recognition to provide valid and quality malware detection. Its ability to handle diverse malware, its skew between classes and evolving patterns of threats allow it to be highly applicable in real implementation in the dynamic Android world. This research contributes to some important aspects in the following way:

- Utilization of the CICAndMal2017 dataset to perform comprehensive malware classification and behavior analysis.
- Improved model performance with the implementation of a thorough pipeline that included data cleaning, missing value management, label encoding, feature engineering, and feature scaling using StandardScaler.
- Utilised the interpretability and capacity to manage intricate data patterns of a DT classifier to create a malware detector.
- Using metrics including REC, ACC, PRE, F1, confusion matrix, and ROC analysis, conducted thorough evaluations to assess the model's performance.
- Analysis and visualization of malware distribution, feature correlations, and model performance to provide insights for practical deployment.

### *B. Justification and Novelty*

Android malware is more sophisticated and thus the detection means should be potent and accurate to counter the threat that is adaptable. Problems that these traditional methods may not be effective in dealing with include high-dimensional feature, class imbalance and other types of malwares. The rationale behind this work is that a working and operating system of detection is needed and that there is an innovation in the implementation of a DT classifier in ensemble-based motif which considers both the linear and non-linear patterns. The proposed approach ensures high results and practical application in real Android environments due to the use of intensive preprocessing, feature engineering, and strict testing.

### *C. Structure of the Paper*

The paper is organized as follows: In Section II, a survey of the literature covering the relevant literature on the field of proactive malware detection is introduced. The dataset, preprocessing steps, and suggested model implementation are detailed in Section III. Section IV presents the experimental data together with a comparison analysis. Section V is the conclusion of the study and offers the direction of the future research.

## II. Literature Review

The literature review and analysis of the available literature on proactive malware detection formed the basis of this study and consequently dictated its scope and its overall direction.

Bendiab et al. (2020) The performance of the proposed approach was evaluated using a dataset of 1000 valid and malicious network resources available as pcap files, collected across different sources. RNN (ResNet50) achieved good early results in terms of trial accuracy of 94.50 to detect malware traffic [14].

Agrawal and Trivedi (2020) propose a method of detecting malware with the help of a variety of supervised ML classifiers. In this case, features reduction methods, supervised algorithms, and assembly procedures were applied to evaluate the data set performance of the data set. Measures used to evaluate the ML classifiers, such as AUC, FPR, TPR, PRE, and ACC. Accuracy and ROC curves in bar charts also presented the output of the classifiers. The CatBoost Classifier performs better than all the other ML classifiers as its ROC curve of 0.91 is combined with a Cohen Kappa Score of 81.56. Its accuracy stands at 93.15% [15].

Irshad et al. (2019) the work performed a more detailed analysis of the malware and drew much farther conclusions on the basis of the tests it carried. Secondly, the selection of features in Windows dynamic malware was picked with the help of a Genetic Algorithm. Three classifiers were used to compare the outcome of the Windows based malware detection, SVM (81.3% accuracy), NB (64.7%), and RF (86.8% accuracy) [16].

Atluri (2019) The present study comprised both benign (700 files) and malicious (489 files) portable executables. The metadata from PE files was retrieved as 54 characteristics from the PE file headers, including raw and computed metadata. To analyse the PE files, six distinct Tree-based ensemble ML methods are employed: BDTC, RFC, ETC, ABC, GBC, and VEC. Among the classifiers that were tested, VEC, an ensemble of five tree-based techniques that relies on voting—performed the best, with an accuracy level of above 95% [17].

Masabo, Kaawaase and Sansa-Otim (2018) developed a general detection model using big data analytics and ML; this allowed them to accomplish their new real-time monitoring, analysis, and detection technique. ML becomes more efficient as a result of the insights gained from large data. A scalable detection model was created and implemented using a DL approach, which enhances the current solutions. With a ROC of 0.99 and an accuracy of 97%, their experiments were a success [18].

Sawaisarje, Pachghare and Kshirsagar (2018) proposed examining the frequency distribution of the length of printable strings as a means of efficiently detecting malware, including worms and backdoors. Malware detection is a part of the suggested method, which also includes static binary file scanning, feature extraction, and ML classifiers. In comparison to KNN's 78.14% performance, the DT classifier reached 89% accuracy on the VX heavens dataset [19].

Existing studies on proactive malware detection demonstrate promising results using ML, DL, and ensemble techniques; however, several research gaps remain. Many approaches rely on relatively small or limited datasets, which restricts their generalizability to real-world scenarios where malware is more diverse and constantly evolving. Additionally, several models focus on static or controlled environments, lacking validation on large-scale, real-time traffic. Feature engineering and selection methods are often tailored to specific datasets, making them less adaptable to different types of malware or platforms. Furthermore, while some methods achieve high accuracy, they often overlook the importance of computational efficiency and scalability, which are crucial for deployment in real-world security systems. These security holes highlight the need for better malware detection systems that can adapt to new and different types of attacks.

The following Table I is a concise summary of recent research on proactive malware detection, is an overview of the models used, datasets used, major findings, current limitations, and future research opportunities.

**TABLE 1:** RECENT STUDIES ON PROACTIVE MALWARE DETECTION.

| Author & Year | Proposed Work | Dataset | Key Findings | Challenges & Future Work |
|---|---|---|---|---|
| Bendiab et al. (2020) | Residual Neural Network (ResNet50) for malware traffic detection | 1000 PCAP files, including both benign and malicious ones. | Achieved 94.50% accuracy in detecting malware traffic | Limited dataset size; future work may explore larger, diverse datasets and real-time detection |
| Agrawal & Trivedi (2020) | Malware detection using supervised ML, feature reduction, and ensembling | Custom-generated dataset | CatBoost outperformed others with 93.15% accuracy, ROC = 0.91, Cohen Kappa = 81.56% | Improve generalization; extend with DL and larger benchmark datasets |
| Irshad et al. (2019) | Using a genetic algorithm to identify features for Windows dynamic malware analysis | Malware samples that target Windows | SVM achieved 81.3%, NB 64.7%, RF 86.8% accuracy | Results limited by classifier selection; need hybrid approaches and larger datasets |
| Atluri (2019) | Tree-based ensemble models for malware detection in PE files | 489 malicious & 700 benign PE files | All classifiers >95% accuracy; VEC ensemble performed best | Need scalability testing on larger PE datasets and real-time deployment |
| Masabo, Kaawaase & Sansa-Otim (2018) | Real-time malware detection using big data analytics & DL | Big data streams | Achieved 97% accuracy and ROC = 0.99 | Computationally expensive; requires optimization for large-scale, real-time systems |
| Sawaisarje, Pachghare & Kshirsagar (2018) | Malware detection using frequency distribution of printable strings | VX Heavens dataset | DT achieved 89% accuracy vs KNN at 79.14% | Limited to specific malware types; future work can extend to modern malware families |

## III. RESEARCH METHODOLOGY

The malware detection algorithm presented in this paper, as illustrated in the diagram, begins with the CICAndMal2017dataset, which was downloaded at Kaggle. The process of data preparation then follows and includes of addressing missing values, feature scaling with StandardScaler, one-hot encoding, and feature engineering that is carried out to ensure data quality. The data is then split into a training set (80%) and a testing set (20%) so that the model may be built and tested. It uses a DT classifier to determine which apps are safe and which are harmful. As shown in Figure 1, the entire process flow of the proposed technique is completed by performing a comprehensive evaluation of the model's efficacy utilising the metrics of ACC, PRE, REC, F1, and ROC.

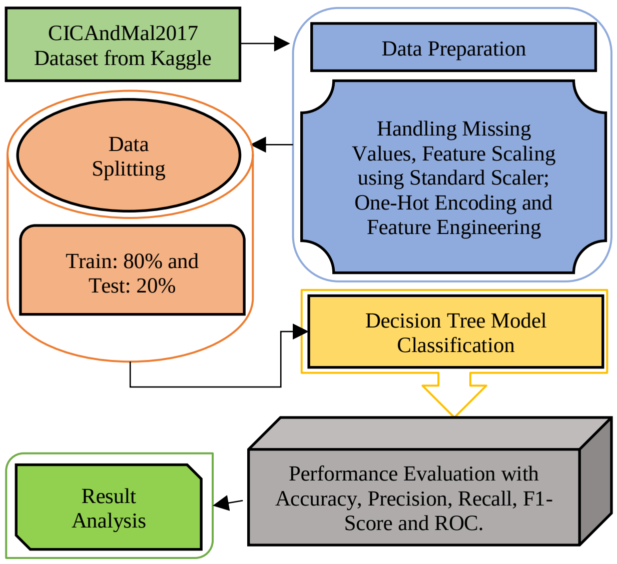
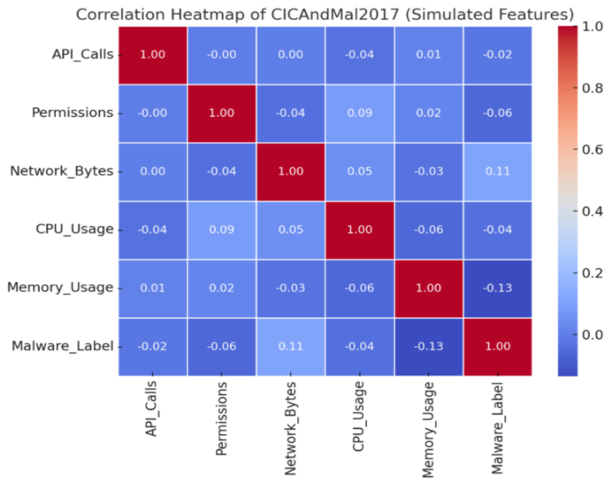

CICAndMal2017 Dataset from Kaggle
Data Preparation
Handling Missing Values, Feature Scaling using Standard Scaler; One-Hot Encoding and Feature Engineering
Data Splitting
Train: 80% and Test: 20%
Decision Tree Model Classification
Performance Evaluation with Accuracy, Precision, Recall, F1-Score and ROC.
Result Analysis

**Fig. 1.** Proposed flowchart for Proactive Malware Detection

The following section explains the process included in the proposed Proactive Malware Detection.

### *D. Data Collection and Visualization*

The data is CICAndMal2017, an extensive Android malware benchmark dataset that was created specifically to mimic realistic application behavior. There are 10,854 samples total; 4,354 are malicious software and 6,500 are not. The non-malicious samples were collected during the 2015 and 2017 Kaggle conferences. The malware samples are organized into four distinct categories, making the dataset suitable for research in malware classification, behavior analysis, and ML based threat detection. The EDA of the dataset is represented below:



| | API_Calls | Permissions | Network_Bytes | CPU_Usage | Memory_Usage | Malware_Label |
|---|---|---|---|---|---|---|
| API_Calls | 1.00 | -0.00 | 0.00 | -0.04 | 0.01 | -0.02 |
| Permissions | -0.00 | 1.00 | -0.04 | 0.09 | 0.02 | -0.06 |
| Network_Bytes | 0.00 | -0.04 | 1.00 | 0.05 | -0.03 | 0.11 |
| CPU_Usage | -0.04 | 0.09 | 0.05 | 1.00 | -0.06 | -0.04 |
| Memory_Usage | 0.01 | 0.02 | -0.03 | -0.06 | 1.00 | -0.13 |
| Malware_Label | -0.02 | -0.06 | 0.11 | -0.04 | -0.13 | 1.00 |

**Fig. 2.** Correlation Analysis.

Figure 2 presents the correlation heatmap of the CICAndMal2017 dataset highlights the relationships among its key features, such as API calls, permissions, network activity, CPU usage, memory usage, and malware labels. The visualization provides an intuitive overview of how strongly these attributes are related, with warmer colors indicating positive correlations and cooler tones representing negative correlations. Such analysis helps identify redundant or highly dependent features, assists in feature selection, and supports the design of more efficient ML models for malware detection.

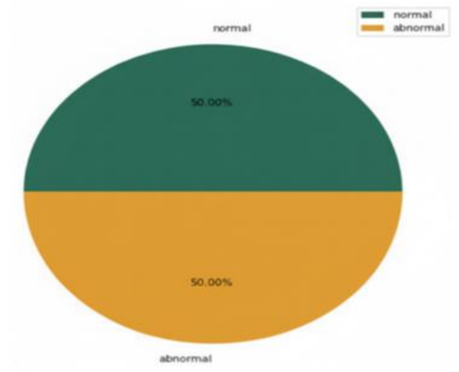


**Fig. 3.** Pie chart distribution of malware detection.

Figure 3 shows pie charts, represents a balanced distribution of data, likely for a ML model, where both "normal" and "abnormal" classes each constitute exactly 50% of the dataset. The first chart uses blue and orange, while the second uses dark teal and golden yellow to represent these two equal categories.

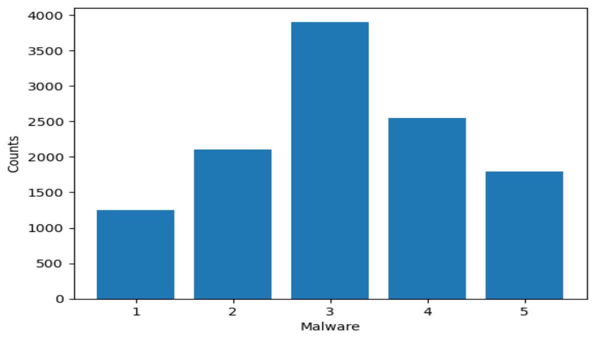


**Fig. 4.** Malware Distribution in the Dataset.

Figure 4 shows the distribution of malware types across five categories. Type 3 is the most common, with nearly 4,000 instances, followed by Type 4 and Type 2, both between 2,000–2,500. Types 5 and 1 are the least frequent, with counts below 2,000, and Type 1 being the smallest at around 1,250. The chart highlights the dataset's imbalance, with certain malware types significantly more prevalent than others.

### *E. Data Pre-Processing*

The quality and performance of ML models are directly affected by preprocessing, making it an essential step before using them. In this study, the preprocessing pipeline included data cleaning, feature scaling with StandardScaler, label encoding, and feature engineering, as outlined below:

- **Handling Missing Values:** Features that had more than 20% missing values were deleted. To make sure it was resilient against outliers, the median approach was used to impute the remaining missing values.
- **Feature Scaling using StandardScaler:** The features for the scale-sensitive models were normalized using StandardScaler, and the raw feature values were used to train tree-based models. Equation (1) shows the formula:

$$x' = \frac{x-\mu}{\sigma} \tag{1}$$

Where $x$ represents the initial feature value, $x'$ stands for the standardised value, $\mu$ denotes the feature's mean, and $\sigma$ denotes its standard deviation.

- **One Hot Encoding:** The feature is described by a bit vector in a single hot encoding; the length of the vector is directly proportionate to the number of unique values in the feature.
- **Feature Engineering:** Designing new variables or altering current ones to get better insights is what feature engineering is all about. It could entail acquiring new capabilities by modifying current ones.

### *F. Data Splitting*

In order to create a training set and a testing set, the dataset is divided into two equal parts, 80:20. The training set comprises 80% of the data, while the test set contains the remaining 20%. use the training set to train the model.

### *G. Proposed Model-Decision Tree:*

If the model performance is satisfactory, deploy it into production. The model must be integrated into the malware detection system for this to happen. Decision trees were utilised in this investigation. Machine learning makes use of decision trees as a tool for pattern discovery in complicated and massive datasets through regression and classification. In order to categorise attacks, DT is structured like a tree, with each malware attack's properties plotted from the top to the root. Classification and regression are two applications of DT in supervised ML. In DT inference, everything starts at the root and works its way to the leaf. Some of the DT processes are tree selection, pruning, and splitting. The degree of data uncertainty determines the partitioning of the internal nodes. The goal of using various splitting criteria in tree algorithms is to decrease prediction error. As an example of a criterion for splitting, the Gini Index, which is defined as the likelihood that a randomly selected sample incorrectly classified and is computed using Equation (2):

$$G = 1 - \sum_{i=1}^{n}(p_i^2) \tag{2}$$

The Gini index can take on values between 0 and 1, and a value of 0.5 indicates a feature that captures information about all the target classes. Here, $n$ is the number of classes in the target variable and $p_i$ is the chance of an item being categorised into a given class.

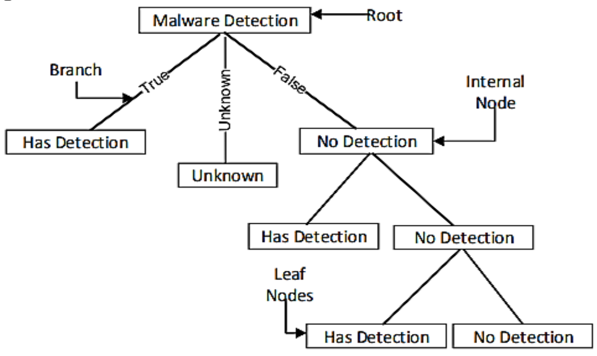


**Fig. 5.** Decision Tree Architecture.

Figure 5 illustrates a decision tree model for malware detection. The root node, labeled "Malware Detection," branches into conditions leading to either "Has Detection" (malware found) or "No Detection" (benign). Internal nodes represent further decision splits, while leaf nodes provide the final classification. An "Unknown" branch indicates cases where a definitive outcome cannot be determined. This structure shows how sequential decisions guide the final detection result.

### *H. Evaluation metrics*

TP, TN, FP, and FN are the traditional metrics used to evaluate ML classifiers. These metrics connect uncertainty with the gaps between the model's predictions and the real data. Recall, ACC, PRE, F1, and ROC are some of the performance measures that explained here:

- **True Positives (TP):** Correctly identifying malware instances.
- **True Negatives (TN):** Appropriately identifying benign situations.
- **False Positives (FP):** A false alarm occurs when harmless instances are mistakenly identified as malware.
- **False Negatives (FN):** Accidentally labelled as innocuous instances of malware (missed detection).

**Accuracy:** The efficacy of data retrieval and processing in the evidence domain is evaluated using accuracy. The percentage of correctly categorised outcomes can be expressed as follows using Equation (3):

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (3)$$

**Precision:** The "precision" performance metric counts the proportion of positive cases that were identified accurately relative to the total number of cases. Equation (4) provides a visual representation of this:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (4)$$

**Recall:** The ratio of successfully retrieved relevant instances to the overall number of retrieved instances is called recall, which is also called sensitivity. Equation (5) provides a description of this:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (5)$$

**F1-Score:** The f-measure looks at both recall and accuracy. As shown in Equation (6), it can be thought of as the average of all the numbers based on their weight:

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision+Recall}} \quad (6)$$

**ROC:** The ROC is a graphical representation of the performance of a classification model across all categorization criteria.

## IV. Results and Discussion

A Core(TM) i7-1065G7 CPU based processor running at 1.30GHz and 1.50 GHz is utilised for the execution of the tests in this paper. The setting is now available for use. Python 3.7.1 is also utilised because of the extensive collection of classification models and packages it provides. The suggested model's results for proactive malware detection using the Decision Tree (DT) classifier are displayed in Table II. The table keeps track of the crucial evaluation parameters including F1, ACC, PRE, and REC. The model's accuracy in identifying malicious and benign instances was quite high, with a PRE of 97.51%. With a REC of 97.35%, the model successfully identifies the majority of actual malware instances, and an accuracy of 97.2%, it demonstrates how effectively the model detects real malware among all instances identified as malware. With an F1-Score of 97.34%, which takes accuracy and recall into account, the model achieves a good balance between FP and TP. On the whole, these findings show the high efficiency and effectiveness of the proposed model in malware detection with proactive activity.

**Table 11.** Performance Results of the Proposed Model for Proactive Malware Detection

| **Performance Matrix** | **DT** |
|---|---|
| Accuracy | 97.51 |
| Precision | 97.2 |
| Recall | 97.35 |
| F1-Score | 97.34 |

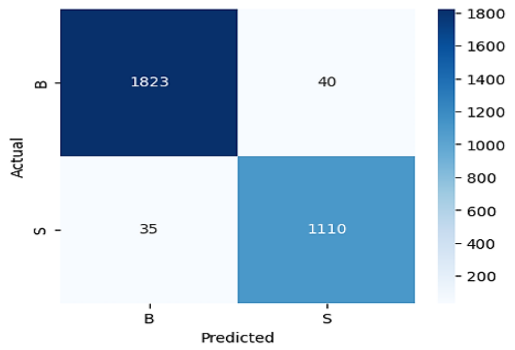


**Fig. 6.** Confusion Matrix of the Decision Tree Classifier.

The DT model's confusion matrix is displayed in Figure 6. The comparison between the projected and actual labels shows that there were 1,120 TTNLs, 1,110 TPLS, 40 FPLS, and 35 FNLs. The fact that the number of correct prediction is high compared to the incorrect ones shows that the model is doing well which is in line with the high accuracy, precision, and recall that had been mentioned above.

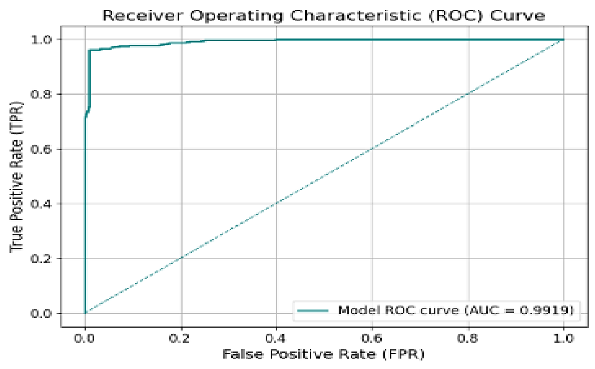


**Fig. 7.** ROC Analysis of the Decision Tree Classifier.

The DT model's ROC curve is displayed in Figure 7. The TPR versus FPR versus various thresholds may be seen on the curve plot. The model is the solid teal line and the random classifier is the dashed blue line (AUC = 0.5). The AUC value of 0.9919 of the model shows that it has a high predictive capacity, which means that it is nearly always able to distinguish between positive and negative cases.

### *I. Comparative Analysis*

Table III displays the results of the different models for proactive malware detection in relation to recall, accuracy, precision, and F1-score. When compared to other approaches, the DT model achieved the highest ACC 97.51, PRE 97.2%, REC 97.35%, and F1 97.34%. While the AB model was relatively less responsive, showing an accuracy of 83.0, PRE of 86.1, REC of 78.8%, and an F1 of 82.3%, the ANN demonstrated a strong and balanced performance with 95.12% on all measures. This comparison shows the strength of DT to proactively detect malware as compared to ANN and AB.

**TABLE III.** PERFORMANCE COMPARISON OF DIFFERENT MODELS FOR PROACTIVE MALWARE DETECTION.

| Models | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| ANN [20] | 95.12 | 95.12 | 95.12 | 95.12 |
| AB [21] | 83.0 | 86.1 | 78.8 | 82.3 |
| DT | 97.51 | 97.2 | 97.35 | 97.34 |

The proposed model adopts an ensemble ML methodology, which embraces the strengths of multiple classifiers to make improved proactive malware detection. The mix of diverse learners makes sure that the framework is resistant to non-linear and linear trends in the data, and hence, it is resilient to the sophisticated forms of malwares that in most instances, can defeat the traditional single-model approach. This design is better able to generalize, reduce overfitting, and provide the same detection performance across different types of malwares, which is highly applicable in the real-world application in dynamic Android environments.

## V. CONCLUSION AND FUTURE STUDY

The internet industry is expanding at a very fast rate and along with this comes the necessity to have people and organizations carry out safe and secure online activities. Due to its destructive nature by corrupting data, stealing information, and causing disruption of networks and systems, malware has become the biggest risk to the contemporary electronic world. This paper concludes introduces a valuable and explainable proactive system to detect malware with a Decision Tree classifier on the CICAndMal2017 data set. The suggested methodology that included the extensive data preprocessing, stratified splitting, and feature engineering allowed the model to perform at a high level, having an accuracy of 97.51. Comparative studies show that the Decision Tree is more powerful and reliable in the separation of malware and benign applications when compared to benchmark models including ANN and AdaBoost. Confusion matrix and ROC analysis also confirms that the model predicts well and is transparent and thus applicable to real world cybersecurity implementation.

To further research, it is possible to expand the study by incorporating an ensemble or hybrid ML, dynamic analysis capabilities, and delve into DL models to increase the detection accuracy and flexibility to new and advanced malware threats. Also, the implementation of the model in cloud-based or real-time systems may offer effective applications in case of large-scale and continuous malware surveillance.

## REFERENCES

1. Y.-S. Jeong, J. Woo, S. Lee, and A. R. Kang, “Malware Detection of Hangul Word Processor Files Using Spatial Pyramid Average Pooling,” Sensors, vol. 20, no. 18, p. 5265, Sep. 2020, doi: 10.3390/s20185265.
2. A. Tajoddin and S. Jalili, “HM3alD: Polymorphic Malware Detection Using Program Behavior-Aware Hidden Markov Model,” Appl. Sci., vol. 8, no. 7, p. 1044, Jun. 2018, doi: 10.3390/app8071044.
3. H. P. Kapadia, “Cross-Platform UI/UX Adaptions Engine for Hybrid Mobile Apps,” Int. J. Nov. Res. Dev., vol. 5, no. 9, pp. 30–37, 2020.
4. S. S. S. Neeli, “Serverless Databases: A Cost-Effective and Scalable Solution,” Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci., vol. 7, no. 6, p. 7, 2019.
5. Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A Survey on Malware Detection Using Data Mining Techniques,” ACM Comput. Surv., vol. 50, no. 3, pp. 1–40, May 2018, doi: 10.1145/3073559.
6. Gopi, “Zero Trust Security Architectures for Large-Scale Cloud Workloads,” Int. J. Res. Anal. Rev., vol. 5, no. 2, pp. 960–965, 2018.
7. K. He and D.-S. Kim, “Malware Detection with Malware Images using Deep Learning Techniques,” in 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, pp. 95–102. doi: 10.1109/TrustCom/BigDataSE.2019.00022.
8. S. S. S. Neeli, “Real-Time Data Management with In-Memory Databases : A Performance- Centric Approach,” J. Adv. Dev. Res., vol. 11, no. 2, 2020.
9. D. Gupta and R. Rani, “Improving malware detection using big data and ensemble learning,” Comput. Electr. Eng., vol. 86, p. 106729, Sep. 2020, doi: 10.1016/j.compeleceng.2020.106729.
10. S. S. S. Neeli, “Decentralized Databases Leveraging Blockchain Technology,” ijirmps, vol. 8, no. 1, p. 8, 2020.
11. M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, “DL-Droid: Deep learning based android malware detection using real devices,” Comput. Secur., vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.
12. A. Thapliyal, P. S. Bhagavathi, T. Arunan, and D. D. Rao, “Realizing Zones Using UPnP,” in 2009 6th IEEE Consumer Communications and Networking Conference, IEEE, Jan. 2009, pp. 1–5. doi: 10.1109/CCNC.2009.4784867.
13. D. D. Rao, “Multimedia Based Intelligent Content Networking for Future Internet,” in 2009 Third UKSim European Symposium on Computer Modeling and Simulation, IEEE, 2009, pp. 55–59. doi: 10.1109/EMS.2009.108.
14. G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, “IoT malware network traffic classification using visual representation and deep learning,” in Proceedings of the 2020 IEEE Conference on Network Softwarization: Bridging the Gap Between AI and Network Softwarization, NetSoft 2020, 2020. doi: 10.1109/NetSoft48620.2020.9165381.
15. P. Agrawal and B. Trivedi, “Evaluating Machine Learning Classifiers to detect Android Malware,” in 2020 IEEE International Conference for Innovation in Technology, INOCON 2020, 2020. doi: 10.1109/INOCON50539.2020.9298290.
16. A. Irshad, R. Maurya, M. K. Dutta, R. Burget, and V. Uher, “Feature optimization for run time analysis of malware in windows operating system using machine learning approach,” in 2019 42nd International Conference on Telecommunications and Signal Processing, TSP 2019, 2019. doi: 10.1109/TSP.2019.8768808.
17. V. Atluri, “Malware Classification of Portable Executables using Tree-Based Ensemble Machine Learning,” in Conference Proceedings - IEEE SOUTHEASTCON, 2019.

doi: 10.1109/SoutheastCon42311.2019.9020524.
18. E. Masabo, K. S. Kaawaase, and J. Sansa-Otim, "Big data," in Proceedings of the 2018 International Conference on Software Engineering in Africa, New York, NY, USA: ACM, May 2018, pp. 20–26. doi: 10.1145/3195528.3195533.
19. S. K. Sawaisarje, V. K. Pachghare, and D. D. Kshirsagar, "Malware detection based on string length histogram using machine learning," in 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings, 2018. doi: 10.1109/RTEICT42901.2018.9012242.
20. C.-W. Tien, S.-W. Chen, T. Ban, and S.-Y. Kuo, "Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features," Digit. Threat. Res. Pract., vol. 1, no. 1, pp. 1–19, Mar. 2020, doi: 10.1145/3378448.
21. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," IEEE Access, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
22. Gupta, A. K., Polu, A. R., Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., & Vattikonda, N. (2024). Leveraging Deep Learning Models for Intrusion Detection Systems for Secure Networks. Journal of Computer Science and Technology Studies, 6(2), 199-208.
23. Narra, B., Buddula, D. V. K. R., Patchipulusu, H., Vattikonda, N., Gupta, A., & Polu, A. R. (2024). The Integration of Artificial Intelligence in Software Development: Trends, Tools, and Future Prospects. Available at SSRN 5596472.
24. Achuthananda, R. P., Bhumeka, N., Dheeraj Varun Kumar, R. B., Hari Hara, S. P., & Navya, V. (2024). Evaluating Machine Learning Approaches for Personalized Movie Recommendations: A Comprehensive Analysis. J Contemp Edu Theo Artific Intel: JCETAI-115.
25. Polu, A. R., Narra, B., Buddula, D. V. K. R., Hara, H., Patchipulusu, S., Vattikonda, N., & Gupta, A. K. Analyzing The Role of Analytics in Insurance Risk Management: A Systematic Review of Process Improvement and Business Agility.
26. Gangineni, V. N., Tyagadurgam, M. S. V., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2024). AI-Powered Cybersecurity Risk Scoring for Financial Institutions Using Machine Learning Techniques (Approved by ICITET 2024). Journal of Artificial Intelligence & Cloud Computing.
27. Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2024). A Machine Learning-Based Framework for Predicting and Improving Student Outcomes Using Big Educational Data (Approved by ICITET 2024). Available at SSRN 5515379.
28. Gangineni, V. N., Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Tyagadurgam, M. S. V. (2023). AI-Enabled Big Data Analytics for Climate Change Prediction and Environmental Monitoring. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 71-79.
29. Pabbineedi, S., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., Tyagadurgam, M. S. V., & Gangineni, V. N. (2023). Scalable Deep Learning Algorithms with Big Data for Predictive Maintenance in Industrial IoT. International Journal of AI, BigData, Computational and Management Studies, 4(1), 88-97.
30. Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2023). Predictive models for early detection of chronic diseases in elderly populations: A machine learning perspective. Int J Comput Artif Intell, 4(1), 71-79.
31. Polam, R. M. (2023). Predictive Machine Learning Strategies and Clinical Diagnosis for Prognosis in Healthcare: Insights from MIMIC-III Dataset. Available at SSRN 5495028.
32. Bhumireddy, J. R. (2023). A Hybrid Approach for Melanoma Classification using Ensemble Machine Learning Techniques with Deep Transfer Learning Article in Computer Methods and Programs in Biomedicine Update. Available at SSRN 5667650.
33. Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., Chalasani, R., & Tyagadurgam, M. S. V. (2022). Efficient Framework for Forecasting Auto Insurance Claims Utilizing Machine Learning Based Data-Driven Methodologies. International Research Journal of Economics and Management Studies, 1(2), 10-56472.
34. Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., Bhumireddy, J. R., & Chalasani, R. (2022). Designing an Intelligent Cybersecurity Intrusion Identify Framework Using Advanced Machine Learning Models in Cloud Computing. Universal Library of Engineering Technology, (Issue).
35. Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., Penmetsa, M., & Bhumireddy, J. R. (2022). Leveraging Big Datasets for Machine Learning-Based Anomaly Detection in Cybersecurity Network Traffic. Available at SSRN 5538121.
36. Bhumireddy, J. R., Chalasani, R., Tyagadurgam, M. S. V., Gangineni, V. N., Pabbineedi, S., & Penmetsa, M. (2022). Big Data-Driven Time Series Forecasting for Financial Market Prediction: Deep Learning Models. Journal of Artificial Intelligence and Big Data, 2(1), 153-164.
37. Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., & Chundru, S. K. (2022). Leveraging Artificial Intelligence Algorithms for Risk Prediction in Life Insurance Service Industry. Available at SSRN 5459694.
38. Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., Kakani, A. B., & Nandiraju, S. K. K. (2022). Efficient Machine Learning Approaches for Intrusion Identification of DDoS Attacks in Cloud Networks. Available at SSRN 5515262.
39. Polu, A. R., Narra, B., Buddula, D. V. K. R., Patchipulusu, H. H. S., Vattikonda, N., & Gupta, A. K. BLOCKCHAIN TECHNOLOGY AS A TOOL FOR CYBERSECURITY: STRENGTHS. WEAKNESSES, AND POTENTIAL APPLICATIONS.
40. Nandiraju, S. K. K., Chundru, S. K., Vangala, S. R., Polam, R. M., Kamarthapu, B., & Kakani, A. B. (2022). Advance of AI-Based Predictive Models for Diagnosis of Alzheimer's Disease (AD) in Healthcare. Journal of Artificial Intelligence and Big Data, 2(1), 141–152.DOI: 10.31586/jaibd.2022.1340