iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68544

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Malware Detection Using Machine Learning Techniques

Shahnawaz Abedin[1], Syed Ali Mehdi[2]
[1]*Student, Department of CSE,Jamia Hamdard*
[2]*Assistant Professor, Department of CSE, Jamia Hamdard*

***Abstract**: Due to the increase in cyber-attacks and the dynamic nature of technology and malware, there is a need to develop a working model capable of detecting malicious files based on certain features.*
The project used the drebin-215-dataset-5560malware-9476-benign.csv dataset, it is the collection of a diverse dataset of both malware and benign samples that include different types of malware. Feature extraction techniques are used to capture relevant attributes from samples, including file system activities, network traffic, and more. Subsequently, a number of machine learning algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Logistic Regression and Convolutional Neural Networks, they are trained and evaluated on the extracted features to classify the samples as malicious or benign.
The evaluation process involves assessing the performance of each algorithm in terms of accuracy, precision, recall and F1 score. In addition, the models are tested for their ability to generalize to unseen data and resist overfitting. A comparative analysis is performed to identify the most effective malware detection algorithm based on the characteristics of the dataset.
The results of this project provide insight into the effectiveness of various machine learning techniques for malware detection and contribute to the development of more robust and proactive cyber security solutions. By leveraging machine learning, organizations can improve their ability to detect and mitigate malware threats in real-time, thereby strengthening the overall security posture of their systems and networks
***Keywords**:Drebin Dataset, Static Analysis, Dynamic Analysis, Malware Detection, Cybersecurity, Anomaly Detection, Random Forest, Logistic Regression*

I. INTRODUCTION

It has now become impossible to imagine a world without the Internet and with the speed of technology, the idea of cyborg seems more real. With the advancement of technology gadgets like mobile phones, laptops, smart watchesare now closely integrated with our lives. Living in a smart city with smart homes, money transactions from ATMs to paying for a candy in shops we are now surrounded with it. A lot of research, time and effort is spent on building these but at the core of all this is data and technology, and it has its flaws and weaknesses. These weaknesses lead a backdoor for the attackers and intruders in our system, which leads to Denial of Service (DOS), Spoofing and Eavesdropping and many more. To protect us from this digital intruders we need a robust protection, to achieve this we need to focus on the way of improving our ways of detecting malware.

In this project, we will detect that malware using the different machine learning techniques, and for that let's have a look on the system share used by the users around the world. There are different OS available in the market like Windows, mac OS, Linux based OS and etc. According to NetMarketShare in Fig. 1, 76.32% of users around the world still uses windows, as its cheap user friendly and does not require high end configurations to install.

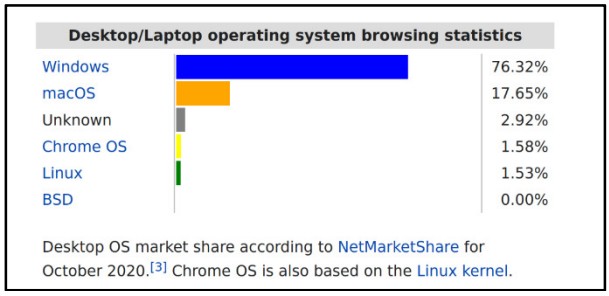


Fig. 1. Net Market Share of OS

Vulnerabilities in the operating system are used to gain access to organizations/people's personal data, putting them at risk and often sold on the dark web or demanding for safe data recovery. A recent example is WannaCry, which spread through the EternalBlue exploit and affected almost all countries.

Current static and dynamic analysis methods cannot effectively detect malware, especially in the case of zero-day attacks or polymorphic malware. This led us to use machine learning to develop better and more efficient models that can detect malicious files and help protect our systems in a better way.

Given the rise of cyber-attacks and the dynamic nature of technology and malware, a business model that can detect malicious files based on certain characteristics needs to be developed. The scope of the model will be limited to Windows executable files.

A. Malware

'Malicious Software' or more commonly known as 'Malware', is a collective name for various malicious software's like Virus, Ransomware, Spyware, Worms, Trojan horses etc. This malware consists of lines of code and once executed, it corrupts and manipulates data and systems based on the behaviour of the created malware. These programs can steal, encrypt, or corrupt data, alter or hijack basic computer functions, and monitor computer operations without the user's consent.

In the past, machines were infected by floppy disks, but as technology has evolved, so has malware. Now the attackers don't need to be there physically to gain access of the system they can do it using a simple Storm Worm Email.

1) Types of Malware

Malware are of many different types and each have their own specific purpose and functionalities. Here we discuss some of them:-

Virus

Virus they're the best malware and require interaction to self-replicate, viruses require some sort of user action like opening an email attachment or visiting a malicious web page to expand. When virus spreads in a system it:

- Can destructs the hardware of the system. Such as corrupting the hard disk.
- Changing the file's size and/or deleting the files.
- Running down background operations and slowing down the system.

Worms

Computer worms are similar to viruses but the only difference is that they are able to execute on their own without any interaction. They are able to spread via networks such as our LAN or Internet connection and replicate themselves in those machines. Worms are of many different types like:-

- E-mail Worms
- File Worms shared in the network
- Internet Worms

Rootkit

Rootkit enables attackers to infiltrate and gain remote and administrative access of a system, it cannot be detected by antimalware software's. Rootkits are extremely difficult to detect and trace as:-

- It can modify active processes to avoid detection.
- It can also hide from antiviruses by changing its signature.

Ransomware

Ransomware prevents users from accessing computers by encrypting all available data, and encryption is only done after the attacker is paid some cryptocurrency to avoid detection by law enforcement.They are able to:-

- Format the whole system in case the ransom is not paid.
- Turn off any anti-virus, firmware, or other security measures that interfere with functionality.

Spyware

This malware is used to spy on the victim's digital activities, personal records, websites visited and actions taken, beliefs, etc. it monitor and collect data to influence the attacker's victim through covert channels. Sometimes this is used by organizations to target consumers for advertising purposes, leading to an increase in spam. It is capable of following functions:-

- Displays unwanted ads and redirects victims to unwanted third-party websites.
- It can cause unwanted changes to the system, resulting in a decrease in security.
- It can also act as a gateway to an open area, allowing hackers to access the system remotely.

Adware

It can also be considered a subset of in-house software whose sole purpose is to serve ads and drive traffic to specific third-party sites. It also ends up using extra RAM and memory which slows down the system.

Trojan

This malware takes its name from the original story from Greek Mythology "The Odyssey by Homer" and "The Aeneid by Virgil" in which the Trojan were defeated by the Greeks in the "Trojan War". In that story the city of Troy enemies' entered in the city using a large wooden horse in which the soldiers were hidden, they pretended it as a gift to the city. When they entered in the city they climbed out and let their soldiers in. Just like that Trojan is a software that seems to be dangerous or a file that contains malicious hidden files and once executed, allows an attacker to gain unauthorized access to the victim's computer. Trojan can do the following to your computer:

- It can steal personal information and details
- It can crash your computer, it can corrupt and make your computer unusable
- It can also open door to install other malwares too
- It can disable your antivirus and can record and send sensitive data to the attacker

Backdoor

As the name suggests, the attacker creates a hidden path to gain unauthorized access to the victim's computer in the open. It is completely harmless by itself, but it provides an environment for various exploits and can also act as a zombie bot used for DDoS attacks.

Keylogger

Keylogger or in short we can say it as "keystroke logger" it is a type of software or sometimes a hardware that is used to capture and record the keystrokes we type using our input device keyboard on the computer. It memorises all the password, credit card information, personal details we have typed on different webpages. To protect yourselves from the keylogger attack we should always:-

- Log out of all passwords that lead to the breach of all confidential information leading to the possibility of blackmail or theft by the user.
- One of the recommended ways to protect yourself from that is to use the on-screen keyboard when entering your credentials.

Remote Access Tool

This tool also called a remote access Trojan, it helps the attacker gain privileges and allows the attacker to gain remote access to the system. Attackers can perform the following unauthorized functions remotely:

- Gaining access to your private data.
- The attacker can use your webcam and even takes all your conversations through the microphone.
- Block your keyboard or even shutdown or make your system useless.
- Install other malware or use it during DDoS attacks.

2) Types of Malware Analysis

Before we can detect malware, we need to understand malware behaviour. Analysis helps monitor malware behaviour and functionality. Each analysis has a different way of achieving results, but the time and knowledge required for each is very different. The types of malware analysis are as follows:

- *Static Analysis-*
  This is achieved by identifying potential behaviours and attributes of malware without executing the code. It is also known as code analysis or "static analysis".

Types of Static Analysis are:-

- Pattern Based Static Analysis-
  It scans the code for pattern that is responsible for the errors.
- Flow Based Static Analysis-
  Examines the program's control flow and data flow to identify potential issue.
- Metric Based Static Analysis
  It uses quantitative measures to find code that may be complex or hard to maintain.
- Semantic Analysis
  In this we understand the meaning of the code to identify deeper, more subtle errors.

- Dynamic Analysis
  Dynamic analysis, also known as behavioural analysis, it refers to the process of examining malware in real time. The file is executed in a virtual environment such as dropbox or virtual machine and all operations are monitored and a complete analysis of all changes and operations performed by the file is performed. This type of analysis is faster than static analysis.
- Hybrid Analysis
  As the name suggests, it combines the static and dynamic analysis to better provide insights of the malware. If any change happens in the computer's memory done by any malware the dynamic analysis can detect that activity. After that the static analysis determines the exact changes that were made.

3) Types of Malware Detection

Malware detection methods are implemented for malware detection and prevent them from compromising a system. They are categorized as:-

- Signature Based Detection
  This static method involves the use of predefined signatures to identify the correct malware. Some signatures contain cryptographic hashes such as SHA1 or MD5, file metadata, and static strings. In this type, if the signature matches an existing malware signature, the file is immediately marked as infected. These techniques are useful for specific malware, but they continue to change hands as polymorphic malware emerges.
- Behaviour Detection
  Also known as heuristics, this involves observing the execution of files and suggesting that something suspicious is malicious. Changing header files or registry keys may seem harmless, but a combination of such actions is definitely cause for concern, and files that exceed a certain threshold will trigger an alert. The best way to do this is through a virtual environment. Although it takes more time, it is a safer and full proof option compared to signature-based detection.
- Feature Detection
  It can be seen as an application of derivative based detection and is able to overcome the false alarms associated with behaviour detection. This method is similar to anomaly detection, but instead of using an ML algorithm, it uses hand-crafted features.
- Blocklisting: A blocklist indicates certain things that are not allowed in a system or in a network. They are used to block any file extensions or known malware from getting installed on a computer.
- Allowlisting: An allowlist specifies the things that are permitted on a system, and everything not on the allowlist is blocked. An allowlist might be used for malware detection to specify permitted files on a system, and all other programs are assumed to be malicious.
- Honeypots: Honeypots are systems that are designed to look like enticing targets to an attacker or a piece of malware. If they are infected by the malware, security professionals can study it and design defences against it for their real systems.

4) Malware Detection Technologies

To detect malware effectively there are lots of techniques, the following are some of the techniques that are used widely, and these are the tools:

- Intrusion Detection System (IDS):This is a security tool that identifies potential malware threats entering or installed within the environment. An IDS is capable of creating alerts for threats when they are detected and allowing an operator to analyse it further.
- Intrusion Prevention System (IPS): An IPS is more proactive in the sense that it prevents attacks against an organization. In addition to notifying management whenever a threat is found, it actively denies the attack from reaching its target system.
- Sandboxing:This method is performing dynamic analysis of malware in a controlled and isolated environment. The activity of malware is conducted in a secure environment called a "sandbox," where malware is analysed by monitoring tools that are designed to evaluate the malice of malware and characterize its capabilities.
- Malware Analysis Tools: There are a good number of available tools for implementing the various malware detection techniques introduced previously. For example, some of the more common disassemblers used for static analysis are IDA, while debuggers are the main debugging tools.
- Cloud-Based Solutions: Utilizing cloud services allows organizations to improve their malware-detecting capability more than what can be achieved in-house. This solution can push out Indicators of Compromise (IOCs) to users and perform sandbox analysis to reach the scale of potential malware.

B. Machine Learning Algorithms

Following machine learning algorithms have been used for this project-

- Decision Tree
  It is a simple and basic diagram that shows different choices and their possible results which helps us to make decisions. It has a hierarchical tree structure the top has one main question called a node it further goes to down like branches which shows different possible outcomes.
  Where:
  - The starting point that represents the entire dataset is called the Root Node
  - The lines that connect nodes are called Branches. These branches display the development from one decision to another.
  - Internal nodes are points where decisions are made based on the input features.
  - The terminal nodes at the end of branches that signify final results or predictions are the Leaf Nodes.
- Support Vector Machine (Linear)
  It uses a linear decision boundary to separate the data points of different classes. This is very suitable when the data are precisely linearly separated. It implies that a single straight line that is in 2-D or a hyperplane (in higher dimensions) can entirely divide the data points into separate classes.
- Support Vector Machine(Polynomial)
  This SVM Polynomial kernel allows for more complex decision boundaries by including polynomial features to the data. It can capture interactions between features up to a certain degree.
- Support Vector Machine (Radial Basis Function – RBF)
  The RBF kernel, also known as the Gaussian kernel, due to its flexible nature it is a popular choice. This kernel can handle very complex and non-linear relationships due to this ability it is distinguished from others.
- K - Nearest Neighbour
  This is one of the simplest and most accurate learning algorithms. This algorithm assumes the resemblance between the new data and available cases and assigns the new case into the category that shares the highest similarity to the available categories.
  - All the available data are stored and then it classified into a new data point based on the similarity of the stored data.
  - K-NN algorithm is primarily used classification problems, but it can also be used for regression problems as well.
- Convolutional Neural Network
  It is a type of deep learning algorithm that is designed for object recognition like images and processing tasks such as detection and segmentation.

Key components of CNN:

- Convolutional Layers: In this layer we used convolutional operations to input the data using kernels to find features like textures, edges and patterns that are more complex.

- Pooling Layers: It lower the network's computational complexity and number of parameters in network by down sampling the inputs spatial dimensions.
- Activation Functions: It enables the model to learn more complex relationships in the data.
- Fully Connected Layers: By learning from the previous layers it makes the predictions based on the high-level.

- Random Forest
  In Machine Learning it is a powerful tree learning technique, it is usedto make predictions and then we make a hierarchical tree structure to make prediction.

Key Features of Random Forest:

- Handles Missing Data: During training it automatically handles the missing data which eliminates the need for human imputation.
- Algorithm provides features selection based on their importance in making predictions, it also offers valuable insights.
- Scales effectivelywith complex and large data without significant degrading performance.

- Logistic Regression
  It is a supervised machine learning algorithm which is used in classification, first it use mathematics to find the instance between two data points and then uses this instance to predict the probability whether it belongs to a certain class or not. Logistic regression is a statistical algorithm which analyse the relationship between two data factors.

Key Points of Logistic Regression:

- It can be either Yes or No, 0 or 1, true or False, etc. instead of giving exact value of 0 and 1, we gets the probability whether the value lies between 0 and 1.
- In Logistic regression, we fit an "S" shaped logistic function, that predicts two maximum values (0 or 1), rather than a regression line.

C. Performance Measures

There are various methods to evaluate the performance of the models. Using these metrics, it provides the quantitative measures to assess the performance of the model and also to compare the model to other models. It help us understand how well our model is performing and whether it is meeting our requirements, this way, we decide if we want to use this model or not. To evaluate the performance of Machine Learning Algorithms, classification as well as regression algorithms there are other various metrics too.
To evaluate the performance measure of the classification model for a dataset where the output can be of two or more type of classes, the confusion matrix table is used.

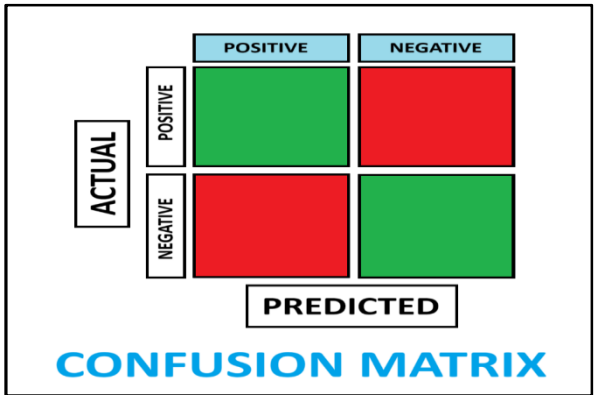


Fig. 2. Figure of Confusion Matrix

The table shown above in fig 2 is known as the confusion matrix and has four sections. The two sections in the green are the True Positive and True Negative and these are the observations which are correctly predicted. The other two sections are in red because these values are wrongly predicted and thus need to be minimized. These sections are false negative and False Positive respectively and occur when there is a contradiction between actual class and the predicted class.

- True Positives (TP)
  These are the values which are correctly predicted, and the output is right. It is denoted by TP.
- True Negatives (TN)
  These are the values which are wrongly predicted, andthe output is wrong. It is denoted by TN.
- False Positives (FP)
  These are the values which arecorrectly predicted but the output is actually wrong. It is denoted by FP.
- False Negative (FN)
  These are the values which are wrongly predicted but the output is right. It is denoted by FN.

Accuracy

For classification algorithms accuracy is the most common performance metric, it measures that how well the model is predicting the outcomes. The calculations are done by taking the number of predictions and divided it by the total number of predictions. We can use the following formulato calculate accuracy–

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

D. Classification Report

This report shows the definition and formulas of Precisions, Recall, and F1 Score. They are as follows–

Precision

It tells us what proportion of predicted positive instances were actually correct. It is calculated as the number of true positive divided by the sum of true positive and false positive.

Formula for calculating precision–

$$\textbf{Precision} = \frac{TP}{TP + FP} \text{ or } \frac{\text{True Positive}}{\text{Predictive Results}}$$

Recall

It calculates the percentage of true positive instances out of all actual positive instances. It is calculated as the number of true positive divided by the sum of true positive and false negative

Formula for calculating recall –

$$\textbf{Recall} = \frac{TP}{TP + FN} \text{ or } \frac{\text{True Positive}}{\text{Actual Results}}$$

F1 Score

It is a performance metric which measures the weighted average of precision and recall, and it is a balanced metric that considers both.

We can use the following formula to calculate F1-Score–

$$\textbf{F - score} = \frac{2 * Recall * Precision}{Recall + Precision}$$

II. LITERATURE REVIEW

With malware attacks on the rise, and with developing strategies for performing cyber-crime these days, never has it been clearer that the other software and techniques used to detect them are struggling for survival, especially against advanced threats like polymorphic and zero-day malware (Anderson et al., 2018)[14]. Due to this limitations people started using machine learning techniques, rather than using statistical modelling methods, pattern recognition, and even basic supervision through machine learning to identify malware amounts (Kolter & Maloof, 2006)[15].

Machine learning has brought a revolution in detecting malware. By looking at multiple big sets of data with different malware samples, ML models can learn how to tell the difference between malicious software and that which is benign (Shafiq et al., 2009)[16]. Different supervised and unsupervised learning algorithms with different advantages and disadvantages have been applied.

One of the simplest is K-Nearest Neighbour. It performs malware classification based on the degree of similarity it has to know threats (Eskandari et al., 2020)17. KNN has proved to be highly accurate, but it may consume too much computational power during higher amounts of data.

Support Vector Machines have been applied in malware detection efforts and have shown certain success, especially in using the kernel functions of linear, polynomial, and radial basis function kernels (Rieck et al., 2008)18. Such variations help SVM perform well on both simple and somewhat complex datasets.

Despite being one of the basic classification algorithms, Logistic Regression is an efficient and optimal choice when malware patterns fit a linear model (Saxe & Berlin, 2015)19.

Decision trees are more interpretable so that security analysts can understand how decisions are made for malware detection. On the other hand, Random Forests combine Decision Trees, which help to improve accuracy and tackle the issue of overfitting (Saxe & Berlin, 2015)19.

Deep learning models such as Convolutional Neural Networks have raised the bar for malware detection; these models can learn complex patterns from data automatically, making them especially competent at detecting advanced and dynamic malware threats.

A. Objective

The growth of malware makes it a serious challenge to cybersecurity, whether this involves signature-based approaches that are failing to mitigate advanced polymorphic and zero-day malware attacks. Static analysis fails because of dynamic behaviour while dynamic analysis is too expensive with regard to computational load. Scalability, real-time capability, and adaptability to new threats are the limitations of current systems.This project aims to address these limitations through the application of machine learning techniques, producing an adaptive, scalable, and intelligent malware detection system. By analysing malware behaviour and generalizing to new threats, the system aims to enhance real-time detection and strengthen cybersecurity defences.

III.METHODOLOGY

A. Data Collection

Datasetthat is used in this research:

Drebin Dataset: drebin-215-dataset-5560malware-9476-benign.csv includes 15,036 samples of Android applications, in which there are 5,560 malware samples and 9,476 benign samplesindicating it as malware (1) or benign (0).

- Features of the dataset: 215 static features, which is categorized into:
  - Permissions (e.g., SEND_SMS, READ_PHONE_STATE)
  - API Calls (e.g., Runtime.exec, getBinder)
  - Network Activity (e.g., HttpPost.init)
  - Hardware Components (e.g., CAMERA, NFC)
- Class Distribution: Imbalanced dataset which includes 37% malware and 63% benign. Visualized using a count plot to highlight dataset imbalance.

B. Data Pre-processing

1) Missing Value Handling:
- Column Removal: Features with >20% missing values were discarded.
- Imputation: Remaining missing values were filled using the median strategy to preserve robustness against outliers.

2) Feature Scaling:
- Applied StandardScaler to normalize features for scale-sensitive models (SVM, KNN, and Logistic Regression).
- Tree-based models (Decision Tree, Random Forest) retained raw feature values.

3) Label Encoding:
- Binary labels: 0 (benign), 1 (malware).

4) Train-Test Split:
- Stratified 80-20 split to maintain class distribution integrity.

5) Exploratory Data Analysis (EDA):

- Class Distribution: Visualized in Fig 3. using a count plot to highlight dataset imbalance.

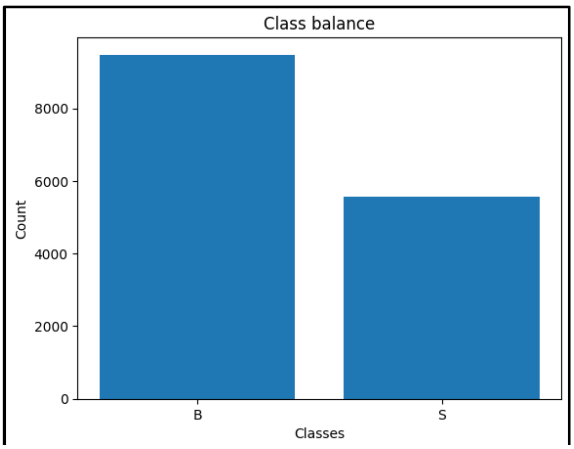


Fig. 3. Class Distribution

- Feature Correlation: Analysed pairwise correlations among the first 20 features via a graph shown in Fig 4.

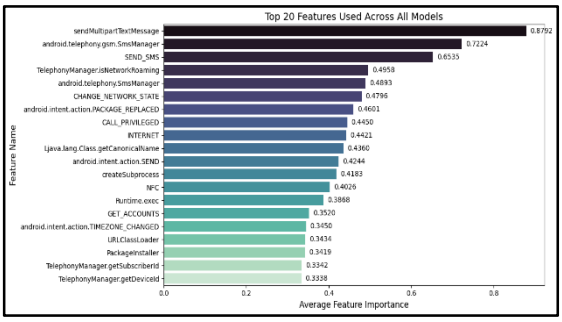


Fig. 4. Feature Correlation

C. Machine Learning Models

The study used the following machine learning models to classify the dataset as malware or benign:

1) Decision Tree (DT):

- A tree-based model trained to split data based on feature recursively to classify malware. We use a max depth of 5 to prevent overfitting
- Gini impurity, max_length=5, min_samples_split=2

2) Random Forest(RF):

- An ensemble of decision trees that improves accuracy by averaging multiple predictions. The model is trained with 100 estimators.

3) Support Vector Machine (SVM):

- Used a kernel-based approach like linear, polynomial and radial basis function (RBF) to find an optimal hyperplane for separating the two classes in a high-dimensional feature space Kernels: Linear (C=1.0), RBF (γ='scale'), Polynomial (degree=3).

4) Logistic Regression (LR):

- A probabilistic model using sigmoid activation function to predict the likelihood of a sample belonging to the malware class.
- L2 regularization, solver='lbfgs', max_iter=1000

5) K-Nearest Neighbour (KNN):

- A distance-based classification approach using k=5 neighbours to identify malware instances.

6) Convolutional Neural Networks (CNN):

- A deep learning model adapted for feature extraction. The architecture consists of:-

- Conv1D layers (32 and 64 filters) for spatial feature extraction.
- MaxPooling layers to downsample representations.
- Dense layers (128 neurons) with ReLU activation.
- Dropout (50%) to prevent overfitting.
- Sigmoid activation for binary classification.
- Adam optimizer with a learning rate of 0.001.

D. Model Architecture

Training Protocol:

1) For classical Machine Learning Models:
- Used 'pandas' and 'NumPy' for data manipulation.
- Used 'scikit-learn' for machine learning models and pre-processing.
- Used "Hyperparameters" for Decision Tree.

2) For Convolutional Neural Network (CNN):
- Used "TensorFlow" and "Keras" for neural network implementation.
- Used "Adam" as optimizer.
- Used Binary Cross-Entropy for Loss Function.

3) Training Validation:
- Model performance is evaluated by splitting the dataset into the training and testing dataset.
- Used 15 epochs, batch size=32, 10& validation split in CNN.
- Dropout = 0.5 to mitigate overfitting in CNN

4) Performance Metrics:

Evaluated models using metrics such as accuracy, precision, recall, F1-score, and confusion matrices.

a) Performance metrics for Decision Tree (Fig. 5.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

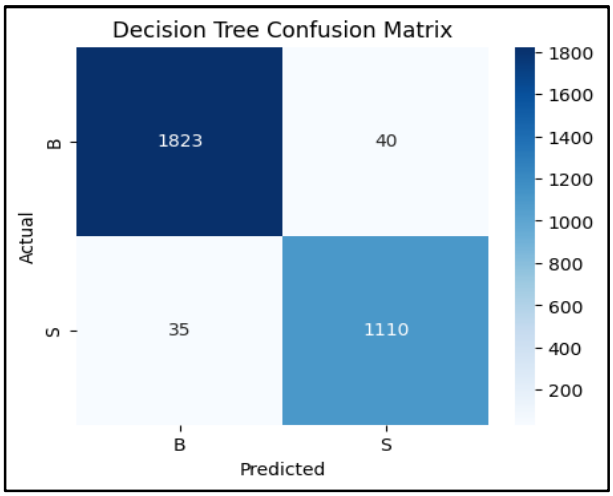


Fig. 5. Confusion Matrix of Decision Tree

	Precision	Recall	F1-Score	Support
B(Benign)	0.98	0.98	0.98	1863
S(Malware)	0.97	0.97	0.97	1145
Accuracy	0.97	0.98	0.98	3008

Decision Tree – Confusion Matrix Analysis:

Actual B, Predicted B:	1823
Actual B, Predicted S:	40
Actual S, Predicted B	35
Actual S, Predicted S:	1110
True Positives (TP):	1110
True Negatives (TN):	1823
False Positives (FP	40
False Negatives (FN):	35
Precision (from CM):	0.9652173913043478
Recall (from CM):	0.9694323144104804
F1-score (from CM):	0.9673202614379085

b) Performance Metrics for Random Forest(Fig. 6.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

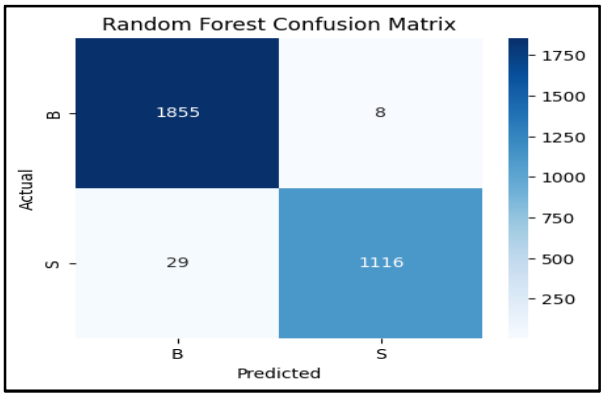


Fig. 6. Confusion Matrix of Random Forest

	Precision	Recall	F1-Score	Support
B(Benign)	0.98	1.00	0.99	1863
S(Malware)	0.99	0.97	0.98	1145
Accuracy	0.99	0.99	0.99	3008

Random Forest - Confusion Matrix Analysis

Actual B, Predicted B:	1855
Actual B, Predicted S:	8
Actual S, Predicted B	29
Actual S, Predicted S:	1116
True Positives (TP):	1116
True Negatives (TN):	1855
False Positives (FP	8
False Negatives (FN):	29
Precision (from CM):	0.9928825622775801
Recall (from CM):	0.9746724890829694
F1-score (from CM):	0.9836932569413839

c) Performance Metrics for SVM (Linear)(Fig. 7.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

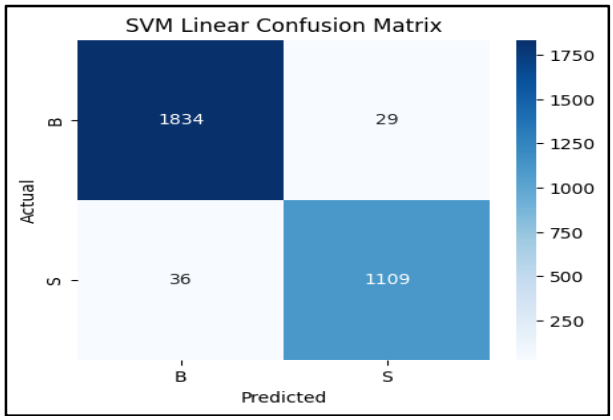


Fig. 7. Confusion Matrix of SVM Linear

	Precision	Recall	F1-Score	Support
B(Benign)	0.98	0.98	0.98	1863
S(Malware)	0.97	0.97	0.97	1145
Accuracy	0.98	0.98	0.98	3008

SVM Linear - Confusion Matrix Analysis:

Actual B, Predicted B:	1834
Actual B, Predicted S:	29
Actual S, Predicted B	36
Actual S, Predicted S:	1109
True Positives (TP):	1109
True Negatives (TN):	1834
False Positives (FP	29
False Negatives (FN):	36
Precision (from CM):	0.9745166959578208
Recall (from CM):	0.9685589519650655
F1-score (from CM):	0.9715286903197548

d) Performance Matrix for SVM (Polynomial)(Fig. 8.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

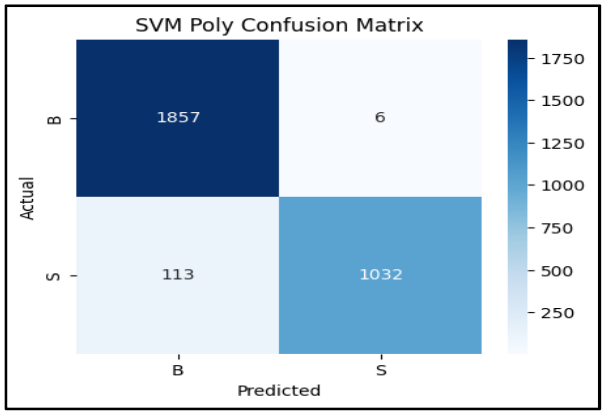


Fig. 8. Confusion Matrix of SVM Polynomial

	Precision	Recall	F1-Score	Support
B	0.94	1.00	0.97	1863
S	0.99	0.90	0.95	1145
Accuracy	0.97	0.95	0.96	3008

SVM Polynomial - Confusion Matrix Analysis:

Actual B, Predicted B:	1857
Actual B, Predicted S:	6
Actual S, Predicted B	113
Actual S, Predicted S:	1032
True Positives (TP):	1032
True Negatives (TN):	1857
False Positives (FP	6
False Negatives (FN):	113
Precision (from CM):	0.9942196531791907
Recall (from CM):	0.9013100436681223
F1-score (from CM):	0.945487860742098

e) Performance Matrix for SVM (RBF)(Fig. 9.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

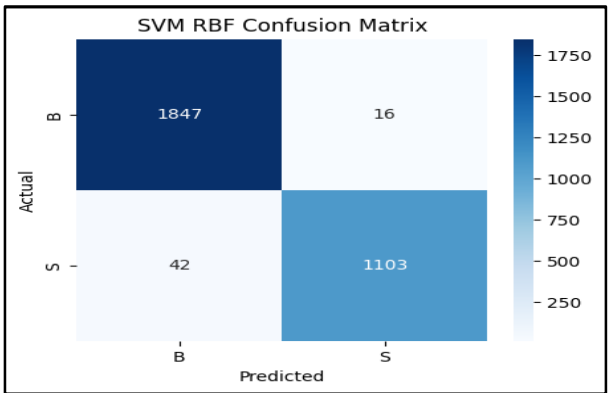


Fig. 9. Confusion Matrix of SVM RBF

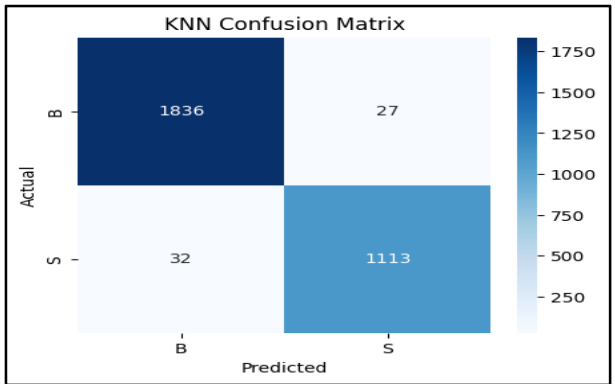
	Precision	Recall	F1-Score	Support
B(Benign)	0.98	0.99	0.98	1863
S(Malware)	0.99	0.96	0.97	1145
Accuracy	0.98	0.98	0.98	3008

SVM RBF - Confusion Matrix Analysis:

Actual B, Predicted B:	1847
Actual B, Predicted S:	16
Actual S, Predicted B	42
Actual S, Predicted S:	1103
True Positives (TP):	1103
True Negatives (TN):	1847
False Positives (FP	16
False Negatives (FN):	42
Precision (from CM):	0.9857015192135835
Recall (from CM):	0.9633187772925764
F1-score (from CM):	0.9743816254416962

f) Performance Matrix for KNN(Fig. 10.)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)



	Precision	Recall	F1-Score	Support
B(Benign)	0.98	0.99	0.98	1863
S(Malware)	0.99	0.96	0.97	1145
Accuracy	0.98	0.98	0.98	3008

KNN - Confusion Matrix Analysis:

Actual B, Predicted B:	1836
Actual B, Predicted S:	27
Actual S, Predicted B	32
Actual S, Predicted S:	1113
True Positives (TP):	1113
True Negatives (TN):	1836
False Positives (FP	27
False Negatives (FN):	32
Precision (from CM):	0.9763157894736842
Recall (from CM):	0.9720524017467249
F1-score (from CM):	0.9741794310722101

g) Performance Matrix for Logistic Regression (Fig. 11)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

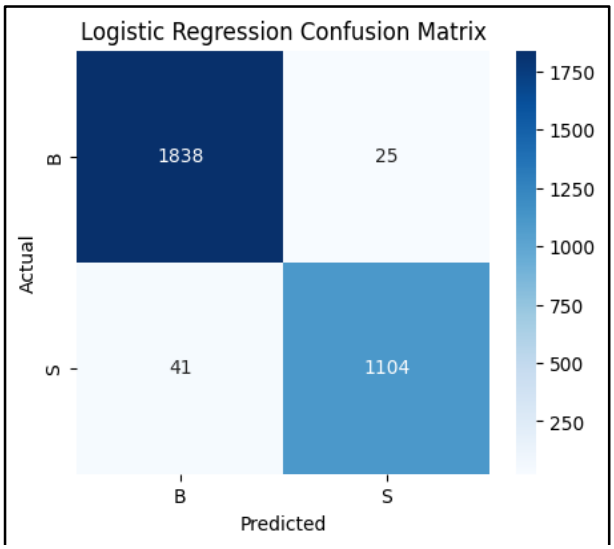


Fig. 11. Confusion Matrix of Logistic Regression

	Precision	Recall	F1-Score	Support
B(Benign)	0.98	0.99	0.98	1863
S(Malware)	0.98	0.96	0.97	1145
Accuracy	0.98	0.98	0.98	3008

Logistic Regression - Confusion Matrix Analysis

Actual B, Predicted B:	1838
Actual B, Predicted S:	25
Actual S, Predicted B	41
Actual S, Predicted S:	1104
True Positives (TP):	1104
True Negatives (TN):	1838
False Positives (FP	25
False Negatives (FN):	41
Precision (from CM):	0.9778565101860053
Recall (from CM):	0.9641921397379912
F1-score (from CM):	0.9709762532981531

h) Performance Matrix for Convolutional Neural Network(CNN) (Fig. 12)

Test Samples: 3,008 (1,145 Malware, 1,863 Benign)

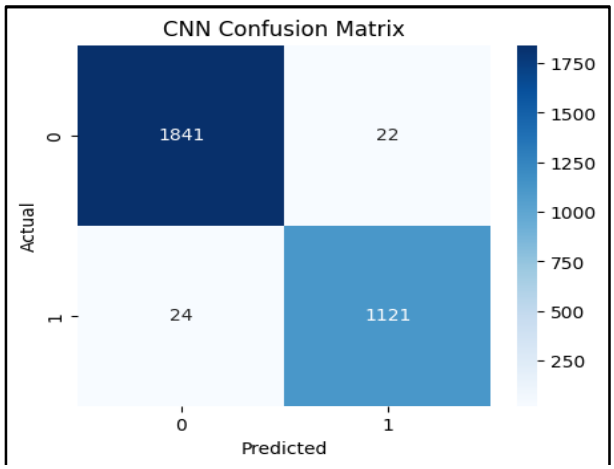
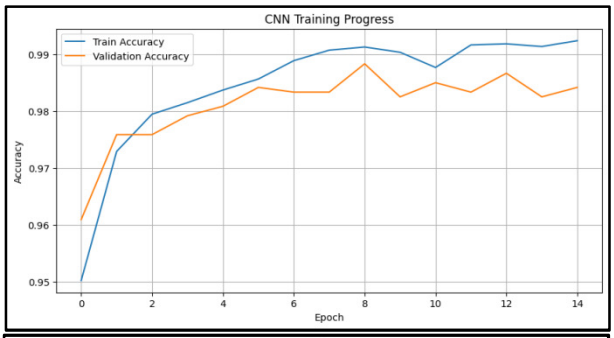


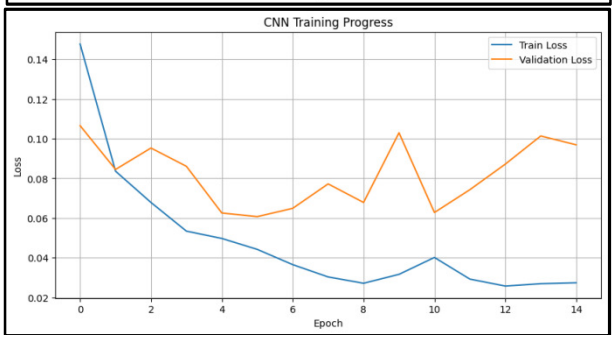
Fig. 12. Confusion Matrix of CNN

	Precision	Recall	F1-Score	Support
0(Benign)	0.99	0.99	0.99	1863
1(Malware)	0.98	0.98	0.98	1145
Accuracy	0.98	0.98	0.98	3008

CNN - Confusion Matrix Analysis:

True Positives (TP):	1114
True Negatives (TN):	1845
False Positives (FP	18
False Negatives (FN):	41
Precision (from CM):	0.9841
Recall (from CM):	0.9729
F1-score (from CM):	0.9785





Performance Metrics Summary

Model	Accuracy	Precision	Recall	F1-Score
SVM Poly	0.960439	0.994220	0.901310	0.945488
Decision Tree	0.975066	0.965217	0.969432	0.967320
Logistic Regression	0.978059	0.977857	0.964192	0.970976
SVM Linear	0.978391	0.974517	0.968559	0.971529
KNN	0.980386	0.976316	0.972052	0.974179
SVM RBF	0.980718	0.985702	0.963319	0.974382
CNN	0.984707	0.980752	0.979039	0.979895
Random Forest	0.987699	0.992883	0.974672	0.983693

IV. SUMMARY

This methodology outlines a comprehensive approach to detect malware using multiple machine learning models. By comparing traditional and neural network approaches, the study aims to identify the most effective technique for detecting malicious applications.

A. Results

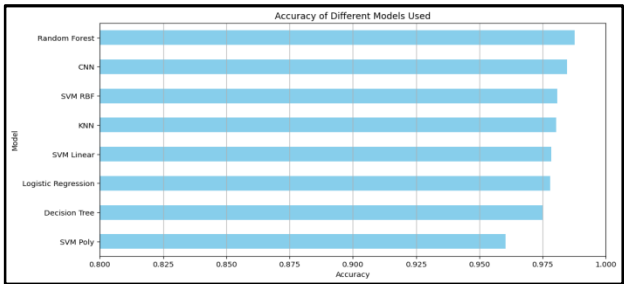


Fig. 12. Accuracy of Different Models Used

- The implemented machine learning algorithms—Decision Tree, Random Forest, Support Vector Machine (SVM), KNN, Logistic Regression and Convolutional Neural Networks all these were evaluated using accuracy, precision, recall, and F1-score and their accuracy chart has been put in the Fig. 12.
- The models effectively differentiated between benign and malicious software using the extracted features from the dataset.

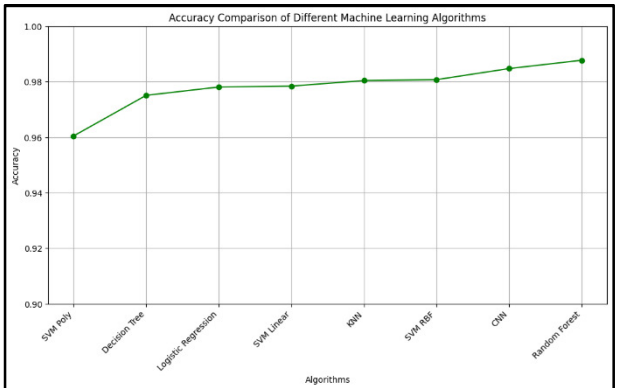


Fig. 13. Accuracy Comparison

- Among these, the Random Forest algorithm achieved an accuracy of 98%, demonstrating its efficacy for malware detection shown in Fig. 13.

V. CONCLUSION

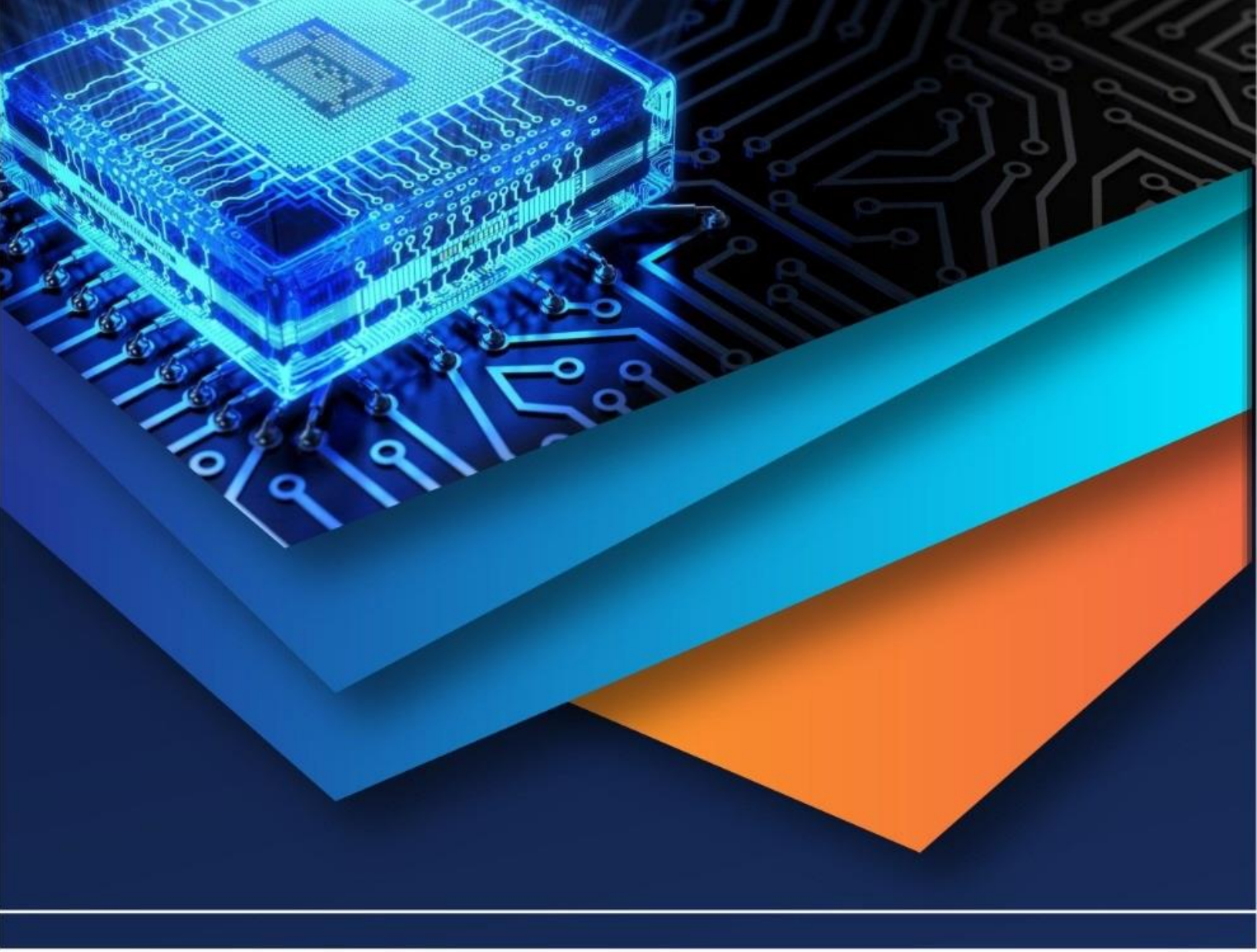
This research shows that for malware detection, machine learning techniques plays an important rolebehavioural and structural attributes of malicious software, the study demonstrates that by using machine learning algorithms we can achieve high accuracy and reliability.

The Random Forest model achieved the highest accuracy (99%) which clearly indicates the suitability for malware detection tasks.Also, the project highlightsthe importance of feature extraction and data pre-processing for model training and evaluation.

This project encourages the need for adaptive and scalable detection mechanisms to combat the evolving malware threats. The insights gained from this study contribute to the development of more robust cybersecurity frameworks capable of real-time threat detection and mitigation.

REFERENCES

[1] https://www.ijert.org/research/an-emerging-malware-analysis-techniques-and-tools-a-comparative-analysis-IJERTV10IS040071.pdf
[2] https://www.researchgate.net/publication/224089748_Malware_detection_using_machine_learning
[3] https://www.mdpi.com/1099-4300/23/8/1009#:~:text=Dynamic%20analysis%20technology%20generally%20analyzes%20the%20characteristics,characteristics%20of%20application%20software%20by%20executing%20programs
[4] https://ijaseit.insightsociety.org/index.php/ijaseit/article/view/6827/pdf_846
[5] https://www.mdpi.com/2073-8994/14/11/2304
[6] https://doi.org/10.1002/cpe.5422
[7] https://arxiv.org/pdf/1606.06897
[8] https://www.ijraset.com/research-paper/malware-detection-using-ml
[9] https://www.ijraset.com/best-journal/malware-detection-using-machine-learning
[10] https://www.ijraset.com/research-paper/a-static-approach-for-malware-analysis-a-guide-to-analysis-tools-and-techniques
[11] https://www.mdpi.com/2073-8994/14/11/2304#B1-symmetry-14-02304
[12] https://www.tutorialspoint.com/machine_learning/machine_learning_performance_metrics.htm
[13] https://spotintelligence.com/2023/03/01/adam-optimizer
[14] https://www.geeksforgeeks.org/
[15] Anderson, H. S., Filar, B., & Kharkar, A. (2018). Evading Machine Learning Malware Detection. Black Hat USA.
[16] Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect malicious executables in the wild. Journal of Machine Learning Research, 7, 2721–2744.
[17] Shafiq, M. Z., Tabish, S. M., Farooq, M., & Mirza, H. (2009). PE-Miner: Mining structural information to detect malicious executables in real time. RAID.
[18] Eskandari, M., Hashemi, S., & Leckie, C. (2020). A fast KNN-based approach for malware detection. Computers & Security, 94, 101877.
[19] Rieck, K., Trinius, P., Willems, C., & Holz, T. (2008). Automatic analysis of malware behavior using machine learning. Journal in Computer Virology, 7(4), 1-15.
[20] Saxe, J., & Berlin, K. (2015). Deep neural network-based malware detection using two-dimensional binary program features. 10th International Conference on Malicious and Unwanted Software.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY