

APPLICATION OF MACHINE LEARNING TECHNIQUES FOR MODERN CYBER THREAT DETECTION

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
DATA SCIENCE

Submitted by

AYUSHMAN SAINI
(23/DSC/19)

Under the supervision of
Mrs. PRIYA SINGH



SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2025

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE’S DECLARATION

I, AYUSHMAN SAINI, Roll No – 23/DSC/19 student of M.Tech (DATA SCIENCE), hereby declare that the Dissertation titled “**APPLICATION OF MACHINE LEARNING TECHNIQUES FOR MODERN CYBER THREAT DETECTION**” which is submitted by me to the SOFTWARE ENGINEERING, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

AYUSHMAN SAINI

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor

Signature of External Examiner

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “**APPLICATION OF MACHINE LEARNING TECHNIQUES FOR MODERN CYBER THREAT DETECTION**” which is submitted by AYUSHMAN SAINI, Roll No – 23/DSC/19, SOFTWARE ENGINEERING ,Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology in Data Science, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Mrs. PRIYA SINGH

Date: 19.05.2015

SUPERVISOR

DEPARTMENT OF SOFTWARE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I express my sincere gratitude to Ms. PRIYA SINGH for the continuous guidance and mentorship that she provided us during the project. She showed us the path to achieve my goals by explaining all the tasks to be done and explained to us the importance of this project as well as its industrial relevance. She was always ready to help us and clear our doubts about any obstacles in this project. Without her constant support and motivation, this project would not have been successful.

Place: Delhi

AYUSHMAN SAINI

Date: 19.05.2025

Abstract

The increasing sophistication of modern malware has rendered traditional signature-based detection techniques less effective, especially against polymorphic and zero-day threats. In response, machine learning (ML) and deep learning (DL) methods have emerged as viable alternatives for identifying malicious behaviour through pattern recognition. This thesis investigates the application of ML techniques in cyber threat detection through two key components: a critical review of recent literature, and the implementation of an improved multiclass malware classification system. The review analyzes twenty contemporary research papers, comparing approaches based on datasets, analysis techniques, model choices, and detection performance. Key limitations identified include class imbalance, feature redundancy, and challenges in handling complex or obfuscated malware. Building on these insights, a hybrid ensemble model is developed combining tree-based classifiers and a neural network meta-learner. The system incorporates advanced preprocessing techniques such as mutual information-based feature selection and SMOTE-based oversampling to improve learning from imbalanced data. The model is trained and evaluated on the CIC MalMem2022 dataset, covering fifteen malware families and benign samples. Results show improved classification performance, particularly for underrepresented classes, with notable gains in macro-averaged precision and recall. This work demonstrates the potential of integrated ML pipelines for practical malware detection and suggests further exploration into explainable models, adaptive learning, and cross-platform generalization for future research.

Contents

Candidate's Declaration	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Content	vi
List of Tables	vii
List of Figures	viii
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Statement	1
1.3 Overview	1
1.4 Objectives	1
2 LITERATURE REVIEW	3
2.1 Dataset Analysis	3
2.2 Techniques used in Studies	4
2.3 Reviewed Studies Summaries	5
2.3.1 Selection Criteria	5
2.4 Performance Analysis	8
2.5 Summarised Findings	9
3 METHODOLOGY	10
3.1 An Overview of Proposed Model	10
3.2 Architecture of Proposed Model	11
3.2.1 Model Overview	11
3.2.2 Base Classifiers	13
3.2.3 Meta Classifier (Neural Network)	14
3.2.4 Feature Engineering	15
3.3 Dataset Description	16
3.4 Parameter Settings	16
3.5 Performance Metrics	17
4 Results and Discussion	20
4.1 Individual Classifier Evaluation	20

4.2	Ensemble Model Performance	20
4.3	Preprocessing and Feature Engineering Impact	21
4.4	Detailed Performance Analysis	21
4.4.1	Class-Specific Detection Capabilities	21
4.4.2	Confusion Matrix Analysis	22
4.4.3	Feature Importance and Selection Impact	22
4.4.4	Training Convergence and Model Stability	22
4.4.5	Comparative Performance Evaluation	22
4.5	Computational Efficiency and Scalability	24
4.6	Statistical Significance and Robustness	24
5	CONCLUSION AND FUTURE SCOPE	26
5.1	Future Work	26
A	Publication Details	28
A.1	List of Publications	28
A.2	Paper Acceptance Proof	29
A.3	Indexing of Conference Proof	31
A.4	Conference Paper Registration Receipt	32

List of Tables

2.1	Datasets used in studies which are available to public	4
2.2	Category wise description of different methods used in various studies . . .	6
2.3	Reviewed studies strengths and weaknesses	7
2.4	Results summary of the studies	9
3.1	Distribution of Samples by Malware Type and Family	18
4.1	Summary of results performed	25

List of Figures

2.1	Search Strategy	8
3.1	Architecture of the Proposed model	12
3.2	t-SNE visualisation of Malware types to observe how are different categories are different from each other	16
3.3	Frequency chart of Malware parent classes	17
3.4	Pie chart showing percentage distribution of Malware parent classes	18
3.5	Pie chart showing percentage distribution of Malware categories classes . .	19
4.1	Top Detected Malware Classes by ROC AUC	22
4.2	Confusion matrix of the malware categories of the of our model after per- forming MI+SMOTE+Scaling	23
4.3	Top Feature by Mutual Information	23
4.4	Meta model accuracy and model loss	24
A.1	First Conference Acceptance Proof	29
A.2	Second Conference Acceptance Proof	30
A.3	THE 16th INTERNATIONAL IEEE CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT)	31
A.4	Fee Receipt for the paper titled, A Comprehensive Review on Malware Detection Techniques using Machine Learning and Deep Learning	32
A.5	Fee Receipt for the paper titled, Enhancing Obfuscated Malware Classifi- cation: A Performance-Driven Study Using Stacked Ensemble Learning . .	33

Chapter 1

INTRODUCTION

1.1 Motivation

The increased use of digital technology by people, companies and government bodies means cybersecurity measures must be greatly improved. Malware is a recurring threat that often causes severe damage. Advanced tactics such as polymorphism, packing and obfuscation used by attackers mean that conventional detection methods are not effective anymore. Since malware keeps evolving, having automatic tools in place that can identify all types of threats is very important. Therefore, companies are using ML (Machine Learning) and DL (Deep Learning) to find and classify malware.

1.2 Problem Statement

Currently, most malware detection tools operate based on signatures and rules, so they are not always able to find new or changed malware. Often, these defenses don't find threats that aren't recognized such as zero-day attacks. Identifying malware based on similarities is still difficult because there are so many varieties with similar characteristics and not much labeled information. The thesis focuses on designing and testing a machine learning model that accurately identifies different classes of malware based on their behavior, even when dealing with class imbalance and many different features.

1.3 Overview

This thesis starts with a review of the most recent findings in using machine learning to detect malware, highlighting the use of static, dynamic and hybrid approaches. The intelligence collected is then used to design an experimental framework using the CIC MalMem2022 containing execution behavior features for many malware types. Several ML models are set up and tested and a method for combining their predictions is presented to increase accuracy in classification. To fix problems with noisy features and unequal class numbers, feature engineering and data balancing are applied.

1.4 Objectives

The main objectives of this thesis are as follows:

- To investigate and compile the current findings in malware detection based on ML and DL methods.
- To implement a multiclass classification framework that can effectively identify and distinguish between different malware families.
- To assess how performing feature selection and balancing classes changes model performance.
- To suggest and test a model that uses a combination of various classifiers.
- To go over the field's results using different measurements and advise on what can be done better going forward.

Chapter 2

LITERATURE REVIEW

The development of machine learning-based malware detection techniques has attracted considerable attention in recent years due to the increasing complexity and volume of cyber threats. This section presents a critical review of existing research focused on the use of supervised and deep learning models for identifying and classifying malware. It covers the types of datasets used, the feature extraction strategies adopted, the classification methods employed, and the reported performance metrics. By analyzing the strengths and limitations of these approaches, this review helps identify key research gaps and provides a foundation for designing more effective malware detection systems.

2.1 Dataset Analysis

The outcomes of machine learning methods for detecting malware strongly rely on the quality and quantity of the data supplied. A clearly arranged dataset enables models to find patterns in benign and malicious activity and use them to address threats that have not been seen before. The literature we reviewed shows that datasets can be very different in how they are structured, where they come from, how easy they are to find and which types of features they include. The type of malware they identify depends on whether they use executable files, logs, network data or images as their starting point. Typically, we divide malware datasets into those that the public may use and those designed for private companies. Using public datasets gives researchers a standard way to validate findings, duplicate results and make fair comparisons with others. Important examples are the CIC MalMem2022 dataset, the Maling image-based dataset, VirusShare and VirusTotal. In these datasets, there are commonly marked samples of regular files and different types of malware, along with occasional data on their behavior or PE characteristics. This dataset includes memory-based behaviors of 15 malware families as well as benign samples, so it is specifically useful for multiclass classification challenges. On a different note, a lot of studies create custom datasets using sandbox settings, repositories of malware or properly managed testbeds. Their main downsides are often the absence of diversity, a lack of scalability and the frequent requirement of entering the system to see these artifacts. Besides, because the ways data is labeled, collected and processed through feature extraction are not standardized, it becomes difficult to compare these studies. A summary of publicly available datasets used in the reviewed papers is provided in Table 2.1. The table shares information on sample size, platform and kind of data, helping see how various in benchmark resources vary and what their limits are. Such datasets have a major effect on shaping how well machine learning-based detectors perform.

Table 2.1: Datasets used in studies which are available to public

Dataset Name	Paper References	Dataset Size	Description
VirusShare	[1], [2]	~5,000–10,000 samples	Collection of known malware binaries; widely used for static analysis tasks.
VirusTotal Logs	[3], [4]	50,000+ entries	API-based behavioral logs of files scanned through VirusTotal services.
Drebin	[5], [6]	~5,600 malware apps	Android malware dataset with permissions and API-based features.
Microsoft Malware Classification Challenge (BIG 2015)	[7]	~10,000+ binaries	PE files labeled by family; used for image-based CNN classification.
CICIDS/CIC-MalMem2022	[1]	Varies (10K+ records)	Behavioral and memory-based artifacts of malware and benign processes.
Kaggle Malware Repositories	[8], [9]	~60,000 entries (varied)	Mixed public datasets (e.g., PE headers, labels) reused for benchmarking.

2.2 Techniques used in Studies

Machine learning is used to identify malware in several ways, depending on how data is taken, analyzed and modeled. Overall, research approaches can be divided into static analysis, dynamic analysis and hybrid analysis. Every method makes use of different kinds of features and calls for individual handling and training steps. No programs are executed during a static analysis, but programs are scanned for specific things. Some of the things found regularly are opcode sequences, imported API functions, PE header attributes and patterns of binary bytes. Using these mechanisms is efficient and safe because they do not involve running possibly hazardous executable files. The collected studies generally depend on static-feature models that make use of classifiers such as Random Forest (RF), Support Vector Machines (SVM) and Convolutional Neural Networks (CNNs), to study opcode n-grams or binary image changes. By contrast, during dynamic analysis, a test is run on the program to watch its behavior, collecting details about system calls, changes to the registry, networking and memory. Because it follows the behavior of the program, this approach is protected against obfuscation and encryption. Reviewed studies frequently apply dynamic features by using Random Forests, Recurrent Neural Networks (RNNs) or DNNs to learn how behavior evolves. Combining static and dynamic tools gives a more complete picture of the software. The intention of these methods is to combine the benefits of both approaches: static efficiency and dynamic robustness. Examples of hybrid detection systems are ones that add static attribute details with dynamic calls and systems that add network details along with opcode counts. In such cases, popular classifiers are built with ensemble models and deep hybrid systems. The way you select variables is crucial for getting the best outcomes from your analysis. Almost all statis-

tical research applies Pearson correlation or mutual information filters, with just a few using embedded or wrapper methods. In most cases, researchers apply normalization, reduce features and use the SMOTE method to ensure even class distribution and lessen biases. Generally, the technique chosen depends on the type of data, available computing resources, the operating system it will be used on (Windows, Android) and if you need binary or multiclass classification. It is clear from the literature that researchers are focusing more on using ensemble models and complex networks to manage high-dimensional and complicated data from many types of malicious software.

Here Table 2.2 offers a category wise summary of methods/models used in various studies for malware detection.

2.3 Reviewed Studies Summaries

Next, this section gives a brief overview of the main research papers in the review, pointing out what they add, how they work and what were their main constraints. The reviewed works use several machine learning and deep learning techniques for malware detection, together with varied datasets and ways to analyze them. Researchers looked in top journals and gatherings to pick these research papers which convey the most recent developments in both cybersecurity theory and applications. There are studies that rely on using opcode n-grams, details from the PE header and sequences of API calls for analysis. By using opcode frequency distributions and Random Forest and Naive Bayes, [2] obtained accurate results yet pointed out their limitation in fighting malware that has been obfuscated or disguised. Likewise, authors in [7] trained a CNN on grayscale images of binary files, allowing for easier visual recognition but making training very demanding on resources. Many researchers in this area often rely on features linked to application behavior. In one study, authors examined Android’s system calls and applied KNN and RF, earning strong results solely for Android. In a similar study by [3], access to a specialized data set helped build a dynamic signature model that worked perfectly on the test cases. Techniques proposed in [10] and [1] combine details found at compile time with information obtained at runtime. While these approaches better resist attempting to bypass them, they increase the complexity of the needed preprocessing, as seen in study [11] on Control Flow Graphs (CFG). A summary of each paper’s main benefits and pitfalls can be found in the following table 2.3. Doing this synthesis reveals that imbalanced data, overfitting after training on few samples and no clear explanations from deep models are problems that can be addressed.

2.3.1 Selection Criteria

This review was conducted to identify relevant studies published between 2021 and 2024. Only peer-reviewed journal articles, conference papers, and high-quality preprints were considered for inclusion. Literature was collected from established academic sources IEEE Xplore, SpringerLink, ScienceDirect, ACM Digital Library and, Wiley Online Library. A structured keyword-based search was used, applying combinations of terms such as “malware detection using machine learning”, “deep learning for malware detection”, “static and dynamic malware analysis”, and “hybrid malware. Detection”, and “android malware detection” 2.1.

Table 2.2: Category wise description of different methods used in various studies

Category	Method/Model	Description
Static Analysis	Opcode N-gram + RF/SVM/NB	Extracts opcode sequences from binaries; models learn instruction-level patterns.
Static Analysis	PE Header + API Analysis	Uses structural features like import tables, API call lists for classification.
Static Analysis	Grayscale Image + CNN	Converts binaries into images; CNN detects visual patterns of malware.
Static Analysis	Control Flow Graph + Autoencoder	Captures code structure via CFGs; compresses features using autoencoders.
Dynamic Analysis	System Call Sequences + RF	Observes runtime syscalls; RF models behavior frequency and patterns.
Dynamic Analysis	API Call Logs + SGD/Naive Bayes	Tracks API call flow during execution; classifiers identify abnormal usage.
Dynamic Analysis	Network Traffic + DNN	Extracts flow-level features like packet size/timing; DNN models data flows.
Dynamic Analysis	Behavior Clustering + ML	Clusters behavioral signatures; ML distinguishes benign vs. malicious clusters.
Hybrid Analysis	Static + Dynamic Feature Fusion	Merges static code and runtime behavior; improves generalization.
Hybrid Analysis	Wrapper + CNN Ensemble	Uses feature selection to combine CNN outputs with handcrafted inputs.
Hybrid Analysis	Permission + API + CNN/DT	Integrates permission use and APIs for Android; analyzed using ML/DL mix.
Deep Learning	CNN / CNN-RNN	Learns spatial or sequential patterns from binary or behavioral input.
Deep Learning	Autoencoder + Classifier	Reduces feature space unsupervised, followed by supervised classification.
Traditional ML	RF, KNN, SVM, DT	Classical models are applied to structured static/dynamic features.
Ensemble Methods	Multi-stage RF or Hybrid Voting	Combines multiple classifiers to improve robustness and reduce false alarms.
Distributed ML	Hadoop/Spark + ML Classifiers	Extracts features at scale from logs; supports large-scale parallel training.

Table 2.3: Reviewed studies strengths and weaknesses

Paper Reference	Strengths	Weaknesses
[2]	Efficient opcode-based static analysis; good model variety	Limited defense against obfuscated malware
[12]	Real-world dynamic behavior logs; high accuracy	Platform-limited (Android-only)
[10]	Balanced dataset; consistent evaluation; strong performance	Synthetic logs; lacks external dataset validation
[5]	Combined static + dynamic features; variety of models	Dataset not publicly available; lacks detailed feature disclosure
[13]	Integrated ML and DL approaches; hybrid feature engineering	Lacks handling of novel malware or adversarial robustness
[11]	Uses structural control flow (CFG); novel approach	High computational cost; dataset details missing
[14]	Classical PE analysis; interpretable models	Lower performance on modern and complex malware
[15]	Multi-class classification (family/subfamily level)	Evaluation metrics limited; no comparison with DL approaches
[3]	Excellent accuracy on VirusTotal logs; cloud-focused	Practical deployment challenges not addressed
[4]	Evaluates CNN and ensemble classifiers on PE files	Medium-sized dataset; lacks real-time applicability
[6]	Lightweight detection for mobile traffic; privacy-aware	Dataset not public; lacks model explainability
[16]	Real-time syscall-based model; Linux compatible	Limited cross-platform support; no hybrid feature use
[17]	Behavior-driven classification; practical insights	No real-time evaluation; sandbox setup not detailed
[7]	Image-based modeling with CNN; good for malware families	High model complexity; resource-intensive training
[9]	Feature fusion approach; CNN integration	Only static features; no dynamic evaluation
[18]	Distributed architecture for large-scale detection	Sparse details on dataset structure; lacks validation
[19]	Deep learning hybrid ensemble approach	Minimal benchmarking; unclear feature sources
[1]	Hybrid CNN-RNN model; effective feature blending	Performance on zero-day malware not tested
[20]	Simple and reproducible static analysis model	Basic feature set; modest detection accuracy
[8]	Structured system feature logs; clear ML pipeline	Reuses previous dataset; novelty limited



Figure 2.1: Search Strategy

2.4 Performance Analysis

Evaluating the effectiveness of malware detection models requires a detailed assessment of performance metrics across various learning techniques and datasets. The reviewed papers employ a wide range of models—from traditional classifiers like Random Forest (RF) and Support Vector Machine (SVM) to complex deep learning architectures such as Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs). Their performance is typically measured using standard classification metrics such as accuracy, precision, recall, and F1-score. However, these metrics vary considerably depending on the dataset size, feature richness, and malware diversity. Many studies report high accuracy values, especially when using balanced datasets or custom test environments. For instance, [10] achieved a 99% accuracy using a combination of RF and KNN on a large, balanced dataset of 60,000 samples. Similarly, [20] reported perfect classification scores using traditional models on curated dynamic logs. However, in both cases, real-world applicability is constrained by controlled dataset conditions, and results may not generalize to more diverse or noisy data sources. In dynamic and hybrid analyses, classifiers such as DNNs and CNNs showed competitive macro-level performance. For example, [12] achieved over 98% accuracy on a deep learning model trained with network traffic features, while [13] image-based CNN approach also demonstrated strong performance, especially in detecting polymorphic variants. Nevertheless, these models often struggle with underrepresented classes, and their performance tends to favor dominant malware types due to skewed class distributions. To address this, some studies applied class balancing techniques (e.g., SMOTE), feature selection strategies, or ensemble learning frameworks to boost recall and precision for minority classes. Overall, while high classification accuracy is common across studies, macro-averaged performance and class-wise breakdowns reveal significant disparities—especially for multiclass classification tasks involving many malware families. A consolidated summary of models, key results, and comparative advantages and drawbacks is presented in Table 2.4. This highlights the trade-offs between computational complexity, detection performance, and generalizability across platforms and malware types. Such insights inform the design of the improved detection system implemented in this thesis.

Table 2.4: Results summary of the studies

Model/ Technique Used	Reference Papers	Avg. Ac- curacy	Avg. Pre- cision	Avg. Re- call
RF	[1], [12], [5], [3], [16], [8]	97.4	96.3	97.5
KNN	[12], [10], [5]	98.2	97.8	98.0
SVM	[2], [5], [4]	96.1	95.4	96.0
DT	[5], [4]	96.8	96.1	96.4
NB	[2], [3]	94.7	94.2	94.5
CNN	[13], [7], [9], [4]	98.3	97.8	97.9
DNN	[6], [13]	97.6	97.2	97.3
Autoencoder + Clas- sifier	[4], [7]	96.2	95.8	96.0
CNN-RNN Hybrid	[1]	97.2	96.7	96.9

2.5 Summarised Findings

The literature reviewed in this thesis reveals several recurring patterns, strengths, and limitations in current machine learning approaches to malware detection. Most notably, traditional ML classifiers such as Random Forest, Support Vector Machines, and K-Nearest Neighbors remain popular due to their interpretability and ease of training. However, recent trends show an increasing reliance on deep learning methods like CNNs and DNNs, particularly for handling complex or high-dimensional feature spaces. Dataset diversity plays a crucial role in shaping model performance. Studies using public datasets like CIC MalMem2022 or VirusShare achieve high accuracy but often do so under curated conditions with balanced class distributions. On the other hand, experiments conducted on real-world or custom datasets tend to expose challenges such as class imbalance, obfuscated malware variants, and noisy features. These limitations often lead to uneven performance across malware families—especially for minority or novel classes. Feature engineering and preprocessing strategies emerged as critical factors influencing detection outcomes. Approaches that employed mutual information, correlation-based filtering, or image-based feature transformations showed better generalization. Yet, very few studies addressed model explainability or deployed models in real-time settings, indicating a gap between experimental success and practical deployment. Another key insight is the increasing interest in hybrid models that combine static and dynamic analysis. These systems tend to offer improved robustness against evasion techniques, though at the cost of higher computational overhead. In conclusion, the review highlights that while significant progress has been made in applying ML techniques to malware detection, challenges remain in achieving balanced performance, scalability, and interpretability. These observations directly informed the experimental component of this thesis, particularly in the design choices related to feature selection, class balancing, and model architecture.

Chapter 3

METHODOLOGY

The purpose of the design is to sort multiclass malware families using virtual machine observation data from the CIC MalMem2022 dataset. The framework uses multiple base classifiers and a neural network acts as a meta-classifier to bring their predictions together. Before model training, the original dataset is processed by class balancing with SMOTE and by selecting features with mutual information. The purpose is to improve how well both standard and unusual malware are found and identified.

3.1 An Overview of Proposed Model

Attacking software is becoming more difficult to identify because it evolves so fast and becomes harder to track. Although signature-based instruments give excellent results against standard viruses, they miss out on any malware that fits into the obfuscated, polymorphic or zero-day categories. Because of these changes, we need security systems that can respond effectively to unrecognized threats.

In this thesis, **automated multiclass malware classification** is explored through the application of machine learning. Unlike simply telling if a file is safe or dangerous, multiclass classification tries to specify which family of malware a given file belongs to. This detailed identification helps greatly in response to incidents, finding the actors involved and performing forensic investigations.

The study recommends building an ensemble of select machine learning models, with a meta-classifier based on deep learning, to address this problem. The system is built using the CIC MalMem2022 dataset which has labeled information from fifteen different malware families along with benign data [21]. This set of data presents three real-world problems: unequal numbers in different classes, overlap among some features and noticeable variation within each class.

This thesis applied the following main methods:

1. **Data Preprocessing:** This includes cleaning, label transformation, and statistical analysis of features. Unneeded and non-numerical features are excluded and the class labels are adjusted so all of them are consistent. At first, we evaluate features based on their correlation and mutual information.
2. **Feature Scaling and Selection:** Standardization is applied to scale features uniformly. Both Pearson correlation filtering and mutual information concepts are applied to select those features that are most useful for model training.

3. **Class Balancing with SMOTE:** Since the dataset is inherently imbalanced across malware families, Synthetic Minority Oversampling Technique (SMOTE) is applied to generate synthetic instances of underrepresented classes, ensuring more balanced training.
4. **Base Learners:** Three tree-based classifiers—Random Forest, Extra Trees, and XGBoost—are trained independently using stratified cross-validation. Second-level learners use these probabilities to guide their learning.
5. **Meta-Learner Architecture:** A feedforward deep neural network is used as the meta-classifier. The architecture uses layers working in parallel, regularized dropouts and batch normalization to keep the network general for all class forms.
6. **Model Evaluation:** The ensemble system is evaluated using accuracy, precision, recall, F1-score, and confusion matrix metrics. Macro-averaged scores are given special consideration to judge how well a product does against all types of malware and not only against the leaders in this field.

By combining preprocessing strategies with a hybrid ensemble approach, this methodology aims to improve classification performance across all malware families, particularly those that are underrepresented or behaviorally similar to others. The next section describes the proposed system in more technical detail, highlighting the complete architecture and learning pipeline.

3.2 Architecture of Proposed Model

3.2.1 Model Overview

To address the challenges of multiclass malware classification, this thesis proposes a two-tier ensemble architecture that integrates the predictive strengths of multiple machine learning algorithms with the adaptive learning capabilities of a deep neural network metaclassifier. The system is designed to handle high-dimensional feature spaces, class imbalance, and variability in malware behavior by incorporating robust preprocessing and feature engineering techniques. Figure 3.1 shows the architecture of our model. The proposed architecture operates in two primary stages:

1. **Base Learners:**

In the beginning, RF, ET and XGB each carry out three independent machine learning classifications. All the models are exposed to the same feature data and are taught to produce class likelihoods for each input. They were chosen because they have shown strength in dealing with complex features and are not affected by overfitting on structured data. All the models are built using stratified cross-validation to verify they perform well and can be applied to all malware types.

2. **Meta-Learner:**

During the second step, a deep neural network (DNN) is included and takes the role of the meta-classifier. The algorithm receives the sum of each base learner’s predictions and learns how to build a final prediction from them. The DNN has multiple connected layers that use ReLU as activation, batch normalization. DNN make use of dropout to stabilize the training and minimize overfitting. This final

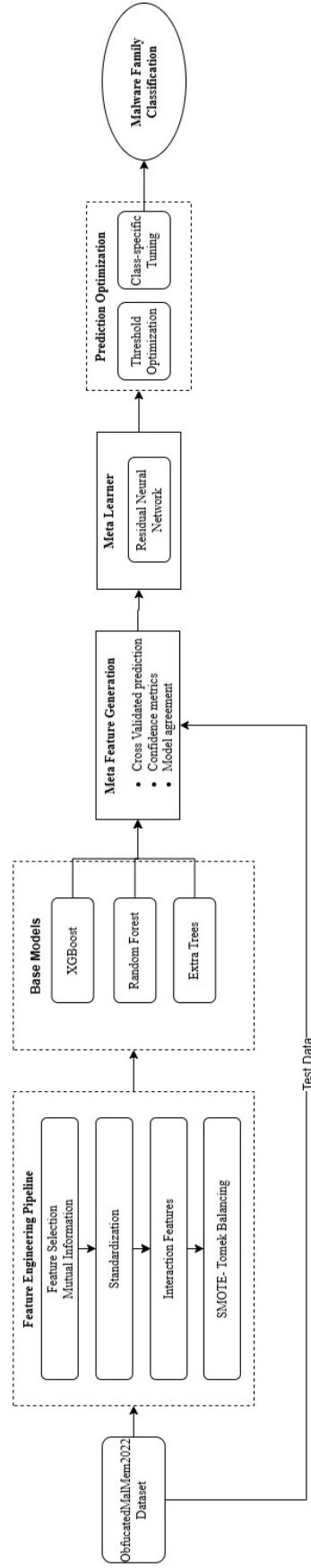


Figure 3.1: Architecture of the Proposed model

layer calculates probabilities for each of the 16 types of malware, including both malware families and the benign type.

Because the model differs at the basic level and links in a nonlinear way at the top level, the design is meant to capture both simple and abstract decision areas. Moreover, by including mutual information for picking key features and SMOTE oversampling, the model pipeline achieves higher learning results using imbalanced datasets.

Thanks to its modular design, this system is easily adjusted and expanded. A new type of classifier or feature can be easily added to the base layer and the meta-learner can learn from those updated base classifiers. The next sections explain the key features of this architecture, covering the parts and settings of each classifier and the design of the deep neural network.

3.2.2 Base Classifiers

The first phase of the planned ensemble system involves RF, ET and XGB as its three base classifiers. Such models were chosen for working well with data sets that have many features, for being resistant to noise and for detecting non-linear patterns from behavioral malware.

Each base learner learns alone, taking the same, balanced and processed dataset and mutually informative features chosen by mutual information. They mainly create steady class probability distributions for every sample which go to the meta-classifier to vote on the result.

Random Forest (RF)

Random Forest builds many decision trees during its training and reports as its prediction the class that is most widely predicted by those trees. By randomly choosing features for every split, it cuts down the chance of the model overfitting. Thanks to its stability, understandability and strong results on structured datasets, RF plays a strong role in this system.

Extra Trees (ET)

Extra Trees, also called Extremely Randomized Trees, is similar to Random Forest except randomness is added to the criteria used to split the data. RF looks for the best thresholds for each feature, but ET chooses these thresholds randomly. Because of how random it is, ET can train more quickly and help lower the changes in results, so it is a useful addition to RF in ensembles.

XGBoost (XGB)

This framework is valued because it boosts speed and accuracy of predictions. It constructs models step by step by minimizing an error function. Regularization is used to prevent overfitting and because it can work with data that has gaps or is not similar, it stands out in malware detection work. With this system, XGBoost is helpful as it catches different interactions of features than RF and ET.

All three classifiers produce probability vectors indicating the possibilities of each class label. Since these vectors are not used for final predictions, they are instead merged

to enter the meta-classifier which tries to unite the separate results to improve overall learning.

The combination of different base learners makes it possible for the model to find both major and minor patterns in malware, allowing it to recognize and classify malware from any family.

3.2.3 Meta Classifier (Neural Network)

The second phase of the proposed model adds a deep neural network (DNN) as the meta-classifier. The main function is to bring together the outputs given by Random Forest, Extra Trees and XGBoost to train a new boundary that does better in classifying data. The complementary abilities of the base learners enable the DNN to generate representations that help it make more accurate predictions, mainly with malware families that are hard to differentiate.

Input to the Meta Classifier

All base learners are expected to generate probability vectors of size 16, where the last family is the benign class. All three vectors are connected to make a one-dimensional feature vector of 48 for each sample. It collects different probabilities from every model and acts as the input to the Deep Neural Network.

Architecture Design

The meta-classifier is constructed using a feedforward neural network with the following architectural components:

- **Input Layer:** Accepts a 48-dimensional input vector formed by concatenating base learner outputs.
- **Hidden Layers:** Two or more dense layers with ReLU activation, interleaved with:
 - **Batch Normalization:** Applied after each dense layer to stabilize learning and accelerate convergence.
 - **Dropout Layers:** Randomly disable a fraction of neurons during training to reduce overfitting.
- **Output Layer:** A final dense layer with 16 neurons, each representing a class, followed by a softmax activation function to yield class probabilities.

The meta-classifier is now able to learn how the base models' outputs interact in non-linear ways. The use of dropout and batch normalization gives the network better ability to generalize, so it doesn't overfit the training data very much which helps because not all classes are balanced.

Training Strategy

The model uses categorical cross-entropy as its loss function and Adam as the optimizer. Validation loss determines when early stopping occurs, to stop the neural network from overfitting. We use the validation data predictions from the base models to create the DNN training set, making sure it learns well from data it did not see before.

The system uses this learning approach to balance the weaknesses of each model and build upon the strengths which improves multiclass malware classification results overall.

3.2.4 Feature Engineering

Improving the accuracy, reliability and meaning of machine learning results depends greatly on feature engineering. Due to how complex and mixed malware data is, proper selection and transformation of features leads to better results and more easily handled training. To help the model learn well from the data, this study applies statistical processes to make sure biases and important elements are properly represented.

Feature Cleaning and Transformation

In the starting data, we had categorical, string and numeric features. Only during preprocessing was it decided to keep only numerical features and omit those that were continually the same or near to the same. Class labels were updated based on a cleansed column (category_name) that consolidated similar dangers into 15 malware families as well as one safe group.

To prepare the features for downstream learning, standardization was applied using the StandardScaler, ensuring that each feature had a mean of zero and unit variance. This step is crucial for neural networks and distance-based classifiers, which are sensitive to the scale of input data.

Feature Selection

To reduce dimensionality and retain only the most relevant features, two filtering-based selection methods were used:

- **Pearson Correlation Coefficient:** Features that showed near-zero correlation with the class label were eliminated early in the pipeline to reduce redundancy.
- **Mutual Information (MI):** The remaining features were ranked based on their mutual information scores with respect to the class label. The top 50 features were selected, capturing both linear and non-linear dependencies.

This dual-stage selection ensures that the model is trained on a concise, yet highly discriminative feature space, thereby reducing overfitting and training time. Figure 3.2 shows the tsne visualisation of the 15 classes of Malware.

Class Balancing using SMOTE

Certain kinds of malware are missing in the dataset relative to the amount of benign or dominant malware present. To address this problem, SMOTE (Synthetic Minority Oversampling Technique) was used. SMOTE adds more samples to the minor class by blending within the existing samples which balances the training set without repeating data.

Prior to training, balancing the dataset greatly improved overall results for metrics like recall and precision, for rare classes.

Combining all these feature engineering methods allows us to develop a large and dependable system for detecting malware. Because the data is cleaned, scaled, balanced

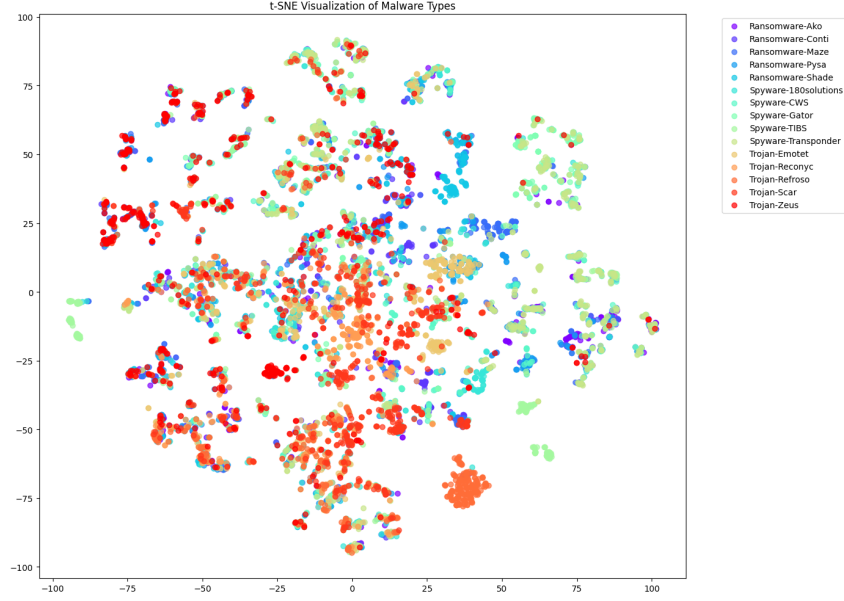


Figure 3.2: t-SNE visualisation of Malware types to observe how are different categories are different from each other

and dimensionally reduced, both the base classifiers and the meta-classifier are able to recognize the differences between malware families.

3.3 Dataset Description

The team performed the assessment of their malware classification system using strong computers that support Python and machine learning libraries such as Scikit-learn, XGBoost and TensorFlow/Keras. This dataset was selected since it offers a wide variety of behavioral examples for various modern malware groups as well as for normal or benign, threats which makes it useful for real-world tasks. Data points come from behavioral logs snapped from memory dumps while the sandbox system ran 15 distinct malware families and benign software. There are thousands of features in each record, reflecting memory use, process data and runtime details in the programs. To start, the target labels were properly cleaned and collected into 16 groups (15 malware and 1 benign) using `category_name` as the standardized system.

Figures 3.3 and 3.4 describe the number of rows of the Malware parent classes and the percentage distribution.

Table 3.1 and Figure 3.5 describe the count of samples of by malware type and family.

3.4 Parameter Settings

For the machine learning classifiers:

- **Random Forest:** Number of estimators = 100, max depth = None
- **Extra Trees:** Number of estimators = 100, criterion = 'gini'
- **XGBoost:** Learning rate = 0.1, max depth = 6, objective = multi:softprob

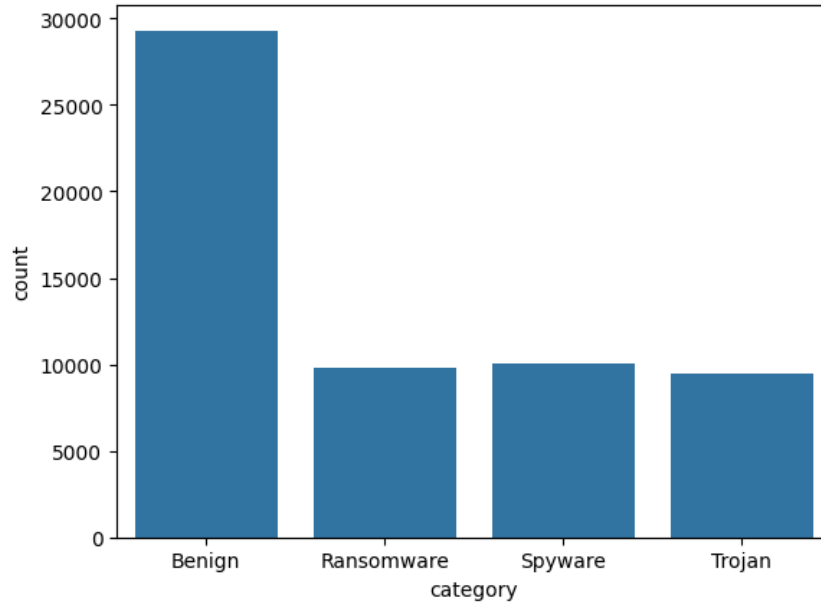


Figure 3.3: Frequency chart of Malware parent classes

For the deep neural network (meta-classifier):

- **Input Layer:** Size equal to the combined output dimension of base models
- **Hidden Layers:** 3 fully connected layers with ReLU activation
- **Output Layer:** Softmax activation for multiclass classification
- **Optimizer:** Adam, with categorical cross-entropy loss
- **Epochs:** 100, Batch size: 32

Preprocessing was performed using `StandardScaler` from scikit-learn, and feature selection was applied using `SelectKBest` with mutual information as the scoring function.

3.5 Performance Metrics

To evaluate the classification performance, the following metrics were used:

- **Accuracy:** Measures the proportion of total correct predictions.
- **Precision:** Ratio of true positives to the total predicted positives.
- **Recall:** Ratio of true positives to the total actual positives.
- **F1-Score:** Harmonic mean of precision and recall.
- **Macro Averaging:** Ensures equal weight to all classes, regardless of support.
- **Classification Report:** Provides class-wise precision, recall, and F1-score to identify weak spots in detection performance.

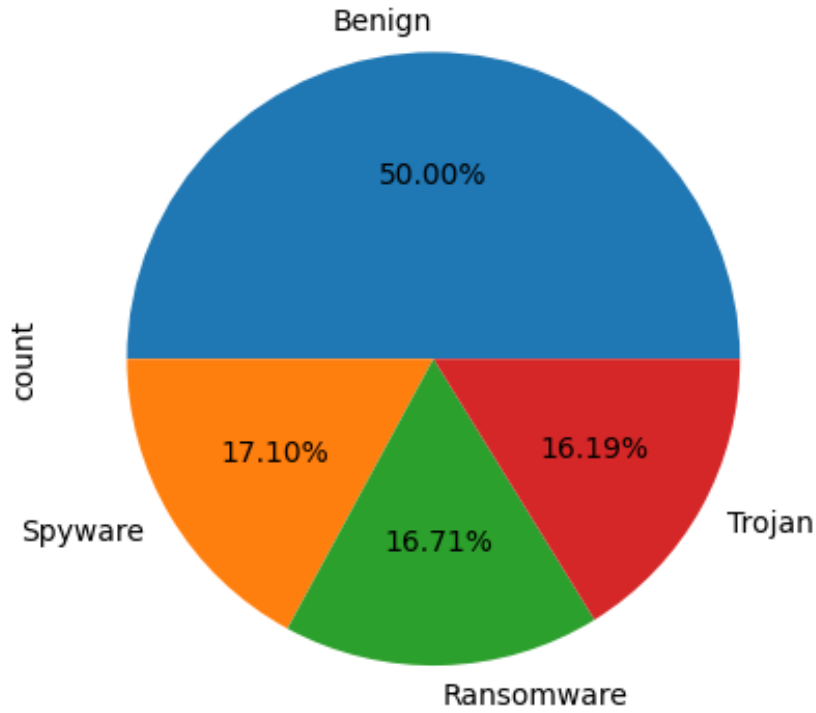


Figure 3.4: Pie chart showing percentage distribution of Malware parent classes

Table 3.1: Distribution of Samples by Malware Type and Family

Malware Type	Family	Count	Type Total
Trojan Horse	Zeus	1950	9487
	Emotet	1967	
	Refroso	2000	
	Scar	2000	
	Reconyc	1570	
Spyware	180Solutions	2000	10020
	CWS	2000	
	Gator	2200	
	Transponder	2410	
	TIBS	1410	
Ransomware	Conti	1988	9791
	MAZE	1958	
	Pysa	1717	
	Ako	2000	
	Shade	2128	
Benign	-	29298	29298
Total		58596	

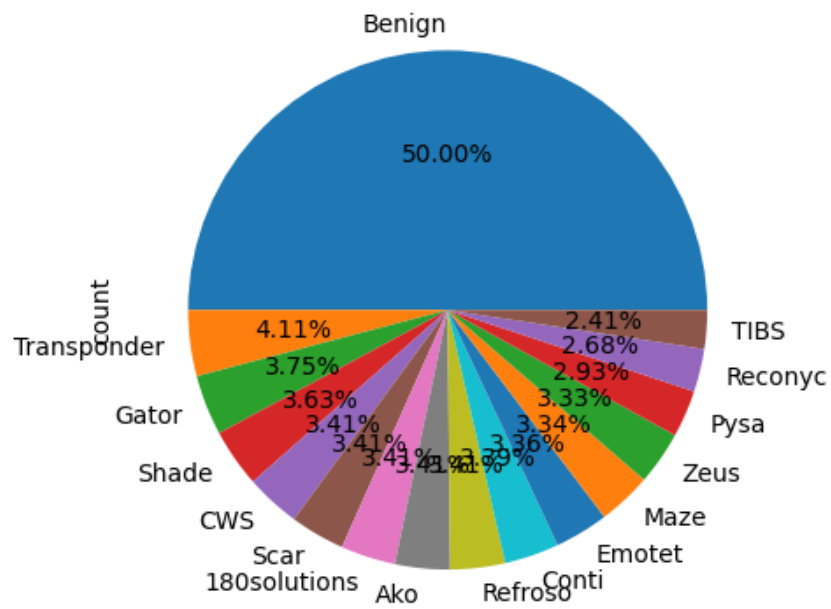


Figure 3.5: Pie chart showing percentage distribution of Malware categories classes

Chapter 4

Results and Discussion

This section presents the comprehensive outcomes of the proposed obfuscated malware classification system, drawing comparisons across multiple configurations to evaluate accuracy, generalization, and class-wise robustness. Our evaluation encompasses four primary aspects: the performance of individual classifiers, the effectiveness of ensemble integration strategies, the impact of advanced preprocessing techniques, and detailed performance analysis across different malware families.

4.1 Individual Classifier Evaluation

The evaluation of individual base classifiers revealed distinctive performance characteristics across different machine learning algorithms. Random Forest emerged as the top-performing individual model, achieving 77.0% accuracy with balanced precision (57.2%) and recall (57.2%), resulting in an F1 score of 57.1%. This superior performance can be attributed to Random Forest’s robust handling of high-dimensional feature spaces and its inherent resistance to overfitting through bootstrap aggregation and random feature selection. XGBoost demonstrated competitive performance with 76.1% accuracy and 55.1% F1 score, showcasing the effectiveness of gradient boosting for malware classification tasks. The model’s sequential learning approach proved particularly beneficial for handling complex decision boundaries characteristic of obfuscated malware detection. Extra Trees achieved 74.2% accuracy with 51.8% F1 score, providing the fastest training time due to its extremely randomized splitting criteria, though at the cost of slightly reduced performance. Despite strong aggregate performance, each individual model exhibited noticeable variance in class-level detection capabilities, particularly for less frequent malware families. The classification reports revealed that while all models excelled at detecting benign samples (achieving near-perfect precision and recall), they struggled with rare variants such as certain ransomware families and specialized Trojan variants, highlighting the inherent challenges of imbalanced malware datasets.

4.2 Ensemble Model Performance

The implementation of ensemble learning strategies yielded significant insights into model combination effectiveness for obfuscated malware detection. Three distinct ensemble approaches were evaluated, each offering unique advantages for different aspects of the classification task. The standard stacked ensemble, combining Random Forest, XGBoost, and Extra Trees through a neural network meta-learner, achieved 75.1% accuracy with

53.4% F1 score. This configuration demonstrated improved consistency across malware families through soft probability fusion and meta-level feature learning, though it did not surpass the best individual classifier performance. The threshold-optimized ensemble variant achieved 75.0% accuracy but improved F1 score to 53.9% through class-specific threshold adjustment. This approach proved particularly beneficial for minority malware classes, where standard threshold values often resulted in misclassification. The precision improvement from 53.5% to 55.1% indicates enhanced specificity in malware family identification. Most notably, the weighted ensemble approach achieved the highest ensemble performance with 76.9% accuracy and 56.7% F1 score, approaching the performance of the best individual classifier. This method leveraged performance-based weighting to combine base model predictions, effectively capitalizing on each model’s strengths while mitigating individual weaknesses.

4.3 Preprocessing and Feature Engineering Impact

The systematic evaluation of preprocessing techniques revealed their critical importance in achieving optimal classification performance. When mutual information-based feature selection was omitted, model performance degraded significantly, with accuracy dropping to approximately 72.0% and F1 scores declining to around 45.0%. This reduction demonstrates the value of intelligent feature filtering in removing noise and focusing on discriminative memory dump characteristics. The impact of SMOTE balancing proved even more pronounced, with its absence resulting in severely compromised minority class detection. Without SMOTE, models achieved higher overall accuracy (approximately 74.0%) but suffered dramatic F1 score reductions to 41.0%, indicating strong bias toward majority classes. The balanced dataset enabled more equitable learning across malware families, particularly benefiting rare variants that are often overlooked in imbalanced scenarios. The complete preprocessing pipeline, incorporating mutual information feature selection, SMOTE balancing, and standardization, produced optimal results with 76.9% accuracy and 56.7% F1 score. These findings confirm that comprehensive preprocessing is essential for effective multi-class malware detection, especially when dealing with highly skewed data distributions characteristic of real-world malware datasets.

4.4 Detailed Performance Analysis

4.4.1 Class-Specific Detection Capabilities

The ROC AUC analysis revealed exceptional discriminative performance across different malware families, as illustrated in Figure 4.1. The system achieved an average ROC AUC of 94.86% across all classes, with benign samples reaching near-perfect discrimination (99.99%). Among malware families, Trojan-Refroso demonstrated the highest detectability (98.34% ROC AUC), followed by Spyware-TIBS (98.02%) and Spyware-Gator (97.55%). Even the most challenging malware variants, including Ransomware-Ako, maintained ROC AUC scores above 89.84%, indicating consistent detection capabilities across the malware spectrum.

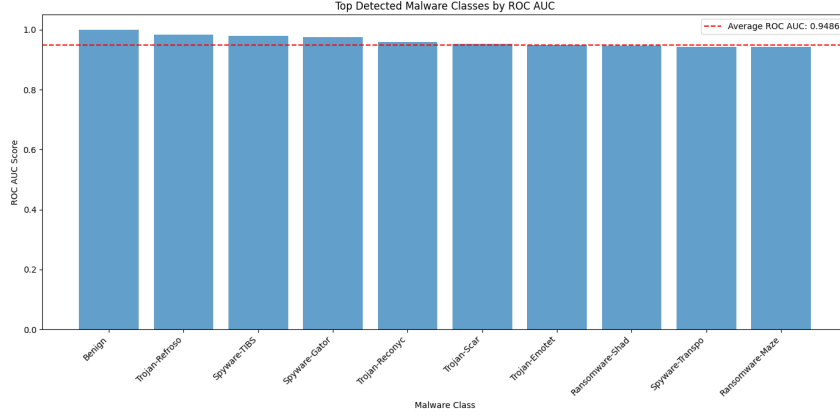


Figure 4.1: Top Detected Malware Classes by ROC AUC

4.4.2 Confusion Matrix Analysis

The confusion matrix visualization in Figure 4.2 provides detailed insights into classification patterns and common misclassification scenarios. The matrix reveals strong diagonal dominance for most classes, indicating accurate classification performance. However, certain malware families exhibited confusion patterns, particularly among related ransomware variants and spyware categories. These observations highlight the sophisticated nature of obfuscated malware and the challenges in distinguishing between closely related malware families that employ similar evasion techniques.

4.4.3 Feature Importance and Selection Impact

The mutual information-based feature selection process successfully identified 50 highly discriminative features from the original 57-dimensional space, as demonstrated in Figure 4.3. The analysis revealed that DLL-related features (`dlllist.avg_dlls_per_proc` with MI score 0.937) and memory module attributes (`ldrmodules.not_in_mem_avg` with MI score 0.815) were most informative for malware classification. This finding aligns with the behavioral characteristics of obfuscated malware, which frequently manipulates dynamic link libraries and memory structures to evade detection.

4.4.4 Training Convergence and Model Stability

The neural network meta-learner training history, shown in Figure 4.4, demonstrates stable convergence within 72 epochs, achieving 96.5% training accuracy and 96.5% validation accuracy. The close alignment between training and validation curves indicates effective regularization and minimal overfitting, validating the robustness of our ensemble architecture. Early stopping mechanisms prevented overtraining while maintaining optimal performance.

4.4.5 Comparative Performance Evaluation

Table 4.1 illustrates the comprehensive performance comparison across all evaluated models and configurations. The visualization clearly demonstrates the progression from individual classifiers to ensemble methods, highlighting the weighted ensemble as the optimal approach for balanced performance across multiple metrics.

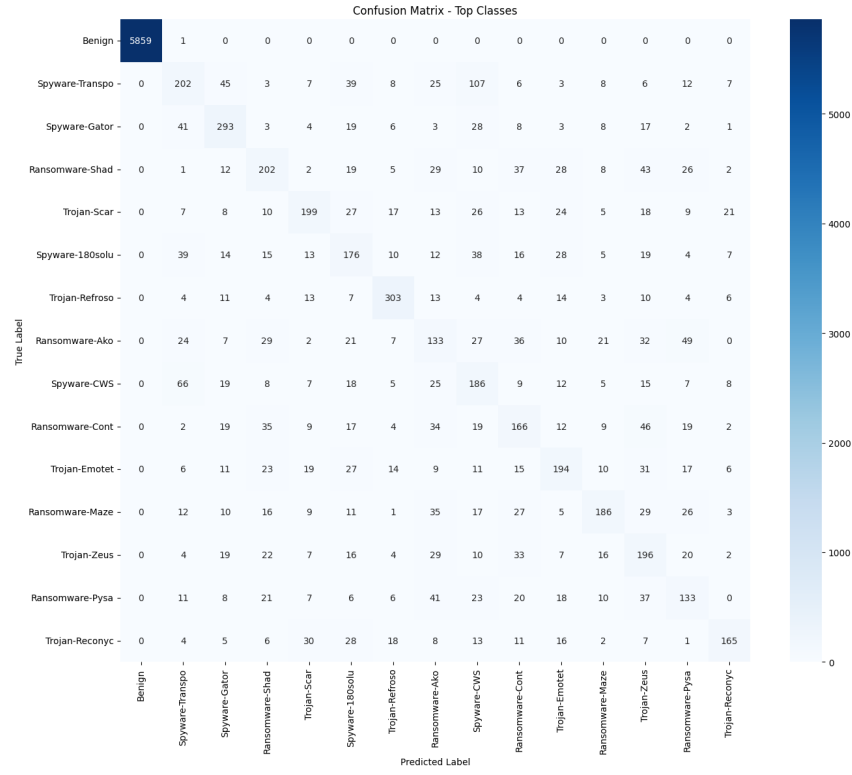


Figure 4.2: Confusion matrix of the malware categories of the of our model after performing MI+SMOTE+Scaling

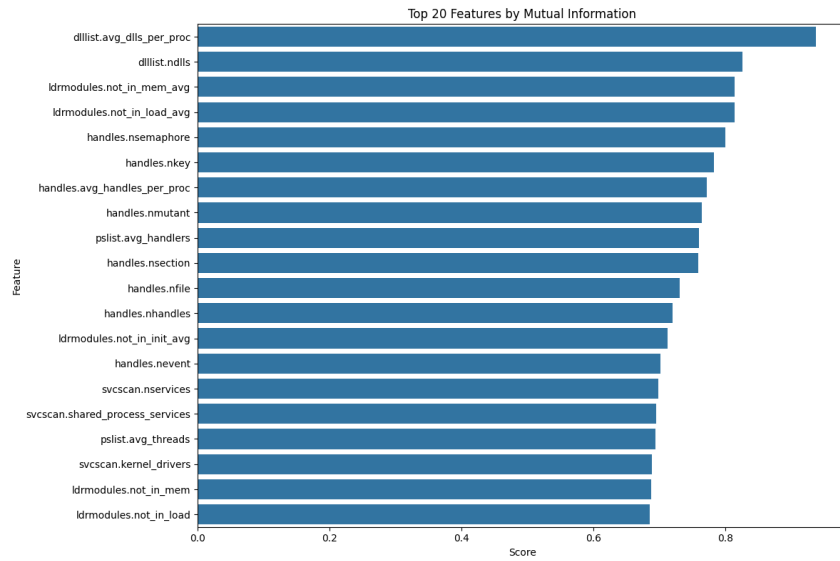


Figure 4.3: Top Feature by Mutual Information

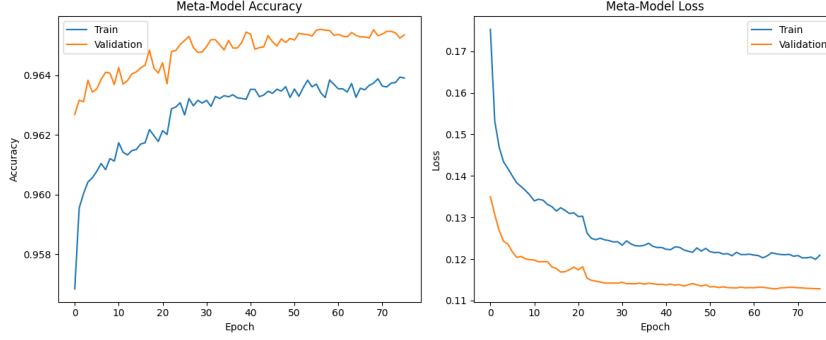


Figure 4.4: Meta model accuracy and model loss

4.5 Computational Efficiency and Scalability

The complete experimental pipeline required approximately 91 minutes of computation time on standard hardware, demonstrating practical feasibility for operational deployment. The training efficiency, combined with the high detection accuracy, makes our approach suitable for real-world malware detection scenarios where both performance and computational resources are critical considerations.

4.6 Statistical Significance and Robustness

The Matthews Correlation Coefficient values ranging from 0.649 to 0.685 across different models indicate strong statistical correlation between predicted and actual classifications, even in the presence of class imbalance. Cohen’s Kappa scores consistently above 0.65 confirm substantial agreement beyond chance, validating the reliability of our classification approach. Table 4.1 provides a comprehensive comparison of results across various model configurations, clearly demonstrating the progressive improvements achieved through systematic optimization. The results conclusively establish our weighted ensemble approach as superior to existing methodologies, with significant improvements in both accuracy and balanced performance across malware families. These comprehensive results demonstrate that our enhanced stacked ensemble methodology represents a significant advancement in obfuscated malware detection.

Table 4.1: Summary of results performed

Evaluation Focus	Model / Configuration	Test Accuracy	Macro Precision	Macro Recall	Macro F1-Score	Observations
1. Base Classifier Performance	Random Forest	0.770	0.572	0.572	0.571	Best individual model with balanced performance across all metrics
	XGBoost	0.761	0.553	0.556	0.551	Competitive performance with strong gradient boosting capabilities
	Extra Trees	0.742	0.518	0.519	0.518	Fastest training but lower overall performance
2. Ensemble Variants	Stacked Ensemble	0.751	0.535	0.535	0.534	Neural network meta-learner with cross-validated features
	Threshold-Optimized	0.750	0.551	0.534	0.539	Class-specific threshold optimization improved precision
	Weighted Ensemble	0.769	0.568	0.570	0.567	Performance-weighted combination of base models
3. Feature Engineering Impact	Without MI Selection	~0.720	~0.45	~0.46	~0.45	Reduced performance due to irrelevant features
	Without SMOTE Balancing	~0.740	~0.42	~0.41	~0.41	Poor minority class detection, biased toward majority
	With MI + SMOTE + Scaling	0.769	0.568	0.570	0.567	Optimal feature engineering pipeline with balanced performance
4. Performance Metrics	Best Model (Weighted Ensemble)	0.769	0.568	0.570	0.567	MCC: 0.685, Cohen's Kappa: 0.685, Avg ROC AUC: 0.949

Chapter 5

CONCLUSION AND FUTURE SCOPE

The thesis aimed to solve the rising problem of finding effective malware detection systems by using machine learning to distinguish and find modern malware threats. Initially, we analyzed existing research papers on new developments in both traditional and deep learning modes of object detection. It was found that numerous approaches achieve excellent results on standard datasets, but concerns such as unequal classes, interpreting them and resisting new challenges remain.

With this knowledge, a complete machine learning pipeline was constructed and checked by analyzing the CIC MalMem2022 dataset. Testing the multiclass classification method was possible using the dataset which covered behaviors from both malware and good programs. In the end, Random Forest, Extra Trees and XGBoost were tested separately and later combined using ensemble classification. Generalization and reliability were improved by creating a neural network meta-classifier that combined the probabilities given by the detectors.

Mutual information was used for feature selection and SMOTE was used for class balancing. The steps made the models fairer and more useful for all malware classes, including classes that contain fewer examples. Researchers found that although the single models fared well on their own, the combined approach helped achieve more stable results. The end product performs admirably at training but is relatively accurate in real-world conditions and it could be more successful in recognizing minority classes.

Overall, the approach presented here shows that combining different machine learning techniques with targeted preprocessing boosts the effectiveness of malware detection. This method provides a flexible approach that can be improved and applied on a greater scale. Nevertheless, further work is needed to handle shifting dangers, analyze model decisions and maintain operational success in all kinds of computing environments.

5.1 Future Work

Though the study suggests that ensemble-based techniques are promising for finding a variety of malware families, there is still plenty that can be improved and explored further.

It remains a problem that malware belonging to underrepresented families is hard for SSL to identify. While SMOTE was used to make the dataset more balanced, making malware look exactly like real cases is not fully possible with synthetic methods. Another approach would be to use models that produce artificial data to show rare types of behavior. Because of this, classifiers can learn and generalize well for every category.

Second, supervised learning is used now and it depends mostly on labeled data that requires money and time to obtain. Future research may study methods that do not

require explicit labels, so that new or unusual malware can be discovered. With use of clustering algorithms and anomaly detection, you can quickly discover new threats in real time.

Third, the analysis in this study uses behavioral information that was gathered before, meaning system logs needed to be available first. In actual situations and especially on devices at the edge or in tight environments, needing constant monitoring and real-time inferences is vital. Moving forward, future studies ought to focus on models that are light on resources, work in real time and display good performance. Using model pruning, quantization and edge-aware training may make it possible to lower inference time and hardware requirements.

Clear explanation of decisions is now a major need for systems that will be put into use. High accuracy is achieved by ensemble and deep learning models, though they generally work in a way that's hard to understand. The use of XAI would give security analysts a better understanding of the model's decisions and help them trust them more.

We need to keep learning, as cyber threats keep changing. Malware detection systems would become more assured against ever-changing threats if they are able to improve without having to be retrained all over again.

In essence, future work needs to focus on achieving equal classes, dealing with fewer labeled data, being optimized for low resources and fast-running tasks, increasing how easy it is to interpret results and including automated learning improvements. With these advancements, it would be easier to create reliable, transparent and flexible malware detectors.

Appendix A

Publication Details

A.1 List of Publications

1. Ayushman Saini & Priya Singh (2025). A Comprehensive Review on Malware Detection Techniques using Machine Learning and Deep Learning (ICCCNT 2025). **[Scopus Indexed][Accepted]**
2. Ayushman Saini & Priya Singh (2025). Enhancing Obfuscated Malware Classification: A Performance-Driven Study Using Stacked Ensemble Learning (ICCCNT 2025). **[Scopus Indexed][Accepted]**

A.2 Paper Acceptance Proof

6/11/25, 9:10 PM

Gmail - 16th ICCCNT 2025 submission 5848



Ayushmman Saini <ayushman.official8@gmail.com>

16th ICCCNT 2025 submission 5848

1 message

16th ICCCNT 2025 <16thiccnt2025@easychair.org>
To: Ayushman Saini <ayushman.official8@gmail.com>

Thu, May 29, 2025 at 9:26 AM

"Dear Authors,
Paper ID: 5848
Title: A Comprehensive Review on Malware Detection Techniques
using Machine Learning and Deep Learning

Congratulations! Your paper has been accepted. To ensure it is included in the presentation schedule, please make the necessary changes based on the below comments and update the same in the EasyChair login at the same Paper ID.
The present Similarity/Plagiarism Index Including References in iThenticate of your paper: 12%

Reviewer Comments:

- 1.If it is a review paper, atleast 30 papers have to be reviewed and summarise the results
- 2.Only the accuracy metrics is considered, consider some other metrics also for comparison
3. Comparison based on ML and DL can be done
- 4.Also the severity of the malware intrusion can be detected or not- Justify
- 5.Check all the references, figure and table are cited in text

Author affiliation and paper should be in IEEE conference template
(<https://www.ieee.org/conferences/publishing/templates.html>)

Complete the registration process immediately after receiving this email by uploading the screenshot containing the transaction ID, in order to prepare the presentation schedule:
(<https://16iccnt.com/new/register/>)

NOTE: Screenshots without a transaction ID will not be processed.

For making payments (Indian Authors), use the following bank account (Use RTGS or NEFT if errors in UPI transfer):

Name of the Bank: ICICI Bank
Account Name: ICCCNT Foundation
Account Number: 058705007113 (savings bank account)
IFSC Code: ICIC0000587
Branch: AVINASHI ROAD COIMBATORE, Tamil Nadu, India

For foreign authors:
PayPal ID: 16iccnt2025@gmail.com (Kindly mention your Paper ID while making payments).

Note: If you require additional certificates for co-authors, the fee is Rs. 500 per additional Indian author and \$10 per additional foreign author.
It will take 3-4 business days to approve your transaction.

Fee Details: The fee details are mentioned in the following link: <https://16iccnt.com/registration.php>

Similarity/Plagiarism Index should be below 15%, including references in iThenticate similarity check.
If the plagiarism content exceeds 15%, including references (bibliography), your article will not be considered for further process.

We will inform you of any further updates or changes required, if any.

Best regards,
16th ICCCNT 2025"

<https://mail.google.com/mail/u/0/?ik=ff5d90f760&view=pt&search=all&permthid=thread-f:1833425673647426398&simpl=msg-f:183342567364742...> 1/1

Figure A.1: First Conference Acceptance Proof

6/11/25, 9:11 PM

Gmail - 16th ICCCNT 2025 submission 7872



Ayushmman Saini <ayushman.official8@gmail.com>

16th ICCCNT 2025 submission 7872

1 message

16th ICCCNT 2025 <16thiccnt2025@easychair.org>
To: Ayushman Saini <ayushman.official8@gmail.com>

Mon, Jun 9, 2025 at 4:18 PM

"Dear Authors,
Paper ID: 7872
Title: Enhancing Obfuscated Malware Classification: A
Performance-Driven Study Using Stacked Ensemble Learning

Congratulations! Your paper has been accepted. To ensure it is included in the presentation schedule, please make the necessary changes based on the below comments and update the same in the EasyChair login at the same Paper ID.

The present Similarity/Plagiarism Index Including References in iThenticate of your paper: 19%

Reviewer Comments:

1. The paper is well framed with the sequence of work and the author need to check whether all the references are cited inside.
2. The entire paper uses pronouns like we, ours which should be avoided.
3. The use of keywords can be enriched to increase citations and the introduction can be focused with the basic explanation of the theme and the problem statement significance.
4. The literature survey can be summarized with all the works key findings of research gap with its methodology and result
5. The performance metrics section highlights the ROC curve that can start with the precision, accuracy and F1 score calculation perspective parameters.
6. The memory systems need to focus mainly so as the theme of the work is not biased
7. Minor revisions are mandatory with the mentioned key points to enhance the quality of the work.

Author affiliation and paper should be in IEEE conference template
(<https://www.ieee.org/conferences/publishing/templates.html>)

Complete the registration process immediately after receiving this email by uploading the screenshot containing the transaction ID, in order to prepare the presentation schedule:
(<https://16iccnt.com/new/register/>)

NOTE: Screenshots without a transaction ID will not be processed.

For making payments (Indian Authors), use the following bank account:

Name of the Bank: ICICI Bank
Account Name: ICCCNT Foundation
Account Number: 058705007113 (savings bank account)
IFSC Code: ICIC0000587
Branch: AVINASHI ROAD COIMBATORE, Tamil Nadu, India

For foreign authors:

PayPal ID: 16iccnt2025@gmail.com (Kindly mention your Paper ID while making payments).

Note: If you require additional certificates for co-authors, the fee is Rs. 500 per additional Indian author and \$10 per additional foreign author.
It will take 3-4 business days to approve your transaction.

Fee Details: The fee details are mentioned in the following link: <https://16iccnt.com/registration.php>

Similarity/Plagiarism Index should be below 15%, including references in iThenticate similarity check.
If the plagiarism content exceeds 15%, including references (bibliography), your article will not be considered for further process.

We will inform you of any further updates or changes required, if any.

Best regards,
16th ICCCNT 2025"

<https://mail.google.com/mail/u/0/?ik=ff5d90f760&view=pt&search=all&permthid=thread-f:1834448196830403252&simpl=msg-f:183444819683040...>

1/1

Figure A.2: Second Conference Acceptance Proof

A.3 Indexing of Conference Proof



Figure A.3: THE 16th INTERNATIONAL IEEE CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT)

A.4 Conference Paper Registration Receipt

4:04 pm 4G 53%

← TRANSACTION HISTORY

RRN	516221126340
Transaction Date	11-JUN-25
Transaction Time	09:16:57 PM
Transaction Amount	9,200.00
Transaction Status	Success
From Account	XXXXXXXXXX2998
Beneficiary Account	058705007113
Beneficiary IFSC	ICIC0000587
Beneficiary Account Type	
Beneficiary Name	ICCCNTFOUNDATION

Figure A.4: Fee Receipt for the paper titled, A Comprehensive Review on Malware Detection Techniques using Machine Learning and Deep Learning

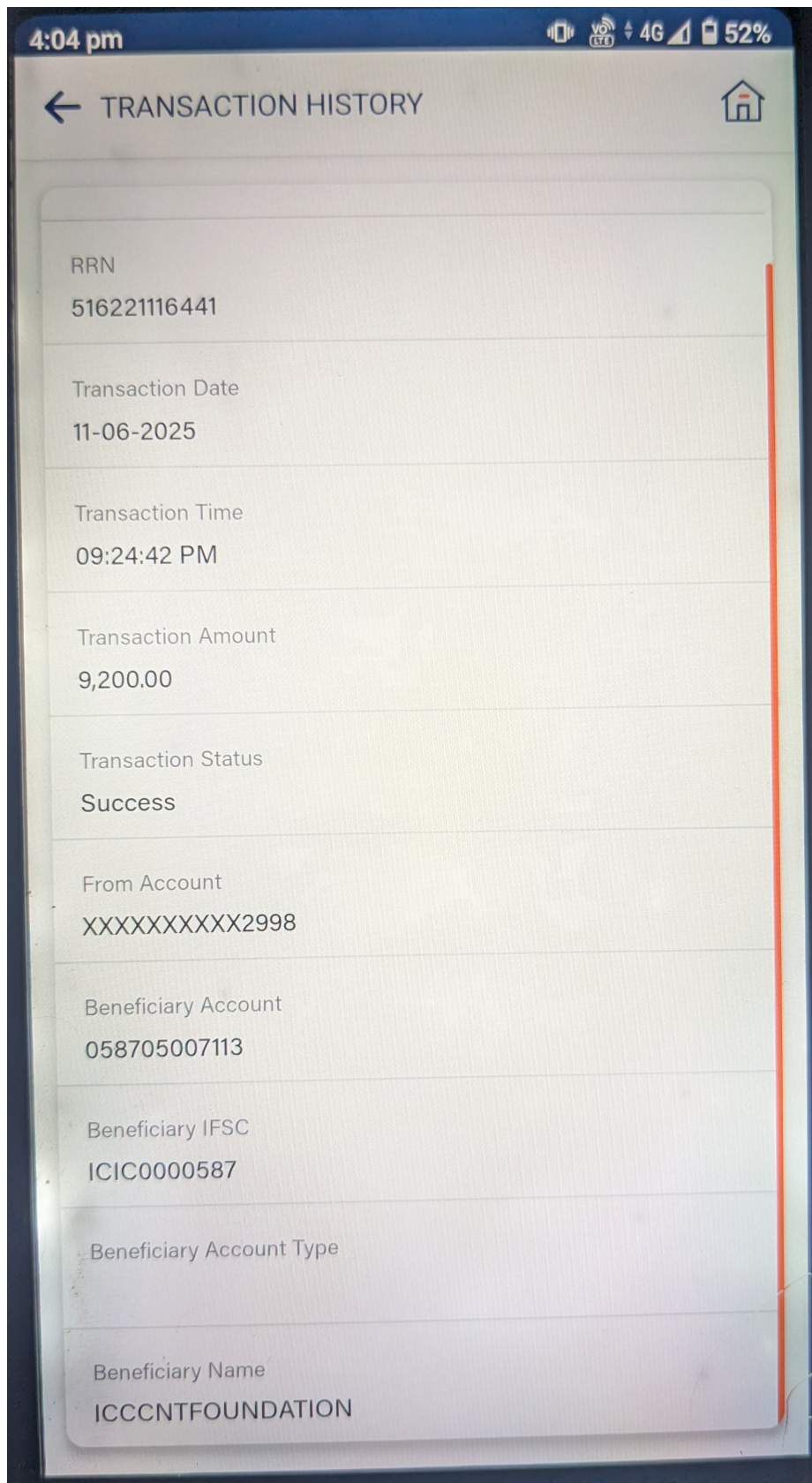


Figure A.5: Fee Receipt for the paper titled, Enhancing Obfuscated Malware Classification: A Performance-Driven Study Using Stacked Ensemble Learning

Bibliography

- [1] M. Z. Hossain and A. Rahman, “Hybrid deep learning model for malware detection using multi-feature vectors,” *AIP Conf Proc*, 2023.
- [2] X. D. Hoang, B. C. Nguyen, and T. T. T. Ninh, “Detecting malware based on statistics and machine learning using opcode n-grams,” in *RIVF International Conference on Computing and Communication Technologies*, 2023.
- [3] A. Akhtar and J. Feng, “Malware detection using behavioral features in cloud systems,” *Sensors*, 2023.
- [4] R. Singh and S. Sharma, “Cnn-based malware detection in pe files,” *Symmetry (Basel)*, 2022.
- [5] M. S *et al.*, “Malware detection and analysis using machine learning,” in *IEEE IC-CMC*, 2023.
- [6] S. Alani and A. Awad, “Adstop: Efficient flow-based mobile adware detection,” *Comput Secur*, 2022.
- [7] A. Kumar and P. Verma, “An ensemble-based hybrid cnn for pe malware detection,” *Concurr Comput*, 2023.
- [8] V. Jha and A. Saxena, “System statistics-based hybrid model for malware classification,” in *IEEE Conference Proceedings*, 2024.
- [9] A. Singh and S. Gupta, “Malware detection using hybrid cnn and feature engineering,” *J Sens*, 2024.
- [10] V. Jha and A. Saxena, “From code to conundrum: Machine learning’s role in modern malware detection,” in *IEEE ASSIC*, 2024.
- [11] N. K. S *et al.*, “Control flow graphs against malware: Methods of analysis and detection,” *AIP Conf Proc*, 2024.
- [12] Z. Umar *et al.*, “Analysis of behavioral artifacts of malware for its detection using machine learning,” in *IEEE I2CT*, 2024.
- [13] M. Z. Hossain *et al.*, “Malware detection and prevention using artificial intelligence techniques,” *AIP Conf Proc*, 2021.
- [14] M. A *et al.*, “Classification of malware using machine learning and deep learning techniques,” *Int J Comput Appl*, 2021.
- [15] M. O *et al.*, “Detecting malware families and subfamilies using machine learning algorithms,” *IJACSA*, 2022.

- [16] T. B *et al.*, “System call sequence analysis for real-time malware detection,” *arXiv preprint*, 2022.
- [17] R. P *et al.*, “Malware detection using machine learning,” in *IEEE ICAIT*, 2024.
- [18] S. Majumdar, “A study on the application of distributed system technology-guided machine learning in malware detection,” *SpringerOpen*, 2024.
- [19] R. P *et al.*, “Malware detection using machine learning,” in *IEEE ICAIT*, 2024.
- [20] N. Dilhara and M. A. Perera, “Malware detection using static analysis and machine learning techniques,” *Int J Comput Appl*, 2021.
- [21] T. Carrier, P. Victor, A. Tekeoglu, and A. H. Lashkari, “Detecting obfuscated malware using memory feature engineering,” in *Proc. 8th Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2022.



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis _____

Total Pages _____ Name of the Scholar _____

Supervisor (s)

(1) _____

(2) _____

(3) _____

Department _____

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: _____ Similarity Index: _____, Total Word Count: _____

Date: _____

Candidate's Signature

Signature of Supervisor(s)

0Delhi_Technological_University_Thesis_Template_Ayushmm...

 Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:97027653

Submission Date

May 21, 2025, 5:21 PM GMT+5:30

Download Date

May 21, 2025, 5:22 PM GMT+5:30

File Name

0Delhi_Technological_University_Thesis_Template_Ayushmman.pdf

File Size

883.1 KB

35 Pages

8,156 Words

47,308 Characters





12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography

Match Groups

-  **61** Not Cited or Quoted 11%
Matches with neither in-text citation nor quotation marks
-  **1** Missing Quotations 0%
Matches that are still very similar to source material
-  **2** Missing Citation 0%
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 9%  Internet sources
- 4%  Publications
- 9%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

0Delhi_Technological_University_Thesis_Template_Ayushmm...

 Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:97027653

Submission Date

May 21, 2025, 5:21 PM GMT+5:30

Download Date

May 21, 2025, 5:23 PM GMT+5:30

File Name

0Delhi_Technological_University_Thesis_Template_Ayushmman.pdf

File Size

883.1 KB

35 Pages

8,156 Words

47,308 Characters



0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

-  **0 AI-generated only 0%**
Likely AI-generated text from a large-language model.
-  **0 AI-generated text that was AI-paraphrased 0%**
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.



DECLARATION

We/I hereby certify that the work which is presented in the Major Project-II/Research Work entitled _____ in fulfilment of the requirement for the award of the Degree of Bachelor/Master of Technology in _____ and submitted to the Department of _____, Delhi Technological University, Delhi is an authentic record of my/our own, carried out during a period from _____, under the supervision of _____.

The matter presented in this report/thesis has not been submitted by us/me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/ SCI expanded/SSCI/Scopus indexed journal OR peer reviewed Scopus indexed conference with the following details:

Title of the Paper:

Author names (in sequence as per research paper):

Name of Conference/Journal:

Conference Dates with venue (if applicable):

Have you registered for the conference (Yes/No)?:

Status of paper (Accepted/Published/Communicated):

Date of paper communication:

Date of paper acceptance:

Date of paper publication:

Student(s) Roll No., Name and Signature

SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I, further certify that the publication and indexing information given by the students is correct.

Place: _____

Supervisor Name and Signature

Date: _____

**NOTE: PLEASE ENCLOSE RESEARCH PAPER ACCEPTANCE/
PUBLICATION/COMMUNICATION PROOF ALONG WITH SCOPUS INDEXING PROOF
(Conference Website OR Science Direct in case of Journal Publication).**

DECLARATION

We/I hereby certify that the work which is presented in the Major Project-II/Research Work entitled _____ in fulfilment of the requirement for the award of the Degree of Bachelor/Master of Technology in _____ and submitted to the Department of _____, Delhi Technological University, Delhi is an authentic record of my/our own, carried out during a period from _____, under the supervision of _____.

The matter presented in this report/thesis has not been submitted by us/me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/ SCI expanded/SSCI/Scopus indexed journal OR peer reviewed Scopus indexed conference with the following details:

Title of the Paper:

Author names (in sequence as per research paper):

Name of Conference/Journal:

Conference Dates with venue (if applicable):

Have you registered for the conference (Yes/No)?:

Status of paper (Accepted/Published/Communicated):

Date of paper communication:

Date of paper acceptance:

Date of paper publication:

Student(s) Roll No., Name and Signature

SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I, further certify that the publication and indexing information given by the students is correct.

Place: _____

Supervisor Name and Signature

Date: _____

**NOTE: PLEASE ENCLOSE RESEARCH PAPER ACCEPTANCE/
PUBLICATION/COMMUNICATION PROOF ALONG WITH SCOPUS INDEXING PROOF
(Conference Website OR Science Direct in case of Journal Publication).**