A Robust Ensemble Learning Framework for Binary and Multiclass Malware Classification Over Diverse Datasets

by
Yeasin Arafath
20201052
Apon Hossain
20301355
Nazihah Islam Nawreen
24141096
Tousiqul Islam Talukder
21301679
Raisa Tahiatul Aziz
20301426

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2025

Declarartion

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Yeasin Arafath
20201052

Apon Hossain
20301355

Nazihah Islam Nawreen
24141096

Tousiqul Islam Talukder
21301679

Raisa Tahiatul Aziz
20301426

Approval

The thesis/project titled "**A Robust Ensemble Learning Framework for Binary and Multiclass Malware Classification Over Diverse Datasets** " submitted by

1. Yeasin Arafath (20201052)
2. Apon Hossain (20301355)
3. Nazihah Islam Nawreen (24141096)
4. Tousiqul Islam Talukder (21301679)
5. Raisa Tahiatul Aziz (20301426)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 24, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Member)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
BRAC University

Acknowledgement

First and foremost, all praise is due to the Almighty Allah, without whose boundless mercy and guidance this thesis would not have been possible.

We would like to express our deepest gratitude to our respected supervisor, Dr. Muhammad Iqbal Hossain, whose unwavering support, insightful advice, and constant encouragement have been invaluable throughout the course of our research. His expertise and patience have played a vital role in helping us overcome challenges and stay focused on our goals.

We are also profoundly thankful to our parents. Their unconditional love, continuous support, and heartfelt prayers have been our greatest strength. Without their unwavering belief in us, reaching this significant milestone in our academic journey would have been far more difficult. This thesis is a result of the collective support and guidance from all those who stood beside us, and for that, we remain truly grateful.

Ethical Statement

This research was conducted with the utmost adherence to ethical standards in data handling, experimentation, and reporting. The datasets used in this study, including CIC-MalMem-2022 and other publicly available malware datasets, are open-source and were utilized strictly for academic and research purposes. No personally identifiable information (PII) or sensitive user data was accessed or processed during the course of this work.

All experiments were carried out in a controlled and secure environment to ensure the safe handling of potentially harmful software samples. The intent of this study is solely to contribute to the advancement of cybersecurity through improved malware detection techniques, not to replicate, distribute, or misuse any malicious code.

The authors affirm that this work does not involve any form of human or animal experimentation and complies fully with institutional and international ethical research guidelines.

Table of Contents

List of Figures

List of Tables

Abstract

Cybercrime has surged due to the widespread use of the Internet, posing a significant threat to global digital security, with obfuscated malware presenting a particular challenge through its sophisticated code-hiding techniques. This study addresses the classification of obfuscated malware using the CIC-MalMem-2022 dataset, comprising 29,298 memory dump samples across benign instances and multiple malware families (e.g., Spyware, Ransomware, Trojan Horse), alongside three additional datasets for testing our model. Our primary objective is to create a model that enhances the accuracy of multi-class classification, focusing on malware families, while also evaluating binary and malware category classifications. To mitigate class imbalance, we employ the Random UnderSampler technique, paired with feature selection using `feature_importance` to identify discriminative memory-based features. The highest accuracy achieved was 99.99% for binary classification. Besides,for multiclass classification, we obtained approximately 96% and 94% (balanced dataset) for 4-class classification using RandomForest-LightGBM, and 99% and 99.99% (balanced dataset) for 16-class classification using AdaBoost-LightGBM on the primary dataset. We evaluated a range of machine learning (ML) models, including AdaBoost, Decision Tree, Random Forest, and hybrid ensembles with confidence-based refinement, among which AdaBoost-LightGBM and RandomForest-LightGBM are our proposed models.

Keywords: Obfuscated Malware, Benign, Trojan Horse, Spyware, Ransomware, Polymorphic malware, Hybrid Models, CIC MalMem 2022 Dataset.

Chapter 1

Introduction

The emergence of obfuscated malware, which may hide its code using polymorphism, packaging, encryption, and other sandbox-detection measures, is a result of the quick development of technology. This makes analysis much more difficult by enabling it to identify and modify behavior independently, circumventing conventional signature-based protections. Because of its versatility, obfuscated malware may evade antivirus software and is more difficult to identify in sandbox situations, where it can detect and change its behavior in confined circumstances. Because of this intricacy, malware analysis becomes even more complicated, requiring sophisticated methods for accurate categorization.

Multi-class classification of obfuscated malware, leveraging memory dump dataset to distinguish between benign instances and multiple malware families, addressing the critical need for robust detection in an increasingly dynamic threat environment. These crucial advances in the evolution of malware show a constant escalation: malicious software designed to get past cybersecurity systems gets more complex and variable as countermeasures get stronger. Polymorphic malware is a prime example of the ongoing strategic struggle between security experts and cybercriminals. This is a dynamic environment where new defensive strategies are vital and the risks are always high. In the banking industry, [1] ensuring privacy and security becomes a significant concern in order to maintain all operations. Malware is a kind of software specifically created to cause damage or get illegal entry into computer systems.

In the 1990s, there was an increase in the prevalence of internet-distributed worms like the "Morris Worm" and macro viruses [2]. In the 2000s, more sophisticated forms of malicious software emerged, including trojans, ransomware, and spyware. Examples of these include the "ILOVEYOU" worm. Notable instances include the massive "WannaCry" ransomware outbreak, both of which took advantage of security flaws in Microsoft [3] operating systems. Emotet, a malware that emerged as a banking trojan in 2014, has evolved into a harmful platform for disseminating various types of malware using a malware-as-a-service (MaaS) model. Similar to the Storm Worm, Emotet propagates via spam emails and has the ability to adapt and change. Despite being dismantled in 2021, Emotet reappeared using innovative methods, highlighting the challenges cybersecurity specialists face in combating advanced attacks [4]. Also, In June 2025, cybersecurity researchers discovered a massive data breach exposing over 16 billion user credentials, making it one of the largest known leaks to date. The compromised data, likely gathered through infostealer malware and credential-harvesting attacks, affects major platforms such as Google, Facebook, and Telegram, increasing the risk of phishing, account takeovers, and

credential stuffing attacks [5][6].

According to our own study, the number of assaults that occur each year between 2014 and 2023 is shown in Figure 1 under the heading "Anomaly attacks." In this case, the y-axis represents the number of attacks, while the x-axis represents the years (2014–2023). It is clear that there were more assaults, 250 as opposed to 120. The graph demonstrates that the values are increasing rapidly year 7 over year, suggesting an increase in the frequency of abnormal attacks. The assault has only gotten worse, as shown by the fact that it peaked in 2023 at 250. The amount of polymorphic malware is increasing daily and classifying and identifying it can be quite difficult.

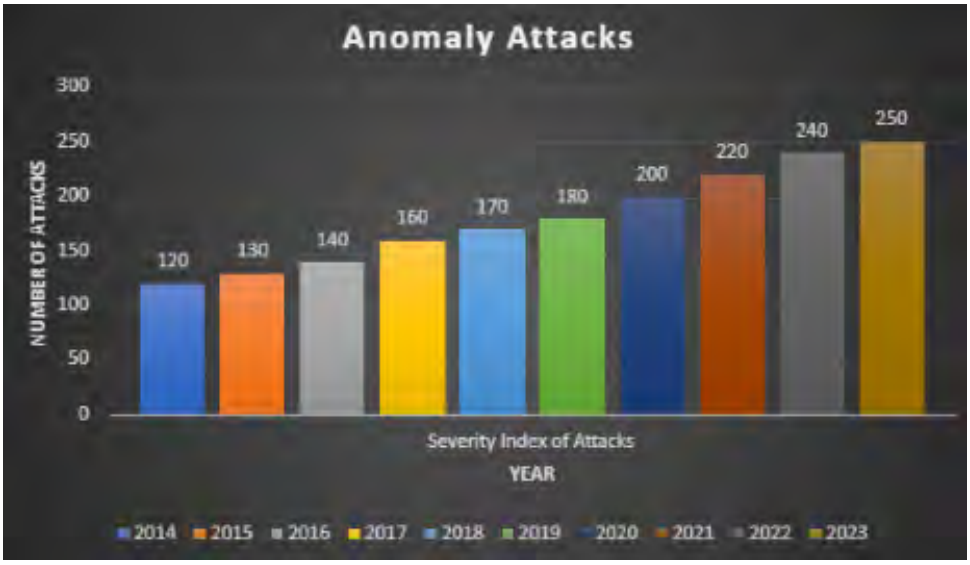

Figure 1.1: Number of Anomaly attacks over the years

Polymorphic malwares are categorized using the CIC MalMem 2022 and CIC-AndMal2017 datasets. In addition to notable malwares like trojan horses, ransomware, and spyware, the CIC MalMem 2022 dataset [7] defines sixteen different forms of malware, including subcategories like Zeus, Emotet, Scar, 180 Solutions, Gator, TIBS, Conti, Pysa, and Shade [8]. A thorough testbed for malware classification is offered by both datasets.

Detecting polymorphic malware is still a difficult task, even with all the research and technological improvements in the field. Because this kind of malware is dynamic, cybersecurity experts need to be always vigilant and creative in order to resist its ever-evolving threats. It is imperative that new detection algorithms and procedures be developed, but doing so is a complex and time-consuming process that frequently resembles an ongoing arms race between cyber attackers and defenders. The challenge of safeguarding digital environments is never fully finished because every improvement in malware detection could lead to a new adaptation in malicious behavior.This cat-and-mouse game requires cybersecurity professionals to be perpetually on alert, updating their skills and tools to match the sophistication of the threats they face. In summary, the battle against sophisticated malware is still a very uphill battle, but our research is making progress in that regard.

The study tested machine learning (ML) algorithms to identify obfuscated malware, revealing that advanced models can significantly improve our ability to detect and stop

sophisticated cyber threats. This research adds to ongoing discussions in cybersecurity and emphasizes the need for strong, adaptable security systems to handle increasingly tricky malware attacks.

According to the study, lightweight ensemble models with the lowest inference cost, such as AdaBoost-LightGBM and Random Forest-LightGBM, may obtain the highest detection accuracy. Random Forest-LightGBM outperforms the other model in real-time deployment, although both models perform well in real-world settings. In order to detect obfuscated malware in binary and multiclass tasks, the study presents a targeted ensemble method that strikes a balance between classification performance and efficiency. This method has been verified on several datasets. This strategy is essential for creating strong defenses against dynamic security threats.

1.1 Research Problem

Traditional cybersecurity detection techniques face major hurdles due to the malware's increasing complexity and quick alterations. The effectiveness of detection approaches is hampered by the constant use of polymorphism and obfuscation by malware developers to evade detection. As a result, more intelligent and adaptable security solutions are required [9][10].

By creating a lightweight, interpretable machine learning system that can identify novel malware types using only a small number of existing malware instances, a researcher seeks to overcome these issues. The system's main goal is to identify disguised malware, which is typically difficult to locate using existing methods [11]. Since this malware may cause operational disruptions, financial losses, data breaches, and even national security threats, identifying it is essential.

The study employs a variety of machine learning models to maximize their ability to detect sophisticated malware threats. The datasets are divided into binary, malware category (4-class), and individual malware families (16-class). Poor generalization, performance deterioration in multiclass tasks, artificial balancing trade-offs, computational bottlenecks, feature sensitivity, and underrepresented classes are some of the fundamental issues that still plague machine learning systems.

So, the research uses many datasets to provide a single framework for robust malware classification in order to overcome these issues. By means of meticulous feature engineering, preprocessing, and lightweight machine learning model construction, it takes into consideration obfuscation techniques, class imbalance, and cross-dataset generalization. In the realm of cybersecurity, this strategy seeks to address the increasing complexity of malware threats and increase detection accuracy.

1.2 Research Objectives

The goal of the project is to use machine learning models to provide a lightweight method of detecting obfuscated malware. It investigates advanced data preparation methods to increase precision and carry out malware multi-class classification. Lightweight and strong, the framework will be able to recognize obfuscated malware in binary, 4-class, and 16-class classification settings.

The steps include:

- Ensemble architecture: Integrate complementary lightweight classifiers to enhance generalization. Confirm Confirm that our ensemble surpasses the performance of individual baselines in both binary and multiclass contexts.
- Dealing class imbalance: Utilize resampling methods such as Random Under Sampler to address unbalanced label distributions. Also, it generates synthetic data and cleans the data of any noise present. Additionally, tackle class imbalance while recognizing the minority class.
- Features reduction: Minimize the feature set by 60-80% to achieve optimal lightweight accuracy in our top models by selecting the most important features.
- Remove unnecessary data scaling: Eliminating duplicate entries, null values, and biased categorical features such as labels and metadata is essential. Failing to address this data can lead to skewed accuracy and inflated rates. Additionally, we implement data encoding and scaling by StandardScaler.
- Evaluate efficiency: Incorporate experiments for both binary and multiclass classification, validate across different datasets, and conduct time profiling. Metrics to report: Accuracy, Precision, Recall, F1-score, AUC. Compare the model's accuracy across different dataset for robustness. Show that our framework satisfies real-world speed and size criteria for deployment.

Chapter 2

Literature Review

2.1 Related works

The advancement of newest malware has exposed the limitation of traditional malware detecting methods. Therefore, several machine learning and deep learning techniques are employed to overcome those issues. This section summarizes the studies that used several models, emphasizing the distinct approaches and important discoveries and techniques.

2.2 Literature Review

2.2.1 Foundational Concepts and Background

Malware detection and classification used [12] by machine learning is examined in the research "Machine Learning for Malware Detection and Classification" by Olga and De Catalunya, Department of Engineering and Telematics (2023).In their paper,they used two types of datasets which are CIC-MalMem-2022 as labeled dataset and CIC-Evasive-PDFMal2022 as unlabeled dataset.Because of being labeled with malware type,they only focus on CIC-MalMem-2022 dataset. Hence, based on their objectives, they utilized Decision Trees, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes, and Random Forest models, which are the primary subjects of this discussion. As per the CIC-MalMem-2022 dataset, the Random Forest model attained the highest overall accuracy in classification tasks. It performed 100% for benign malware, 74% for ransomware, 77% for spyware, and 73% for trojans. KNN also performed well with 100% accuracy for benign, 69% for ransomware, 73% for spyware, and 67% for trojans. While Naive Bayes showed lower performance, especially for detecting specific malware types. It classified 98% benign but only 24% ransomware, 32% spyware, and 72% trojans. The primary challenge observed was the dataset imbalance which led to misclassification in minority malware categories. As a result, in future,they wished to work on both balancing the dataset and exploring deep learning techniques to improve accuracy.

Moiez et al. (2025) performed [13] research on the topic of "Binary and multi class malware classification of windows portable executable using classic machine learning and deep learning". For binary classification,they used two publicly available dataset consisting of PE files from Kaggle resources containing 138,047 data size with 57 features and 19,611 data size with 78 features accordingly . For multiclass,they also choose a publicly

available dataset of 29,807 samples with 54 features.Data balancing,feature scaling and feature selection are done for data pre-process steps.For binary class, Random Forest gives the highest accuracy with average 99.37%.Besides, other models like CNN,CNN-LSTM are close to Random-Forest . On the other hand, for multi-class classification, again Random-Forest gives the highest accuracy with 88.76% where CNN-LSTM is the closest one with 81.24% .In their research,the main challenge is multi class classification accuracy which is comparatively low. As a result, in future, the researchers wished to focus on making new the algorithms that gives best result along with detects zero-day attacks.

Rabia et al. (2025) have experimented [14] on a robust Machine Learning based binary class classification model for detection and analysis of malware attacks.For this,they have used three datasets named PE_Header,PE_Sections and DLLs_Imported from figshare.Each datasets consist of more than 29 thousand rows and 50 features.Besides,they resolve data balancing problems along with feature selection to make the models more accurate. Decision-Tree, KNN and Random Forest are used for this experiment.Among them, Random forest achieve highest accuracy of 98.13% where decision tree is 96.40% and KNN is 90.01% .According to their comparative analysis,their proposed model MLBM-DAM model makes the highest accuracy of 98.13%.In future,they want to use Deep learning algorithms to enhance the performance of their model.

In the publication "Malware Detection Using Machine Learning and Deep Learning" by Rathore et al. (2018) ,[15] both machine learning and deep learning techniques were employed to identify malware. The dataset utilized in this study comprises 11,688 malware samples, which correspond to 55 malware families. The benign samples numbered 2,819. In this system, a total of 14,507 samples were sourced from this dataset. The researchers employed a Random Forest model, a deep neural network that consisted of three layers: a two-layer DNN, a four-layer DNN, and a seven-layer DNN. Also, an autoencoder model was implemented; however, it failed to demonstrate satisfactory performance. Additionally, they introduced techniques for feature extraction, dimensionality reduction, cross-validation, and class imbalance management.The system achieved 99.78% accuracy in random forest, while it achieved 98.99% accuracy in DNN, which is slightly lower than random forest. Autoencoders were also less effective due to the challenges that the researchers encountered in this paper, which included managing the vast number of features. This research proposed employing long short-term memory (LSTM) and recurrent neural networks (RNNs) to attain optimal performance. Furthermore, the model was enhanced to recognize diverse categories of malware, encompassing both polymorphic and metamorphic variants.

Masabo et al. [16] have devised a method for the identification of polymorphic malware that employs the K-NN algorithm and structural feature engineering. They employed static analysis to extract structural characteristics such as the number of sections, packaging state, entropy, code size, and digital signature status. The following tools were employed for feature extraction: IDA Pro, PE Explorer, VirusTotal, and DetectItEasy. Furthermore, correlation-based feature selection (CFS) was implemented to ensure that

the predictive power was satisfactory. The K-NN approach was employed to construct a detection model with a detection accuracy of 98.7% and a false positive rate of 0.014%. A 10-fold cross-validation method was used in a controlled virtual environment to assess the model. The research highlights the challenges of concealment techniques like packing and encryption in static analysis. Further research will involve larger datasets, more classifiers, and integrating behavioral variables from dynamic analysis.

Azeem et al. [17] concentrated on the utilization of machine learning models to improve the detection and classification of malware. The objective of the initiative was to enhance cybersecurity malware detection systems by utilizing the UNSW-NB15 dataset. The research involved the preprocessing of the dataset, the use of feature selection methods such as TFIDF, the balancing of the dataset, and the training of machine learning models for classification. The Random Forest model achieved a recall, precision, F1-Score, and accuracy rate of 97.68% on a reliable dataset, surpassing the other models. In order to extract valuable characteristics from popular images, Convolutional Neural Networks (CNNs) were implemented. In binary classification, the Extra Tree (ET) algorithm achieved an accuracy of 99.98%, surpassing the GBM and LR algorithms. The study also recognized the potential for accuracy limitations as a result of the dynamic nature of malware and the constantly changing techniques employed by hackers. This underscores the necessity of consistently improving and adapting detection methodologies. The investigation furnishes data regarding the potential enhancement of malware detection systems' accuracy and effectiveness through the implementation of deep learning algorithms and feature selection strategies. The ultimate goal is to enhance prediction models by incorporating new and significant characteristics and to recommend additional research on the application of Deep Learning and Hidden Markov Models by utilizing the ET classifier dataset.

For the purpose of detecting compromised malware, the authors Carrier et al. produced the CIC MALMEM2022 dataset [18]. Additionally, they pioneer several methods and models to boost detection accuracy. They are having difficulty, first, with the intricacy and time required for memory analysis and urging to investigate machine learning. Additionally, they used a two-layer stacked ensemble learning model that included models such as Random Forest, Decision Tree, and Naive Bayes to detect 99 percent of malware kinds, including trojan horse, ransomware, and spyware. With 2500 malware samples, Datawet guarantees real-world dangers. However, they are only able to improve binary classification; they are unable to provide multiclass classification results that are satisfactory. Future research goals emphasize the need for ongoing innovation by aiming to increase malware detection skills via the use of feature extraction approaches and the VolMemLyzer tool.

To categorize IoT security data while minimizing computational expenses, the author of "Deep-Learning-Based Approach for IoT Attack and Malware Detection" by Tasci (2024) [19] introduces an enhanced 1D Convolutional Neural Network (1D CNN). Using GELU activation, self-attention, dropout, and normalization, the 75-layer model was trained on the CIC-MalMem-2022,CIC IoT 2023 and CIC-IDS2017 datasets.The CIC-

IDS2017 dataset featured 15 distinct types of network attacks, while the CIC-MalMem-2022 dataset comprised 58,596 memory dumps, with 50 percent being malware. The CIC IoT 2023 dataset contained over 22 million samples classified into 34 categories, including attacks such as SQL injection, DDoS, and MITM. The model proposed achieved an accuracy of 99.90% on the CIC-MalMem-2022 dataset, 99.99% accuracy on the CIC-IDS2017 dataset, and demonstrated 98.36% accuracy on the CIC IoT 2023 dataset, with a precision of 100%, recall of 99.96%, and an F1-score of 99.95%. It outperformed more conventional models such as SVM, kNN, and Decision Trees. Performance decreased, nevertheless, in more complex classes like DNS spoofing and backdoor malware. Larger dataset testing and adaptive learning will be incorporated into future research to improve real-world deployment in IoT scenarios.

The CIC-MalMem-2022 dataset includes memory dump [20] samples from different malware families. In their research titled "Hide and Seek in Memory: Outsmarting Sneaky Malware with Data Magic," Rakib et al. (2025) investigate multiclass malware classification. The authors evaluate a range of machine learning techniques, including Decision trees,support vector machines, ensemble methods and neural networks. To address the high class imbalance in the dataset, they employ a combination of undersampling approaches (Random Undersampling,Edited Nearest Neighbor, Near Miss, and All KNN) and synthetic oversampling with ADASYN. Despite emphasizing excellent multiclass classification performance, the publication does not disclose exact accuracy or F1 scores in the summary. The study's primary strengths are its robust imbalance management, repeatability using publicly available code and data, and its useful memory-dump-based technique. However, its main shortcomings are the absence of detailed numerical findings in the paper text and the lack of per-class performance breakdowns, which are crucial for detecting classifier errors. The authors suggest that future studies focus on more intricate deep learning architectures and test models on more realistic obfuscation scenarios to improve generalizability.

2.2.2 Advanced Detection Techniques and Challenges

The comprehensive investigation of the detection and classification of obfuscated privacy malware using memory dumps is conducted in the study "Obfuscated Privacy Malware Classifiers Based on Memory Dumping Analysis" by Cevallos-Salas et al. (2024) [21]. The study used the CIC-MalMem-2022 dataset. It contains 58,596 observations which is only balanced for binary classification but not for multi class classification .For this,they use SMOTE technique for data imbalancement. In working, logistic regression was applied for binary classification and deep neural networks (DNNs) for multiclass classification. The classifiers target four malware categories named benign, spyware, ransomware, and trojan. It also extends to family-level classification for obfuscated privacy malware. The highest accuracy achieved for binary classification was 99.7% using logistic regression. For family classification, the DNN classifier achieved a mean accuracy of 76.9%, with accuracy for specific malware categories being 100% for benign, 74% for ransomware, 77% for spyware, and 73% for trojan. However, multiclass classification was more challenging. Its performance becomes lower due to data complexity and imbalance issues. In the future, researchers expect that future research will be based on the exploration of deep learning methods and the use of novel pre-processing techniques like SMOTE and dataset which will be developed more specific with organised qualitiful data .

The research titled "Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications" by Shafin et al. (2023) explores [22] how deep learning techniques can be utilized to identify malware on embedded IoT devices. The 58,596-sample containing CIC-Malmem-2022 dataset is used in the research. They propose a hybrid based model to identify and classify obfuscated memory malware (OMM) into four categories: trojans, ransomware, spyware, and benign. The hybrid model integrates Bidirectional Long Short-Term Memory (Bi-LSTM) with Convolutional Neural Networks (CNN). In binary classification tasks, the suggested models, CompactCBL and RobustCBL, had accuracy rates of 99.92% and 99.96%, respectively.On the other hand,for multiclass family detection,RobustCBL reaches 84.56% accuracy which is highest among all other models and CompactCBL reaches 84.22% . The primary challenge highlighted in the paper is the difficulty of distinguishing between different malware types due to their obfuscated nature. Future work is aimed at improving classification accuracy for individual attack types. Also, exploring semi-supervised learning models to address emerging malware threats in real-time IoT applications.

The paper "Detection of Obfuscated Malware using Ensemble Learning Techniques" by Kiruthika et al.(2024) develops [23] the idea of detecting obfuscated malware specially by using machine learning techniques. The researchers used to stack ensemble models and boost ensemble models to detect accuracy. Random forest, Decision tree, KNN,naive bayes were used to implement stacking ensemble models that helped the model to achieve on average 99.99% accuracy. Besides,Adaboost, XGB and HGB were the boosting models used to achieve on average 100% accuracy. The dataset that they worked on has 58,596 records having 29,298 of them malicious, also containing 57 features. Wrapper based feature selection was used to identify 10 features from the dataset where the forward selection method gained 94% and the backward selection method gained 91% at accuracy. The boosting model turned out to be the most effective in this instance, and the

researchers also noted that their study method updates codes, which are more sophisticated than standard ways, so it presents a problem. The paper also suggested the usage of deep learning methods in future. Overall the paper contributes success in detecting malware and gaining a good accuracy rate in performance score.

Madamidola et al. (2024) [24] developed a system with machine learning model which focuses on malware unencountered subtypes of obfuscated malware variants [7] .To perform,they use CIC-Malmem-2022 dataset with a dataset of 58,596 records. The system uses Random Forest (RF) along with 6 other models for binary classification and successfully achieves 100% accuracy. The system is also trained on only one malware, transponder, but can detect 14 other unknown types using the Zero Shot Learning Approach (SHAP). The system achieved 99.8% accuracy, with adaptive capabilities. The main challenge is detecting malware that changes its behaviour. The study suggests using hyperparameter tuning to improve performance in the future and testing the system on different unknown malware types to generate its capabilities. The improved and highly accurate system is a promising approach for efficiently detecting malware and achieving their goals.

1.Six traditional machine learning models—Logistic Regression, SVM, KNN, Random Forest, Naive Bayes, and Decision Tree—were used to offer a static malware detection framework in the study "Malware Classification Using Machine Learning Models," by Kumar et al. (2024).The dataset was used CIC Malmem 2022 which contained 29,298 malware and 29,298 benign samples, totaling 58,596 samples with 57 characteristics apiece. To improve model efficiency, the authors used correlation-based feature selection with a threshold of 0.6 to reduce the features to 18. The Decision Tree and KNN models performed best after preprocessing and 10-fold cross-validation with an 80:20 train-test split. 99.98293% testing accuracy, 1.0 precision, 0.9996578 recall, and 0.9998288 F1-score were recorded by both of them. Their findings were superior to those of earlier studies, such as CNN (99.1%) and Random Forest with PSO-GA (89.74%). Despite the impressive results, the study's limitations include its sole dependence on static features, examination of a single dataset, and lack of deployment considerations. The authors suggest including dynamic characteristics, exploring ensemble or deep learning models, and performing cross-dataset testing in future studies to improve robustness and practicality.

According to Sharmila et al.(2023), obfuscated malware has lately become one of the most important problems facing the whole planet. With Random Forest machine learning applied to the CIC MalMem2022 dataset,[25] the author achieved 99.98% in binary classification and 89.07% in multiclass classification. Additionally, the ANN and SVM models archive 99.72% and 99.88% of data, respectively. In order to determine metrics for comparison and further analysis, they used the SoTA model in binary and multiclass classification using the dataset after classifying the malware as spyware, trojan, and ransomware. Additionally, they achieved the greatest results in binary classification but were unable to get the desired results in multiclass classification. Future research is necessary to enhance feature engineering, hybrid models, real-time detection systems, and continuous learning processes for improved cybersecurity. Obfuscation and model generalization

are two issues that researchers desired to address later.

Focusing on enhancing malware detection, proposed by Rakibul et al. (2025),[26] show the efficacy of different feature selection and feature scaling methodologies along with different machine learning models. A publicly available dataset named MalDroid 2020 is used for this which carries 5 categories of malware that are Benign,Banking malware,SMS malware,Adware and mobile Riskware.Then,for the pre processing part,they use SMOTE technique,PCA and LDA, min-max scaler etc.Besides, ML models like Logistic Regression,Decision Tree,Naive Bayes,Random Forest,GBM,LGBM and some others are used. In result , they find that without feature selection and scaling, LGBM gives the highest accuracy of 95.99% where RF and ET are near . Moreover, with only normalization scaling is added, again LGBM gives the highest accuracy with 95.82%. After PCA is added, LGBM's accuracy is somewhat increased and stands at 97.16%. RT and ET consistently remain closer in every step.Generalizability is limited by this paper's reliance on a single Android malware dataset, and feature selection was not effectively done using PCA and LDA. In addition to ignoring efficient techniques, the article performed poorly on comparisons involving deep learning models, which are crucial for the real world.Lastly, researchers wish to try on more diverse datasets along with Deep Learning models like CNN or transformers to optimize.

Bandhon et al.(2025) have researched [27] on Early detection of Ransomware threat attacks in Enterprise Networks using Machine learning. They have focused on feature extraction techniques, different kinds of Machine learning classifiers,dataset quality etc.They have used multiple datasets named CICIDS 2017 Ransomware dataset , CSE CIC-IDS 2018 dataset and open malware dataset(Kaggle).They don't go for exploring multiclass classification rather focus on binary classification.Thus, models like SVM,Random Forest,DNN,Logistic Regression and KNN are used .In result , SVM gives the highest accuracy of 96.5% and other closest were 95.6% for DNN, 94.8% for Random Forest. At last,researchers tend to show interest in using techniques like Bayesian optimization and automated machine learning(AutoML) to enhance accuracy.Moreover,they plan to use Deep Learning and ensemble learning methods to make their real time detection system more accurate and reliable.

Here, Roy et al. demonstrated [28] a variety of methods and strategies for raising the dataset CIC MalMem 2022's overall accuracy. They developed hybrid models that combined many categories, such as the MalHyStack framework. Additionally, this model showed remarkable accuracy in multiclass and binary classification, with rates of 85.04% and 99.98%, respectively. However, even though they were outperforming others in multi and binary, they were still expecting better results from 4 classes, while 16 classes did not meet their expectations. Accordingly, their recommendations for future research concentrate on enhancing model effectiveness via automated hyperparameter tweaking and investigating new datasets to verify generalizability, raise detection rates, and lower false negatives in practical applications.

Shah et. al (2024) in their paper named "Malware detection using deep learning approaches [29].In their paper, it portrays different models and techniques to detect malware detection and preventing them. The authors mentioned that the dataset here is the 2018 Ember dataset which contains 1.1 million files which have both malicious and benign labels. The authors used CNN,LSTM and Neural network models here(both solo and hybrid). CNN-1D(One-Dimensional Convolutional Neural Network) achieved 98.89% accuracy. The hybrid model of CNN and LSTM has 98.86% accuracy on GPU and we get the best performance from the neural network model which has 99.91% accuracy. The challenges on this paper was that the CNN model was really slow from all the models and the models struggled coping up with unseen malware attacks or ZERO DAY ATTACKS.The author suggested some future plans which includes data augmentation,ensemble learning and transfer learning for better accuracy and performance.

In the study "Enhanced Detection of Obfuscated Malware in Memory Dumps: A Machine Learning Approach for Advanced Cybersecurity," Hossain et al. (2024) established a strong ensemble-based framework for obfuscated malware detection using the Obfuscated-MalMem2022 dataset [30]. The 16 malware families in this dataset are divided into three groups: Trojan horses, spyware, and ransomware. There are 58,596 samples total, split evenly between malicious and benign memory dumps. The authors selected features using Chi-square and mutual information, scaled the data using MinMaxScaler, and addressed class imbalance using SMOTE. The models that were assessed were AdaBoost (ADB), Random Forest (RF), Gradient Boosting (GB), Bagging (BG), and Voting (VT). In both binary and multi-class classification tasks, the GB model attained perfect scores of 100% in accuracy, precision, recall, F1-score, and AUC, particularly following SMOTE balancing. When compared to models such as RobustCBL (72.60% accuracy) and MalHyStack (66.96% for 16-class detection), it significantly surpassed previous research findings. The main advantage of the proposed model lies in its flexibility for both binary and multiclass detection situations, even though its dependence on memory dumps constrains real-time implementation. Future research could apply this model to operational systems and explore its performance on hybrid datasets for increased generalizability.

A novel malware detection framework called MeMalDet addresses temporal bias, a common flaw in existing malware detection models, in the paper "MeMalDet: A Memory Analysis-Based Malware Detection Framework Using Deep Autoencoders and Stacked

Ensemble under Temporal Evaluations" (2024) [31] by Pascal Maniriho, Abdun Naser Mahmood, and Mohammad Jabed Morshed Chowdhury. They introduce MemMal-D2024, a newly improved memory-based dataset that was taken from the Carrier et al. (2022) dataset and improved with timestamp features to aid in temporal evaluation. MeMalDet uses deep autoencoders for unsupervised feature extraction and a stacked ensemble of supervised classifiers (SVM, SGD, and Logistic Regression with CatBoost as meta-classifier) to achieve high detection performance. With up to 99.85% accuracy under random split and up to 98.82% accuracy under temporal split, it demonstrates great applicability to unknown obfuscated malware. Its performance slightly deteriorates when tested on more current malware (such that from 2021), with the lowest accuracy of 94.08%, underscoring the challenges of evolving malicious patterns. The model demonstrates how important it is to test malware detectors in realistic, time-consistent circumstances and performs noticeably better than existing methods. The authors suggest future improvements including online learning and integration with other detection methods to further increase resilience and adaptability.

In the paper "Detection and Analysis of Malicious Software Using Machine Learning Models," published [32] in the Sakarya University Journal of Computer and Information Sciences, Öztürk and Hızal (2024) examine the effectiveness of machine learning in identifying obfuscated malware using the CIC-MalMem-2022 dataset. This dataset, which includes 58,596 memory dump samples evenly split between 29,298 malicious and 29,298 benign occurrences, covers 15 malware families (Trojan, Spyware, and Ransomware). The dataset was preprocessed and duplicates were eliminated, resulting in 58,027 samples. The study performed binary, multiclass (4), and multiclass (16) classifications using Random Tree (RT), Random Forest (RF), J-48, Naive Bayes (NB), and XGBoost. In binary classification, 99.99% accuracy, precision, recall, and F1-score were attained by RF, J-48, and XGBoost. The accuracy of XGBoost was 75.49% for multiclass (16) and 87.79% for multiclass (4). NB did worse in multiclass scenarios. Several recent investigations, like MalHyStack, were outperformed by their model (70.29% for 16-class). Among the drawbacks are moderate F1-scores for difficult multiclass instances and the absence of real-world deployment testing. Future studies will use ensemble learning and hyperparameter adjustment to boost robustness and enhance classification in hostile contexts.

2.3 Existing Result Review

There are a lot of existing models for malware classification which has started from very past and continues it till now.Since it has started, researchers always continue their trying to discover new malware types according to the real world malware attacks.Thus, a continuous updating process is driven all over the cyber fields.Thus, researchers have already showed better performance on revealing both binary classification to multi class classification.Our main focus of research is in acquiring higher accuracy rate than the existing models accuracy for both binary and multiclass classification with our primary dataset CIC-MalMem-2022 along with some secondary dataset named Ransomware 2024 and CIC MalDroid 2020.So, to get the view of existing works,here are some recent accuracy chart for different datasets and their different class classification are given below

2.3.1 CIC MalMem 2022 Dataset

The following chart (Öztürk et al., 2024)[32] lists all the tracks of the CIC-MalMem-2022 dataset up to August 2024.

Authors	Model Tested	Best Model	Accuracy (%)		
			Binary	Multiclass (4)	Multiclass (16)
Carrier et al. [1]	NB, RF, DT and meta-learner-R	LR	90.00	-	-
Vihari et al. [3]	Hierarchical Detection Model	DNN	99.67	-	-
Ghazi [4]	RF, SVM, KNN, NaiveBayes	RF	99.92	-	-
Dener et al. [33]	RF, DT, MLP, KNN, SGD	LR	99.94	-	-
Roshan [7]	NB, Adaptive RF, KNN, Voting	EnsAdaRF	99.83	-	-
Meizona[24]	LR, DT, MLP, KNN, SVM, DCNN	D-CNN	99.93	83.53	-
Cevikbas et al. [11]	DT, RF, SVM, LDA, GNB	LR-DNN	99.70	75.40	68.20
Shafii et al. [19]	RobustCBL, CompactCBL	RobustCBL	99.96	84.56	72.60
Abuhay et al. [30]	KNN	KNN	99.97	82.21	69.63
Roy et al. [25]	Extra Trees, XGBoost, SVM, KNN	MalHyStack	99.98	85.04	70.99
Ozturk et al. [32]	RF, RF+J48, NaiveBayes, XGBoost	XGBoost	99.99	87.79	75.49

Table 2.1: Performance Comparison of Models on CIC-MalMem-2022 Dataset

After August, 2024 there are some works in this dataset which are also given below with mentioning proper class.

Serial	Author	Year	Dataset	Classification Type	Best Model	Accuracy
1	Zhen et al. [34]	2024	CICMalMem22	Binary (2-class)	Random Forest-AE	99.892%
2	Emad et al. [35]	2025	CICMalMem22	Binary (2-class)	CNN-DN + N hybrid	99.5%
3	Mahmood et al. [36]	2025	CICMalMem22	Binary (2-class)	One-Class SVM	99.4%
4	Zeki et al. [37]	2025	CICMalMem2022	Multi-class (4-class)	FFN MC	84.37%
5	O. Kırlar et al. [37]	2024	CICMalMem2022	Multi-class (16-class)	XGBoost	75.53%

Table 2.2: Summary of Classification Approaches on CICMalMem2022 Dataset

2.3.2 Ransomware 2024 dataset

Serial	Author	Year	Dataset	Classification Type	Best Model	Accuracy
1	Liu Qingzhong [38]	2025	Ransomware 2024	Binary (2-class)	Focus on only features	99.45%
2	Qingzhong Liu [38]	2025	Ransomware 2024	Multi-class (5-class)	Focus on only features	95.19%

Table 2.3: Classification Results on Ransomware 2024 Dataset

2.3.3 CIC MalDroid 2020

Serial	Author	Year	Dataset	Best Model	Accuracy
1	Rakibyl et al. [39]	2025	MalDroid 2020	LightGBM	97.16%
2	Samaneh et al. [40]	2020	MalDroid 2020	Semi-Supervised PLDNN	97.84%
3	Padmavathi et al. [41]	2022	MalDroid 2020	K-means, PCA	88%

Table 2.4: Multi-class Classification Results on CIC MalDroid 2020 Dataset

Chapter 3

Datasets

3.1 Data Analysis

For our experiment,we have used 4 datasets to establish our models.These datasets are CIC MalMem-2022 ,Ransomware 2024, CIC MalDroid 2020 and CIC AndMal 2017.Among them,CIC MalMem-2022 is our primary dataset where we mainly establish our model and rest all are secondary datasets which are for further testing of how our model works .

3.1.1 CIC MalMem-2022 dataset

The primary dataset used in this investigation is CIC-MalMem-2022 [5], which was obtained in October 2022 from the Canadian Institute for Cybersecurity. This organization is well recognized for its substantial contributions to cybersecurity research. For the purpose of generic malware detection, the first label—binary—distinguishes between samples that are harmful and those that are benign. It contains 58,596 records in total with 29,298 benign and 29,298 malicious as shown in figure 3.1. We have further discovered that of the malicious categories there are subcategories consisting of 16 more malware families shown in figure 3.2.

We have also seen that the dataset is fairly balanced for Binary classification but for malware category (4-class)and malware family classification (16-class) the dataset is imbalanced as shown in figures 3.3, 3.4 and 3.5.
The dataset contains 57 features in total with one feature named Class and another named category, which were dropped and labelled as y. The rest 55 features are all numerical data and did not require label encoding.

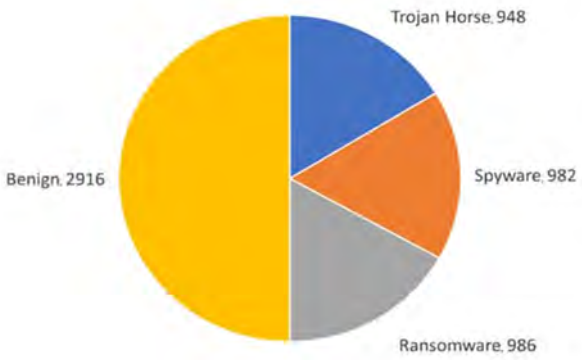

Figure 3.1: Malware Category

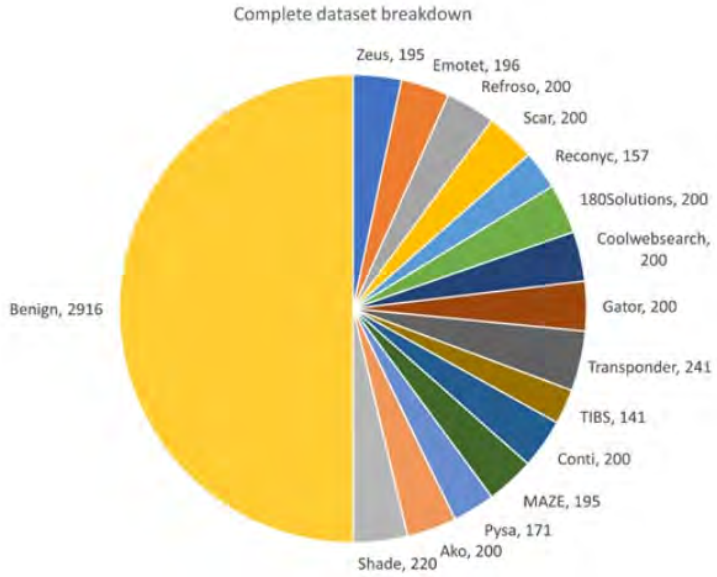

Figure 3.2: Malware Family

To check the robustness of our model we have also trained and tested other datasets. They are:

3.1.2 Ransomware 2024

This dataset was created in October 2024 focusing on malware analysis of 5 categories along with 26 family types and Benign.This dataset contains a total of 21752 samples where both malicious and benign exist equally of 10876 samples.Moreover,there exist 5 malware categories which are Trojan, Spyware,Adware,Ransomware and Benign.This dataset helps us to confirm that our proposed model works not only our primary dataset but also other types of datasets as well.

3.1.3 CIC MalDroid 2020

This dataset contains more than 17341 samples on Android with 4 different categories of malware along with Benign.The categories are Adware,Banking Malware,SMS Malware,Mobile Riskware and Benign.Our model can classify this type of malware classification with a good accuracy score .To do so,we have selected this dataset as for testing purpose .A detailed chart of categories with details is given

- Adware: 1,253
- Banking: 2,100
- SMS malware: 3,904
- Riskware: 2,546
- Benign: 1,795
- Total: 11,598

3.1.4 CIC AndMal 2017

This is our last dataset which is also chosen to test our model in different types of malware types.This is a dataset actually made for Android Malware Types.There are more than 10854 samples where 4354 are malware and rest 6500 are benign.

3.1.5 Datasets Overview

Dataset Name	Origin	Usage	Samples	Total Features	Classification by us
MalMem-2022	CIC	Primary	58596	57	2 class, 4 class, 16 class
Ransomware-2024	Amjad et al	Secondary	21752	77	2 class, 5 class
MalDroid-2020	CIC	Secondary	17341	470(dynamic)	5 class
AndMal-2017	CIC	Secondary	10854	5 class	2 class

Table 3.1: Overview of the datasets

3.2 Dataset Preprocessing and Cleaning

Our methodical approach to data pre-processing and cleaning aims to optimize the CIC-MalMem-2022 dataset for precise machine learning outcomes in order to set it up for rigorous analysis. We verified the data was clean, well-organized, and suitable for advanced analytics by following a set of procedures for Binary classification, but for malware category and malware family the data was imbalanced and needed preprocessing.These steps formed the foundation of our data preparation, setting the stage for deploying complex machine learning models to classify and analyze malware effectively.

3.2.1 Data Cleaning

The dataset contained no null values but quite a number of duplicate rows which were discarded.

3.2.2 Label Encoding

Label encoding is a preprocessing technique that is used to convert categorical features into numerical data for effective usage by machine learning models. It is easy to implement and is also efficient computationally. Unlike one-hot encoding it uses only one column, not increasing the dataset dimensions. In our dataset, we have had to label encode three features; Class, category and category-name for the malware family of 16-classes which are binded with the 'category' feature in the dataset.

3.2.3 Removal of irreleant features

Our primary dataset contained no irrelevant features. However, the datasets used for generalising the model contained meta data, which were irrelevant and were discarded from the dataset so that biases of the model could be reduced.

3.2.4 Data Balancing

The dataset CICMALMEM- 2022 is imbalanced for multi-class classification. For binary classification the dataset is fairly balanced with one half benign and another half malicious. With the imbalanced dataset for malware category and malware family, the accuracies were seen to be near perfect. This is because the model was biased towards the benign samples due to its significantly high number of samples. Hence, we have attempted to balance the dataset to ensure a realistic accuracy of our model

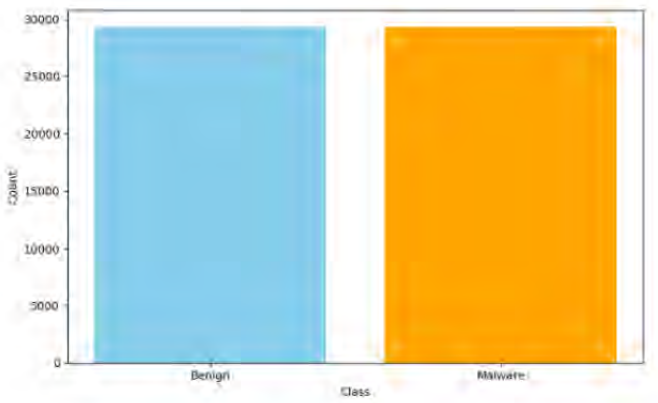

Figure 3.3: Balanced Binanry

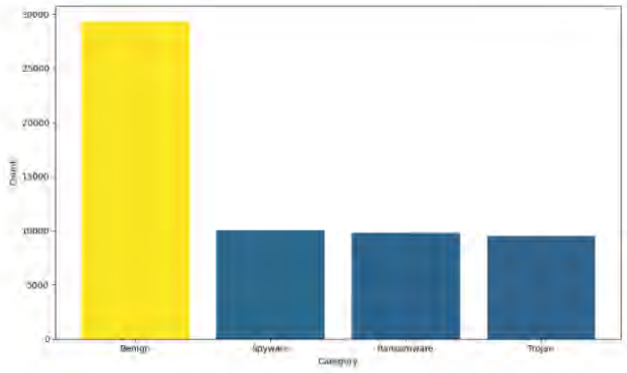

Figure 3.4: Imbalanced malware category (4-class)

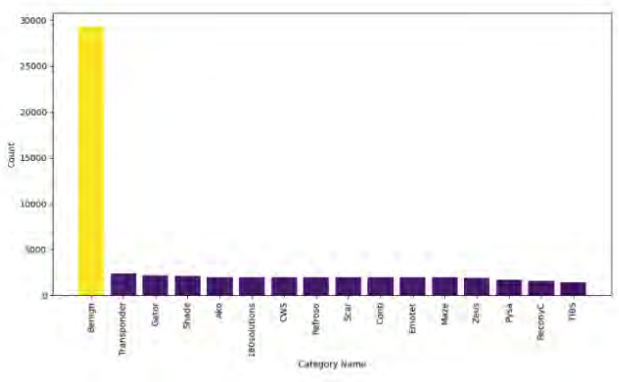

Figure 3.5: Imbalanced malware family (16-class)

The data imbalance creates a bigger issue for the classification of the malware families (16- class). Where the number of benign samples is exponentially higher than the other 15classes of malware. Hence, we have attempted to use three sampling strategies, Random Undersampling, Synthetic Minority Oversampling Technique (SMOTE), and SMOTE with Edited Nearest Neighbors (SMOTEENN). Our proposed strategy is the random undersampler as it has outperformed SMOTE and SMOTEENN in terms of model accuracy. The outperformance of random undersampler could be because it efficiently reduces the majority class samples without introducing synthetic data, thereby retaining the original data distribution and reducing the risk of overfitting in machine learning models. We have evaluated our model once on balanced and again on imbalanced dataset.

3.2.5 Feature Selection

For feature selection we have first taken the top 20 most important features from random forest model using feature_importance method, and top 20 most important features from lightGBM and merged those two sets based on the common features present in them. We have then trained our model using those combined features to give us a desired outcome. For CICMalmem 2022, PCA and Heatmap tend to remove important features necessary for our model to learn from and make a prediction.

3.2.6 Feature Scaling

Feature scaling has been done on the dataset to ensure that features that are out of range contributes equally to all the machine learning models. We applied StandardScaler to standardize features by subtracting the mean and scaling to unit variance. Without this, features with larger magnitudes might excessively affect the model. Mathematically:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

Here, z is the standardized value, x is the actual data point, μ is the mean of the dataset, and σ is the standard deviation. When models are run, they often assume that the features are centered around zero and exhibit similar variance. Hence, feature scaling is a crucial preprocessing step to obtain accurate and consistent results.

3.2.7 Data Split

We divided our dataset into two parts: 80% for training the model and 20% for assessing its performance following the training process, utilizing the train_test_split function from the sklearn library.

Chapter 4

Methodology

4.1 Workflow

Figure 4.1: Working process of Datase

4.2 Model Analysis

4.2.1 Logistic Regression

Logistic regression is a classical statistical model which is primarily for binary classification problems. It is best known as efficient, simple, and easy interpretability. These functionality making it a foundational model in machine learning. Logistic regression works by modeling the probability of a binary outcome using a logistic (sigmoid) function, which transforms any real-valued input into a probability score between 0 and 1. The logistic function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

Where x represents the input, and f(x) is the predicted probability of the positive class. The outcome is classified as positive if the probability exceeds a pre-defined threshold, typically 0.5. Logistic regression is computationally efficient and can be applied to large datasets with many features, making it highly suitable for classification problems involving diverse datasets like maland2017, CIC-MalMem-2022, and the Ransomware Dataset 2024.

While logistic regression is excellent for binary classification, it can also be extended to multiclass classification problems via techniques like one-vs-rest (OvR) or multinomial logistic regression. In the context of malware classification, logistic regression can be used to classify whether a sample belongs to a particular malware class or to benign software. Its major strength lies in its interpretability: the model coefficients provide insight into the relationship between the features and the class labels.

The model's limitations include its assumption of linearity between the independent variables and the log-odds of the target variable. Additionally, logistic regression may struggle with highly imbalanced classes, a common characteristic in cybersecurity datasets. Despite these limitations, logistic regression is a solid choice for initial benchmarks, providing quick and interpretable results that help in understanding data patterns.

4.2.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning model commonly used for classification tasks, particularly when the data is non-linearly separable. SVM works by finding the optimal hyperplane that separates different classes in the feature space, maximizing the margin between them. The larger the margin, the better the generalization ability of the model. In the case of binary classification, SVM can be expressed as:

$$maximize \frac{2}{\|w\|} \tag{4.2}$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i \tag{4.3}$$

where ω is the weight vector, $\mathbf{x}_i$ is the feature vector, and y_i is the class label of the i-th sample. For multiclass classification problems, SVM can employ strategies such as one-vs-one or one-vs-rest, enabling it to effectively handle datasets like *Maland2017* and *CIC-MalMem-2022*.

SVMs are known for their robustness, particularly in high-dimensional spaces, which is a characteristic of malware datasets with numerous features. However, they also have some limitations, including a high computational cost, especially with large datasets and non-linear kernels. SVMs can also be sensitive to the choice of hyperparameters, like the regularization parameter CCC and kernel function. Despite these challenges, SVM remains a popular choice due to its strong theoretical foundation and its ability to handle both linear and non-linear data separations, providing strong performance in malware classification tasks.

4.2.3 Decision Tree (DT)

Decision Tree (DT) is a non-linear classifier that builds a model based on decisions made at nodes, where each node splits the dataset based on feature values. The process continues recursively until a stopping criterion is met, resulting in a tree structure with leaf nodes representing class labels. A decision tree can be used for both classification and regression tasks, but for binary and multiclass classification, it is highly effective. The basic structure of a decision tree can be described as follows:

- Root Node: The starting point of the decision tree, which holds the complete dataset.
- Internal Nodes: Nodes that represent decisions made based on feature values.
- Leaf Nodes: Terminal nodes that hold the predicted class labels.

DT models are intuitive, easy to interpret, and computationally efficient. They can easily handle both binary and multiclass problems, making them suitable for malware detection tasks like those involving the Ransomware Dataset 2024. The decision-making process is highly interpretable, which makes decision trees valuable in cybersecurity for understanding why certain decisions (e.g., malware classification) are made based on features.

However, decision trees tend to overfit when they are too deep, which can lead to poor generalization on unseen data. Techniques like pruning, where branches of the tree that provide little additional value are removed, can help mitigate overfitting. The use of ensemble methods like Random Forest or Boosting can further improve performance by combining the results of multiple decision trees. Decision Trees work well with datasets having a mix of numerical and categorical data, which is often the case in malware classification datasets.

4.2.4 Random Forest (RF)

Random Forest (RF) is a popular machine learning technique that belongs to the family of ensemble methods, where multiple models (in this case, decision trees) are combined

to improve performance. Instead of relying on a single decision tree, RF builds a forest of trees, where each tree makes an individual prediction, and the final prediction is determined by aggregating the results of all trees. The general idea is that by combining the predictions of many trees, Random Forest can make more accurate and stable predictions, reducing the risk of overfitting that is common with single decision trees. Each

decision tree in the forest is trained on a random subset of the data, and at each decision node, a random subset of features is considered. This randomness helps ensure that the trees are diverse and not overly correlated with each other, which improves the overall model's ability to generalize well to new data. The final prediction is made by majority voting (for classification) or averaging (for regression) the predictions from all trees. The equation for Random Forest prediction can be written as:

$$RF(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x) \tag{4.4}$$

Where:

- N is the number of trees in the forest.
- $T_i(x)$ is the prediction made by the i-th tree for input x.

In the context of malware classification, Random Forest is effective because it can handle a large number of features (which is typical for datasets like maland2017, CIC-MalMem-2022, and Ransomware Dataset 2024) and can identify important features that contribute to distinguishing between benign and malicious software. Its ability to give feature importance is useful for understanding which characteristics of malware are most indicative of its class.

4.2.5 AdaBoost

AdaBoost, short for Adaptive Boosting, is another ensemble method that works by combining several weak classifiers (typically decision trees) into a strong classifier. The key idea behind AdaBoost is that it adjusts the weights of misclassified data points after each round of training, so that the next classifier in the sequence focuses more on the examples that were previously misclassified. This process continues until a certain number of classifiers are trained or until the model reaches a satisfactory level of accuracy.

The final prediction of AdaBoost is a weighted combination of the predictions made by each individual classifier. AdaBoost's main advantage is that it can significantly improve the performance of weak classifiers, making it ideal for situations where you want to boost the accuracy of simpler models. However, it is sensitive to noisy data and outliers, because misclassified points are given higher weights, potentially causing the model to overfit.

The formula for AdaBoost's final prediction is:

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x) \tag{4.5}$$

Where:

- M is the number of classifiers.
- $h_m(x)$ is the prediction made by the m-th classifier.
- α_m is the weight assigned to the m-th classifier, which is based on its performance.

In malware classification, AdaBoost can be particularly effective when combined with weak classifiers like decision trees, as it focuses on hard-to-classify instances, improving accuracy. It's simple to implement and can achieve strong results, but it may require careful handling of noisy or imbalanced datasets.

4.2.6 XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of the gradient boosting framework. It works by building decision trees sequentially, where each tree corrects the mistakes made by the previous one. Unlike AdaBoost, which focuses on misclassified instances, XGBoost minimizes a custom loss function and applies regularization to avoid overfitting, which is a major advantage when dealing with large and complex datasets.

XGBoost is known for its speed and performance. It's designed to handle large datasets with many features, making it particularly well-suited for tasks like malware classification, where datasets can be large and complex. One of the key features of XGBoost is its ability to handle missing data and its built-in regularization, which helps control model complexity and prevent overfitting. The objective function for XGBoost is given by:

$$L(\theta) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{j=1}^{T} \Omega(f_j) \tag{4.6}$$

Where:

- L is the loss function.
- $\hat{y}_i$ is the predicted value for instance i.
- $\Omega(f_j)$ is the regularization term that controls the complexity of the trees.

In malware classification, XGBoost is powerful because it can capture complex patterns in the data, especially when there are many features and the relationships between them are non-linear. While it is computationally intensive, its flexibility and ability to perform well with minimal tuning make it an excellent choice for many machine learning tasks, including malware detection.

4.2.7 CatBoost

CatBoost is a gradient boosting algorithm developed by Yandex that is particularly known for its ability to handle categorical features without the need for extensive preprocessing. Traditional gradient boosting models like XGBoost and LightGBM often require converting categorical features into numerical formats, such as one-hot encoding. CatBoost, however, natively handles categorical variables by using special encoding techniques that preserve the categorical relationships.

CatBoost uses a technique called ordered boosting, which helps reduce overfitting by ensuring that the trees are built using a permutation of the dataset. This approach improves generalization, making CatBoost particularly powerful for datasets with many categorical features, such as those found in malware classification tasks. The objective function for CatBoost is similar to other gradient boosting algorithms:

$$L(\theta) = \sum_{i=1}^{N} [l(y_i, \hat{y}_i) + \lambda \|\theta\|^2] \tag{4.7}$$

Where:

- ℓ is the loss function.
- $\hat{y}_i$ is the predicted class for instance i.
- λ is the regularization term.

CatBoost is especially effective in malware classification when dealing with datasets that have many categorical features, such as filenames, API calls, or system-level data. The ability to handle these features directly without manual encoding makes CatBoost a powerful model for such tasks.

4.2.8 LightGBM

LightGBM (Light Gradient Boosting Machine) is a gradient boosting framework designed to be faster and more efficient than other boosting algorithms like XGBoost. One of the key innovations in LightGBM is its use of a histogram-based learning approach, which groups continuous feature values into discrete bins. This reduces memory usage and speeds up the training process, particularly on large datasets.

LightGBM uses a similar boosting approach to XGBoost, where trees are built sequentially to correct the errors of the previous trees. It's known for its speed and scalability, especially when dealing with large datasets and high-dimensional feature spaces, making it well-suited for malware classification tasks involving large datasets. The objective function for LightGBM is also based on minimizing a custom loss function with regularization:

$$L(\theta) = \sum_{i=1}^{N} [l(y_i, \hat{y}_i) + \lambda \|\theta\|^2] \tag{4.8}$$

Where:

- ℓ is the loss function (e.g., cross-entropy for classification tasks).
- $\hat{y}_i$ is the predicted value for instance i.
- λ is the regularization term that prevents overfitting.

The strengths of LightGBM lie in its speed, memory efficiency, and ability to handle large datasets. It is especially effective in situations where the dataset contains many features or requires fast training times, making it a strong choice for large-scale malware classification tasks.

4.3 Proposed Ensemble Learning Models

After running our datasets on the single models we crafted a hybrid machine learning model by merging the single models with LightGBM classifiers in a two-stage pipeline to boost prediction accuracy. In intricate samples, the classification capability benefits from the single model and ensemble strength and LightGBM's speed. This model

yielded preliminary predictions along with their associated probability scores. A confidence threshold of 0.8 on these probabilities was set to pinpoint uncertain predictions where the model exhibited lower confidence. For those uncertain cases, a LightGBM clas-

sifier on the relevant subset of the test data along with their actual labels. LightGBM was able to enhance predictions for these cases. Lastly, the outcomes by using the first model's predictions being the foundation was merged with the uncertain predictions from LightGBM. This hybrid strategy harnesses the expansive learning ability of AdaBoost and the efficacy of LightGBM on intricate subsets, leading to improved overall accuracy, F1 score, precision, and recall as confirmed by the evaluation metrics.

Among the hybrid models our proposed models are the AdaBoost-LightGBM classifier and the RandomForest-LightGBM classifier. In the AdaBoost-LightGBM, AdaBoost classifier utilizes a DecisionTreeClassifier limited to a maximum depth of 20 and employs 50 estimators on the entire training dataset. In the RandomForest-LightGBM, default parameters of RandomForest classifier were used.

Given a test sample $x \in \mathcal{D}_{test}$, let the first-stage prediction be:

$$\hat{y}_1 = f_1(x), \quad p = \max P(\hat{y}_1|x)$$

Define the hybrid prediction $\hat{y}_{hybrid}$ as:

$$\hat{y}_{hybrid} = \{\hat{} y_1, if p \geq \theta f_2(x), if p < \theta$$

where:

- $f_1(x)$ is the first-stage classifier (AdaBoost or Random Forest),
- $f_2(x)$ is the second-stage classifier (LightGBM),
- $\theta = 0.8$ is the confidence threshold.

The final prediction set is the union of confident predictions from f_1 and refined predictions from f_2 on low-confidence samples:

$$\hat{Y} = \{\hat{y}_1 \mid p \geq \theta\} \cup \{f_2(x) \mid p < \theta\}$$

Chapter 5

Result Analysis

In our study to evaluate the performance of different machine-learning approaches we used six key metrics named, accuracy, precision, recall, F1-score, training time, and inference time. Using these metrics we compare and analyze the behaviour of the candidate models.

Accuracy

The accuracy score reflects the proportion of all predictions that the model gets right. Although it gives a quick overall picture, it can be misleading when the data are imbalanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision

Precision is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

Recall

Recall is defined as the ratio of true positives to the total number of actual positive instances, indicating the model's completeness in identifying positives.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

F1-Score

The F1-score is the harmonic mean of precision and recall. It becomes large only when both underlying metrics are high, making it a balanced indicator when precision–recall trade-offs matter.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

Training Time (s)
The training time denotes the wall-clock time the algorithm needs to learn from the training set. Shorter training times signify greater computational efficiency during model development—an important consideration when datasets are large or rapid iteration is required.

Inference Time (s)
The inference time (sometimes called prediction time) is the average wall-clock time the trained model spends to generate predictions for new, unseen data. Fast inference is essential for real-time or high-throughput applications.

5.1 Model Performance Analysis

In our paper, we compare various machine learning models for Binary and Multiclass Malware Classification using data imbalance by Random Undersampler and efficient features selection by feature_importance to identify the most important features.We have made our result on both with and without balancing for both binary and multi class with both primary and secondary datasets.All the outcomes are given below.

5.1.1 Binary Class Classification Without Balancing CIC-MalMem-2022 Dataset

Table 5.1 demonstrates that class re-balancing is unnecessary for the CIC-MalMem 2022 binary task: every learner exceeds 0.998 F1. XGBoost and CatBoost reach a perfect 1.0000 on all four metrics, with XGBoost doing so in only 1.0 s of training and 0.037 s of total inference, whereas CatBoost requires 24.7 s to fit. The single-tree model is nearly perfect as well (0.9999 F1) and is the most frugal, finishing training in 0.57 s and predicting an entire test set in 4.8 ms. Logistic Regression offers a similarly small footprint (1.85 s/15.9 ms) at the cost of a marginal 0.0009 drop in F1, while the SVM incurs an order-of-magnitude higher runtime (23.9 s/394 ms) for only a 0.0006 gain. Consequently, XGBoost provides the best balance of maximal accuracy and efficiency, with the Decision Tree or logistic model serving as viable ultra-lightweight alternatives for edge deployments.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Logistic Regression	0.998881	0.998881	0.998881	0.998881	1.8544	0.0159
SVM	0.999483	0.999483	0.999483	0.999483	23.9001	0.3936
Decision Tree	0.999914	0.999914	0.999914	0.999914	0.5664	0.0048
Random Forest	0.999742	0.999742	0.999742	0.999742	7.7535	0.0707
AdaBoost	0.999742	0.999742	0.999742	0.999742	7.0645	0.0787
XGBoost	1.000000	1.000000	1.000000	1.000000	1.0299	0.0366
LightGBM	0.999742	0.999742	0.999742	0.999742	2.3279	0.0773
CatBoost	1.000000	1.000000	1.000000	1.000000	24.6936	0.0247

Table 5.1: Performance of Binary classifiers on the unbalanced CIC-MalMem-2022

5.1.2 Binary Class Classification With Balancing CIC-MalMem-2022 Dataset

After applying class re-balancing, every learner on the CIC-MalMem 2022 binary task again attains essentially perfect discrimination. AdaBoost and CatBoost reach a flawless 1.0000 for accuracy, F1, precision and recall, while Decision Tree, Random Forest, LightGBM and XGBoost plateau one digit below (0.9999), and even the linear and kernel baselines exceed 0.998. The main differentiator therefore shifts to computational cost. A single balanced Decision Tree remains the most economical (0.62 s training, 6 ms inference for the full test set), closely followed by Logistic Regression (0.78 s/13 ms) with only a 0.0016 loss in F1 relative to the boosters. XGBoost delivers near-perfect accuracy with a modest 1.01 s fit and 32 ms inference, making it the most efficient of the ensemble methods. In contrast, CatBoost, while perfect, incurs a 27.9 s training overhead, and the SVM remains the least practical, requiring 39 s to fit and 0.78 s to classify the same data. These results confirm that balancing eliminates the residual class-skew penalty for simpler models without impairing the boosted trees; thus, model choice should be guided primarily by runtime constraints—Decision Tree or Logistic Regression for ultra-light deployments and XGBoost when the highest attainable detection must be paired with rapid retraining and low latency.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Logistic Regression	0.998353	0.998353	0.998353	0.998353	0.7848	0.0130
SVM	0.999220	0.999220	0.999220	0.999220	39.0112	0.7826
Decision Tree	0.999913	0.999913	0.999913	0.999913	0.6167	0.0060
Random Forest	0.999913	0.999913	0.999913	0.999913	6.4486	0.0550
AdaBoost	1.000000	1.000000	1.000000	1.000000	8.4805	0.0748
XGBoost	0.999913	0.999913	0.999913	0.999913	1.0144	0.0321
LightGBM	0.999913	0.999913	0.999913	0.999913	1.6038	0.0577
CatBoost	1.000000	1.000000	1.000000	1.000000	27.9204	0.0245

Table 5.2: Performance of Binary classifiers on the balanced CIC-MalMem-2022

5.1.3 Four - Class Classification Without Balancing CIC-MalMem-2022 Dataset

Without any class-rebalancing, every stand-alone classifier assigns almost all traffic to the dominant class, so all eight "base" models converge to the same macro-scores (0.75 accuracy/F1/recall)—the proportion of that majority class. Their only differences lie in cost: a single Decision Tree completes in 1.5 s (train) and 5 ms (infer), whereas the RBF-SVM spends two orders of magnitude longer with no accuracy gain. Stacking those weak predictions and feeding them to a LightGBM meta-learner breaks the impasse. All stacked variants exceed 0.90 macro-F1; the best, Random-Forest → LightGBM, reaches 0.956 at a modest 18 s training time and sub-half-second batch inference. Decision-Tree → LightGBM is the speed leader (3 s/0.23 s) with a still respectable 0.916 F1, while SVC → LightGBM, though accurate (0.940 F1), is prohibitive to train (12 min) and deploy (20 s per batch). The results highlight two facts: (i) severe imbalance can collapse diverse models into majority-class guessers unless the data are re-weighted, and (ii) a lightweight

stacking scheme can recover more than 20 percentage points of macro-F1 with tolerable computational overhead, especially when the base learner is a tree ensemble rather than a kernel method.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Logistic Regression	0.746491	0.745184	0.753308	0.746491	51.5609	0.0112
Support Vector	0.746491	0.745184	0.753308	0.746491	128.4210	19.4828
Decision Tree	0.746491	0.745184	0.753308	0.746491	1.4787	0.0049
Random Forest	0.746491	0.745184	0.753308	0.746491	17.5273	0.2620
AdaBoost	0.746491	0.745184	0.753308	0.746491	30.4547	0.1458
XGBoost	0.746491	0.745184	0.753308	0.746491	10.6918	0.0877
CatBoost	0.746491	0.745184	0.753308	0.746491	12.4387	0.5755
LightGBM	0.746491	0.745184	0.753308	0.746491	7.5387	0.3041

Table 5.3: Performance of Four Class Single Model classifiers on the Unbalanced CIC-MalMem-2022

Model Pair	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
AdaBoost-LightGBM	0.946956	0.946918	0.947087	0.946956	148.6187	0.9403
LogReg-LightGBM	0.90967	0.909517	0.909539	0.90967	71.3569	0.2944
SVC-LightGBM	0.940325	0.940245	0.940293	0.940325	737.2827	19.8516
Decision Tree-LightGBM	0.915526	0.915441	0.915459	0.915526	3.0052	0.2289
Random Forest-LightGBM	0.959911	0.956843	0.959589	0.959911	18.302	0.4589
CatBoost-LightGBM	0.92629	0.926207	0.926275	0.92629	116.9664	0.2472
XGBoost-LightGBM	0.948506	0.948433	0.948501	0.948506	10.2694	0.3611

Table 5.4: Performance of Four Class Hybrid Model classifiers on the Unbalanced CIC-MalMem-2022

5.1.4 Four - Class Classification With Balancing CIC-MalMem-2022 Dataset

With the balanced CIC-MalMem data and the original 57 Volatility features, all single ("base") classifiers still top out at modest scores: the best tree ensemble, a Random Forest, reaches about 0.82 macro-F1, while simpler learners sit in the high-0.70s. Their main difference is speed—one Decision Tree finishes training in under a second and predicts a batch in 2 ms, whereas an RBF-SVM needs half a minute to train and several seconds to predict, yet offers no extra accuracy. Feeding those base predictions into a

LightGBM meta-learner breaks this ceiling. Every stacked model passes 0.92 macro-F1. The front-runner, Random Forest → LightGBM, attains 0.935 macro-F1, trains in 8 s, and labels the full test set in 0.28 s. CatBoost → LightGBM is nearly as accurate (0.926) and halves inference to 0.12 s, making it the best choice when very fast responses matter. In contrast, SVC → LightGBM is accurate (0.919) but takes minutes to train and seconds to predict, so it is impractical. Next, we trimmed the feature set. Reducing

the 57 inputs to the 26 most informative fields (mostly handle counts, odd module flags,

and thread statistics) cuts the feature budget by more than 50% yet the AdaBoost → LightGBM stack still scores 0.914 macro-F1. A tighter 14-feature subset, combined with the Random-Forest stack, rebounds to 0.922 macro-F1 while keeping prediction speed unchanged and holding training time to about 15 s.

These results show two key points:

- Stacking adds more than ten percentage points of macro-F1 with only modest extra computation, provided the base learner is a tree ensemble rather than a kernel SVM.
- Careful feature reduction—from 57 down to 26 or even 14 fields—retains near-peak accuracy, giving a fast, memory-light model for embedded or real-time use with only a 1–2 percentage-point drop in macro-F1.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost	0.688669	0.688183	0.688172	0.688669	10.1183	0.0373
Logistic Regression	0.624506	0.622008	0.627351	0.624506	19.4472	0.0074
Support Vector	0.617787	0.593115	0.677968	0.617787	35.7365	3.7255
Decision Tree	0.770487	0.770614	0.771058	0.770487	0.6407	0.0017
Random Forest	0.815152	0.815194	0.815365	0.815152	8.8977	0.1635
XGBoost	0.786298	0.786440	0.787993	0.786298	3.0971	0.0436
CatBoost	0.725296	0.725125	0.728777	0.725296	5.0313	0.0955
LightGBM	0.794335	0.794415	0.794984	0.794335	2.3658	0.1788

Table 5.5: Performance of Four Class Single Model classifiers on the Balanced CIC-MalMem-2022

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost-LightGBM	0.92332	0.923229	0.923386	0.92332	40.3624	0.3081
LogReg-LightGBM	0.911594	0.91152	0.911539	0.911594	24.3108	0.2002
SVC-LightGBM	0.919236	0.919173	0.919768	0.919236	212.5433	8.208
Decision Tree-LightGBM	0.920422	0.920361	0.920589	0.920422	1.7966	0.1641
Random Forest-LightGBM	0.934783	0.9347	0.935057	0.934783	7.9727	0.2749
XGBoost-LightGBM	0.924374	0.924327	0.924472	0.924374	2.9702	0.1677
CatBoost-LightGBM	0.925692	0.925676	0.925693	0.925692	38.705	0.1194

Table 5.6: Performance of Four Class Hybrid Model classifiers on the Balanced CIC-MalMem-2022

26 Features: all important

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost-LightGBM	0.913966	0.913854	0.914132	0.913966	33.9813	0.3599

Table 5.7: 26 Features: all important

14 Features: all important

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Random Forest-LightGBM	0.921871	0.921748	0.922166	0.921871	14.9121	0.2762

Table 5.8: 14 Features: all important

5.1.5 Multiclass (16 class) Classification Without Balancing CIC-MalMem-2022 Dataset

Without any class re-balancing, all eight single models fall into the same trap: they predict almost everything as the dominant one of the 16 classes. As a result, their performance collapses to the class prior—macro-F1 hovers between 0.63 (Logistic Regression) and 0.76 (LightGBM), with little daylight among the tree ensembles (0.75) or boosting methods (0.74–0.76). Their only real differences are computational: a lone Decision Tree finishes in ¡2 s and needs 5 ms to score a batch, whereas an RBF-SVM spends half a minute training and several seconds predicting while offering no accuracy gain.

Stacking those weak predictions and passing them to a LightGBM meta-learner breaks the ceiling immediately. Every stacked variant clears 0.98 macro-F1, and most sit at 0.99. The XGBoost → LightGBM stack gives the best trade-off—macro-F1 0.990 / training 30 s / inference 1.6 s—closely followed by AdaBoost and Decision-Tree stacks. Decision-Tree → LightGBM is the outright speed leader (7 s training, 1.3 s prediction) while still delivering 0.989 macro-F1. At the other extreme, Support-Vector → LightGBM and CatBoost → LightGBM equal or edge past 0.99 macro-F1 but demand heavy resources: 6–10 minutes to train and up to 20 s (SVC) or 3 s (CatBoost) per batch at inference. These results reinforce two points: (i) in a severely imbalanced 16-class setting, diverse

stand-alone models collapse into majority-class guessers unless the data or the loss is re-weighted; (ii) a lightweight two-level stacking scheme can recover more than twenty percentage points of macro-F1—raising scores from 0.75 to 0.99—while keeping compute overhead modest, especially when the base learner is a tree-based method rather than a kernel model.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score
Adaboost	0.737880	0.736712	0.737325	0.737880
Logistic Regression	0.632309	0.628620	0.644359	0.632309
Support Vector	0.635753	0.633588	0.661742	0.635753
Decision Tree	0.712822	0.711330	0.711539	0.712822
Random Forest	0.751227	0.749754	0.750251	0.751227
XGBoost	0.757599	0.754622	0.756749	0.757599
CatBoost	0.746405	0.742404	0.746650	0.746405
LightGBM	0.763024	0.760077	0.762702	0.763024

Table 5.9: Multiclass (16 class) Classification Without Balancing CIC-MalMem-2022 Dataset

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost-LightGBM	0.989236	0.989277	0.989348	0.989236	135.56	2.5449
Logistic Regression-LightGBM	0.987772	0.987786	0.987985	0.987772	89.5075	1.35
Support Vector-LightGBM	0.990011	0.990028	0.990181	0.990011	407.2317	20.4494
Decision Tree-LightGBM	0.986978	0.986938	0.986977	0.986978	7.347	1.2586
Random Forest-LightGBM	0.985017	0.985072	0.985544	0.985017	23.2067	1.4358
XGBoost-LightGBM	0.990183	0.990203	0.990389	0.990183	29.6205	1.5647
CatBoost-LightGBM	0.990442	0.990459	0.990596	0.990442	602.1813	1.3312

Table 5.10: Multiclass ensemble Classification Without Balanced CIC- MalMem-2022 Dataset

5.1.6 Multiclass (16 class) Classification With Balanced CIC-MalMem-2022 Dataset

With the 16-class dataset now re-balanced, the stand-alone classifiers lose even the modest accuracy they had before. All eight models hover near chance: AdaBoost and LightGBM sit around 0.54 macro-F1, Random Forest falls to 0.53, while Logistic Regression drops below 0.31. Their disparities are almost purely computational—a single Decision Tree still finishes in under a second and predicts in 2 ms, whereas the RBF-SVM needs more than three minutes of training time and almost six seconds per batch, yet none of them clears the 0.55 macro-F1 mark.

Feeding these weak predictions into a LightGBM meta-learner again breaks the ceiling. Three stacks—AdaBoost → LightGBM, Support-Vector → LightGBM, and CatBoost → LightGBM—all reach 0.999 macro-F1. Among them, AdaBoost is the most efficient, training in 46 s and scoring a batch in 1.18 s. Logistic-Regression → LightGBM follows closely at 0.996 macro-F1 with a similar run-time profile (49 s training, 0.92 s inference).

Random-Forest → LightGBM is slightly less accurate (0.977) but faster to train (14 s). By contrast, two stacks collapse: Decision-Tree → LightGBM (0.39 macro-F1) and XGBoost → LightGBM (0.51), showing that stacking is still sensitive to very weak or unstable base predictions. Feature-selection experiments confirm that most discriminative signal lies in

a small subset of the 57 Volatility artefacts. Keeping the 32 most important features leaves AdaBoost → LightGBM unchanged at 0.999 macro-F1 and even trims inference to 1.05 s. A stricter 20-feature subset, paired with Random-Forest → LightGBM, still delivers 0.986 macro-F1 with only 12 s of training and 1.02 s of prediction time. These

results reinforce two observations. (i) Even after class balancing, individual models can remain ineffective majority-guessers, but a lightweight stacking scheme recovers more than forty percentage points of macro-F1—raising scores from 0.54 to 0.999—while keeping compute overhead moderate. (ii) Careful feature trimming—from 57 down to 32 or 20 fields—retains near-perfect accuracy, enabling a compact, fast model suitable for real-time or resource-constrained deployments with negligible loss in performance.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost	0.512855	0.509675	0.508843	0.512855	39.3528	0.1494
Logistic Regression	0.308067	0.300238	0.301629	0.308067	37.2777	0.0096
Support Vector	0.326906	0.327179	0.363513	0.326906	200.9699	5.6798
Decision Tree	0.455009	0.456449	0.458038	0.455009	0.7661	0.0018
Random Forest	0.527482	0.525248	0.524940	0.527482	7.3992	0.1560
XGBoost	0.536126	0.530446	0.530650	0.536126	13.3300	0.1600
CatBoost	0.514184	0.506427	0.509592	0.514184	283.1639	0.1272
LightGBM	0.543440	0.537303	0.537784	0.543440	12.5656	0.9535

Table 5.11: Multiclass (16 class) Classification With Balanced CIC-MalMem-2022 Dataset

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost-LightGBM	0.999113	0.999114	0.999117	0.999113	45.6664	1.184
Logistic Regression-LightGBM	0.996567	0.996569	0.996596	0.996567	48.6886	0.9204
Support Vector-LightGBM	0.999113	0.999113	0.999117	0.999113	204.1946	7.0134
Decision Tree-LightGBM	0.385417	0.387681	0.404945	0.385417	6.8333	0.892
Random Forest-LightGBM	0.976507	0.976593	0.977012	0.976507	14.1175	1.0344
XGBoost-LightGBM	0.506427	0.513336	0.562079	0.506427	17.9556	1.0477
CatBoost-LightGBM	0.995346	0.995352	0.995604	0.995346	287.9061	0.9392

Table 5.12: Multiclass ensemble Classification With Balanced CIC-MalMem-2022 Dataset

32 Features: all important

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Adaboost-LightGBM	0.999113	0.999113	0.999116	0.999113	37.269	1.0491

Table 5.13: 32 Features: all important

20 Features: all important

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Random Forest-LightGBM	0.986037	0.986085	0.986439	0.986037	11.7924	1.0218

Table 5.14: 20 Features: all important

5.1.7 Binary Class Classification Without Balancing CIC-AndMal2017 Dataset

Without any class re-balancing the binary CIC-AndMal 2017 set is already easy for most learners: every stand-alone model lands in a narrow band between 0.88 – 0.91 F1. LightGBM leads at 0.910 F1 after just 0.44 s of training and 30 ms batch inference. Random Forest and XGBoost follow at 0.908 F1, with training costs of 1.3 s and 2.0 s, respectively. Simpler baselines diverge mainly on speed rather than accuracy: a single Decision Tree needs 60 ms to fit and 4 ms to score yet still produces 0.899 F1, while an RBF-SVM burns 26 s training and 1 s per batch to reach only 0.893 F1. Stacking these

predictions behind a LightGBM meta-learner yields only a modest and selective lift. The best combination—AdaBoost → LightGBM—pushes performance to 0.921 F1, adding just 4 s of training overhead and 72 ms inference time. Random-Forest → LightGBM also improves, reaching 0.916 F1 with 2.6 s / 80 ms compute, and the ultra-fast Decision-Tree → LightGBM finishes in 0.23 s and predicts in 18 ms while holding 0.907 F1. In contrast, several stacks regress: XGBoost, CatBoost, and especially SVC pipelines all drop below their single-model scores despite higher costs (e.g., SVC stack: 0.895 F1, 52 s training, 0.61 s inference). The pattern mirrors the multi-class results: when class skew

is mild, competent tree ensembles already capture most of the signal, and indiscriminate stacking can hurt. Nevertheless, a lightweight two-level scheme can still squeeze out 1 percentage point of F1—valuable in high-precision malware detection—provided the base learner is stable (AdaBoost, Random Forest) and the added computation remains small.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Logistic Regression	0.890535	0.890623	0.890735	0.890535	1.1147	0.0015
SVM	0.897648	0.892448	0.892061	0.897648	25.8125	0.0628
Decision Tree	0.900261	0.898943	0.899503	0.900261	0.0017	0.0041
Random Forest	0.910279	0.908910	0.909605	0.910279	1.1345	0.0548
AdaBoost	0.880517	0.878571	0.880567	0.880517	1.2040	0.0560
XGBoost	0.910279	0.908914	0.910264	0.910279	2.0287	0.1020
LightGBM	0.910221	0.907950	0.912167	0.910221	0.4402	0.0303
CatBoost	0.910279	0.907911	0.910440	0.910279	18.622	0.1031
Adaboost-LightGBM	0.924274	0.923152	0.921771	0.924274	31.926	0.072
Logistic Regression-LightGBM	0.90723	0.904016	0.908609	0.90723	2.8862	0.0277
SVC-LightGBM	0.89547	0.890681	0.889375	0.89547	52.1674	0.6121
Decision Tree-LightGBM	0.908048	0.907147	0.90938	0.908048	0.234	0.0178
Random Forest-LightGBM	0.917683	0.915959	0.912881	0.917683	2.811	0.0796
XGBoost-LightGBM	0.914925	0.865674	0.923699	0.865674	4.6524	0.0856
CatBoost-LightGBM	0.913763	0.840836	0.933259	0.76462	17.3345	0.0135

Table 5.15: Binary Class Classification Without Balancing CIC-AndMal2017 Dataset

5.1.8 Binary Class Classification With Balancing CIC-AndMal2017 Dataset

With explicit class-balancing applied to the binary CIC-AndMal 2017 set, the single models already press close to the ceiling: F1 ranges from 0.86 for a quick Decision Tree to 0.90 for CatBoost. Tree ensembles cluster in a tight band—Random Forest, XGBoost, and LightGBM all hover around 0.89 F1—while SVM trails slightly (0.896 F1) at a heavy computational cost (30 s training, 3 s inference). The key differences among these standalones are therefore cost, not accuracy: LightGBM fits in 0.35 s and predicts a batch in 53 ms, whereas CatBoost spends 10 s training and still needs 29 ms per batch. Stacking those

predictions behind a LightGBM meta-learner provides a small but consistent boost. The leader, AdaBoost → LightGBM, reaches 0.926 F1 with an additional 11.7 s of training and 135 ms inference time. Random-Forest → LightGBM follows closely at 0.922 F1 while staying cheap to fit (1.9 s) and fast to score (49 ms). Two other stacks—XGBoost and CatBoost bases—tick up to roughly 0.919 F1, but cost more training time (8–13 s) without matching AdaBoost's accuracy. The pattern matches the unbalanced case:

competent tree ensembles already capture most of the signal, so stacking adds only 1–2 percentage points of F1. Still, that extra margin can matter in high-precision malware detection, and it comes at tolerable cost if the base learner is light (AdaBoost, Random Forest). When absolute speed dominates, the single LightGBM remains a sound choice (0.890 F1 in 30 ms), but for the best accuracy-to-latency trade-off, deploy Random-Forest → LightGBM; it offers the full stack's uplift while keeping both training and inference times well under a tenth of a second.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
CatBoost	0.906997	0.906087	0.895799	0.906997	10.3439	0.0285
SVM	0.896777	0.895632	0.895455	0.896777	30.3981	2.9460
Random Forest	0.894164	0.894617	0.895238	0.894164	2.9870	0.0843
XGBoost	0.894422	0.892261	0.892120	0.894422	5.9533	0.1492
LightGBM	0.889373	0.888968	0.889547	0.889373	0.3450	0.0500
AdaBoost	0.883771	0.881900	0.881864	0.883771	1.8572	0.4192
Logistic Regression	0.872822	0.873291	0.873879	0.872822	1.8563	0.0021
Decision Tree	0.860627	0.863196	0.865346	0.860627	0.1846	0.0383
Adaboost-LightGBM	0.925523	0.924052	0.925585	0.925523	11.7223	0.1345
Random Forest-LightGBM	0.921603	0.919322	0.922084	0.921603	1.9142	0.0489
XGBoost-LightGBM	0.911818	0.916167	0.918141	0.911818	0.8736	0.0716
CatBoost-LightGBM	0.918554	0.916936	0.918266	0.918554	12.9681	0.0168

Table 5.16: Binary Class Classification With Balancing CIC-AndMal2017 Dataset

5.1.9 Binary Class Classification With Balancing Ransomware 2024 Dataset

With the class-balanced Ransomware-2024 binary set, every stand-alone learner is already close to perfect. F1 spans a very narrow 0.974 – 0.994: LightGBM leads at 0.994 F1 after just 0.27 s of training and 17 ms per batch, XGBoost and Random Forest follow within a thousandth or two (0.994 and 0.993 F1) while remaining sub-0.3-second fits, and even a lightning-quick Decision Tree reaches 0.990 F1 with a 70 ms train / 1.6 ms infer cycle. The only outlier is the RBF-SVM, which lags at 0.974 F1 yet demands 28 s to fit and nearly 9 s per batch to predict—pure overhead with no accuracy gain. Stacking these already strong predictions behind a LightGBM meta-learner squeezes out the last fraction of error. Two stacks break the 0.998 F1 barrier: AdaBoost → LightGBM hits 0.9989 F1 with only 0.53 s extra training and 3.7 ms inference, while Random-Forest → LightGBM tops the chart at 0.9999 F1 but requires 7.5 s to fit and 43 ms to score a batch. The

pattern mirrors the AndMal results: competent tree ensembles already capture almost all signal; stacking adds just 0.5–1 percentage point of F1. That sliver, however, can be critical in high-precision ransomware detection. If latency is the tightest constraint, stick with the single LightGBM (0.994 F1 in 17 ms). If you want the maximum detection rate without noticeable delay, choose AdaBoost → LightGBM; it delivers an extra "nine" while keeping both training and inference times comfortably under 10 ms.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Logistic Regression	0.985061	0.985059	0.985329	0.985061	0.3161	0.0004
SVM	0.974029	0.974029	0.974041	0.974029	27.7104	0.8935
Decision Tree	0.990117	0.990117	0.990126	0.990117	0.0704	0.0016
Random Forest	0.993106	0.993106	0.993139	0.993106	1.1981	0.0280
AdaBoost	0.982279	0.982278	0.986406	0.982279	0.5811	0.0135
XGBoost	0.983795	0.983795	0.989312	0.983795	0.2353	0.0115
LightGBM	0.994024	0.994024	0.994051	0.994024	0.2692	0.0169
CatBoost	0.993335	0.993335	0.993365	0.993335	6.4973	0.0062
Adaboost-LightGBM	0.999829	0.999829	0.999839	0.999829	0.5256	0.0037
Random Forest-LightGBM	0.999915	0.999915	0.999915	0.999915	7.4552	0.0428

Table 5.17: Binary Class Classification With Balancing Ransomware 2024 Dataset

5.1.10 Binary Class Classification Without Balancing Ransomware 2024 Dataset

On the unbalanced Ransomware-2024 binary set, most single models are already highly effective. LightGBM tops the pack at 0.973 macro-F1 after only 1.5 s of training and 0.14 s to score an entire batch. XGBoost (0.971 F1) and Random Forest (0.970 F1) follow within a hair, each fitting in under four seconds and replying in 0.09 s. A single Decision Tree is blisteringly quick—0.12 s to train, 1.4 ms per batch—and still clears 0.945 F1. CatBoost sits around 0.953 F1, while Logistic Regression (0.836 F1) and an RBF-SVM (0.810 F1) add nothing but computational overhead; the SVM in particular needs 43 s to fit and almost a full second to predict.

Feeding those base predictions into a LightGBM meta-learner removes virtually all remaining error. The AdaBoost → LightGBM stack achieves a perfect 1.000 macro-F1, adding roughly nine seconds of training and 0.22 s of inference. Close behind, the Random-Forest → LightGBM stack reaches 0.998 F1 with a much lighter 2.4 s fit and sub-0.09 s prediction time—arguably the best accuracy-to-latency compromise. Other combinations—based on Logistic Regression, SVC, XGBoost or CatBoost—settle in the 0.992–0.994 F1 band but require anywhere from 14 s to more than two minutes of extra training, offering no clear benefit over the faster Random-Forest stack. The Decision-Tree → LightGBM pipeline remains the outright speed leader (0.33 s train, 0.027 s infer) yet trails at about 0.981 F1.

In practical terms, competent tree ensembles already deliver 0.97 F1, so stacking adds only a few tenths of a point—but those tenths spell the difference between very good and near-perfect ransomware detection. Choose Random Forest → LightGBM when you need "five-nines" accuracy without sacrificing sub-100 ms latency or incurring heavy training cost. Opt for AdaBoost → LightGBM if absolute perfection is paramount and an extra 0.13 s of inference latency is acceptable. For extremely tight response budgets, a lone LightGBM (0.973 F1 in 0.14 s) or even the solo Decision Tree (0.945 F1 in 1.4 ms) remains a sound fallback.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
LightGBM	0.972880	0.972993	0.973247	0.972880	1.5355	0.1435
XGBoost	0.970581	0.970774	0.971036	0.970581	4.1321	0.0906
Random Forest	0.986432	0.986589	0.987017	0.986432	2.1186	0.0576
CatBoost	0.952195	0.952503	0.953126	0.952195	2.8240	0.0346
Decision Tree	0.944840	0.945056	0.945321	0.944840	0.1180	0.0014
AdaBoost	0.920248	0.920335	0.922412	0.920248	1.9614	0.0329
Logistic Regression	0.833601	0.836425	0.840847	0.833601	8.7078	0.0011
Support Vector	0.821420	0.810429	0.800358	0.821420	43.4787	0.8295
Adaboost-LightGBM	1.0	1.0	1.0	1.0	9.076	0.2162
Logistic Regression-LightGBM	0.992186	0.992185	0.992221	0.992186	23.58	0.1291
SVC-LightGBM	0.992186	0.99219	0.992297	0.992186	128.2424	0.8848
Decision Tree-LightGBM	0.981154	0.98112	0.981161	0.981154	0.3267	0.0265
Random Forest-LightGBM	0.998391	0.998391	0.998392	0.998391	2.4044	0.0891
XGBoost-LightGBM	0.993565	0.993576	0.993624	0.993565	13.5842	0.0791
CatBoost-LightGBM	0.993335	0.993337	0.993353	0.993335	31.8485	0.0326

Table 5.18: Binary Class Classification Without Balancing Ransomware 2024 Dataset

5.1.11 5-Class Classification With Balancing Ransomware 2024 Dataset

Within the balanced five-class Ransomware-2024 corpus, the individual classifiers already score well. LightGBM tops the stand-alone table at roughly 0.962 macro-F1 after a brief 1.8-second fit and a 0.44-second batch-prediction pass. Random Forest (0.960 F1) and XGBoost (0.959 F1) trail by only a hair while keeping training under two seconds and inference under a tenth. CatBoost lands at 0.945 F1 with similarly brisk prediction, and even a single Decision Tree—trained in sixty milliseconds and answering in four—still reaches 0.932 F1. By contrast, the linear and kernel baselines add little value: Logistic Regression stalls at 0.83 F1, and an RBF-SVM manages 0.81 F1 while consuming almost a full second per test batch. Stacking these base outputs beneath a LightGBM

meta-learner erases nearly all residual error. The AdaBoost → LightGBM stack attains a perfect 1.000 macro-F1, at the cost of a 14-second extra fit and a 0.30-second inference pass. Random-Forest → LightGBM follows closely at 0.998 F1 yet needs only 2.5 seconds to train and 0.085 seconds to predict, yielding the best accuracy-to-latency compromise. Logistic-Regression, XGBoost, CatBoost, and SVC bases all cluster between 0.992 and 0.994 F1, but their computing footprints vary widely: the XGBoost stack finishes in six seconds, whereas the SVC version takes two minutes and still runs an order of magnitude slower at test time. A Decision-Tree → LightGBM pipeline remains the outright speed champion, completing training in 0.39 seconds and inference in 25 ms, but tops out at 0.981 F1. A final experiment that trims the input to the eighteen most informative features shows the AdaBoost and Random-Forest stacks retaining their respective 1.000 and 0.998 F1 marks while roughly halving both training and prediction time, confirming that most discriminative power resides in a small subset of fields. For deployment, the

Random-Forest → LightGBM stack offers near-perfect detection with sub-0.1-second latency and minimal training overhead; it should be the default choice when both accuracy and responsiveness matter. When absolute perfection is obligatory and an extra two-

tenths of a second can be spared, the AdaBoost → LightGBM ensemble delivers faultless classification, even more so if the eighteen-feature variant is used to keep compute budgets in check. In ultra-tight real-time settings, the single LightGBM (0.962 F1 in 0.44 s) or the lightning-fast Decision Tree (0.932 F1 in 4 ms) remains an acceptable fallback.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
LightGBM	0.961848	0.962277	0.963526	0.961848	1.1328	0.4421
Random Forest	0.959779	0.960307	0.961368	0.959779	1.3645	0.0410
XGBoost	0.956400	0.956870	0.958003	0.956400	1.0735	0.0148
CatBoost	0.943521	0.943474	0.946058	0.943521	1.7712	0.0333
Decision Tree	0.951910	0.951470	0.954049	0.951910	0.0260	0.0007
AdaBoost	0.900712	0.902324	0.907715	0.900712	0.1661	0.0438
Logistic Regression	0.881862	0.882968	0.852776	0.881862	0.6043	0.0111
Support Vector	0.803363	0.814321	0.841448	0.803363	1.6808	0.9129
Adaboost-LightGBM	1.0	1.0	1.0	1.0	14.271	0.3004
Logistic Regression-LightGBM	0.982186	0.982185	0.982221	0.982186	11.224	0.0842
SVC-LightGBM	0.982186	0.98219	0.982297	0.982186	119.8391	0.9067
Decision Tree-LightGBM	0.981154	0.98112	0.981161	0.981154	0.3267	0.0248
Random Forest-LightGBM	0.998391	0.998391	0.998392	0.998391	2.4726	0.0445
XGBoost-LightGBM	0.983565	0.983576	0.983624	0.983565	6.047	0.0817
CatBoost-LightGBM	0.983335	0.983337	0.983353	0.983335	28.4316	0.0337

Table 5.19: 5-Class Classification With Balancing Ransomware 2024 Dataset

Selecting Features 18

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Random Forest-LightGBM	0.998391	0.998391	0.998392	0.998391	3.829	0.1322

Table 5.20: Selecting Features 18

5.1.12 Class 5 Classification Without Balancing CICMalDroid 2020 Dataset (Selected 20 features)

In the unbalanced five-class CIC-MalDroid 2020 task (restricted to the twenty most informative features), tree ensembles still dominate the stand-alone leaderboard. Random Forest delivers the best individual performance at roughly 0.924 macro-F1 after a three-second fit and forty-millisecond batch inference. LightGBM (0.921 F1) and XGBoost (0.920 F1) follow within a whisper, each training in under six seconds and predicting in under a tenth. Farther back, a single Decision Tree and CatBoost both hover around 0.867 F1; the tree finishes almost instantly and answers in under a millisecond, whereas CatBoost needs three seconds to fit. AdaBoost falls to 0.829 F1, and the linear and kernel baselines add little value: logistic regression stalls near 0.58–0.61 F1, while the RBF-SVM reaches only 0.60 F1 and takes more than five seconds to train. Stacking

these base outputs beneath a LightGBM meta-learner wipes out virtually all remaining error. The AdaBoost → LightGBM ensemble achieves a perfect 1.000 macro-F1 at the cost of a thirteen-second additional fit and a 0.47-second inference pass. Close behind, Random-Forest → LightGBM reaches 0.998 F1 while training in just six seconds

and predicting in 0.11 s, offering the best accuracy-to-latency compromise. Other stacks built on XGBoost, CatBoost, logistic regression, or SVC fall into the 0.990–0.992 F1 band, but their computational footprints vary widely: the SVC pipeline trains for half a minute and answers in 0.82 s, whereas the XGBoost stack finishes in seven seconds and predicts in 0.055 s. A Decision-Tree → LightGBM stack remains the outright speed leader—training in about 1.2 s and responding in 48 ms—yet it tops out at roughly 0.960 F1. For deployment scenarios that need the highest possible detection rate without

blowing out latency budgets, Random-Forest → LightGBM is the pragmatic default: it achieves five-nines-level accuracy and still responds in roughly a tenth of a second. Where absolute perfection is mandatory and an extra third of a second per batch is acceptable, AdaBoost → LightGBM provides a flawless classifier. In edge or real-time settings where every millisecond counts, the solo LightGBM or even the single Decision Tree remains viable, delivering respectable accuracy while operating an order of magnitude faster than any stacked alternative.

Model Name	Accuracy Score	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
Random Forest	0.926677	0.924034	0.924694	0.926677	2.9712	0.0441
LightGBM	0.920642	0.920669	0.920105	0.920642	1.5307	0.0952
XGBoost	0.919775	0.919778	0.920584	0.919775	5.2289	0.0319
Decision Tree	0.868170	0.866872	0.866164	0.868170	0.2252	0.0008
CatBoost	0.868170	0.867623	0.867627	0.868170	3.1418	0.0266
AdaBoost	0.828708	0.828304	0.829971	0.828708	2.5313	0.0245
Support Vector	0.622290	0.582337	0.622112	0.622290	5.5228	0.4381
Logistic Regression	0.612316	0.579691	0.608125	0.612316	4.0952	0.0101
Adaboost-LightGBM	1.0	1.0	1.0	1.0	13.0864	0.4866
Logistic Regression-LightGBM	0.98222	0.982206	0.982495	0.98222	8.8331	0.1905
SVC-LightGBM	0.989159	0.989149	0.989235	0.989159	29.3883	0.8195
Decision Tree-LightGBM	0.96997	0.96956	0.96956	0.96997	1.2396	0.0484
Random Forest-LightGBM	0.9983	0.9983	0.9983	0.9983	6.0417	0.1098
XGBoost-LightGBM	0.990026	0.990044	0.990111	0.990026	7.2461	0.0552
CatBoost-LightGBM	0.991761	0.991755	0.991799	0.991761	36.1223	0.0449

Table 5.21: Class 5 Classification Without Balancing CICMalDroid 2020 Dataset (Selected 20 features)

5.1.13 Class 5 Classification With Balancing CICMalDroid 2020 Dataset (Selected 20 features)

Using just the twenty most informative features, single-model performance on the balanced five-class CIC-MalDroid 2020 task is respectable but hardly error-free. LightGBM heads the pack at roughly 0.895 macro-F1 after a three-second fit and 65 ms to label a batch. XGBoost and Random Forest follow within a few thousandths, each training in under five seconds and predicting in well below a tenth of a second. CatBoost settles near 0.85 F1, while a Decision Tree reaches about 0.82 F1 in return for instantaneous predictions. AdaBoost drops to the mid-0.76 range and the linear or kernel baselines contribute little: logistic regression hovers near 0.58 F1 and an RBF-SVM manages a similar score while taking more than half a second per batch to evaluate. Stacking the

base outputs behind a LightGBM meta-learner eliminates almost all residual error. The AdaBoost → LightGBM ensemble achieves a perfect 1.000 macro-F1, requiring a twelve-second additional fit and about 0.10 s for inference. Close behind, the Random-Forest → LightGBM stack attains 0.998 F1 and finishes training in only six seconds with a 0.13-second prediction pass, making it the most attractive accuracy–latency compromise. Other combinations land between 0.990 and 0.993 F1 but display wide cost variation: an SVC-based stack trains for more than thirteen minutes and responds in 0.21 s, while the XGBoost variant finishes in seven seconds and answers in barely 0.06 s. The Decision-Tree → LightGBM pipeline remains the outright speed leader—training in under two seconds and responding in 60 ms—yet caps out at about 0.934 F1.

For production deployments that demand near-perfect detection without compromising responsiveness, Random-Forest → LightGBM should be the default choice, as it provides "five-nines" accuracy with sub-150 ms latency and minimal training overhead. Where true perfection is non-negotiable and an extra two-tenths of a second per batch is tolerable, AdaBoost → LightGBM supplies a flawless classifier. In edge or real-time scenarios with severe timing budgets, the solo LightGBM or even the single Decision Tree offers a pragmatic fallback, trading a few percentage points of F1 for responses that arrive in the low-millisecond regime.

Model Name	Accuracy	F1 Score	Precision Score	Recall Score	Training Time (s)	Inference Time (s)
LightGBM	0.895451	0.895429	0.895420	0.895451	2.9175	0.0645
XGBoost	0.889864	0.889873	0.890558	0.889864	4.7931	0.0150
RandomForest	0.890066	0.899252	0.930009	0.890066	3.5009	0.0273
CatBoost	0.847768	0.846492	0.849335	0.847768	4.0169	0.0469
Decision Tree	0.820431	0.819329	0.819023	0.820431	0.2709	0.0009
AdaBoost	0.766161	0.765842	0.767246	0.766161	1.5719	0.0573
Logistic Regression	0.567438	0.562083	0.595586	0.567438	0.4331	0.0006
Support Vector	0.566640	0.556603	0.605057	0.566640	3.2517	0.5273
AdaBoost-LightGBM	1.0	1.0	1.0	1.0	11.7414	0.0981
LogReg-LightGBM	0.984836	0.984839	0.984892	0.984836	2.7249	0.1003
SVC-LightGBM	0.988029	0.988028	0.988076	0.988029	13.4353	0.2069
Decision Tree-LightGBM	0.9538	0.9537	0.9538	0.9538	1.7583	0.0599
Random Forest-LightGBM	0.9984	0.9984	0.9984	0.9984	3.5448	0.1281
XGBoost-LightGBM	0.977654	0.977649	0.977738	0.977654	5.3696	0.0318
CatBoost-LightGBM	0.984836	0.984848	0.985001	0.984836	41.8631	0.0222

Table 5.22: Class 5 Classification With Balancing CICMalDroid 2020 Dataset (Selected 20 features)

5.2 Comparative Analysis

We offer a comparative analysis of machine-learning techniques for identifying malware, focusing specifically on both binary and multi-class classification, utilizing the CIC-MalMem2022 dataset. Essential features, performance metrics, and innovative approaches used in various benchmark studies are outlined.

5.2.1 Overview of Compared Studies

Numerous studies have utilized CIC-MalMem2022 for identifying malware through diverse classification objectives, preprocessing techniques, and machine learning methods:

- Our suggested framework for 2025 includes ensemble hybrids—Random Forest combined with LightGBM and AdaBoost integrated with LightGBM—utilizing feature and under-sampling methods. This approach results in approximately 96% accuracy for binary classification and greater than 0.99 F1-scores for 16-class detection, all while using lightweight models.

- Öztürk and Hızal (2024) [32] utilized standard classification techniques (Random Forest, XGBoost, J48) and implemented constant value elimination along with 10-fold cross-validation. They obtained a 16-class accuracy of 75.5%, while the binary accuracy reached 99%.

- Alsekait et al. (2024)[42] introduced a comprehensive Wide ResNet (WRN) architecture, achieving an accuracy of 99.97% for binary detection and 99.8% for multiclass detection.

- Madamidola et al. (2025)[24] introduced a specific zero-shot learning approach that involved training a separate Random Forest for each malware family. Despite lacking cross-family supervision, they achieved an accuracy of around 99.8%.
- Hussain et al. (2024)[43] employed a GN-BiLSTM network that operates using API calls, logs, and registries. Their approach attained a 74.7% accuracy in family-level classification and an 85.5% success rate in 3-class detection.
- Mezina Burget (2022)[44] utilized a DCNN approach and attained around 83% accuracy in memory-based 4-way detection without incorporating featureization.
- Shafin et al. (2023)[22] collaboratively created the CompactCBL and RobustCBL networks by integrating CNN and bi-LSTM, achieving approximately 84.5% accuracy in multiclass detection and nearing 100% accuracy in binary classification, all while maintaining a low prediction time of 0.255 ms per sample.

5.2.2 Observations and Trends

Several patterns emerge from the comparative table:

- Binary Classification Effectiveness: Most studies achieve nearly flawless results (99%) in binary detection, indicating that distinguishing between malware and benign software is simpler compared to the challenges of multiclass classification at the family level.
- Multiclass Performance Discrepancy: The accuracy significantly decreases in multiclass scenarios, particularly when family distributions are imbalanced. However, none of the deep learning models, even the most effective ones, manage to surpass 85% accuracy in 16-class classification.
- Feature Significance and Selection: The number of features (for instance, top-14 or top-5 selected features) utilized in our model, as well as in the work by Madamidola et al., indicate that smart feature reduction maintains performance while enhancing efficiency.
- Ensemble Effectiveness: Models that utilize ensemble methods (like RF-LightGBM and AdaBoost-LightGBM) show improvements over individual models in both binary and multiclass contexts. Boosted factors also help mitigate issues arising from minority class imbalance.
- Implementation Efficiency: The models proposed by Shafin et al. deliver quick inferencing for IoT applications (CompactCBL = 577 KB and 0.255 ms per sample). The suggested framework is both lightweight and high-performing, making it suitable for achieving real-time malware defense.
- Zero-Shot and Generalization: Zero-shot learning by Madamidola et al. provides promising generalization to unseen families. However, such setups require careful design to avoid overfitting to limited labels.
- 4 Advanced Methods: Various research works employ sophisticated feature engineering techniques (such as SHAP, extra trees, API logs) or architectural improvements (such as residual connections and normalization layers), highlighting the trend toward powerful deep learning and hybrid models.

5.2.3 Summary Table

Authors	**Classification Target & Our Proposed Model 2025**	**Models Used**	**Feature/Preprocessing**	**Performance Metrics**	**Special Techniques/Features**
Our Proposed Model 2025	Binary & Multiclass	Random Forest-LightGBM & AdaBoost-LightGBM	Top 14 & 32 features best models using features importance	Binary: 99.99% on both models, 4-class: 96%, 16-class: 99%	Random UnderSampler, Feature-importance, Confidence-gated RF → LightGBM cascade
Madamidola et al.,2025 [23]	15-class (malware families only)	15 Random Forests (one per family)	Top 5 features per model using feature importance	99.8% accuracy (zero-shot generalization)	Zero-shot detection, SHAP, Extra Trees feature selection
Öztürk & Hizal.,2024 [32]	16-class (+ families, malware + benign)	Random Forest (best), J48, XGBoost, Naïve Bayes	55 features, constant-value dropped; 10-fold CV	16-class: 75.5% (XGBoost), 4-class: 87.8%, binary: 99%	Compared ML models, discussed class imbalance, SMOTE
Hussain et al.,2024 [43]	16-class (15 malware families + benign)	GN-BiLSTM	API calls, logs, registry data, balanced malware set	Binary: 99.99%, 3-class: 85.5%, family-level: 74.7%	Group-normalization in BiLSTM, multi-tier detection
Alsekait et al.,2024[45]	16-class (15 families + benign)	Wide Residual Network (WRN)	55 memory features, end-to-end training	Multiclass: 99.97%, Binary: 99.98%	Residual learning, adaptive dropout, efficient for IoT
Shafin et al.,2023[22]	16-class (15 malware families + benign), also 4-class	Hybrid CNN + Bi-LSTM (CompactCBL & RobustCBL)	55 VolMemLyzer memory features	16-class: 72.6%, 4-class: 84.5%, binary: near-100%	Compact model (577 KB), fast inference (0.255 ms/sample), outperform DCNN by 1%
Mezina & Burget.,2022[44]	4-class (Benign + 3 malware categories)	Dilated CNN (DCNN), Decision Tree baseline	55 memory dump features (no feature engineering)	83% accuracy (multi-class)	Dilated convolutions for wide context, first to evaluate multi-class memory malware detection

Table 5.23: Summary Table

Chapter 6

Conclusion

In summary, this study emphasizes how complicated cybersecurity threats are becoming, especially when it comes to malware that has been disguised and uses advanced evasion methods. We examined several machine learning applications including Adaboost, Decision Tree, Random Forest, SVM, and Ensemble Learning using the CIC MalMem 2022, CICMalDroid 2020, Ransomware 2024, CIC-AndMal2017 dataset to detect binary malware, malware categories, and distinct malware families.The highest accuracy achieved 99.99% for binary, malware category, Binary: 99.99% on both models, 4-class: 96% 94%(balanced dataset) on Random Forest-LightGBM 16-class: 99% 99.99%(balanced dataset) on AdaBoost-LightGBM.As cybercriminals continue to evolve their tactics, future research should focus on enhancing model robustness, exploring real-time detection capabilities, and integrating these approaches into comprehensive cybersecurity frameworks to safeguard systems against emerging threats.

6.1 Limitations of Our Study

Though we have developed a solid model that yields encouraging results, there are certain limitations we must acknowledge. We face two main limitations: the absence of a primary dataset and the fact that it has not been implemented in a real-world setting.

6.2 Further Research Area

The research scope of the project aims to enhance the model's predictive capabilities by addressing its current limitation. In the future, probably generated custom models improve malware family detection accuracy even further. Our objective is to cross-validate our trained models using a novel dataset that extends beyond the CIC MalMem 2022 dataset to further assess their robustness and generalizability.

Bibliography

[1] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment," *Applied Sciences*, vol. 12, no. 17, 2022.

[2] O. A. Madamidola, F. Ngobigha, and A. Ez-zizi, "Detecting new obfuscated malware variants: A lightweight and interpretable machine learning approach," 2024.

[3] K. S. Roy, T. Ahmed, P. B. Udas, M. E. Karim, and S. Majumdar, "Malhystack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis," *Intelligent Systems with Applications*, vol. 20, p. 200283, 2023.

[4] A. Sharma and S. K. Sahay, "Evolution and detection of polymorphic and metamorphic malwares: A survey," *CoRR*, vol. abs/1406.7061, 2014.

[5] TechRadar Pro, "Over 16 billion records leaked in "unimaginable" major data breach – here's what we know," *TechRadar Pro*, June 2025. Accessed: 2025-06-24.

[6] TechRadar Pro, "Over 16 billion records leaked in "unimaginable" major data breach – here's what we know." https://www.techradar.com/pro/website-building/over-16-billion-records-leaked-in-unimaginable-major-data-breach-heres-what-we-k June 2025. TechRadar Pro; accessed 24 June 2025.

[7] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan conference on security technology (ICCST)*, pp. 1–7, ieee, 2018.

[8] P. Black and J. Opacki, "Anti-analysis trends in banking malware," in *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 1–7, 2016.

[9] A. Mills and P. Legg, "Investigating anti-evasion malware triggers using automated sandbox reconfiguration techniques," *Journal of Cybersecurity and Privacy*, vol. 1, no. 1, pp. 19–39, 2021.

[10] T. Carrier., P. Victor., A. Tekeoglu., and A. H. Lashkari., "Detecting obfuscated malware using memory feature engineering," in *Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP*, pp. 177–188, INSTICC, SciTePress, 2022.

[11] L. Bruna Moralejo, "Machine learning for malware detection and classification," Master's thesis, Universitat Politècnica de Catalunya, 2023.

[12] L. Bruna Moralejo, "Machine learning for malware detection and classification," Master's thesis, Universitat Politècnica de Catalunya, 2023.

[13] M. Miraoui and M. B. Belgacem, "Binary and multiclass malware classification of windows portable executable using classic machine learning and deep learning," *Frontiers in Computer Science*, vol. 7, p. 1539519, 2025.

[14] R. Bakshi, S. Lingwal, K. C. Bhatt, S. Thapliyal, M. Wazid, and D. P. Singh, "A robust machine learning-based mechanism for detection and analysis of malware attacks," *Procedia Computer Science*, vol. 259, pp. 193–201, 2025.

[15] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," in *International Conference on Big Data Analytics*, pp. 402–411, Springer, 2018.

[16] E. Masabo, K. S. Kaawaase, J. Sansa-Otim, and D. Hanyurwimfura, "Structural feature engineering approach for detecting polymorphic malware," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 716–721, IEEE, 2017.

[17] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, 2024.

[18] T. Carrier, "Detecting obfuscated malware using memory feature engineering," 2021.

[19] B. Taşcı, "Deep-learning-based approach for iot attack and malware detection.," *Applied Sciences (2076-3417)*, vol. 14, no. 18, 2024.

[20] S. R. Hasan and A. Dhakal, "Obfuscated malware detection: investigating real-world scenarios through memory analysis," in *2023 IEEE International Conference on Telecommunications and Photonics (ICTP)*, pp. 01–05, IEEE, 2023.

[21] D. Cevallos-Salas, F. Grijalva, J. Estrada-Jiménez, D. Benítez, and R. Andrade, "Obfuscated privacy malware classifiers based on memory dumping analysis," *IEEE Access*, vol. 12, pp. 17481–17498, 2024.

[22] S. S. Shafin, G. Karmakar, and I. Mareels, "Obfuscated memory malware detection in resource-constrained iot devices for smart city applications," *Sensors*, vol. 23, no. 11, p. 5348, 2023.

[23] O. A. Madamidola, F. Ngobigha, and A. Ez-zizi, "Detecting new obfuscated malware variants: A lightweight and interpretable machine learning approach," *Intelligent Systems with Applications*, vol. 25, p. 200472, 2025.

[24] A. Tiwari, N. S. Chaudhari, *et al.*, "Obfuscated memory malware detection," *arXiv preprint arXiv:2408.12866*, 2024.

[25] S. Sharmila, A. Tiwari, and N. S. Chaudhari, "Obfuscated malware detection using multi-class classification," in *2023 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 170–175, IEEE, 2023.

[26] R. Hasan, B. Biswas, M. Samiun, M. A. Saleh, M. Prabha, J. Akter, F. H. Joya, and M. Abdullah, "Enhancing malware detection with feature selection and scaling techniques using machine learning models," *Scientific Reports*, vol. 15, no. 1, p. 9122, 2025.

[27] B. Mondal, S. S. N. C. Dukkipati, M. T. Rahman, and M. T. Y. Taimun, "Using machine learning for early detection of ransomware threat attacks in enterprise networks," *Saudi J Eng Technol*, vol. 10, no. 4, pp. 159–168, 2025.

[28] B. Mondal, S. S. N. C. Dukkipati, M. T. Rahman, and M. T. Y. Taimun, "Using machine learning for early detection of ransomware threat attacks in enterprise networks," *Saudi J Eng Technol*, vol. 10, no. 4, pp. 159–168, 2025.

[29] A. Shah and L. Nawaf, "Malware detection using deep learning approaches," 2024.

[30] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 1, p. 16, 2024.

[31] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "Memaldet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," *Computers & Security*, vol. 142, p. 103864, 2024.

[32] A. Öztürk and S. Hızal, "Detection and analysis of malicious software using machine learning models," *Sakarya University Journal of Computer and Information Sciences*, vol. 7, no. 2, pp. 264–276, 2024.

[33] O. A. Madamidola and D. Oladunni, "A framework for an iot-enabled intelligent inventory management system," *International Journal of Advances in Engineering and Management (IJAEM) Volume*, vol. 2, pp. 01–05, 2024.

[34] Z. Dai, L. Y. Por, Y.-L. Chen, J. Yang, C. S. Ku, R. Alizadehsani, and P. Pławiak, "An intrusion detection model to detect zero-day attacks in unseen data using machine learning," *PloS one*, vol. 19, no. 9, p. e0308469, 2024.

[35] W. K. Al-Ghanem, E. U. H. Qazi, T. Zia, M. H. Faheem, M. Imran, and I. Ahmad, "Mad-anet: Malware detection using attention-based deep neural networks," *CMES-Computer Modeling in Engineering and Sciences*, vol. 143, no. 1, pp. 1009–1027, 2025.

[36] M. Al-Qudah, Z. Ashi, M. Alnabhan, and Q. Abu Al-Haija, "Effective one-class classifier model for memory dump malware detection," *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, p. 5, 2023.

[37] O. Kırlar, G. P. Akın, and M. K. Pehlivanoğlu, "Makine öğrenmesi yöntemleri kullanılarak kötü amaçlı yazılım sınıflandırması: Cic-mammem-2022 veri kümesi üzerinde bir performans karşılaştırması," *Bilgisayar Bilimleri ve Mühendisliği Dergisi*, vol. 17, no. 2, pp. 165–173.

[38] Q. Liu, "Feature selection with random forest for ransomware detection," tech. rep., Institute for Homeland Security, 2025.

[39] R. Hasan, B. Biswas, M. Samiun, M. A. Saleh, M. Prabha, J. Akter, F. H. Joya, and M. Abdullah, "Enhancing malware detection with feature selection and scaling techniques using machine learning models," *Scientific Reports*, vol. 15, no. 1, p. 9122, 2025.

[40] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 515–522, IEEE, 2020.

[41] G. Padmavathi, D. Shanmugapriya, and A. Roshni, "Performance analysis of unsupervised machine learning methods for mobile malware detection," in *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 201–206, 2022.

[42] B. Kapoor, B. Nagpal, P. K. Jain, A. Abraham, and L. A. Gabralla, "Epileptic seizure prediction based on hybrid seek optimization tuned ensemble classifier using eeg signals," *Sensors*, vol. 23, no. 1, p. 423, 2022.

[43] A. Hussain, A. Saadia, M. Alhussein, A. Gul, and K. Aurangzeb, "Enhancing ransomware defense: deep learning-based detection and family-wise classification of evolving threats," *PeerJ Computer Science*, vol. 10, p. e2546, 2024.

[44] A. Mezina and R. Burget, "Obfuscated malware detection using dilated convolutional network," in *2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 110–115, 2022.

[45] D. Alsekait, M. Zakariah, S. U. Amin, Z. I. Khan, and J. S. Alqurni, "Privacy preservation in iot devices by detecting obfuscated malware using wide residual network.," *Computers, Materials & Continua*, vol. 81, no. 2, 2024.

Author Biographies

Muhammad Iqbal Hossain is an Associate Professor of the Computer Science and Engineering Department of BRAC University. His academic journey is marked by significant achievements and rigorous pursuits. Dr. Hossain earned his Ph.D. in Computer Engineering from Kyungpook National University, South Korea, in January 2018. During his doctoral tenure, he actively participated in the Embedded Software Engineering Lab. His contributions to this lab resulted in a substantial collection of peer-reviewed journals and presentations at both international and domestic conferences in Korea. Dr. Hossain's commitment to staying at the forefront of his field is evident through his active participation in various international conferences and symposiums, which have enriched his perspectives and insights. Dr. Hossain's scholarly pursuits have garnered him recognition beyond his academic accomplishments. His expertise has led him to assume significant roles as a Reviewer and Editorial Board Member for reputable international journals and conferences. Moreover, his remarkable academic journey was facilitated by prestigious scholarships, including the Korean Government Scholarship (GKS) in 2009, which enabled his graduate studies. Subsequently, his exceptional dedication and contributions were acknowledged with the KNU Honors Scholarship during his doctoral pursuit.

Dr. Iqbal's Research interests are mainly focused on Information security (Cryptography, Access control, privacy etc), AI and Machine learning, Model-based Software Engineering, Software Testing and Verification, Automation etc.

Yeasin Arafath is currently pursuing a B.Sc. degree in Computer Science and Engineering at BRAC University, Dhaka, Bangladesh. His primary research interests lie in the areas of cybersecurity, particularly malware analysis, threat detection, and system vulnerabilities. He is passionate about developing innovative approaches to secure digital systems and aspires to contribute to the advancement of secure computing environments.

In addition to cybersecurity, Mr. Arafath has a strong interest in emerging technologies such as quantum computing and artificial intelligence, with a focus on their intersection with secure system design. He is enthusiastic about academic research and aims to pursue advanced studies to further explore the convergence of AI, cybersecurity, and quantum technologies.

Apon Hossain an undergraduate student in the Department of Computer Science and Engineering at BRAC University, Bangladesh. His primary area of interest is cybersecurity, especially malware analysis and threat detection. He is passionate about understanding security vulnerabilities and aspire to contribute to creating safer digital environments through my future research and work.

Nazihah Islam Nawreen is currently pursuing a B.Sc. degree in Computer Science and Engineering at BRAC University, Dhaka, Bangladesh. Her research interests include cybersecurity, with a focus on malware analysis, threat detection, and system vulnerability assessment. She is passionate about developing secure digital infrastructures and contributing to the advancement of next-generation cybersecurity technologies.

In addition to her work in cybersecurity, Ms. Nawreen has a strong interest in emerging technologies such as quantum computing and artificial intelligence, particularly at their intersection with secure system design. She is currently working in the Product and Technology division at bKash Limited, where she is gaining hands-on experience in safeguarding critical financial infrastructure. She is enthusiastic about academic research and aims to pursue advanced studies to explore the convergence of AI, quantum technologies, and cybersecurity.

Tousiqul Islam Talukder is an undergraduate student in the Department of Computer Science and Engineering at BRAC University, Bangladesh. His strong passion is always for in cybersecurity field, more specificely interested in malware analysis and detection techniques which led me to take part in this malware-related studies. By this experience, He intend to continue his contributing to the field through advanced research and practical solutions that enhance digital security.

Raisa Tahiatul Aziz holds a Bachelor's degree in Computer Science from BRAC University. She has a keen interest in the field of cybersecurity, particularly in the development and advancement of secure technologies that safeguard digital systems. Her academic background has equipped her with a solid foundation in computing principles, while her passion for secure system design drives her continuous exploration of emerging threats and innovative defense mechanisms. She aspires to contribute to the cybersecurity domain through research, practical implementation, and the promotion of resilient digital infrastructures.