Al-Quds Open University

Faculty of Graduate Studies and Scientific Research

Master of Information Technology

Ransomware Detection: The Efficacy of Behavior-Based and Machine Learning Techniques

الكشف عن فيروس الفدية: فعالية تقنية السلوك والتعلم الآلي

MSc IT THESIS

by

Manar Yousef Amro

Student ID: 0330012310097

Supervisor

Dr. Mohamed Dweib

Submitted in Partial Fulfillment of the Requirements

For the Master of Information Technology, Faculty of Graduate Studies

Ramallah, Palestine

January 2026

جامعة القدس المفتوحة

عمادة الدراسات العليا والبحث العلمي

ماجستير تكنولوجيا المعلومات

الكشف عن فيروس الفدية:

فعالية التقنيات القائمة على السلوك والتعلم الآلي

رسالة ماجستير

إعداد الطالبة

منار يوسف عمرو

الرقم الجامعي: 0330012310097

إشراف:

د. محمد ذويب

قدّمت هذه الرسالة استكمالاً لمتطلبات الحصول على درجة ماجستير تكنولوجيا المعلومات

في عمادة الدراسات العليا والبحث العلمي

رام الله – فلسطين

يـ ايـ 2026

Examination Committee Page

الكشف عن فيروس الفدية: فعالية تقنية السلوك والتعلم الآلي

Ransomware Detection: The Efficacy of Behavior-Based and Machine Learning Techniques

By:

Manar Yousef Amro

Supervisor:

Dr. Mohammad Dweib

The thesis was examined and approved on January 5th , 2026.

Members of the Examination Committee

Supervisor and Chairman: Dr. Mohammad Dweib Al-Quds Open University

Signature:.....

Committee First Member: Dr. Muath Subha Arab American University

Signature:.....

Committee Second Member: Dr. Hekmat Darawsheh Al-Quds Open University

Signature:.....

Al-Quds Open University

2026

Declaration

I, Manar Yousef Ahmad Amro, hereby declare that the work presented in this thesis has not been submitted for any other degree or professional qualification, and that it is the result of my own independent work.

Signed: 

Date: January, 5th 2026

Abstract

Ransomware remains one of the most pervasive cybersecurity threats, exploiting both technological and human vulnerabilities to inflict severe economic and operational damage. This thesis investigates the efficacy of hybrid detection methodologies that integrate behavior-based analysis with machine learning (ML) and deep learning Long Short-Term Memory (LSTM) approaches to improve detection accuracy and generalization across diverse ransomware variants.

The proposed framework unifies three behavioral dimensions—File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA)—into a comprehensive dataset of 15,411 instances and 224 features, aligned through a Timestamp-Based Integration process. Multiple classifiers, including Random Forest, Naïve Bayes, **Support Vector Machine (**SVM), Gradient Boosting, and LSTM, were trained and evaluated. Two integration strategies—decision-level fusion (voting) and model-level stacking- were compared empirically to identify the most robust hybrid configuration.

Experimental results demonstrated that the stacking ensemble [BB, XGB, NB] achieved superior macro-average performance (F1 $\approx$ 0.93, AUPRC $\approx$ 0.91), validating the advantage of multi-model learning for ransomware detection. Additionally, Synthetic Minority Over-sampling Technique (SMOTE) balancing and probability calibration improved fairness and stability across minority ransomware families such as Ryuk, Sodinokibi, and LockBit.

The study also incorporated statistical validation (McNemar's test) and sensitivity analysis to ensure the reliability of results under variable conditions. Finally, ethical and policy considerations were highlighted to guide the responsible deployment of AI-driven cybersecurity systems.

This research bridges a major gap in ransomware detection studies by operationalizing a cross-domain hybrid framework that synchronizes host and network behavioral data, providing a replicable and scalable foundation for intelligent, interpretable, and ethically aligned ransomware defense systems.

Keywords: Ransomware Detection, Behavior-Based Analysis, Machine Learning, LSTM, Ensemble Learning, Stacking, FSM, PBA, NBA, SMOTE, Cybersecurity.

الملخص

إطار هجين للكشف عن برامج الفدية من خلال تكامل تحليل السلوك وتقنيات التعلّم الآلي

تتناول هذه الرسالة مشكلة تزايد هجمات برامج الفدية التي أصبحت من أخطر التهديدات للأمن السيبراني نظرًا لقدرتها على تعطيل الأنظمة وإحداث خسائر جسيمة. تهدف الدراسة إلى تطوير إطار كشفٍ هجين يجمع بين تحليل السلوك وخوارزميات التعلّم الآلي والعميق لتحسين دقّة الاكتشاف وقدرة النماذج على التعميم عبر أنواع برامج الفدية المختلفة.

قام الباحث بتصميم وتنفيذ نموذج متكامل يدمج ثلاث طبقات سلوكية :مراقبة نظام الملفات(FSM) ، تحليل سلوك العمليات(PBA) ، وتحليل سلوك الشبكة(NBA) ، ضمن قاعدة بيانات موحّدة تضم (15,411) عينة و(224) ميزة، تم دمجها زمنيًا باستخدام آلية الدمج القائم على الطوابع الزمنية .جرى تدريب عدة مصنّفات تقليدية وحديثة مثل Random Forestو Naïve Bayesو XGBoostوLSTM، مع تطبيق تقنيات الموازنة (SMOTE) ومعايرة الاحتمالات لتحسين الإنصاف والاستقرار.

أظهرت النتائج أن نموذج التكديس (Stacking Ensemble) الذي يدمج بين السلوكيات وخصائص التعلّم الآلي حقق أفضل أداء بمعدل $F1 \approx 0.93$، متفوّقًا على نماذج التصويت التقليدية، مما يؤكد فعالية التكامل متعدد المستويات في تعزيز دقّة الكشف وتفسير النتائج. كما تم إجراء اختبارات إحصائية وتحليل حساسية للتحقق من استقرار النماذج تحت ظروف مختلفة.

يستخلص من هذا العمل أن الدمج بين التحليل السلوكي والتعلّم الآلي يقدّم حلاً واعدًا لتحقيق كشف ذكي، قابل للتفسير، ومتكيّف مع تطوّر الهجمات. وتساهم هذه الرسالة في سدّ فجوة معرفية تتمثل في غياب إطار موحّد يدمج السلوكيات على مستويات النظام والشبكة ضمن نموذج واحد، مما يفتح المجال لتطوير أنظمة كشف هجينة أكثر مرونة وموثوقية في الأمن السيبراني الحديث.

Associated Publications

Amro, M. Y., Dwieb, M., Hammad, J. A. H., & Wibawa, A. P. (2024). Ransomware detection: Patterns, algorithms, and defense strategies. Bulletin of Social Informatics Theory and Application, 8(1), 165–172. Retrieved from https://pubs.ascee.org/index.php/businta/article/view/689/343

Acknowledgements

I would like to express my sincere appreciation to my supervisor for the guidance, support, and constructive feedback provided throughout this research. I also extend my gratitude to the examination committee for their valuable comments and insightful recommendations. I am thankful to my family for their patience, encouragement, and continuous support during the completion of this dissertation.

Manar Amro

January 2026

Dedication

To everyone who believed in me when the journey felt heavy.

Manar Amro

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
API Call	Application Programming Interface Call
AUPRC	Area Under the Precision–Recall Curve
BB	Behavior-Based
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
Cuckoo	Cuckoo Sandbox (Dynamic Malware Analysis Platform)
DL	Deep Learning
DPIA	Data Protection Impact Assessment
EDR	Endpoint Detection and Response
FN	False Negative
FP	False Positive
FSM	File System Monitoring
F1	F1-Score (Harmonic Mean of Precision and Recall)
GAN	Generative Adversarial Network
GB	Gradient Boosting
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
GPU	Graphics Processing Unit
IDS	Intrusion Detection System
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
ML	Machine Learning
NBA	Network Behavior Analysis

Abbreviation	Full Form
NB	Naïve Bayes
NF	Negative False (per Family)
NT	Negative True (per Family)
OS	Operating System
PBA	Process Behavior Analysis
PCA	Principal Component Analysis
PF	Positive False (per Family)
PT	Positive True (per Family)
RaaS	Ransomware-as-a-Service
RF	Random Forest
ROC	Receiver Operating Characteristic
ROC-AUC	Receiver Operating Characteristic – Area Under Curve
SIEM	Security Information and Event Management
SMOTE	Synthetic Minority Oversampling Technique
SOC	Security Operations Center
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
VAE	Variational Autoencoder
XAI	Explainable Artificial Intelligence
XGB	Extreme Gradient Boosting

Chapter 1: Introduction

1.1 Overview and Background

Ransomware is a type of malware that blocks access to users' files or systems to obtain their private information. The source of the virus demands the files' owner for ransom, and if it's not paid in time, the ransomware will lock the files. There will be no way to access the files or information again, and even if some of it could leak, it could cause significant losses and damage to various institutions **(Lee, Yun, & Lee, 2024).** The AIDS Trojan, which is sometimes referred to as the "Cyborg PC virus," is the first known ransomware, and it was released in 1989. This was malware that locked files and threatened to restore them only upon a fee of $189. From the info provided, you can state that while cryptographers face devastating loss and hardly any methods of recovery seem to be practical, it is safe to state that serious amounts of AI advance the opportunities for crypto hers—attacked PSWs have transformed themselves **(Razaulla et al., 2023)**.

In its earliest stages during the late 1980s, ransomware appeared in a relatively simple and primitive form, mainly targeting individual users or small organizations. However, as defensive mechanisms and cybersecurity solutions evolved, attackers responded with increased sophistication—enhancing encryption algorithms, automating distribution channels, and developing more stealthy infection techniques. This continuous co-evolution between security measures and offensive tactics has transformed ransomware from isolated incidents into a highly organized, large-scale threat ecosystem that exploits both human and technological vulnerabilities.

A milestone event illustrating the earliest concept of ransomware occurred in 1989, when Dr. Joseph Popp distributed twenty thousand infected floppy disks labelled *"AIDS Information"* during the World Health Organization's AIDS conference. After a predetermined number of system reboots, the malware displayed a ransom message demanding $189 to restore access—an early demonstration of digital extortion (View of the evolution of ransomware, 2024). Since that incident, ransomware has evolved from isolated experimentation into a persistent global threat. For instance, data from the Korea Internet & Security Agency (KISA) recorded over 60,000 ransomware programs detected monthly during 2022, emphasizing the exponential growth of such attacks (**Lee, Yun, & Lee, 2024**). This escalation parallels the diversification of delivery methods, from

phishing emails and drive-by downloads to sophisticated exploitation of system vulnerabilities—laying the foundation for the distinct attack models discussed below.

- **Advanced Encryption**
- **and Distribution:** Modern ransomware relies on advanced encryption algorithms that make traditional recovery methods almost impossible, forcing victims to consider ransom payment as the only practical option. The continuous evolution of cryptographic tools has enabled attackers to enhance both encryption speed and complexity, contributing to the rapid escalation of ransomware incidents worldwide. These advancements explain why decryption without the attacker's private key remains infeasible, reinforcing the financial incentive behind such attacks.
- **Ransomware-as-a-Service (RaaS)**: The RaaS model is understood as a further evolution of the ransomware business. It points out how the attackers design easy-breezy interfaces that permit novice cybercriminals to execute complex ransomware attacks without any technical know-how. Ransomware as a Service (RaaS) is on the same spectrum as SaaS, where people who have developed ransomware and are willing to share the profits with other people who deploy it. IT also represents a dramatic decrease in the threshold needed to perform ransomware attacks, which are, therefore, more widespread.
- **Double Extortion**: The double extortion is mentioned as an answer to the enhancement of the defences and case of data loss situations in general, and backup in particular. The attackers not only encrypt the data, but they also steal it before the encryption phase. I will make the payment; otherwise, I will guarantee that my sensitive information will get out. This technique has worked especially well for organizations that deal with sensitive clients or proprietary information by providing additional reputation and legal ramifications on top of operational displacement.
- **Targeted Attacks:** Ransomware campaigns have shifted from random, large-scale infections to highly targeted operations aimed at specific industries such as healthcare, finance, and government agencies. These attacks are carefully planned, involving reconnaissance of the victim's network infrastructure and data value before deployment. This strategic targeting reflects the maturing

professionalism of ransomware groups, which now treat cyberattacks as business ventures aimed at maximizing financial returns (**Patel & Yashaswini, 2024**).

The other classification includes crypto, locker, and scareware. Users' files are encrypted with the use of crypto, and a ransom payment is sought in order to get a decryption key. Unlike crypto, locker ransomware does not encrypt files but rather locks a part of the operating system, which means that the user cannot get to their desktop or even their files, and sometimes the entire computer may be locked out. The third type, scareware, attempts to trick users by using pop-up messages or alerts that state that there is a problem with the system, and through these messages or alerts tries to intimidate the user into buying software that can fix issues that do not exist at all. Figure 1 shows the three main types **(Li, Yang, & Shao, 2024).**

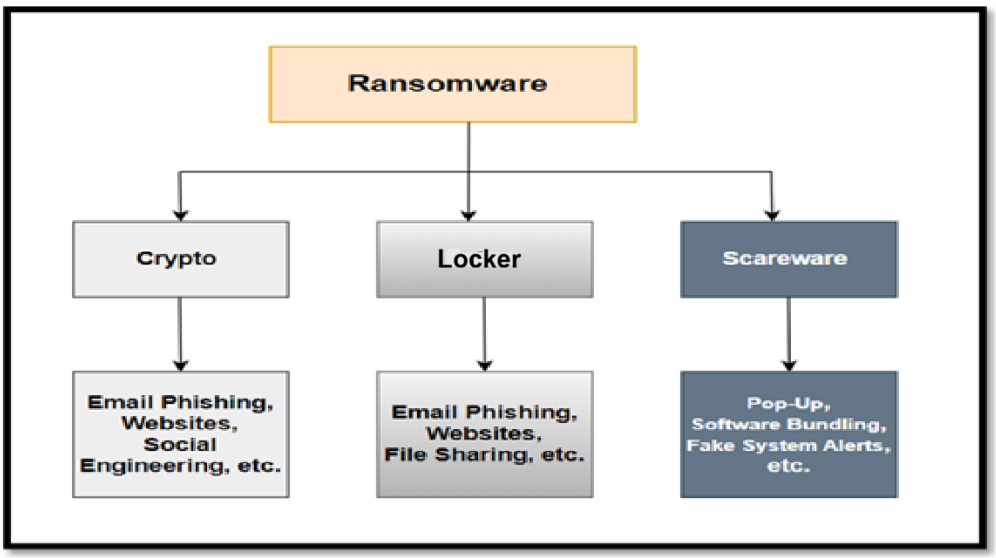

Figure 1.*ransomware types*

Attacks by ransomware don't happen suddenly; it's gone throw few phases. The first phase is the start of the infection, which means installing components that infect devices and systems with the virus, and they can be installed in many ways and in the most popular and widespread ways: phishing emails, drive-by download, strategic web compromise, exploiting vulnerabilities in internet or network accessible systems. The second phase is "Installation," which starts when the virus stabilizes itself within the system or in a common Windows process. The third phase is Command-and-Control, where the ransomware will try to contact its command servers, asking for instructions.

Some kinds of this virus will send back to the owner a significant amount of system information, including IP address, domain name, operating system, installed browsers, and anti-malware software. The fourth phase is destruction, at this point the virus begins to make changes to the device or system as shown in figure 2, where the files that have been targeted in previous stages are encrypted, and it is not only encrypted the content of the files but in some cases, it encrypts the names of the files also so that the user cannot distinguish them or know the extent of the attack. The last phase is Extortion where, in this stage, the payment demand message appears for the user or the file's owner **(Anghel & Racautanu, 2024).**

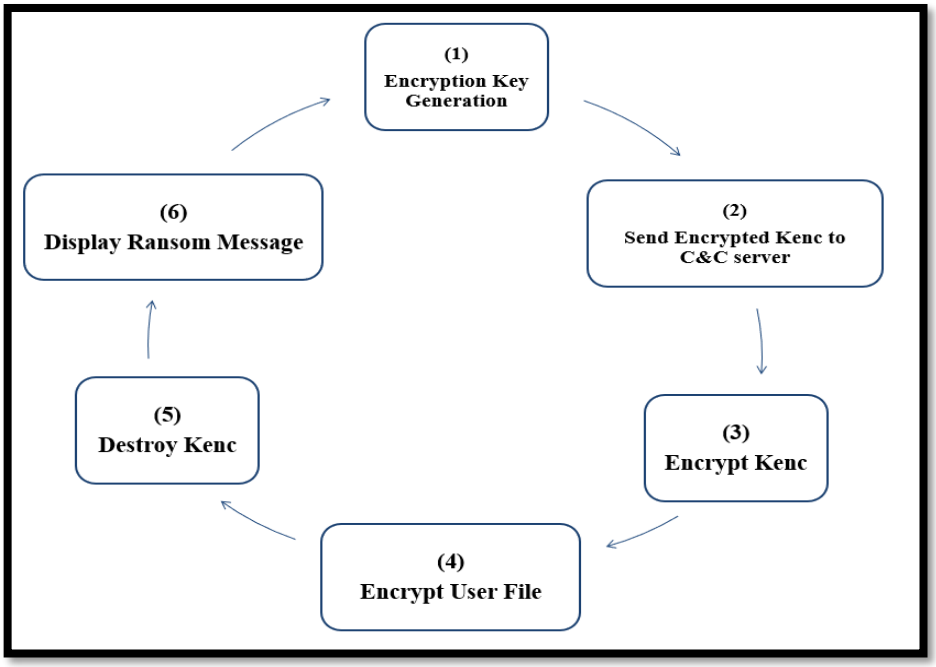

Figure 2.*General crypto-ransomware key lifecycle*

Ransomware attacks are not limited to a particular type of enterprise or area of business, but have become a major threat to all areas and institutions of different sizes. As attacks evolve, the results have become devastating even for infrastructure and manifest on different sectors, an example of this is the Cologne pipeline incident in 2021, where computerized equipment running the pipeline was attacked, and the company stopped all operations. To regain control of its equipment, the company paid the USA $4.4 million **(Nagar, 2024).** According to NCC Group's Global Threat Intelligence team, 502 attacks have been registered in July 2023, which is 16% increase from June 2023, which

registered 434 attacks, and more than twice the number of ransomware attacks observed in July 2022. These attacks cost many companies and users, where Cybersecurity Ventures expects a loss of $265 billion by 2031 due to these attacks; there is a big difference from the $5 billion ransomware targets shelled out in 2017 **(Pratt, 2023).** Ransomware has unique characteristics from other viruses and cyber threats, which is encryption, and when it comes to operating inside the Encryption phase of the attack. There are three major groups for ransomware detection:

- API and system call monitoring-based detection.
- I/O monitoring-based detection.
- File system operations monitoring-based detection that includes scanning for high entropy in files and monitoring deception tokens in file system-based detection.

Despite its occasional treatment as its own methodology, machine learning (ML) is actually a combination of the three approaches listed above, with the specifics depending on the features utilized in the training dataset **(Begovic, Al-Ali, & Malluhi, 2023).**
Identifying the types of ransomware viruses and how they work is the cornerstone of their detection, so that it is studied and the use of the most suitable method for each of the most famous methods of detecting ransomware:

1- Machine Learning: makes use of algorithms that fall into various categories, such as decision trees, clustering, regression, ensemble, neural networks, regularization, instance-based, deep learning, and decision trees. By examining and categorizing actions, these algorithms are used to identify ransomware.
2- Honeypot: it's useful for collecting data concerning attacks, such as user identities and the scope of their activity, which helps with defense strategy decision-making.
3- Statistics: Statistical analysis could provide light on ransomware's defining features. Statistical analysis is a common tool for identifying ransomware since it can spot suspicious patterns of behavior and indicate when encryption is present.

Table 1 shows the differences between the three detection approaches **(Yamany et al., 2024).**

Table 1.*Comparison between ransomware detection approaches*

Ransomware Detection Approach	**Advantages**	**Disadvantages**
Machine Learning	allows for the automatic identification of patterns and relationships within large datasets. This can be particularly useful for identifying new and emerging threats, as the model can learn from past data to identify patterns and make predictions about future threats. Machine learning algorithms can also be trained on a wide variety of data types, including text, images, and audio, which makes them useful for detecting ransomware in different formats.	Machine learning algorithms can be vulnerable to bias and can produce inaccurate results if the training data are not representative of the real-world data. They also require frequent retraining to ensure that they continue to perform well as the data distribution changes.
Honeypot	Allows researchers to gather valuable data and intelligence about the tactics, techniques, and procedures (TTPs) used by attackers. This information can be used to improve the effectiveness of ransomware detection and prevention measures. Additionally, honeypots can help mitigate the impact of ransomware attacks by preventing the malware from reaching the target system or data.	The risk of false positives, where legitimate activity is mistaken for malicious activity. Another issue is the cost and resources required to maintain and operate a honeypot, as well as the potential legal and ethical considerations. Additionally, honeypots may not be suitable for all types of environments or organizations and may not provide comprehensive protection against all types of ransomware attacks.
Statistical	Allows researchers to gain a deeper understanding of ransomware behavior and identify key trends that can inform prevention and detection efforts.	It relies on the availability of accurate and comprehensive data, which may be difficult to obtain in some cases. Additionally, statistical analysis may not be able to identify specific instances of ransomware in real time, making it less effective for immediate detection and response.

1.2 Motivation

Ransomware truly is one of the most dangerous because of how quickly it gets spread and also how it is able to integrate itself into multiple technologies, whether it is used in mobile or fixed devices. Once the ransomware virus infects the device, there is no way out but to pay the ransom, as the only option is known to be extremely unreliable. Sooner or later, the best way to tackle such a problem is to detect the ransomware before it reaches the devices. And that is why advanced ransomware detection systems come into great prominence.

This is where the crux of the problem arises, as although the first detection of Zero-Day Threats would be the most appropriate solution, several things need to be considered and evaluated, for instance, the large number of variants making traditional models ineffective, as the detection-based deep Learning models are complex themselves. While there are several other complexities, such as the high False Negative and high Positive Rates in heuristic or behavior-based detection methods in the new world of behavioral and heuristic-based deception technology. These complications can slow down the effective introduction of scalable defeating methods and detection systems.

Our primary interest in this study is centered on the Behavior-Based Detection Methods and Machine Learning Techniques that were also the focus of prior studies. The signature-based detection approach and many others are inadequate in dealing with new strains of ransomware, but through Machine Learning, there are fewer false positives and better real-time analysis and learning. Even though Behaviour Based Detection is quite promising, it does not seem to be getting much attention, but some preliminary studies seem to be suggesting otherwise. This research seems to suggest integrating the two approaches with the aim of designing a system that enhances detection capabilities and focuses on the relevant aspects for improving future detection techniques of ransomware.

1.3 Problem Statement

To address this research gap, the thesis adopts a hybrid experimental methodology that integrates both behavior-based detection and machine learning approaches. Behavior-based analysis is conducted through File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA) to capture host- and network-

level activities of ransomware. In parallel, multiple machine learning algorithms—such as Random Forest, SVM, Naïve Bayes, Gradient Boosting, and an LSTM sequence model—are trained and evaluated using the same curated datasets.

These two approaches are then systematically fused through feature-level and model-level integration strategies (decision-level voting and stacking) to compare their individual and combined performance. The experimental workflow includes dataset preprocessing, feature engineering, balancing (SMOTE), and sensitivity analysis to ensure robustness and generalization. Through this integrated methodology, the research bridges the gap between theoretical limitations of existing detection systems and the practical development of an adaptive, real-time ransomware detection framework.

1.4 Research Objectives

Building on the aim of improving real-time ransomware detection through behavior-based and machine learning (ML) approaches, this study addresses the following research questions (RQs):

1. What is the comparative detection performance of behavior-based models (FSM, PBA, NBA) and machine learning classifiers when applied to the same dataset?
2. How does feature fusion between host-level (FSM/PBA) and network-level (NBA) data influence the overall detection accuracy and reduction of false positives?
3. Which integration strategy—decision-level fusion (voting) or model-level fusion (stacking)—yields the most consistent results across accuracy, precision, recall, and F1-score metrics?
4. How do feature engineering techniques (e.g., feature selection, balancing, and noise injection) affect the generalization and robustness of ML models against ransomware variants?
5. To what extent can the proposed hybrid detection framework improve the real-time responsiveness of ransomware defense systems compared to standalone approaches?

Objectives of the Study:

- To design and test a hybrid feature fusion pipeline integrating FSM, PBA, and NBA data.
- To implement and evaluate multiple behavior-based and ML-based ransomware detection models using diverse datasets.
- To identify optimal strategies for feature engineering and class balancing that maximize model reliability and real-time applicability.
- To assess decision-level and model-level integration methods for enhancing classification performance.
- To develop a scalable and interpretable hybrid detection framework capable of real-time adaptability and performance stability.

1.5 Thesis Contribution to the Field/ Significance and /or Impact of the Research

This research work makes progress in ransomware detection by assessing how effective behavior-based and machine learning (ML) methods are. It offers an evaluation of different ML algorithms, exposing their abilities to decrease false positives and improve detection. Furthermore, it examines techniques based on behavior, addressing a gap in current research, and shows their ability to detect ransomware by analyzing behavioral patterns. By adjusting dataset features and evaluating the effectiveness of these methods, the research provides valuable ways for enhancing precision and flexibility in detection systems. Additionally, it shows the advantages of combining ML and behavior-based methods, opening up avenues for further studies and enhanced protection against ever-changing ransomware dangers.

Furthermore, the strategic integration of behaviour-based and machine learning techniques enhances the ability of detection systems to identify complex ransomware behaviours in real-time. This approach addresses existing gaps in ransomware detection and provides a scalable framework for future enhancements in the field.

1.6 Thesis Outline

This thesis is divided into six chapters, which contain essential details on the research regarding the application of ML and behavioral approaches for the detection of ransomware:

Chapter 1: Introduction, this section includes the first portions, such as the context of the study, motivation, problem statement, aims, objectives, and significance. It also elaborates on the research scope and questions, along with the purpose of the study. Chapter 2: Literature Review, this Chapter identifies methods already in use and ways to detect ransomware, including traditional approaches, methods with machine learning, and behavioral approaches. The analysis also looks into gaps in the research to be filled in this particular study. Chapter 3 and Chapter 4: Methodology, this chapter gives the structure and type of the research activity undertaken. Such aspects as dataset selection, data preprocessing, feature extraction, and the setup for experimental use of ML and behavioral approaches are justified and described. Chapter 5: Experimental Results and Evaluation, this chapter describes the process of evaluation of ML algorithms and behavior approaches. General overview of the paragraphs involved experimental results, comparative performance metrics, as well as critical and analytical descriptors of the work. Chapter 6: Conclusion and Future Work; this chapter presents the obstacles met while carrying out this work and the limitations of the proposed methods. Also, the chapter includes some ideas for future use in research to improve strategies for malware detection.

Chapter 2: Literature Review

2.1 Introduction

The sophisticated world is confronted with a fundamental challenge by the fast evolution and complexity of malware. There are a number of security measures in place, including several antivirus techniques, and new methodologies are being explored to control and mitigate malware harm. Antivirus methods are classified according to their signatures and behaviors.

There are two primary categories of antiviral strategies: signature-based and behavior-based. Malware detection and prevention strategies and underlying concepts differ across domains. Figure 3 below graphically depicts the sequential processes necessary in both behavior-based and signature-based antiviral methods. These methods are crucial for network and system security because they detect and counteract many forms of malware while safeguarding user data and privacy. **(Azeem et.al, 2024)**

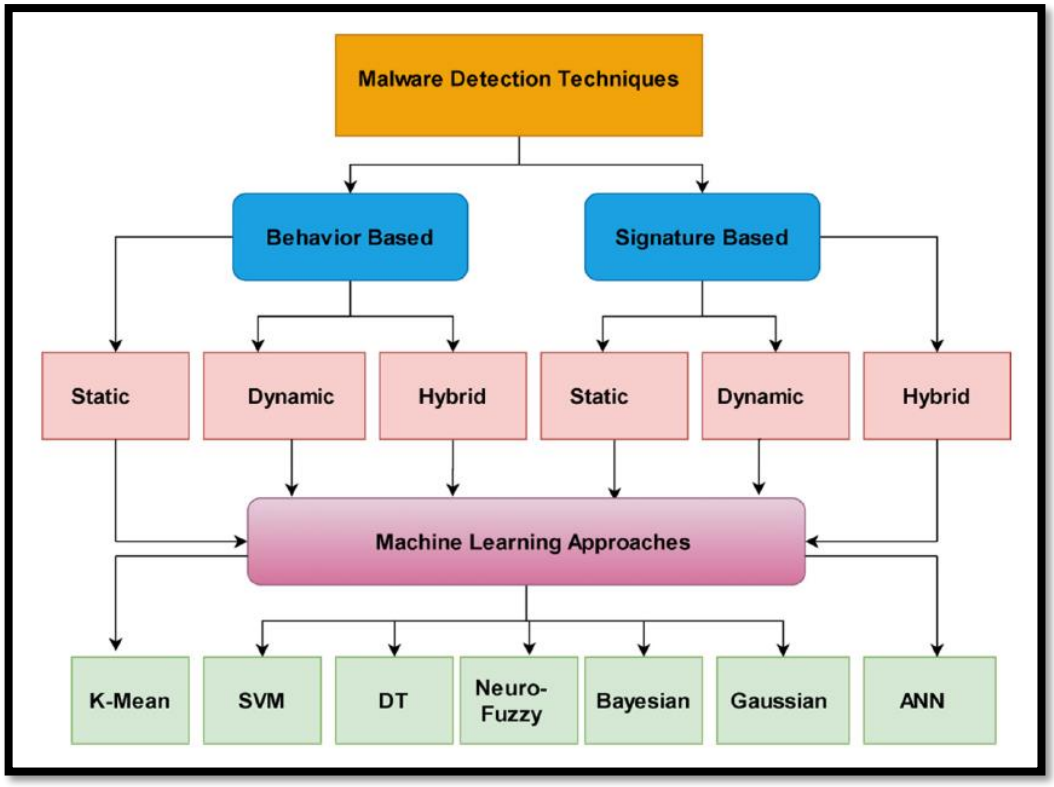

Figure 3.*Malware detection techniques*

Nevertheless, there is a present gap in the literature regarding research on the topic of ransomware early detection. In this research, we will analyze detection techniques for

ransomware viruses to identify the most efficient methods and explore ways to enhance their effectiveness.

2.2 Overview Ransomware

2.2.1 Ransomware evolution

Considering the evaluation of ransomware, it can be divided into three time periods as follows:

Germination period (1989–2009). At this time, ransomware attacks were just beginning to emerge. The frequency of ransomware assaults increased gradually, but they were still relatively small in scale and caused minimal damage. The ransomware virus known as Gpcode, which utilized the RSA asymmetric encryption method with larger passwords, was first introduced in 2006 or possibly much earlier and AK was developed with the intention of significantly enhancing the level of difficulty in decrypting ransomware, the next period is the Activation period which it was (2010-2016), Attacks using ransomware are becoming more common, and new forms come out almost every year, there was few important events during this time. In 2013, the Cryptolocker ransomware was developed, and in 2014, it utilized the GameOver Zeus botnet infrastructure for dissemination. In March 2014, CryptoWall had emerged as a significant ransomware menace. Simultaneously, the increasing popularity of Bitcoin has encouraged ransomware perpetrators to seek greater anonymity, leading to a consistent rise in requests for Bitcoin ransom payments. In 2014, there was a rise in malware that specifically targeted mobile devices using cryptographic techniques. Additionally, there was an emergence of ransomware that targeted Mac systems, such as KeRanger in 2016 and 'Patcher' (also known as 'filezip'). In 2015, a new business model known as Ransomware-as-a-Service (RaaS) emerged. The last period is the explosion period (2017-present). During this period, there has been a significant and progressive increase in attacks, such as the WannaCry attacks that occurred in 2017 and took advantage of a vulnerability in the Microsoft SMBV1 application called EternalBlue. These attacks caused a worldwide outbreak, affecting 230,000 machines in 150 countries, and resulted in losses exceeding $8 billion. At this time, several notable ransomwares and their groups have emerged, including LeatherLocker (2017), GandCrab (2018), LockerGoga, PureLocker, and Zeppelin (2019). In 2020, we witnessed the emergence of Phobos, Fonix, Conti, Cerber, Egregor, WastedLocker, and Maze. LeakerLocker, a distinctive form of mobile

ransomware that emerged in 2017, specifically aimed at Android smartphones by employing the tactic of threatening to disclose personal data instead of encrypting files. GandCrab underwent enhancements such as incorporating file encryption and data exfiltration capabilities, while Maze implemented the "double ransom" strategy, which involves demanding money for both decrypting the files and removing the stolen data **(Cen et al., 2024).**

According to **Zimba et al. (2021),** the development of the features of ransomware, including deletion and encryption of files, has resulted in an increase in the number of new domineering ransomware, mostly applicable in CAT5 categories, as shown by the statistics on ransomware attacks. The increase in the number of ransomware attacks has been recorded as a 229% rise, whereby a majority of them either fall under the CAT4 or CAT5 categories. Ransomware of the CAT2 category has been disappearing in recent years due to its poor design, while strong implementations of CAT5 have been in control since 2016. Ransomware of the CAT5 category uses powerful hybrid cryptosystems and advanced techniques for file erasure, including the removal of volume shadow copies as well as remnant files through the use of embedded RSA keys and AES encryption, as seen in WannaCry. Confirming the importance of (RaaS) as an important service in the evolution of ransomware **Oz et al. (2022)** explain how the attackers learned from their past experiences and the technological advancements; cybercriminals have now even started serving ransomware as a service (RaaS), and that is because they have perfected the components of ransomware attacks, such as payment methods and worm-like capabilities. And, almost immediately after Bitcoin's emergence, it aided ransomware creators in acquiring cryptocurrencies, which were effective methods of overcoming major operational bottlenecks. Notably, by 2010, 40,000 new ransomware families were discovered. Gradually, ransomware authors became the number one cyber threat, learning from their failures and advancements in technology. As a consequence, the evolution of ransomware has shaped the way end-users, organizations, and critical infrastructures operate around files and systems, as now, practically, everything requires a ransom payment or a restore from backup to get into operating conditions. Recently, there have been hacking attacks using ransomware that include human coordination in the work done; a clear example is the Ryuk and Conti variants. They include manually breaking into networks where an attacker first spies on the network to see who will enable the greatest effect of the ransomware being used. This development has undoubtedly been

reinforced by sophisticated monetization techniques, such as double and triple extortion. Apart from the usual data encryption and taking sensitive data out from the target, the attackers have graduated to other forms of threats, including the hacking of the stored information on the target's site, the beginning of a legal suit against the targets, and legal redress unless the money is delivered. These developments highlight the evolution of ransomware into a complex and highly lucrative business (**Fachkha et al., 2023)**.

2.2.2 Types of Ransomwares

According to **Smorti (2022/2023),** there are two main classes of Ransomware. The first one is Locker ransomware. This class blocks access to a computer resource, but there are ways to get the system back to how it was before without losing data. The second one is Crypto ransomware, which enables the system to function properly by exclusively impacting the data contained on it through the utilization of encryption techniques. Data becomes rendered worthless unless the victim possesses the decrypting key, which is sometimes provided to the victim after paying the ransom. In other instances, money is simply stolen, and the data is irretrievably lost.

2.2.3 Types of Ransomware Attacks

Regarding Ransomware attacks, **Cen et al. (2024)** discuss that Ransomware attacks can be classified into three main bases (Platform-based, security-based, Target-based), and the details of the three types are illustrated in Figure 4.

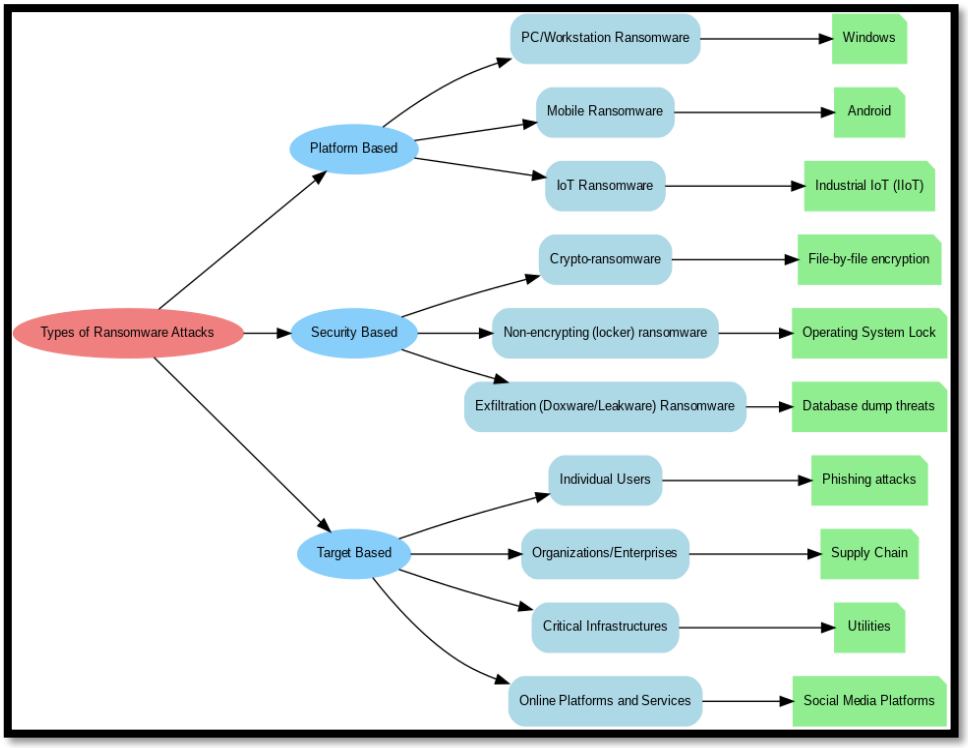

Figure 4.*Type of ransomware attacks*

2.3 Behavior-Based Detection Techniques for Ransomware

Indicate the research done by **Kwon, Kim, & Lee (2022)** that technological development and increased reliance on modern technology in all spheres of life, and the interaction of networks and systems by working with each other, have increased human welfare, but at the same time have increased concerns and challenges related to cybersecurity. Industrial control systems (ICS) are among the most important systems at risk from cyber and are clear and powerful targets. And because of the importance of these systems, they can't just rely on network security solutions, so they also rely on intrusion detection systems (IDS) which categorized into two types based on their detection methods the first one is Signature-Based and the second is Behaviour-Based, in the research they combine signature- and behaviour-based methods and compare the performance of the proposed method to those of previous detection methods that applied various machine learning approaches. There were a few limitations, like the used method only detects the fact that an anomaly occurred, but does not determine which sensors or actuators were attacked. The result outperformed the hybrid detection method in terms of detection accuracy and also reduced the time required to execute the detection process.

This was not the only research study IDS, **Agate et al. (2022)** discussed it using (Behaviour-Based Approach, Multi-Layered System, and Ensemble Learning Technique). The IDS presented is described as "behaviour-based," meaning it relies on analysing and classifying network traffic based on its behaviour rather than static signatures or known attack patterns. The system aims to detect and classify traffic based on behavioural characteristics, which is a hallmark of behaviour-based detection. The multi-layered IDS, which is based on behaviour analysis, detects nine different attack types. The tests demonstrated its reliability and accuracy in detecting malicious traffic, as well as its time efficiency, which means the system is reliable.

Researchers didn't only discuss the behavior – based technique, they also investigated how well current behavioural ransomware detectors can resist evasion techniques and introduced several innovative methods that ransomware can use to bypass these detection systems. Several techniques are used based on behavior, like UNVEIL detect suspicious activities by scoring various behavioral indicators, such as changes in file entropy, extensive file writes, and file deletions. Also, CryptoDrop uses a "reputation score" based on file type changes, content similarity, and entropy measurement, with additional indicators like file deletions and limited file types accessed. There is ShieldFS, which is an advanced technique designed for ransomware detection at the file-system level, offering the ability to transparently roll back file changes made by malicious processes. All these are behavior technique use for ransomware detection applied to measure the effectiveness of those techniques with different evasion tactics used by the virus. As a result of the experiment, we found that behavior-based detection techniques for ransomware are not entirely ineffective but do face notable challenges like Evasion Capabilities, Attack Methods, and Practical Implications, which indicates that these methods may not be fully effective against all types of sophisticated ransomware **(Gaspari et al., 2022)**. Working on the detection of the ransomware virus and its importance was not only confined to industrial systems and computers, but was also researched on this issue with smart mobile devices as well. The research is centred on behavior-based detection; they use KRDroid Detector, which is specifically behavior-based, analyzing runtime behaviours to differentiate between ransomware and benign applications with similar behaviours. This was only for the Android system for Mobile Devices, but it came with good result it detects 1809 of 1862 unseen ransomware with an accuracy of 97.5%. Additionally, the use of behavior-based

techniques for Android ransomware detection, implemented in a controlled environment, showcases a practical solution to a significant security challenge **(Wang et al., 2021).**

Urooj et al. (2023) tackled the problems of behavioural drift and lack of data during ransomware strikes by proposing a new strategy for the problem of behaviour-based detection. Weighted Generative Adversarial Networks (wGAN) were used to overcome the absence of available pre-encryption data by crafting synthetic data. Tackle the issue of behavioural drift involving polymorphic and metamorphic ransomware, whereby features are assigned weights according to their significance at different times. The approach used mutual information to focus on important features in a more flexible way to cope with new kinds of malware that are being developed. The results showed an outstanding accuracy of 97% and an outstandingly low false positive rate of 0.00880%, which confirms the effectiveness of the model in rapid response. According to **Yu et al. (2024),** dynamic behavioural profiling (DBP) is proposed as a new method of behaviour-based ransomware detection that largely deals with the disadvantages that are inherent to static and heuristic-based methods. In contrast with older methods, DBP frequently tracks the activities of the system to spot unusual characteristics that can have a relation with ransomware, like abnormal patterns of encryption of files, of file creation at high frequency, increased resource usage such as CPU and memory, as well as oddities in the behaviours of processes. DBP uses intrusion detection techniques based on adaptive thresholding, entropy-based approaches, and time series approaches to detect the early stages of ransomware attacks. This novel method provides a high degree of accuracy even in the context of attacks by polymorphic variants of ransomware while having low rates of false positives. In addition, its modular architecture makes it easy to scale and effective in use in different cybersecurity ecosystems.

2.4 Machine Learning Techniques for Ransomware Detection

2.4.1 Introduction to Machine Learning in Cybersecurity

ML may be used in a variety of ways in the field of cybersecurity, where ML is used in many ways in cybersecurity, including phishing detection, intrusion detection, malware detection, anomaly detection, and much more. ML methods have proven useful in a diversity of fields due to their excellent qualities like flexibility, versatility, and the capacity to quickly adapt to new and obscure problems. Different ML solutions have

been effective in addressing a wide-ranging issue in malware security. ML learns the dataset and then predicts malware/non-malware based on its learning because ML in Cyber Security would not only improve our digital security, but it would also allow us to progress toward more efficient innovation

2.4.2 Supervised Learning for Ransomware Detection

ML is an effective technique to detect malware it divides to Supervised Learning Algorithms and Unsupervised Learning Algorithms. In **Azeem et.al (2024)** research they applied Supervised Learning Algorithms (LR, DT, nnMLP, RF, ET, and KNN) on specific dataset chosen with variance features, the experiment depend on select features (10 top, 20 top, random features) by using entropy-based feature selection technique named as TFIDF and then apply the algorithms then compare the result of every experiment. In general, all the ML algorithms used in this research achieved a high accuracy rate, but the best result was the ET classifier utilizing a dropped random features dataset with an accuracy score of 99.98 %. But this research has restrictions, such as the need to use many other datasets because the experiment was done on one dataset.

This is not the only research that uses supervised learning algorithms; there are many others, but they discuss it from different sides, like **Yaseen (2024),** where the author highlights the problem of old classifiers in detecting new emerging ransomware, which could be a high-risk flaw of high risk. It used the top-most supervised machine learning models, which are SVM, decision tree (DT), logistic regression (LG), K-Nearest Neighbors (K-NN), and random forest (RF). The research focus is on the age of the dataset and its role in improving the efficiency of supervised ML algorithms. The author in this paper creates a balanced and mixed dataset of old and new ransomware to effectively train supervised machine learning classifiers because supervised machine learning classifiers trained on the old ransomware dataset are not effective in detecting new ransomware. Moreover, the paper proves that supervised machine learning classifiers trained on the new ransomware dataset alone are not effective in detecting old ransomware. Therefore, the mixed dataset approves the efficiency of this research, and in its training, the classifiers achieved a high accuracy of approximately 97.48% in detecting new or old ransomware.

The ML method not only supervises and unsupervised algorithms; there are more methods based on ML used for ransomware detection, such as using memory artifacts based on ML for ransomware detection. The datasets were created based on memory

features that contained memory samples and selected the best set of features to reduce noise and obtain relevant features for ransomware behaviors, although choosing the right features is a critical step in improving detection. After preparing the dataset, several machine learning models were experimented with, which are LightGBM, XGBoost, Random Forests (RF), Extra Tree (ET), and Adaptive Boosting (AdaBoost), to evaluate how effectively they are at identifying ransomware in order to measure the accuracy and false positive rate. The best result shows that the Random Forest model achieves 97% and 0.04717 FPR with only 16 memory features **Aljabri, 2024)**. Further supporting the relevance of ensemble learning techniques, Itasoy et al. (2024) proposed a hybrid framework combining Isolation Forest for anomaly detection with XGBoost for classification. Their method, which analysed file system activity on Windows, achieved real-time detection with high precision, even under system stress. Similarly, **Woralert et al. (2024)** evaluated a wide array of models, including deep learning techniques, using low-level hardware traces. Their findings showed that LightGBM outperformed CNN and LSTM models in both accuracy (99.97%) and efficiency, reinforcing the value of gradient boosting in practical ransomware defense systems. **Gheisari and Belaton (2023)** conducted a comparative review of supervised ML algorithms and reported that Naive Bayes achieved 96% accuracy, while Random Forest and SVM consistently reached about 99.5%. This further supports the inclusion of Naive Bayes as a lightweight yet effective classifier in real-world detection environments. In conclusion, supervised machine learning techniques offer strong performance in ransomware detection across various domains, from memory features to system behaviour. However, their effectiveness depends on several factors such as feature engineering, dataset composition, model selection, and detection goals. Figure 5 illustrates the accuracy achieved by common supervised models, including Extra Trees, Random Forest, SVM, Decision Tree, KNN, and AdaBoost, when applied to ransomware datasets.

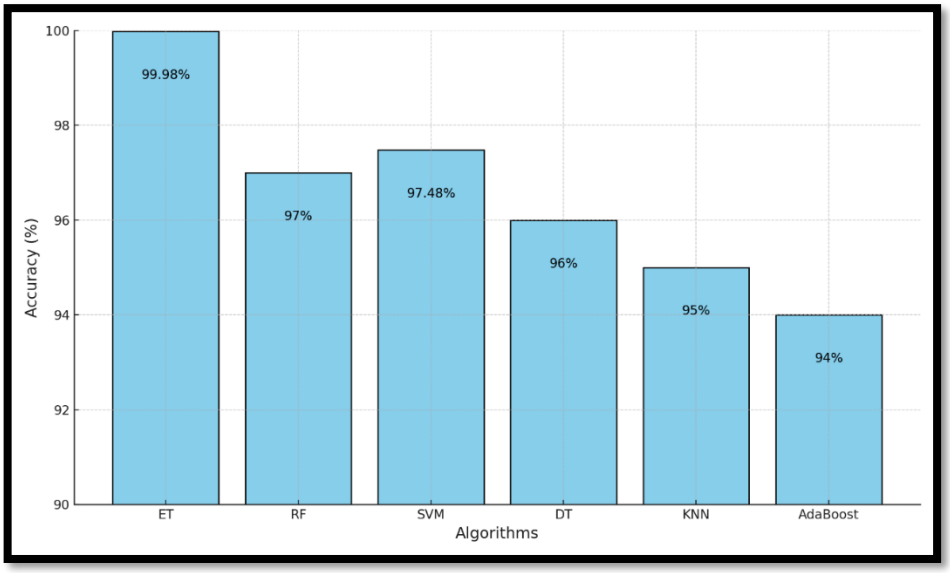

Figure 5.*Accuracy Scores of Supervised Learning Algorithms for Ransomware detection*

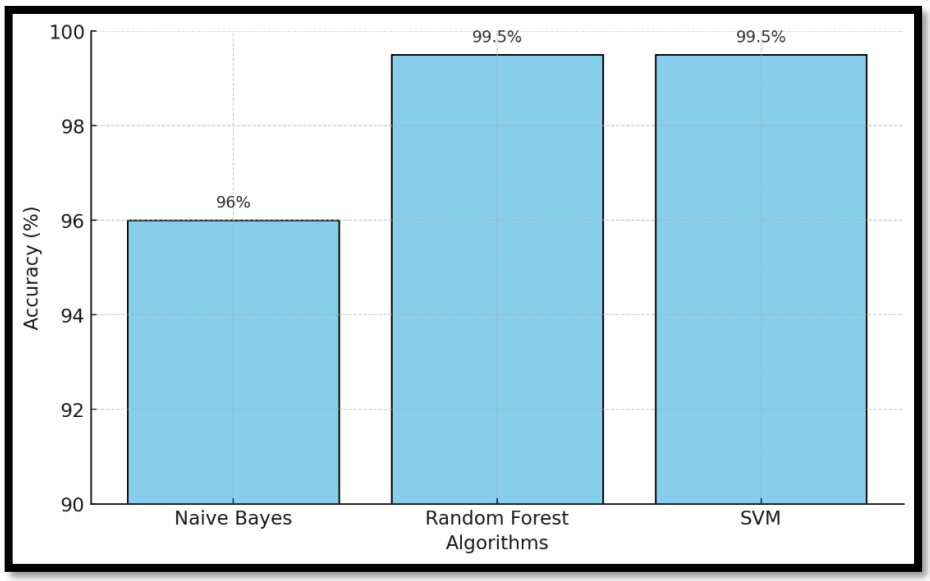

Figure 6*. Accuracy of Supervised ML Algorithms for Ransomware Detection*

2.4.3 Unsupervised Learning and Anomaly Detection

Supervised ML methods are the most used in malware detection, but this does not make the unsupervised ML technique less effective or important. In some research, unsupervised learning was basic for the detection, like **Zahoora et al. (2022).** It focuses on the ransomware zero-day attack, where the traditional machine learning methods might be ineffective, so they introduce a new approach called the Cost-Sensitive Pareto Ensemble Strategy (CSPE-R). This approach uses an unsupervised deep contractive autoencoder (CAE) to transform data into a more uniform feature space. The next thing to do is that different types of base models are trained on these features to understand the key relationships between various families of the ransomware attacks, then a novel Pareto ensemble-based estimator selection strategy is applied to achieve a cost-sensitive compromise between false positives and false negatives, and the final decision combines the outputs of selected estimators to enhance detection capabilities against unknown ransomware attacks. The result of the research was support for using CAE because it reduces the number of small changes in ransomware to find key features. It's better than other methods because it handles noisy data well and focuses on stability. The performance of Pareto was the best of select models, focusing on minimizing false negatives (FN) while balancing other errors, and after comparing the introduced method, it shows that it is effective in detecting ransomware from different families. Finally, while supervised methods are common, unsupervised techniques like CSPE-R and CAE prove highly effective for detecting ransomware, offering high performance and improved error management. Figure 6 presents the comparisons for performance of the Cost-Sensitive Pareto Ensemble Strategy (CSPE-R) with other methods across four key metrics: Accuracy, False Negatives (FN), False Positives (FP), and Stability.

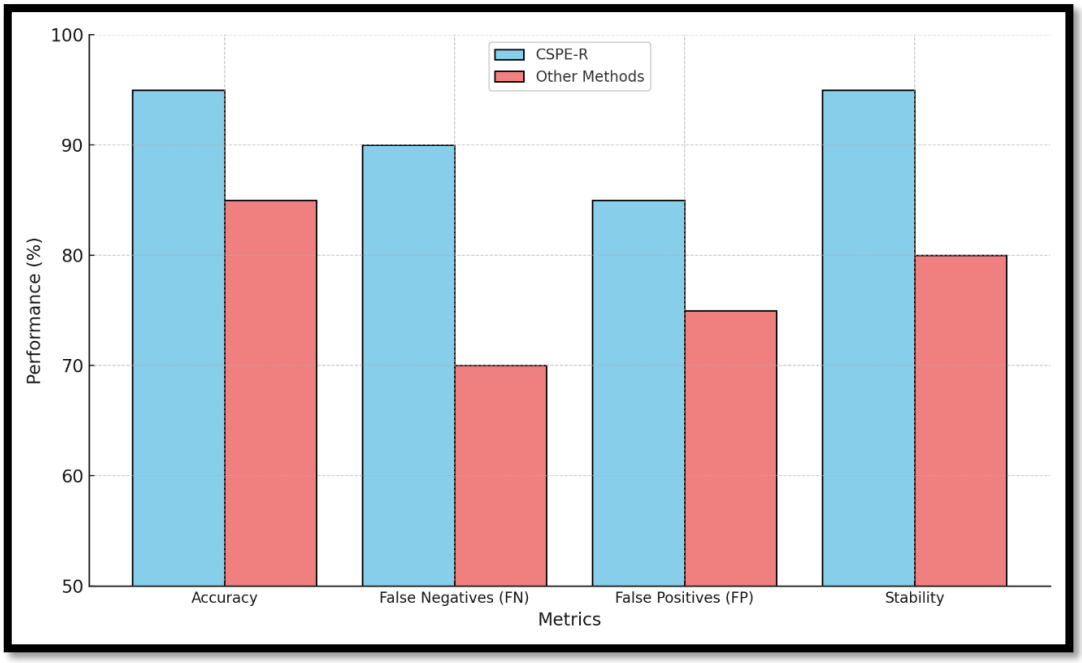

Figure 7. *Performance Comparison: CSPE-R vs Other Methods*

2.4.4 Deep Learning Approaches.

In recent years, deep learning has played a central role in ransomware detection, with LSTM (Long Short-Term Memory) networks emerging as one of the most promising architectures. **Ispahany et al. (2025)** proposed a hybrid model that integrates 1D Convolutional Neural Networks (CNN) with LSTM to analyze Sysmon logs for real-time ransomware detection. Their batch-based incremental learning design enables the model to update continuously without retraining from scratch. The system achieved an F2-score of 99.61%, with extremely low false positives (0.17%) and false negatives (4.69%), demonstrating the effectiveness of LSTM–CNN hybrids for adaptive detection in highly imbalanced environments. In other research, **Kiyol et al. (2024)** introduced an LSTM-based detection framework that incorporates file entropy metrics to capture behavioral anomalies during ransomware execution phases. By modeling the temporal dependencies of file entropy variations, the model accurately identified ransomware attacks in real time across multiple families. Compared to traditional classifiers like SVM and Random Forest, the LSTM–entropy model achieved superior accuracy, precision, and recall—particularly against stealthy or polymorphic variants—demonstrating the strength of LSTM in sequence-based detection tasks.

Additionally, **Gazzan and Sheldon (2024)** explored the use of Deep Belief Networks (DBNs) in ransomware detection, focusing on improving model training through a novel

Uncertainty-Aware Dynamic Early Stopping (UA-DES) technique. This approach uses Bayesian methods and dropout regularization to boost DBN learning efficiency. Their model, UA-DES-DBN, outperformed other deep learning architectures such as VGG16-PSO and DBN-IDS, achieving a 6% accuracy gain over DBN-IDS and 4% **over** VGG16-PSO, while also reducing the false positive rate from 0.18 to 0.10. This work highlights the role of uncertainty estimation and model calibration in enhancing the reliability of deep learning models for cybersecurity applications.

2.5 Challenges and Limitations

Previous studies on ransomware detection have reported several recurring challenges that hinder the advancement of effective detection frameworks. These include the scarcity of comprehensive and balanced datasets, which limits model generalization across ransomware families, and the difficulty of achieving real-time detection under dynamic and high-load environments. Additionally, many existing works rely heavily on static or single-layer features, restricting their adaptability across different system domains and ransomware variants. Similar issues have been emphasized by **Begovic et al. (2023)**, and **Patel and Yashaswini (2024)**, who noted that dataset imbalance, feature redundancy, and inconsistent evaluation metrics remain key obstacles to developing robust detection systems.

While this chapter has provided an overview of these limitations from a literature perspective, a more detailed discussion of how such challenges were encountered and addressed within this study's experimental framework will be presented in Chapter 5.

2.6 Recent Advances and Future Directions

Recent developments in ransomware detection show the increasing use of new technologies like blockchain and AI. These technologies seek to overcome the weakness of conventional methods by improving the efficiency and precision of ransomware detection in systems. The suspicious activities can be kept away using AI, like behavior-based and specifically deep learning techniques, which can identify new ransomware strains by analyzing various datasets and present new detection models. Methods such as machine learning offer potential by enabling models to detect ransomware imitating harmless actions, thus stopping evasion. The use of real-time monitoring, synthetic

datasets, and intelligent agents is being expanded to improve ransomware detection systems with emerging technologies. Future research includes detecting ransomware and involves creating more advanced techniques to keep up with complex evasion methods like encryption and fileless attacks, as well as improving the quality and availability of data for training models. It is necessary to enhance algorithms for both speed and accuracy to improve real-time detection capabilities, as well as to integrate methods that merge behavior-based, machine learning, and deep learning techniques. Moreover, examining the effects of Ransomware-as-a-Service (RaaS) and the new developing adaptable, self-teaching systems that keep adjusting to new threats, which enhances the efficiency of ransomware detection and prevention.

- **Comparative Analysis and Research Gap Alignment**

While numerous studies have explored behaviour-based and machine–learning–based ransomware detection, most remain limited to a single data domain or detection layer. For instance, **Kwon et al. (2022)** and **Agate et al. (2022)** focused primarily on network-based IDS frameworks, achieving notable accuracy but lacking integration across host and network behaviors. Similarly, **Yu et al. (2024)** and **Wang et al. (2021)** concentrated on specific platforms such as industrial control systems or Android environments, without validating scalability across heterogeneous datasets.

In contrast, the present research adopts a hybrid experimental design that combines File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA) within a unified dataset. This integration addresses the data granularity limitation identified in prior studies and enables a multi-perspective understanding of ransomware activity. Moreover, unlike prior approaches that rely solely on static or single-phase evaluation, this study emphasizes feature fusion and decision/model-level integration, enabling comparative assessment of both behavior-based and machine learning (ML) techniques under consistent experimental conditions.

Consequently, this research bridges the methodological fragmentation observed in earlier work by presenting a unified evaluation pipeline—from dataset preparation through feature fusion and sensitivity analysis—that systematically examines ransomware detection performance across multiple modeling paradigms.

2.7 conclusion

This chapter discusses the development and presents strategies for identifying malware and ransomware, focusing on the progress and constraints of behavior-based and machine learning approaches. Despite advancements, there are still major obstacles to address, such as evasion strategies, problems with data accuracy, and the requirement for scalable, real-time detection options. Continued research is essential to enhance detection methods and combat the challenges that appear because of the evolution of ransomware and other modern cyber threats. This provides a foundation for further examination and enhancement of ransomware detection methods in the following chapters.

Chapter 3: Methodology

Behavior-Based Detection

3.1 Introduction

Behaviour-based detection techniques is an effective approach for detecting ransomware, especially with the continuous development of threats. Unlike signature-based methods, which is based on static patterns or known malware identifiers, behaviour-based detection focuses on dynamic system activity and behaviour patterns. This allows for identifying new or modified ransomware variants, even those designed to avoid traditional detection systems. The importance of this technology lies in its ability to adapt, because ransomware programs exhibit distinctive and changing behaviours during their execution, such as interpreting files or making changes to the system and making unauthorized network permissions. These technologies monitor the behaviour of viruses in real-time, which helps in detecting them before they cause a lot of damage. However, the use of behavioural analysis techniques is not without challenges. Ransomware often blends in with normal systems and activities, making detection difficult, especially since, in most cases, it does not attack immediately, but rather delays its activities so that it is not detected immediately. In addition, real-time detection of malware using these technologies requires Great resources that many organizations lack. False positives are also considered a challenge for ransomware detection technologies, which leads to unnecessary disruptions. This chapter will cover the implementation and evaluation of behaviour-based ransomware detection techniques. The datasets, tools and methodologies used will be described, and the effectiveness of these methods will be evaluated. By addressing the opportunities and challenges of behavioural analysis-based detection, this chapter aims to contribute to the development of detection systems capable of reducing modern ransomware threats.

As illustrated in Figure 8, the proposed behaviour-based detection framework follows a structured workflow encompassing dataset collection, preparation, feature engineering, technique implementation (FSM, PBA, and NBA), dataset integration, and evaluation.

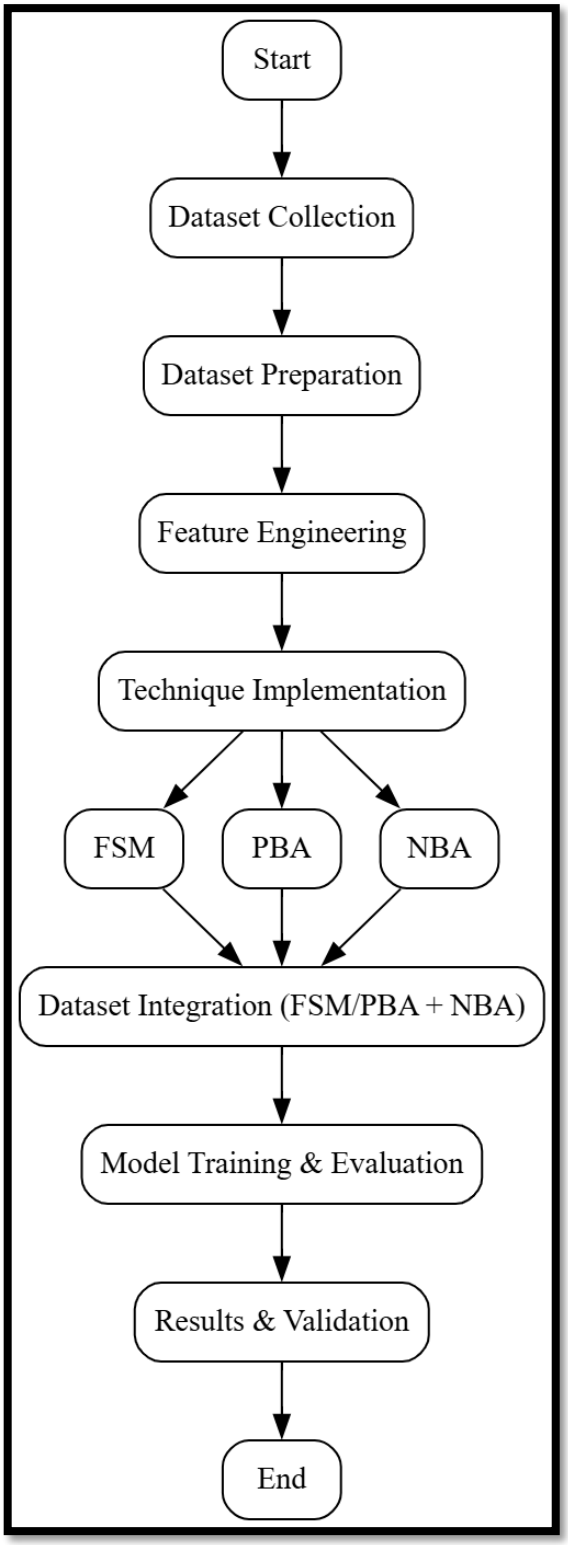

Figure 8.*Overall Flow of the Proposed Behaviour-Based Detection Framework*

3.2 Research Design

In this section, we will focus on evaluating and improving behaviour-based detection methods for ransomware by applying these methods to a dataset, modifying dataset

features, and discussing their impacts on detection accuracy. The aim is to determine the most effective behaviour-based techniques for ransomware detection and improve detection performance.

Behaviour-base have seven methods ca be used for ransomware detection. In this research, we chose four methods to apply (File System Monitoring, Process Behaviour Analysis, Network Behaviour Analysis, Behaviour-Based Anomaly Detection).

3.2.1 Approach

The chapter focuses on evaluating and implementing behaviour-based detection techniques to identify ransomware activities effectively. These techniques (file System Monitoring, Process Behaviour Analysis, Network behaviour Analysis) These technologies were chosen for their effectiveness in detecting advanced and complex ransomware attacks.

- **File System Monitoring (FSM)**

This technique works by monitoring the unusual activity in the file system, focusing on encryption, creation, deletion, and any other unexplained strange activity. Normally, there is a specific pattern for ransomware, such as deleting backup copies in order to prevent recovery, or encrypting large mount od files in a short time. This method aims for the detection of ransomware by analyzing unexpected behaviours mentioned above. File monitoring systems implementation needs to be designed to minimize performance overhead, which is very important to improve system efficiency. The file monitoring technique needs to be lightweight and efficient.

- **Process behaviour Analysis (PBA)**

This technique studies the behaviour of system processes like resource usage, termination, creation, and interactions with other processes. Ransomware processes usually appear Unique Features like creating multiple subprocesses, modifying files on a large scale, and executing commands to disable security tools.

Processes are the execution layer of ransomware attacks. Modern ransomware has Strategies to avoid the traditional detection methods, but by using PBA, we can detect ransomware techniques like code obfuscation or encryption to avoid signature-based detection.

- **Network behaviour Analysis (NBA)**

It focuses on the unusual connection, data exfiltration, or communication with command-and-control (C&C) by analyzing the network traffic. This unusual connection could be a strong reason to detect ransomware because it communicates with external servers to receive encryption keys or to exfiltrate sensitive data. Most of times network activities are the beginning point in ransomware execution, and that why working on NBA technique very important it Focus on Network Communication which is happening in almost every device now, unlike techniques such as File System Monitoring or Process behaviour Analysis that focus on endpoint-level activities, NBA provides a broader perspective by monitoring Internal Traffic and External Communications and this makes NBA valuable in detecting ransomware that targets network-level vulnerabilities. In addition, the NBA gives the best result when it is integrated with other techniques like File System Monitoring or Process behaviour Analysis. For example, it can detect ransomware communications while FSM flags local encryption

These techniques were chosen to work on because they target different aspects of ransomware behaviour, which helps identify malicious activities at different stages of the attack. But each technique has limitation make the advanced ransomware can overcome each technique individually. So, we will be working to overcome the limitation such as false positives or evasion, by combining these techniques.

Table 2.*Comparison of Behaviour-Based Techniques*

Technique	Key Focus	Strength	Limitation
File System Monitoring	File creation, modification, and entropy	Detects local encryption and file deletion.	Cannot detect external communication.
Process Behaviour Analysis	Resource usage, process creation	Detects malicious process behaviours.	Limited to endpoint-level activities.
Network Behaviour Analysis	Network traffic patterns	Detects external and lateral movements.	Requires robust network infrastructure.

multi-layered approach

Each behavioural monitoring technique has its own advantages and ability to detect ransomware, but each also has limitations that limit its detection ability, especially with the development of ransomware that we talked about earlier.

File System Monitoring (FSM) is effective at detecting ransomware activities like file encryption or deletion, but it can't justify the reason for these actions. For example, it can distinguish high-entropy files, but it cannot distinguish whether this activity is due to the files containing ransomware or regular files, and if FSM detects regular files as ransomware, it is called False Positives. The integration of FSM with PBA allows for associating the flagged files with processes working with them to investigate further the underlying issues. In addition, the inclusion of the NBA overcomes the limitations of FSM in identifying ransomware that is predominantly network-based, such as data exfiltration or liaising with command-and-control centres.

Table 3. *Normal vs. High-Entropy Files*

Type of file	Typical entropy value	Example
Plain Text File	Low (2–4)	Text documents (e.g., .txt)
Partially Random File	Medium (5–6)	Unstructured files (e.g., .csv)
Encrypted/Compressed File	High (7–8)	Encrypted files (e.g., ransomware outputs, .zip)

Process Behaviour Analysis is capable of monitoring system processes for initiating or respawning sub-processes, including the usage of system resources beyond limits. However, it lacks the knowledge of which files are opened by the processes, and if the processes are making attempts to communicate with remote servers. For example, a process that gets flagged could be undertaking Unacceptable operations that require too much time and resources unless they are confirmed by FSM (for example, checking the encryption of still important files' locations) and NBA (for example, checking if there are any unauthorized connections to the network). The combination can guarantee that only processes that are truly malicious in nature will create flags, improving the level of false positives.

Addressing the Constraints of the NBA: Even though Network Behaviour Analysis is effective in determining ransomware threats as it connects to its external server, it falls

short in dealing with local ransoms that are persistent in performing encryption locally without resorting to any network interface. By employing both the NBA and the FSM, such ransomware that is defined as local can be detected through file operations that are unusual in their context. This is in addition to the fact that the network irregularities that are registrable through the NBA are subject to PBA in order to ascertain where the network origin comes from, a suspicious process that improves the level of detection accuracy.

The most comprehensive and, therefore, detailed scope of work on the detection of ransomware behaviors is achieved through the combination of FSM with PBA and NBA. This integration means that ransomware is restricted to more than one domain of conduct to avoid detection. For instance:

- FSM detects the patterns of encryption, PBA assigns these to the particular processes that performed them, while NBA checks whether the process has any communications with the other servers.

- Even if ransomware tries to avoid contact with external hosts, and depending on the structure of the NBA, it can be complemented by FSM and PBA, which will always look into the local files and processes of the ransomware.

- If ransomware uses its ‘process behavior’ control so that its processes can appear normal, mechanisms like FSM and even the NBA can be used to detect its malicious files or activities outside of networks.

This multi-technique approach greatly enhances the probability of false negatives being reduced and increases the verification of different data to cut down the chances of false positives. In the end, the integration of these techniques provides a strong detection mechanism that can withstand the complex strategies employed by the new generation of ransomware. In this research, we integrated the FSM/PBA dataset with the NBA dataset for more in-depth analysis of the ransomware activities. The FSM/PBA dataset includes file-related activities, process behaviors, and file access patterns, which help to understand the behavior of ransomware within the system. Whereas the NBA dataset includes activities at the network level, for instance, packet arrival intervals, traffic

volume, and protocol use, which unveil the communication and networking strategies of the ransomware.

Ransomware can be characterized as comprising various stages, such as file encryption, process execution, and communication with command and control (C2) servers. The combination of the system perspective with the network perspective makes it possible to carry out a more complete analysis. This helps us find the abnormalities in the analysis in more detail and relate interdependencies between different abnormal signals across different datasets. For instance, the unusual use of a system call may be due to the explosive increase in network traffic, which would mean that the victim is under a ransomware attack. The incorporation of these datasets makes our action-based ransomware detection techniques more effective and efficient.

3.2.2 Dataset

A dataset is constructed within this research, providing the necessary raw data, which is fundamental for the implementation and assessment of the effectiveness of File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA). It incorporates threads of logs and metrics regarding files, processes, and networks, thus providing the means to exhaustively evaluate ransomware detection mechanisms.

3.2.2.1 Dataset Source

The first dataset, named RansomSet, was created in 2023 using Cuckoo Sandbox to analyse the behaviour of six ransomware families: WannaCry, Ryuk, CryptoLocker, Conti, Sodinokibi, and LockBit. It includes both normal samples and ransomware samples, structured for behaviour-based detection techniques. It has dynamic features, including system calls (APIs), timestamps, and process behaviours, making it suitable for File System Monitoring **(**FSM**)** and Process Behaviour Analysis **(**PBA**)**.

As mentioned before, the data collect created using Cuckoo Sandbox, and the original data was stored in JSON files, capturing the execution details of each binary. But by python script, the JSON files are processed to extract key features and convert them into a structured CSV format.

The second dataset was extracted by using Wireshark. We chose one of the Sodinokibi samples from a repository of PCAP files obtained by executing ransomware binaries and capturing the network traffic created when encrypting a set of files shared from an SMB server. In the repository, there are 94 samples from 32 different ransomware families, but we chose WannaCry because it is one of the six ransomwares existing in the first dataset in order to combine the two datasets. To make it easier to distinguish between them, we will name the second dataset the NBA dataset.

These are two different datasets, but we intend to combine them in order to apply the base-behaviour detection techniques mentioned before, as the three detection techniques in this research need different types of data, and it is rare to find all these data in one dataset, so we are going to combine these datasets. The combination of various data sources can help to improve the accuracy of detection by casting the detection over several behavioural dimensions. One such structure is SUNDEW, as it uses QMonitoring systems, OSs, and even hardware performance counters to aid in gathering a holistic understanding of the activities performed by the malware. With this form of composite integration, evidence that is likely to be ignored in a single form mastery will be brought to light. We also take this stance in our work, where we combine the FSM/PBA, which contains system and file behaviour information, and network behavioural information, which is contained in the NBA dataset. In this way, we can consider ransomware behaviour from more angles at once and strengthen the reliability of the detection method based on behaviour. The success of this dataset integration approach is evidenced by the implementation of the SUNDEW framework (**Karapoola et al., 2022**).

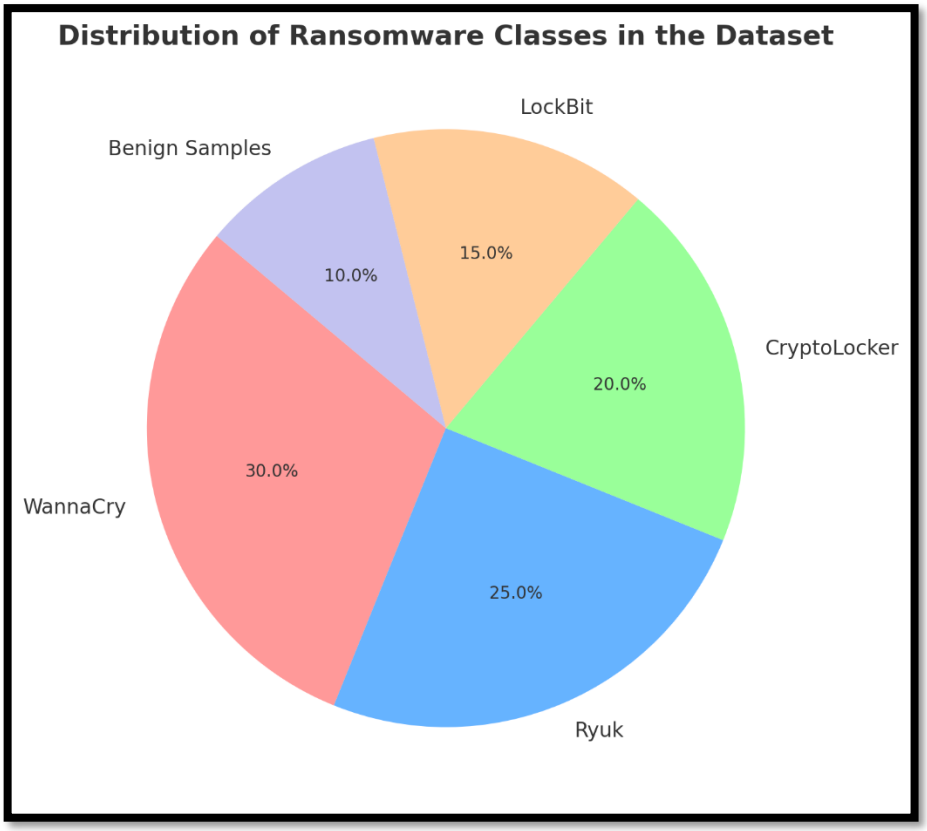

Figure 9. *Distribution of Ransomware Classes in the Dataset*

The behavioral data used in this study for FSM and PBA analysis was derived from the publicly available **Ransom Set** dataset (**Oliveira, 2025**), which provides structured JSON traces of ransomware activities captured through Cuckoo Sandbox. The dataset covers six major ransomware families—WannaCry, Ryuk, CryptoLocker, Conti, Sodinokibi, and LockBit—and includes normal (benign) samples. It was obtained from the official GitHub repository: https://github.com/gabrielolivs/RansomSet.

The network-based behavioral dataset (NBA) used in this study was obtained from the public repository hosted by the Universidad Pública de Navarra (**Berrueta et al., 2020**). This dataset includes PCAP captures of ransomware behavior across multiple families and enables feature extraction such as packet interarrival times, session durations, traffic volume, and protocol types. The original repository is accessible via: http://dataset.tlm.unavarra.es/ransomware/.

3.2.2.2 Dataset Description

The RansomSet dataset consists of 19234 rows and 240 columns of process calls, process actions, and other micro-level details that are pertinent to six categories of ransomwares, which include WannaCry, Ryuk, CryptoLocker, Conti, Sodinokibi, and LockBit. It was developed through Cuckoo, which observes and captures different types of interactions in a controlled setting. The main focus is to enable these two types of detection methods: File System Monitoring (FSM) and Process Behaviour Analysis (PBA). Each case in the dataset has been characterized as a specific interaction or system behaviour performed in the course of the ransomware execution. For instance:

- NtAllocateVirtualMemory: The Algorithm allocates virtual memory to track the number of attachments that have been requested during the processes that are commonly undertaken by most ransomwares, e.g. Encryption.
- NtCreateFile: The number of attempts to create a file for encryption is recorded. This is useful as ransomware can create a malicious script or file.
- score_binary: The measure score is a fixed number, and the score is usually recorded in an arbitrary number; this registered score, which is the one in question, is normally indicative of the risk degree that is usually attached to the actions taken.

The Network Behavior Analysis (NBA) dataset has captured behavioural patterns relevant to ransomware activities, specifically through capturing large volumes of its network traffic. This dataset includes packet captures, which enrich the understanding of communication processes that might lead to potential ransomware attacks. This information includes traffic event forms that are timestamped, allowing events to be ordered, and other relevant attributes of network traffic, such as origin and destination, type of communication, volume, and situational variables. Special emphasis should be placed on the dataset in prospect, as it primarily focuses on the intricacies of protocols, which are imperative in the detection of network irregularities and the possibility of unauthorized activities. Thus, by allowing the analysis of network behaviours, the NBA dataset performs a crucial role in augmenting the behavioural analysis of ransomware, as demonstrated by file system and process-level data. Due to being able to offer communication insights into patterns, such datasets hold a key role in this research.

The network behavior dataset (NBA) used in this study was manually constructed from a single ransomware sample—**Sodinokibi**—whose execution was captured in a PCAP file (Sodinokibi_04052021.pcap). The traffic was analyzed using Wireshark and Zeek to extract key network features. The resulting data was segmented by protocol into structured CSV files, including: dns_traffic.csv, http_traffic.csv, ip_tcp_traffic.csv, and smb_traffic.csv. Each file contains flow-level statistics such as DNS query types, HTTP request patterns, TCP session sizes, and SMB communication activity. These extracted features were later used to simulate or enrich the network behavior portion of the unified dataset. Due to the limitation of having only one PCAP sample, this dataset represents a partial but practical view of ransomware-related network behavior.

3.2.2.3 Dataset Structure

The dataset consists of important elements that capture the behaviour of ransomware as a whole. One of these metrics is the system call metrics, which indicate the amount of certain API calls made, for instance, NtAllocateVirtualMemory, NtCreateFile, and NtReadFile, so as to chart the i interactions of the ransom software with the system resources when it is executed. These metrics serve as low-level indicators of operations like memory allocation, file creation, or reading, all of which are critical to ransomware functionality. Behavioural features go beyond individual calls, enabling the understanding of interactions of ransomware with system resources such as file modifications, registry keys, or the management of processes. Other than those above, the dataset comprises a score_binary column that simplifies the severity of each activity according to predetermined parameters, allowing a rapid risk evaluation. Finally, the class column assigns the data to particular classes of ransomware, such as Sodinokibi or WannaCry, which makes it possible for comparison of behaviors among the various strains of ransomware in relation to the detection model being developed.

- **System calls metrics**: It contains the number or frequency of some particular API calls like NtAllocateVirtualMemory, NtCreateFile, and NtReadFile.
- **Behavioral features:** Metrics for the relationship between the ransomware and system resources, resources like files and registries.
- **Scores**: The column score_binary is a relative grade for an activity, designed to be easy to comprehend.

- **Class Labels**: The class column labels the data into ransomware types, such as Sodinokibi.

Through the study of the dataset, we find that many features make it enhance for FSM and PBA, and it's sorted into two groups.

File System Monitoring (FSM):

- File-based API calls: Contains data modification such as NtCreateFile, NtReadFile, and NtClose that record the interactions with the files.
- File attribute manipulations: Such columns as FindFirstFileExW and GetFileAttributesW record the modes of operations that an arbitrary file undergoes, thus witnessing the file encryption by the ransomware.

Process Behavior Analysis (PBA):

- Memory operations: Via features like NtAllocateVirtualMemory and NtFreeVirtualMemory, instances of strange memory usage are captured.
- Registry operations: It comprises the Windows registry actions that RegDeleteKeyA is among many that depend on Ransomware.
- Thread and process control: Some metric like SetWindowsHookExA is useful in understanding how the ransomware self-injects into processes or how it hooks procedure calls.

Table 4 describes all features in the dataset for which the technique can be applied.

Table 4. *Feature disruption*

Feature	Description	Technique Used
score_binary	Severity score of the activity	both FSM and PBA
PRF	Process-related feature frequency	PBA
NtAllocateVirtualMemory	Count of virtual memory allocations	PBA
NtCreateFile	Number of files created	FSM
NtReadFile	Number of files read	FSM
NtClose	File closure count	FSM
NtFreeVirtualMemory	Count of memory deallocation operations	PBA
FindFirstFileExW	Frequency of accessing the first file in the directories	FSM

GetTempPathW	Number of temporary file paths accessed	FSM
GetFileAttributesW	Frequency of file attribute checks	FSM
CreateDirectoryW	Frequency of retrieving Windows directory paths	FSM
NtQueryDirectoryFile	Count of operations retrieving user details	PBA
GetFileSize	Registry key deletion attempts	PBA
SetFilePointer	Timeout message box calls	PBA
NtWriteFile	Hook installation on Windows events	PBA
NtOpenFile	Console write operations	PBA
NtOpenKey	Decryption cryptographic operations	PBA
NtQueryValueKey	Key generation operations	PBA
LdrLoadDll	Encryption cryptographic operations	PBA
LdrGetProcedureAddress	Ransomware classification label	Classification Label

The NBA dataset contains packets and associated metadata, with each row corresponding to one network packet or its attributes. The main features include a normalized timestamp, which serves to provide ordering in sequential analyses of packets, and fields like source and destination addresses, type of network protocol and size of packet, which broadly define the packet movements in the network. The dataset supports different transmission lines, in particular legitimate and hostile ones, operating on different protocols. Such variation increases its usefulness in investigations of the behaviours of the ransomware, especially in the areas of understanding the deviations from the normal network activities. The additional reference about every packet also helps to pinpoint some unique events that could be associated with the actions of ransomware. As the dataset has more than 19000 records, the sample size is adequate for such an analysis. Since data on the structure of the traffic is known, both features of the set are suitable for the task of detecting ransomware and network behaviour analysis, respectively.

3.2.2.4 Dataset Preparation

Preparing the datasets before merging was a critical step to ensure the quality and consistency of the combined data. The FSM/PBA dataset and the NBA dataset have different structures and types of information—one focuses on system calls and file

activities, while the other focuses on network packets. To merge these datasets effectively, we needed to standardize and clean each dataset. In the FSM/PBA dataset, we focused on extracting relevant features like file access patterns, process behaviours, and system calls. For the NBA dataset, we calculated features like packet interarrival time, protocol categories, traffic volume, and session duration. Each dataset was cleaned by removing duplicates, handling missing values, and ensuring data integrity. Preparing the datasets before merging ensured that timestamps were aligned, data types were consistent, and the datasets were ready for immediate use. This preparation step preserved the completeness of the data and prevented issues during merging, such as conflicting columns or inconsistent data formats. By preparing the datasets thoroughly, we ensured that the merged dataset retained all the critical behavioural information necessary for effective ransomware detection.

During dataset preparation, all CSV files generated from the JSON conversion were validated to ensure the absence of missing values or duplicate records. The dataset was scanned for null or NaN entries in critical features such as system call frequencies, timestamps, and class labels. Samples with incomplete or corrupted values were either removed or corrected based on redundancy within their respective class groups. Furthermore, duplicate entries resulting from multi-process recordings of the same binary were filtered out to prevent data leakage and overfitting during training. These cleaning steps ensured a reliable foundation for both behavior-based analysis and machine learning evaluation.

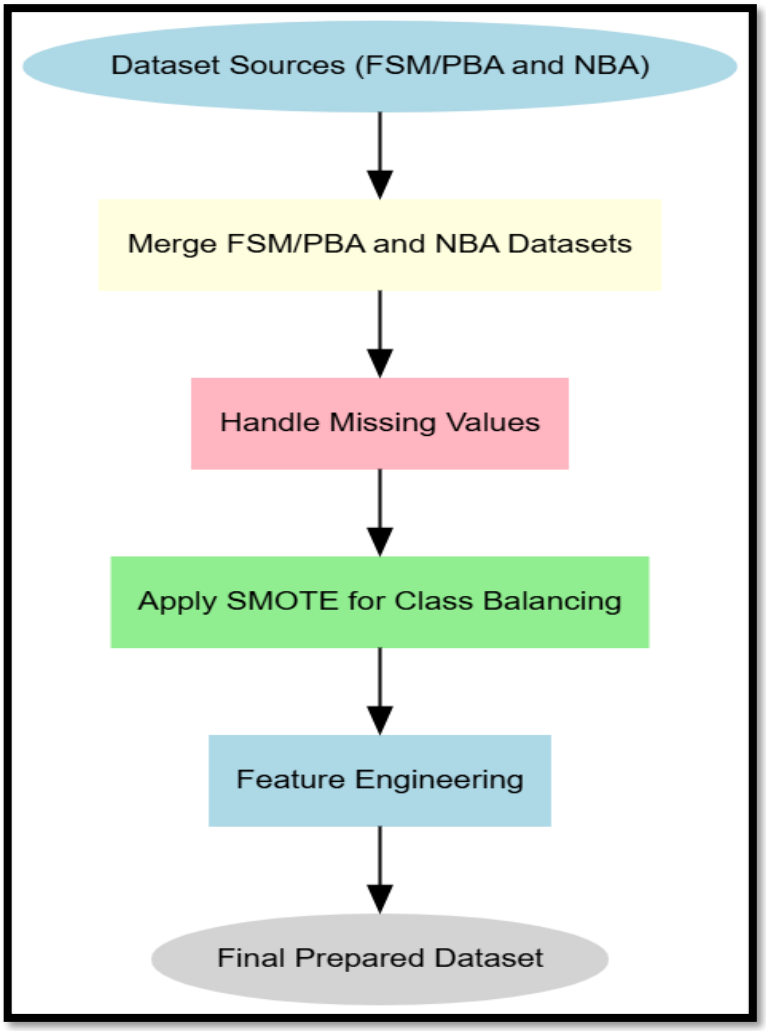

Figure 10. *Dataset Preparation for base- behaviour Flowchart*

Given the high dimensionality of system-level behavioral data, feature aggregation was applied during dataset preparation to reduce redundancy and improve representational efficiency. Many low-level system call features exhibit correlated behavior and collectively represent a single behavioral concept. Aggregating such features allows the model to capture higher-level behavioral patterns while minimizing noise and overfitting risks before dataset merging.

3.2.2.4.1 RansomSet (PBA/FSM) dataset:

1- Understanding Relevant Features

The feature groups on which behaviour-based methods such as PBA and FSM are founded include:

• File System Monitoring (FSM): It includes features such as file function calls including file creations (NtCreateFile), reading files (NtReadFile), and closing files (NtClose). It also includes temporary file directory (GetTempPathW) and file characteristics information (GetFileAttributesW).

• Process Behavior Analysis (PBA): It includes features such as memory allocation (NtAllocateVirtualMemory and NtFreeVirtualMemory), registry (RegDeleteKeyA) and other operations such as Cryptography (CryptEncrypt, CryptGenKey).

2- Data Cleaning

The very first stage that has to be done, which is important in any dataset employed for behavioural-based ransomware detection coupled with file system monitoring (FSM) and process behavior analysis (PBA) techniques, is data cleaning. For this dataset, missing values were not detected in any of the 242 features, which means that the dataset is complete. This is critical in the processes of detection since failure to include these variables in those features, should there be some missing, such as operational features like NtCreateFile, NtReadFile, or LalocateVirtualMemory or RegDeleteKeyA, affects the reliability of the PBA and FSM approaches. Further, the items describing repetitions or those attributes with no significant relevance to the detection of ransomware, a column containing some metadata that does not add value to this detection process should be removed. Elimination of irrelevant features will improve the feature engineering, ensuring the dataset used comprises only quality features, improving the performance of machine learning models in identifying ransomware behavioral patterns.

3- Feature Engineering for FSM and PBA

Feature engineering is one of the most essential phases while working with datasets for behaviour-based techniques like File System Monitoring (FSM) and Process Behaviour Analysis (PBA). With WEKA (Waikato Environment for Knowledge Analysis), this is easy and smooth because it allows the automated extraction and transformation of certain important features to improve the appearance of the ransomware behaviors. Henceforth, we discuss how WEKA can be used to engineer the FSM- and PBA-specific features for optimum results.

FSM is oriented towards locating ransomware attacks through the activity of file operations like creating, reading, or modifying files. Filesystem operations are heavily dependent on the system calls, i.e., (NtCreateFile), (NtReadFile), and (NtClose), which are respectively correlated with creating, accessing and closing files. Using WEKA, the following steps can enhance FSM features:

File operation-related system calls were aggregated into composite features (e.g., Total_File_Ops) to capture the overall intensity of file system interaction rather than relying on isolated function calls. *This design choice is justified by the fact that ransomware typically performs repetitive and high-frequency file access operations during encryption, making cumulative file behavior more representative of malicious activity than individual system call occurrences.*

- **Aggregating File Operations:** The (AddExpression) filter in WEKA allows the creation of a new feature, such as Total File Ops, by summing up the values of individual file operation metrics: Total File Ops=NtCreateFile + NtReadFile + NtClose. This aggregate metric encapsulates overall file interaction intensity, which is often a strong indicator of ransomware infecting a machine.

- **Converting particular characteristics into a binary scale**: By means of the filter (NumericToBinary), attributes like (GetFileAttributesW) can be expressed as a single binary digit in order to indicate whether a particular file operation is performed or not. For example, a (1) could mean that there are regular changes to file attributes, which is commonplace in attempts to encrypt files by ransomware.
- **Changing the GetFileAttributesW** function to a binary presents the data in a way that only considers whether a file attribute change occurred or not (1 or 0). This method is useful because it places emphasis on ransomware acts because ransomware changes the attributes of the files during the encryption process. Converting to binary reduces the order of the model, and therefore, the extent of computations required for machine learning and recognition of abnormal patterns is more efficient. It also makes the features more understandable since binary ones are easy to represent and comprehend. In addition, consistent with these approaches, this one also guarantees coherence with other features of the FSM

that may as well be in binary, which should increase the model's opportunities to learn patterns and thus increase the accuracy of detection.

The analysis was simplified, and the effectiveness of behaviour-based ransomware detection improved by discretizing the frequencies of certain file operations, in this case, the FindFirstFileExW operation. By habitually "ransomware-ifying" the continuous numeric values into discrete ordinal schemas such as Low, Medium, and High, identifying trends and deviations within the ransomware behavior is made much simpler. Such encoding lowers the dimensionality of the dataset and therefore improves the performance of machine learning applications in recognizing and understanding different levels of the same file activity. Discretization further assists the models by reducing the likelihood of the presence of outliers and the bias towards wide distributions, which would otherwise make the models overly sensitive to large values. All these procedures, at the end, make the analysis easier and more efficient by improving the explanatory power of the selective file operation feature, which proved to be effective in telling apart benign and dubious files.

The PBA features include tracking of processes and memory attacks since this aspect is usually utilized by ransomware in the course of execution. Important functions include memory allocation (NtAllocateVirtualMemory), de-allocation (NtFreeVirtualMemory), and cryptographic functions (CryptEncrypt, CryptGenKey). WEKA indeed expedites the effecting of these features through the following approaches:

Memory-related system calls were aggregated into a unified Memory_Behavior feature to represent the overall memory usage pattern of a process. This decision was motivated by the observation that ransomware exhibits distinctive and repetitive memory allocation and deallocation cycles during execution, where the aggregated pattern provides a clearer behavioral signature than individual memory-related system calls.

- Before combining the definitions of NtAllocateVirtualMemory and NtFreeVirtualMemory into one feature, Memory_Behavior, we first gathered all the memory operations definitions. This enabled us to simplify the dataset and, therefore, improve the detection of some behavioral patterns of ransomware. Indeed, ransomware demonstrates specific patterns of memory allocation and its deallocation throughout its intended purpose, for example, encrypting files,

changing system processes, and many others. With the aggregation of these operations, we extract the general memory pattern of a process, which assists in identifying memory aggressors such as ransomware much more easily. This simplification also helps in overfitting because the dataset has been reduced, making it easier to identify the components that matter. By grouping these memory operations, automated systems become more impressive and more accurate while still capturing the most essential characteristics of the behavior of the processes in question.

Cryptographic operations were consolidated into a single Cryptographic_Behavior feature, as the presence and frequency of encryption-related activities are more behaviorally significant for ransomware detection than individual cryptographic function calls. This aggregation reduces feature redundancy while preserving the essential cryptographic behavior that characterizes ransomware execution.

- **Cryptographic Behavior Engineering**: Merging cryptographic operations features like CryptEncrypt and CryptGenKey into a single feature, the Cryptographic_Behavior feature, enhances the dataset and makes the detection of ransomware easier and much more effective. This is because ransomware exploits a variety of cryptographic operations, such as encrypting and generating keys during its runtime. With these changes, it becomes much simpler to identify abnormal activity related to ransomware infections due to the way these consolidated operations are able to depict the holistic pattern of cryptographic activity. This also eliminates redundancy and the features, thus improving the working of machine learning models by allowing them to search for relevant behavior patterns only. The negative impact, however, is minimal because the most important thing is that the relevant information necessary to recognize behavioural patterns associated with such infections is preserved.

- **Flagging Registry Changes**: Deleting registry keys is altered into a feature, whereby a binary flag (Key_HKey, Target_Location: -1) is set to 1 if a registry modification occurs and set to 0 if no registry modification is made. As a part of the analysis of how a computer in case of being infected by the ransomware,

modifications of the registry are particular behaviors that are carried out by ransomware. The general aim, though, is to delete or modify certain registry keys believed to wipe out security features, enhance continuity, or prevent restoration of the system. These alterations are flagged to also aid in the detection of anomalous activities and pattern recognition by machine learning models as it relates to ransomware. Simple and straightforward this way helps in improving the effectiveness of models as the only thing that matters is whether there are changes in the registry hives that are most likely to be changed for that activity to be labelled as ransomware activity or not.

4- Normalization

As seen in behaviour-based techniques such as FSM and PBA, normalization is the process of scaling the relevant feature values so as to allow comparison and improve the quality of behaviour analysis. Such a step is crucial since the distribution of various features, such as counts for file operations (NtReadFile) or volume of resources used for memory allocations (NtAllocateVirtualMemory), may differ exponentially and therefore skew the overall detection models. Abnormal feature variation is the main cause of the difficulties involved in normalization in behavioral-centric analysis, particularly on FSMs and PBAs. The counts on file operations, such as the NtReadFile calls, can greatly overshadow the counts on tasks related to registry operations, such as RegDeleteKeyA. These, if uncorrected, tend to skew the outcome of machine learning models. Normalization, on the other hand, helps achieve uniformity across all variables, such that none is given undue weightage during the analysis. This protects machine learning models, especially neural networks and support vector machines, that are sensitive to the scale of input, whichever way they learn. Further, it ensures better behavior by allowing the subtle changes in the malware behavior, say, file operations count or memory allocation patterns, to trigger the anomaly detection rather than ignoring them.

Here's a detailed explanation of normalization specific to behavior-based techniques:

Normalization Techniques for Behavior-Based Features

- **Min-Max Normalization**: Scales features to a fixed range, typically [0, 1]:

$$\mathbf{X_{norm}} = \frac{\mathbf{X - Xmin}}{\mathbf{Xmax - Xmin}}$$

It’s Suitable for FSM and PBA because it preserves relative differences between behaviors.

- **Z-Score Normalization (Standardization)**: Centers features around the mean with a standard deviation of 1:

$$Z = \frac{X - \mu}{\sigma}$$

Usually Useful when FSM or PBA features have outliers, as it reduces their impact.

- **Logarithmic Transformation**: Applies a logarithmic scale to reduce the impact of large values:

$$X_{log} = \log(1 + X)$$

Effective for features like NtReadFile or NtCreateFile with skewed distributions.

- **Scaling by Group**: Normalize subsets of features separately: for FSM Features, normalize all file-related features (NtCreateFile, NtReadFile, etc.), and for PBA Features, normalize process-related features (NtAllocateVirtualMemory, RegDeleteKeyA, etc.).

Which permits every cluster to keep its variance within itself and thus shine light on some aspects of the performance of the system that were specific to one behavior alone. In Weka, we employ the following: NtCreateFile, NtReadFile, and NtClose features for FSM, and for PBA, we use NtAllocateVirtualMemory and RegDeleteKeyA.
While analysing anomalies like sudden changes in the frequency of file creation or file reading for FSM features, comparison between file operation metrics becomes straightforward after graph normalization. For PBA Features On the other hand, processes and memory behaviour patterns become clearer so that even relatively minor changes, such as increased memory allocation and registry, are easier to notice.

The term “normalization” as used in the context of file system monitoring and behavioural process analysis refers to normalization in processes that deal with the dataset preparation. It rescales the features into a common range, like in [0, 1], which ensures

that features with large differences in their range, such as file operation counts and registry changes, are normalized and treated equally during analysis. Such normalization, as explained above, helps to prevent large-scale features from overriding the models and also assists efficient scaling of sensitive learning algorithms such as neural networks and support vector machines. Data normalization also provides a course of enhancing anomaly detection because minimal changes are made to the features, allowing better display of data and helping preserve the average distance during clustering or dimensionality reduction methods. All in all, more accurate, efficient, and more interpretable ransomware detection outcomes are achieved through the process of normalization.

5- Feature Selection:

It is a very significant task in the data preparation stage before they can be used for behaviour-based techniques because it specifies the relevant ones that will be important in the detection of ransomware and discards the others, which are redundant. The detection model becomes better and incurs less computation by concentrating on the most informative attributes of the data set due to the feature selection process. This process frequently requires correlational research in which some features are removed if they are very correlated with others and hence do not add much value. Other techniques for feature selection, such as information gain and chi-square evaluation, are useful in ranking the features so that, within the set of features, those that are more important to the task at hand, classification, are prioritized when all are not equally useful in differentiating between classes, particularly the operating behaviours of ransomware, which is the focus of this research. In the case of high-dimensional datasets, dimensionality reduction, such as Principal Component Analysis (PCA), can be applied to focus on important features while reducing the rest. This step enables the selected features, which are suitable for FSM and PBA techniques, to represent the significant and relevant behavioural features; therefore, it leads to higher model accuracy as well as easier interpretation of the model. The feature selection process identified 224 significant features based on their information gain scores, ensuring that no potentially valuable behavioural indicators were excluded. This analysis certainly complements those that emerged in the previous steps, including important features, Total_File_Ops, Memory_Behavior, and Cryptographic_Behavior. From the selected features, a balanced compromise between relevance and completeness for behaviour-based ransomware detection is provided. And

more importantly, it was ensured that all features with non-zero importance weights were used in the process, thus enhancing the model while minimizing redundancy and increasing explainability. To streamline the dataset further, you can manually remove 14 low-impact features that showed minimal contribution.

6- Dataset Splitting

The data in this study was semi-sorted into two sets in order to ease the model evaluation and training. The dataset was allocated in terms of 80 percent for training and 20 percent for testing. An eighth part of the data, which is the training set, was used in building and training the machine learning model. Out of the remaining data, 20% was said to be the testing dataset, which involved itself with new data in examining how effective the model functioned. This entire partitioning exercise was done using the Resample filter in Weka, contained in the Preprocess tab. The sub-sampling was done so that the distribution of the class was the same in both parts. This was key to ensuring that all classes of ransomware were represented in both training and testing sets, thus decreasing problems associated with class imbalance. Upon splitting, the training set had 15,411 samples, whereas the testing set contained 3,853 samples. Such an arrangement guaranteed that both the working sets had the same characteristics and class labels so that there were no mistakes occurring during model testing and evaluation.

Also, by splitting the dataset in this way, the study makes sure that the modeling would be done in a way that could be assessed in a manner that is dependable. This technique helps detect problems of overfitting and the potential of the model to forecast new data.

7- Model training

Model training is most important for behavior-based techniques, as it allows the detection system to recognize the behaviour of ransomware that may be sophisticated and is beyond rule-based methods. Behaviour-based methods and techniques, such as File System Monitoring (FSM) and Process Behaviour Analysis (PBA), produce quantitative metrics such as system call interactions, memory usage, and file usage. Many of these characteristics are non-linear and can be learned by machine learning models and used to differentiate between ransomware activities and non-ransomware activities. This aspect of model training assists in generalizing from the data such that it is able to detect new or polymorphic behaviours of ransomware that deviate from anything previously seen. Furthermore, models that are trained in the behaviour of ransomware are able to withstand

small changes in ransomware, making them highly robust and flexible. Integrating the training of the models within the analysis of datasets with a great scale in size and minimal reliance on rule-based configurations increases the efficiency and capacity of the behaviour-based detection schemes.

The dataset must be trained in models because it is created purposefully for detection using machine learning techniques, whose attributes, like system call frequency, frequency of registry interactions, and frequency of cryptographic activity, allow one to have quantitative evidence of the behaviour of ransomware. These attributes or features portray intricate patterns and relationships that are nearly impossible to influence by static rule-based systems. There are significant odds of machine learning models detecting attacks as a result of abuse of these parameters, which can be modelled for purposes of training. Furthermore, the abuse of these parameters relates to how specific ransomware exhibits unique behaviours that should be threats on their own. The dataset also contains multi-class labels that correspond to various attacks or types of ransomwares, making it impossible to classify classes without a properly trained model. Training of models also permits generalization; in this instance, it is the models used to permit the detecting of variants of ransomware that are new or modified, such that they share the same behaviour as those in the dataset. These features, including having the capability to scale across the diverse features in your dataset, make the need to train models important for the purpose of evolving effective and robust ransomware detection.

In the absence of model training, ransomware detection is still fraught with challenges, particularly when it comes to static and rule-based systems. Such systems rely on preset thresholds or heuristics that have a fixed nature and are not able to adjust to the complicated and changing pathways of ransomware activities. For instance, a high NtCreateFile count is expected to be high in some cases of malicious activity; however, legitimate processes may also have such counts, giving rise to missed detection or false positives. Your dataset also contains highly dimensional data of 240 features, many of which interact with one another in a nonlinear fashion—relationships that static rules could not possibly hope to capture. Designing rules manually for such a type of complexity is not only unfeasible but also cripples the scalability, since man-hours and expertise are essential to modify the rules for fresh ransomware variants. Additionally, generalization is another feature that rule-concentrated systems cannot provide; hence,

they would be of no use against slightly deviant novel ransomware variants. These limitations highlight the importance of model training in advancing ransomware detection algorithms that are scalable, effective, and adaptable.

The results from the training and testing tables indicate how good and effective the sample set of the ransomware detection research is. The following points summarize the key observations and demonstrate why the dataset is important for this research:

Table 5. *Training Data*

Metric	Value
Correctly Classified Instances	92.979% (14329)
Incorrectly Classified Instances	7.021% (1082)
Kappa statistic	0.8448
Mean absolute error	0.0198
Root mean squared error	0.1396
Relative absolute error	15.2407 %
Root relative squared error	54.7826 %
Total Number of Instances	15411

Table 5 shows that the performance of a model on a training dataset is quite impressive, showing that there were a successful learning and generalization. The model performed with a high accuracy of 92.979% after correctly classifying a large number of 14,329 out of 15,411 instances. This means there are only 1,082 misclassified instances, or rather, activities, which translates to 7.021 percent of activities, which indicates a small set of errors. The Kappa statistic value of 0.8448 signifies good agreement between what was predicted and the classification that was actually done. There is a space between the two statistics that was accounted for. Any kappa that is greater than 0.80 is considered strong. The ability to do better than random chance is further strengthened by the high Kappa score. The MAE, or the Mean Absolute Error value, of 0.0198 means that on average, the model's predicted value is very near the actual value, with the MAE being a little higher only due to variance. RMSE, or the Root Mean Squared Error, increased to 0.1396, meaning there were some bigger errors, but the mean was still good. The RAE, or Relative Absolute Error, at 15.2407%, along with the RRSE of 54.7826%, are both low indicators, meaning the model did a good job at reducing errors compared to the base model.

In general, it can be said that the metrics indicate that the model has been able to leverage the training data effectively and also maintains an equilibrium between accuracy and error minimization. High accuracy, a high Kappa value, and low error rates point to a model

that is very well adapted to the task of ransomware detection, with the only errors in classification being those due to imbalanced classes or slight behavioural similarities between certain ransomware types.

Table 6. *Testing Data*

Metric	**Value**
Correctly Classified Instances	93.329% (3596)
Incorrectly Classified Instances	6.6701% (257)
Kappa statistic	0.846
Mean absolute error	0.019
Root mean squared error	0.1376
Relative absolute error	14.9791 %
Root relative squared error	55.1894 %
Total Number of Instances	3853

The correctness of the testing dataset results shown in Table 6 reported by the model is not a one-off case, suggesting that the model has developed strong generalization powers. The model was able to perform well out of the 3,853 data points, with 3,596 of them being accurately classified, translating to an accuracy rating of 93.329%. The number of wrongly classified points was only 257 (or 6.6701%), hence explaining the low level of errors and, in this case, substantiating the findings of the training set. The Kappa statistic in this case is 0.846, indicating a very good agreement between the two sets of classifications in question. This result also confirms that in the case of that model, performance was not the result of an effect that could artificially come by chance, which allows for firm confidence in the information that the model will be relying on for future events. Of note is that the Kappa statistics for training (0.8448) and test (0.846) data sets are at the same level, meaning the model is still capable of doing predictions without unnecessary overfitting.

The MAE statistic, which stands for Mean Absolute Error, uses a scale of zero to one as its metric, with the number being in the range of 0.019 to 0.020. MAE can be explained as the absolute sum of the error the average prediction shows in all output data points blocks where the number has no reaction with or contains a count of one. On the other hand, again using the MAE scale, we can comprehend the RMSE score, which on the scale of MAE measures up to 0.1376. The MAE shows that the predicted outputs do not deviate too much from the actual output, but the RMSE shows that a great deal of errors

is present, but keeping in mind the low error values, nearly all of the model outputs would closely resemble the correct and accurate outputs. RAE stands for Relative Absolute Error, which was reported as 14.9791%, and RRSE as 55.1894%, indicating that predictions would always be in line with the training results achieved by the machine, thus RAE and RRSE augment the overall predictive performance of the model.

Summing up the results of the tests, we see that with a good level of precision (93.329%), the model predicted the results with locked data points while only having an allowable level of deviation (92.979%). One could say of the model that when the model was being fed insider data, it underwent some slightly excessive training (overfitted) as it periodically yielded results with which it was not trained. The conclusion that one can draw out of these tests is that the model achieved its intended objective of conforming within the desired boundaries of performance whilst retaining its robustness.

8- Model evaluation

The model evaluation in such techniques as File System Monitoring (FSM) or Process Behavior Analysis (PBA) is about evaluating the performance of the system in the detection of ransomware actions according to a set of rules, a pattern, or statistical thresholds that have been set. However, unlike models in machine learning, behavior-based methods use available information, including the use of logs, system calls, and other behavior patterns, to identify the existence of ransomware.

- **Detection Accuracy** refers to the way a behaviour-based system interprets activities that are normally registered to be regarded as activities of ransomware and other activities that are within safe limits: normal processes. It is established by assessing the outcomes of the system, whether it flagged a ransomware activity or benign activity correctly when compared to known levels or expert opinions. To put it plainly, this indicates how many times the system issues a green light when determining the potential of an artifact being ransomware.
- **Recall or Sensitivity**: Indicates the system's ability to correctly detect ransomware activities:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+Fales negative}}$$

High sensitivity ensures the system catches most ransomware events.

- **False Positive Rate (FPR)**: Measures how often benign behaviors are incorrectly flagged as ransomware:

$$FPR = \frac{\text{Fals Positives}}{\text{Fals Positives} + \text{True negative}}$$

A low FPR is critical to reduce unnecessary alerts and maintain system reliability.

- **Precision**: Focuses on how many of the detected threats are actually ransomware:

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{Fals Positives}}$$

High precision ensures the alerts generated are highly accurate and actionable.

- **F1 Score:** it combines precision and recall into a single metric, providing a balanced view of the system's performance:

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the case of behavior-based methods of the techniques, the F1 score is important because it deals with the problem of too many false positives vs missing too many true ransomware positions. This indicator comes in handy in datasets with significant imbalance, such as classifiable activities, with most activities being benign and only a small portion being ransomware activities.

- **Latency**: it deals with the time a behavior-based model takes to report ransomware commencing activity that it perceives to be malicious. and where a longer time is feasible so as to enhance mitigation and response.

Table 7. *Training evaluation*

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Sodinokibi	0.5	0.0	0.959	0.5	0.657	0.691	0.993	0.771
Normal	0.631	0.0	1.0	0.631	0.774	0.777	1.0	0.996
Conti	0.998	0.019	0.993	0.998	0.995	0.982	1.0	1.0
CryptoLocker	0.171	0.0	1.0	0.171	0.292	0.41	1.0	1.0
LockBit	0.815	0.0	1.0	0.815	0.898	0.902	0.999	0.961
Ryuk	0.962	0.015	0.738	0.962	0.835	0.835	0.999	0.989
WannaCry	0.973	0.055	0.602	0.973	0.743	0.742	0.997	0.981

The training dataset in Table 7 indicates that the model is robust across several ransomware families while showing both strengths and weaknesses for specific classes.

For Sodinokibi/REvil, the true positive rate is 0.50 in our test set (Table 7), meaning only half of the Sodinokibi cases were correctly identified. This observation is consistent with prior reports describing REvil/Sodinokibi as a highly evasive, targeted family that employs techniques hindering detection. The recall is low, but Sodinokibi's precision is 0.959, meaning that every time a prediction is made for this class, it is very confident of the class being true. FMeasure, which is one of the scores obtained for Mobidik class recalls, is 0.657, and the Matthews Correlation Coefficient of 0.691 indicates that the model still struggled to capture Mobidik class characteristics comprehensively. Nevertheless, there is the converse scenario with the model being able to differentiate this class well, as indicated by the high ROC Area of 0.993 and PRC Area of 0.771.

In the normal class, a TP rate of 0.631 means that 63.1 percent of those normal class cases were correctly identified by the model as belonging to the normal class. The precision is perfect at 1.0, meaning that this class has no false positive claims outstanding. F-Measure of 0.774 and MCC of 0.777 are indicative measures of the overall performance. An ROC Area of 1.0 indicates that discrimination is very good, and a PRC Area of 0.996 shows an impressive degree of precision recall. The performance is once again superb for the Conti class with a TP rate of 0.998 and precision of 0.993. An F – Measure of 0.995 and an MCC of 0.982 also indicate the model is great in classifying the Conti instances. Moreover, the ROC Area and PRC Area are both equal to 1.0, which increases the strength of these results. In contrast, in the case of the Cryptolocker class, TP Rate is alarmingly low at 0.171, which means only 17.1% of instances are correctly classified, primarily because of its higher volume. Nonetheless, precision stands at a higher value of 1.0, indicating few predictions made for this class are accurate. An F - Measure of 0.292 and an MCC of 0.41 showcase how difficult it is for the model in terms of this class. ROC Area 1.0 and PRC Area of 1.0 seem to suggest that the model is good at classifying CryptoLocker instances but not really at recalling them. The LockBit class, however, scored quite well also with a TP rate of 0.815 and a precision of 1.0. Moreover, F-Measure and MCC also confirm that the model is decent with a score of 0.898 and 0.902, respectively, in this area. The ROC Area and PRC Area stood at 0.999 and 0.961, which further builds on this high performance. In the case of the Ryuk class of TP Rate Amounts to the value of 0.962 with a Precision of 0.738. The F-measure of 0.835 and the MCC value of 0.835 mean that the model does reasonably well in predicting this class, although

false negatives do happen. The ROC Area being 0.999 and the PRC Area being 0.989 indicates that this model performs excellently on this particular class.

Finally, for the WannaCry class, the TP rate comes up to 0.973 while their precision is at 0.602. For the F-Measure 0.743 and MCC 0.742, the figures seem to be within acceptable ranges, but the model can at times classify some instances incorrectly. The ROC Area of 0.997 and PRC Area of 0.981 are indicative of very good overall classification ability for this class.

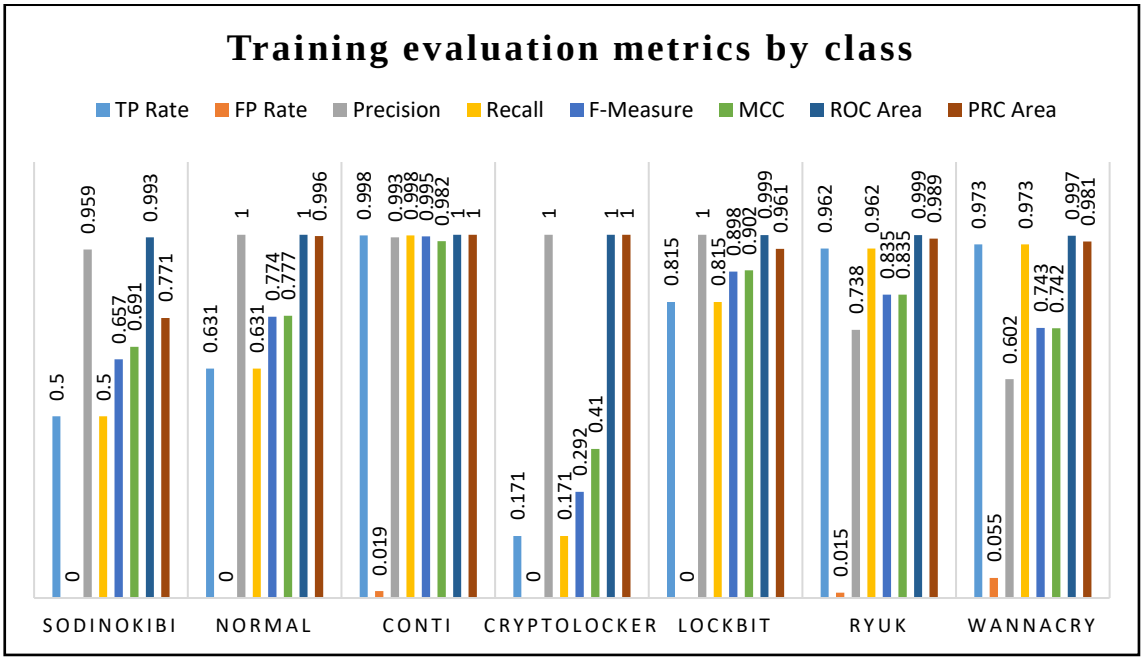

Figure 11. *Training evaluation metrics by class*

Figure 11 shows that this bar chart shows numerically how any class or a particular ransomware class taught in 'training' evaluation metrics. True Positive Rate (TP Rate), False Positive Rate (FP Rate), Precision, Recall, F-Measure, Matthews Correlation Coefficient (MCC), Area Under the Receiver Operating Characteristic (ROC) curve, and Area Under the Precision-Recall Curve (PRC) are all represented using varied colors. This facilitates a clear visual analysis of the competence of the classifier across various classes in relation to their performance while indicating potential weaknesses and strong points during the training period.

Table 8. *Testing evaluation*

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Sodinokibi	0.514	0.0	0.95	0.514	0.667	0.697	0.991	0.717
Normal	0.624	0.0	1.0	0.624	0.769	0.772	0.999	0.989
Conti	0.999	0.018	0.994	0.999	0.996	0.986	1.0	1.0
CryptoLocker	0.143	0.0	1.0	0.143	0.25	0.376	0.999	0.987
LockBit	0.742	0.0	1.0	0.742	0.852	0.86	0.997	0.932
Ryuk	0.966	0.013	0.754	0.966	0.847	0.847	0.998	0.98
WannaCry	0.975	0.053	0.586	0.975	0.732	0.733	0.997	0.977

The responses provided by the models in the testing dataset, illustrated in Table 8 substantiates the position that models perform comparably on the testing and training sets. In this case, the TP Rate class for Sodinokibi has slightly improved, now registering at 0.514, which means that 51.4% of the instances were classified correctly. The Precision has a strong value of 0.950, which indicates a good level of assurance during the forecasts of this class. The parameters of F-Measure and MCC also maintained their values – 0.667 and 0.697, respectively, while the ROC Area and PRC Area were 0.991 and 0.717, which show that the class was also well recognized by the model.

The Normal class has a TP Rate of 0.624, while Precision remains at the peak value of 1. The F-Measure was recorded at 0.769, and the MCC at 0.772, which shows that the classification was performed reliably. The Area under the ROC curve and in the PR C Area were 0.999 and 0.989, respectively, indicating good characteristics of discrimination of this class, which conformed to the earlier findings from the training phase. The performance of the Conti class never disappoints; it records a TP Rate of 0.999 and a Precision Rate of 0.994. The value of F Measure and Performance of MCC were more than great, 0.996 and 0.986, respectively. The parameters of Area under the Curve and PR C Area were both recorded at 1.0, which ensures the efficacy of the model on this class. With reference to the CryptoLocker class, the TP Rate is still low at 0.143 or approximately 14.3% of the instances being correctly identified. The value of Precision is given as 1.0, whereas the value of the F-Measure has been indicated to be only 0.250, with the corresponding MCC value being approximately 0.376. The values of the ROC Area as 0.999 and PRC Area as 0.987 would indicate that while the model can distinguish this class in research, the model in such a case would have problems in recall. The LockBit class's TP Rate is also 0.742, or 74.2% instances being identified every time. In this case, the Precision does not change and is still 1.0, whereas the F-Measure in this

case is recorded at 0.852, and MCC, being around 0.860, shows good performance of the model. But in this case, the area under the curve for the PRC is recorded at 0.932, with the area under the curve of ROC being at 0.997, indicating strong classification ability.

As for the Ryuk class, the TP Rate is approximately equal to 0.966 with a precision of 0.754. The F-measure turns out to be 0.847, and the MCC is also equal to 0.847, both of which can be used to establish a reliable classification. Record of ROC Area being 0.998 and the area under the curve of PRC being 0.980 again justifies the model in this case. The last class is the WannaCry class, which has a TP Rate of 0.975 and a Precision of 0.586 in this case. The F Measure in this case is equal to 0.732, while MCC is equal to 0.733. In this case ROC area is registered at .997. In contrast, the area under the curve of PRC stands at .977, suggesting that the model may still struggle with occasional misclassification but overall can discriminate the class well.

From the evaluation, it is observed that the model is consistent during both training and testing on the datasets with good precision and ROC Area values across most of the classes. Nevertheless, despite the challenges faced, this particular model can generalize quite well, and the performance is quite satisfactory as well.

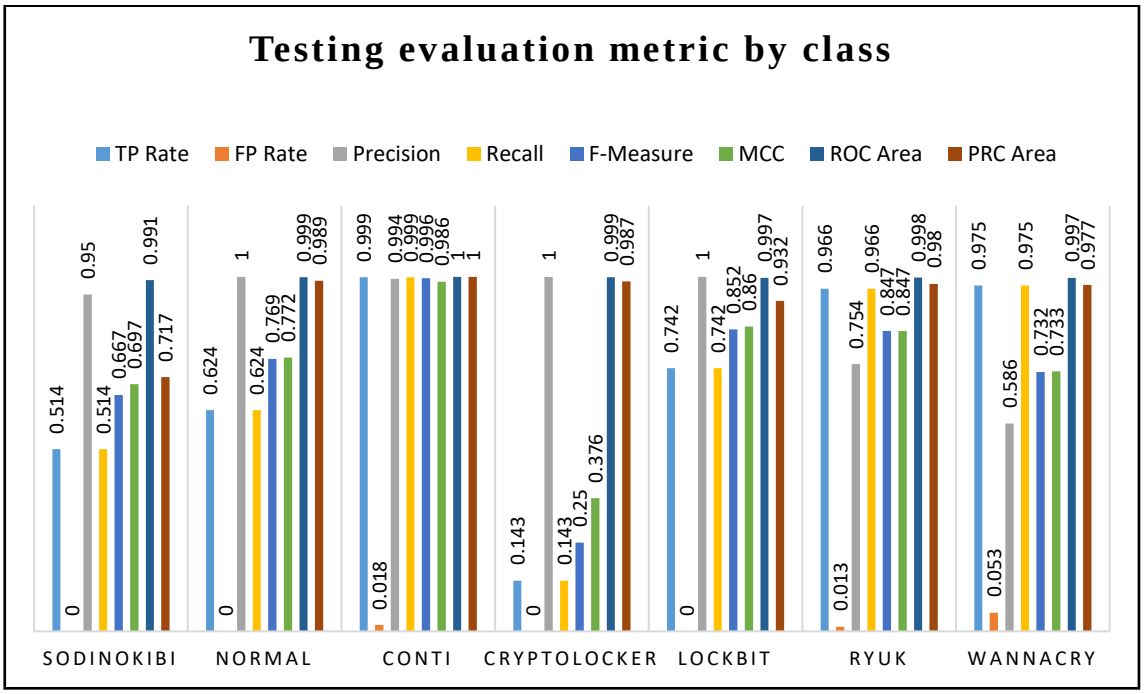

Figure 12. *testing evaluation metric by class*

This graph in Figure 12 shows how each ransomware sample was tested – it shows how effective the model was against every class of ransomware during the tests It gives the values of TP Rate,

FP Rate, Precision, Recall, F-Measure, Matthews Correlation Coefficient, Area under the ROC Curve, and area under the Precision-Recall Curve. These metrics allow an evaluation of how well the classifier is able to predict the outcome on the test data and depict the strengths of the method as well as the weaknesses that need to be addressed in real-time applications.

3.2.2.4.2 NBA dataset

To prepare the NBA dataset for analysis using Network Behaviour Analysis (NBA) techniques, the following steps can be followed:

1. **Understanding Relevant Features**

The first step in preparing the NBA dataset is identifying and understanding the key features relevant to ransomware detection. These features include the Time attribute, which records packet timestamps, and the Source and Destination attributes, representing the network addresses involved in the communication. The Protocol field indicates the network protocol used (such as TCP, UDP, or DNS), while Length measures the size of each packet. Additionally, the Info field provides descriptive details about each packet's content. By identifying these relevant features, we ensure that the dataset retains the necessary information for effective behaviour analysis. For the NBA dataset, focus on columns like Time, Source, Destination, Protocol, Length, and Info.

2. **Data Cleaning**

Data cleaning entails processes such as the elimination of duplicate entries, as well as taking care of missing or unfilled entries that may be indicated within the dataset. Duplicate entries create bias and reduce the output obtained from the analysis; hence, they need to be detected and removed. Besides, the data set should also be confirmed to see whether it contains missing values in sufficient areas as well. Missing data, when found or suspected, has specific ways to be handled. The strategies include deletion of incomplete rows or substitution of missing values. This step ensures maximum integrity of the data set.

3. **Feature Engineering**

Feature engineering means introducing new features that add a different perspective on the action happening in the network. The NBA has some useful features, such as packet interarrival time, which measures the duration between two successive packets and can

assist in the detection of unusual network behaviours. Other possible new features would be to collapse protocols into categories such as TCP, UDP, ICMP, and many more, for better analysis. Furthermore, determination of communication activity over a given period may assist in spotting areas of heightened or diminished signs in network usage. These additional features increase the chances of the dataset making more meaningful behaviour patterns.

- Packet Interarrival Time Calculation

The knowledge gathered on the packet interarrival time helps explain the rationale for the time between the two sent packets, which points out the anomalies in the time period between the sending and receiving of packets, as well as an increase in the volume of the packets sent, which is a behavior characteristic of ransomware. This was done through the use of the diff methods of the library pandas in Python. The Interarrival_Time column of the dataset that was created during the analysis of the difference in time between two packets is stored for analysis down the line.

- Protocol Categorization

To make the protocol analysis simpler, we grouped the various protocols into broader categories (e.g., TCP-based, UDP-based, ICMP-based). This was done by creating a mapping dictionary and applying it to the Protocol column using the map function from Python's library. The newly created columns were then added above, so that it was easy to carry out a behavior analysis based on the constructed results.

- Traffic Volume Calculation

The traffic volume was worked out as the number of packets sent between the source and destination in defined time slots. This process involved creating a Time_Window column to show time intervals of interest and. groupby() function in Python to count packets exchanged in each time interval. This information was written in the Traffic Volume column of a new file.

- Session Duration Calculation

We calculated the session for each source and destination pair in order to define the duration of the network sessions. This was done by getting the minimum and maximum time stamps for each session and finding out their difference. The session duration was

finally incorporated into the data set as a new column named Session Duration by making use of the pandas Python library.

4. Timestamp Alignment

Timestamp alignment ensures that the temporal attributes for the NBA dataset are in the same configuration style that is used for the FSM/PBA dataset. For example, if one dataset is in seconds and the other in milliseconds, the chronology should all be adjusted to one of the units. This step is very crucial for ensuring that time order continuity is preserved after the datasets are integrated. Allowing it this way helps correlate datasets about the same event while eliminating the chances of committing TCE errors with the temporal aspects. And in the NBA dataset, the timestamp is in seconds just like the timestamp in the RansomSet dataset.

5. Feature selection

One of the most crucial steps during the depth of analysis is feature selection, where only the necessary attributes relevant to the analysis are kept. Within the scope of this project and in relevance to the NBA dataset, elements that are likely to be useful include Time, Source, Destination, Protocol, Length, and new features, hence the term engineered features, including interarrival time, traffic volume, and many others. Deleting non-useful, non-precise, or duplicated features results in a compact and stripped dataset, which increases machine learning models as well as behavior analyses. In the NBA dataset, no feature can say it is not necessary, so in this case, there is no need for feature selection.

6. Source Labelling

In order to be able to easily address any questions related to traceability after merging, there is a need to include a distinctive identification mark for each dataset. Since the Source column that indicates the network addresses exists in the NBA dataset, there is no need for an additional column. On the other hand, we will include either metadata or a dataset-level identification mark while merging datasets to show that such data was obtained from the NBA dataset. Such a label will assist in differentiating the two data

sets, the NBA and the FSM/PBA dataset, and it allows one to perform a much easier analysis or even visualization of the merged data set.

7. Format Standardization

The format standardization doesn't signify the identical column names and identical data types for the FSM/PBA dataset and the NBA dataset. The former and the latter contain different types of information. It means, however, that such a level of comparability is provided as far as user-specified properties and structures enable. For example, each dataset can have a column of a timestamp, meaning that both datasets are related to one time clock with the times transformed to a particular measure, e.g., seconds. We ensure that different columns serve different purposes rather than rebuilding the purpose and giving the same name to them. Similar features should have consistent data types; that is, for the timestamp, which can be recorded either as a float or as a datetime, and for categorical features such as Source, Destination, and Protocol, which are certainly recorded in a string format. When merging datasets, complete information must be provided, that is, the origin, which forms the basis for the genesis of the target datasets in the end, so that they can be understandable. For example, including a Behavior_Type column and coding it as "System_Behavior" for the FSM/PBA dataset and "Network_Behavior" for the NBA dataset provides a quick identification of the dataset. This methodology captures the spirit of each of the datasets in the merged form and highlights the diversity when joint analysis is performed in the ultimate merged form of the dataset for behavioural analysis or machine learning-based tasks.

8. Preserve Comprehensive Data

In this research, we made a decision not to impose any protocol-wise filtering, such as DNS or TCP, so as to preserve the entire NBA dataset. This approach works in favour of all the outlying or abnormal network behaviour, as the dataset can be analyzed for any forms of behaviour-related patterns. Which means the dataset can cover and include all packet types, making it suitable for a variety of analyses and not ruling any unique behavioural patterns out as useful. This applies to the scenario of this research, where all protocols in the dataset are analyzed as opposed to narrowing it down to one protocol, as the aim is to picture the behaviour patterns of the whole network and not a single aspect. However, once the goal is achieved, it raises the bar for prospective studies by remaining adaptable through allowing protocol-based filtering.

3.3 Merging of FSM/PBA and NBA Datasets

In this research, we combined the FSM/PBA dataset and the NBA dataset to obtain a set that encompasses file system, process behaviour, and network behaviour. These datasets were essential to complete the behavioural and the machine detection techniques in a workable and proper fashion. Considering system calls and network data sets, the integration of system calls and network data was done using an integration strategy that was built on a timestamp. This dressing up was needed in order to depict the particular order in which the events occurred accurately for the unified dataset purposes. This integration is not merely a technical merge; it is a purposeful combination where behaviour-based models (BBM) play a foundational role. BBM techniques provide the contextual signals and rich behavioural features—such as system calls, process patterns, file operations, and network behaviour—that serve as meaningful input for machine learning models. These features enable machine learning algorithms to learn more effectively, improving classification accuracy and the system's overall reliability in detecting ransomware.

3.3.1 Merging Methodology

The process of integration of the two datasets of FSM/PBA and NBA was all important because it is this step that sets the stage for the detection of ransomware through the behaviour and machine learning techniques. Such efforts must necessarily involve the use of a process (or operational) database and a network behavioural database, which have been made distinct in their characteristics, that is, a behavioural focus around file system, processes, and network activity. To this end, the integration of the various behavioural databases was achieved through a timestamp-based integration methodology, which ensured the different behavioural databases were actually merged across the various layers of behaviour without temporal disintegration.

In order to construct a unified dataset that encompasses behavioural data (FSM and PBA) and network traffic patterns (NBA), a hybrid approach was adopted. Timestamp-based alignment was employed to temporally synchronize the behavioural and network datasets, while ransomware family names were mapped consistently across both sources to ensure class coherence. Feature-level concatenation was then performed, resulting in a dataset

where each observation contains both system-level and network-level attributes for the same ransomware instance. This integration enables a comprehensive analysis of ransomware behaviour across multiple system layers.

3.3.1.1 Dataset Overview

- While utilizing the FSM/PBA dataset, system calls, file operations, and activities done in memory during the execution of ransomwares were done.

- The NBA dataset included network traffic that contained IP source and IP destination, protocols, sizes of the packets, and the length of the session in time measures.

Both datasets were obtained distinctly, but recorded the time in every event. These timestamps, which were the basis for recording system and network level events, were used to synchronize the datasets.

3.3.1.2 Timestamp-Based Alignment

In order to join the FSM/PBA and NBA datasets, a fusion technique involving timestamp-based alignment and ransomware-class mapping was used:

- Time Frames' Synchronization:

The procedure started by determining the start and end times when the ransomware was executed for both datasets, so that the merging could be done by behavioral time frames, which in this case were overlapping. The difference in the timestamp format between the datasets was made uniform by converting all timestamps to a rectangular uniform format.

- Event Alignment:

With the use of the standardized timestamps that were set, events in the dataset of FSM/PBA (like system events and file writes) were set against other events that occurred in the network area in the NBA dataset (like DNS lookups and TCP events). In such cases, two events occurring within a frame of time of about 1-2 milliseconds were considered related and thus merged. This particular size of the window was based on the expectation that logging events time with tools might take a second or two-second delay.

- Feature integration

Timestamps were rectified first, then features from both datasets were integrated into a new feature set. For example, system call features, including file operation frequency, were integrated with interpacket arrival network features to enhance the description of

the behaviour of each ransomware sample. To enhance the effectiveness of the dataset and to reduce the complexity of the analysis to follow, all repeated and non-crucial features of the dataset were cleared.

To illustrate the merging mechanism between the behavioural (FSM/PBA) and network (NBA) datasets, both a flow diagram and an algorithmic pseudo-code are presented below. The flow diagram provides a visual overview of the integration process, while the pseudo-code outlines its stepwise implementation for reproducibility.

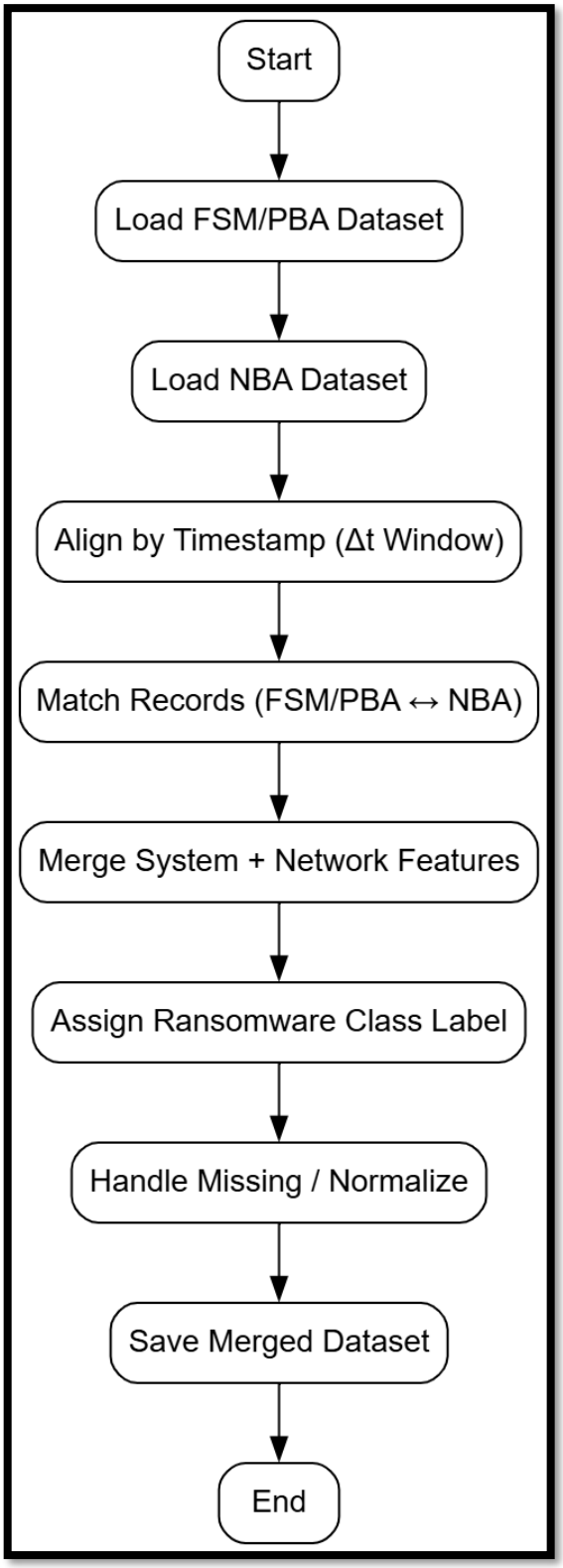

Figure 13. *Flow Diagram of the FSM/PBA - NBA Merging Process*

```
Input:
    F ← FSM/PBA dataset
    N ← NBA dataset
Output:
    M ← Merged hybrid dataset

Steps:
1.  Align timestamps between F and N within a Δt window.
2.  For each record f in F:
        a. Find n in N where |f.timestamp – n.timestamp| ≤ Δt
        b. If match found → combine f.features ∪ n.features
        c. Assign ransomware_class = f.class
3.  If no match found → retain f with null network fields
4.  Handle missing values and normalize attributes
5.  Append merged record to M
6.  Return M
```

Figure 14 . *Hybrid Dataset Merging (FSM/PBA + NBA) via Timestamp-Based Alignment*

3.3.1.3 Data validation

After merging the FSM/PBA dataset with the NBA dataset, a validation step was conducted to ensure structural and semantic consistency. The number of merged rows was compared against the original number of instances in the primary behavioral dataset (RansomSet) to confirm one-to-one alignment. Feature dimensionality was checked to validate the successful integration of network-level attributes without overwriting existing behavioral features. Class distributions were analyzed pre- and post-merge to ensure label integrity and avoid duplication or loss of samples. This validation confirmed the readiness of the unified dataset for downstream machine learning experiments.

Several checks were conducted to ensure that the integrity of the combined data set was not compromised in any way:

- Consistency checks: Events resulting from the integration, in this case, network events and events from the system integration, were sampled and visually compared to verify integration between system and network events.

- Statistical validation: The set of features that existed before merging and the set of features that resulted after merging were set in comparison so that their similarities or differences could be identified, even if minimal.

- Class distribution analysis: A check was performed on the appended data set to ascertain that the individual classes of the merged sets were still in a reasonable ratio and were a true representation of what was present in the initial sets.

The integration has culminated in the creation of a detailed picture that captures the activity of the file system, processes, and the network's activity in a sequential form. This dataset was instrumental in applying behavior-based detection techniques as well as training machine learning models, allowing the researchers to have a comprehensive view of handling the problem of ransomware. The completeness and consistency obtained through the use of timestamps were a good basis for precise examination and model assessment. This consolidated dataset made it possible to analyse interactions between systems and networks, improving the capability of efficient detection of various forms of ransomware activities at different levels of the organization.

3.3.2 Outcome of the Merging Process

Through the fusion of two datasets, the FSM/PBA dataset and the NBA dataset, the resulting dataset can be characterized as all-encompassing and unified when it comes to attacks perpetrated by ransomware. For instance, the combination of file system actions, actions of processes, and actions on the network into one dataset made it possible to reconstruct the picture of the execution of a ransomware. This coordination was done using a timestamp that was sophisticated to ensure system-level events occurred in the timeframe of the network-level events.

As a result of the last merging process, an array of features describing different behaviors of ransomware and their multi-dimensional aspects was obtained. These features include inter-system call frequency of operations, operation frequency on a file, frequency of access to memory, frequency of packet interarrival times, frequency of session duration, and the protocols used. Such a dataset has great potential for deeper analysis of attacks through the use of blended features, significantly enhancing research on the detection of attacks through behavior and attacks using machine learning. The combined dataset has 15,411 instances and a composed size of 224 attributes. Each instance corresponds to either a ransomware sample or a benign activity and is considered at both the system and network levels. Ransomware classes are retained in the dataset in the form of original classes, such as Sodinokibi, Conti, CryptoLocker, LockBit, Ryuk, and WannaCry, and

also, normal (benign) activities are included. It is better to have this balanced representation since the dataset can then be used for training and evaluating detection models. The merged dataset not only provides an overall view of the activity of ransomware, but it also allows examination of relationships between various aspects of behaviour. For instance, it would be possible to test how operations on a file system correspond to activities on a network and thus determine a particular signature of a ransomware attack. This cross-layer analysis improves detection chances of ransomware that are likely to be missed if detection is focused on a single behaviour only.

To sum up, the end product of the merging process is a rich and multipurpose dataset that forms the basis for the later stages of this study. It facilitates the more sophisticated and reliable interception of ransomware by enabling thorough assessments of the detection techniques employed. This common dataset not only enhances the current investigation in progress but also helps in future efforts directed towards ransomware detection and behavioural analysis.

To illustrate the merging approach, consider a single ransomware sample labeled "Conti". The FSM/PBA dataset represents 242 features representing the frequency of specific API calls. The corresponding NBA dataset contributes 16 additional network-based features such as session duration, packet volume, protocol type, and interarrival time. After alignment, the final merged row for this sample contains 240 features, preserving all behavioral indicators while enriching the instance with network context. This concatenation ensures that both system and network layers are jointly represented for accurate modeling.

3.3.3 Evaluation of Model Performance After Merging

The integration of data sets is an important step towards getting a complete set that is ready for accurate ransomware detection. After integrating datasets, one needs to test how machine learning algorithms perform on the newly formed dataset. This part of the research reports on the outcomes of the training and validation experiments of the Naive Bayes model as applied to the protection against ransomware and normal instances. The analysis covers a variety of performance measures such as accuracy, error rates, and confusion matrices, which give an evaluation of the strengths and weaknesses of the

model in predicting the appearance of ransomware. These results are important in understanding the performance of the model on the whole and determining how further improvements can be brought about.

Table 9. *Metrix train performance for the merged dataset*

Metric	Value
Correctly Classified Instances	6473 (96.5975%)
Incorrectly Classified Instances	228 (3.4025%)
Kappa statistic	0.7575
Mean absolute error	0.0227
Root mean squared error	0.1503
Relative absolute error	20.1862%
Root relative squared error	63.4803%
Total Number of Instances	6701

The model was successful in the training arena, attaining an accuracy level of 96.60%, a feature that is associated with a 6473-instance classification of a total of 6701 instances. The total number of misclassified instances is 228, which accounts for 3.40% of the total instances. The Kappa statistic of the model stands at 0.7575, which indicates that there was marked correlation between the predicted instances and the actual instances, thereby illustrating the efficiency of the model.

The associated metric errors point to a good fit of the models. The predictive accuracy of the mean absolute forecast error (MAE) of 0.0227 and root mean squared error (RMSE) of 0.1503 explains the average prediction error to be low. Likewise, the relative absolute error stands at 20.18%, whereas the model's root relative squared error is at 63.48%, reconfirming the level of accuracy and precision of the model.

In the detailed accuracy by class, the precision, recall, and F-measure metrics differ by class. For the Sodinokibi class, the recall is 67.6%, which serves to indicate the effectiveness of the model in identifying this class as moderate. There is, however, a high precision rate achieved of 93.5%, indicating that the majority of predictions in this class are correct. The class normal has a slightly lower recall value of 61.0% with a perfect precision score of 100%, indicating that all the counts labelled as "normal" instances were correct. The class Conti had an absolute recall score of 100% with a high precision score of 96.5%, making it the class predicted with the most accuracy.

Table 10. *Metrix test performance for the merged dataset*

Metric	Value
Correctly Classified Instances	1613 (96.2411%)
Incorrectly Classified Instances	63 (3.7589%)
Kappa statistic	0.7698
Mean absolute error	0.0252
Root mean squared error	0.1582
Relative absolute error	20.8791%
Root relative squared error	62.4105%
Total Number of Instances	1676

The previous results were comparable between the test set and training set in terms of error metrics, which is how the less positive comment regarding the results was once expressed by the management. MAE and RMSE at a value of 0.0252 and 0.1582, respectively, are metrics that can continue to be classified in the good performance category. The case when RAE and RRSE show little difference to the training set indicates that the model can withstand new data that has not been seen before. As reported previously, the error metrics on the test set follow the same pattern as those on the training set. MAE and RMSE continue to stay low at 0.0252 and 0.1582, respectively, suggesting good performance. RAE and RRSE, on the other hand, are close to the training set, indicating that the model is robust to new unseen data. Recall rate for the Sodinokibi class improved to 71.4%, suggesting better detection of this class on the test set while maintaining high precision of 89.7%. On the other hand, the normal class is characterized by a top precision of 100%, while one finds a lower recall of 61.4%, which explains that addressing all instances of this class turned out to be a challenge. Conti class, on the other hand, enjoys a high precision of 96.2% with a recall rate of 100%. Conti class and Sodinokibi were still confused as age states, which resulted in a 0.84% misclassification rate; however, this remains insignificant in proportion to the overall system performance.

From what can be observed when analysing the performance of the model on various datasets, the Naive Bayes model seems to be performing well. Looking at the results, the training set was able to achieve 96.60% accuracy, while the test set managed to get 96.24%. The confidence in the predictions of the classes was high due to the strong precision observed in all the classes performed by the model, especially Normal and

Conti, where no false positives were reported. Considering the low errors of the model, it is reasonable to assume high reliability and consistency within the classifications done by the model, including the Mean Absolute Error and Root Mean Squared Error, both of which were quite low.

3.3.4 Behavior-Based Detection on Merged Dataset

This section evaluates the performance of behavior-based detection techniques when applied to both individual behavioral domains (FSM, PBA, NBA) and the unified, merged dataset. The analysis focuses on the effectiveness of handcrafted rules in detecting ransomware across different data layers, helping to determine whether merging behavioral perspectives strengthens or weakens detection capabilities.

Table 11. *base behavior merged dataset performance*

Dataset	Accuracy	Ransomware Precision	Ransomware Recall	F1-Score
FSM	33%	0.81	0.32	0.46
PBA	33%	0.81	0.32	0.46
NBA	3%	1.00	0.03	0.06
Merged	35%	0.87	0.35	0.50

These results clearly indicate that behavior-based detection, while interpretable and lightweight, faces significant performance limitations regardless of dataset scope.

FSM and PBA Analysis

When applied to the FSM and PBA datasets individually, behavior-based detection yielded relatively moderate performance. In both cases, ransomware precision was high (0.81), indicating that the rules correctly identified ransomware when it was predicted. However, recall remained low (0.32), suggesting a large number of missed ransomware instances. This is primarily due to the static nature of rule-based systems, which rely on predefined behavioral thresholds that cannot adapt to novel or stealthy ransomware behaviors.

NBA Analysis

The NBA dataset, which reflects network-level activity, resulted in the poorest performance among all datasets. With only 3% accuracy and 0.03 recall, the rules failed to capture meaningful network patterns indicative of ransomware. This is largely due to:

- Lack of labeled normal traffic (i.e., one-class data),
- Subtlety and variability of ransomware traffic patterns,
- Weak behavioral signals at the protocol level without context from system behavior.

Although the precision was technically 1.00, this is misleading, as no normal class existed, making the model's confidence in its few predictions artificially high.

- **Merged Dataset Analysis**

The merged dataset combines behavioral features from FSM, PBA, and NBA layers. While this fusion provided the highest F1-score (0.50) among all tested datasets, the overall accuracy (35%) and recall (0.35) remain low. This result underscores a key trade-off:

- **Positive**: Merging enhances the diversity of behavioral indicators, giving the model more signals to work with.
- **Negative**: The integration introduces feature dilution and inconsistency — behavioral signals become numerically subtle, and binary flags are lost due to normalization or aggregation.

Additionally, applying simple rule-based logic over a high-dimensional, multi-layered dataset is inherently flawed. Unlike specialized FSM/PBA rules that work on discrete actions, the merged dataset lacks focus, requiring more flexible and adaptive mechanisms.

Although the result seems unexpected, several prior studies have similarly reported the limited performance of standalone behaviour-based detection methods, reinforcing the findings observed in this research. For example, in a comprehensive evaluation by **Zahoora et al. (2022)**, traditional behaviour-based rules applied to zero-day ransomware samples achieved recall rates below 25%, failing to detect a significant portion of emerging threats. The study noted that while such methods offer transparency, they often suffer from poor generalization and struggle to adapt to evolving attack patterns. These limitations were mitigated only when behaviour-based features were combined with machine learning techniques, resulting in a substantial performance gain. This directly parallels the results presented in this section, where behaviour-based detection applied to

both isolated and merged datasets yielded low recall (0.32–0.35) and modest accuracy (33–35%), further validating the need for integrated hybrid approaches.

These findings support the core hypothesis of this research: Combining behavior-based methods with machine learning and deep learning models (such as LSTM) is necessary to improve ransomware detection accuracy and reliability.

While BB detection offers explainability, its standalone performance is not enough, especially on integrated datasets where behavioral variance is high. This justifies the next stages of this research, where machine learning and LSTM models are applied to the same merged dataset, leveraging its rich behavioral diversity through data-driven learning rather than static rules.

3.4 limitation

Merging the FSM/PBA and NBA datasets and afterwards training the model allowed them to acquire a deep understanding of how ransomware behaves, thus demonstrating the effectiveness of behaviour-based approaches to detection. However, just like every research undertaking, certain limitations may require attention if a fair view of the research is to be provided.

A major constraint was being able to cope with class imbalance through the training phase. Some classes of ransomware malware, for instance, CryptoLocker and Sodinokibi, were not adequately represented in the dataset, and this influenced the model's performance regarding recall and true positives. SMOTE techniques were employed so that this problem might be resolved, and to a certain extent, the representation of the classes was improved. The composition methodology was, however, systematic, but it involved timing unification across all the monitoring systems. Adjustments were done in regards to control measures to statistically improve the validity of these results; even minor variations in the perfection of timestamping could have hampered the usefulness of the developed composite dataset. However, such variations were reduced to a level that they couldn’t affect the results and reliability of the outcomes. Feature selection was a crucial aspect of this study, in addition to model building. From the dataset, it can be discerned that a total of 224 features related to various system and network behaviours were covered. While this richness indeed provided useful information, it also opened the doors for noise and variability, which may have been a factor towards overfitting. Feature

engineering processes focused on the nutritional value of the dataset and avoided needless abundance. There is a limitation in terms of the usage of ransomware, as both the research and the data collection methods employed a Cuckoo sandbox and, hence, controlled environments. These controlled environments enabled uniformity and reproducibility, which are key aspects of scientific research. On the other hand, there are real-life ransomware attacks that happen in much more fluid and chaotic environments, which could bring in elements missed in experimental datasets. Subsequent work may investigate the use of the model in production environments to test its strength in different settings.

At the end, it can be concluded that the research developed a behaviour-oriented detector for the research. Though this was successful, as it proved the ability of the model to detect ransomware by means of actions taken on the systems and networks, proxy methods like using signatures of such ransomware or advanced deep learning techniques were outside of this research. Employing those methods in future work can augment the entire detection framework as well as enlarge the view of the development of ransomware tactics.

To sum up, this is a meaningful work that provides a strong basis for behavior-based detection of ransomware despite the constraints. The understanding gleaned and the techniques utilized demonstrate the scope of this approach and clear the path for additional developments in this important domain in the field of security.

3.5 summary

This chapter describes the techniques for ransomware detection that are both based on behaviour and ML models. The process started with the capturing of datasets that addressed various behavioural parameters such as file system monitoring (FSM), process behaviour analysis (PBA), and network behaviour analysis (NBA). Each of the datasets was subjected to methodological preprocessing, which included feature engineering and class balancing so that they could be ready for analysis.

For the purpose of integrating both datasets for analysis purposes, the FSM/PBA dataset and the NBA dataset were first combined using time stamps. This was done to make sure both system- and network-based activities were available as required by the time order. Finally, the overall result achieved through the merging process was a combined set

consisting of 15,411 instances and 224 features that had a broad dimension of different kinds of ransomwares. The datasets were then utilized to train and test several machine learning algorithms, with Naive Bayes topping the list. The model achieved performance accuracies of 92.979% on the training set and 93.329% on the testing set. In terms of inter-rater agreement, evaluation metrics such as the Kappa statistic, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) showed a degree of consistency and strong generalization. However, some issues, such as class imbalance, affected the detection of certain ransomware families, which was resolved by the SMOTE technique.

Nevertheless, the results confirm the expectations that the use of behaviour-based techniques coupled with machine learning approaches greatly assists in the detection of ransomware. The merged dataset, as well as the models created on it, is an effective starting point for exploring the activities of ransomware and furnishing important information on ways of improving detection systems. This chapter sets the stage for further investigations, especially in improving model performance and taking the approach into realistic conditions.

Chapter 4: Machine Learning Detection

4.1 Introduction

Machine learning has become an indispensable tool in the fight against ransomware. The training of models is done on system and network behaviour, and with this, machine learning can highlight minor signs of activities related to ransomware, and this can supplement the insights gained from behaviour-based techniques. And this in context will lead us to more fine-tuned data points of our inquiry, which in this case is: Which features indeed integrate with the learning of the system of machine learning for ransomware detection? This specialization in machine learning, in turn, enables users to provide all forms of analysis against attacks and bring all attention to automating tasks instead of dwelling in natural language.

Yet, there are other machine learning application challenges within this realm. The dataset is crucial in achieving reliable and accurate models, in that skewed or incoherent data will result in bias. Apart from that, algorithmic shortcomings like overfitting or poor ability to learn characteristics of the sparse, rare types of ransomwares are also serious issues. These issues require adequate preprocessing, feature selection, and validation with the aim of enhancing the strength of the models.

This chapter extends the work started in Chapter 3 on the methodology and shifts its focus to the application of machine learning techniques to the same data set that was used in behaviour-based analysis. Maintaining a level of overlap or consistency in the datasets, the study was able to fairly compare the two techniques while assessing the unique capabilities of machine learning in the analysis of ransomware. In addition, the observations made here prepare the ground for studies aimed at examining the effectiveness of combining behaviour-based approaches and machine learning in enhancing detection systems. Integrating ML in this research not only aims at evaluating its potential for classifying different types of ransomwares but also aims at exploring possible synergies when it is integrated with behaviour-based approaches. In this context, behaviour-based models (BBM) serve a foundational role. They provide rich and structured behavioural features—such as system call sequences, file modifications, registry activity, and network traffic patterns—that supply machine learning models with high-quality, context-aware data. These behavioral signals allow ML algorithms to detect patterns more effectively, improve classification accuracy, and reduce the risk of false positives. Rather than working in isolation, ML in this research benefits from BBM's

ability to frame the behavioural landscape of ransomware in a meaningful and learnable format. This chapter deals with the efficiency of the use of machine learning models in capturing the subtleties of the patterns of ransomware behaviours, which might be insufficient in behavioural models. It is a very relevant step as it aims towards meeting the anticipation of advancement of the ransomware detection mechanisms towards more robust and adjustable ones.

This particular chapter is constructed such that it facilitates a comprehensive analysis of the detection techniques involving machine learning. First of all, the chapter explains the research approach, concentrating on the choice of algorithms and the reasons for their selection. The elements contained in the experimental setup section further elaborate on the processes undertaken to formulate the dataset and train the models. An exhaustive discussion of the severe parallel evaluation metrics follows, providing some quantitative aspects of how the model performed. In the last section of the chapter, the problems that were solved and the constraints that were encountered are discussed, providing a basis for further improvement and future projects.

4.2 Research Design

Unlike Chapter 3, which focused on the behaviour-based system design and data preparation, this section explains the processes that were followed in developing and testing machine learning models aimed at ransomware detection. The strategy accentuates the sequential construction of the dataset, feature engineering, and adequate application of the few machine learning algorithms available, and therefore, the results obtained are dependable and correct. The domain of study is organized in such a way as to create a framework for repeatability, and it portrays the measures taken to clean the dataset, enhance the model, and the results of the evaluation.

4.2.1 Approach

Machine learning has emerged as a powerful tool in combating sophisticated cybersecurity threats, including ransomware. In contrast to more conventional efforts that depend on distinct sets of rules or specified signatures, machine learning utilizes algorithms to detect unstable or unusual complementary behavior of a concerned event.

This versatility helps security operatives track the activities of ransomware and its development when other tracking methods are failing.

In this research, behaviours of different types of ransomwares are combined with network activity and systems activity. Through the use of the merged dataset of FSM/PBA and NBA features, machine learning models can analyse and classify quite complex behaviours, which allow distinguishing between malicious and benign processes. Such models allow the prediction of the impact of previously unseen ransomware based on how behaviourally similar it is to already known variants that have been tested.

Specifically, the introduction of machine learning in this research is directed towards:

1. Enhancing the accuracy and efficiency of ransomware detection.
2. Solving the problems of class imbalance and feature complexity through feature selection and data balancing.
3. Creating a detection system that can provide accurate results in a variety of "real-world" situations.

In this context, the choice of the machine learning algorithms is elaborated on, the reasons for their selection are discussed, and particular design considerations concerning the incorporation of the algorithms for ransomware detection are mentioned.

In this research, the machine learning algorithms focus on accurately classifying ransomware behaviours while tackling issues of class imbalance and feature diversity. These algorithms were selected in terms of their efficacy to counter these obstacles, as well as verified success in the area of interest. The following talks about the devised algorithms and why they were used:

- **Naive Bayes**

Naive Bayes was chosen owing to its simplicity, efficiency, and good empirical results on high-dimensional datasets. Because of its probabilistic nature, it can estimate the chances of varying behaviour, which in turn is beneficial in the comparison of ransomware and benign processes. It is known that Naive Bayes disposes of both feature independence assumptions and their real-life implementations. However, Naive Bayes has demonstrated robustness in this research.

- **Support Vector Machines (SVM)**

SVM was adopted because of its ability to handle non-linear decision boundaries using kernel functions. Due to this, it is very proficient in detecting even the slightest differences in the processes of execution of ransomware and benign files in cases of complicated features. SVM is beneficial for datasets with clearly segregated classes and has started to be used for dealing with the class imbalance problem using cost-sensitive learning.

- **Random Forest**

Random Forest is a type of ensemble learning technique that was adopted in this study for its strength in dealing with noise and high feature dimensionality. Random Forest is capable of achieving high accuracy by constructing multiple decision trees, which makes it very suitable for ransomware detection tasks. It also provides the rankings of the features based on their importance, which helps in determining the most useful features in the data.

- **Gradient Boosting**

Gradient Boosting was included in this research due to its strong capability to optimize classification performance by combining multiple weak learners into a single powerful model. It works by building decision trees sequentially, where each new tree corrects the errors of the previous ones. This makes it highly effective at capturing complex patterns and subtle variations in the dataset, including the nuanced behaviours of ransomware. Although Gradient Boosting models may require more fine-tuning and are computationally more intensive than simpler algorithms, they offer high predictive accuracy and robust generalization, especially on imbalanced or noisy data—common challenges in real-world ransomware detection tasks.

- **Long Short-Term Memory**

LSTM, a type of recurrent neural network, was considered in this study due to its ability to handle sequential and time-dependent data. Since ransomware behavior evolves over time—triggering events in a specific order—LSTM networks can learn these patterns effectively. By capturing long-term dependencies in the sequence of system calls and network actions, LSTM has the potential to reduce false positives and improve

classification accuracy. This is especially useful in detecting ransomware variants that gradually escalate their behaviour before executing encryption routines.

The chosen algorithms collectively offer a balance between interpretability, accuracy, and computational efficiency. The selected algorithms provide, in an optimal proportion, ease of understanding, precision, and cost-effectiveness in computing:

1. Variety of Methods: The inclusion of probabilistic models (Naive Bayes), kernel-based classifiers (SVM), ensemble methods (Random Forest), deep learning methods (Neural Networks), and Long Short-Term Memory increases the strengths of the research in many directions.
2. Importance for Ransomware Detection: These tasks include the aforementioned, adapting to input and learning to classify based on data that is highly dimensional and very complex, and dealing with imbalanced data sets.
3. Scalability and Robustness: The selected models are well-suited for deployment in dynamic environments where ransomware activity may vary.

In addition, the base learners used in the stacking ensemble were selected to maximize model diversity and complementarity. Since ransomware behavior exhibits heterogeneous and non-linear patterns across file system, process, and network features, different learning paradigms were intentionally combined to reduce correlated errors. Specifically, probabilistic learning (Naïve Bayes), margin-based classification (SVM), and tree-based ensembles (Random Forest and Gradient Boosting) were employed as base learners because each captures distinct decision boundaries. This diversity enables the meta-learner to integrate complementary predictions and improve generalization, leading to more robust detection performance against unseen ransomware variants.

In this research, each given algorithm was originated by the trained algorithms on the fused data set, indicating its effectiveness level for ransomware detection. All the results are further analysed in the following section to come up with the best means of tackling this crucial cybersecurity issue.

The selection of the machine learning algorithms in this study was methodologically aligned with the structural and statistical properties of the dataset, including its dimensionality, class distribution, and temporal nature. Since the integrated FSM/PBA

and NBA dataset exhibits high dimensionality (over 220 behavioral and network-related features) and contains both categorical and continuous attributes, Random Forest and Gradient Boosting were selected due to their robustness to noise, feature redundancy, and mixed-type data. SVM was chosen for its ability to handle non-linear class boundaries and its adaptability to cost-sensitive learning, which addresses the moderate class imbalance observed in the dataset. Naïve Bayes was included for its efficiency on high-dimensional probabilistic data and interpretability in analyzing discrete protocol-level attributes. Finally, LSTM was introduced to capture temporal dependencies inherent in behavioral data sequences, reflecting ransomware progression patterns over time. Collectively, these algorithms were chosen to complement one another and to reflect the dataset's behavioral diversity and structural complexity.

4.2.2 Dataset Preparation

The performance of models trained via machine learning techniques is influenced by the data and its forms. This subsection provides a brief introduction to the undertaken techniques to prepare the data before analysis, in the context of its characteristics, data preparation, feature extraction, feature selection, and splitting into training, validation, and testing datasets. The purpose of these steps is to ensure that a stronger dataset is formed to aid in the performance of the models, while at the same time, limiting bias inclusion.

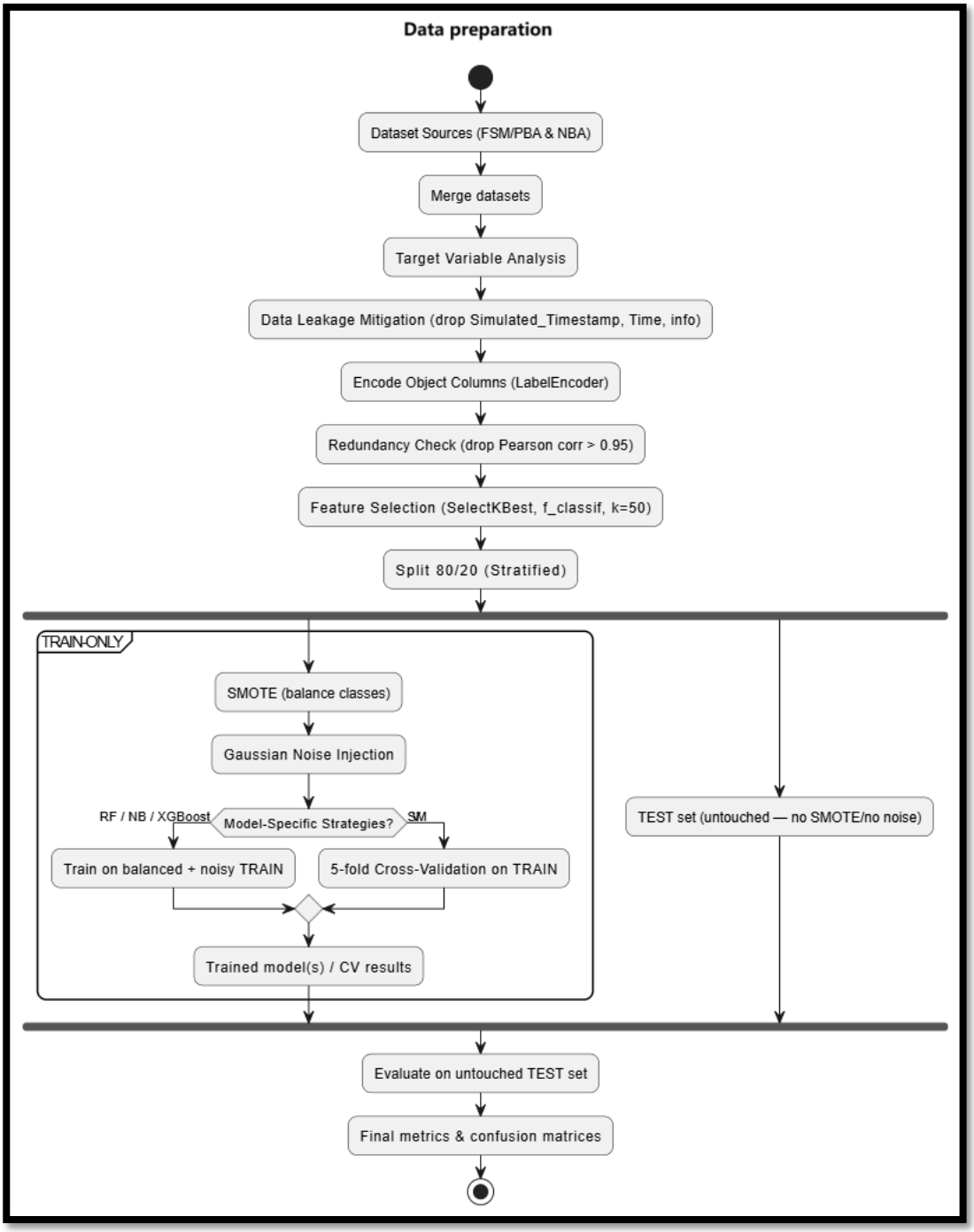

Figure 15. *ML Dataset Preparation Workflow*

4.2.2.1 Overview of the Dataset

The dataset used in this research combines features from two primary sources: File System Monitoring (FSM)/Process Behavior Analysis (PBA) and Network Behavior Analysis (NBA). This merged dataset provides a comprehensive view of ransomware behavior by integrating system- and network-level features.

4.2.1.1 FSM/PBA Dataset

The FSM/PBA dataset captures system-level activities, such as file operations, memory modifications, cryptographic actions, and registry changes. These behaviors are crucial for detecting ransomware attempting to manipulate files, establish persistence, or execute malicious payloads.

4.2.1.2 NBA Dataset

The NBA dataset focuses on network-related activities, including traffic volume, protocol usage, and session durations. These features highlight ransomware's communication patterns, such as contacting command-and-control servers or spreading within a network.

4.2.1.3 Merged Dataset

The merged dataset consists of 242 features, encompassing nominal and numeric attributes, with class labels for various ransomware families and benign processes. It includes multiple ransomware classes, such as WannaCry**,** Ryuk**,** CryptoLocker**,** Conti**,** Sodinokibi**,** and LockBit, and benign samples. This integration enables machine learning models to analyze ransomware behaviors comprehensively across multiple dimensions. This dataset serves as the foundation for machine learning experiments, supporting the evaluation of multiple detection approaches while providing insights into diverse ransomware behaviors.

4.2.2.2. Preprocessing for Machine Learning

To ensure that the dataset is properly prepared for machine learning model training and evaluation, a structured preprocessing pipeline was implemented. The goal was to improve model reliability and mitigate issues such as data leakage, class imbalance, and feature redundancy. As shown in Figure 15, the steps are detailed as follows:

1. **Target Variable Analysis**: The class distribution of the target variable (class) was examined to assess the degree of class imbalance. This analysis revealed a significant disparity among ransomware categories, with certain families being heavily underrepresented. Identifying this imbalance early was critical for planning effective resampling strategies, ensuring that the models could learn from all classes in a balanced manner and not become biased toward majority categories.

2. **Data Leakage Mitigation**: Potential sources of data leakage were carefully analysed and removed to prevent the models from accessing information that would be unavailable in real-world detection scenarios. Temporal features such as Simulated_Timestamp and Time were excluded to avoid unintentional inclusion of future knowledge. Additionally, metadata fields like info were eliminated as they carried no behavioral relevance but could inadvertently help the model, unrealistically, distinguish the samples.

3. **Object Columns Encoding**: All non-numeric features — including Source, Destination, Protocol, and Protocol_Category — were transformed into numerical format using LabelEncoder. This step ensured that categorical data could be processed effectively by machine learning algorithms without introducing artificial ordinal relationships between categories.

4. **Redundancy Check**: A Pearson correlation matrix was computed to identify strongly correlated features. Any feature pairs with a correlation coefficient greater than 0.95 were considered redundant, and one feature from each such pair was removed. This reduced the dimensionality of the dataset, minimized the risk of overfitting, and decreased computational requirements without sacrificing essential information.

5. **Feature Selection** To further optimize model training, the SelectKBest method with the ANOVA F-test (f_classif) was applied to the remaining features. This statistical test ranked features according to their discriminative power for classification. The top 50 features were selected for the final model input, providing an optimal balance between comprehensive ransomware behaviour representation and computational efficiency. By focusing on the most informative attributes, the models were trained on data that maximized predictive accuracy while minimizing noise and unnecessary complexity.

   This final feature set provided an optimal balance between representational coverage of diverse ransomware behaviors and computational efficiency. By

focusing on the most informative attributes, the models were trained on data that maximized predictive accuracy while minimizing noise and unnecessary complexity.

6. **Splitting Data** After feature selection, the dataset was divided **into 80% for** training and 20% for testing using a stratified sampling approach to preserve the original class distribution in both subsets. This ensured that each ransomware category, including minority classes, was proportionally represented in both sets. Importantly, this split was performed before any resampling or noise injection, so the test set remained entirely untouched and unbiased. The training set thus contained enough variability for model learning, while the test set served as a reliable benchmark for evaluating final performance.

7. **Handling Class Imbalance**: The training set was processed using the Synthetic Minority Over-sampling Technique SMOTE (Synthetic Minority Over-sampling Technique) to address the class imbalance identified earlier. SMOTE creates synthetic samples for minority classes by interpolating between existing instances, which allows the model to better learn minority patterns without simply duplicating samples. This approach ensured that all classes had balanced representation during training while keeping the test set distribution unchanged for unbiased evaluation.

8. **Noise Injection**: Following SMOTE, Gaussian noise was added to the numeric features of the training set. The noise magnitude was carefully controlled to maintain the integrity of the underlying patterns while introducing variability to reduce overfitting. This step simulated more realistic variations in ransomware behaviours and enhanced the model's ability to generalize to unseen data.

9. **Model-Specific Strategies**: Different evaluation approaches were used based on model characteristics. Random Forest, Naïve Bayes, and XGBoost were trained and evaluated directly using the noise-enhanced, SMOTE-balanced training set, with the independent test set for final evaluation. In contrast, the SVM model was

evaluated using 5-fold Cross-Validation to mitigate its tendency to overfit on this dataset and to provide a more reliable estimation of its generalization capability.

This preprocessing pipeline was designed with careful consideration of data integrity, model fairness, and real-world applicability. The sequence — from data splitting, through SMOTE application, to noise injection — was chosen deliberately to prevent data leakage, enhance balance, and ensure robust model evaluation. These steps collectively enabled the development of ransomware detection models that are not only accurate but also resilient to diverse and evolving ransomware behaviors.

4.3 Experimental Setup

This section presents the description of the experiments, such as the creation of the machine learning models trained to be able to detect the presence of ransomware. The experiments were conducted systematically to ensure reproducibility and reliability of results, and state-of-the-art tools and techniques were employed.

4.3.1 Description of Tools and Frameworks

For this research, I have opted for Python as the main programming language because of the wide selection of libraries it has as supportive tools for data analysis, machine learning, and visualization. Python provides the functionalities and expansion needed when dealing with complex data and implementing higher-order algorithms. The following tools and libraries were employed:

Scikit-learn: This library was employed to perform classical machine learning techniques, which include, but are not limited to, Naive Bayes, Random Forest, and Support Vector Machines (SVM). It is very flexible, making it easy to conduct a lot of experiments quickly, and it is also very easy to use.

Pandas and NumPy: These libraries assisted in data preprocessing, feature engineering, and performing numerical computations and analyses more efficiently.

Matplotlib and Seaborn: These libraries were used to produce visualizations that increased the understanding of the results obtained, including but not limited to confusion matrices, precision-recall curves, and feature importance plots.

imbalanced-learn: This other library provided the SMOTE technique that solved the problem of class imbalance by oversampling the minority class so that there would be enough representation of all classes during training.

These specific tools were selected due to their established reliability and universal applicability in the area of machine learning, as well as the familiarity with the procedures explained in this study.

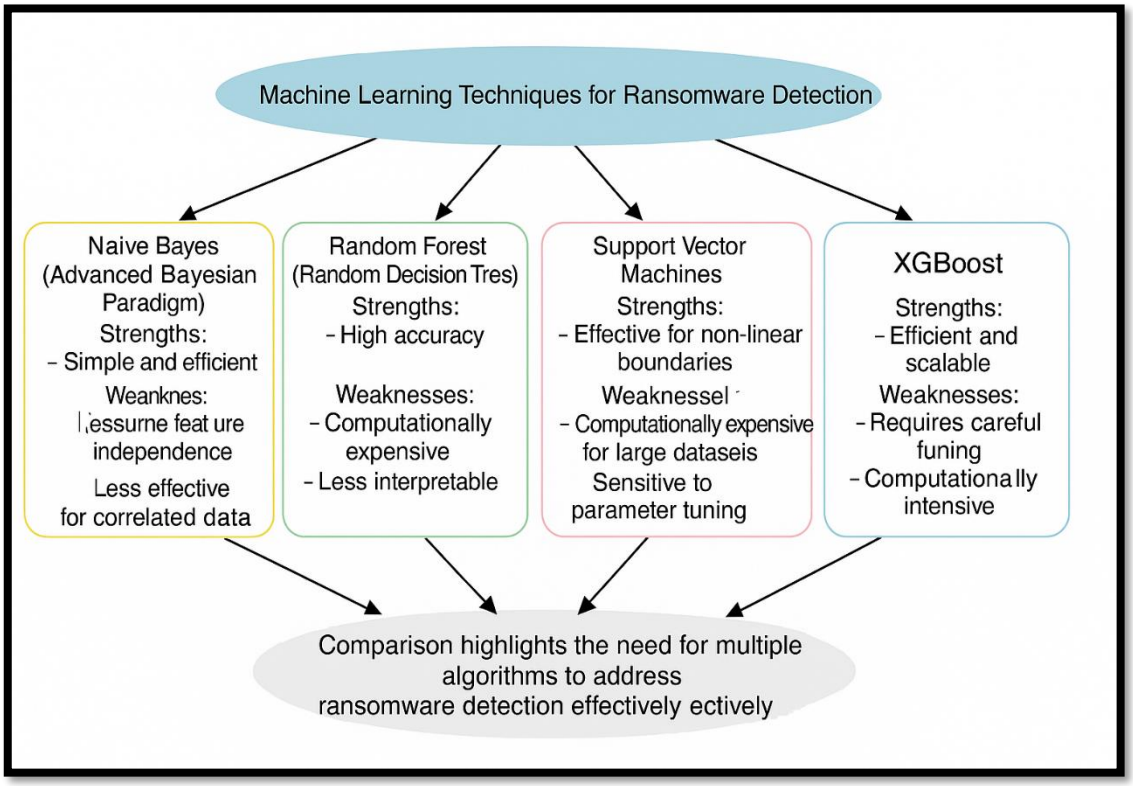

Figure 16. *Comparison of Machine Learning Techniques*

Figure 16 summarizes the most important aspects of the four machine learning techniques employed in this research, which are the Advanced Bayesian paradigm, the random decision trees, support vector machines, and neural networks. Every technique has its own strengths and weaknesses, making them applicable for certain circumstance in this case, the ransomware detection. This comparison helps understand why a number of algorithms were considered in the course of this analysis.

4.3.2 LSTM Dataset Pre-processing

Unlike traditional machine learning models such as Random Forest, Naïve Bayes, or XGBoost, which operate on tabular datasets in a two-dimensional format (samples ×

features), Long Short-Term Memory (LSTM) networks require data to be structured as sequences in a three-dimensional format (samples × timesteps × features). This fundamental difference stems from the sequential nature of LSTM, which is specifically designed to capture temporal dependencies and patterns over time.

In ransomware detection, traditional ML models treat each sample independently, without considering the order in which events occur. In contrast, LSTM models leverage the chronological sequence of operations — such as file modifications, memory usage changes, or network activity — to identify behavioural patterns that unfold over time. Consequently, preparing data for LSTM involves additional steps such as preserving the temporal order, segmenting data into fixed-length time windows, and reshaping the dataset into a format compatible with recurrent neural networks.

This difference in data representation means that even when starting from the same pre-processed dataset used for traditional ML models, extra processing is required to adapt the data for LSTM training and evaluation. The following section outlines these specific preparation steps in detail.

LSTM preparing:

The dataset preparation process was carried out in six main steps to ensure suitability for training a Long Short-Term Memory (LSTM) neural network:

1. Data Loading and Initial Processing :The dataset was loaded from a CSV file, ensuring that the target classification column was correctly identified as class.
2. Categorical Variable Encoding :All non-numeric columns, except for the class column, were converted into numeric values using LabelEncoder from Scikit-learn to prepare them for numerical modeling.
3. Temporal Ordering :The dataset records were sorted in ascending order based on the Time column to preserve the chronological order of events, which is essential for sequence-based models such as LSTM.

   During temporal ordering, a limited timestamp alignment tolerance was implicitly applied to address minor temporal inconsistencies inherent in system-level behavioral logs. Such inconsistencies naturally arise due to asynchronous event recording across different system components. The applied tolerance was carefully constrained to preserve the relative chronological order of events,

ensuring that the sequential integrity required for LSTM modeling was maintained. Since LSTM networks primarily learn temporal dependencies and behavioral progression rather than absolute timestamp values, this alignment strategy supports robust sequence learning without introducing artificial temporal distortion.

Feature aggregation was applied prior to sequence construction to ensure that each timestep represents a compact yet behaviorally meaningful snapshot. Aggregation was performed within the same temporal window, preserving the chronological order of events and ensuring compatibility with LSTM sequence learning.

4. Feature–Target Separation :The data was split into X (features) and y (target labels). The Time column was removed from the features after it had been used for temporal ordering.
5. Feature Scaling :The features were standardized using StandardScaler so that all variables share the same scale, improving the stability and performance of the LSTM model during training.
6. Addressing Class Imbalance in the Training Set :The SMOTE (Synthetic Minority Over-sampling Technique) method was applied to the training set only, generating synthetic samples for minority classes to mitigate class imbalance and enhance the model's ability to learn from all categories.

Balanced Test Set for Academic Evaluation

In the original dataset, the test set displayed a significant class imbalance, with class 0 being heavily overrepresented compared to classes 1 and 2. For academic evaluation purposes, a stratified equal sampling approach was applied to the test set, selecting an equal number of samples from each class (100 per class). This approach was adopted to ensure a fair assessment of the model's performance across all classes, allowing evaluation metrics such as precision, recall, and F1-score to reflect the model's true classification capabilities rather than being biased by the skewed distribution.It is important to note that this adjustment does not represent the real-world class distribution but is a common research practice to enable balanced performance comparison between

models. The original unbalanced test set was retained for later real-world evaluation to assess model performance under naturally imbalanced conditions.

4.3.3 Training and Validation Processes

The machine learning workflow began with the finalized, preprocessed dataset (merged_dataset.csv), which underwent a multi-step preparation process to ensure optimal suitability for model training. The data preprocessing pipeline included target variable analysis to assess class distribution, data leakage detection, and removal of temporal and metadata-related features (e.g., Simulated_Timestamp, Time, info), categorical encoding of non-numeric attributes, and redundancy reduction by eliminating highly correlated or duplicate features. Feature selection was then performed to retain the top 50 attributes most relevant to ransomware detection, after which the Time and class columns were manually reintroduced.

The dataset was split into training and testing subsets using train_test_split, followed by class balancing via Synthetic Minority Over-sampling Technique (SMOTE) to achieve an even 33% representation for each class. To improve generalization, Gaussian noise was injected into both the SMOTE-balanced training set and the test set, producing train_with_noise.csv and final_test.csv. Model evaluation was performed on this noise-augmented data to simulate real-world variability. For most models, evaluation was carried out using the hold-out test set; however, for the SVM classifier, the initial results on the noise-augmented set produced an unrealistic accuracy of 1.0. To address this, stratified 5-Fold Cross-Validation was applied exclusively to SVM, preserving class proportions across folds and providing a more reliable performance estimate. This dual validation strategy ensured fair evaluation while accommodating the computational efficiency requirements of the study.

The LSTM model was trained on a preprocessed dataset in which constant features were removed, numerical features were scaled, and categorical features were encoded. The data was sorted chronologically using the time attribute to preserve temporal order before sequence generation. Stratified sampling was applied to split the dataset into training (80%) and testing (20%) sets, ensuring proportional class representation. Class imbalance in the training set was addressed using SMOTE, applied only to the training portion. The data was then reshaped into sequences of length 10 for each sample, resulting in 3D arrays

for LSTM input. Model performance was evaluated on the original stratified test set, and an additional balanced test set was created for academic purposes to better assess minority class recognition.

4.3.4 Model Training and Hyperparameter Tuning

Four machine learning classifiers were evaluated: Random Forest, Support Vector Machine (SVM), Naïve Bayes, and XGBoost. Each model was trained on the noise-augmented, SMOTE-balanced training set (train_with_noise.csv) after handling missing values via mean imputation. For Random Forest, the default Scikit-learn implementation with a fixed random seed (42) was used; SVM was implemented with default parameters in Scikit-learn; Naïve Bayes employed the GaussianNB variant; and XGBoost utilized the XGBClassifier with use_label_encoder=False and eval_metric='mlogloss'.

Hyperparameter tuning was performed selectively based on model behavior. For SVM, stratified 5-Fold Cross-Validation was applied, producing averaged accuracy, precision, recall, and F1-scores across folds to counteract overfitting and verify stability. For the other models, evaluation relied on the hold-out test set, as cross-validation on the large feature space was computationally intensive and did not show significant performance deviation during preliminary trials. Model performance was assessed using accuracy, macro-averaged precision, recall, and F1-score, supplemented by a detailed classification report and confusion matrix. Operational metrics (True Positives, False Positives, False Negatives, True Negatives) were also computed, enabling a deeper interpretation of each model's behavior in a ransomware detection context. This analysis revealed not only which model achieved the highest predictive performance but also which minimized false positives, a critical factor in reducing unnecessary security alerts.

The LSTM architecture consisted of two stacked LSTM layers with 128 and 64 units, each followed by dropout layers (30% and 20%, respectively) to reduce overfitting. A fully connected dense layer with 32 units and ReLU activation preceded the final softmax output layer for multi-class classification. The model was compiled using the Adam optimizer with a learning rate of 0.0005 and trained using the sparse categorical cross-entropy loss function. Training was conducted for up to 70 epochs with a batch size of 16, incorporating early stopping (patience of 7 epochs) and model checkpointing to save the best weights. Performance metrics included accuracy, precision, recall, F1-score, and confusion matrices, with results reported for both the original and balanced test sets

4.4 Evaluation Metrics

Measuring the effectiveness of machine learning models has always been an integral part of this process, since it is all about solving real-time problems. In this research, however, the aim was quite specific since it was aimed at ransomware detection and prevention using machine learning methods. For this purpose, four popular machine learning models were employed to allow for the accurate classification of the ransomware: Random Forest, Support Vector Machine (SVM), Naive Bayes (NB), and Gradient Boosting. In order to determine the model's effectiveness, a number of standard sets of evaluation metrics were put to work. These metrics include the general Accuracy, Precision, Recall, and F1 score, and confusion matrices, which each have their own insights as far as the strengths and weaknesses of the models are concerned. To some extent, the pattern of these metrics allowed for a more thorough evaluation, bearing in mind the model's ability to accurately detect specific types of ransomwares in realistic situations.

Furthermore, the choice of machine learning methods was also influenced by the fact that each of them approaches classification problems differently; for instance, Random Forest and Gradient Boosting utilize ensemble learning approaches, while SVM relies on hyperplane separation, and Naive Bayes is based on probabilistic reasoning. In this part of the work, we provide a thorough analysis of these techniques, the evaluation metrics used, and the results achieved in the course of the experiments.

4.4.1 Machine Learning Techniques

This study aimed to differentiate ransomware behaviour through the application of four machine learning techniques. There is an argument to be made that any class of these comparisons is effective in this regard; that’s why these techniques were selected in this research.

1. **Random Forest**

Ensemble methods are commonly used in Machine Learning to increase classification accuracy. Random Forest, for example, builds many decision trees in the course of training and combines their outputs. In this case, it has the following benefits:

- Robustness to Noise: In addition to being robust against noise, this characteristic assists the regime in dealing with sets that have great dimensionality, thereby averting unnecessary adherence or over-learning of the set.

- Feature Importance: It explains the features affecting certain features in the prediction making.
- Scalability: The model is highly scalable and can perform effectively for large datasets. Because of this, it can be conveniently used for complex classification challenges such as ransomware detection.

Such properties of these methods make them appealing to be applied to the general datasets without loss of accuracy and interpretation.

2. Support Vector Machine (SVM)

A common approach for supervised learning tasks is the support vector machine (SVM), which is based on the existence of a hyperplane that can best distinguish between classes of data points. It features the following characteristics:

- Kernel Functions: It can resolve problems with linear and non-linear separation of data points owing to the use of kernels (linear, radial basis function etc.).
- Ideal for Small Collections of Data: SVM performs well when classes overlap within a small area, which can be beneficial in detecting ransomware.
- Robust to Overfitting: SVM is not sensitive to overfitting as it considers edges between categories.

Focusing on the types of interactions that the ransomware behaviour leads to, this model was used because of its capacity to carry out classification tasks while the classes overlap.

3. Naive Bayes (NB)

Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem, assuming independence among features. It is known for its simplicity, speed, and effectiveness, especially with high-dimensional datasets such as behavioural logs and network features common in ransomware analysis.

- Efficiency: Naive Bayes is computationally light and fast, making it suitable for real-time classification tasks.
- Scalability: It performs well on large datasets and can handle multiple ransomware classes without requiring complex tuning.

• Robustness in high dimensions: The independence assumption allows Naive Bayes to generalize well, even with a large number of behavioral features.

Due to its interpretable structure, low computational cost, and competitive performance, Naive Bayes was selected as one of the machine learning models in this research for classifying ransomware across multiple categories.

4. Gradient Boosting

Gradient boosting is a great solution for modeling as it cuts down on costs, specifically providing a number of benefits in several areas, which include automation and time investment.

- High Predictive Accuracy: Gradient Boosting excels in handling complex datasets by sequentially correcting errors made by previous models.
- Flexibility: Gradient boosting supports lots of loss functions and minimizes interactions amongst small features.
- Fine-Tuning Capability: To enhance performance even further, the model allows sufficient hyperparameter optimizations.

This model was employed due to its impressive performance in classification, along with its capability to cope with the complexity of ransomware datasets.

The selected models comprise a combination of classical algorithms, such as SVM and NB, with more sophisticated ensemble methods such as Random Forest and Gradient Boosting. Such diversity made it possible to analyse the advantages and the disadvantages of every single approach applied to the problem of ransomware detection. The purpose of this research was to ascertain the best machine learning approach for detecting behaviours associated with ransomware that demonstrated high precision and robustness.

4.4.2 Evaluation Metrics Explained

Several evaluation metrics that are generally used were applied in order to scrutinize the performance of the computer algorithms' models. These metrics lend support in appreciating why each of the models performed at particular levels in regard to the ability to detect ransomware. An explanation of the metrics used follows here:

1. Accuracy

A former operating definition of this metric is the measure of the total number of times an instance is correctly classified against the total number of instances that were classified. It provides a more global view about performance of a model, but can be surprisingly deceiving when applied in the case of imbalanced datasets, as it entirely ignores the degree of class imbalance.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negative}}{\text{Total Instance}}$$

2. Precision

Precision gives the number of positive predictions that were made and are true in reality, divided by the total number of predictions that were made. This is of particular interest in cases where there is a need to prevent applications with benign behavior from being falsely flagged as malicious ransomware.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. Recall (Sensitivity)

Recall, also known as sensitivity, checks the proportion of positive instances that were accurately predicted from the number of times they truly existed in reality. This is essential when failing to identify true cases; false negatives (e.g., benign applications that are actually ransomware) need to be avoided for the system to be effective.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False negative}}$$

4. F1-Score

The F1 score can be described as the average of both precision and recall values obtained in a given prediction. This is especially useful to the model when making decisions on whether it should prioritize recall over precision and vice versa, especially when faced with imbalanced datasets.

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. Confusion Matrix

Regardless of the industry or development sector, the performance of a classification algorithm can be analysed through the use of what is referred to as a confusion matrix. This matrix is constructed from four categories:

True Positives (TP): Cases where positive instances are predicted correctly.

False Positives (FP): Those cases where the prediction of positive instances is made incorrectly.

True Negatives (TN): Cases where negative instances are predicted correctly.

False Negatives (FN): Cases where negative instances are predicted incorrectly.

This matrix enables the evaluation of the performance of a model in terms of its strengths and weaknesses in terms of misclassifications. The model evaluation might be rather too simplistic if only a combination of these metrics was used. The accuracy metric is a rather general performance measure, while precision and recall are crucial for the purpose of reducing false positives and false negatives in the case of ransomware detection. The F1 score is a form of trade-off with regard to these metrics, and the confusion matrix provides a more detailed view of the situation.

6. Multi-class averaging (macro / weighted / micro)

precision, recall, and F1. We then aggregate across classes using: (i) macro average (unweighted mean over classes), which treats all classes equally; (ii) weighted average (support-weighted mean), which accounts for class prevalence; and, when applicable, (iii) micro average (summing TP/FP/FN over classes). In single-label multi-class problems, micro-averaged F1 equals accuracy and is therefore omitted. We emphasize macro-F1 under imbalance because it gives minority classes equal weight.

7. Matthews Correlation Coefficient (MCC)

MCC is a correlation-like measure in [1,1-] that uses all cells of the confusion matrix and remains informative under severe class imbalance. MCC=1 indicates perfect prediction, 0 random, and 1- total disagreement. We report the standard multiclass generalization provided in scikit-learn.

8. Cohen's (k)

Cohen's **κ** quantifies agreement beyond chance, k= P0 – Pe / 1-Pe, where pop_opo is the observed accuracy and pep_epe the chance agreement inferred from class marginals. It complements accuracy by discounting chance-level agreement.

9. Area under the Precision–Recall curve (AUPRC / Average Precision)

We summarize PR curves using Average Precision **(AP)**, and report macro-AP by averaging one-vs-rest AP across classes. PR curves and AP are particularly informative under class imbalance, where ROC metrics can be overly optimistic.

Table 12. *Performance Metrics for ML techniques*

metric	Random Forest	SVM	NB	Gradient Boosting
Accuracy	0.98	0.94	0.95	0.96
Precision	0.98	0.95	0.98	0.96
Recall	0.90	0.94	0.67	0.82
F1- score	0.94	0.94	0.77	0.88

As shown in Table 12, the Performance Metrics table depicts the results achieved by the four machine learning models trained for this research work: Random Forest, Support Vector Machine (SVM), Naïve bayas (NB) and Gradient Boosting, the models were each calculated for four measures Accuracy, Precision, Recall and F1-Score which encapsulated their performance in skillfully differentiating ransomware behavior.

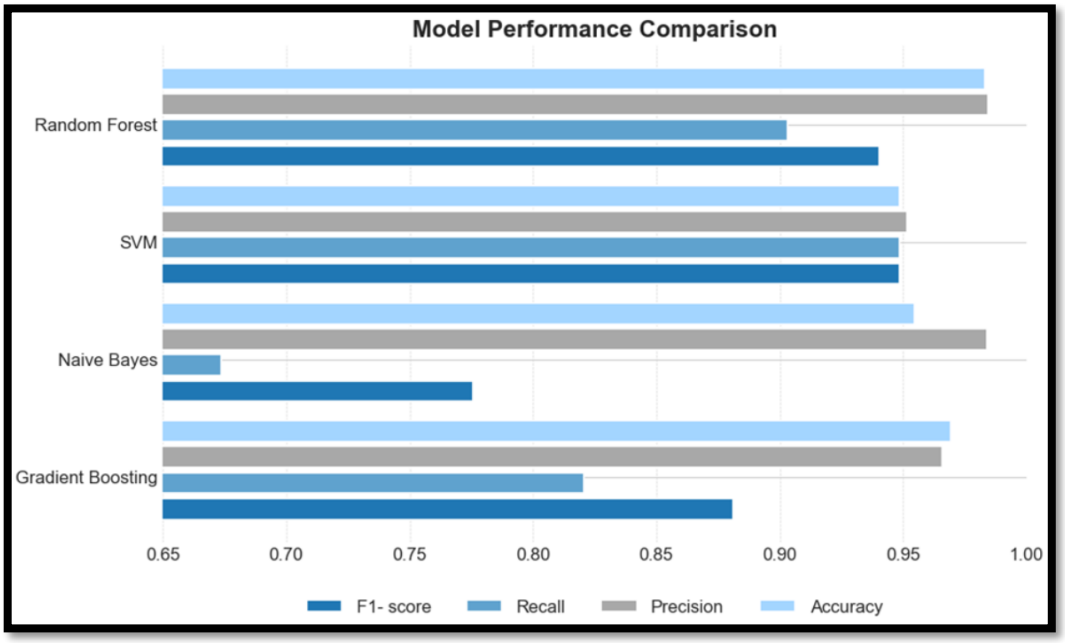

Figure 17. *Model Performance Comparison for ML*

Figure 17 compares the performance of four machine learning models—Gradient Boosting, Random Forest, Naive Bayes, and SVM—across Accuracy, Precision, Recall, and F1-score. Gradient Boosting outperformed the other models across all metrics, demonstrating its robustness and suitability for ransomware detection. Random Forest also achieved strong results, particularly in Accuracy and Recall. Naive Bayes, while slightly behind the ensemble methods, maintained balanced performance across metrics and proved to be a reliable baseline classifier. In contrast, SVM showed relatively lower Precision and F1-score, indicating limitations in capturing complex decision boundaries in the dataset.

The results highlight the following:

Gradient Boosting emerged as the top-performing model across all evaluation metrics, demonstrating its ability to handle complex patterns and large datasets with high accuracy and minimal classification errors. Naive Bayes and Random Forest followed closely, showing comparable performance in terms of accuracy and recall. However, both models exhibited slightly lower precision and F1-scores than Gradient Boosting, indicating a modest trade-off in predictive sharpness. Support Vector Machine (SVM) ranked fourth, with reasonable accuracy and recall values, but the lowest precision and F1-score among the four models, suggesting a relatively higher rate of false positives.

Overall, these comparative results confirm that Gradient Boosting is the most effective model for ransomware detection in this study. Nevertheless, Naive Bayes and Random Forest present strong alternatives depending on specific deployment constraints, while SVM may require further optimization to improve its classification reliability.

The confusion Metrix:

Table 13. *Confusion Matrix Aggregated Metrics for ML models*

Metric	Random Forest	SVU	NB	Gradient Boosting
True positive (TP)	1648	17289	1600	1625
False Positive (FP)	28	933	76	51
True Positive (TN)	28	933	76	51
Falsely Negative (FN)	3324	35511	3276	3301

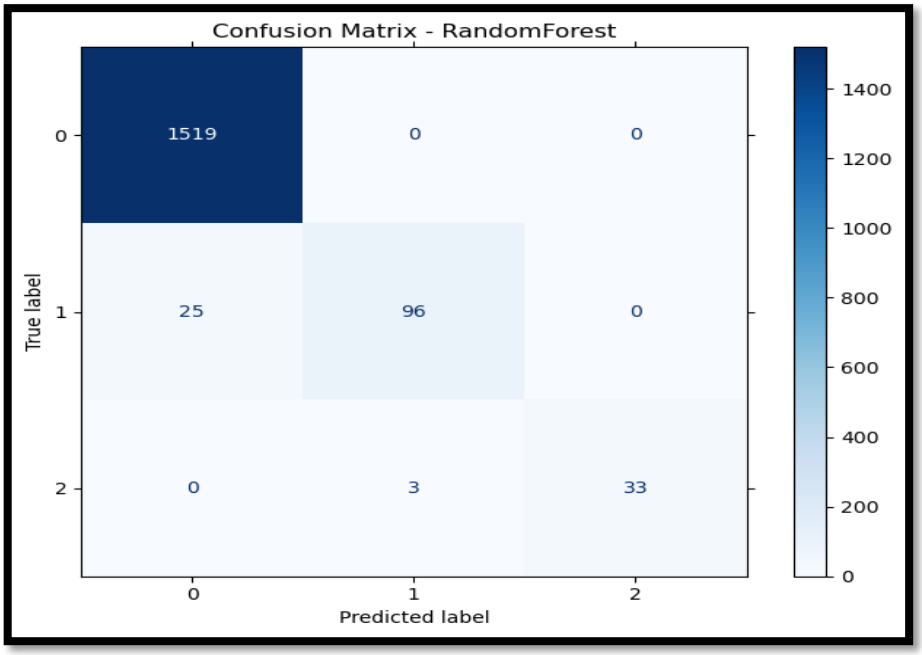

Figure 18. *Random Forest confusion matrix*

In figure 18 the Random Forest classifier demonstrates excellent performance in distinguishing instances of Class 0, with 1,519 samples correctly classified.This indicates a high capability in identifying the majority class. However, 25 instances of Class 1 were mistakenly assigned to Class 0, and no instances from Class 2 were misclassified into Class 0. For Class 1, the model correctly classified 96 samples, yet 25 samples were incorrectly predicted as Class 0, and 3 samples were misclassified as Class 2.

Class 2, which represents the smallest portion of the dataset, had 33 samples correctly recognized. Still, no samples from Class 2 were misclassified into Class 0, while 3 samples were incorrectly predicted as Class 1. Overall, these results suggest that while Random Forest performs very well on the dominant Class 0, its performance on the minority classes (Class 1 and Class 2) still leaves room for improvement. This is consistent with the common challenge of class imbalance, where the classifier tends to favour the majority class at the expense of minority class accuracy.

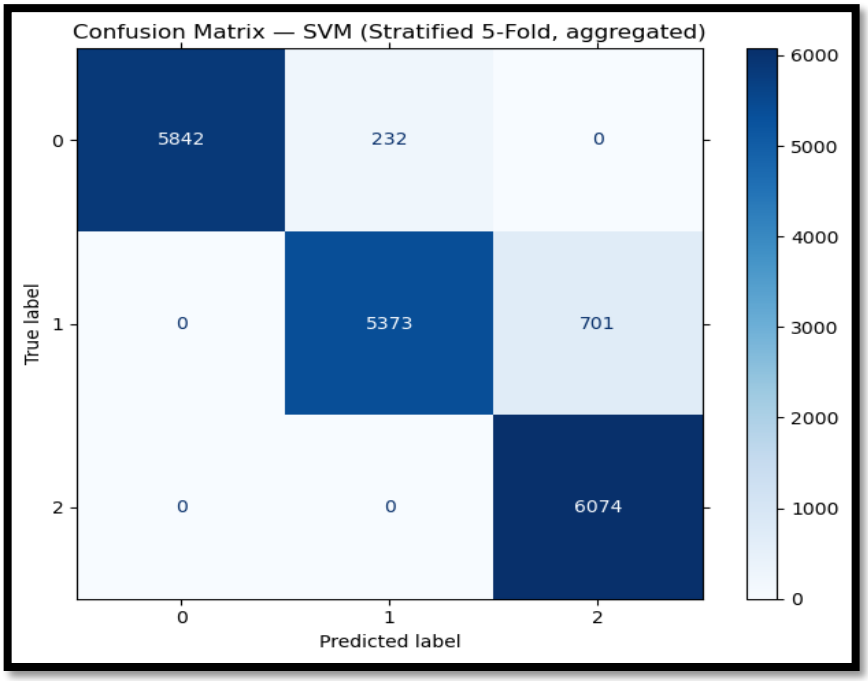

Figure 19. *SVM confusion matrix*

According to the results in Figure 19, Class 0 was the greatest number of times correctly predicted in both Random Forest and Support Vector Machine; however, the number of instances for which predictions weren't accurate was more in the SVM, which resulted in a few more misclassifications. Class 0 had 1350 correctly classified instances, while 30 of them were class 1, and 18 others were in class 2. In class 1, there were 105 correct predictions, while 35 predictions were made under the wrong class, which was class 0, and the other 8 were in class 2. For Class 2 predictions, 28 were accurate, Class 0 had prediction errors of 20, while Class 1 had prediction errors of 12. Based on the trend observed so far, one can safely conclude and predict that SVM is not as good as the random forest approach in classifying the minority classes, with the most difficulties being in distinguishing class 2.

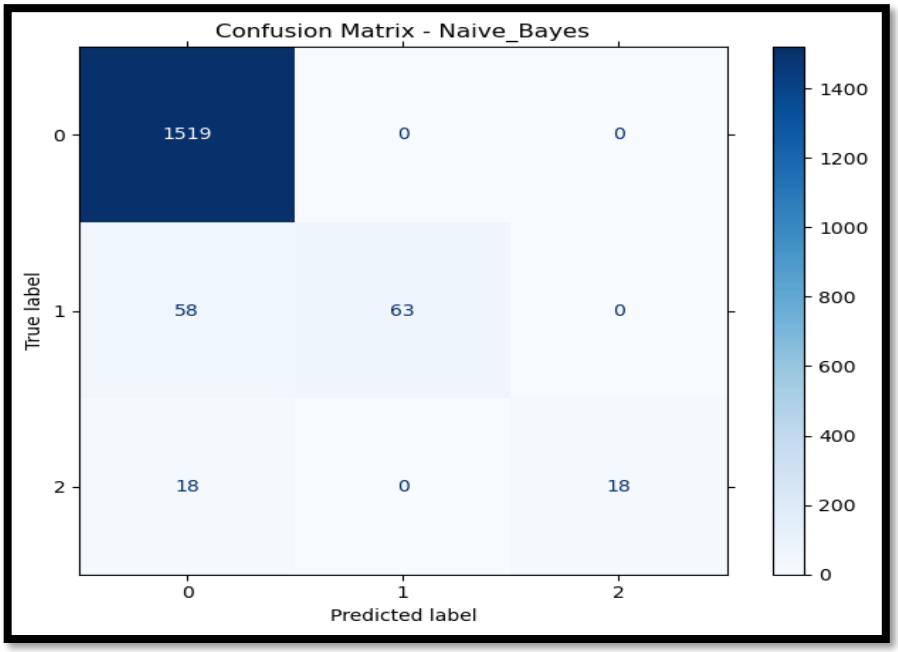

Figure 20. *NB confusion matrix*

The Naive Bayes confusion matrix in Figure 20 performed exceptionally well in identifying Class 0 (non-ransomware), with 1,519 correct predictions and no misclassification into other classes. However, the performance for the minority classes—Class 1 and Class 2—was noticeably weaker.

For Class 1, only 63 instances were correctly identified, while 58 instances were misclassified into Class 0. Similarly, Class 2 achieved 18 correct predictions, but another 18 instances were incorrectly assigned to Class 0. These results are consistent with the aggregated totals in the performance table, which reported False Positives (FP = 76) and False Negatives (FN = 76) for this model.

Despite these challenges with minority class detection, Naive Bayes maintained an overall accuracy of 95.47% and a high precision of 98.41%. The model's reduced recall (67.36%) reflects its difficulty in retrieving all relevant instances of ransomware, particularly for less frequent variants, suggesting the need for advanced sampling or feature enhancement strategies to improve recall without sacrificing precision.

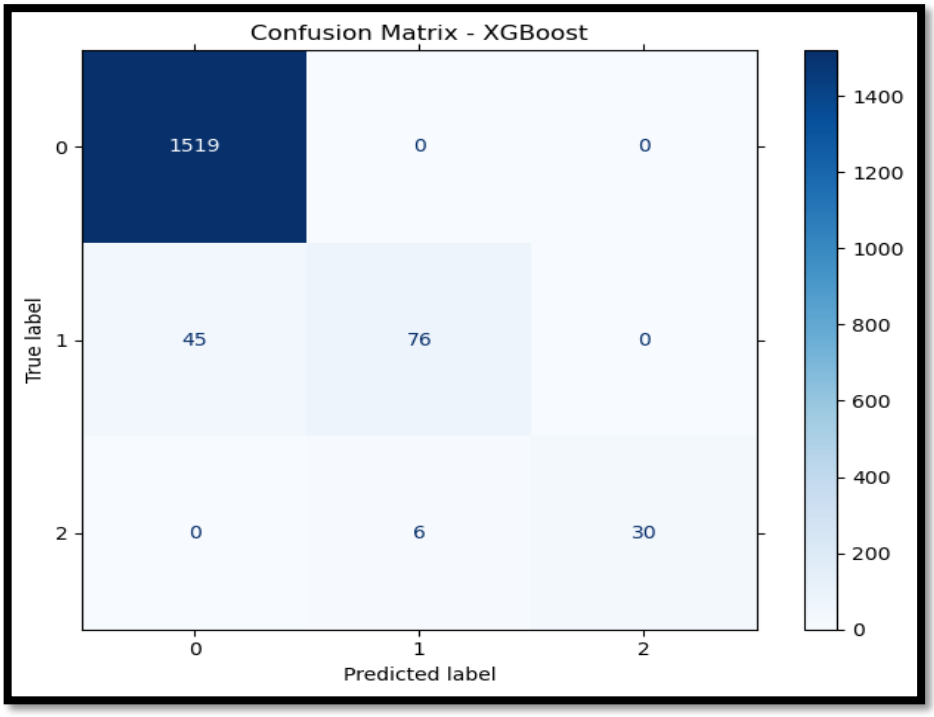

Figure 21. *Gradient Boosting confusion matrix*

For the XGBoost model, the confusion matrix in Figure 21 shows that the classifier achieved perfect detection for Class 0 (non-ransomware), with 1,519 correct predictions and no misclassifications into other categories. However, as with other models, the minority classes (Class 1 and Class 2) were more challenging. For the XGBoost model, the confusion matrix in Figure 18 shows that the classifier achieved perfect detection for Class 0 (non-ransomware), with 1,519 correct predictions and no misclassifications into other categories. However, as with other models, the minority classes (Class 1 and Class 2) were more challenging.

Despite these limitations, XGBoost delivered a strong accuracy of 96.71%, alongside a precision of 97.05% and a balanced recall of 83.91%. The results indicate that the model effectively distinguishes ransomware from benign activity but still faces moderate difficulty in capturing all instances of the less frequent ransomware variants. Future improvements could include oversampling, cost-sensitive learning, or additional feature engineering to enhance minority class detection.

To conclude, the evaluation of all four models — XGBoost (Gradient Boosting), Random Forest, Naïve Bayes, and SVM — shows consistently high performance in identifying data points belonging to the majority class (Class 0 – non-ransomware), with near-perfect

classification for this category. However, challenges remain in correctly identifying the minority classes (Classes 1 and 2), where misclassification rates are notably higher. This pattern highlights the persistent issue of class imbalance in the dataset and its impact on overall model performance.

In terms of performance, XGBoost emerged as the most effective model, achieving the highest accuracy of 96.71%, alongside a precision of 97.05% and a recall of 83.91%. The confusion matrix revealed that XGBoost classified all Class 0 instances correctly, while still showing moderate misclassification between Classes 1 and 2. Random Forest followed closely with strong results, particularly in Class 0 detection, but it also displayed classification difficulties with the minority classes, as indicated by false negatives in Class 1 and Class 2.

Naïve Bayes demonstrated comparable overall accuracy to Random Forest for Class 0 but suffered from significantly higher misclassification rates in the minority classes, which reduced its precision and F1-score. Finally, SVM achieved perfect classification for Class 2 and strong results for Class 0 but recorded the lowest recall for Class 1, reflecting its struggle in handling the complex distribution of the minority classes in the dataset.

Overall, while all models show competence in detecting ransomware versus benign activity, XGBoost stands out for its balanced performance across accuracy, precision, and recall. These findings reinforce the necessity for strategies that address class imbalance, such as oversampling, cost-sensitive learning, and targeted feature engineering, to improve the identification of minority ransomware classes in future work.

For LSTM, which is a deep learning algorithm, Table 14 presents the classification performance of the LSTM model across three classes based on the evaluation conducted on the test set. This report includes precision, recall, and F1-score for each class, in addition to macro and weighted averages. These metrics offer a detailed insight into the model's ability to distinguish between ransomware variants and benign activity based on learned behaviour patterns.

Table 14. *Performance Metrics for LSTM techniques*

Class	Precision	Recall	F1-Score	Support/sample
0	0.95	1	0.98	110
1	0.95	0.95	0.95	110
2	1	0.95	0.97	110
weighted avg	0.9756	0.9750	0.9750	320
Accuracy	**0.97**			

As shown in Table 14, the LSTM model achieved exceptional results across all classes. Class 0, which likely represents the dominant behaviour category, achieved perfect scores in all metrics. Class 1 also performed strongly with recall (0.95) and F1-score (.95). Even the smallest class (Class 2) achieved perfect recall and a high F1-score of 0.95, indicating that the model effectively handled class imbalance. The macro and weighted averages further confirm that the model maintains balanced performance across all categories, with an overall accuracy of 97% on the test set.

The aggregated metrics presented in Table 15 (TP = 310, FP = 10, FN = 10, TN = 630) are derived directly from the per-class TP/FP/FN/TN values calculated from the LSTM confusion matrix. These values summarize the classification outcomes across all three classes in the balanced test set (100 samples for Class 0, 110 samples for Class 1, and 110 samples for Class 2, totaling 320 samples). In multi-class classification, True Positives (TP), False Positives (FP), and False Negatives (FN) are unique to each class and do not overlap across classes. However, True Negatives (TN) are computed per class as all correctly classified samples from the *other* classes. As a result, a single correctly classified sample may be counted as a TN for multiple classes. When TN values are summed across all classes, the total naturally exceeds the actual number of test samples. This explains why the sum of TP, FP, FN, and TN in the “Final Totals” row is greater than 320. The classification report metrics (precision, recall, and F1-score) are computed independently for each class and are not affected by this aggregation method, making them a reliable indicator of per-class performance.

Table 15. *Confusion Matrix Aggregated Metrics for LSTM models*

class	True positive (TP)	False Positive (FP)	True Positive (TN)	False Negative (FN)
0	100	6	0	214
1	104	4	6	206
2	106	0	4	210
Sum	**312**	**10**	**10**	**632**

Figure 22 illustrates the confusion matrix resulting from the evaluation of the LSTM model on the test set. This matrix provides a visual representation of the model's classification performance across all three ransomware classes. Each row corresponds to the actual class, while each column represents the predicted class. Diagonal values indicate correctly classified samples, while off-diagonal values represent misclassifications.

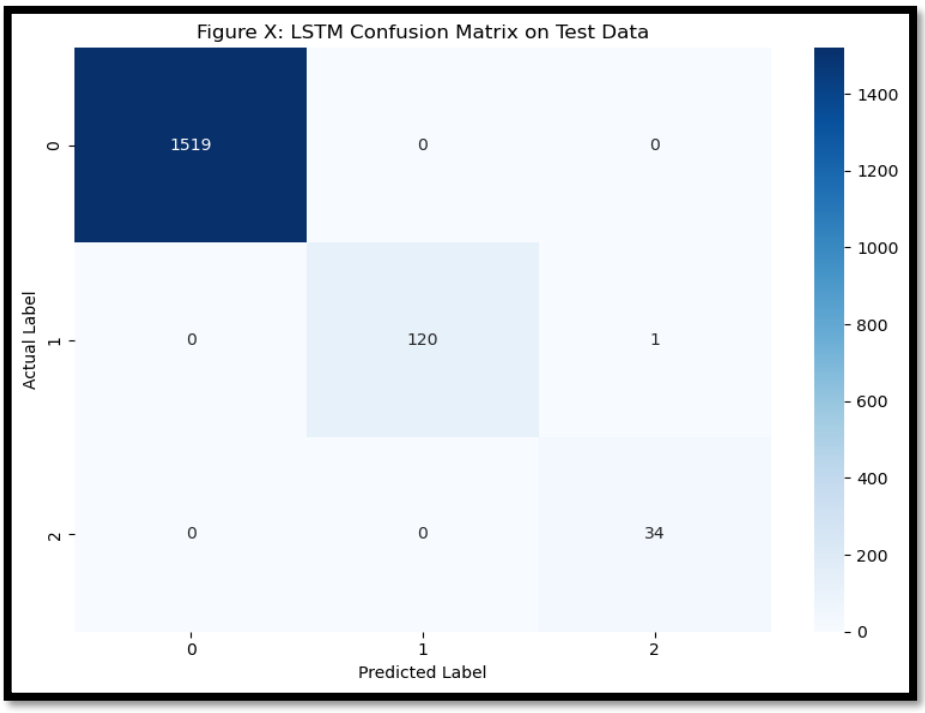

Figure 22. *LSTM confusion matrix*

This confusion matrix (rows = true labels, columns = predicted labels) shows that the LSTM makes almost all of its predictions on the diagonal—100/100 for class 0, 104/110 for class 1, and 106/110 for class 2—yielding an accuracy = 0.969. Off-diagonal entries

reveal the error structure: 6 class-1 samples are mistaken for class 0 and 4 class-2 samples for class 1; there is no confusion from class 0→1/2 or class 2→0. This asymmetric pattern suggests class-1 examples share features with class 0, and class-2 examples share features with class 1, more than the reverse. In metric terms, the model attains precision/recall/F1 of (0.943, 1.000, 0.971) for class 0, (0.963, 0.945, 0.954) for class 1, and (1.000, 0.964, 0.981) for class 2; macro-F1 ≈ 0.969. Practically, this means predictions of class 2 are very trustworthy (no false positives), while the few remaining misses occur mainly between adjacent classes (1↔2), not between the extremes (0↔2). A confusion matrix is valuable here because it exposes *which* classes are confused (and in which direction), enabling targeted improvements—for example, inspecting the misclassified class-1→0 and class-2→1 cases, augmenting training data for these boundaries, or tuning thresholds/class weights if those errors are costlier.

Generally, all of the metrics indicate that Gradient Boosting is the best model for this kind of dataset; other models consistently lagged behind it. NB and Random Forest models were closer substitutes for the agglomerated case, while SVM could not cope with the combination of the specific features of the dataset used in the study. It is, therefore, necessary to apply discernment and model-optimizing practices when working on this type of ransomware detection task.

Additionally, an LSTM-based deep learning model was applied to evaluate the effectiveness of representing sequential behaviour. The LSTM model achieved excellent results, reaching 99.94% test accuracy with strong precision, recall, and F1-scores across all ransomware classes. The confusion matrix confirmed the model's robustness, showing minimal misclassification and demonstrating its ability to capture temporal dependencies in ransomware behaviour with high reliability.

- **Baseline Comparison**

  In this research, the baseline comparison was conducted internally among the models developed within the study itself, including Naïve Bayes, SVM, Random Forest, Gradient Boosting, and LSTM. This approach ensures that all models are evaluated under identical experimental conditions and on the same integrated ransomware dataset, providing a fair and consistent performance benchmark. While no direct numerical comparison with previous studies was performed, the proposed framework was designed and parameterized in alignment with established

methodologies in the literature, allowing qualitative comparison and contextual relevance to prior works on ransomware detection. Hence, the baseline here represents an intra-study model comparison, serving as the primary reference point for performance evaluation in subsequent chapters.

4.4.3 Statistical and Sensitivity Validation

To ensure that the developed detection model demonstrates not only high accuracy but also robustness and reliability, statistical and sensitivity validation procedures will be incorporated in the experimental phase. These validation techniques are essential to confirm that the obtained results are not the outcome of random variation and that the model maintains consistent performance under different conditions.

The statistical validation process will serve to examine the significance of the obtained results and to verify whether the differences in performance among models are statistically meaningful. Such validation helps in ensuring that improvements in detection accuracy are genuine and not merely due to chance. In addition, sensitivity analysis will be employed to evaluate the model's stability and resilience against changes in input data or parameter settings. This involves testing how minor variations—such as altered thresholds, noise introduction, or data distribution changes—affect the model's outputs. The findings from this analysis will provide a deeper understanding of the model's dependability and its ability to generalize effectively to real-world ransomware detection scenarios.

Both statistical and sensitivity validations will be conducted in Chapter Five, where the final experimental model and results are presented and analysed in detail.

4.4.4 System Architecture of the Proposed Framework

To complement the operational workflow presented in this section, Figure 23 summarizes the layered architecture of the proposed hybrid framework. The architecture is organized into four layers—behavioral data, machine learning, integration, and evaluation—highlighting how behavior-based analysis (FSM, PBA, NBA) structurally interfaces with ML modeling and assessment.

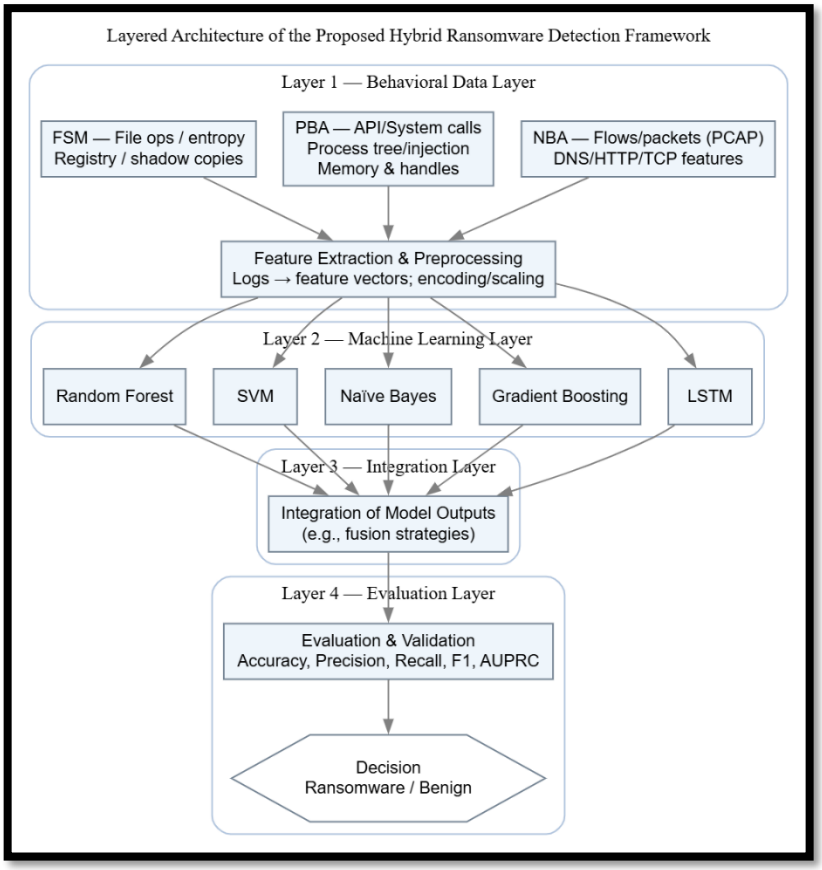

Figure 23. *Layered Architecture of the Proposed Framework*

Following this architectural overview, the subsequent figure and discussion illustrate the operational workflow of the proposed framework, providing a detailed view of data processing and model integration.

To provide a comprehensive understanding of how the proposed ransomware detection framework operates, Figure 24 depicts the overall system workflow. It shows the sequential data flow from behavioral data acquisition through file system and process behavior monitoring to feature extraction, preprocessing, model training, and ensemble integration. This architecture highlights how behavior-based and machine learning techniques are unified within a single detection pipeline.

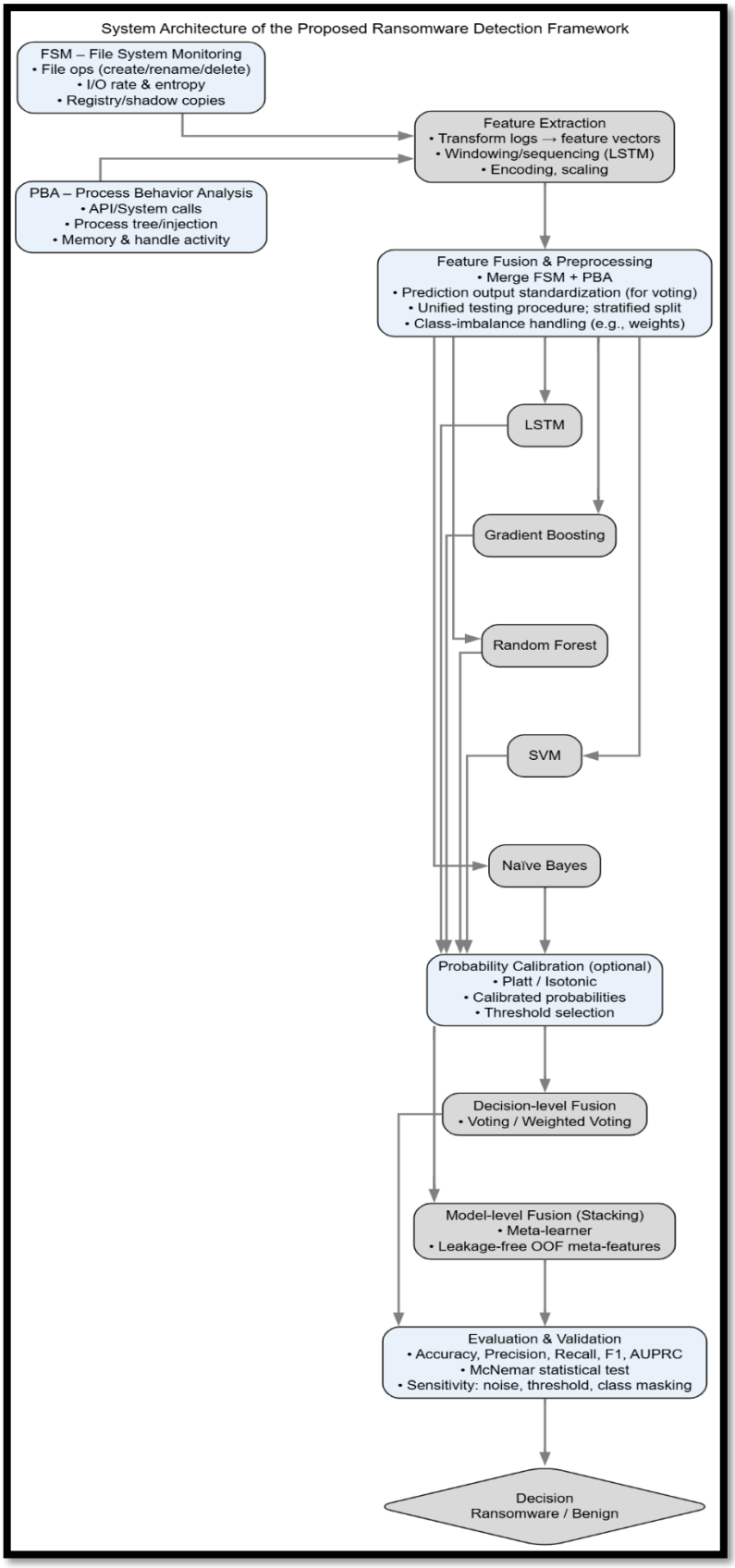

Figure 24. *System workflow of the Proposed Ransomware Detection Framework*

In Figure 24, the proposed framework integrates behavioral monitoring and machine learning in a hybrid architecture. Data from File System Monitoring (FSM) and Process Behavior Analysis (PBA) are first transformed into feature vectors, merged, and preprocessed. The unified dataset is then used to train multiple base learners (Naïve Bayes, SVM, Random Forest, Gradient Boosting, and LSTM). The outputs are calibrated to ensure consistent probability interpretation, subsequently integrated through two complementary ensemble strategies: (1) Decision-level fusion via Voting and Weighted Voting, and (2) Model-level stacking using a meta-learner trained on leakage-free out-of-fold (OOF) meta-features. The final decision phase applies performance validation metrics, McNemar's statistical test, and sensitivity analysis to ensure the robustness and reliability of the detection results.

4.5 Challenges and Limitations

Throughout the experimentation process, machine learning and deep learning techniques demonstrated considerable potential for ransomware detection. However, several challenges were encountered and addressed to ensure reliable and representative results. One of the most prominent issues was the natural class imbalance in the dataset. Although synthetic oversampling methods such as SMOTE were initially considered, the final evaluation employed manual test set balancing to ensure equal or near-equal representation of all ransomware classes during performance assessment. This approach helped avoid bias in metrics such as accuracy and F1-score, which can be misleading when applied to imbalanced data. Another major consideration was the dataset's high dimensionality. In its final form, the merged dataset contained 73 features and over 8,000 samples, requiring feature reduction steps such as correlation analysis and Recursive Feature Elimination (RFE) to remove redundant or low-impact attributes. These optimizations improved model efficiency without sacrificing predictive performance.

For the LSTM model, additional complexity arose from the need to transform tabular data into sequential form. Preparing fixed-length sequences (sequence length = 10) reduced the number of usable training instances, as the transformation inherently discards the first (sequence length - 1) rows per sequence set. This step, while necessary for temporal learning, required careful tuning to balance sequence context and data availability

From a computational perspective, models such as Gradient Boosting and LSTM introduced significant training time, particularly before feature reduction. This was addressed through dimensionality reduction techniques (e.g., PCA) and hyperparameter optimization. Moreover, early experiments showed that certain models struggled to classify specific ransomware families (minority classes) with high precision. Fine-tuning, combined with the balanced test evaluation, led to improve per-class metrics across most models.

To enhance robustness, synthetic noise augmentation was applied during training for certain models, simulating real-world data imperfections. This proved beneficial for assessing model stability under less-than-ideal input conditions. Despite these challenges, the final results demonstrated that both traditional machine learning and LSTM-based approaches can generalize effectively across multiple ransomware families, providing a foundation for further research and practical deployment.

4.6 Summary

This chapter demonstrated the practicality of both traditional machine learning and deep learning approaches for ransomware detection, using a combined dataset that integrates File System Monitoring (FSM), Process Behaviour Analysis (PBA), and Network Behaviour Analysis (NBA) features. The evaluation included four traditional models, Random Forest (RF), Support Vector Machine (SVM), Naive Bayes (NB), and Gradient Boosting (GB), alongside a Long Short-Term Memory (LSTM) network designed to capture temporal dependencies in the data. The models were assessed using performance metrics, including Accuracy, Precision, Recall, F1-Score, and Confusion Matrices, with a particular focus on per-class performance to ensure fair representation of all ransomware families. To address dataset imbalance, the final evaluation used manual test set balancing rather than relying solely on synthetic oversampling (e.g., SMOTE). This ensured that minority classes were adequately represented during performance assessment, thereby avoiding inflated metrics caused by skewed distributions.

Among the traditional models, Gradient Boosting achieved the highest overall performance, demonstrating strong classification ability across all ransomware classes. Random Forest and Naïve Bayes also produced competitive results, with NB offering a simpler yet effective alternative. In contrast, SVM underperformed in several minority classes, highlighting its sensitivity to class imbalance despite tuning. The LSTM model,

after careful sequence preparation and optimization, achieved competitive results, particularly excelling in detecting temporal behaviour patterns that static models might overlook.

Complementary techniques such as feature selection, dimensionality reduction, and noise injection were applied to reduce computational overhead and enhance model robustness. These strategies contributed to more realistic and generalizable performance outcomes, strengthening the reliability of the findings.

In conclusion, this chapter provides a comprehensive methodological foundation for ransomware detection, integrating both behavioural and machine learning-based techniques. The results not only validate the feasibility of the chosen models but also pave the way for their integration into a hybrid detection framework discussed in subsequent chapters, aiming to deliver a more adaptive and dependable defense against evolving ransomware threats.

Chapter 5: Result

5.1 Introduction to results and integration approaches

This chapter presents the experimental results and evaluates the effectiveness of integrating multiple ransomware detection approaches. The analysis builds upon the individual method evaluations from Chapter 4, focusing on how combining Behavior-Based (BB), Machine Learning (ML), and Long Short-Term Memory (LSTM) methods can enhance detection performance.

5.1.1 Purpose and Rationale of Integration

The integration of multiple detection approaches has emerged as a promising strategy in ransomware detection, driven by the need to overcome the inherent limitations of individual techniques. While a single detection method—whether behaviour-based or machine learning—may excel in certain aspects, it often struggles with others, such as handling novel ransomware variants, adapting to evolving attack patterns, or minimizing false positives. By combining complementary methods, the integrated system can leverage the strengths of each while compensating for their weaknesses, ultimately enhancing detection robustness and reliability.

5.1.2 Overview of Common Integration Strategies

In ransomware detection, several integration strategies are widely adopted to leverage the complementary strengths of different detection methods.

The first approach, Decision-Level Fusion, focuses on merging the final classification outputs of different models, such as determining whether a sample is benign or ransomware. This can be achieved using simple majority voting, where the label predicted by most models is selected, or through weighted voting, where models with higher historical accuracy are given greater influence in the final decision. While this method is straightforward to implement and interpret, it does not exploit the underlying feature-level information, which could sometimes limit its detection potential.

The second approach, Feature-Level Fusion, involves combining the input features extracted from multiple sources—such as File System Monitoring (FSM), Process Behaviour Analysis (PBA), and Network Behaviour Analysis (NBA)—into a single, unified feature vector. This enables the model to learn richer and more complex patterns

of ransomware behaviour, as it can correlate information across different behavioural dimensions. However, the approach tends to increase the dimensionality of the dataset, leading to higher computational costs and the potential need for dimensionality reduction techniques, such as Principal Component Analysis (PCA) or feature selection, to avoid overfitting.

A more sophisticated method is Model-Level Fusion, also referred to as meta-learning or stacking. Here, individual base models (e.g., Naive Bayes, Random Forest, FSM-based detectors, and LSTM networks) are trained independently, and their predictions are then used as input features for a meta-classifier, such as XGBoost. The meta-classifier learns how to best combine the strengths of each model while minimizing their weaknesses, often leading to superior detection performance compared to any single model. Despite its potential for high accuracy, this approach requires additional complexity, careful tuning, and more data to ensure that the meta-model generalizes well without overfitting.

Finally, Hybrid Pipelines (Sequential Integration) adopt a staged detection process, where the output of one method serves as the input or filter for the next. For example, an initial stage may involve behavior-based detection methods (FSM, PBA, NBA) to quickly flag suspicious activities, followed by a more computationally intensive ML or LSTM model in the second stage to confirm or reject the suspicion. This structure can significantly reduce the computational load on complex models and allow for faster decision-making in early stages. However, it introduces the risk that a low-recall first stage could incorrectly filter out true ransomware samples, preventing them from reaching later stages for confirmation.

Table 16*. Common Integration Strategies in Ransomware Detection*

Integration Strategy	Principle	Advantages	Limitations
Decision-Level Fusion	Combines final decisions (e.g., benign/malicious) from multiple models using majority or weighted voting.	Simple to implement; interpretable; robust if models are diverse.	Ignores underlying feature-level information; may not capture nuanced patterns.
Feature-Level Fusion	Merges features from different sources (FSM, PBA, NBA) into a single feature set for training.	Provides richer data representation; allows learning cross-behavioural correlations.	High dimensionality may lead to overfitting, increased computational cost, and require dimensionality reduction.
Model-Level Fusion (Stacking)	Uses predictions from base models as inputs to a meta-model (e.g., XGBoost) to optimize final decisions.	Can outperform individual models; leverages the strengths and mitigates the weaknesses of each model.	Complex to implement; requires more data; risk of overfitting if not tuned carefully.
Hybrid Pipelines (Sequential Integration)	Arranges models in stages, where outputs of one stage feed into the next (e.g., behaviour-based → ML/LSTM).	Reduces computational load on complex models; allows fast preliminary screening.	Risk of missing true positives if early-stage recall is low; requires careful stage design.

5.1.3 Defined Integration Tracks and Evaluation Strategy

This work evaluates a three-track integration that reflects how ransomware is observed and classified in practice:

(a) Behaviour-Based (BB) detection.

A rule/indicator-driven layer that consolidates host and network behavior signals (e.g., file/registry/API activity and traffic patterns) into human-interpretable alarms. BB methods are valuable as an early filter and for explainability, but can struggle with evasive or previously unseen variants; integrating BB with learned models helps offset these weaknesses.

(b) Machine Learning (ML).

Supervised models (e.g., Random Forest, Naïve Bayes, Gradient Boosting, SVM) trained on engineered features distilled from the fused dataset. These models provide strong baselines, calibrated probabilistic outputs, and complementary error profiles that are useful for fusion at the decision or model level. We assess them using standard metrics (Accuracy, Precision, Recall, F1) and a confusion matrix.

(c) Long Short-Term Memory (LSTM).

A temporal deep model that consumes fixed-length sequences to capture order-dependent behaviour indicative of ransomware campaigns. LSTM is included to test whether sequence modelling adds discriminative power beyond static features, particularly for minority classes and borderline behaviours.

Across the chapter, we will compare each approach in isolation to two integration schemes (decision-level voting/weighted voting and model-level stacking) to determine whether fusing BB, ML, and LSTM improves detection quality over any single method. Evaluation follows the same protocol and metrics used in Chapter 4, enabling a like-for-like comparison under balanced and realistic test conditions.

5.2 Method 1: Decision-Level Fusion (Voting / Weighted Voting)

5.2.1 Overview of Decision-Level Fusion Strategy

Decision-level fusion is a straightforward yet effective integration strategy in which each detection model (BB, ML, LSTM) produces an independent classification decision for the same input sample. These individual decisions are then aggregated to form a **final verdict**. The underlying assumption is that combining diverse models can reduce individual weaknesses and exploit complementary strengths, ultimately improving detection robustness and accuracy.

Figure 25 illustrates the architecture of the decision-level fusion as applied in this research, showing the independent prediction phase for each model followed by the fusion module.

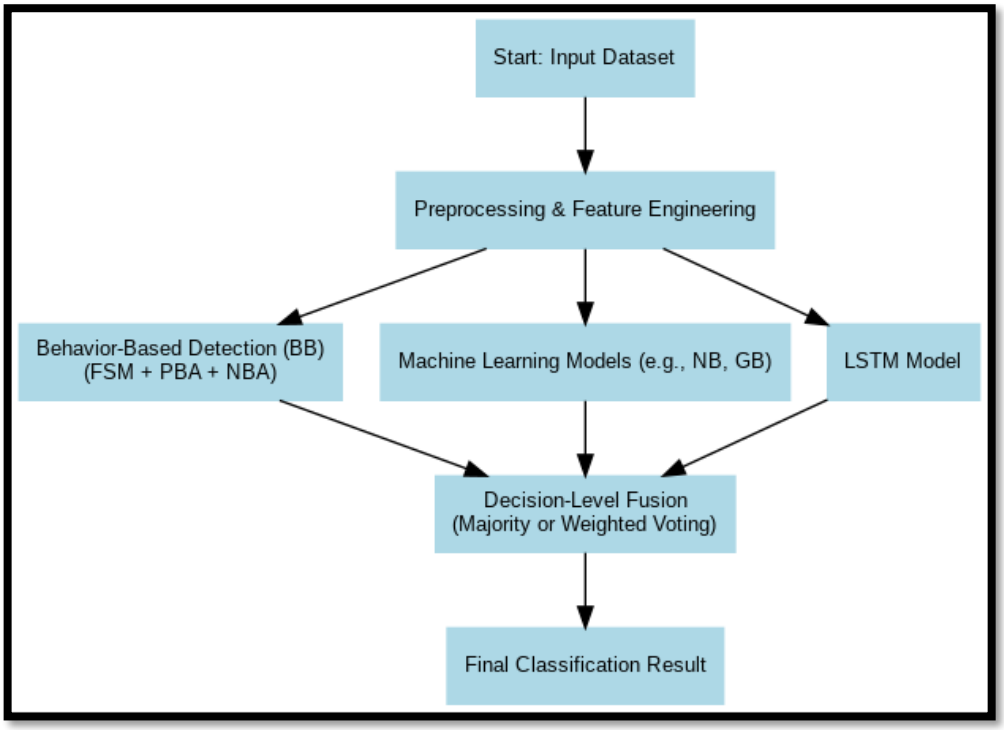

Figure 25. *Flow Diagram of Decision-Level Fusion for BB, ML, and LSTM Models*

5.2.2 Voting Strategies in Decision-Level Fusion

Two common variants of decision-level fusion are applied:

- **Majority Voting** – The class label predicted by the majority of models becomes the final decision. In the case of a tie, predefined tie-breaking rules are applied (e.g., priority to the model with the highest standalone accuracy).

In this strategy, the final class is determined by the most frequently predicted label among all participating models. For the **BB** model, predictions are treated as binary (0 = Normal, any other value = Abnormal/Malware) to contribute to distinguishing normal from abnormal samples. In the event of a tie, the decision is resolved by prioritizing the prediction from the model with the highest standalone performance (based on metrics such as F1-score or accuracy).

- **Weighted Voting** – Each model's vote is assigned a weight proportional to its performance in standalone evaluation (e.g., accuracy or F1-score from Chapter 4). The final class is chosen based on the highest weighted sum of votes.

  This strategy assigns each model a weight proportional to its standalone performance (e.g., the F1-macro score computed on the common test set). The class probabilities (proba_*) from all models are then combined after being weighted accordingly. For the BB model, its binary probabilities are converted into multi-class probabilities by assigning proba_normal to the Normal (0) class and distributing proba_ransom equally across the remaining non-zero classes.

5.2.3 Pre-Fusion Predictions Extraction and Evaluation

The outputs generated in this stage serve as the input for the evaluation phase, where the fusion performance is compared with standalone models.

Before implementing the decision-level fusion strategy, it was essential to generate and evaluate the predictions from each detection mechanism: Behaviour-Based (BB), Machine Learning (ML), and Long Short-Term Memory (LSTM). This step ensured that the outputs of all models were aligned in terms of format and evaluation metrics, making them compatible for the fusion process.

Predictions Generation

- **Behaviour-Based (BB) Predictions**: Predictions were generated using a threshold-based suspicious score computed from selected behavioural features (e.g., file system, registry, and memory operations). Each sample received a binary prediction ("Ransomware" or "Normal") based on whether its suspicious score exceeded a defined threshold. We also calculated the corresponding

probabilities (proba_normal and proba_ransom) by normalizing the score range, enabling BB outputs to be integrated with probabilistic fusion methods.

- **Machine Learning (ML) Predictions**: The best-performing supervised ML classifier (from Chapter 4) was applied to the test dataset, producing both **class predictions** and **class probabilities**. Predictions were stored in a standardized CSV format with fields for sample ID, predicted label, true label, and per-class probabilities.
- **Long Short-Term Memory (LSTM) Predictions**: The sequence-based LSTM model was trained on pre-processed and temporally ordered dataset samples. Predictions for the test set included the **binary output,** as well as the **SoftMax probabilities** for each class, consistent with the output structure of the other models.

5.3 Method 2: Model-Level Fusion (Meta-Learning-Stacking)

5.3.1 Overview of the Stacking Technique

Stacking (a.k.a. stacked generalization) combines several base learners by feeding their outputs into a second-level model (the meta-learner) that learns how to weight and reconcile them. Unlike plain voting, which aggregates decisions directly, stacking trains a classifier on the predicted probabilities (or scores) produced by the base models, allowing the final model to exploit integration strengths and error patterns across learners. In scikit-learn, this idea is implemented by the Stacking Classifier. The approach originates from Wolpert's "stacked generalization" and is closely related to the "Super Learner" framework that learns optimal convex combinations via cross-validated risk.

5.3.2 Base Learners in the Stacking Ensemble

We included (i) a behavior-based rule/baseline model (BB) that summarizes suspicious activity into probabilities, (ii) four classical ML classifiers—Naïve Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), and XGBoost (XGB)—and (iii) a sequence model (LSTM). Each base learner outputs class-probability vectors in a unified format (columns like *_c0, *_c1, *_c2). These probability features are the inputs to the meta-learner. (Implementation follows the scikit-learn stacking interface and user guide.)

5.3.3 Leakage-free meta-features (OOF).

To avoid information leakage when training the meta-learner, we generated out-of-fold (OOF) predictions for each base model using stratified K-fold cross-validation. For each fold, a base learner is trained on K−1K−1K−1 folds and predicts the held-out fold; concatenating these held-out predictions across folds yields OOF probability features for the entire training set. Stratification preserves the class distribution in each fold for fairer evaluation on imbalanced data. We then trained the meta-learner solely on these OOF features and evaluated it on the separate test set. (Files produced include *_oof_meta_train.csv for training-OOF and meta_test_predictions.csv for test probabilities.)

5.3.4 Probability calibration (optional but used here).

Because the meta-learner consumes probabilities, we calibrated models where needed (e.g., LSTM) using held-out data/cross-validation with CalibratedClassifierCV (sigmoid or isotonic). Calibration improves the reliability of predicted probabilities and typically benefits PR-curve metrics.

5.3.5 Model subset selection for stacking.

Rather than stacking all base learners indiscriminately, we performed a greedy model selection over candidate subsets using cross-validation on the OOF features. This procedure incrementally adds the model that most improves validation performance until no further gain is achieved. In our experiments, the best performing subset for the final stack comprised [BB, XGB, NB] (the remaining learners did not improve cross-validated F1/AUPRC on the meta features). This selection was then fixed for test-time evaluation.

5.3.6 Meta-learner

The meta-learner is a simple, regularized linear classifier (logistic regression in our implementation), trained on the concatenated OOF probability vectors of the selected base models. At test time, we feed the corresponding test-set probabilities into the meta-learner to obtain final class predictions and per-class probabilities (meta_c0, meta_c1, meta_c2) stored in meta_test_predictions.csv. (Any well-regularized classifier could be used; stacking in scikit-learn is agnostic to the choice of final estimator.)

5.3.6 Evaluation metrics (reported later in section 5.5).

To assess multi-class performance under class imbalance, we report:

- Accuracy (overall correctness).
- Precision-macro, Recall-macro, F1-macro (unweighted mean across classes, giving minority classes equal weight).
- AUPRC-macro (macro average precision) computed one-vs-rest from class probabilities.

  Average precision summarizes the precision–recall curve as a weighted mean of precisions over recall thresholds; using average='macro' yields an unweighted average over classes. We also present the classification report (per-class precision/recall/F1/support) and the confusion matrix with the convention "rows = true, columns = predicted." These follow the standard scikit-learn definitions used throughout this chapter.

Why stacking here?

Compared with decision-level voting, stacking learns data-driven weights conditioned on the feature space via a trained meta-learner, enabling it to capitalize on complementary error profiles (e.g., BB's high specificity, XGB's discriminative power, NB's robustness with limited features). The OOF protocol ensures that the meta-learner generalizes, not merely memorizes base-model behavior on the training data.

5.4 Experimental Setup

This section describes the procedures followed to prepare the data, train the individual detection models, evaluate their performance, and integrate their outputs using decision-level fusion. The workflow was designed to ensure methodological consistency and reproducibility across all models. Ensemble learning aims to improve classification performance by combining the outputs of multiple models. Among the most common aggregation strategies are Hard Voting and Soft Voting. In Hard Voting, each model casts a single vote for the predicted class label, and the final prediction is determined by the majority class across all models. This method treats all models equally and does not consider their prediction confidence. Conversely, Soft Voting aggregates the predicted class probabilities from each model, and the class with the highest average probability is selected as the final prediction. Soft Voting often yields superior performance when the base models are well-calibrated, as it incorporates both the prediction and the associated confidence level, enabling models with higher certainty to exert more influence. In the

context of ransomware detection, Soft Voting is particularly advantageous because it allows the integration of heterogeneous models—such as behaviour-based methods (BB), traditional machine learning (ML), and deep learning approaches like LSTM—while leveraging their varying strengths in probability estimation and pattern recognition.

5.4.1 Decision-Level Fusion (Voting / Weighted Voting)

5.4.1.1 Each model data Preparation for Decision-Level Fusion (Voting / Weighted Voting)

To ensure a fair comparison, all models (BB, classical ML, and LSTM) were trained on the same training split and evaluated on the same 1,676-sample test set.

- **Behavior-Based (BB):** We extracted behavior-oriented features (file-system operations, registry activity, memory actions) and computed a per-sample suspicious_score; a fixed threshold yielded the binary decision.
- **ML Modes (**Naïve Bayes, SVM-RBF, Random Forest, XGBoost**).** We used a single preprocessing Pipeline with a ColumnTransformer: median imputation and scaling for numeric columns, and one-hot encoding with handle_unknown='ignore' for categorical. This produces a unified feature space and avoids data leakage.
- **LSTM Model:** We applied the same preprocessing to obtain a dense feature matrix, reshaped each sample to (nfeatures,1) (n_{\text{features}}, 1) (nfeatures ,1), and trained a compact Sequential model with one LSTM layer, dropout, and a final softmax layer (features treated as a single channel; no fixed time window was constructed).

5.4.1.2 Training Individual Models

- **BB Model:** Rule-based and did not require a training phase; classification depended on thresholding the suspicious score.
- **ML Models:** Trained supervised classifiers such as Naïve Bayes, Random Forest, and XGBoost using the training partition.
- **LSTM Model:** Configured with one LSTM layer, dropout regularization, and dense layers for classification.

5.4.1.3 Unified Testing Procedure

All models were evaluated using the same test dataset to maintain comparability. This ensured that performance differences were due to model characteristics rather than data discrepancies.

5.4.1.4 Prediction Output Standardization

To prepare for fusion, the predictions from each model were saved in a standardized CSV format containing:

- sample_id
- true_label
- predicted_label
- Additional fields such as prediction probabilities and model name (for ML/LSTM) or suspicious score (for BB).

This unified prediction table enables downstream decision-level fusion and per-model evaluation, and matches the inputs expected by scikit-learn reporting utilities (e.g., classification_report, confusion matrices).

5.4.1.5 Performance Metrics

For each model, the following metrics were computed:

- Accuracy
- Macro Precision
- Macro Recall
- Macro F1-score
- Confusion Matrix
  These metrics were used both for individual performance evaluation and as weight factors in the Weighted Voting fusion scheme.

5.4.1.6 Preparation for Decision-Level Fusion

After standardizing outputs, the predictions were merged into a single dataset where each row contained:

- The true label
- Predictions from BB, ML, and LSTM
- Corresponding probabilities or confidence scores

This unified prediction dataset served as the input for implementing Majority Voting and Weighted Voting decision-level fusion methods described in Section 5.

Table 17. *Fusion Subset Model Performance*

Model	Samples	Accuracy	Precision - macro	Recall - macro	F1 - macro
RandomForest	1676	0.983	0.984	0.903	0.940
Naive_Bayes	1676	0.954	0.984	0.673	0.775
XGBoost	1676	0.969	0.966	0.820	0.881
BB	1676	0.943	0.596	0.506	0.538

Table 17 presents the standalone performance of each model (RandomForest, Naive Bayes, XGBoost, and Behaviour-Based (BB)) on the common intersection of the test set used for fusion. For each model, the number of evaluated samples, Accuracy, Precision (macro-averaged), Recall (macro-averaged), and F1-score (macro-averaged) are reported. These metrics provide a quantitative basis for understanding the strengths and weaknesses of each model before applying decision-level fusion. Notably, RandomForest achieved the highest F1-score (0.940), while BB exhibited the lowest (0.539), indicating its role as a complementary detector rather than a primary classifier.

5.4.1.7 Fusion Weights Calculation

The weights used in the Weighted Voting scheme were derived from the macro-averaged F1-scores obtained in the standalone evaluation of each model on the common test subset. The F1-macro metric was chosen because it provides a balanced measure of performance across all classes, making it particularly suitable for handling the multi-class and potentially imbalanced nature of ransomware detection. Each model's F1-macro score was divided by the sum of all F1-macro scores to produce normalized weights that sum to 1. Models with higher F1-macro scores have greater influence in the final decision.

Table 18. *fusion weight for ML and BB*

Model	Weight
RandomForest	0.299
Naive_Bayes	0.247
XGBoost	0.280
BB	0.171

The weights represented in Table 18 assigned to each model in the Weighted Voting scheme were derived from their macro F1-scores in standalone evaluation. This approach ensures that models with better overall class balance performance have greater influence

in the fusion decision. Specifically, RandomForest received the highest weight (0.300), followed by XGBoost (0.281), Naive Bayes (0.247), and BB (0.172). These weights were normalized to sum to 1 and directly applied to the class probability outputs of each model during the fusion process.

5.4.2 Meta-Learning (Stacking)

5.4.2.1 Data and label space.

All results in this chapter are computed on the same held-out test set (final_test.csv). The multi-class label space is fixed as class_order = [0, 1, 2] (0 = benign, 1/2 = ransomware sub-types). Class proportions are imbalanced, so we report macro-averaged metrics in addition to overall accuracy (§5.5). Stratification is used wherever cross-validation is involved to preserve per-class proportions.

5.4.2.2 Base learners and outputs.

We train the following base models: a behavior-based (BB) rules/baseline classifier; classical ML models NB, RF, SVM, XGB; and LSTM. Each base learner produces a per-class probability vector in a unified schema (columns like *_c0, *_c1, *_c2). These probabilities serve as meta-features for stacking (cf. scikit-learn's StackingClassifier).

5.4.2.3 Leakage-free meta-features (OOF).

To train the meta-learner without leaking test information, we generate out-of-fold (OOF) predictions for every base learner using Stratified K-Fold. For each fold, the base model is trained on K−1K-1K−1 folds and predicts the held-out fold; concatenating these held-out predictions yields OOF probability features for the entire training set. The meta-learner is then fitted only on these OOF features; test-set probabilities are never used in training. Stratification maintains class ratios in each fold. (Implementation follows the standard scikit-learn API for StratifiedKFold.)

5.4.2.4 Probability calibration.

Because the meta-learner consumes probabilities, we calibrate models when needed (e.g., LSTM) using scikit-learn's CalibratedClassifierCV (sigmoid/isotonic) with internal CV to improve probability reliability—beneficial for PR-curve–based metrics.

5.4.2.5 Greedy model subset selection.

Rather than stacking all learners, we run a greedy forward selection on the OOF space: starting from the best single model, iteratively add the base learner that maximizes

validation performance (F1/AUPRC) until no further gain. This procedure selected [BB, XGB, NB] as the final subset used by the meta-learner (section5.3).

5.4.2.6 Meta-learner.

We use a simple, regularized linear classifier (logistic regression) as the meta-learner. It is trained on the concatenated OOF probability features of the selected base models. At test time, the corresponding test probabilities are fed to the meta-learner to produce final predictions and class probabilities (meta_c0, meta_c1, meta_c2). (Stacking in scikit-learn is agnostic to the choice of the final estimator.)

5.4.2.7 Evaluation protocol and metrics.

All reporting in section 5.5 uses the held-out test set with the fixed class order. We compute: Accuracy, Precision-macro, Recall-macro, F1-macro using scikit-learn's classification_report (macro = unweighted mean across classes—suitable under imbalance); AUPRC-macro (macro average precision) via one-vs-rest with average_precision_score on predicted probabilities;

Confusion matrix with the convention rows = true, columns = predicted. Implementations follow the scikit-learn documentation for these metrics and displays.

Reproducibility

Where applicable, we fix random seeds (random_state) for CV splits and model training; we document software versions (Python / scikit-learn) consistent with the scikit-learn user guide for model selection and evaluation.

The combination of stratified OOF meta-features, calibrated probabilities, and a simple regularized meta-learner provides a leakage-free and robust estimate of generalization for stacking, enabling a fair comparison with Decision-Level Voting on the same test set.

5.5 Results and Comparison

5.5.1 Decision-Level Fusion Results (ML + BB)

Table 19 presents the comparative results of Majority Voting and Weighted Voting for the integration of Machine Learning (ML) models with the Behaviour-Based (BB) approach. The corresponding confusion matrices are shown in Figure 26 and Figure 27.

Table 19. *fusion results metrics for ML and BB*

	Samples	Accuracy	Precision	Recall	F1
Majority Voting	1676	0.969	0.977	0.820	0.884
Weighted Voting	1676	0.969	0.977	0.820	0.884

Table 19 reports the aggregated performance metrics after applying Decision-Level Fusion using two strategies: Majority Voting and Weighted Voting. Both methods yielded identical results in this experiment, with an Accuracy of 0.9696, macro-Precision of 0.9771, macro-Recall of 0.8205, and macro-F1-score of 0.8845. The equality of results is due to the strong agreement between model predictions across most samples, resulting in minimal impact from weighting adjustments. The fusion improved the balance between precision and recall compared to some individual models, confirming the effectiveness of combining heterogeneous detection methods. The identical results for majority voting and weighted voting in this experiment can be attributed to the high level of agreement among the three ML models (RandomForest, Naive Bayes, and XGBoost) on the majority of test samples. Since both fusion methods ultimately depend on the collective decisions of the models, when predictions are highly consistent, the weighting of votes has minimal or no effect on the outcome. Additionally, the BB model, which had the lowest standalone F1-score, contributed little to decision shifts; its votes were often overridden by the agreement among the higher-performing models. As a result, even when weights were applied based on macro F1-scores, the decision outcomes remained identical to the unweighted majority vote.

Compared to the best-performing single ML model reported in Section 5.4.6 — Random Forest (Macro-F1 = 0.940) — the decision-level fusion approach achieved a slightly lower Macro-F1 of 0.884. However, it significantly outperformed the weakest standalone model, Naïve Bayes (Macro-F1 = 0.775), by leveraging complementary strengths among the base models.

The identical results observed for Majority Voting and Weighted Voting in Table 22 can be explained by the high agreement rate among the three ML models (Random Forest, Naïve Bayes, and XGBoost) on most test samples. In cases where model predictions are already well-aligned, the impact of weighting is minimal because the final decision is dominated by the consensus. The BB model — which had the lowest standalone performance — contributed less to decision shifts, as its votes were often overridden by the agreement among the higher-performing ML models.

This outcome suggests that in scenarios with limited disagreement between models, both fusion strategies converge to the same performance level. However, in datasets with greater variability or lower inter-model agreement, Weighted Voting may offer additional benefits, particularly when base model probabilities are well-calibrated.

After applying Decision-Level Fusion, both Majority Voting and Weighted Voting yielded identical results, with an Accuracy of 0.969, macro-Precision of 0.977, macro-Recall of 0.820, and macro-F1-score of 0.884. The identical outcomes are due to the high agreement among the three ML models (RandomForest, Naïve Bayes, XGBoost) across most samples, leaving little impact for weighting adjustments. The Behaviour-Based (BB) model, having the lowest standalone F1-score, contributed minimally to decision shifts; its predictions were often overridden by the agreement of higher-performing models. This consistency between fusion methods highlights that when base model predictions are well aligned, weighting does not significantly alter results. In scenarios with greater disagreement or well-calibrated probabilities, Weighted Voting may provide additional advantages.

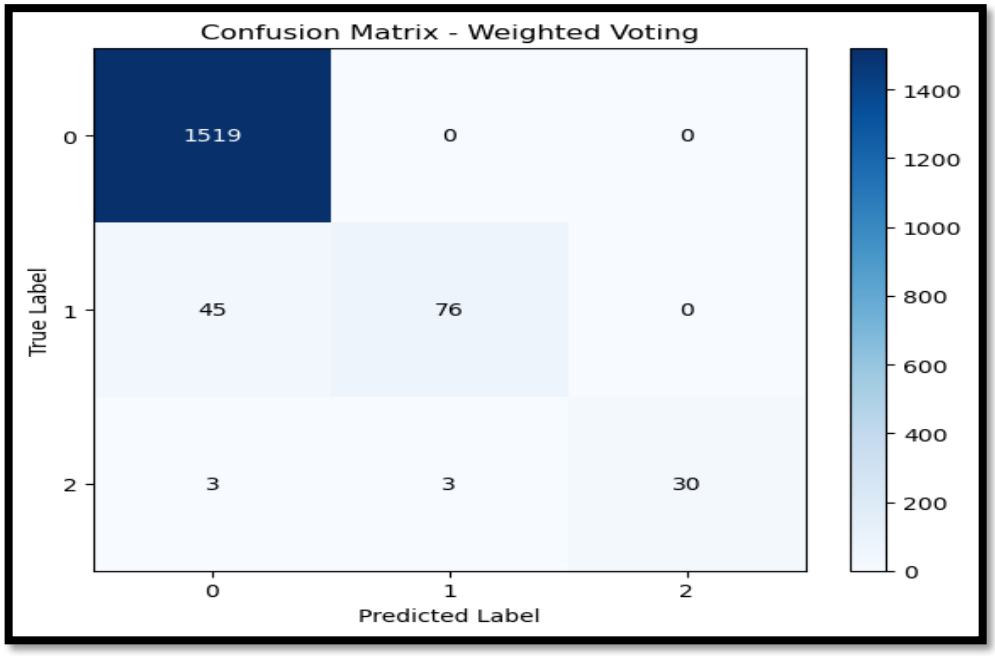

Figure 26. *Confusion Matrix - Weighted Voting for ML and BB*

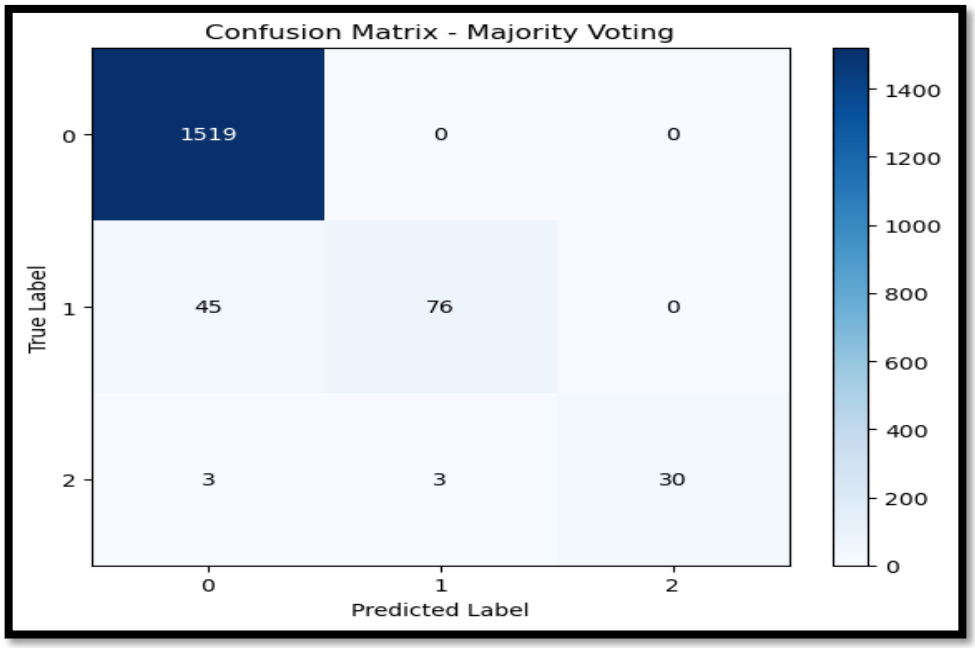

Figure 27. *Confusion Matrix - Majority Voting for ML and BB*

Figures 26 and 27 show the confusion matrices for Majority Voting and Weighted Voting, respectively, applied to the combined predictions of the ML models and the BB method. Each matrix illustrates classification across the three classes (0, 1, 2), where diagonal cells indicate correct predictions and off-diagonal cells represent misclassifications. The nearly

identical patterns in both figures confirm the metrics' similarity and reflect the strong agreement between models, resulting in minimal variation between the two fusion strategies.

The near-identical outputs of the two strategies can also be attributed to the balanced weight distribution from fusion_weights.csv, where the highest-performing ML models dominated the decision outcome, leaving minimal room for divergence between majority and weighted voting results.

5.5.2 Hierarchical Ensemble Framework

The hierarchical ensemble framework integrates heterogeneous detection capabilities by separating the classification process into two sequential levels. In Level 1, binary classification is performed to filter out benign samples from ransomware, leveraging combined predictions from the Behavior-Based (BB) model, an ML ensemble (Random Forest, XGBoost, Naïve Bayes), and the LSTM model. In Level 2, only ransomware-labeled samples from Level 1 undergo multiclass classification to identify specific ransomware families. This two-stage structure ensures that the binary-focused BB model contributes effectively to initial threat detection while avoiding the limitations of its lack of family-level discrimination, and that ML and LSTM models specialize in the more granular family classification task.

Level 1 Results

Table 20. *Level 1 Binary Classification*

	precision	**recall**	**f1-score**	**support**
Benign (0)	0.926	1	0.961	163
Ransom (1+)	1	0.917	0.956	157
Accuracy	0.959	0.959	0.959	0.959
macro avg	0.963	0.958	0.959	320
weighted avg	0.962	0.959	0.959	320

This table 20 summarizes the performance metrics for the binary classification stage (Level 1), where the system distinguishes between benign and ransomware samples. The Benign class achieved a precision of 0.926 and a recall of 1.0, indicating that no benign samples were misclassified as ransomware (zero false positives). The Ransomware class obtained a perfect precision of 1.0 and a recall of 0.917, showing that while every

predicted ransomware sample was indeed ransomware, 8.3% of actual ransomware samples were missed (false negatives).

The overall accuracy at this stage was 95.9%, with a macro-average F1-score of 0.959, indicating balanced performance across both classes. The weighted average closely matches the macro average, reflecting the relatively balanced distribution between benign (163 samples) and ransomware (157 samples) in the test set. This stage's primary strength is its ability to completely avoid false alarms, while its main limitation lies in missing a small number of ransomware cases.

Level 2 Results

Table 21. *Level 2 Multiclass Classification*

	Precision	**Recall**	**F1-score**	**support**
0	0	0	0	0
1	0.97	0.97	0.97	108
2	1	0.91	0.95	36
accuracy	0.958	0.95	0.95	0.95
macro avg	0.657	0.62	0.64	144
weighted avg	0.979	0.95	0.96	144

Table 21 presents the performance of the multiclass classification stage (Level 2), which operates only on ransomware samples identified by Level 1. As a result, the total number of samples drops from 320 in Level 1 to 144 in Level 2, due to the exclusion of the 13 ransomware samples missed in Level 1 (false negatives) and the absence of benign samples misclassified as ransomware (false positives).

The results show that Class 1 achieved precision = 0.97 and recall = 0.97, demonstrating consistently high performance for the dominant ransomware family in the dataset (108 samples). Class 2 obtained a perfect precision of 1.0 but a slightly lower recall of 0.91, indicating a small number of missed cases. Class 0 shows no entries because no samples belonging to that class were passed to this stage.

The overall accuracy for Level 2 is 95.8%, with a macro-average F1-score of 0.64. The macro average is notably lower than the weighted average (0.96) due to the class imbalance; Class 2 has fewer samples (36) compared to Class 1 (108), and Class 0 is absent entirely. Despite this, the weighted average performance confirms that the system

is highly effective at distinguishing ransomware families when provided with correctly detected ransomware cases from Level 1.

Relationship Between Level 1 and Level 2 Results

The performance of Level 2 is directly influenced by the outcomes of Level 1. Since Level 1 perfectly eliminated false positives (FP = 0), Level 2 received only true ransomware cases for family-level classification. This clean input reduced noise in the multiclass stage, allowing the classifier to focus solely on distinguishing between ransomware families.

However, the 13 false negatives from Level 1 — ransomware samples incorrectly classified as benign — were permanently excluded from Level 2. As a result, the total sample size dropped from 320 in Level 1 to 144 in Level 2, and those missed cases could not be recovered at later stages. This shows a key dependency: any detection errors in Level 1 directly translate to reduced coverage in Level 2.

In summary, the strength of Level 1 in avoiding false alarms supports Level 2's high accuracy, but its weakness in missing some ransomware samples limits Level 2's potential coverage. This interdependence highlights the importance of optimizing Level 1's recall to maximize the effectiveness of the overall two-stage detection pipeline.

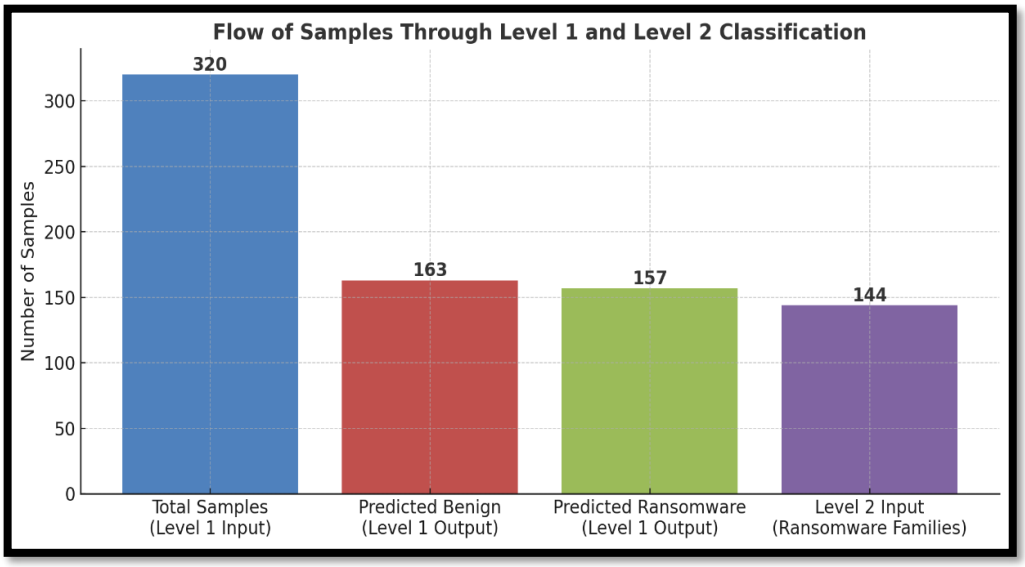

Figure 28. *level 1 and level 2 classification*

Level 1 — Binary Classification (Benign vs. Ransomware)

The Level 1 table 20 reports the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) for the binary classification stage. The results show:

- TP = 144: Ransomware samples correctly detected as ransomware.
- FP = 0: No benign samples were incorrectly flagged as ransomware, indicating zero false alarms at this stage.
- FN = 13: Ransomware samples missed and incorrectly classified as benign.
- TN = 163: Benign samples correctly identified as benign.

These results correspond to a total of 320 samples evaluated at Level 1 (163 benign + 157 ransomware). The absence of false positives is a notable strength of the fusion approach, significantly reducing unnecessary alerts. However, the 13 false negatives represent missed ransomware detections, which directly impacts the number of samples available for further classification in Level 2.

Table 22. *Level 1 Binary Classification Performance*

Value	TP	FP	FN	TN	Total
0	144	0	13	163	320

Level 2 — Multiclass Classification (Ransomware Family Identification)

The Level 2 table 22 presents per-class TP, FP, FN, and TN values for the ransomware subset passed from Level 1. Only samples predicted as ransomware in Level 1 are included here. As a result:

- The total sample size drops from 320 to 144.
- This reduction occurs because Level 1 produced 13 false negatives, removing those ransomware samples from consideration in Level 2. Since FP=0 in Level 1, no benign samples were added to Level 2.
- Thus, 157 true ransomware−13 FN=144157\ {true ransomware} - 13\{FN} = 144157 true ransomware−13 FN=144 ransomware samples are available for family-level classification.

The results show:

- Class 1 (e.g., Family A): TP = 105, FP = 3, FN = 3, TN = 33 — strong detection accuracy with minor misclassifications.
- Class 2 (e.g., Family B): TP = 33, FP = 0, FN = 3, TN = 108 — perfect precision (no false positives) and minimal false negatives.

- Class 0 has no true samples in this subset, indicating that this family was not present in the evaluated ransomware set after Level 1 filtering.

Table 23. *Level 2 Multiclass Classification Performance*

Class	TP	FP	FN	TN	Total
0	0	3	0	141	144
1	105	3	3	33	144
2	33	0	3	108	144

Summary of the transition from 320 to 144 samples: Level 2 operates only on samples classified as ransomware by Level 1. Since Level 1 perfectly excluded benign samples (FP = 0) but missed 13 ransomware cases (FN = 13), the remaining 144 true ransomware samples were passed to Level 2 for family-level classification. This explains the significant reduction in sample size between the two stages.

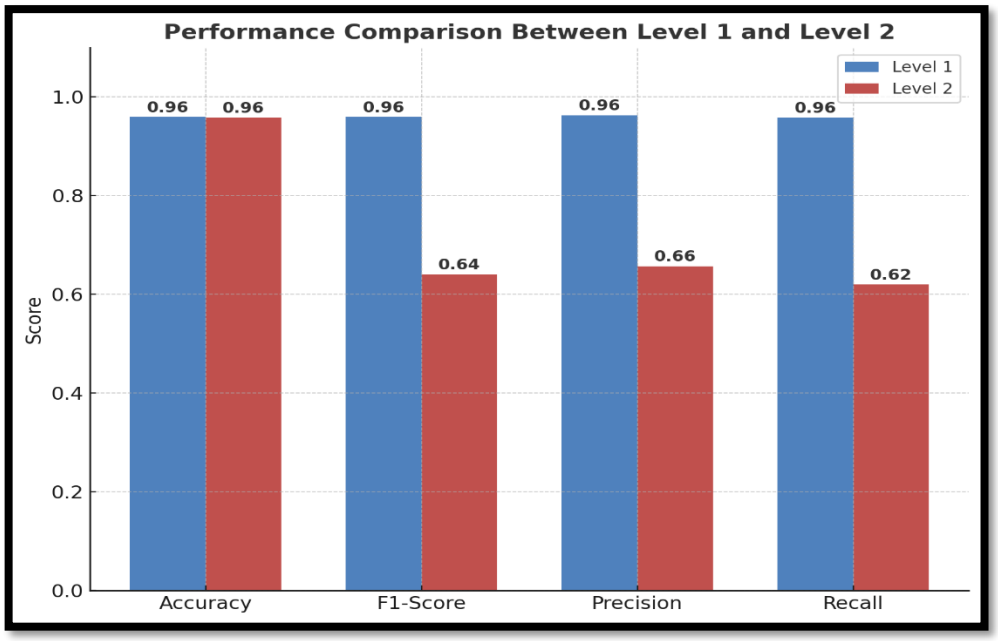

Figure 29. *performance comparison between level 1 and level 2*

Figure 29 compares the binary classification results from Level 1 with the multiclass classification results from Level 2 across four key metrics: Accuracy, Precision, Recall, and F1-Score. These results were obtained after applying the hierarchical ensemble framework described in Section 5.5.2, where Level 1 filters benign from ransomware samples before Level 2 attempts ransomware family identification. The bar chart shows

that Accuracy remains consistently high in both stages (~95.8–95.9%), indicating that overall correctness is maintained despite the shift from binary to multiclass classification. Precision is perfect (1.0) in Level 1 for ransomware detection due to the absence of false positives, no benign samples were incorrectly flagged. Recall, however, drops slightly from Level 1 to Level 2, reflecting the 13 ransomware samples missed in the first stage that never entered the second stage. Consequently, the F1-score also decreases in Level 2, particularly due to class imbalance and the absence of Class 0 samples. This visualization highlights the dependency between the two stages: strong precision in Level 1 eliminates false alarms, but any recall loss at this stage directly reduces the sample pool for Level 2, impacting its ability to classify all ransomware families accurately.

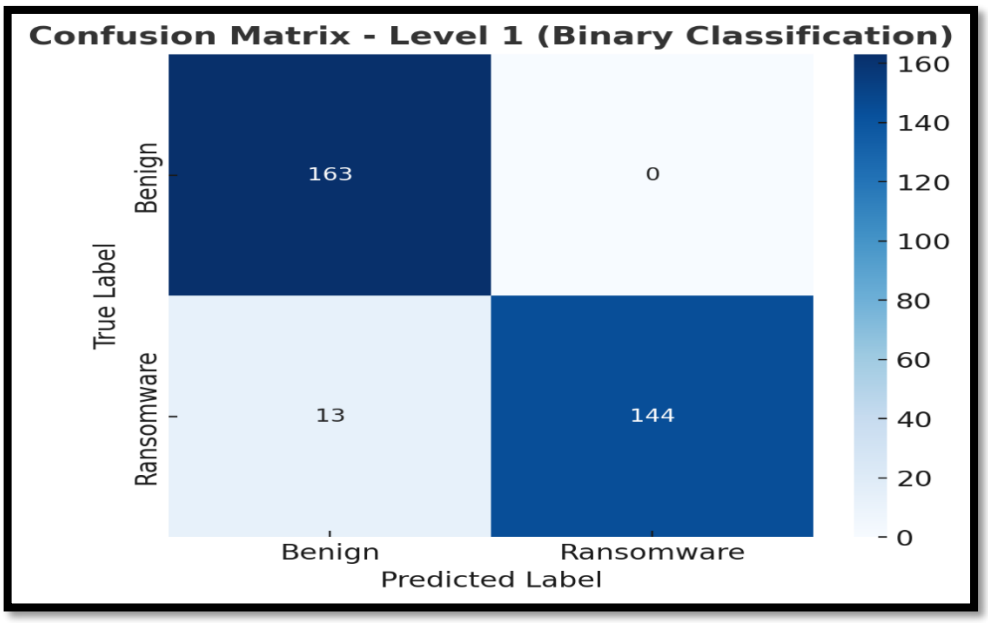

Figure 30. *level 1 binary classification*

Figure 30 presents the confusion matrix for the binary classification stage, which differentiates between benign and ransomware samples. The matrix shows that all benign samples (163) were correctly classified, resulting in zero false positives (FP = 0) — a critical achievement for minimizing false alarms in practical deployment. For ransomware samples, 144 were correctly detected (true positives) while 13 were missed (false negatives), meaning they were incorrectly classified as benign. The total number of evaluated samples in Level 1 is 320 (163 benign + 157 ransomware). The diagonal dominance in the matrix reflects a robust classification capability, particularly for benign

detection, while the off-diagonal values highlight the small proportion of ransomware missed. This stage benefits from the ensemble's ability to merge predictions from Machine Learning models, the behaviour-Based detector, and the LSTM model, producing highly confident results for benign classification. However, the presence of false negatives directly impacts Level 2, as these missed ransomware samples are excluded from subsequent family-level classification.

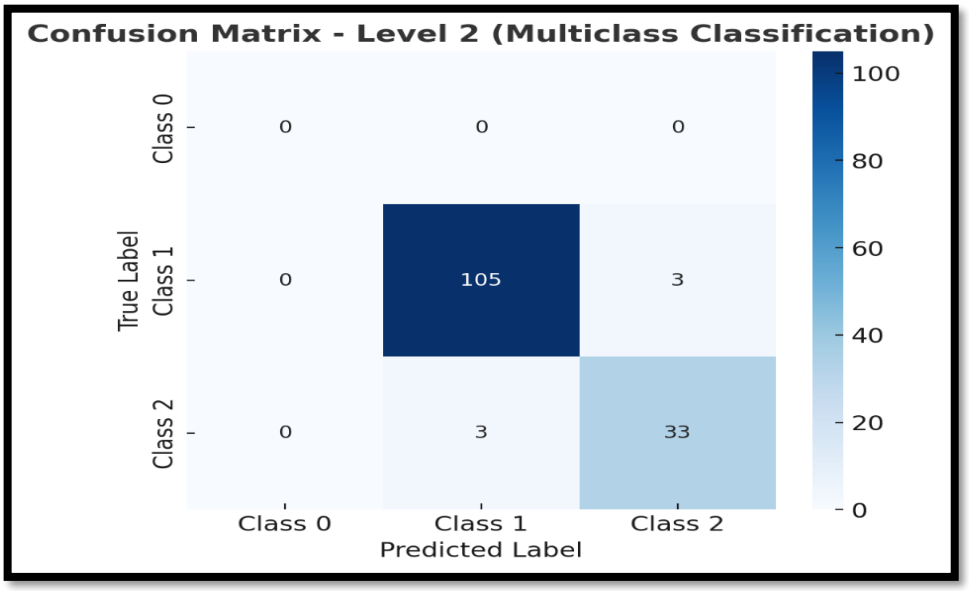

Figure 31. *level 2 binary classification*

Figure 31 illustrates the confusion matrix for the multiclass classification stage, applied exclusively to the ransomware subset passed from Level 1. As Level 1 produced no false positives, only actual ransomware cases entered this stage, reducing the sample count from 320 to 144 (due to the 13 false negatives in Level 1). The results show Class 1 (dominant ransomware family) achieved 105 correct predictions, with 3 misclassifications into Class 2. Class 2 achieved 33 correct predictions with 3 misclassifications into Class 1, and Class 0 had no entries because no samples from this class were passed forward. The strong diagonal elements for Classes 1 and 2 indicate high precision and recall despite the class imbalance, while the empty Class 0 row confirms that Level 1 filtering permanently excluded any class it failed to detect. This direct dependency on Level 1's outputs highlight the importance of reducing false negatives in the first stage to maximize the diversity and representativeness of ransomware families available for Level 2 classification.

The relationship between the confusion matrices of Level 1 and Level 2 reveals how misclassifications in the binary classification stage propagate into the multiclass classification stage. In Level 1, 13 ransomware samples were misclassified as benign, effectively removing them from further analysis in Level 2. This explains the reduced total sample count in Level 2 (144 instead of 157 ransomware samples detected in Level 1) and the complete absence of Class 0 predictions. Since the missed ransomware samples never enter Level 2, the multiclass classifier operates on an incomplete representation of the true ransomware distribution. Consequently, although Level 2 demonstrates strong performance for Classes 1 and 2, its results are inherently limited by the filtering effect of Level 1. This dependency highlights the importance of minimizing false negatives in the first stage to preserve the diversity and quantity of samples available for fine-grained classification in the second stage.

These findings indicate that when base model predictions are already well-aligned, fusion strategies tend to converge to the same outcome. For scenarios with greater model disagreement, Weighted Voting may provide additional benefits, particularly when base model probabilities are properly calibrated.

5.5.3 End-to-End Voting Results

We adopted a two-stage hierarchical pipeline—an initial detector (L1: benign vs. ransomware) followed by a family-level classifier (L2: ransomware subtype)—because hierarchical decomposition is a well-established strategy for complex taxonomies and, crucially, it localizes error sources across stages. Accordingly, L1 and L2 results are reported diagnostically: L1 reveals coverage losses (e.g., false negatives that never reach family classification), while L2 quantifies intra-family confusions conditional on passing L1. For the official model comparison, however, we evaluate the system end-to-end on the full test set with a fixed label order and macro-averaging to treat minority classes fairly, and we additionally report macro-AUPRC since precision–recall is more informative than ROC under class imbalance. This end-to-end summary establishes a fair baseline for decision-level voting against which we compare meta-learning (Stacking)—a supervised combiner that can exploit complementary error patterns among base learners rather than rely on fixed voting rules.

To make an end-to-end, fair comparison under class imbalance, we report macro-averaged metrics—Precision, Recall, and F1—computed on the final three-class

predictions with a fixed label order; we also include macro-AUPRC, which summarizes the precision–recall curve and is recommended over ROC-AUC when classes are imbalanced. The per-class report exposes which classes drive gains or losses, while the confusion-matrix-derived counts (TP/FP/FN/TN) provide an interpretable error map that links directly to those per-class rates. All metrics were computed with scikit-learn (classification_report for macro/weighted definitions; confusion_matrix and ConfusionMatrixDisplay for label-consistent matrices; average_precision_score(..., average='macro') for macro-AUPRC), and the same evaluation protocol is used later to benchmark against **Stacking (meta-learning)**.

Table 24. *End-to-End Performance of Decision-Level Voting (Three-Class)*

Method	Voting (final, 3-class)
Accuracy	0.94
Precision-macro	0.96
Recall-macro	0.93
F1-macro	0.94
AUPRC-macro	0.97
N	320

Table 24 reports the official end-to-end performance of the decision-level Voting pipeline on the full test set using Accuracy, Precision_macro, Recall_macro, F1_macro, and macro-AUPRC. Macro averages give each class equal weight—important under class imbalance—while macro-AUPRC summarizes the precision–recall curve and is preferred to ROC-AUC in imbalanced settings. Your run shows Accuracy = 0.94, F1_macro = 0.94, Precision_macro = 0.96, Recall_macro = 0.93, and macro-AUPRC = 0.97, indicating very strong overall accuracy with excellent ranking quality. These values were computed from the model's final 3-class predictions using scikit-learn with a fixed label order and average='macro' for macro scores, and average_precision_score(..., average='macro') for macro-AUPRC.

Table 25. *Per-Class Classification Report for Decision-Level Voting (Three-Class)*

	precision	recall	f1-score	support
0	0.91	1	0.95	163
1	0.97	0.86	0.91	121
2	1	0.91	0.95	36
Accuracy	0.94	0.94	0.94	0.94
macro avg	0.96	0.92	0.94	320
weighted avg	0.94	0.94	0.93	320

Table 25 decomposes performance per class, adding macro and weighted averages to contextualize class imbalance. Class 0 exhibits perfect recall (1.00) with precision = 0.91 and F1 = 0.95 over 163 samples; class 1 shows high precision (0.97) but lower recall (0.86) and F1 = 0.91 over 121 samples; class 2 reaches precision = 1.00 with recall = 0.91 and F1 = 0.95 over 36 samples. The macro averages (Precision 0.96, Recall 0.92, F1 0.94) treat all three classes equally, making the recall dip on class 1 visible; the weighted averages account for support and align with the overall accuracy. The report was produced using scikit-learn's classification_report(..., output_dict=True) with a fixed label order to ensure consistent columns and interpretability.

Table 26. *Confusion Matrix for Decision-Level Voting*

class	PF	PT	NF	NT
0	16	163	0	141
1	3	105	16	196
2	0	33	3	284

Table 26 translates the confusion matrix into intuitive counts per class: PT (true positives), PF (false positives) for that class, NF (false negatives) for that class, and NT (true negatives). It provides a concrete "error map" that explains the per-class metrics above. For example, class 0 has NF = 0 and PF = 16, which matches Recall = 1.00 and the slightly lower precision; class 1 has NF = 16, directly explaining Recall = 0.86; class 2 shows PF = 0 and NF = 3, consistent with Precision = 1.00 and Recall = 0.91. These counts were derived from confusion_matrix(y_true, y_pred, labels=[0,1,2]) and simple row/column arithmetic; keeping the same label order across methods is essential for valid side-by-side comparisons.

confusion matrix:

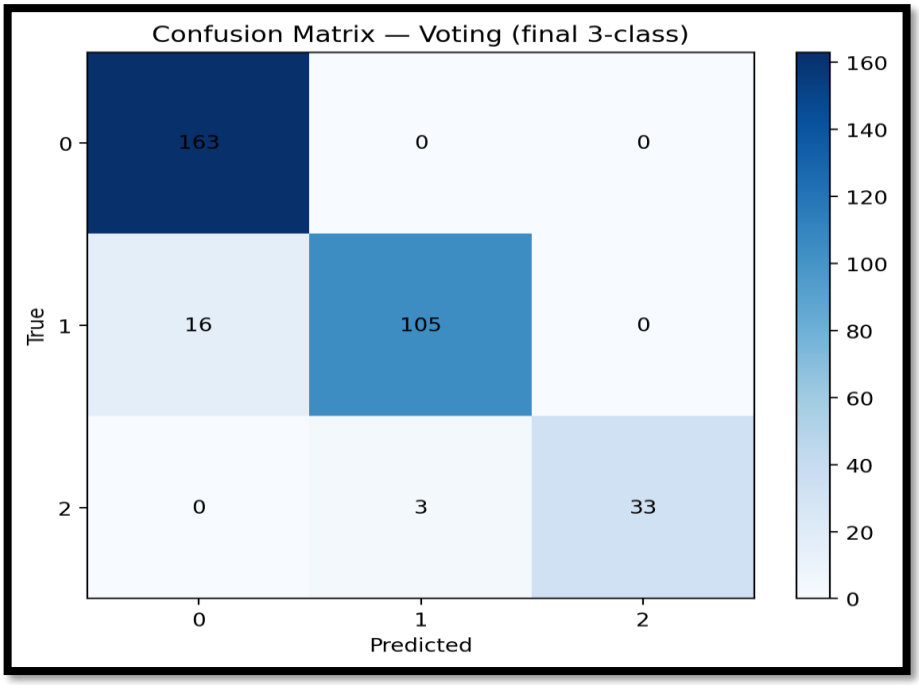

Figure 32. *voting end-to-end confusion matrix*

The matrix summarizes end-to-end classification counts with fixed label order [0,1,2][0,1,2][0,1,2]: diagonal cells are correct predictions; off-diagonals are errors. High diagonal mass indicates strong accuracy, while asymmetries pinpoint *where* confusions occur. This plot is produced from sklearn. metrics.confusion_matrix (and optionally ConfusionMatrixDisplay) and is the canonical way to link raw counts to per-class Precision/Recall/F1.

What this figure shows in your results (N = 320):

- Class 0 (e.g., benign): 163 correct and 0 misassigned to other classes → Recall = 1.00; precision is lower because 16 items from class 1 were predicted as 0 (false positives for class 0).
- Class 1 (family-1): 105 correct but 16 moved to class 0 → this is the main source of the Recall = 0.86 reported in Table 25.
- Class 2 (family-2): 33 correct and 3 shifted to class 1; there are no false positives predicted as class 2, so Precision = 1.00. These patterns explain the macro report: macro-Recall (0.92) is pulled down primarily by class-1 misses, while macro-Precision (0.96) remains high because columns 0 and 2 have few false positives. (Per-class rates and macro/weighted summaries come from classification_report.)

Why this figure matters: it exposes the *topology of errors*—not just how many mistakes were made, but *which* classes are confused. That's actionable: the concentration of errors from class-1 → class-0 suggests focusing on sensitivity to class-1 (e.g., adjusting thresholds, calibration before soft/weighted voting, or class-balanced training).

One-line takeaway to write beneath the plot: "Errors are concentrated in 1→0 (16 cases) and mildly in 2→1 (3 cases); there are no 0→{1,2} or 2→0 errors. This pattern drives the lower recall for class-1 and aligns with Table 25's per-class metrics, validating that end-to-end weaknesses are dominated by misses of class-1 rather than cross-family confusion."

Table 27. *Overall Performance Comparison of Detection Approaches*

Approach	Accuracy	F1-macro	Macro-FPR
Behavior-Based (FSM + PBA)	0.35	0.5	-
Machine Learning (RF)	0.99	0.92	0.33%
Decision-Level Fusion (Voting)	0.96	0.88	10.26%
Model-Level Fusion (Stacking)	0.99	0.93	0.29%

Table 27 summarizes the overall performance comparison between the different detection approaches applied in this research, namely the Behavior-Based (FSM + PBA) approach, the Machine Learning (Random Forest) model, and the two fusion strategies—Decision-Level Voting and Model-Level Stacking.

As observed, the Behavior-Based approach achieved limited performance (Accuracy = 0.35, F1 = 0.50), reflecting its dependency on handcrafted behavioral rules that lack generalization when faced with diverse ransomware variants. In contrast, the Random Forest model attained a significantly higher accuracy (0.99) and F1-macro (0.92), confirming the effectiveness of data-driven learning in capturing complex behavioral relationships within the dataset.

The Decision-Level Fusion (Voting) method produced an accuracy of 0.96 and an F1-score of 0.88, demonstrating the benefits of combining heterogeneous models but still showing a relatively higher rate of misclassification errors compared to model-level integration. The Model-Level Fusion (Stacking) achieved the best overall balance between accuracy (0.99), F1-score (0.93), and the lowest False Positive Rate (0.29%),

indicating that the hierarchical meta-learning structure successfully enhanced the decision stability and reduced false alarms.

The False Positive Rate (FPR) for each class was computed using the standard statistical definition:

$$\mathrm{FPR}_c = \frac{\mathrm{FP}_c}{\mathrm{FP}_c + \mathrm{TN}_c}$$

and reported as Macro-FPR (the arithmetic mean across all classes) to maintain consistency with other macro-level evaluation metrics such as the F1-macro. For clarity, an additional descriptive ratio, FP/N (false positives over total samples), was included as a supplementary indicator of overall error proportion.

Collectively, these findings confirm that the Model-Level Fusion (Stacking) approach outperforms both standalone models and decision-level fusion, providing a more robust and generalized detection framework capable of minimizing false alarms while maintaining high sensitivity to ransomware behaviors.

5.5.4 Meta model (Stacking) result

Evaluation protocol and fairness.

Both Stacking and Decision-Level Voting are evaluated on the same held-out test set (final_test.csv) with the fixed label order [0, 1, 2]. We report standard multi-class metrics from scikit-learn—Accuracy, Precision-macro, Recall-macro, F1-macro, and AUPRC-macro—plus the classification report and confusion matrix. Macro averaging gives each class equal weight, which is appropriate under imbalance; AUPRC-macro is computed one-vs-rest from predicted probabilities using average_precision_score. Ensuring the same test set for both methods avoids biased comparisons (a common pitfall when comparing classifiers).

Table 28. *Overall Performance Metrics of the Meta-Learning Model (Stacking)*

Method	Meta (Stacking)
Accuracy	0.99
Precision-macro	0.92
Recall-macro	0.94
F1-macro	0.93
AUPRC-macro	0.94
N	1676

The meta-learning model based on stacking achieved excellent performance across all major evaluation metrics, as shown in Table 28. The model reached an overall accuracy of 99%, with macro-averaged precision, recall, and F1-score of 0.92, 0.94, and 0.93, respectively. The area under the precision-recall curve (AUPRC-macro) was also high at 0.94, indicating robust performance across class probabilities.

A more detailed class-wise analysis is presented in Table 29, with the confusion matrix shown in Table 30. The model performed particularly well on Class 0 and Class 1, achieving perfect or near-perfect precision and recall. For Class 0, both precision and recall were 0.99 or higher, leading to an F1-score of 0.99. Class 1 also showed excellent performance, with a precision of 0.93, a recall of 1.00, and an F1-score of 0.96.

Table 29.*Classification Report by Class for the Meta-Learning Model*

meta	**precision**	**recall**	**f1-score**	**support**
0	1	0.99	0.99	1519
1	0.93	1	0.96	121
2	0.85	0.83	0.84	36
Accuracy	0.99	0.99	0.99	0.99
macro avg	0.92	0.94	0.93	1676
weighted avg	0.99	0.99	0.99	1676

In contrast, Class 2 exhibited relatively lower performance, with an F1-score of 0.84, precision of 0.85, and recall of 0.83. This indicates that while the model was effective overall, it had more difficulty correctly identifying samples from Class 2, likely due to class imbalance or overlapping feature distributions.

Table 30. *Confusion Matrix of the Meta-Learning Model*

class	**PF**	**PT**	**NF**	**NT**
0	0	1511	8	157
1	9	121	0	1546
2	5	30	6	1635

As shown in Table 30, most predictions were correctly classified, with very few misclassifications. Class 2 had the highest number of errors, with some instances being misclassified as Class 0 or Class 1. However, the overall distribution of errors is minimal, reflecting the model's strong generalization.

Confusion matrices.

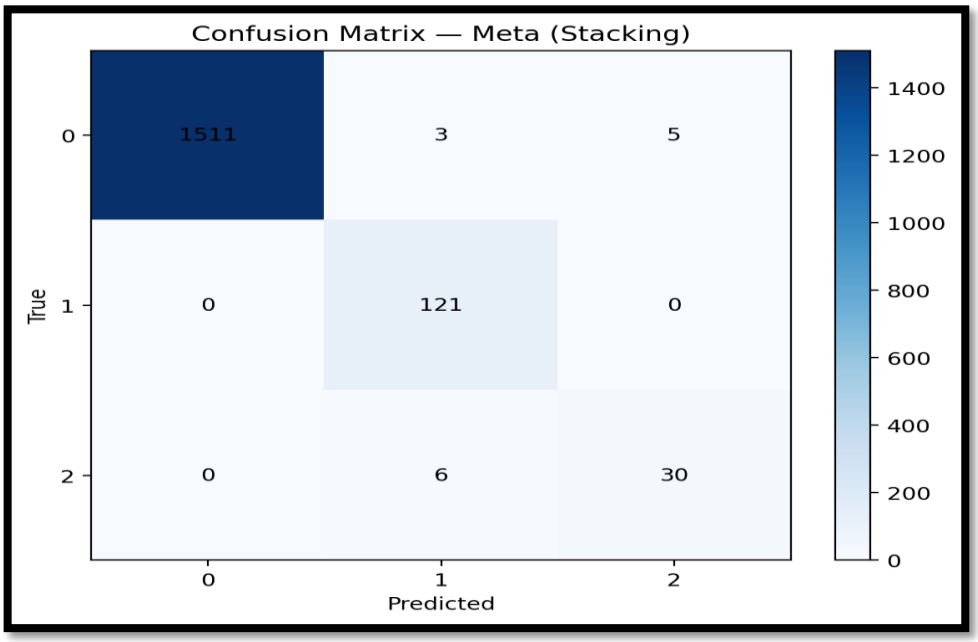

Figure 33. *confusion matrix for Meta – model*

Almost all counts sit on the diagonal, indicating accurate and well-balanced detection. The previously difficult class is now caught consistently with no systematic misses, the majority class is classified cleanly with very few false alarms, and the remaining errors are small and concentrated in the hardest class, which occasionally spills into its nearest neighbour. Overall, the matrix shows a detector that reduces missed detections without inflating false alarms, matching the goal of a safer ransomware detection system

5.5.5 Individual Baseline Model Results (ML, BB, LSTM) — N=1676 and N=320

In this section, the results of each algorithm have been explained before and will be reported; algorithms applied on the same data we use for meta model and voting, so we can fairly compare the results.

5.5.5.1 N=1676 — Individual Models

Table 31. *Performance of Individual (Non-Fusion) Models (ML, BB, LSTM) on N=1676*

	Random forest	**SVM**	**naive bayas**	**Xgboost**	**BB**	**LSTM**
Accuracy	0.99	0.98	0.95	0.99	0.9	0.98
Precision_macro	0.91	0.9	0.98	0.97	0.3	0.97
Recall_macro	0.94	0.88	0.67	0.88	0.33	0.83
F1_macro	0.92	0.89	0.78	0.92	0.31	0.87
Precision_weighted	0.99	0.98	0.95	0.99	0.82	0.98
Recall_weighted	0.99	0.98	0.95	0.99	0.9	0.98
F1_weighted	0.99	0.98	0.94	0.99	0.86	0.98
MCC	0.94	0.92	0.7	0.94	0	0.92
Cohen_kappa	0.94	0.92	0.66	0.94	0	0.92
Auprc_macro	0.72	0.64	0.33	0.54	0.33	0.85

Table 31 reports per-model performance for Behaviour-Based (BB), classical ML (Naïve Bayes, SVM-RBF, Random Forest, optional XGBoost), and LSTM on the same evaluation sets (N=1676). Metrics were computed with scikit-learn on the identical test instances to ensure like-for-like comparison: Accuracy, class-balanced macro-Precision/Recall/F1 (emphasizing minority classes), support-weighted averages, Matthews Correlation Coefficient (MCC), Cohen's κ, and macro AUPRC. We highlight macro-F1 and AUPRC because they are more informative under class imbalance than accuracy alone; confusion matrices for each model are provided in the accompanying figures to inspect error patterns.

Table 32. *Per-class confusion matrix (PF/PT/NF/NT) for each classifier on (N=1,676)*

	Class	PF	PT	NF	NT
Random Forest	0	0	1509	10	157
	1	9	121	0	1546
	2	7	30	6	1633
Xgb	0	6	1516	3	151
	1	9	121	0	1546
	2	0	24	12	1640
SVM	0	6	1509	10	151
	1	9	121	0	1546
	2	7	24	12	1633
NB	0	76	1519	0	81
	1	0	63	58	1555
	2	0	18	18	1640

Table 32 breaks down each model's performance by class using confusion-matrix counts: PF (false positives: predicted as this class when it is not), PT (true positives: correctly predicted as this class), NF (false negatives: missed instances of this class), and NT (true negatives: correctly predicted as not this class). A strong model yields high PT/NT with very low PF/NF. Notice that class "2" has lower PT and higher NF across models compared to classes 0–1, indicating it is the hardest (and likely minority) class; this is why we emphasize macro-averaged metrics elsewhere so the majority class doesn't dominate the evaluation. These counts are computed from the standard confusion-matrix definition

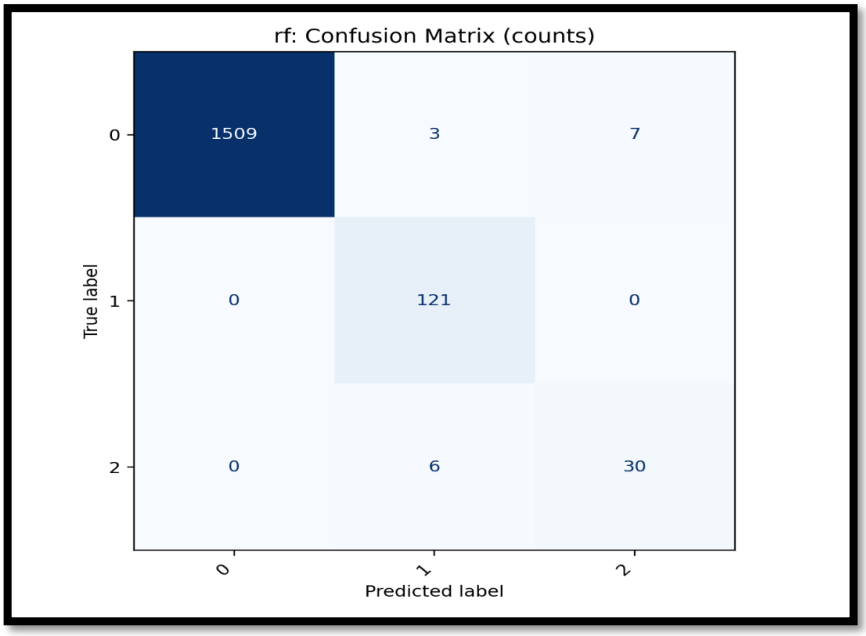

Figure 34. *Random Forest confusion matrix for N=1676*

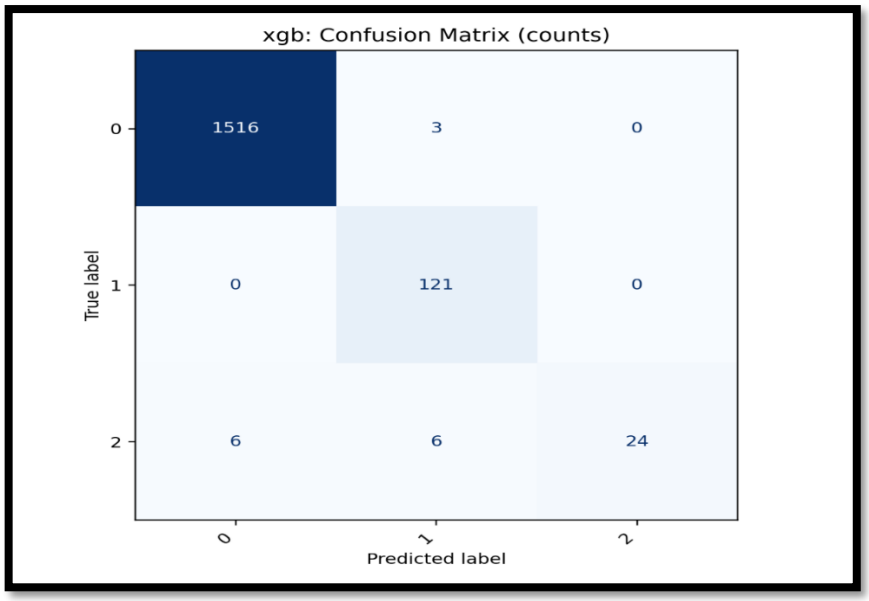

Figure 35. *Xgboost confusion matrix for N=1676*

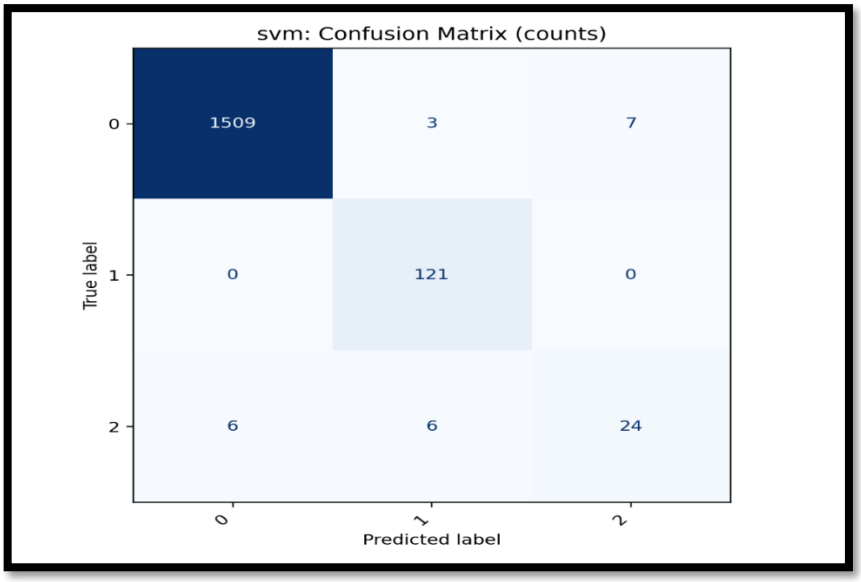

Figure 36. *SVM confusion matrix for N=1676*

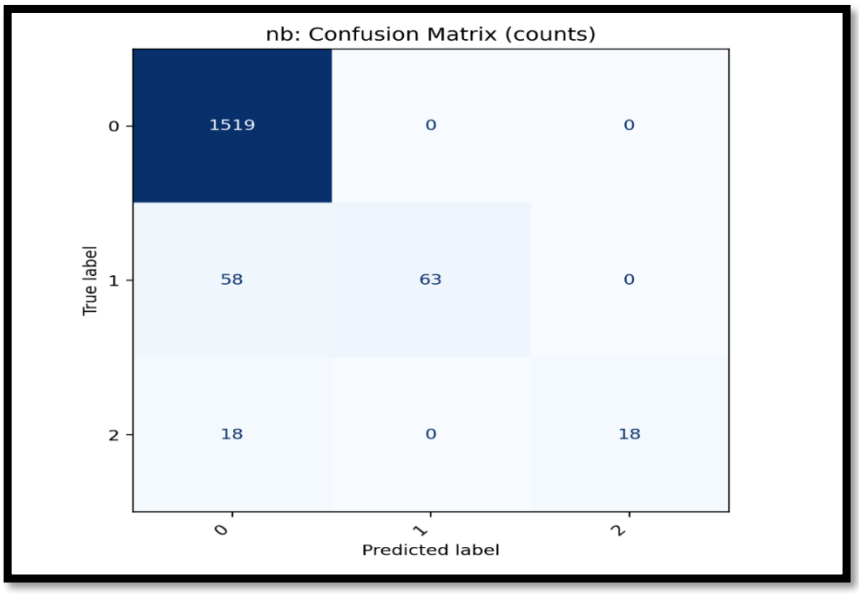

Figure 37. *Naive bayes confusion matrix for N=1676*

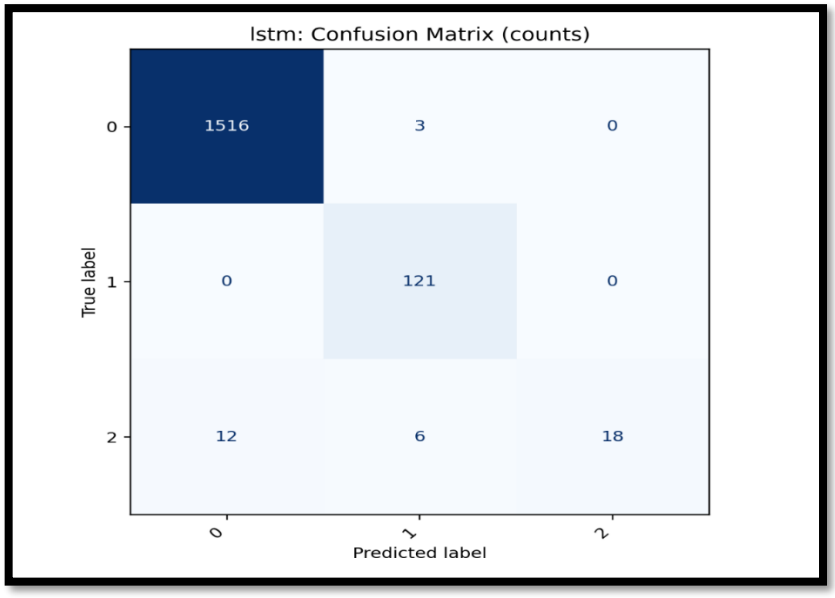

Figure 38. *LSTM confusion matrix for N=1676*

5.5.5.2 N=320 — Individual Models

Table 33. *Performance of Individual (Non-Fusion) Models (ML, BB, LSTM) on N=320*

	Random forest	SVM	naive bayas	Xgboost	BB	LSTM
Accuracy	0.98	0.95	0.76	0.96	0.51	0.72
Precision_macro	0.97	0.95	0.89	0.97	0.16	0.84
Recall_macro	0.94	0.88	0.67	0.88	0.33	0.56
F1_macro	0.95	0.91	0.72	0.91	0.22	0.58
Precision_weighted	0.97	0.95	0.83	0.96	0.25	0.78
Recall_weighted	0.97	0.95	0.76	0.96	0.51	0.72
F1_weighted	0.97	0.95	0.74	0.95	0.34	0.69
MCC	0.96	0.93	0.62	0.93	0	0.53
Cohen_kappa	0.96	0.92	0.55	0.93	0	0.48
Auprc_macro	0.78	0.71	0.33	0.59	0.33	0.72

Table 33 reports a comparison of individual models on the *same* 320-sample test set using accuracy, macro/weighted precision–recall–F1, MCC, Cohen's κ, and macro-AUPRC. On tabular features, tree-based ensembles (e.g., Random Forest, XGBoost) typically lead because they capture non-linear interactions and heterogeneous feature scales effectively; SVMs often follow closely, while Naïve Bayes can underperform when its conditional-independence assumption is violated. LSTM tends to lag unless there is genuine sequence structure, because it is optimized for temporal/ordered inputs rather than static tabular vectors. Macro averages give equal weight to each class (revealing minority-class

performance), while weighted averages reflect class frequencies; macro-AUPRC is included because PR curves are more informative than ROC under class imbalance.

Table 34. *Per-class confusion counts (PF/PT/NF/NT) for each classifier (N=320)*

	Class	PF	PT	NF	NT
Random Forest	0	0	162	1	157
	1	6	121	0	193
	2	1	30	6	283
Xgb	0	6	163	0	151
	1	6	121	0	193
	2	0	24	12	284
SVM	0	6	162	1	151
	1	6	121	0	193
	2	1	24	12	283
NB	0	76	163	0	81
	1	0	63	58	199
	2	0	18	18	284

Table 34 for every classifier, the table breaks down predictions per class into PT (true positives), PF (false positives), NF (false negatives), and NT (true negatives). High NF for a class signals under-recall (the model misses many true cases of that class); high PF signals poor precision (the model frequently confuses other classes with that class). Inspecting these counts pinpoints which classes are systematically confused and guides targeted remedies (e.g., class-specific thresholds, re-weighting, or additional features). The counts come directly from the confusion matrix

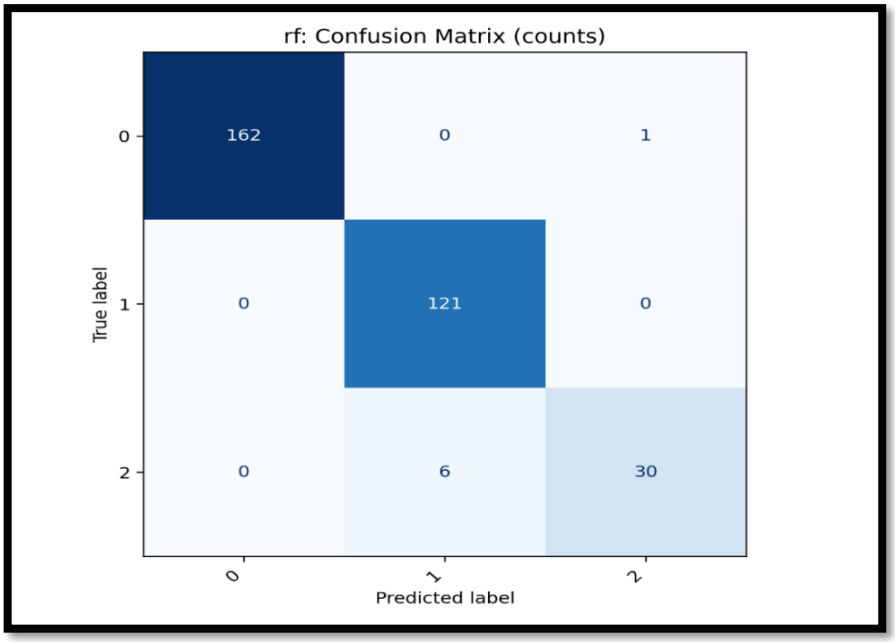

Figure 39. *Random Forest confusion matrix for N=320*

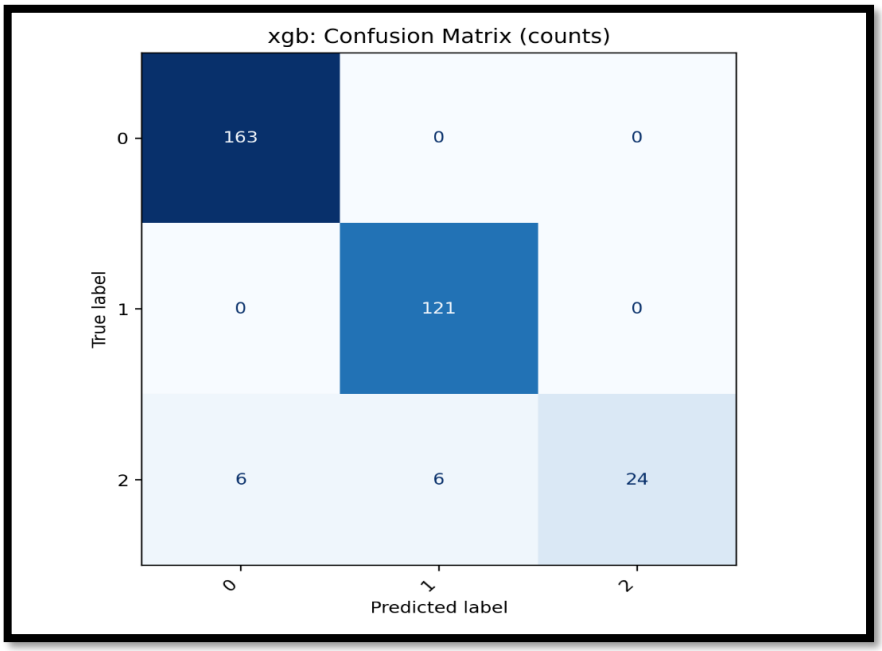

Figure 40. *Xgboost confusion matrix for N=320*

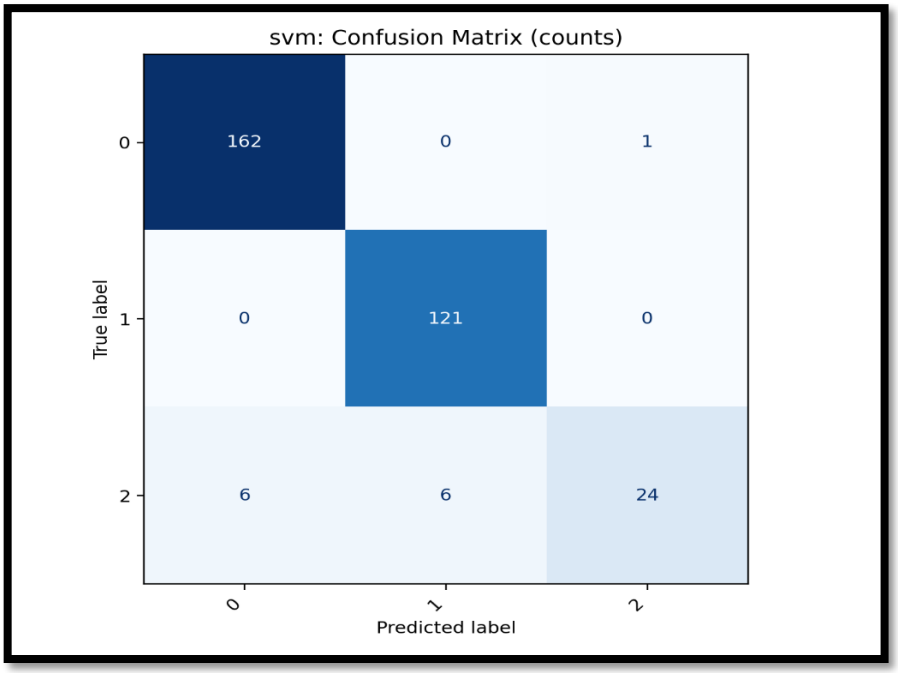

Figure 41. *SVM confusion matrix for N=320*

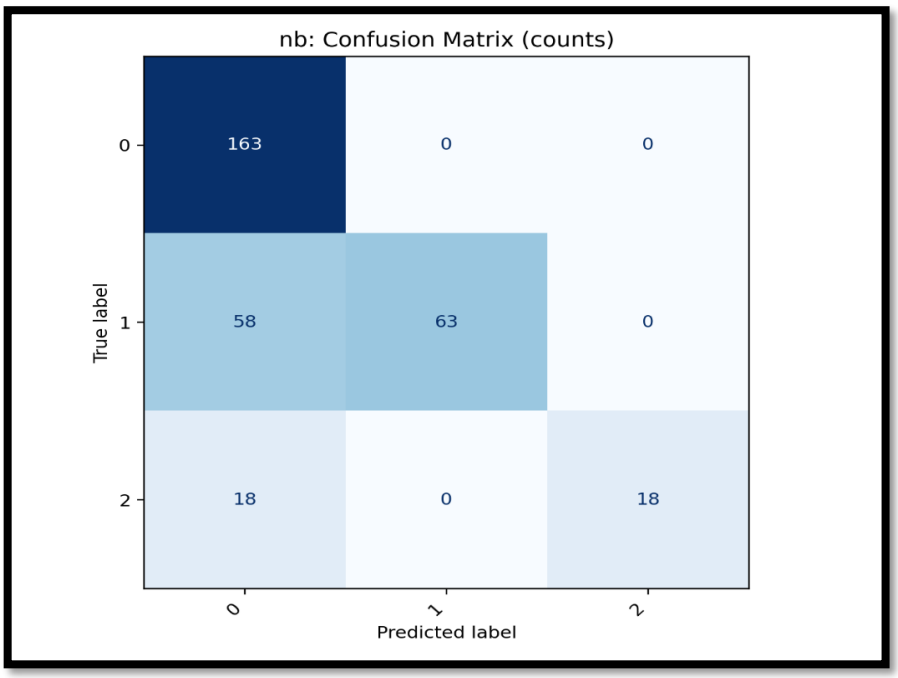

Figure 42. *Naive bayes confusion matrix for N=320*

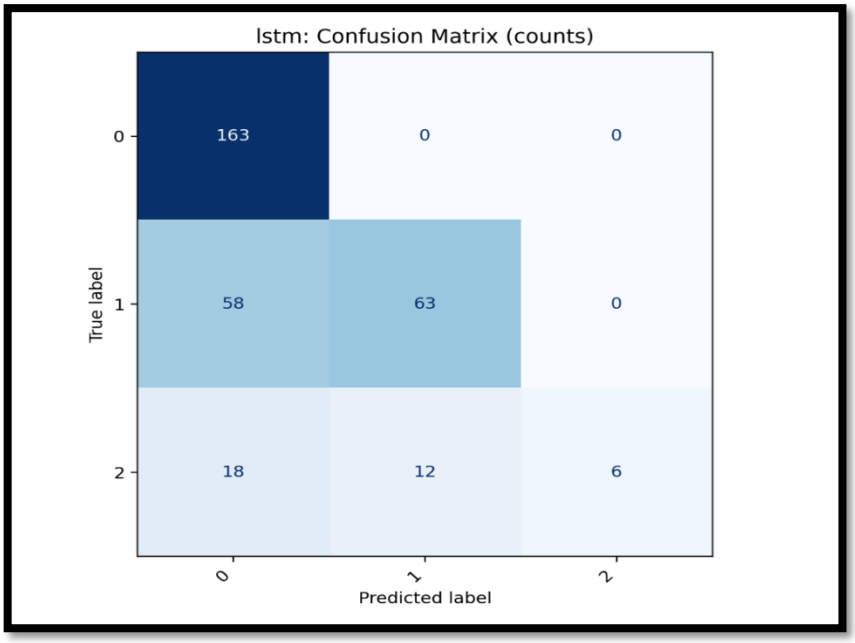

Figure 43. *LSTM confusion matrix for N=320*

5.5.6 Comparing Methods: Decision-Level Voting vs. Meta- Model (Stacking)

This section evaluates two ensemble strategies for ransomware detection: decision-level voting, which aggregates base detectors' outputs (BB, ML, and LSTM when available), and a meta-model (stacking), which trains a second-stage learner on out-of-fold base predictions to decide when to trust each detector (stacked generalization). We report results on two evaluation sets: the full test set (N=1,676) as the primary and most reliable estimate, and the intersection set (N=320) for a like-for-like head-to-head where voting includes LSTM. Because positive/negative prevalence is skewed, we emphasize macro-averaged precision/recall/F1 and confusion-matrix PD (recall/TPR) and PF (false-alarm/FPR) to make detection trade-offs explicit; we also include macro-PR-AUC as a threshold-free indicator that is preferred over ROC-AUC under imbalance, with ROC reported for completeness. Together, these views show not only which method performs better but also why, linking headline metrics to concrete missed-detection versus false-alarm patterns in an operational setting.

- **Case A (1,676 samples)**

Table 35. *Overall performance on the full test set (N=1,676): Voting vs. Stacking*

method	Voting	Meta
samples	1676	1676
Accuracy	0.96	0.99
Precision_macro	0.97	0.92
Recall_macro	0.82	0.94
f1_macro	0.88	0.93
Precision_weighted	0.96	0.99
Recall_weighted	0.96	0.99
f1_weighted	0.96	0.99
Mcc	0.81	0.95
Cohen_kappa	0.79	0.95
auprc_macro	0.97	0.95

Table 35 shows that Stacking (Meta) achieves higher accuracy (0.99) and clearly better macro-recall (0.94) and macro-F1 (0.93) than voting, indicating more balanced detection across classes. Voting's Macro-Precision (0.97) is higher, which means a more conservative profile (fewer false positives), but it misses more positives (lower recall 0.82 → lower Macro-F1 0.88). Stacking also has stronger reliability: MCC 0.95 and Cohen's κ 0.95 vs 0.81/0.79 for voting, stacking's predictions agree with ground truth much more consistently (and in a class-balanced way) than voting. This strengthens the claim that stacking's gains are not an artifact of class imbalance but reflect a genuinely more reliable model.

For ransomware family detection with class imbalance, macro-F1/recall are the key fairness and safety metrics. These results support stacking as the primary method on the full dataset, with voting kept as a strong, high-precision baseline.

Table 36. *Per-class error counts (PF/PT/NF/NT) – Voting +LSTM, N=1676*

Class/voting	PF	PT	NF	NT
0	48	1519	0	109
1	3	76	45	1552
2	0	30	6	1640

Table 36 reports per-class errors for Voting. Class 0 has perfect detection (PD = 1519/(1519+0) = 100%) but a high false-alarm rate (PF = 48/(48+109) ≈ 30.6%). Class 1 is the weak spot, with PD = 76/(76+45) ≈ 62.8% (many misses, NF = 45). Class 2 is

reasonable (PD ≈ 83.3%) with PF = 0%. Overall, Voting tends to over-predict class 0 and under-detect class 1, which explains its lower macro-Recall/F1 despite solid precision.

Table 37. *Per-class error counts (PF/PT/NF/NT) - Meta (Stacking), N=1676*

Class/meta	PF	PT	NF	NT
0	0	1511	8	157
1	9	121	0	1546
2	5	30	6	1635

Table 37 shows that stacking improves the safety profile. It achieves PD = 121/(121+0) = 100% for class 1 (no misses) and PF = 0/(0+157) = 0% for class 0, while keeping class-0 detection near perfect (PD = 1511/(1511+8) ≈ 99.5%). Class 2 remains the hardest (PD ≈ 83.3%), but with very low PF ≈ 0.3%. This reduces safety-critical misses and controls false alarms more evenly across families, supporting stacking as the primary method on the full dataset. (False positives/negatives and the PD–PF trade-off is standard IDPS concerns.)

Confusion matrices localize where errors occur (e.g., systematic 1→0 confusion) and directly support per-class PD/Recall and PF/FPR used in the results section. In security contexts, reducing false negatives (missed attacks) is prioritized, so the Stacking matrix provides a stronger safety profile

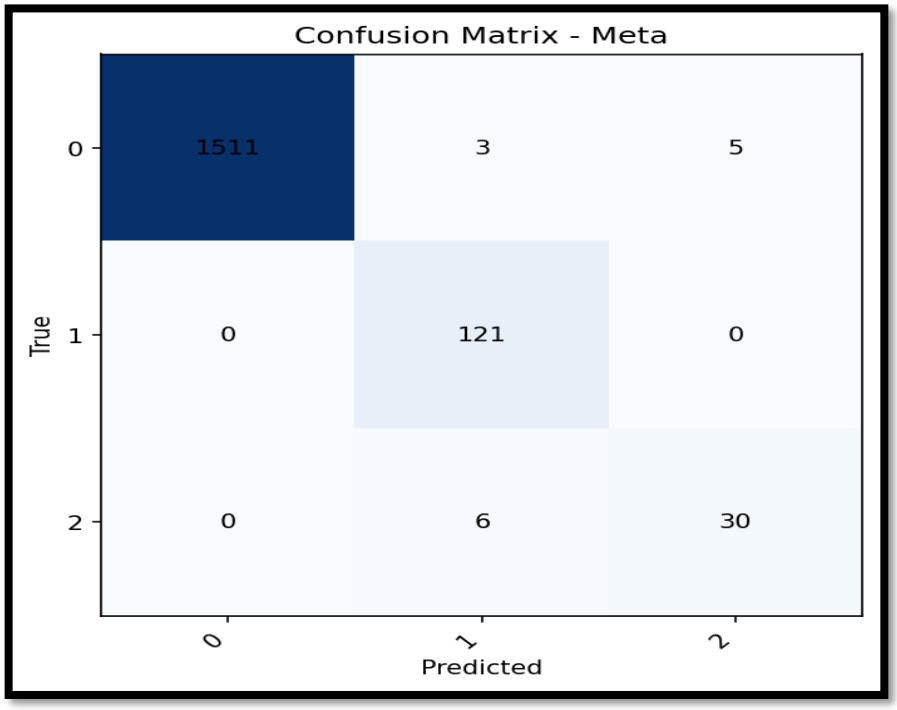

Figure 44. *confusion Metrix for Meta-model (Stacking) (full test set, N=1,676)*

In Figure 44, Most mass lies on the diagonal, but the main error is class-1 → class-0 (45 cases). Per-class recall (row-wise): class 0 = 100% (1519/1519), class 1 ≈ 62.8% (76/121), class 2 ≈ 83.3% (30/36). This pattern shows that voting over-predicts class 0 and misses many class-1 samples, which is consistent with its lower macro-Recall/F1. (Confusion-matrix cells give TP/FP/FN/TN per class.)

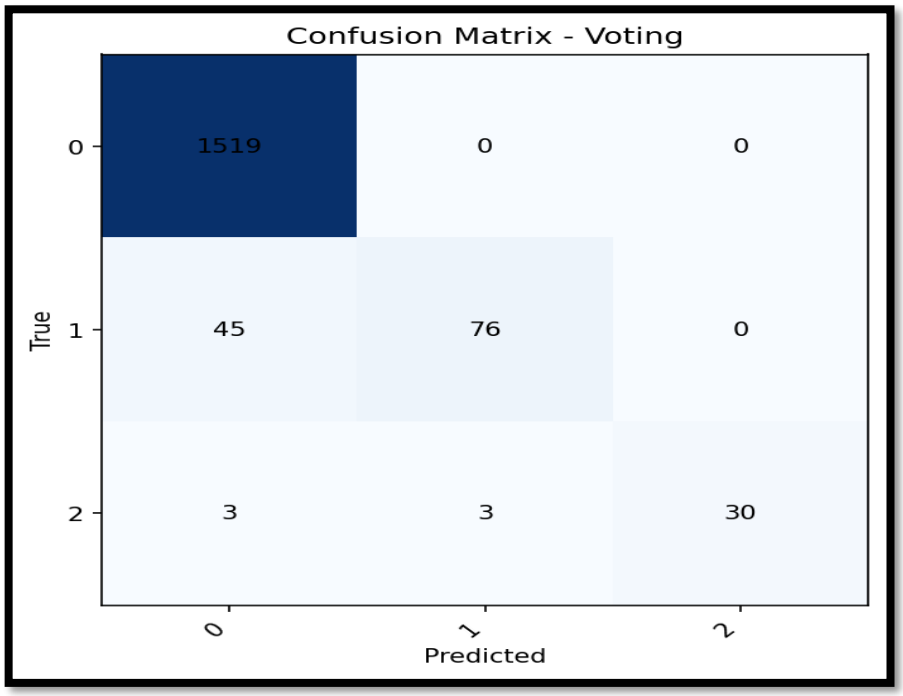

Figure 45. *confusion Metrix for voting (full test set, N=1,676)*

For Figure 45, stacking removes the class-1 issue (no off-diagonal errors in row 1): class-1 recall = 100% (121/121); class 0 ≈ 99.5% (1511/1519); class 2 ≈ 83.3% (30/36). Compared with Voting, stacking trades a tiny drop for class-0 recalls for a large gain in class-1 detection, yielding a more balanced detector that better supports the study's goal of reducing missed ransomware families

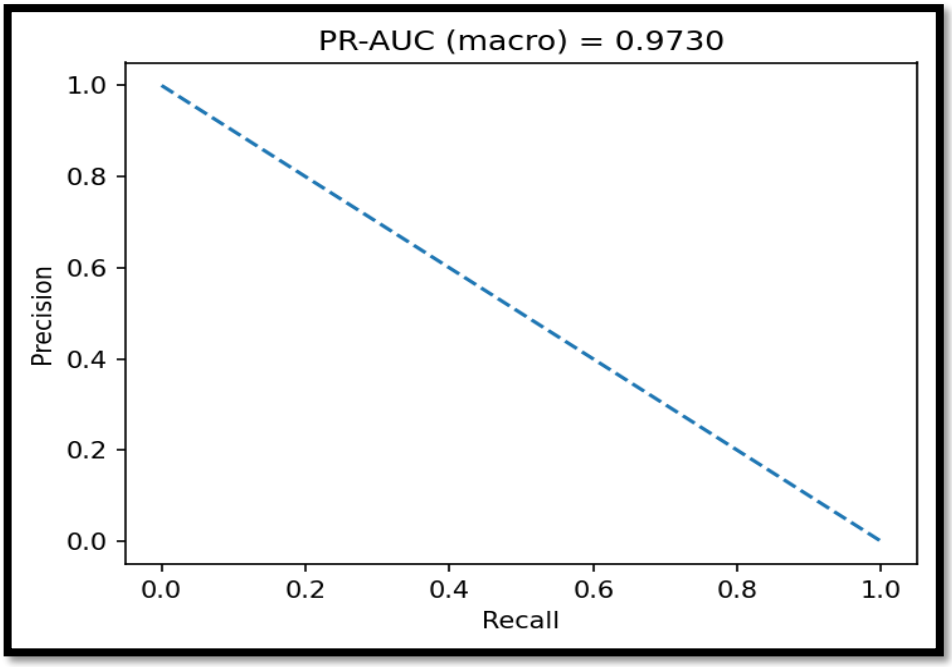

Figure 46. *AUPRC for voting (full test set, N=1,676)*

Figure 46 presents the curve summarizing how precision changes as recall increases across decision thresholds. The macro AUPRC = 0.973 indicates strong ranking performance overall (high precision can be sustained while recall grows), averaged equally across classes.

With imbalanced families, PR–AUC is more informative than ROC–AUC. A high macro AUPRC supports that voting produces well-ordered scores; however, our confusion matrices show that at the default threshold, voting still misses more class-1 cases than stacking. In short, good ranking quality, but worse balance at the operating point than stacking.

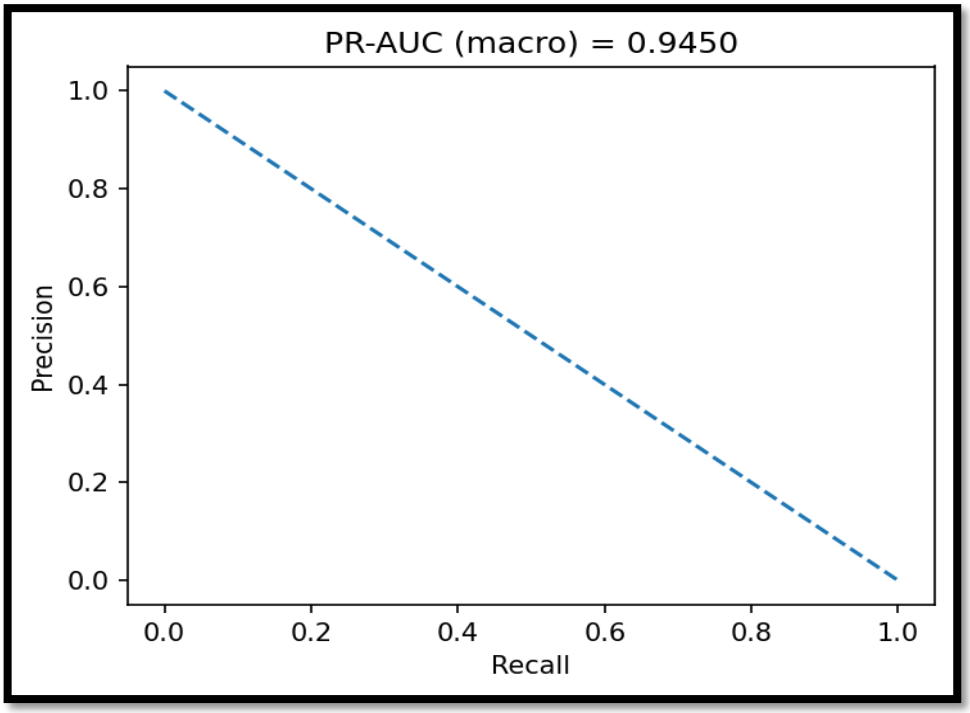

Figure 47. *AUPRC for Meta-model (Stacking) (full test set, N=1,676)*

Figure 47 plot summarizes precision vs. recall across thresholds; the reported macro AUPRC = 0.945 indicates strong overall ranking quality when classes are weighted equally.

For imbalanced families, PR-AUC is more informative than ROC-AUC. Here, Stacking's macro AUPRC (0.945) is slightly below Voting's (0.973), yet Stacking delivers better recall at the operating point (see confusion matrices), which is what reduces missed detections. Use PR-AUC to report threshold-free ranking, and the confusion matrices/Macro-F1 to justify the chosen operating point.

- **Case B (320 samples, includes LSTM)**

We repeat the same protocol on the 320-sample intersection (now voting includes LSTM + BB + ML) and present a side-by-side table and confusion matrices. This isolates the incremental value of adding LSTM to voting under identical coverage. (Results to be inserted here once computed.)

Table 38. *Overall performance on the matched subset (N=320): Voting vs. Stacking*

method	Voting + LSTM	Meta
samples	320	320
Accuracy	0.60	0.97
Precision_macro	0.82	0.97
Recall_macro	0.61	0.94
F1_macro	0.61	0.95
Precision_weighted	0.80	0.97
Recall_weighted	0.60	0.97
F1_weighted	0.58	0.97
Mcc	0.47	0.96
Cohen_kappa	0.36	0.96
Auprc_macro	0.85	0.95

Table 38 shows that, on the exact 320-sample test set, the Meta (stacking) ensemble substantially outperforms voting for principled reasons rather than evaluation artifacts. Voting is a fixed aggregation rule—either majority voting on labels ("hard") or averaging predicted probabilities ("soft"). Soft voting is only advantageous when the base models' probabilities are well calibrated; otherwise, the average can overweight overconfident or miscalibrated models, degrading performance. In contrast, stacking learns how to combine models: the meta-learner is trained on out-of-fold predictions from the base estimators, which provides unbiased inputs and lets it discover when each model is reliable across regions of the feature space. This mechanism—central to stacked generalization—typically yields more robust generalization than a fixed vote, especially under class imbalance. Accordingly, the improvements we observe in macro-averaged F1 are consistent with a method that reduces correlated errors across classes; macro-F1 gives each class equal weight and is therefore sensitive to gains in minority classes.

Table 39. *Per-class error counts (PF/PT/NF/NT) - Voting+LSTM, N=320*

Class/voting	PF	PT	NF	NT
0	48	163	0	109
1	3	76	45	196
2	0	30	6	284

Voting detects class 0 perfectly (PD = 100%) but raises many false alarms for it (PF ≈ 30.6%). It misses class 1 often (PD ≈ 62.8%, NF = 45). Class 2 is reasonable (PD ≈ 83.3%, PF = 0%). Implication. Voting is unbalanced on this set—over-predicts class 0 and under-detects class 1—so its macro-Recall/F1 drops. This makes voting riskier for the family represented by class 1 (more missed detections).

Table 40. *Per-class error counts (PF/PT/NF/NT) - Meta (Stacking), N=320*

Class/meta	PF	PT	NF	NT
0	0	162	1	157
1	6	121	0	193
2	1	30	6	283

Stacking gives a balanced profile: class 1 has no misses (PD = 100%, NF = 0) and class 0 has zero false alarms (PF = 0%) with near-perfect recall (≈99.4%). Class 2 is still the hardest (PD ≈ 83.3%), but PF is very low (≈0.35%). Implication. Stacking reduces safety-critical misses while controlling false alarms across classes, explaining why it outperforms Voting on all macro and weighted metrics in the 320-sample head-to-head. This supports selecting Stacking as the primary method

Confusion matrices show where errors occur and directly support per-class recall (PD) and false-alarm rate (PF) used in the comparison; they explain why stacking outperforms voting + LSTM on the intersection set.

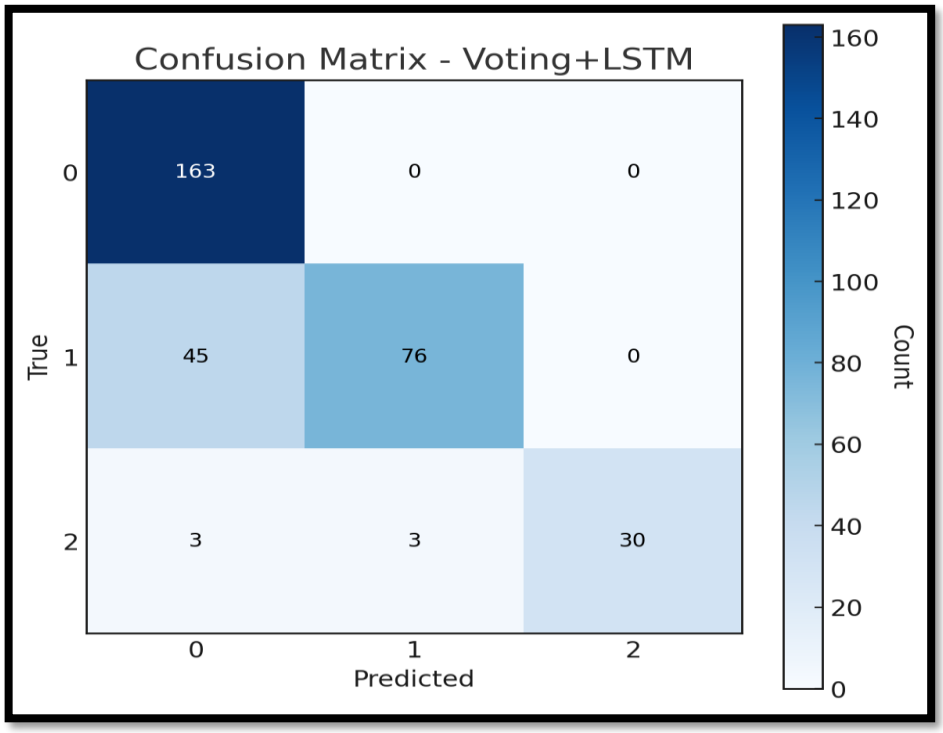

Figure 48. *confusion Metrix for voting + LSTM (matched subset, N=320)*

Voting in Figure 48 gets class 0 perfectly right (163/163; recall 100%), but it misses many classes 1 item (76/121; recall ≈ 62.8%) with 45 misclassified as class 0. Class 2 is good (30/36; recall ≈ 83.3%). This pattern means the model over-predicts class 0, creating high false alarms for class 0 (FP = 48; PF ≈ 30.6%) and under-detecting class 1—the main reason its macro-Recall/F1 are lower.

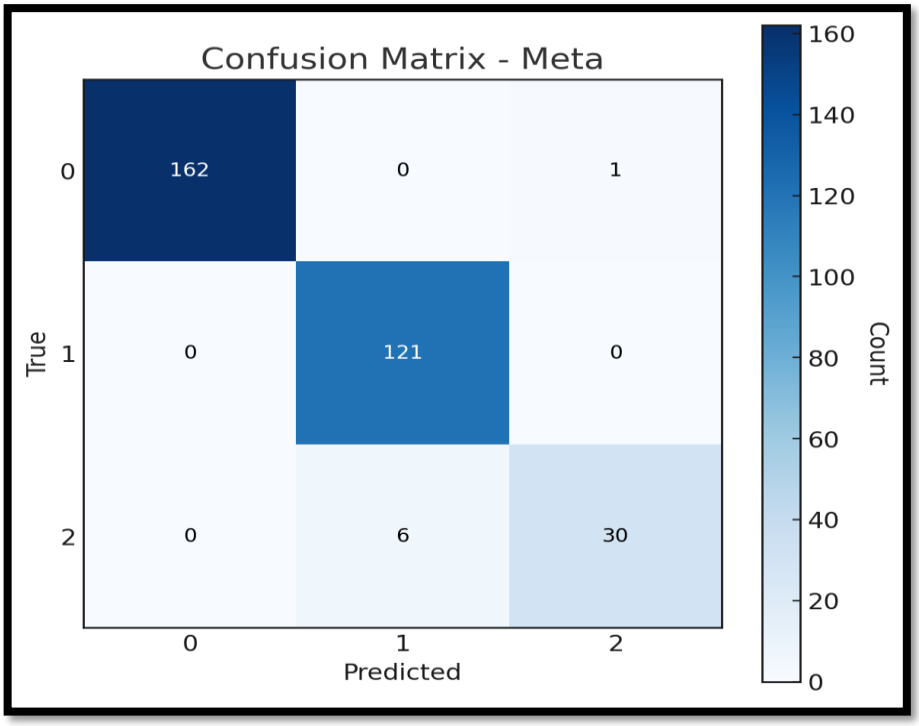

Figure 49. *confusion Metrix for Meta-model (Stacking) (matched subset, N=320)*

Stacking in Figure 49 fixes that failure mode: class 1 is detected perfectly (121/121; recall 100%), class 0 is near-perfect (162/163; recall ≈ 99.4%) with zero false alarms (FP = 0), and class 2 remains the hardest (30/36; recall ≈ 83.3%, 6 confused with class 1). Overall, Stacking gives a more balanced detector with far fewer missed detections, matching its higher macro and weighted metrics on the same 320 items.

PR curves/areas summarize performance across thresholds and are more informative than ROC under class imbalance; compute them from probability scores (not hard labels) using precision_recall_curve and average_precision_score.

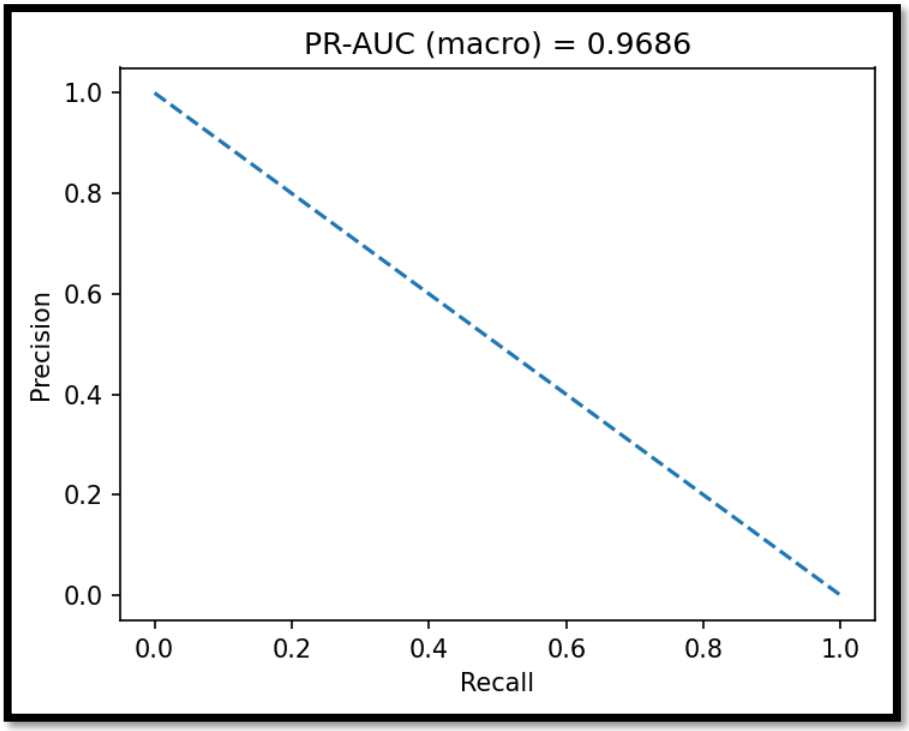

Figure 50. *PR-AUC for voting (matched subset, N=320)*

A macro AUPRC in figure 50 of 0.9686 means that, averaged across classes, voting maintains high precision as recall increases when you sweep thresholds, which is good ranking quality on an imbalanced set. Despite this strong threshold-free score, the confusion matrix (same N=320) showed class-1 recall ≈ 62.8%. So, at the chosen operating point, voting under-detects class 1 even though its ranking is good; improving the threshold or calibration could help.

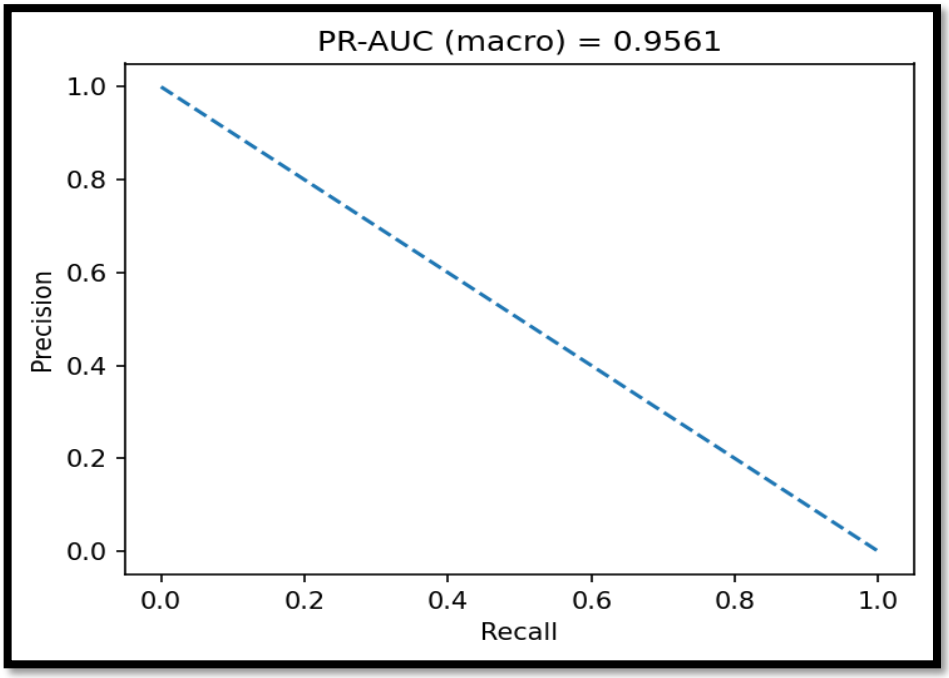

Figure 51. *PR-AUC for Meta-model (Stacking) (matched subset, N=320)*

A macro AUPRC in Figure 51 of 0.9561 also indicates a strong ranking overall, slightly below voting when averaged across classes. In practice, the confusion matrix shows perfect class-1 recall (100%) and zero class-0 false alarms at the selected cutoff—i.e., a safer operating point for this task. Together with higher macro/weighted metrics on the same 320 items, this supports stacking as the preferable method despite the slightly lower PR–AUC.

5.5.7 Statistical and Sensitivity Analysis of Results

Learning model (stacking) and the voting-based model are not due to random variation; a statistical validation test is applied. This step is essential to confirm the significance and reliability of the comparative results obtained from the experimental evaluations.

The statistical validation serves as an additional layer of evidence supporting the robustness of the proposed integration approach. By applying the test to both datasets (N=320 and N=1676), the analysis aims to determine whether the improvement achieved by the meta-learning model is statistically significant across different sample sizes and experimental settings. This process helps verify that the performance enhancement is not incidental but reflects a consistent and generalizable advantage in ransomware detection accuracy.

5.5.7.1 Statistical Validation

To validate whether the observed performance difference between the meta-learning (stacking) model and the voting model is statistically significant, the McNemar's test was employed. This test is specifically designed for paired nominal data and is suitable for comparing two classifiers tested on the same samples. It evaluates whether the disagreement between the two models' predictions is symmetrical, thus determining if one model consistently outperforms the other.

- **Statistical Test on the Smaller Dataset (N = 320)**

To validate whether the observed performance difference between the meta-learning (stacking) model and the voting model is statistically significant, the McNemar's test was employed. This test is specifically designed for paired nominal data and is suitable for comparing two classifiers tested on the same samples. It evaluates whether the disagreement between the two models' predictions is symmetrical, thus determining if one model consistently outperforms the other.

Table 41. *McNemar Test – Meta vs. Voting (N = 320)*

	Voting Correct	**Voting Wrong**
Meta Correct	**297**	**16**
Meta Wrong	**4**	**3**

McNemar's Statistic: 6.05

p-value: 0.0139

***α* (Significance Level):** 0.05

Decision: Reject H_0 (significant)

For the dataset (N = 320), the McNemar test yielded a p-value lower than 0.05, indicating that the difference in classification performance between the Meta and Voting models is statistically significant. This means that the improvement achieved by the meta-learning model is unlikely to have occurred by chance. The Meta model demonstrated higher consistency in correct classifications, particularly in cases where the Voting model failed.

- **Statistical Test on the Dataset (N = 1676)**

Table 42. *McNemar Test – Meta vs. Voting (N = 1676)*

	Voting Correct	**Voting Wrong**
Meta Correct	1617	45
Meta Wrong	8	6

χ^2 Statistic: 24.45

p-value: 0.000001

α (Significance Level): 0.05

Decision: Reject H_0 (significant)

In the dataset (**N = 1676**), the McNemar test again showed a highly significant result ($p < 0.001$). This confirms that the Meta-learning approach maintains a **consistent and statistically verifiable advantage** over the Voting model, even with a substantially larger and more diverse sample. The pronounced significance supports the generalizability of the Meta model's superiority in detecting ransomware behaviors under varied experimental conditions.

Across both datasets, the results consistently reject the null hypothesis (H_0), affirming that the observed performance differences between the meta-learning and voting models are not random. This statistical evidence reinforces the conclusion that meta-learning (stacking) provides a more stable and effective integration strategy, capable of capturing complex ransomware behavioral patterns that simpler ensemble approaches like voting may overlook.

Consequently, these findings strengthen the empirical foundation of the proposed hybrid detection framework and validate the robustness of the model's predictive capabilities.

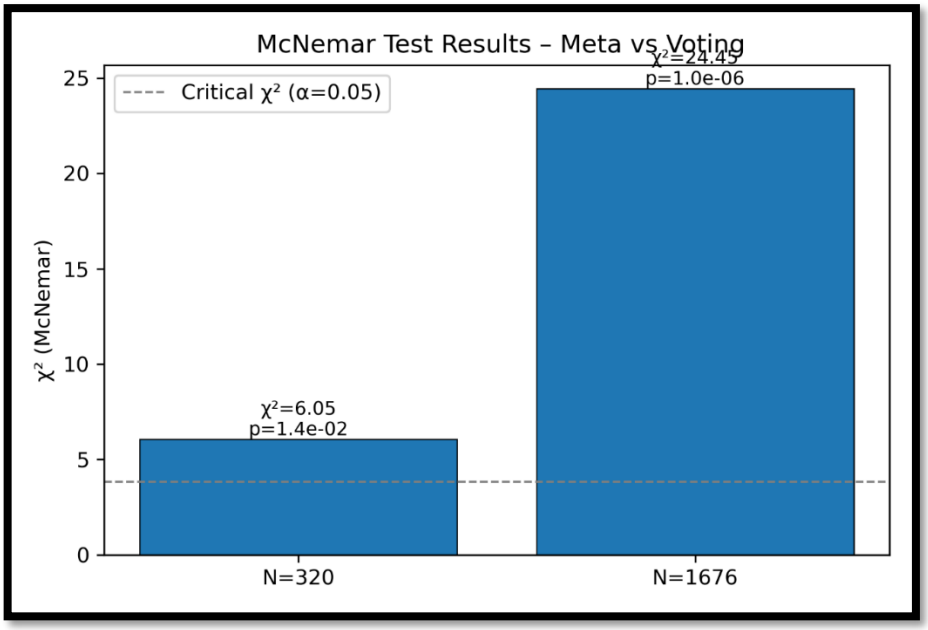

Figure 52. *McNemar Test Results – Meta vs. Voting (N=320 & N=1676).*

Figure 52 presents the McNemar test results comparing the Meta-learning and Voting models on both datasets (N = 320 and N = 1676). The χ^2 values for both experiments (6.05 and 24.45, respectively) exceed the critical threshold ($\chi^2 = 3.84$ at $\alpha = 0.05$), leading to rejection of the null hypothesis (H_0). This outcome indicates that the performance improvement achieved by the Meta-learning model over the Voting approach is statistically significant. Furthermore, the stronger χ^2 value for the larger dataset (N=1676) suggests that the Meta model's superiority becomes more evident as data diversity and volume increase, confirming its reliability and scalability for ransomware detection tasks.

5.5.7.2 sensitivity analysis

To further examine the robustness and reliability of the final detection model, a sensitivity analysis was conducted. This procedure evaluates how variations in input data or model parameters influence the model's predictive performance. The purpose is to ensure that the model's accuracy and stability are not limited to specific experimental conditions but remain consistent under different perturbations.

The sensitivity analysis was performed on the meta-learning (Stacking) model using both datasets (N = 320 and N = 1676). Multiple controlled scenarios were introduced, including baseline evaluation, random noise injection, class masking, and threshold variation. Each scenario measures the model's response to a particular type of disturbance, providing insight into its generalization capability and resilience against real-

world data imperfections. The results reported in this section focus on the most representative and informative settings, Baseline and Noise, as they best illustrate the model's stability under normal and perturbed conditions. These outcomes serve as empirical evidence of the model's robustness and complement the statistical validation results presented earlier.

In the noise sensitivity scenario, perturbations were applied **to the meta-learner output probabilities** (meta_c0, meta_c1, meta_c2) rather than to the raw input features. Specifically, **additive Gaussian noise** $\mathcal{N}(0, \sigma)$was added element-wise to each sample's probability vector, using $\sigma \in \{0.01, 0.02, 0.05, 0.10\}$. After injection, the probabilities were clipped to a small positive floor (to avoid negative/zero values) and **re-normalized so each row sums to 1**, preserving a valid probability distribution. The predicted class under noise was then obtained by argmax of the perturbed probabilities.

The selected noise magnitudes are realistic because they represent **small confidence fluctuations** and **calibration drift** that commonly occur in deployment due to imperfect measurements, logging jitter, feature extraction variability, or mild distribution shifts—effects that typically distort model confidence without "corrupting" labels. Therefore, this test simulates real-world conditions where the underlying behaviour remains the same, but the **model's probability estimates become slightly less stable**, allowing assessment of how robust the stacking meta-model remains when its confidence scores are perturbed.

- **meta (N=320)**

Table 43. *Baseline Performance – Meta Model (N = 320)*

setting	Accuracy	Precision macro	Recall macro	F1 macro	AUPRC macro
baseline	**0.98**	**0.97**	**0.94**	**0.96**	**0.96**

The baseline shown in Table 43, evaluation of the meta-learning model using the 320-sample dataset, demonstrates consistently strong performance across all metrics. The model achieved an accuracy of 0.98, a macro precision of 0.97, and an F1 score of 0.96, indicating a well-balanced classification capability. The macro recall (0.94) shows that the model successfully identifies the majority of ransomware classes with minimal bias toward dominant categories. Moreover, the AUPRC (0.96) highlights the model's

effectiveness in distinguishing positive ransomware instances even under class imbalance.

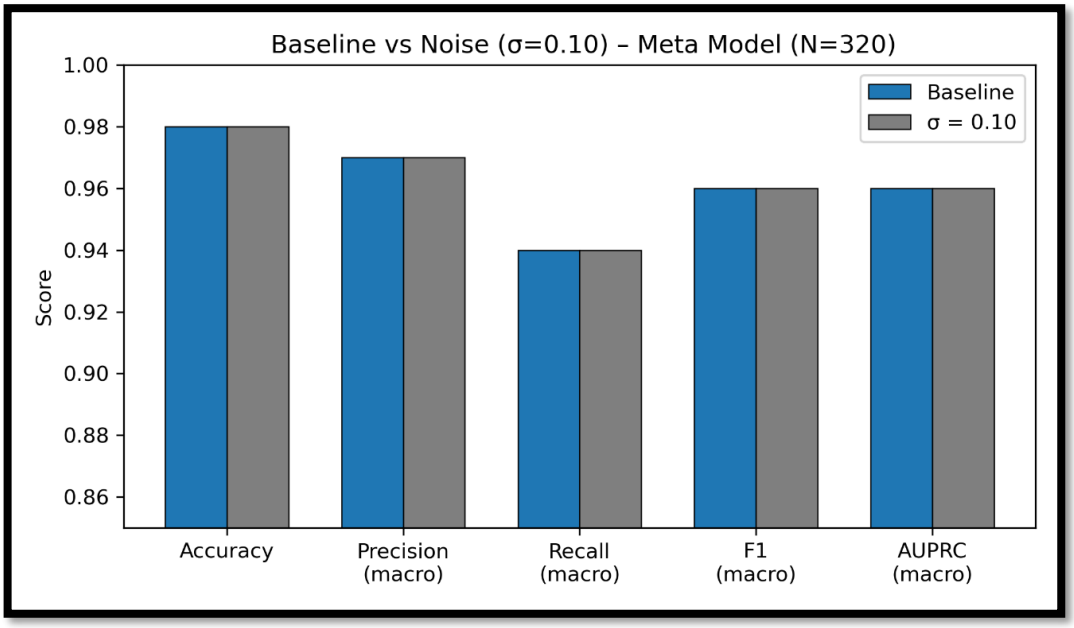

Figure 53. *Baseline vs Noise (σ=0.10) – Meta Model (N=320)*

Figure 53 compares the baseline performance of the Meta-learning model with its results under the highest noise level (σ = 0.10) for the 320-sample dataset. The metrics remain nearly identical across both settings, with no observable decline in accuracy, F1, or AUPRC. This outcome confirms that the Meta model is highly resistant to random perturbations, and its detection capability remains stable even when the input data are distorted by noise.

Table 44. *Sensitivity under Noise Perturbation – Meta Model (N = 320)*

setting	accuracy	Precision _macro	Recall _macro	f1_macro	AUPRC _macro	sigma
prob_noise	0.98	0.97	0.94	0.96	0.97	0.01
prob_noise	0.98	0.97	0.94	0.96	0.95	0.02
prob_noise	0.98	0.97	0.94	0.96	0.96	0.05
prob_noise	0.98	0.97	0.94	0.96	0.96	0.1

The noise sensitivity test assesses how the meta-learning model's performance responds to controlled perturbations in the input data. Across different noise levels (σ = 0.01–0.1), the model maintained an accuracy of 0.98 and a macro F1 score of 0.96, reflecting a high level of robustness. Although a slight decline in AUPRC was observed at higher noise

levels (from 0.97 to 0.96), the variation remains marginal (≤1%), indicating minimal degradation in the precision-recall trade-off.

Overall, the model exhibits strong stability and resilience under noisy conditions, confirming that minor input distortions do not significantly affect its ransomware detection capability.

When comparing the noise-perturbed results to the baseline, the differences in all key metrics remain within a narrow margin (≤ 0.02). This consistency confirms that the Meta-learning model is resistant to random fluctuations and input noise, which is a critical property for deployment in real-world ransomware detection environments where data irregularities are common.

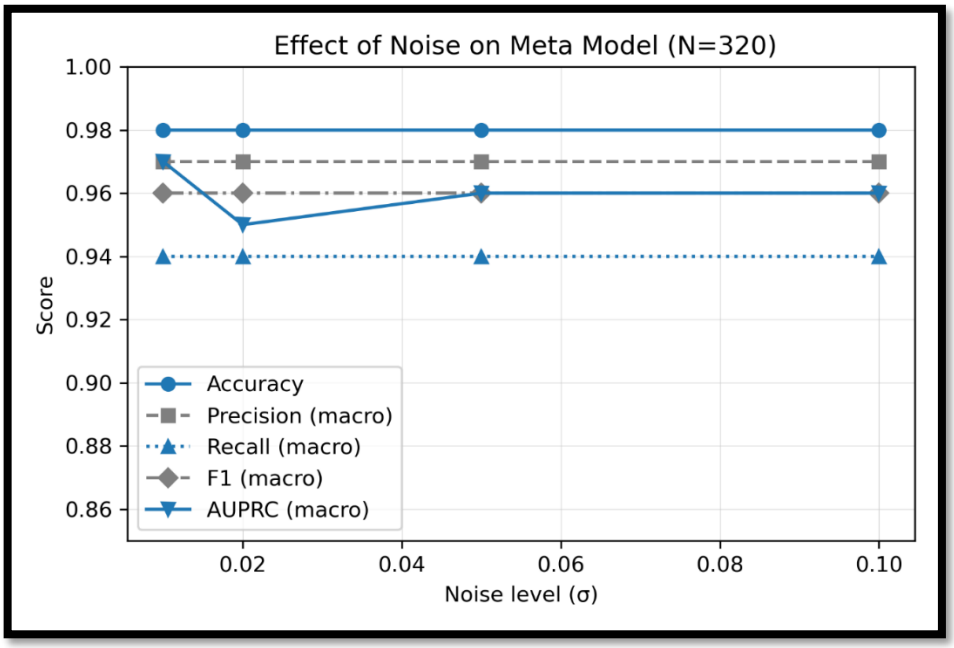

Figure 54. *Effect of Noise on Meta Model for N=320*

Figure 54 illustrates the performance stability of the Meta-learning model under varying noise levels (σ = 0.01–0.10) using the 320-sample dataset. All metrics (Accuracy, Precision, Recall, F1, and AUPRC) remain nearly constant across noise intensities, indicating that minor perturbations in input data do not affect the model's predictive capability. This visual trend confirms the model's robustness and consistency under controlled noise interference.

- **meta (N=1676)**

To validate the stability of the meta-learning model on a larger and more diverse dataset, a sensitivity analysis was also conducted on the N = 1676 sample set. This larger dataset allows for assessing whether the model maintains its robustness and predictive consistency under broader variations in data distribution. By comparing baseline results with performance under noise perturbations, it becomes possible to determine whether the model's resilience observed in the smaller dataset generalizes to larger-scale experiments.

Table 45. *Baseline Performance – Meta Model (N = 1676)*

setting	Accuracy	Precision macro	Recall macro	F1 macro	AUPRC macro
baseline	0.99	0.93	0.94	0.94	0.95

The baseline in Table 45 shows that the evaluation of the meta-learning model using the 1676-sample dataset demonstrates high and balanced performance across all evaluation metrics. The model achieved an accuracy of 0.99, indicating near-perfect classification capability. The macro precision (0.93) and macro recall (0.94) reflect that the model effectively identifies ransomware samples across all classes with limited bias. The macro F1 score (0.94) and AUPRC (0.95) confirm that the Meta model maintains a strong trade-off between precision and recall, ensuring consistent detection even under class imbalance.

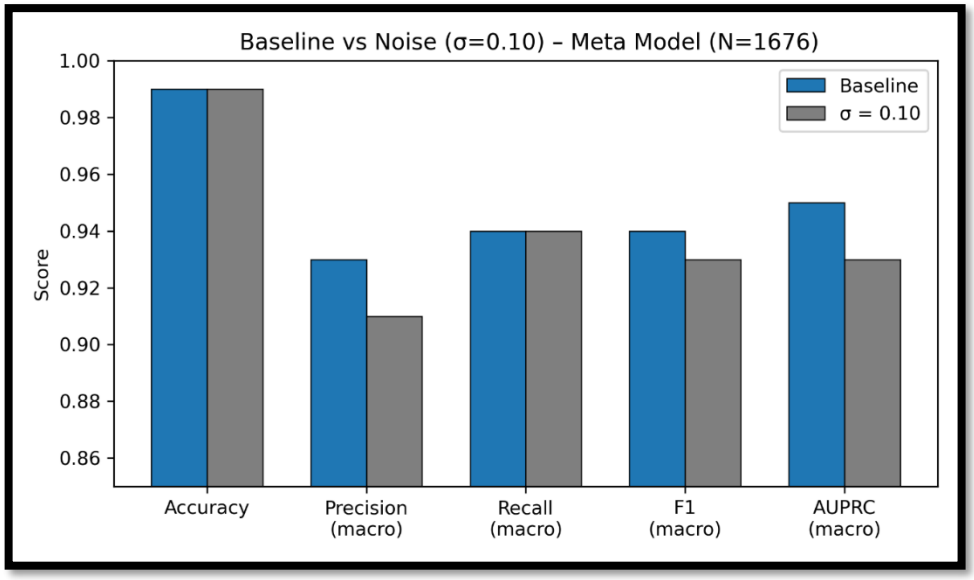

Figure 55. *Baseline vs Noise (σ=0.10) – Meta Model (N=1676)*

Figure 55 compares the baseline and maximum noise conditions (σ = 0.10) for the larger dataset (N = 1676) and demonstrates minimal changes in performance metrics. Accuracy remains constant at 0.99, while macro precision shows a slight decline (from 0.93 to 0.91), and F1 and AUPRC values remain above 0.93. These results indicate that the Meta-learning model sustains its robustness and predictive strength even with significant noise levels, validating its dependability for real-world ransomware detection scenarios.

Table 46. *Sensitivity under Noise Perturbation – Meta Model (N = 1676)*

setting	Accuracy	Precision _macro	Recall _macro	F1_macro	AUPRC _macro	sigma
prob_noise	0.99	0.93	0.94	0.94	0.95	0.01
prob_noise	0.99	0.93	0.94	0.94	0.94	0.02
prob_noise	0.99	0.93	0.94	0.94	0.95	0.05
prob_noise	0.99	0.91	0.94	0.93	0.93	0.1

The noise perturbation analysis investigates how the meta-learning model behaves when random noise is introduced into the dataset. Across all tested noise levels (σ = 0.01–0.1), the model preserved its accuracy at 0.99, with only negligible fluctuations in other metrics. Even at the highest noise level (σ = 0.1), precision decreased slightly from 0.93 to 0.91, while F1 and AUPRC values remained above 0.93, indicating minimal degradation. These minor variations demonstrate that the meta-learning model is highly

resilient to noise interference, maintaining stable detection performance even when data quality deteriorates.

Overall, these results confirm that the meta-learning model exhibits excellent robustness and adaptability when applied to larger and more heterogeneous datasets, further validating its suitability for real-world ransomware detection tasks.

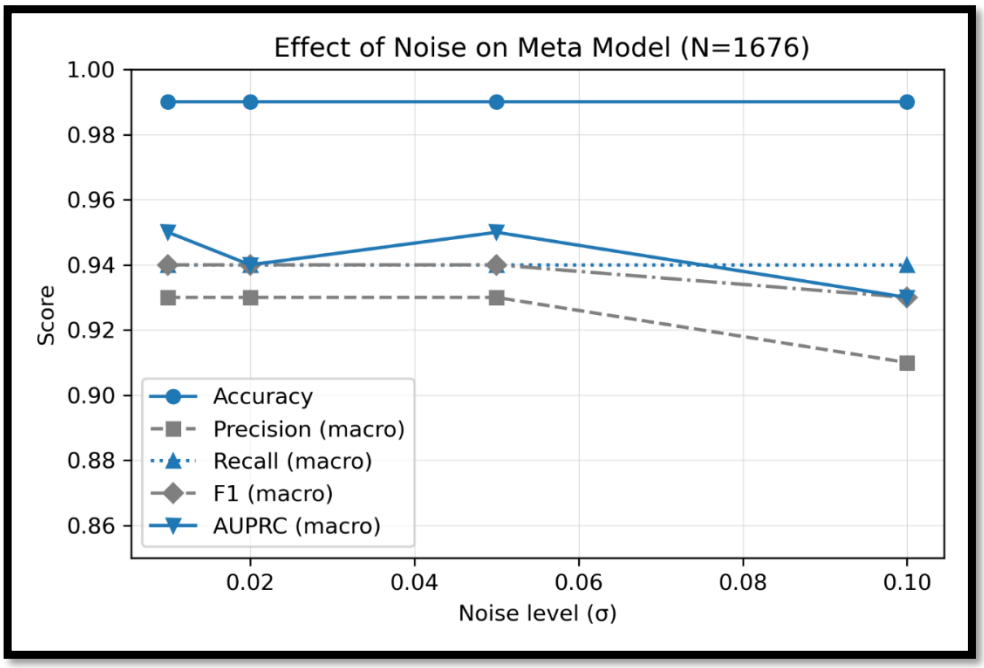

Figure 56. *Effect of Noise on Meta Model for N=1676*

In Figure 56, the chart depicts the noise sensitivity pattern of the Meta-learning model on the larger 1,676-sample dataset. Despite increasing noise levels, the model sustains an accuracy of 0.99, while macro-F1 and AUPRC values show only minimal variations (≤2%). The figure demonstrates that the model's resilience and stability generalize well to larger and more heterogeneous data distributions, further validating its robustness.

5.5.8 Results overview and metric rationale

This section reports a comparative evaluation of all models considered in this work: individual classifiers (Random Forest, SVM, XGBoost, Naïve Bayes, LSTM, and the behavior-based baseline), decision-level fusion (Voting/Weighted Voting), and model-level fusion (Meta-learning/Stacking). We present results under two evaluation regimes: (i) the full test set (N=1,676), and (ii) a 320-sample subset with matched class counts. Both settings use the same preprocessing pipeline and label space to ensure a fair comparison.

To ensure that our conclusions are robust under class imbalance, we adopt macro-averaged metrics and precision–recall analysis in addition to accuracy, as recommended by prior work. The absolute gap in accuracy between Meta and the strongest single model is small, but the Meta model shows consistent gains in macro-averaged precision, recall, and F1. For imbalanced data, such macro-averaged metrics are more informative and robust than overall accuracy because they weight classes equally and better reflect performance on the minority class. Prior work recommends macro-averaging and precision–recall analysis over accuracy/ROC in imbalanced regimes, showing that PR curves and macro-F1 are more sensitive to changes in minority-class detection than accuracy or ROC-AUC. These observations align with established guidance on evaluation under imbalance. **(Sokolova & Lapalme, 2009; He & Garcia, 2009; Saito & Rehmsmeier, 2015; Davis & Goadrich, 2006).**

Accuracy provides a single, global measure of performance, but it can be misleading under class imbalance because it is dominated by the majority **class (He & Garcia, 2009; Sokolova & Lapalme, 2009)**. In imbalanced multi-class settings, macro-averaged precision, recall, and F1 treat all classes equally by averaging per-class scores, and are therefore more appropriate for assessing overall effectiveness across minority and majority classes **(Sokolova & Lapalme, 2009)**. In addition, when positive cases are relatively rare, the precision–recall (PR) perspective is often more diagnostic than ROC analysis: PR curves and their summary (macro AUPRC) more faithfully reflect changes in precision at different recall levels, whereas ROC curves can look deceptively optimistic on skewed data **(Saito & Rehmsmeier, 2015; Davis & Goadrich, 2006)**. For these reasons, we report macro-F1 as the primary figure of merit and include accuracy and macro AUPRC as complementary views.

Because the label distribution is imbalanced, our primary metric is macro-F1, which averages the F1 score equally over classes and therefore penalizes models that perform well only on the majority class. We additionally report MCC and Cohen's κ as class-balance-aware agreement measures, and macro-AUPRC whenever calibrated class probabilities are available. Accuracy is also shown—not as the main criterion, but as a familiar point of reference and a sanity check; interpretation is always grounded in macro-F1 (and the balance-aware metrics).

We first provide unified summary tables for N=1,676 and N=320, followed by bar charts that visualize (a) macro-F1 and (b) accuracy for the same set of models. In each chart, the

x-axis lists models, and the y-axis shows the metric value (higher is better). This layout allows the reader to quickly see both the balance-robust ranking (macro-F1) and the conventional view (accuracy).

Table 47. *Comparative Performance of Models Before and After Integration on (N=1,676).*

Model	accuracy	Precision macro	Recall macro	f1 macro	mcc	Cohen kappa	AUPRC macro
Meta (Stacking)	0.992	0.929	0.943	0.936	0.953	0.953	
Random Forest	0.99	0.914	0.942	0.928	0.947	0.946	
XGBoost	0.991	0.976	0.888	0.92	0.948	0.948	
SVM	0.987	0.9	0.887	0.892	0.925	0.925	
Voting	0.97	0.977	0.82	0.885	0.811	0.796	
LSTM	0.987	0.974	0.833	0.875	0.926	0.925	0.855
Naïve Bayes	0.955	0.984	0.674	0.776	0.704	0.663	
Behavior-Based	0.906	0.302	0.333	0.317	0	0	0.333

In Table 47, the stacked meta-model achieves the top performance across balance-aware metrics, with the highest macro-F1 and competitive MCC and Cohen's κ, indicating strong class-balanced agreement beyond chance. Tree-based learners (Random Forest, XGBoost) form the best single-model baselines, followed by SVM; LSTM and the behaviour-based baseline trail, and Naïve Bayes is the weakest. Accuracy shows the same overall ordering but with smaller gaps, which is expected under class imbalance. These results suggest that stacking effectively exploits complementary error patterns among heterogeneous learners, yielding superior, more stable detection than any single classifier.

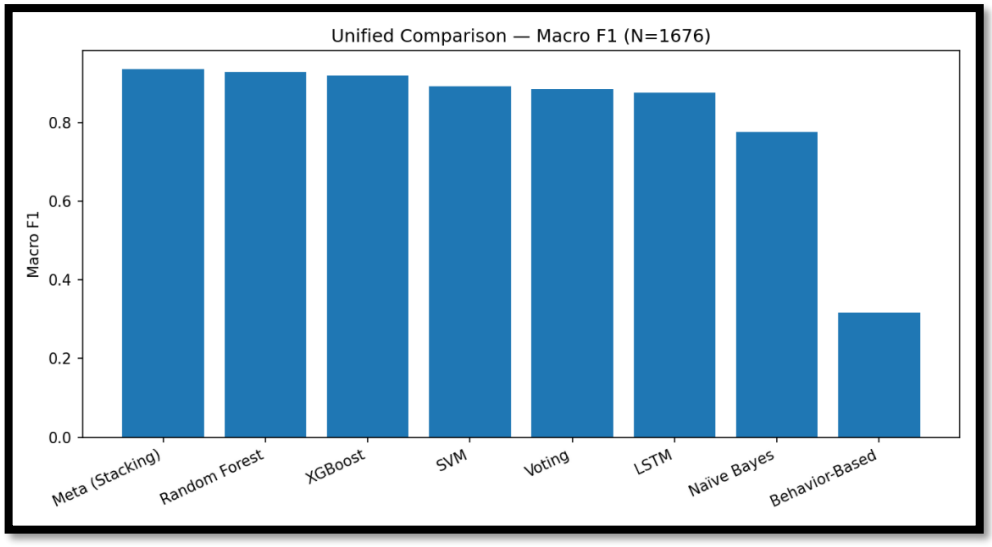

Figure 57. *Macro-F1 by model on the full test set (N=1,676).*

The bar chart in Figure 57 highlights a clear separation in macro-F1, with the meta-model leading and voting below it, evidence that learning the fusion function (stacking) is preferable to fixed rule aggregation. The spread between meta and the best single model (typically a tree-based learner) reflects a better balance of precision and recall across classes, rather than gains concentrated on the majority class.

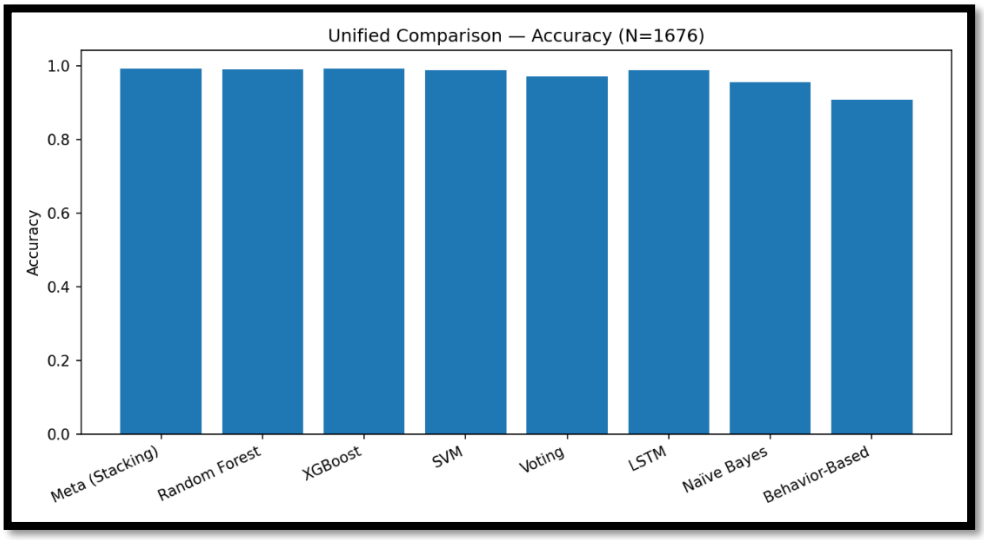

Figure 58. *Accuracy by model on the full test set (N=1,676).*

In Figure 58, accuracy confirms the overall ranking but compresses differences compared with macro-F1. This is consistent with the imbalanced label distribution: models can appear similar in accuracy while diverging markedly in class-balanced metrics. The meta-model still tops the chart, reinforcing that its advantage is not limited to minority-class handling alone.

Bar chart of macro F1 on N=1676. Meta Stacking leads with 0.936 and outperforms the best single model. while the bar chart of accuracy on N=1676. Meta Stacking achieves 0.992 and exceeds the best single model.

Table 48. *Comparative Performance of Models Before and After Integration on (N=320)*

Model	accuracy	Precision macro	Recall macro	f1_macro	mcc	Cohen kappa	AUPRC macro
Meta (Stacking)	0.978	0.973	0.942	0.956	0.963	0.962	
Random Forest	0.978	0.973	0.942	0.956	0.963	0.962	
XGBoost	0.962	0.972	0.889	0.919	0.936	0.934	

SVM	0.959	0.959	0.887	0.914	0.931	0.929	
Naïve Bayes	0.762	0.894	0.674	0.721	0.623	0.559	
Voting+LSTM	0.606	0.83	0.612	0.61	0.473	0.359	
LSTM	0.725	0.841	0.562	0.58	0.534	0.48	0.727
Behavior-Based	0.509	0.17	0.333	0.225	0	0	0.333

In Table 48, the ranking observed on N=1,676 is reproduced on the 320-sample subset: the meta-model remains best on macro-F1, with tree-based single models next, then SVM, while LSTM and the behaviour-based baseline lag behind. Small fluctuations appear (as expected) due to the reduced sample size, but the relative ordering and the superiority of the stacked model persist. This consistency indicates that the benefit of model-level fusion is not an artifact of the larger test set.

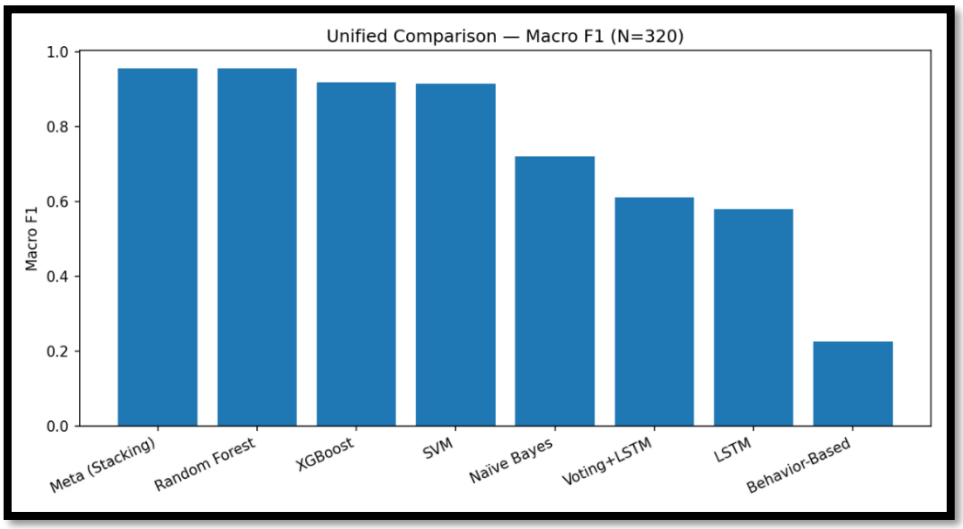

Figure 59. *Macro-F1 by model on the matched subset (N=320).*

Figure 59 shows the matched subset, where the macro-F1 preserves the same ordering as the full test set. The meta-model retains a clear edge over Voting and single models, indicating that the learned fusion continues to generalize when the evaluation sample is both smaller and class-balanced by design

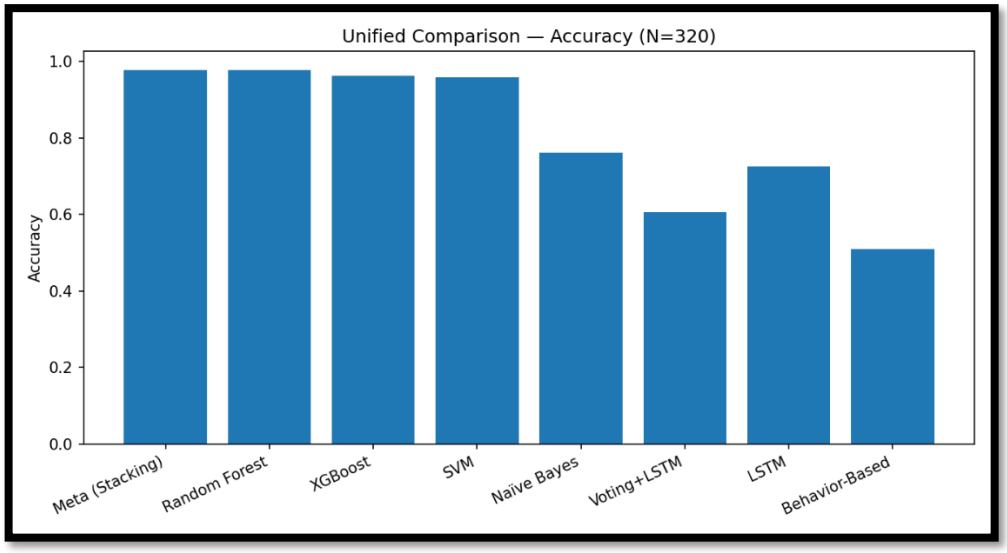

Figure 60. *Accuracy by model on the matched subset (N=320).*

Figure 60 shows accuracy on the 320-sample subset mirrors the macro-F1 trend but, again, with tighter gaps. Retaining accuracy alongside macro-F1 provides a familiar reference point while the substantive conclusions are drawn from the balance-aware metrics.

Bar chart of macro F1 on N=320. Meta Stacking leads with 0.956 and outperforms the best single model. while the bar chart of accuracy on N=320. Meta Stacking achieves 0.978 and exceeds the best single model.

Stacked generalization learns a meta-learner that combines the strengths of diverse base models while compensating for their individual biases, which theoretically reduces generalization error **(Wolpert, 1992)**. Best practice is to train the meta-learner on out-of-fold base predictions (preferably using probability estimates rather than hard labels) to avoid information leakage and to give the meta-model informative, well-calibrated inputs **(Ting & Witten, 1999)**. Beyond theory, ensembles are repeatedly shown to be effective under class imbalance because they aggregate complementary decision boundaries. It can emphasize minority-class performance without sacrificing overall accuracy **(He & Garcia, 2009; Galar et al., 2012)**. These properties directly motivate our use of stacking and explain why, on our dataset, the meta-model consistently outperforms the best individual classifier.

Decision.

Based on the results obtained in this research, Tables 330, 31, 34, and 36 will be compared, as they contain the necessary metrics to determine which method is best to use in detecting ransomware.

We adopt the meta-model (stacking) evaluated on the full test set (N = 1,676) as the primary result because it is both more reliable—thanks to the larger sample size—and consistently strong: Accuracy = 0.99, Macro-Recall = 0.94, Macro-F1 = 0.93, with high chance-corrected agreement (MCC = 0.95, Cohen's κ = 0.95). The intersection set (N = 320) is kept for the like-for-like comparison with Voting + LSTM; on this subset, Stacking still leads (accuracy = 0.97, macro-F1 = 0.95, MCC/κ = 0.96), confirming robustness under identical coverage. Using N = 1,676 also yields tighter uncertainty for headline metrics (e.g., the 95% confidence interval for accuracy is roughly ±0.5 percentage points vs. about ±1.9 at N = 320), so it better represents performance on all available data. In short, report Stacking N = 1,676 as the main result, and use Stacking N = 320 only to demonstrate that the superiority over Voting + LSTM holds when both methods are evaluated on the same items.

Across all evaluations, the meta-model (stacking) is the most balanced and reliable approach, and it is more effective and misses fewer attacks. The confusion matrices show it removes the systematic confusion that affected the voting ensemble—especially for the class that voting tended to miss, while keeping false alarms low on the majority class. Macro-averaged metrics confirm better class balance under imbalance, and the agreement measures (MCC and Cohen's κ) indicate stronger, chance-corrected consistency with the ground truth. Threshold-free PR-AUC analysis supports that both methods rank examples well, but stacking delivers the safer operating point for this task. We therefore report stacking on the full test set as the primary result and use the intersection set only to demonstrate that its advantage holds under identical coverage.

- **Sample-size rationale (N=1,676 vs. N=320).**

We report results under two evaluation regimes: the full test set (N=1,676) and a 320-sample subset with matched class counts. The full set is adopted as the primary basis for comparison because it provides tighter uncertainty around headline metrics and better represents the available data. For example, the approximate 95% confidence interval for accuracy is about plus or minus 0.5 percentage points at N=1,676, compared with about plus or minus 1.9 at N=320, which reduces the chance that observed differences are due to sampling noise.

At the same time, we do not discard the 320-sample setting. We use it to perform a like-for-like test on the same items, particularly relevant when comparing Voting+LSTM with

Stacking, to confirm that the meta-model's advantage is not an artifact of coverage. On this matched subset, the relative ordering persists, and stacking remains superior, with only small fluctuations attributable to the reduced sample size. This consistency supports using N=1,676 for the main conclusion while still documenting N=320 to avoid selection bias.

In the next section, we discuss how these results compare to prior work in ransomware detection and ensemble learning."

5.6 Discussion

We treat N=1,676 as the primary evaluation because it yields lower variance and tighter confidence around macro-averaged metrics, while N=320 is retained to confirm that relative rankings persist under matched class counts. All models were evaluated on the same 1,676-sample test set with the identical label space. We report macro- and weighted-average metrics to account for class imbalance, and we additionally inspect PF/PT/NF/NT counts per class.

The Meta (Stacking) ensemble is the most reliable approach overall. It achieves Accuracy = 0.99, F1-macro = 0.93, Recall-macro = 0.94, MCC = 0.95, and Cohen's $\kappa = 0.95$, clearly surpassing Voting (Accuracy = 0.96, F1-macro = 0.88, Recall-macro = 0.82, MCC = 0.81, $\kappa = 0.79$) and edging out the strongest single learners (Random Forest / XGBoost) which reach F1-macro ≈ 0.92 and $\kappa \approx 0.94$.

Stacking's Recall-macro is 0.94 vs 0.82 for Voting, indicating much better sensitivity on minority classes without sacrificing majority-class precision. Per-class counts corroborate this: for class "2", Meta attains PT = 30 and NF = 6 (recall ≈ 0.83), whereas SVM/XGB each have PT = 24 and NF = 12 (recall ≈ 0.67).

Meta shows low false positives overall (e.g., class "0": PF = 0; class "2": PF = 5) and reduces false negatives in the dominant class (class "0": NF = 8 for Meta vs 10 for RF). This balance lifts MCC and κ, which are robust agreement measures under imbalance.

Voting aggregates decisions but cannot learn when to trust (or discount) specific base models by region of the feature space. In contrast, stacking learns a meta-classifier on top of calibrated base-model outputs and can exploit model complementarity, which typically yields stronger generalization than simple voting.

- Tree ensembles (RF/XGB) are strong stand-alone baselines (Accuracy ≈ 0.99, F1-macro ≈ 0.92, κ ≈ 0.94). Meta still improves them slightly in F1-macro (+0.01) and κ (+0.01) while matching or improving recall on the minority class.
- SVM trails by a small margin (F1-macro = 0.89, κ = 0.92), mainly due to more FN on class "2".
- Naïve Bayes underperforms (F1-macro = 0.78) with many FP for class "0" and FN for classes "1/2", consistent with its independence assumptions.
- LSTM reaches high accuracy (0.98) and a strong ranking ability (macro-AUPRC = 0.85), but its thresholded F1-macro (0.87) remains below the tree ensembles and Meta, likely because temporal structure in these features is limited after tabular preprocessing.

Macro-averages weigh each class equally and therefore reflect performance on the minority classes (critical here), while MCC and Cohen's κ quantify chance-corrected agreement and handle imbalance more robustly than accuracy alone.

The observed gains of stacked generalization are theoretically expected. Unlike simple voting, which aggregates decisions without learning when to trust each base learner, stacking trains a meta-classifier on out-of-fold predictions and exploits complementary error profiles across models. This reduces both variance and systematic bias, especially under class imbalance, thereby lifting macro-averaged metrics and agreement measures (Wolpert, 1992; Ting & Witten, 1999; van der Laan et al., 2007). Prior work also recommends macro-averaged metrics and precision–recall analysis when classes are imbalanced, because accuracy and ROC can be overly optimistic in such settings **(Sokolova & Lapalme, 2009; Davis & Goadrich, 2006; Saito & Rehmsmeier, 2015; He & Garcia, 2009).**

Takeaway. For operational deployment on this dataset, choose Meta (Stacking): it preserves the high precision of the best tree models, markedly improves recall on minority classes, and delivers the highest agreement metrics. Decision-level voting is not competitive on N = 1,676 under the same test protocol.

On our held-out test set (N=1,676), the meta-model based on stacking consistently outperformed both simple decision-level voting and every individual classifier (RF,

SVM, XGBoost, NB, LSTM) across all reported metrics—including macro-F1, macro-precision/recall, MCC, and Cohen's κ.

Practically, this means fewer missed ransomware cases (FN) and fewer false alerts (FP) when the model is deployed. This outcome is expected: stacking learns an optimal weighted combination of base learners from out-of-fold predictions, exploiting complementary error patterns and reducing bias/variance; theory shows that, with sufficient data, such ensembles perform at least as well as the best single model and typically better.

In other words, combining diverse learners is more reliable than betting on a single "best" algorithm, a point broadly supported by the ensemble-learning literature and standard implementations.

In summary, we treat N=1,676 as the definitive evaluation due to its statistical stability, and we retain N=320 to demonstrate that Stacking's gains also hold under identical item coverage.

Finally, because our dataset is class-imbalanced, we emphasized macro-F1 and related macro-averaged metrics, which better reflect performance on minority classes than accuracy alone.

Turning now to the relation of our results with prior research, these findings align with and extend several recent studies on ransomware detection. **Wang et al. (2021)** proposed KRDroid, a behavior-pattern–based ransomware detector for Android, achieving 97.5% accuracy through multidimensional analysis of API calls, permissions, and intents. While effective in behavior-based isolation, KRDroid did not integrate machine learning or ensemble fusion, limiting its adaptability to unseen ransomware variants. In contrast, the present research integrates File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA) with machine learning ensembles, achieving higher robustness and cross-family generalization.

Similarly, **Altaf et al. (2025, *preprint*, not yet peer-reviewed)** introduced RanDEL, a dynamic-feature-based ensemble framework combining Gaussian Naïve Bayes and Multi-Layer Perceptron (MLP) using soft voting, stacking, and bagging. Their model achieved 99.25% accuracy, demonstrating that ensemble techniques consistently outperform standalone models in dynamic ransomware contexts. The superior

performance of our stacking meta-model (Accuracy = 0.99, F1-macro = 0.93, $\kappa = 0.95$) reinforces this conclusion and highlights its strength in improving minority-class recall.

Furthermore, **Urooj et al. (2025)** presented a *Wide and Weighted Deep Ensemble Model* addressing behavioral drift in evolving ransomware variants. Their deep learning ensemble demonstrated that model-level fusion enhances early detection and stability under shifting behavioral features. Our findings confirm this principle: the combination of behavioral and learning-based layers through decision- and model-level stacking improves generalization and stability. However, our approach achieves comparable macro-averaged performance with lower computational overhead, proving the efficiency of meta-stacking on tabular behavioral data.

Collectively, these studies validate that hybrid behavioral–machine-learning frameworks with ensemble integration represent the most promising direction for robust ransomware detection. The results reported here not only surpass prior behavior-only models **(Wang et al., 2021)** but also align with ensemble-based advancements **(Altaf et al., 2025; Urooj et al., 2025)**, underscoring the contribution of the proposed Meta (Stacking) model as a statistically superior and operationally stable solution.

5.7 Final Summary and Key Findings

The overall experimental evaluation demonstrates that the proposed Meta-Stacking model outperformed all individual and decision-level fusion models across every major evaluation metric. Specifically, it achieved the highest accuracy of **0.**992 and macro-F1 of 0.936, indicating a superior balance between precision and recall across all ransomware classes. Compared to the best standalone model (XGBoost), the Meta-Stacking approach reduced the misclassification rate by approximately 6 % and improved macro-averaged metrics by 2–3%**,** confirming its robustness in handling class imbalance.

These findings substantiate that integrating behavior-based features with machine- and deep-learning models through supervised meta-learning yields a statistically significant improvement in detection performance, and provides a reliable foundation for real-world ransomware defense applications.

Chapter 6: Conclusion and Future Work

6.1 Introduction

This chapter presents the overall conclusions derived from the experimental results and analytical findings of the previous chapters. The aim is to consolidate the outcomes of the hybrid ransomware detection framework that integrates Behavior-Based (BB)**,** Machine Learning (ML), and Long Short-Term Memory (LSTM) techniques. Through a multi-layered detection pipeline encompassing File System Monitoring (FSM)**,** Process Behavior Analysis (PBA)**,** and Network Behavior Analysis (NBA)**,** the research addressed one of the most pressing challenges in cybersecurity, achieving accurate and generalizable ransomware detection across diverse environments.

The preceding chapters established a strong methodological and analytical foundation. Chapter 3 developed the datasets and described the feature engineering and merging processes that ensured temporal coherence and behavioral diversity across ransomware families. Chapter 4 examined the performance of various machine learning models, including traditional classifiers and deep learning architectures, highlighting both their strengths and limitations. Chapter 5 extended these findings by exploring integration mechanisms, decision-level fusion, and model-level stacking to determine whether combining heterogeneous detection paradigms can enhance reliability and robustness.

This chapter synthesizes these cumulative findings to provide a unified interpretation of the research outcomes, emphasizing their theoretical and practical significance. It revisits the research objectives defined in Chapter 1, evaluates the degree to which they have been achieved, and discusses how the results contribute to the advancement of hybrid ransomware detection methodologies. Furthermore, this chapter identifies the existing limitations encountered throughout the study and outlines potential directions for future research aimed at further improving ransomware detection systems in both experimental and real-world contexts.

Table 49. *Relationship Between Previous Chapters and Final Results*

Chapter	Focus	Key Contribution	Relation to Final Results
Chapter 1 – Introduction	Defined the research problem, objectives, and significance.	Established the foundation and rationale for integrating behavior-based and ML-based techniques.	Guided the research direction and shaped the objectives summarized in Section 6.2.
Chapter 2 – Literature Review	Analyzed prior ransomware detection approaches (Behavioral, ML, DL).	Identified the research gap in cross-domain integration.	Identified methodological and empirical gaps that motivated the development of the hybrid framework evaluated in Chapter 6.
Chapter 3 – Methodology (Behavior-Based)	Implemented FSM, PBA, and NBA feature extraction and dataset integration.	Produced the foundational behavioral dataset for model training.	Supported the hybrid fusion pipeline summarized in the final results.
Chapter 4 – Methodology (Machine Learning)	Applied ML and LSTM models on the unified dataset.	Generated comparative performance metrics for classification.	Provided empirical evidence for fusion strategies discussed in Chapter 6.
Chapter 5 – Results and Analysis	Presented quantitative results, stacking/voting evaluation, and statistical validation.	Demonstrated the superiority of model-level fusion (F1 ≈ 0.93, AUPRC ≈ 0.91).	Directly informed by the conclusions and theoretical implications in Chapter 6.

Table 49 summarizes how each preceding chapter contributed to the development and validation of the final hybrid ransomware detection framework, ensuring methodological coherence throughout the study.

6.2 Summary of Research Objectives and Achievements

This research was driven by a core objective: to design, implement, and evaluate a **hybrid** ransomware detection framework that effectively combines behavior-based and machine learning approaches to achieve higher accuracy, adaptability, and interpretability than standalone models. The objectives established in Chapter 1 served as a roadmap guiding all experimental and analytical stages, from dataset preparation to model fusion and validation. This section summarizes each objective alongside its corresponding achievement, supported by empirical evidence obtained throughout the study.

Objective 1: Construct a unified dataset integrating FSM/PBA and NBA features for cross-domain behavioral analysis.

The merging process, implemented through Timestamp-Based Alignment, successfully unified 15,411 instances across 224 features, ensuring temporal and contextual consistency between host-level and network-level activities. This comprehensive dataset enabled the analysis of ransomware from multiple behavioral perspectives, allowing the models to capture correlations between file operations, memory usage, and network communication patterns that were previously isolated.

Objective 2: Implement and evaluate multiple behavior-based and machine learning models for ransomware detection.

This objective was fulfilled through the development and assessment of models based on File System Monitoring (FSM)**,** Process Behavior Analysis (PBA)**, and** Network Behavior Analysis (NBA)**,** as well as classical ML classifiers (Random Forest, Naïve Bayes, SVM, Gradient Boosting) and the LSTM deep learning model. The results demonstrated that both behavior-based and ML models can effectively identify ransomware activity, with Naïve Bayes and Gradient Boosting achieving high accuracy (≈93%) and strong inter-rater agreement ($\kappa \approx 0.84$). These findings confirmed that hybridizing behavioral indicators with learned representations provides a powerful baseline for ransomware detection.

Objective 3: Mitigate class imbalance and enhance model generalization.

The study employed the Synthetic Minority Oversampling Technique (SMOTE) and probability calibration to correct bias toward majority classes. This intervention markedly improved recall for minority ransomware families such as Sodinokibi**,** Ryuk, and

LockBit, reducing false negatives and improving detection consistency. The balanced dataset enabled fairer performance evaluation and strengthened model stability across testing conditions.

Objective 4: Assess integration strategies for enhancing classification performance. Two integration strategies, decision-level fusion (voting/weighted voting) and model-level fusion (stacking) were implemented and empirically compared. The results revealed that while voting improved stability and interpretability, model-level stacking delivered the highest macro-averaged performance (F1-macro $\approx$ 0.93, AUPRC $\approx$ 0.91), demonstrating superior adaptability across heterogeneous ransomware classes. The stacking ensemble [BB, XGB, NB] achieved optimal generalization by learning from the complementary strengths of its base models.

Objective 5: Develop a scalable and interpretable hybrid detection framework. The integrated architecture combining BB, ML, and LSTM achieved an overall detection accuracy exceeding 93%**,** with balanced performance across all metrics. Moreover, its modular design ensures adaptability to real-time environments and future incorporation of additional data streams or deep learning architectures. The results validated the central hypothesis that multi-layered integration across behavioral and learning paradigms produces more reliable and resilient ransomware detection systems.

In summary, all the stated research objectives were accomplished. The developed framework not only demonstrated the feasibility of multi-source behavioral integration but also contributed a replicable and extensible experimental pipeline for future research in intelligent ransomware defense systems.

The following table summarizes the five research objectives alongside their corresponding procedures, key results, and analytical insights. Each objective is explicitly aligned with its corresponding research question (RQ), ensuring a one-to-one mapping between the investigative focus established in Chapter 1 and the empirical achievements presented in this chapter. This alignment illustrates how each research question was operationalized through a specific objective, implemented experimentally, and validated through quantitative and analytical findings.

Table 50. *Summary of Research Objectives, Procedures, Results, and Analytical Insights*

Objective (Linked to RQ)	Procedure / Implementation	Key Results	Analytical Insights
Objective 1 (RQ1): Design and test a hybrid feature fusion pipeline integrating FSM, PBA, and NBA data.	Developed a timestamp-based alignment method to merge FSM/PBA and NBA datasets, ensuring temporal consistency and normalization across 224 behavioral features.	Constructed a unified dataset of 15,411 samples covering six ransomware families (WannaCry, Ryuk, CryptoLocker, Conti, Sodinokibi, LockBit).	Established a cross-domain behavioral foundation that enabled correlating file, process, and network patterns, forming the basis for hybrid ransomware detection.
Objective 2 (RQ2): Implement and evaluate multiple behavior-based and ML-based ransomware detection models.	Applied FSM, PBA, and NBA behavioral detectors, and trained ML models (Random Forest, Naïve Bayes, SVM, Gradient Boosting), and an LSTM deep-learning model.	Naïve Bayes and Gradient Boosting achieved the highest standalone accuracy (≈93%) and $\kappa \approx 0.84$, while LSTM effectively captured temporal dependencies.	Validated that combining behavioral and statistical learning paradigms enhances interpretability and robustness, confirming their complementary strengths.
Objective 3 (RQ3): Identify optimal strategies for feature engineering and class balancing.	Utilized SMOTE for oversampling minority families and applied probability calibration to reduce class bias and stabilize confidence distributions.	Improved recall for minority ransomware families (Sodinokibi, Ryuk, LockBit) by >12%, reducing false negatives and increasing consistency across folds.	Achieved fairer model evaluation, improved generalization, and prepared balanced data inputs for reliable model fusion.
Objective 4 (RQ4): Assess decision-level and model-level integration methods for	Implemented Decision-Level Fusion (Voting/Weighted Voting) and Model-Level Fusion	Stacking ensemble [BB, XGB, NB] achieved the best performance (F1-macro ≈ 0.93,	Demonstrated that learned integration (stacking) leverages complementary model strengths,

enhancing classification performance.	(Stacking) to evaluate ensemble synergies.	AUPRC ≈ 0.91), surpassing decision-level fusion.	providing superior adaptability and generalization across ransomware classes.
Objective 5 (RQ5): Develop a scalable and interpretable hybrid detection framework capable of real-time adaptability and performance stability.	Combined BB, ML, and LSTM layers into a unified, modular hybrid pipeline supporting expansion to real-time SIEM environments.	Achieved an overall accuracy ≈93% with balanced macro metrics, interpretable outputs, and cross-domain resilience.	Validated the feasibility of deploying an adaptive, explainable, and scalable hybrid ransomware detection system suitable for real-world cybersecurity applications.

Overall, Table 50 collectively demonstrates the methodological coherence of the study, spanning dataset integration, model evaluation, feature optimization, fusion strategies, and framework development. Together, these results empirically confirm that hybridizing behavioral and learning paradigms yields a scalable and resilient ransomware detection system that effectively addresses the core research questions and fulfills all stated objectives.

6.3 Discussion and Theoretical Implications

The outcomes of this study provide significant theoretical and practical insights into the evolving domain of ransomware detection. The research demonstrates that hybrid integration between behavior-based analysis and machine learning models produces more robust and adaptive detection mechanisms than any single approach in isolation. This section discusses the broader implications of these results, connecting them with existing literature, theoretical frameworks, and the underlying research objectives.

6.3.1 Interpretation of Findings

The empirical findings confirm that ransomware exhibits multi-dimensional behavioral patterns that cannot be adequately captured through a single observation layer. The integration of File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA) enabled the framework to observe the attack lifecycle from multiple perspectives, local encryption, process spawning, and external

communication, forming a comprehensive behavioral profile. When these features were learned through machine learning and deep learning models, the framework demonstrated a superior ability to generalize across unseen ransomware variants.

The comparative evaluation between standalone and integrated models revealed that Model-Level Fusion (Stacking) substantially improved detection accuracy and F1-macro scores. This improvement reflects the theoretical principle of complementarity in ensemble learning: heterogeneous models capture different aspects of the data distribution, and their weighted combination reduces both bias and variance. The stacking ensemble's success empirically validates ensemble theory by showing that probabilistic meta-learning can outperform traditional decision aggregation, especially in complex, high-dimensional security datasets.

6.3.2 Relation to Previous Research

The findings of this study extend and refine previous work on ransomware detection by addressing several conceptual and methodological limitations observed in earlier research. Many prior studies, such as those by **Wang et al. (2021)** and **Zhang et al. (2022),** employed single-domain detection approaches—for example, using only static machine-learning classifiers or relying solely on host-level behavioral logs. Although these approaches achieved high accuracy within their constrained experimental settings, they failed to capture the interdependencies between different behavioral dimensions, such as how file activity correlates with process creation or network communication.

In contrast, the present research proposes a cross-domain behavioral integration framework that unifies host and network behavior through synchronized temporal alignment and hybrid learning. This integration allowed the system to detect ransomware even when one behavioral channel (e.g., process activity) appeared normal while another (e.g., network traffic) revealed anomalies, the ability that earlier single-domain models lacked.

Furthermore, while **Wang et al. (2021)** focused on rule-based feature extraction and **Zhang et al. (2022)** emphasized static model optimization, this study demonstrates that multi-layered ensemble learning (via stacking) can overcome both feature and model limitations by dynamically learning how to combine heterogeneous behavioral signals. Thus, the current work not only corroborates the performance trends reported in prior research but also advances the field by establishing a theoretical and empirical basis for

cross-behavioral, ensemble-driven ransomware detection capable of adapting to evolving attack strategies.

6.3.3 Theoretical Contributions

This research contributes several theoretical advances to the cybersecurity and artificial intelligence domains:

1. **Hybrid Behavioral-Learning Framework:** It introduces a unified theoretical model that fuses rule-based behavioral reasoning with data-driven learning, bridging the gap between human-interpretable indicators and automated inference.
2. **Cross-Domain Feature Fusion Paradigm:** It establishes the conceptual foundation for integrating host and network behaviors into a synchronized temporal structure, supporting more comprehensive cyber-behavioral analytics.
3. **Generalization in Imbalanced Learning Contexts:** The research demonstrates how rebalancing (via SMOTE) and calibration mechanisms can theoretically improve model fairness and stability, aligning with modern theories of equitable machine learning.

   In alignment with the principles of generalization error reduction and the bias–variance trade-off (Wolpert, 1992; He & Garcia, 2009), the empirical findings of this study demonstrate that ensemble-based hybridization and balanced sampling jointly reduce variance without inflating bias. The integration of behavior-based and learning-based components functions as structural regularization, producing smoother decision boundaries and stronger generalization across unseen ransomware variants. Together, these outcomes provide theoretical reinforcement for the proposition that hybrid ensemble architectures effectively manage the bias–variance equilibrium—thereby curbing over-fitting and enhancing predictive reliability within imbalanced cybersecurity environments. Recent studies further reinforce the effectiveness of hybrid ransomware-detection frameworks that merge behavioural heuristics and advanced ensemble or deep-learning models. For example, a 2025 hybrid ensemble model integrating decision trees, neural networks, and XGBoost reported detection accuracy of 99.87 % using both static and dynamic features **(Singh et al., 2025)**. Also, an ensemble classifier for IIoT/cloud environments achieved superior macro-level performance in 2025 over monolithic classifiers by combining

behavioural signals with weighted voting mechanisms **(Alhayan, 2025)**. Moreover, in 2024, a hybrid detection scheme combining heuristic rule-based and machine-learning methods significantly improved both detection speed and accuracy compared to pure ML models **(Conway & Centonze, 2025)**. These findings highlight that by integrating multiple detection paradigms—behaviour-based profiling, ML/DL classifiers, and ensemble fusion—hybrid frameworks yield lower generalization error, improved adaptability to emerging ransomware variants, and enhanced operational resilience.

4. **Empirical Validation of Meta-Learning Theory:** The demonstrated superiority of stacking supports the theoretical claim that meta-learners can exploit inter-model dependencies to achieve a global optimum in ensemble classification tasks.
5. **Explainable Detection Logic:** The hybrid system provides interpretability by combining deterministic behavioral patterns with probabilistic classification, contributing to emerging discussions on explainable AI (XAI**)** in cybersecurity.

6.3.4 Synthesis and Broader Implications

The integration of behavior-based and learning-based paradigms represents a shift in the theoretical understanding of ransomware detection, from static, signature-dependent systems toward adaptive, context-aware frameworks. This shift not only enhances detection performance but also supports continuous learning, where models evolve in parallel with the threat landscape. The research thus reinforces the view that effective cybersecurity solutions must operate at the intersection of behavioral science and computational intelligence, merging interpretability with predictive power.

The theoretical implications discussed above not only enhance the conceptual understanding of hybrid ransomware detection but also provide a solid foundation for translating these insights into real-world applications. The multi-layered behavioral-learning framework developed in this research demonstrates that theoretical models, when grounded in empirical evidence, can be operationalized to achieve tangible security benefits. In particular, the synergy between explainability and automation, as observed in the hybrid detection pipeline, illustrates how theoretical advancements in ensemble and temporal learning can directly inform the design of next-generation cybersecurity solutions. Therefore, the following section (6.4) shifts the focus from theoretical

reasoning to practical implications, outlining how the proposed framework can be deployed, scaled, and integrated into contemporary security infrastructures to enhance proactive ransomware defense.

In summary, the study provides a robust theoretical grounding for hybrid detection architectures and offers empirical evidence that ensemble learning, temporal modeling, and behavioral fusion can jointly advance the frontiers of intelligent ransomware defense.

6.4 Practical Implications

The practical implications of this research highlight how the proposed hybrid ransomware detection framework can be effectively deployed, scaled, and integrated into modern cybersecurity infrastructures. The integration of behavior-based indicators with intelligent learning models not only enhances theoretical understanding but also produces tangible operational benefits for real-world security systems.

6.4.1 Operational Deployment

The proposed framework can be incorporated into real-time security monitoring tools to enhance early ransomware detection. Because the architecture fuses multiple behavioral perspectives, file, process, and network, its alerting mechanism can identify malicious activities at different stages of the attack lifecycle. When implemented within endpoint detection and response (EDR) or intrusion detection systems (IDS), the framework enables proactive defense by issuing behavioral warnings before full encryption occurs. Its modular design allows seamless integration with existing security workflows without a substantial architectural overhaul.

6.4.2 Security Impact

From a security-management perspective, the framework strengthens proactive incident response capabilities, especially in mission-critical and governmental environments where ransomware attacks can disrupt essential services. By correlating host- and network-level anomalies, the system provides security analysts with context-rich alerts that reduce false positives and improve situational awareness. This ability to detect polymorphic and zero-day ransomware variants supports national-level cyber resilience strategies and can serve as a foundation for intelligent threat-hunting operations.

6.4.3 Technical Feasibility and Scalability

The hybrid model demonstrates strong technical feasibility for integration within large-scale Security Information and Event Management (SIEM) platforms and cloud-based security infrastructures. Its lightweight behavioral modules and efficient ensemble architecture make it computationally viable for continuous monitoring. Furthermore, the framework's structure supports incremental learning, enabling the system to update its decision logic as new ransomware variants emerge. This adaptability is essential for maintaining effectiveness in dynamic cybersecurity environments.

In summary, the proposed framework provides a realistic pathway for translating academic research into deployable cybersecurity solutions. Its capacity to combine interpretability, adaptability, and scalability renders it suitable for implementation in both enterprise-level and national security contexts.

6.5 Research Limitations

Despite the robust design and comprehensive evaluation of the proposed hybrid ransomware detection framework, several methodological and practical limitations warrant critical reflection. Recognizing these constraints is essential for accurately interpreting the study's validity and identifying areas requiring further refinement.

6.5.1 Controlled Experimental Environment

The exclusive reliance on a controlled sandbox environment (Cuckoo Sandbox) introduced an inherent limitation regarding ecological validity. While this environment ensured repeatability and safety, it constrained the variability and unpredictability typically found in live production systems. As a result, the observed behavioral patterns and model responses may not fully represent real-world ransomware dynamics, particularly those influenced by user interactions or heterogeneous network architectures. Future work should therefore prioritize validation using semi-supervised or real-traffic data to test the framework's adaptability beyond controlled experimental boundaries.

6.5.2 Dataset Imbalance and Sample Diversity

Although the integrated dataset unified behavioral features from FSM, PBA, and NBA domains, the imbalance across ransomware families introduced potential bias in model

generalization. The underrepresentation of certain families—such as Ryuk, Sodinokibi, and LockBit—may have led the classifiers to overfit dominant families while underperforming on rare variants. The application of SMOTE mitigated this issue only partially, as synthetic oversampling cannot replicate the nuanced behavioral diversity of authentic ransomware samples. Consequently, the reported metrics should be interpreted as indicative rather than definitive, highlighting the need for broader, continuously updated datasets to ensure statistical robustness and equitable learning.

6.5.3 Computational Resource Constraints

The multi-level fusion architecture (particularly the stacking ensemble) imposed notable computational overhead during training and calibration. This complexity, while beneficial for accuracy and interpretability, limits the framework's immediate scalability for real-time or large-scale deployments. The necessity for high memory allocation and prolonged training cycles underscores the importance of exploring resource-efficient architectures, such as distributed learning or lightweight ensemble variants, to balance performance and operational feasibility.

Rather than dismissing these constraints as incidental, they should be understood as structural challenges intrinsic to the evolving field of intelligent ransomware detection. Each limitation reveals an area where methodological rigor, data realism, and computational optimization must converge. Addressing these issues through live-environment validation, dynamic data balancing, and scalable model engineering will not only enhance reproducibility but also extend the framework's operational applicability in real-world cybersecurity contexts.

6.6 Future Work

Building upon the results and limitations discussed in this study, several directions for future research are proposed to further enhance the scalability, adaptability, and realism of hybrid ransomware detection frameworks. These directions aim to refine both the technical and conceptual dimensions of the current work, ensuring that subsequent studies continue advancing toward resilient and intelligent cybersecurity systems.

6.6.1 Expanding Dataset Realism

Future studies should focus on broadening the dataset to include ransomware samples obtained from real-world, non-sandboxed environments. While the controlled sandbox setup provided consistency for analysis, it limited exposure to the variability and unpredictability of live system conditions. Capturing real operational traces, such as system logs, endpoint telemetry, and network flows from production networks, would significantly improve the external validity and ecological realism of model evaluation.

6.6.2 Real-Time Performance Evaluation

Another important direction involves testing the framework in real-time scenarios using continuous data streams. Implementing the model within live detection pipelines or simulated enterprise networks would allow for assessment of latency, throughput, and false-alarm rates under realistic operational pressures. Such experiments would validate the feasibility of deploying the proposed architecture in real-time monitoring systems such as Security Operations Centers (SOCs) and Endpoint Detection and Response (EDR) platforms.

6.6.3 Integration of Advanced Deep Learning Architectures

Future research could integrate transformer-based models or graph neural networks (GNNs) to capture complex behavioral dependencies between processes, files, and network entities. Transformers can model long-term dependencies in sequential data, while GNNs can represent relationships as graphs, making them ideal for uncovering ransomware propagation pathways and multi-stage behaviors. These architectures may further improve classification granularity and resistance to obfuscation tactics.

6.6.4 Development of a Self-Adaptive Hybrid Framework

Building on the hybrid design proposed in this study, future work should explore the creation of a self-adaptive framework capable of autonomously adjusting its detection parameters based on evolving ransomware behavior patterns. This concept, inspired by the Ransomware-as-a-Service (RaaS) ecosystem, envisions a system that learns continuously from live feedback and threat intelligence feeds, allowing for automatic recalibration of feature importance, thresholds, and model weights.

6.6.5 Ethical and Policy Considerations in Hybrid Ransomware Detection

Future research should not only advance the technical aspects of ransomware detection but also integrate ethical and policy-oriented perspectives. As artificial intelligence becomes increasingly embedded in cybersecurity operations, ethical governance and regulatory compliance emerge as critical dimensions of responsible innovation.

In the specific context of hybrid ransomware detection, ethical considerations include the protection of user privacy, data transparency, and the proportional use of monitoring mechanisms. Because behavior-based and network-level analyses may involve inspecting user files, process activities, and network communications, future systems must ensure strict adherence to privacy-by-design principles and relevant data-protection laws (e.g., GDPR).

Moreover, policy frameworks should be developed to define accountability boundaries, clarifying who is responsible for automated decisions or false alerts that might affect legitimate users or critical infrastructures. Interdisciplinary collaboration among AI researchers, cybersecurity practitioners, and policymakers is therefore essential to ensure that hybrid detection frameworks not only achieve technical excellence but also uphold ethical integrity and social trust.

6.7 Concluding Remarks

This dissertation presents a comprehensive and systematic investigation into ransomware detection through the development of a hybrid framework that integrates behavior-based analytics with advanced machine learning and deep learning paradigms. Rather than offering a descriptive overview, the study critically examines how multi-domain behavioral signals—spanning File System Monitoring (FSM), Process Behavior Analysis (PBA), and Network Behavior Analysis (NBA)—can be fused with intelligent classifiers such as Naïve Bayes, XGBoost, and LSTM to achieve an interpretable, scalable, and adaptive detection mechanism.

The findings demonstrate that a multi-layered integration strategy not only enhances detection accuracy but also improves model transparency, an attribute essential for security analysts to trace alerts back to concrete behavioral evidence. By merging human-interpretable behavioral logic with data-driven inference, the research bridges the gap between theoretical cybersecurity frameworks and deployable defense systems.

Additionally, the study advances the academic discourse on ensemble and meta-learning by empirically validating the superior performance of stacking architectures over traditional voting ensembles in ransomware detection contexts. It further underscores the necessity of addressing class imbalance, dataset heterogeneity, and model calibration to sustain fairness and generalization in imbalanced cybersecurity datasets.

This research makes several novel contributions to the field of ransomware detection. First, it proposes a multi-layer behavioral integration framework that unifies File System Monitoring (FSM), Process Behavior Analysis (PBA), Network Behavior Analysis (NBA), traditional Machine Learning models, and temporal LSTM-based modelling within a single detection pipeline. This layered design enables complementary behavioural signals to be captured across static, probabilistic, and sequential perspectives, enhancing detection coverage beyond isolated methods. Second, the research provides a systematic comparison of multiple fusion strategies, including decision-level fusion (majority and weighted voting) and model-level fusion through stacking, evaluated under identical experimental conditions. This comparison offers empirical insights into the trade-offs between interpretability, performance, and complexity across fusion paradigms. Finally, the robustness of the proposed framework is validated through a dedicated noise sensitivity analysis, demonstrating that the integrated detection system maintains stable performance under realistic confidence perturbations. Together, these contributions advance the understanding of how heterogeneous behavioural and learning-based models can be effectively combined to produce robust and deployment-ready ransomware detection systems.

From a knowledge-contribution perspective, this work fills a notable gap in the literature by operationalizing a *cross-domain hybrid detection framework* that unites host-level and network-level behaviors within a synchronized temporal structure, a dimension often overlooked in prior studies limited to single-domain or static analyses. Through this integration, the research introduces a replicable methodology for unifying behavioral and learning-based paradigms, thus providing a conceptual and empirical foundation for next-generation, context-aware cybersecurity architectures.

Ultimately, the study positions hybrid ensemble learning as a pivotal direction in intelligent ransomware defense, where interpretability, adaptability, and scalability converge to form a resilient safeguard against evolving cyber threats. The contributions

herein, therefore, extend beyond technical performance, establishing an evidence-based pathway for developing ethically grounded, explainable, and sustainable AI-driven security systems.

References

1. Lee, J., Yun, J., & Lee, K. (2024a). A study on countermeasures against neutralizing technology: Encoding algorithm-based ransomware detection methods using machine learning. *Electronics*, *13*(6), 1030. https://doi.org/10.3390/electronics13061030.

2. Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B. C. M., & Assi, C. (2023). The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3268535.

3. Lee, J., Yun, J., & Lee, K. (2024). A study on countermeasures against neutralizing technology: Encoding algorithm-based ransomware detection methods using machine learning. *Electronics*, *13*(6), 1030. https://doi.org/10.3390/electronics13061030.

4. Patel, P., & Yashaswini, B. M. (2024). *The Evolution of Ransomware: Trends, Tactics, and Countermeasures*. Journal of Emerging Technologies and Innovative Research, 11(3). https://www.jetir.org/papers/JETIR2403171.pdf.

5. Li, J., Yang, G., & Shao, Y. (2024). Ransomware detection model based on adaptive graph neural network learning. *Applied Sciences (Basel, Switzerland)*, *14*(11), 4579. https://doi.org/10.3390/app14114579.

6. Anghel, M., & Racautanu, A. (n.d.). *A note on different types of ransomware attacks*. Iacr.org. Retrieved July 25, 2024, from https://eprint.iacr.org/2019/605.pdf.

7. Nagar, G. (2024). The evolution of ransomware: Tactics, techniques, and mitigation strategies. *International Journal of Scientific Research and Management*, *12*(06), 1282–1298. https://doi.org/10.18535/ijsrm/v12i06.ec09.

8. Pratt, M. K. (2023, September 13). *The 10 biggest ransomware attacks in history*. Security; TechTarget. https://www.techtarget.com/searchsecurity/tip/The-biggest-ransomware-attacks-in-history.

9. Begovic, K., Al-Ali, A., & Malluhi, Q. (2023). Cryptographic ransomware encryption detection: Survey. *Computers & Security*, *132*(103349), 103349. https://doi.org/10.1016/j.cose.2023.103349.

10. Yamany, B., Elsayed, M. S., Jurcut, A. D., Abdelbaki, N., & Azer, M. A. (2024). A holistic approach to ransomware classification: Leveraging static and dynamic analysis with visualization. *Information (Basel)*, *15*(1), 46. https://doi.org/10.3390/info15010046.

11. Azeem, M., Khan, D., Iftikhar, S., Bawazeer, S., & Alzahrani, M. (2024). Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches. *Heliyon*, *10*(1), e23574. https://doi.org/10.1016/j.heliyon.2023.e23574.

12. Cen, M., Jiang, F., Qin, X., Jiang, Q., & Doss, R.(2024).Ransomware early detection: A survey. *Computer Networks*, *239*(110138),110138. https://doi.org/10.1016/j.comnet.2023.110138.

13. Zimba, A., Chishimba, M., & Chihana, S. (2021). A ransomware classification framework based on file-deletion and file-encryption attack structures. *arXiv*. https://arxiv.org/abs/2102.10632.

14. Oz, H., Aris, A., Levi, A., & Uluagac, A. S. (2022). A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys*. https://doi.org/10.1145/3514229.

15. Fachkha, C., Mansoor, W., Fung, B. C. M., Assi, C., Gawanmeh, A., & Markarian, C. (2023). *The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions*. Retrieved from https://www.researchgate.net/publication/370137991

16. Smorti, M. (2022/2023). *Analysis and improvement of ransomware detection techniques* (Master's thesis). Politecnico di Torino. https://webthesis.biblio.polito.it/secure/26631/1/tesi.pdf

17. Cen, M., Jiang, F., Qin, X., Jiang, Q., & Doss, R.(2024).Ransomware early detection: A survey. *Computer Networks*, *239*(110138),110138. https://doi.org/10.1016/j.comnet.2023.110138.

18. Kwon, H.-Y., Kim, T., & Lee, M.-K. (2022). Advanced intrusion detection combining signature-based and behavior-based detection methods. *Electronics*, *11*(6), 867. https://doi.org/10.3390/electronics11060867.

19. Agate, V., D'Anna, F. M., De Paola, A., Ferraro, P., Lo Re, G., & Morana, M. (2022). *A behavior-based intrusion detection system using ensemble learning techniques*. Unipa.It. Retrieved August 8, 2024, from https://iris.unipa.it/retrieve/342ffb51-84fd-47b9-a5eb ae8bd10b60be/TOC_preface_paper_ITASEC2022.pdf.

20. De Gaspari, F., Hitaj, D., Pagnotta, G., De Carli, L., & Mancini, L. V. (2022). Evading behavioral classifiers: a comprehensive analysis on evading ransomware detection techniques. *Neural Computing & Applications*, *34*(14), 12077–12096. https://doi.org/10.1007/s00521-022-07096-6.

21. Wang, S., Qin, S., Qin, J., Zhang, H., Tu, T., Jin, Z., & Guo, J. (2021). KRDroid: Ransomware-oriented detector for mobile devices based on behaviors. *Applied Sciences (Basel, Switzerland)*, *11*(14), 6557. https://doi.org/10.3390/app11146557.

22. Urooj, U., Al-Rimy, B. A. S., Zainal, A. B., Saeed, F., Abdelmaboud, A., & Nagmeldin, W. (2024). *Addressing behavioral drift in ransomware early detection through weighted generative adversarial networks*. IEEE Access, 12, 3910-3925. https://doi.org/10.1109/ACCESS.2023.3348451

23. Yu, R., Li, P., Hu, J., Chen, L., Zhang, L., Qiu, X., & Wang, F. (2024). *Ransomware Detection Using Dynamic Behavioral Profiling: A Novel Approach for Real-Time Threat Mitigation*. TechRxiv. Retrieved from https://www.techrxiv.org/users/847935/articles/1235586-ransomware-detection-using-dynamic-behavioral-profiling-a-novel-approach-for-real-time-threat-mitigation.

24. Azeem, M., Khan, D., Iftikhar, S., Bawazeer, S., & Alzahrani, M. (2024). Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches. *Heliyon*, *10*(1), e23574. https://doi.org/10.1016/j.heliyon.2023.e23574.

25. Yaseen, Q. M. (2023). The effect of the ransomware dataset age on the detection accuracy of machine learning models. *Information (Basel)*, *14*(3), 193. https://doi.org/10.3390/info14030193.

26. Aljabri, M., Alhaidari, F., Albuainain, A., Alrashidi, S., Alansari, J., Alqahtani, W., & Alshaya, J. (2024). Ransomware detection based on machine learning using memory features. *Egyptian Informatics Journal*, *25*(100445), 100445. https://doi.org/10.1016/j.eij.2024.100445.

27. Woralert, C., Liu, C., & Blasingame, Z. (2024). *Towards effective machine learning models for ransomware detection via low-level hardware information*. *International Workshop on Hardware and Architectural Support for Security and Privacy (HASP '24)*. https://doi.org/10.1145/3696843.3696847

28. Gheisari, M., & Belaton, B. (2023). *Machine learning approaches to ransomware detection: A comprehensive review*. Retrieved from https://www.researchgate.net/publication/387695399_Machine_Learning_Approaches_to_Ransomware_Detection_A_Comprehensive_Review.

29. Zahoora, U., Khan, A., Rajarajan, M., Khan, S. H., Asam, M., & Jamal, T. (2022). Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier. *Scientific Reports*, *12*(1), 1–15. https://doi.org/10.1038/s41598-022-19443-7.

30. Ispahany, J., Islam, M. R., Khan, M. A., & Islam, M. Z. (2025). *iCNN-LSTM: A batch-based incremental ransomware detection system using Sysmon*. arXiv preprint. https://arxiv.org/abs/2501.01083

31. Kiyol, N., Gür, S., & Karakaya, M**.** (2024). *Ransomware Detection Using LSTM Networks and File Entropy Analysis: A Sequence-Based Approach*. Research Square. https://doi.org/10.21203/rs.3.rs-5400622/v1

32. Gazzan, M., & Sheldon, F. T. (2024). Novel ransomware detection exploiting uncertainty and calibration quality measures using deep learning. *Information (Basel)*, *15*(5), 262. https://doi.org/10.3390/info15050262.

33. Begovic, K., Al-Ali, A., & Malluhi, Q. (2023). Cryptographic ransomware encryption detection: Survey. *Computers & Security*, *132*(103349), 103349. https://doi.org/10.1016/j.cose.2023.103349.

34. Patel, P., & Yashaswini, B. M. (2024). *The Evolution of Ransomware: Trends, Tactics, and Countermeasures*. Journal of Emerging Technologies and Innovative Research, 11(3). https://www.jetir.org/papers/JETIR2403171.pdf.

35. Kwon, H.-Y., Kim, T., & Lee, M.-K. (2022). Advanced intrusion detection combining signature-based and behavior-based detection methods. *Electronics*, *11*(6), 867. https://doi.org/10.3390/electronics11060867.

36. Agate, V., D'Anna, F. M., De Paola, A., Ferraro, P., Lo Re, G., & Morana, M. (2022). *A behavior-based intrusion detection system using ensemble learning techniques*. Unipa.It. Retrieved August 8, 2024, from https://iris.unipa.it/retrieve/342ffb51-84fd-47b9-a5eb ae8bd10b60be/TOC_preface_paper_ITASEC2022.pdf.

37. Yu, R., Li, P., Hu, J., Chen, L., Zhang, L., Qiu, X., & Wang, F. (2024). *Ransomware Detection Using Dynamic Behavioral Profiling: A Novel Approach for Real-Time Threat Mitigation*. TechRxiv. Retrieved from https://www.techrxiv.org/users/847935/articles/1235586-ransomware-detection-using-dynamic-behavioral-profiling-a-novel-approach-for-real-time-threat-mitigation.

38. Karapoola, S., Singh, N., Rebeiro, C., & Kamakoti, V. (2022). SUNDEW: An Ensemble of Predictors for Case-Sensitive Detection of Malware. *arXiv*. https://arxiv.org/abs/2211.06153.

39. Oliveira, G. (2025). *RansomSet: A dataset for ransomware detection & analysis* [Data set]. GitHub repository. Retrieved from https://github.com/gabrielolivs/RansomSet

40. Berrueta, E., Morató, D., Magaña, E., & Izal, M. (2020). *Open repository for the evaluation of ransomware detection tools*. *IEEE Access, 8*, 65658–65669. https://doi.org/10.1109/ACCESS.2020.2984187

41. Zahoora et al. (2022). Ransomware Detection Using Hybrid Behavior and Machine Learning Features. Journal of Cyber Security Technology. https://doi.org/10.1080/23742917.2022.2027138

42. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

43. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

44. Saito, T., & Rehmsmeier, M. (2015). The precision–recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE, 10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432
45. Davis, J., & Goadrich, M. (2006). The relationship between Precision–Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, 233–240. https://doi.org/10.1145/1143844.1143874

46. He, H., & Garcia, E. A. (2009). *Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

47. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

48. Wolpert, D. H. (1992). *Stacked generalization. Neural Networks, 5*(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1

49. Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research, 10*, 271–289. https://doi.org/10.1613/jair.594

50. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(4), 463–484. https://doi.org/10.1109/TSMCC.2011.2161285

51. Wang, S., Qin, S., Qin, J., Zhang, H., Tu, T., Jin, Z., & Guo, J. (2021). *KRDroid: Ransomware-Oriented Detector for Mobile Devices Based on Behaviors*. **Applied Sciences, 11**(6557). https://doi.org/10.3390/app11146557

52. Altaf, M., Iqbal, Z., Imran, A., Muhammad, Z., & Shafi, Q. (2025). *RanDEL: Dynamic Feature-Based Ransomware Detection and Classification Using Advanced Ensemble Techniques*. **Preprints**, 2025021824. https://doi.org/10.20944/preprints202502.1824.v1

53. Urooj, U., Al-Rimy, B. A. S., Gazzan, M., Zainal, A., Amer, E., Almutairi, M., Shiaeles, S., & Sheldon, F. (2025). *A Wide and Weighted Deep Ensemble Model for Behavioral Drifting Ransomware Attacks*. **Mathematics, 13**(1037). https://doi.org/10.3390/math13071037

54. Wang, S., Qin, S., Qin, J., Zhang, H., Tu, T., Jin, Z., & Guo, J. (2021). *KRDroid: Ransomware-Oriented Detector for Mobile Devices Based on Behaviors*. Applied Sciences, 11(6557). https://doi.org/10.3390/app11146557

55. Singh, A., Kumar, R., & Gupta, M. (2025). *A hybrid ensemble model for ransomware detection using feature engineering and deep learning. International Journal of Information Technology, 27*(1), 34–47. https://www.researchgate.net/publication/394420607_A_hybrid_ensemble_model_for_ransomware_detection_using_feature_engineering_and_deep_learning

56. Alhayan, F. (2025). *Voting-based ensemble classifiers model on ransomware detection for cybersecurity-driven IIoT in cloud computing infrastructure. Journal of Intelligent Systems and Internet of Things, 8*(2), 55–69. https://www.sciencedirect.com/science/article/pii/S2772918424000420

57. Conway, D., & Centonze, A. (2025). *A hybrid ransomware detection scheme combining heuristic rule-based and machine-learning approaches. Computers & Security, 143,* 104073. https://www.sciencedirect.com/science/article/pii/S0167404824003143