1.5em 0pt

A Dual Feature Approach to Ransomware Detection Using CatBoost and LightGBM

Capestone Project-II report

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY
In
CYBER SECURITY

by

Venkata Sai Swapna Pallaothu 24EP03011

Under the Guidance of

Dr.T. Meena , M.Tech, Ph.D
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

Kanuru, Vijayawada 520007

April 2025

SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Capestone Project-II report entitled **”A Dual Feature Approach to Ransomware Detection Using CatBoost and LightGBM”** being submitted by **Venkata Sai Swapna Pallapothu (24EP03011)** in partial fulfillment for the award of the Degree of Master of Technology in Cyber Security to the Siddhartha Academy of Higher Education, is a record of bonafide work carried out during the period from 2024 - 2025.

Dr.T. Meena, M.Tech, Ph.D	**Dr. D. Rajeswara Rao,** M.Tech, Ph.D
Assistant Professor & Guide	**Professor & HOD**

DECLARATION

I hereby declare that the Capestone Project-II Report entitled "A Dual Feature Approach to Ransomware Detection Using CatBoost and LightGBM " submitted in partial fulfillement for the award of M.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles.

Place: Vijayawada

Date: Venkata Sai Swapna Pallapothu (24EP03011)

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without whom it would ever have come into existence.To them I lay the words of gratitude imprinted with me.

I would like to thank **Professor P.Venkateswara Rao**,Vice-Chancellor and **Dr.A.V.Ratna Prasad** Pro Vice-Chancellor of Siddhartha Academy of Higher Education for the facilities provided during the course of Capestone Project II. I have been bestowed with the privilege of thanking **Dr. D. Rajeswara Rao**, Professor and Head of the Department for his moral and material support.

I would like to express my deep gratitude to my guide **Dr.T.Meena**, Assistant Professor for her persistent encouragement, everlasting patience, and keen interest in discussion, and for her numerous suggestions which I had at every phase of this project. I owe my acknowledgments to an equally long list of people who helped me in Capestone Project-II work.

Place: Vijayawada

Date: Venkata Sai Swapna Pallapothu (24EP03011)

Abstract

Ransomware poses a significant cybersecurity risk, frequently leading to considerable financial and data loss by encrypting files and requesting payment for their recovery. Conventional detection approaches frequently fail because of the swift advancement of ransomware strategies. This study introduces a hybrid malware detection system that integrates static and dynamic analysis techniques to enhance detection precision. Static characteristics are obtained from Portable Executable (PE) headers utilizing the Ransomware PE Header Feature Dataset. Simultaneously, dynamic characteristics like file alterations, registry updates, and network activities are examined to capture the live behavior of ransomware. By combining these two feature types, the system can more effectively recognize and distinguish ransomware from regular applications. To identify and classify ransomware, the system employs two robust machine learning models: CatBoost and LightGBM. Feature importance analysis helps determine which traits are the best indicators of malicious behavior. The system is assessed with common metrics like accuracy, precision, recall, and F1-score, demonstrating excellent capability in identifying sophisticated ransomware attacks. This study offers a dependable and scalable method for detecting ransomware in real time within modern dynamic computing settings.

Keywords: Ransomware Detection, Static and Dynamic Analysis, Machine Learning, CatBoost, LightGBM, PE Header Features, Behavioral Analysis, Cybersecurity

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

Ransomware has risen to become one of the most perilous and prevalent types of cyber threats in recent times, impacting individuals, businesses, and governments equally. It is a form of harmful software that enciphers a victim's files or restricts their system, requesting a ransom for regaining access. The emergence of advanced ransomware types like WannaCry, Ryuk, LockBit, and Conti has revealed the shortcomings of conventional antivirus tools, which mainly depend on static signatures and heuristic detection methods. These traditional methods frequently struggle to identify polymorphic or zero-day ransomware variants that quickly change and conceal their actions.

To overcome these limitations, this project suggests a dual-feature ransomware detection system that utilizes both static and dynamic analysis along with sophisticated machine learning methods. Static analysis entails examining files without running them, gathering characteristics like attributes from the Portable Executable (PE) header. In contrast, dynamic analysis observes live behaviors such as system calls, changes to the registry, file encryption, and network activity while the program is running. Integrating these two viewpoints can lead to a more thorough and precise understanding of ransomware behavior.

This combination of features is utilized to train two gradient boosting classifiers—CatBoost and LightGBM—that are especially well-suited for managing complex, structured, and categorical data. CatBoost is recognized for its excellent management of categorical variables, whereas LightGBM offers rapid speed and effectiveness in training extensive datasets. These models are utilized to identify and categorize ransomware using both structural and behavioral indicators.

The main objective of this project is to develop a strong, scalable, and interpretable ransomware detection system that functions in real-time and minimizes false positives. It aims to go beyond the limitations of conventional signature-based and solely behavior-based detection methods. Utilizing a dual-feature strategy and incorporating ensemble machine learning, this system seeks to identify recognized, unrecognized, and emerging ransomware with improved precision and durability.

In addition, this project adds to the expanding literature on cybersecurity and machine learning by incorporating insights from current studies and tackling research deficits like real-time detection, model interpretability, and hybrid feature fusion. In the end, the system could be utilized in business security solutions, endpoint detection and response (EDR), and Security Operations Centers (SOC) aiming for enhanced ransomware protection strategies.

1.1 Basic Concepts

1.1.1 Ransomware

Ransomware is a type of malware that encrypts the victim's files or locks them out of their systems, requiring a ransom to regain access. In recent years, it has emerged as one of the most perilous and expensive cyber threats, impacting individuals, enterprises, and vital infrastructure. There are two main categories of ransomware: locker ransomware, which restricts access to the whole system, and crypto ransomware, which encrypts user files while still allowing system access. After infection, users usually receive a ransom note requesting payment, frequently in cryptocurrency, for a decryption key. Sophisticated ransomware groups also pose a risk of publicly exposing data if the ransom goes unpaid, referred to as double extortion. These attacks frequently take advantage of software weaknesses or employ social engineering strategies, like phishing emails, to breach systems

1.1.2 Static Analysis

Static analysis is a method employed to examine software or files without running them. In malware detection, it entails analyzing the framework and metadata of files, including their Portable Executable (PE) headers, to identify relevant features. PE headers include information such as file size, section names, import/export tables, and compilation details. As this analysis doesn't need code execution, it is quite quick and secure, making it perfect for examining substantial amounts of files. Nonetheless, static analysis has its constraints—it can be bypassed by code obfuscation, packing, or encryption employed by advanced ransomware to conceal malicious intentions. In spite of these drawbacks, static features are beneficial as they offer a rapid and scalable initial detection layer.

1.1.3 Dynamic Analysis

Dynamic analysis, on the other hand, entails running the program in a managed setting (such as a sandbox) and monitoring its actions as they occur. This encompasses tracking alterations to files, changes in registry keys, the creation of processes, system calls, and network activity. Dynamic analysis can identify ransomware behavior by examining a program's actions during execution, which static analysis may overlook, particularly with polymorphic or metamorphic malware. Nonetheless, dynamic analysis demands additional computational resources and time. Ransomware may also circumvent it by recognizing virtual environments and postponing execution. Regardless, it offers crucial behavioral signals for creating strong detection systems based on machine learning.

1.1.4 Portable Executable (PE) Headers

PE headers are components of the Windows executable file format and play a vital role in static malware analysis. They hold organized metadata regarding how the operating system ought to load and run the file. Key PE header fields consist of the section count, import directory, timestamp, image size, and subsystem. These areas frequently show concerning patterns—such as odd import functions or altered time stamps—that may suggest harmful intentions. Because developers of ransomware seldom reconstruct these headers by hand, numerous types of ransomware exhibit noticeable irregularities in their PE structures, which can be advantageous for feature extraction during static analysis

1.1.5 Behavioral Features

Behavioral characteristics are derived from dynamic analysis and illustrate how the ransomware engages with the system while being executed. This encompasses actions like bulk file renaming (encryption), illicit entry to user directories, registry alterations (for instance, turning off system restore), and outgoing links to command-and-control (C2) servers. These characteristics aid in identifying ransomware by its actions instead of its appearance, thereby enhancing detection's resistance to obfuscation. To capture behavioral characteristics, tools such as Cuckoo Sandbox, Process Monitor, or Wireshark are necessary to observe real-time actions while malware is executing.

1.1.6 Machine Learning in Ransomware Detection

Machine learning (ML) is being utilized more in cybersecurity because it can identify patterns and anomalies not governed by clear rules. In ransomware detection, ML models are developed using labeled datasets that include both benign and malicious samples, allowing the system to understand intricate relationships between features and results. The dual-feature method integrates both static and dynamic attributes as inputs to these models, enhancing overall accuracy and reliability. ML models additionally offer feature importance ranking, granting interpretability and understanding of the decision-making process.

1.1.7 CatBoost

CatBoost is a gradient boosting algorithm created by Yandex, intended to manage categorical features effectively without the need for extensive preprocessing such as one-hot encoding. It constructs decision trees in an iterative manner and employs ordered boosting along with symmetric trees to minimize overfitting and enhance generalization. CatBoost is ideal for ransomware detection because it can effectively manage structured, tabular datasets like those obtained from PE headers and behavioral logs. It also provides rapid training and impressive accuracy, which makes it perfect for real-time cybersecurity uses.

1.1.8 LightGBM

LightGBM (Light Gradient Boosting Machine) is a quick, distributed, high-efficiency gradient boosting framework created by Microsoft. It employs a histogram-based approach and a leaf-wise tree growth method, enhancing training efficiency and lowering memory consumption. LightGBM is especially efficient in managing large-scale datasets and enables parallel training, making it ideal for cloud-based detection systems. In the realm of ransomware detection, LightGBM can utilize both static and dynamic characteristics to accurately classify files or activities while ensuring minimal latency.

1.1.9 Hybrid Analysis

Hybrid analysis involves the combination of static and dynamic analysis methods to offer a thorough detection strategy. Static analysis swiftly sifts through files according to their structural features, whereas dynamic analysis offers more profound understanding of behavior during execution. Merging both forms of analysis allows detection systems to more effectively manage evasive, obfuscated, and zero-day ransomware threats. This twofold strategy utilizes the advantages of each method and offsets their specific shortcomings.

1.1.10 Feature Engineering and Importance

Feature engineering involves choosing and converting data into significant inputs for machine learning models. In this project, characteristics are extracted from both PE headers (e.g., count of imports, entropy) and runtime behaviors (e.g., count of files altered, changes to registry keys). Feature importance analysis is subsequently utilized to identify which features most significantly aid in the detection of ransomware. Grasping feature importance enhances model performance and boosts the interpretability of predictions, which is essential in cybersecurity applications where trust and transparency are critical

1.2 Project Motivation

Ransomware assaults have emerged as one of the most widespread and harmful types of cybercrime, leading to significant disruptions, data theft, and financial damage to individuals, businesses, and governments. Conventional antivirus solutions and signature-based detection approaches are becoming less effective against emerging and advancing ransomware variants that employ obfuscation, polymorphism, and advanced evasion strategies. This concerning trend highlights the necessity for smart, preemptive detection systems that can precisely recognize ransomware before serious harm occurs. The aim of this project is to develop a strong, real-time ransomware detection system that leverages both structural and behavioral traits of executable files to identify threats more efficiently, even when encountering unknown or zero-day ransomware variants.

1.3 Problem Statement

Conventional ransomware detection techniques, like signature-based and heuristic methods, frequently struggle to identify newly developed or hidden ransomware variants because they depend on established patterns. This makes systems susceptible to zero-day exploits and advanced malware that can bypass standard detection methods. Current static or dynamic analysis methods on their own are inadequate for thorough threat detection. Consequently, there is a requirement for a smart and adaptable detection system that efficiently merges both static (structural) and dynamic (behavioral) characteristics to accurately identify and classify ransomware. This project seeks to address this shortfall by suggesting a dual-feature detection system enhanced by sophisticated machine learning models—CatBoost and LightGBM—to boost accuracy, robustness, and adaptability against contemporary ransomware threats.

1.4 Objectives

- To create a hybrid ransomware detection system integrating static analysis (PE header characteristics) and dynamic analysis (behavioral information).
- To derive significant features from executable files by utilizing both pre-execution and runtime behavior.

- To develop and train machine learning models CatBoost and LightGBM for precise classification of ransomware and non-malicious files.
- To combine static and dynamic characteristics into one dataset for thorough analysis. To assess the system with conventional performance metrics like accuracy, precision, recall, and F1-score.
- To conduct an analysis of feature importance to identify the factors that have the greatest impact on ransomware detection.
- To guarantee that the system can effectively generalize to both known and unknown ransomware variations.

1.5 Scope

This initiative centers on creating and assessing a ransomware detection system that employs both static and dynamic characteristics for machine learning-driven classification. It includes feature extraction from Portable Executable (PE) headers as well as behavioral logs produced during runtime, like file changes, registry alterations, and network activity. The system utilizes labeled datasets for training and testing, employing CatBoost and LightGBM as the main classifiers. The scope encompasses the design of the detection framework, training and testing of the model, evaluation of performance, and presentation of findings. Nonetheless, the project lacks automated reply systems or real-time implementation in production settings, which can be tackled in subsequent efforts.

Chapter 2

LITERATURE REVIEW

This chapter contains the list of journals that we have studied under literature survey. In these papers, I focussed on the approaches for maintaining accuracy.

1. Svet et al. – "Unveiling Zero-Space Detection: A Novel Framework for Autonomous Ransomware Identification in High-Velocity Environments"

Methodology:

The article suggests the Zero-Space Detection framework that employs autonomous learning agents driven by unsupervised machine learning techniques. It utilizes real-time telemetry data streams along with anomaly detection through clustering and ensemble deep learning techniques. The model functions in fast-paced settings (e.g., cloud infrastructure), where the system constantly updates its training using emerging patterns. It removes reliance on signatures through the use of latent behavior-based modeling and a modular, pluggable architecture that works with real-time pipelines.

Advantages:

- Extremely flexible system proficient in independently acquiring new ransomware patterns.
- Real-time retraining guarantees applicability to zero-day and polymorphic threats.
- Ensemble learning boosts decision precision and minimizes the risk of a single failure point.
- Modular architecture allows for seamless integration with current cybersecurity tools.

Disadvantages:

- Demands significant computational power for processing telemetry in real-time.
- Unsupervised learning can lead to more false positives if not tuned correctly.

- If not optimized, real-time model retraining can cause delays.
- Restricted clarity in deep ensemble models might influence interpretability.

2. Urooj et al. – "Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions"

Methodology:

This document reviews different methods for ransomware detection that rely on dynamic analysis. It emphasizes recognizing behavioral signs like file encryption actions, registry alterations, and API usage trends. The study classifies detection into behavior-based, hybrid, and entirely machine learning-driven models, which encompass Random Forests, SVMs, and deep learning methods. It highlights a hybrid approach that merges dynamic behavioral signals with powerful ML classifiers for strong detection.

Advantages:

- Offers an in-depth comprehension of the fluctuating indicators of ransomware.
- Emphasizes the advantages of various machine learning classifiers.
- Promotes immediate feature gathering for precise malware identification.
- Provides perspectives on identifying advanced persistent threats (APTs).

Disadvantages:

- Outcomes are influenced by the caliber of dynamic analysis tools utilized.
- The overhead caused by runtime monitoring can impact system performance.
- Dynamic analysis by itself might fail to identify inactive ransomware.
- Extracting features is intricate and can overlook inconspicuous behavior.

3. Fernando et al.– "A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques"

Methodology:

This research follows the progression from conventional ML models (Decision Trees, Naive Bayes, SVM) to advanced deep learning models including CNNs, RNNs, and Autoencoders. It offers a two-tier detection system that combines Markov Chains and Decision Trees, while also assessing multi-layer streaming analytics across a period of 30 days. It focuses on utilizing Autoencoders for detecting anomalies and covers topics such as dataset imbalance, interpretability, and the computational expense in deep learning models.

Advantages:

- Displays trends in the shift from traditional approaches to deep learning for ransomware.
- Establishes a basis for employing Autoencoders in identifying anomalies.
- Streaming analytics enables the system to function effectively in real-time settings.
- Deep learning identifies intricate behavioral trends in the evolution of ransomware.

Disadvantages:

- Significant resource usage of deep learning models.
- Autoencoders might need adjustment to decrease false positives.
- Deploying and maintaining streaming frameworks is challenging.
- Might not generalize effectively without a variety of training data.

4. Woralert et al. – "Towards Effective Machine Learning Models for Ransomware Detection via Low-Level Hardware Information"

Methodology:

This paper presents a low-level ransomware detection system utilizing hardware performance counters (HPCs). It tracks CPU cycle counts, memory accesses, and cache misses to identify ransomware-like behavior. Lightweight machine learning classifiers, such as Logistic Regression and Ensemble Trees, are trained on hardware-level data to detect system anomalies almost in real-time. This approach allows for discreet, low-resource identification directly from hardware signals.

Advantages:

- Low system overhead enables real-time detection even on edge devices.
- Hardware telemetry is impervious to obfuscation and API evasion methods.
- Appropriate for embedded or IoT systems that have limited resources.
- Provides a fresh aspect of data sources that traditional models have not investigated.

Disadvantages:

- Demands particular hardware and access to HPCs, restricting widespread use.
- Not all ransomware shows distinct hardware patterns at the beginning of execution.
- Might trigger false alerts because of harmless high-resource applications.
- Challenging to link hardware actions with specific harmful behavior.

5. Kolodenker et al.– "PayBreak: Defense Against Cryptographic Ransomware"

Methodology:

PayBreak is an active cryptographic protection tool that captures encryption keys while the program is running. It connects to Windows Crypto API functions and logs encryption keys utilized by ransomware prior to their use for harmful encryption. This enables the decryption of compromised files without the necessity of paying a ransom. The tool functions autonomously of ML methods, concentrating exclusively on intercepting cryptographic processes.

Advantages:

- Proactive strategy that is effective even prior to the response of detection systems.
- Facilitates the restoration of files without requiring decryption keys for ransomware.
- Does not rely on previous understanding of ransomware types or identifiers.
- Minimal false positive rate as it concentrates on genuine encryption methods.

Disadvantages:

- Applicable solely to ransomware employing standard cryptographic APIs.
- Ineffective against ransomware that employs custom or obfuscated encryption methods.
- Restricted effectiveness when used alone—more effective within a multi-layered defense system.
- Doesn't categorize or gain insights from ransomware actions.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Functional Requirements

It reqires the various software and hardware requirements
File Upload and Input Management : The system should enable users to submit executable files (.exe) for examination.
Static Feature Extraction: The system must retrieve PE header characteristics from the submitted executable.
Dynamic Behavior Observation : The system needs to run the file in a sandbox environment while observing changes to the file system, alterations to the registry, and network behavior.
Module for Feature Fusion : Static and dynamic attributes need to be integrated into a cohesive feature vector.
Classifying with Machine Learning : The system needs to categorize the executable as either benign or ransomware utilizing CatBoost and LightGBM models.
Assessment of Performance Metrics : The system should assess model performance by utilizing accuracy, precision, recall, and F1-score.
Analysis of Feature Importance: The system must emphasize important attributes that affect the model's choices.

3.1.1 Software Requirements

Python
The project utilizes the Python programming language, thus requiring Python to be installed on the system. It is advisable to utilize Python version 3.7 or newer to guarantee compatibility with contemporary libraries employed in the project.
Libraries
Matplotlib:
Matplotlib is an extensive library for generating static, animated, and interactive visual representations in Python. This project utilizes it to graph evaluation metrics like accuracy, precision-recall curves, and feature importance, helping in the visual understanding of model effectiveness.
NumPy:
NumPy is the essential library for scientific computation using Python. It offers assistance for extensive, multi-dimensional arrays and matrices, alongside a set

of mathematical functions to efficiently manipulate these arrays. It is crucial for managing feature vectors throughout the model training procedure.

scikit-learn:

Scikit-learn, commonly referred to as sklearn, is a popular library for machine learning in Python. It offers effective tools for data preprocessing, feature scaling, model assessment, and training traditional ML models like Decision Trees and Random Forests. In this project, it is utilized to calculate accuracy, precision, recall, and F1-score.

Pandas:

Pandas is an influential tool for data analysis and manipulation that is developed on the foundations of NumPy. It offers data structures such as DataFrames that help in structuring and preparing the dataset prior to inputting it into machine learning models.

CatBoost

CatBoost is a powerful gradient boosting library created by Yandex. It is particularly designed for managing categorical features and is utilized in this project to categorize ransomware using the integrated static and dynamic feature set.

LighGBM

LightGBM is an effective gradient boosting framework created by Microsoft. Renowned for its rapidity and precision, it is utilized to enhance CatBoost in developing a dual-model strategy for ransomware identification.

Google Colab

Google Colab offers a cloud environment for coding and running Python scripts. It is utilized for training and testing models, especially when local resources are scarce. To access and use notebooks in the cloud, a Google account is necessary.

3.1.2 Hardware Requirements

Central Processing Unit (CPU) :

It is advised to have at least an Intel Core i5 (8th generation or newer) or AMD Ryzen 5 processor. A multi-core processor facilitates effective management of extensive datasets and simultaneous execution of calculations during feature extraction and model training.

RAM (Random Access Memory):

A minimum of 8 GB of RAM is necessary to efficiently carry out feature processing and train machine learning models like CatBoost and LightGBM. For improved performance and larger datasets, it is suggested to have 16 GB or more.

Storage(HDD/SSD):

The system must possess at least 256 GB of accessible storage. SSD (Solid State Drive) storage is favored over HDD due to its quicker read/write speeds in data loading and model training tasks.

Graphics Processing Unit (GPU):
Although not required, having a GPU with a minimum of 4 GB of VRAM (like the NVIDIA GeForce GTX series) can speed up model training, particularly for upcoming enhancements related to deep learning.

Internet Connection:
An internet connection is required for accessing external datasets, libraries, and resources, especially if using cloud-based platforms like Google Colab or downloading geospatial data.Stable internet connection for downloading libraries, accessing online tools, updating models, and possibly sandboxing remote samples.

Operating System:
Operating System The project can be carried out on systems running Windows, Linux, or macOS. Nevertheless, Linux-based systems are typically advised for improved compatibility with open-source libraries and enhanced performance during batch process execution.

3.2 Non Functional Requirements

Performance:
The system must enable real-time or nearly real-time detection of ransomware through the analysis of both static and dynamic features. The classification model needs to provide a response within several seconds of feature input to maintain usability in real-world security settings.

Scalability:
The solution should be scalable to manage larger data volumes, including a rising quantity of executable files or expanded feature sets. Utilizing cloud-based platforms such as Google Colab guarantees scalability for training bigger models and handling large datasets.

Reliability:
The detection system needs to reliably provide precise outcomes while minimizing false positives and negatives. It should uphold consistent performance in various execution settings and file samples.

Security:
Given that the system handles potentially harmful files, it is crucial to guarantee secure sandboxing for dynamic analysis. All uploaded files and model results must be managed safely to avoid jeopardizing the host environment.

Usability:
The user interface, when created, ought to be intuitive and user-friendly for both IT professionals and security researchers. The output results, including detection labels and confidence scores, should be clearly shown.
Maintainability:
The architecture and codebase of the system must be modular and thoroughly documented to facilitate easy maintenance, updates, or future integration with additional security tools.
Portability:
The project must operate smoothly on various operating systems such as Windows, Linux, and macOS, guaranteeing cross-platform compatibility by leveraging Python's platform-agnostic characteristics.
Performance Metrics:
The employed machine learning classifiers (CatBoost and LightGBM) need to attain high accuracy, precision, recall, and F1-score metrics to guarantee successful ransomware detection and minimize false positives.

Chapter 4

ANALYSIS AND DESIGN

This chapter consists of the design of the software life cycle model diagrams and their detailed explanation. Design is about choosing the architecture and solutions appropriate to the problem

4.1 System Lifecycle

The Social Engineering Attack Lifecycle represents the sequential stages typically followed by attackers to manipulate individuals and exploit human vulnerabilities for malicious gain. The first stage is Investigation, where the attacker identifies and studies the target, gathering personal, professional, or organizational information through techniques such as social media analysis, phishing, or open-source intelligence. This is followed by the Hook phase, in which the attacker initiates contact and begins to engage the target with the intent of building trust. This may involve impersonation, fake identities, or crafting believable scenarios to gain the victim's confidence.

Once trust is established, the attacker moves to the Play stage, where the actual exploitation occurs. Here, the attacker extracts sensitive information such as credentials, financial data, or access codes, often without the victim realizing they are being deceived. The next phase is Exit/Cleanup, during which the attacker covers their tracks to avoid detection. This may involve deleting logs, removing communication traces, or manipulating systems to hide the breach. Finally, in the Aftermath stage, the attacker analyzes the impact of the attack, determines if further exploitation is possible, and evaluates how successfully the operation was executed. Understanding this lifecycle is crucial for developing effective security awareness programs and implementing proactive measures to detect and disrupt social engineering attacks. Detection Stage, where it starts examining live or incoming messages as they occur. The tokenized input is processed by the trained BERT model, which subsequently forecasts the likelihood of the message representing a social engineering attack. An appropriate classification is determined using thresholds or decision boundaries. Alert Generation entails that the system reacts to a positive detection by marking the message and notifying the relevant security staff or initiating an automated action (such as blocking the email or isolating the message). Notifications may feature confidence scores and the reasoning for detection to aid analysts in their decision-making process.

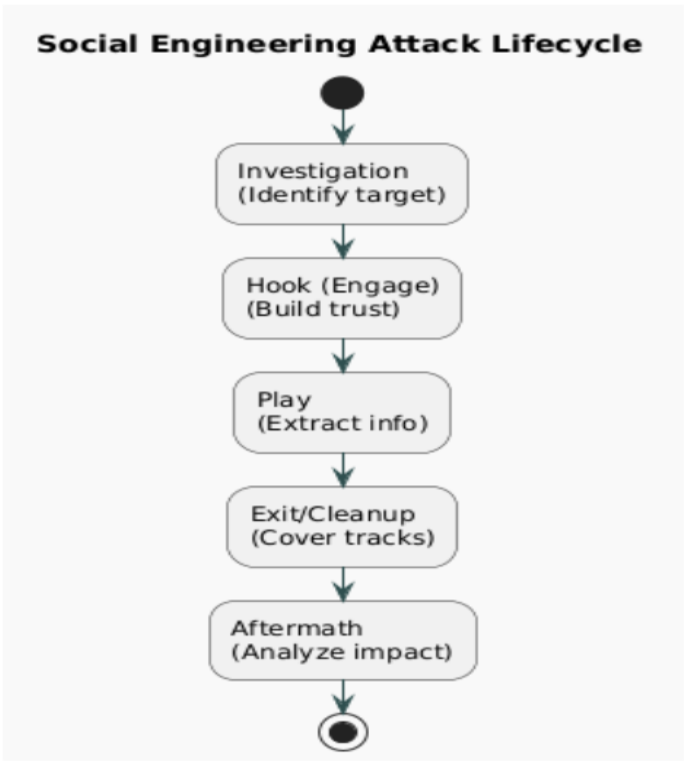

Social Engineering Attack Lifecycle
Investigation (Identify target)
Hook (Engage) (Build trust)
Play (Extract info)
Exit/Cleanup (Cover tracks)
Aftermath (Analyze impact)

Figure 4.1: Social engineering attack Lifecycle

4.2 Block Diagram

The block diagram integrates both static and dynamic analysis techniques in a unified, modular architecture to enhance the accuracy and robustness of ransomware identification. The system begins with the Data Acquisition Module, which collects executable files and distributes them to both the Static Analysis Module and the Dynamic Analysis Module. The static analysis module extracts features from the Portable Executable (PE) headers without executing the file, capturing characteristics like entropy, file size, and imported functions. In parallel, the dynamic analysis module runs the executables in a controlled sandbox environment to monitor their real-time behavior, including file modifications, registry changes, and network activity. The outputs from both modules are then passed to the Feature Engineering Module, where the extracted features undergo preprocessing, encoding, and normalization to ensure compatibility and consistency. Subsequently, the processed static and dynamic features are combined in the Feature Fusion Module, producing a single, comprehensive feature vector. This merged feature set is then fed into the Machine Learning Module, where two powerful models—CatBoost and LightGBM—are used to classify the executables as either ransomware or benign software. These models not only provide predictions but also contribute to the Feature Importance Module, which identifies the most significant features influencing the classification decision, aiding in model interpretability. The predictions from the machine learning module are further evaluated using the Evaluation Module, which computes standard performance metrics such as accuracy, precision, recall, and F1-score. Finally, the Detection Output Module delivers the classification result, marking the executable as either ransomware or safe. This comprehensive approach ensures a scalable, accurate, and interpretable solution for real-time ransomware detection in modern computing environments.

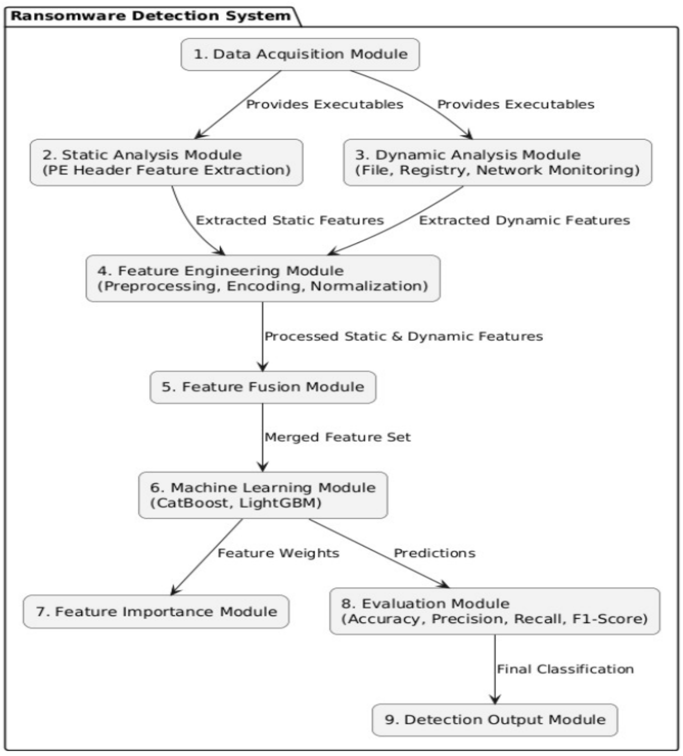

Ransomware Detection System
1. Data Acquisition Module
Provides Executables
Provides Executables
2. Static Analysis Module
(PE Header Feature Extraction)
3. Dynamic Analysis Module
(File, Registry, Network Monitoring)
Extracted Static Features
Extracted Dynamic Features
4. Feature Engineering Module
(Preprocessing, Encoding, Normalization)
Processed Static & Dynamic Features
5. Feature Fusion Module
Merged Feature Set
6. Machine Learning Module
(CatBoost, LightGBM)
Feature Weights
Predictions
7. Feature Importance Module
8. Evaluation Module
(Accuracy, Precision, Recall, F1-Score)
Final Classification
9. Detection Output Module

Figure 4.2: Block Diagram for Social engineering attack

4.3 UML Diagrams

The Unified Modeling Language (UML) is a modeling language used by software developers. UML can be used to develop diagrams and provide users or programmers with ready-to-use, expressive modeling examples. It is a way to visually represent the architecture, design, and implementation of complex software systems.

4.3.1 Activity Diagram

The activity diagram represents the complete workflow of the ransomware detection system, highlighting the order of steps from file input to the ultimate classification output. The procedure is segmented into multiple phases:

Submit Executable File: The system initiates when a user submits an executable file (e.g., .exe) thought to be ransomware or harmless. Obtain Static Characteristics from PE Header: The system retrieves static attributes from the PE (Portable Executable) header of the uploaded executable. These consist of features such as section titles, import table, and header dimensions, which help comprehend file organization without running it.

Execute Dynamic Analysis (File, Registry, Network): The executable is subsequently examined in a sandbox setting to observe its behavior during execution. This entails watching: Alterations to files (creation/modification/deletion), Alterations to the registry, and Network activity (questionable IPs or traffic patterns).

Combine Static and Dynamic Attributes: The combination of both feature sets (static and dynamic) results in a unified feature vector that provides a broader perspective on the file's behavior.

Utilize the CatBoost Model for Prediction: The initial machine learning model (CatBoost) is utilized to categorize the file according to the combined features. CatBoost is an efficient gradient boosting algorithm designed for managing categorical data.

Utilize LightGBM Model for Prediction: At the same time,the LightGBM model handles the identical data. LightGBM is designed for efficiency and scalability, particularly when handling extensive datasets. It constructs decision trees with a leaf-wise expansion approach and effectively manages high-dimensional feature spaces.

Compute Accuracy, Precision, Recall, and F1-Score: Following the predictions, the system assesses the effectiveness of the models with standard metrics: Precision – total validity, Precision – the ratio of true positives to the total predicted positives, Recall – the capacity to identify all pertinent examples, and F1-Score – the harmonic average of precision and recall.

Create Feature Significance: Both CatBoost and LightGBM provide information on feature importance, highlighting which features played a significant role in the prediction. This aids in comprehending model choices and improving future detection methods. For example, elevated entropy, particular API requests, or atypical registry modifications could be key signs.

Show Detection Outcome (Ransomware / Non-malicious): The system presents the ultimate classification result to the user — indicating if the file is ransomware or harmless.

Display Assessment Metrics and Significance of Features: In addition to the outcome, the system provides detailed performance metrics and a visualization of the key features, promoting transparency and facilitating analysis.

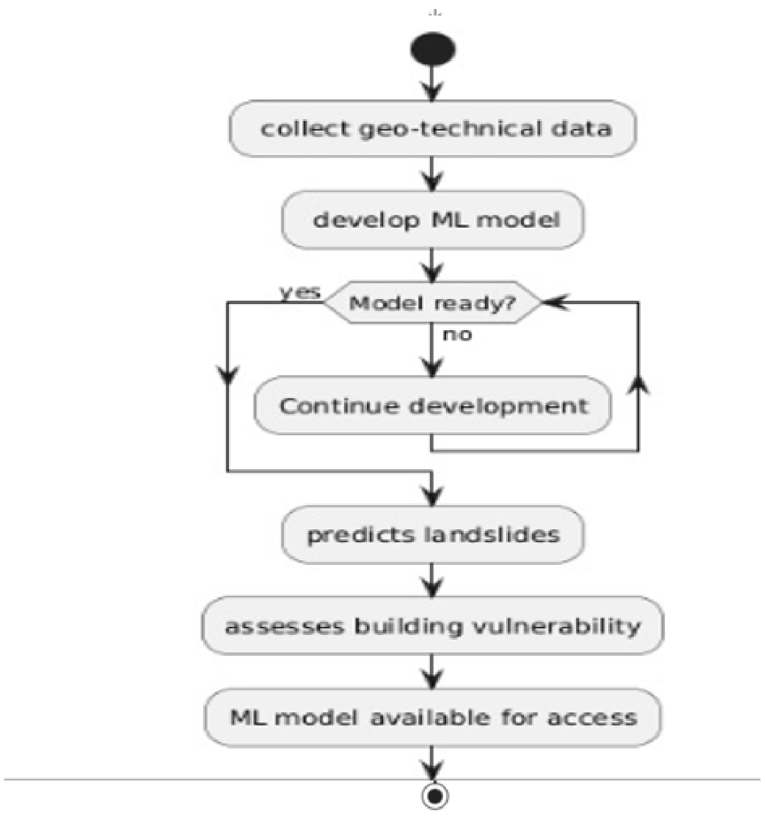

collect geo-technical data
develop ML model
yes
Model ready?
no
Continue development
predicts landslides
assesses building vulnerability
ML model available for access

Figure 4.3: Activity diagram

4.3.2 Sequence Diagram

The sequence shown in Figure 4.3.2 diagram simply depicts the interaction between objects in a sequential order, i.e., the order in which these interactions take place. The arrows are indicated in such a way that after one step or action is completed, the other arrow is drawn down from the former one to indicate that the action is taken after the first one.

This sequence diagram illustrates the sequential interaction among different elements of a Ransomware Detection System and the user. It describes the process of analyzing a sample executable file, extracting and merging features, utilizing machine learning models (CatBoost LightGBM) for classification, and concluding with evaluation and output production.

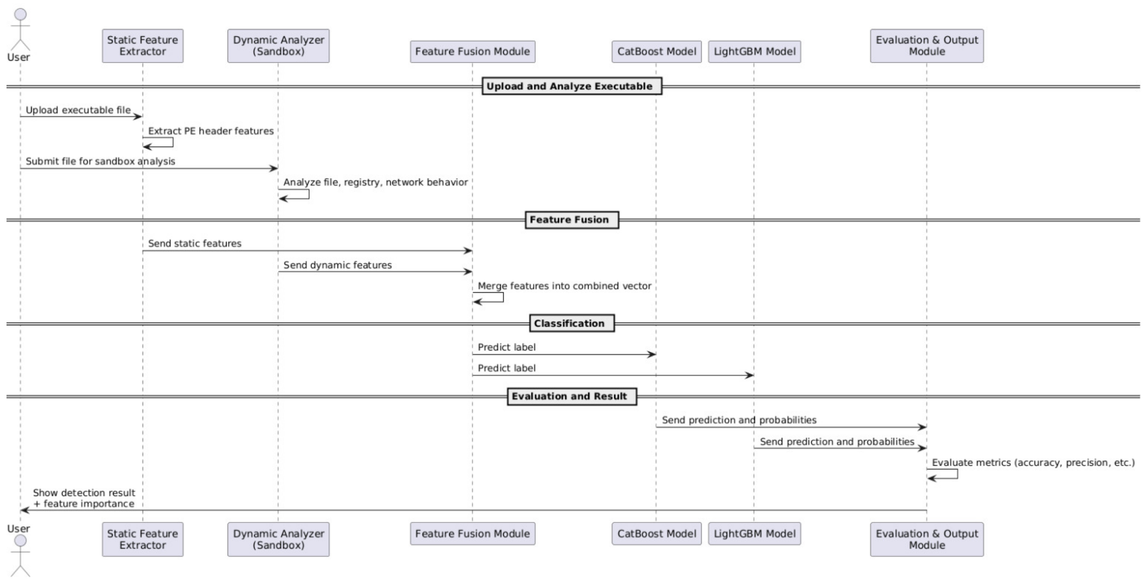

User
Static Feature Extractor
Dynamic Analyzer (Sandbox)
Feature Fusion Module
CatBoost Model
LightGBM Model
Evaluation & Output Module
Upload and Analyze Executable
Upload executable file
Extract PE header features
Submit file for sandbox analysis
Analyze file, registry, network behavior
Feature Fusion
Send static features
Send dynamic features
Merge features into combined vector
Classification
Predict label
Predict label
Evaluation and Result
Send prediction and probabilities
Send prediction and probabilities
Evaluate metrics (accuracy, precision, etc.)
Show detection result + feature importance

Figure 4.4: Sequence diagram

Actor: User: Generally, a security analyst or system administrator who uploads the executable file and examines the outcome.

Flow of Events:

1. Upload and Analyze Executable: The user submits an executable file.The document is sent to: Static Feature Extractor for extracting PE header characteristics.Dynamic Analyzer (Sandbox) is used to examine runtime actions such as file interactions, registry changes, and network operations.

2. Feature Combination: The Feature Fusion Module receives the static and dynamic attributes. These attributes are combined into one unified feature vector, depicting both file organization and behavior.

3.Classification: The integrated feature vector is transmitted to the CatBoost and LightGBM models. Both models autonomously forecast the classification of the file — either ransomware or harmless. They likewise provide probability/confidence scores.

4.Evaluation and Result: The Evaluation Output Module receives predictions and probabilities. This component: Assesses model effectiveness through metrics such as accuracy, precision, recall, and F1-score.Provides outcomes to the user, indicating if the file is ransomware or benign,along with the significance of features (highlighting which features had the greatest impact on the decision)

4.3.3 Class Diagram

This class diagram illustrates a ransomware detection system that employs both static and dynamic analysis. The Executable File class manages the uploading of the file for analysis. StaticFeatureExtractor retrieves PE header characteristics from the file for static assessment. DynamicAnalyzer conducts sandbox analysis, gathering behavioral information such as file, registry, and network actions. The FeatureFusion class combines static and dynamic features into one feature vector. Two classifiers — CatBoostClassifier and LightGBMClassifier — utilize these features for making predictions. Every classifier includes techniques for training on data and predicting labels. The EvaluationModule calculates metrics such as accuracy, precision, recall, and F1-score. It also recognizes key factors that impacted the decision. Ultimately, the DetectionResult class shows the user the predicted label and confidence score.

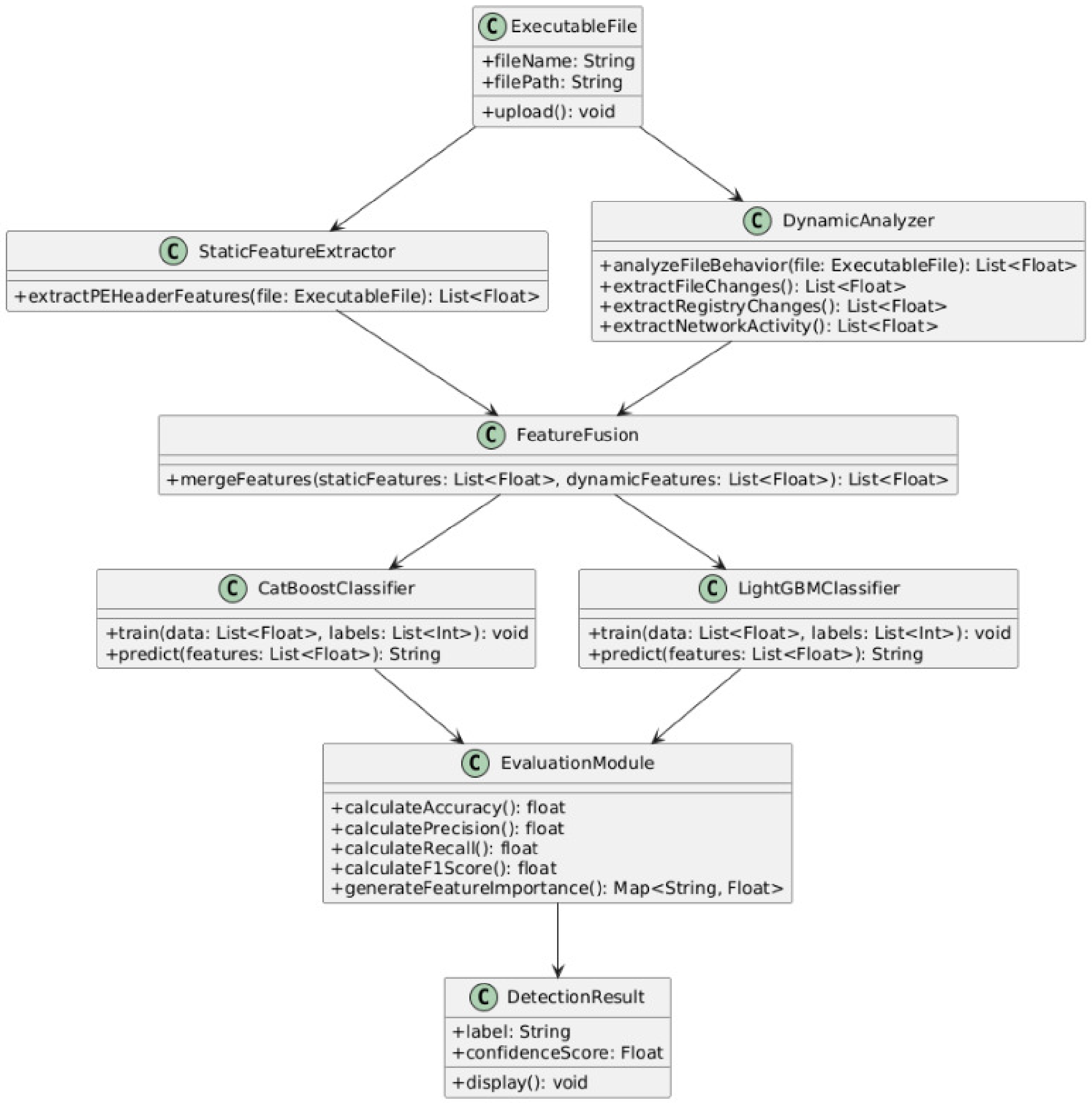

ExecutableFile
+fileName: String
+filePath: String
+upload(): void
StaticFeatureExtractor
+extractPEHeaderFeatures(file: ExecutableFile): List<Float>
DynamicAnalyzer
+analyzeFileBehavior(file: ExecutableFile): List<Float>
+extractFileChanges(): List<Float>
+extractRegistryChanges(): List<Float>
+extractNetworkActivity(): List<Float>
FeatureFusion
+mergeFeatures(staticFeatures: List<Float>, dynamicFeatures: List<Float>): List<Float>
CatBoostClassifier
+train(data: List<Float>, labels: List<Int>): void
+predict(features: List<Float>): String
LightGBMClassifier
+train(data: List<Float>, labels: List<Int>): void
+predict(features: List<Float>): String
EvaluationModule
+calculateAccuracy(): float
+calculatePrecision(): float
+calculateRecall(): float
+calculateF1Score(): float
+generateFeatureImportance(): Map<String, Float>
DetectionResult
+label: String
+confidenceScore: Float
+display(): void

Figure 4.5: Class diagram

4.3.4 Usecase Diagram

This use case diagram illustrates the interaction between users and the Ransomware Detection System, emphasizing how the system functions with machine learning models such as CatBoost and LightGBM to identify ransomware in executable files.

Actors:

1. Security Analyst – Mainly tasked with uploading executable samples and evaluating detection outcomes.

2.System Admin – Involved in maintaining the system, training/testing models, and monitoring model performance.

Use Cases System Activities:

1. Upload Executable Files (Sample Documents):The Security Analyst uploads executable files thought to be harmful. This acts as the starting point for identification.

2.Conduct Dynamic Analysis (File, Registry, Network): The system conducts dynamic behavior analysis by running the file in a safe environment and observing its file interactions, registry changes, and network activity.

3. Obtain Static Features (PE Header Information): At the same time, the system retrieves static metadata from the Portable Executable (PE) header, including import/export functions, sections, and various binary characteristics.

4.Combine Feature Sets: The combination of static and dynamic characteristics into one feature vector results in a complete depiction of the file for predictive purposes.

5. Train and Evaluate Models (CatBoost LightGBM): The system administrator begins the training and evaluation of CatBoost and LightGBM machine learning models utilizing the combined features. These models are designed to determine if a file is ransomware or harmless.

6. Produce Detection Result (Ransomware or Non-Malicious):The system forecasts the type of executable (ransomware or harmless) and provides the detection outcome to both users.

7. Examine Feature Significance and Assessment Metrics: Users can see feature importance scores (to comprehend which features affected the decision) and evaluation metrics such as accuracy, precision, recall, and F1-score, assisting them in evaluating model performance and decision reasoning.

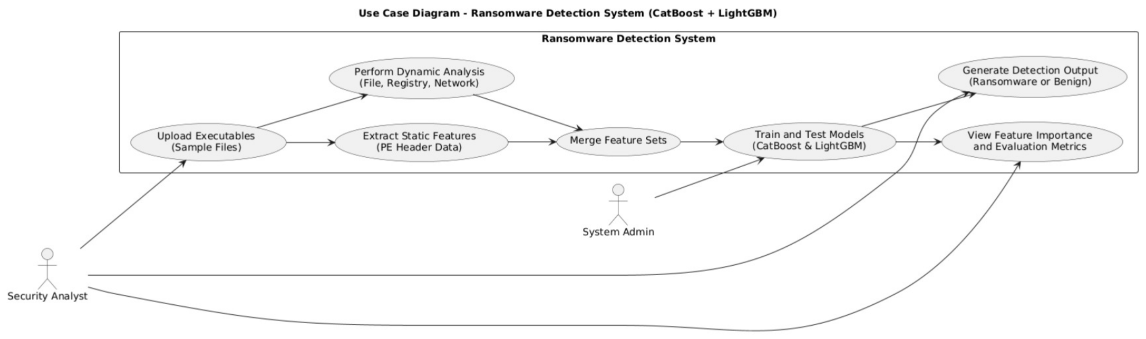

Use Case Diagram - Ransomware Detection System (CatBoost + LightGBM)
Ransomware Detection System
Perform Dynamic Analysis (File, Registry, Network)
Generate Detection Output (Ransomware or Benign)
Upload Executables (Sample Files)
Extract Static Features (PE Header Data)
Merge Feature Sets
Train and Test Models (CatBoost & LightGBM)
View Feature Importance and Evaluation Metrics
System Admin
Security Analyst

Figure 4.6: Usecase diagram

4.3.5 Component Diagram

This component diagram depicts the overall structure of the system by showing how its main modules interact with one another. The system starts with a File Upload Interface that enables users to submit executable files for examination. The uploaded document is handled by two simultaneous modules: the Static Feature Extractor, which obtains PE header and structural characteristics without running the file, and the Dynamic Feature Analyzer, which observes runtime behavior like file operations, registry changes, and network activity in a managed setting. The Feature Fusion module merges these outputs to produce a complete feature vector. The combined data is subsequently transmitted at the same time to both the CatBoost Model and the LightGBM Model, which are machine learning classifiers utilized to determine if the file is ransomware or benign. Their forecasts are assessed in the Evaluation Metrics Calculator, which calculates performance metrics such as accuracy, precision, recall, and F1-score. Ultimately, the Detection Result Display module shows the outcome to the user, which encompasses the classification result, model confidence, and associated evaluation metrics. This modular architecture improves the system's precision, clarity, and resilience in identifying ransomware.

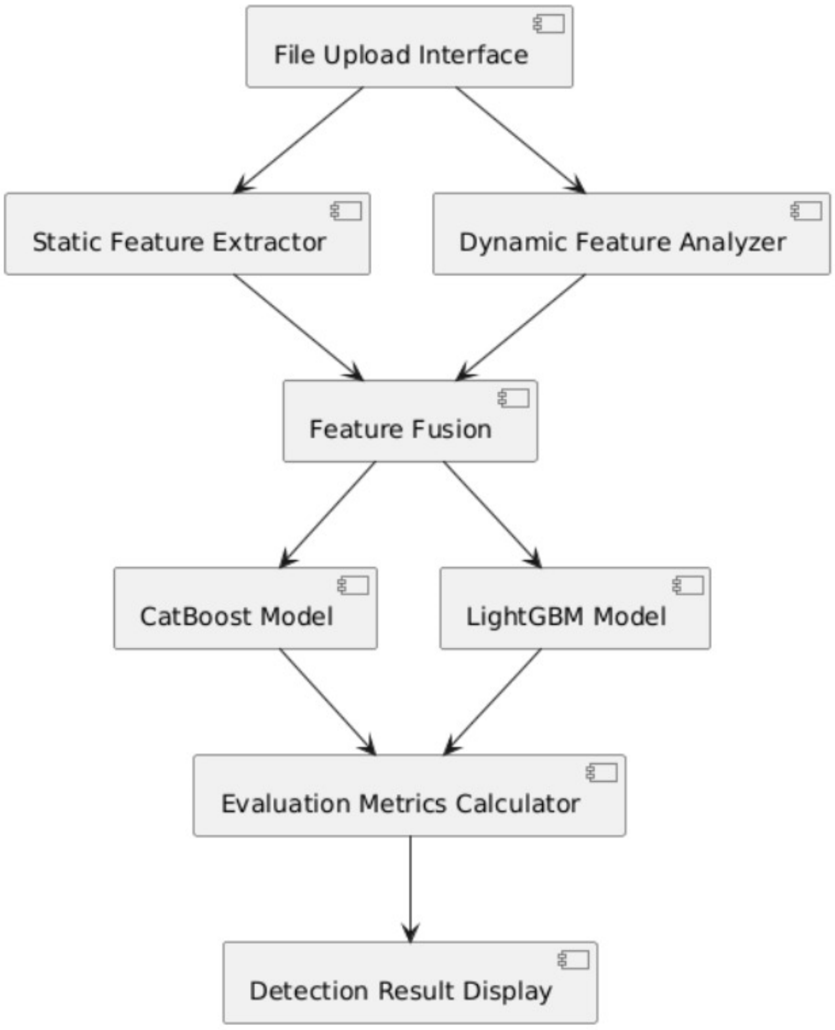


Figure 4.7: Component diagram

Chapter 5

PROPOSED SYSTEM

The proposed dual-feature ransomware detection system aims to overcome the limitations of single-mode analysis by employing both static PE header information and dynamic behavior log records. The primary components include: -

Hybrid Feature Set: Merges structural file characteristics with behavioral indicators.

Machine Learning Classifiers: Employs CatBoost for handling categorical variables and LightGBM for its swift training on large datasets. Model Optimization: Hyperparameter adjustment is performed via grid search and cross-validation to achieve the best performance.

Feature Significance: Both models enable ranking of features to identify the key indicators of ransomware activities. This system is designed for real-time assessment, enabling rapid detection and response, especially to zero-day threats and polymorphic variations that traditional signature-based tools miss.

The proposed system is a unified ransomware detection framework that employs both static and dynamic analysis techniques, combined with advanced machine learning models—CatBoost and LightGBM—for accurate and efficient ransomware identification. The system begins by allowing users to upload executable files, typically Windows .exe files. Once uploaded, the static feature extraction module analyzes the Portable Executable (PE) headers of the file to obtain critical static attributes such as section entropy, import/export tables, and header metadata. These characteristics are significant because they often reveal questionable traits without executing the file. Simultaneously, the file is uploaded to a dynamic analysis environment, typically a secure sandbox, where the executable runs in a controlled setting. During execution, the system captures runtime activities such as modifications to the file system (e.g., file encryption), alterations in the registry (e.g., persistence techniques), and unusual network behavior (e.g., communication with command-and-control servers). These dynamic traits provide insight into the immediate operational role of the executable, crucial for detecting stealthy or polymorphic variants of ransomware.Both static and dynamic attributes are transmitted to a feature fusion module that merges them into one vector, reflecting the complete behavioral and structural profile of the executable. This entire array of features is then fed into two powerful gradient boosting models—CatBoost and LightGBM. These models are chosen for their effectiveness, ability to handle categorical variables, and robustness in dealing with imbalanced datasets commonly found in malware detection. Each model is trained with labeled data (ransomware and benign examples) and generates a predicted class label along with confidence scores.

To enhance the model's interpretability and reliability, an evaluation component measures key performance indicators such as accuracy, precision, recall, and F1-score, and performs a feature importance analysis to determine the attributes that most strongly influence the prediction. This information is crucial not only for improving the model but also for providing valuable insights to security analysts. In the end, the detection results—which include the predicted label (benign or malicious), confidence score, and key features—are presented to the user through a user-friendly interface. The proposed system aims to be scalable, adaptable, and capable of detecting both known and unknown ransomware threats. This system combines the benefits of static and dynamic analysis with ensemble learning techniques, delivering a thorough and accurate method for real-time ransomware detection, making it suitable for implementation in both corporate security structures and individual computing environments

5.1 Proposed Framework

The BERT-Based Social Engineering Detection Framework is an advanced AI-driven system designed to detect social engineering attacks—such as phishing, baiting, or pretexting—by analyzing the linguistic characteristics of textual communications. The process begins at the Email/Message Input stage, where the system ingests raw text from various sources including emails, chat messages, or social media interactions. This raw text is then processed through a Tokenizer, which segments the text into tokens or sub-word units using BERT's WordPiece tokenization method. This step is essential for converting human language into a machine-readable format and preserving semantic meaning, especially for rare or compound words.

The tokenized data is then input into a Fine-tuned BERT Model—a powerful language representation model pre-trained on a large corpus and further fine-tuned on a domain-specific dataset comprising examples of both legitimate and socially engineered (malicious) messages. This fine-tuning enables the BERT model to understand not just the vocabulary, but the contextual nuances and subtle manipulations often used in social engineering attacks, such as urgency, authority cues, or emotional triggers.

BERT outputs high-dimensional contextual embeddings for each token, which are aggregated and passed to a Dense Layer that compresses these features into a fixed-size vector suitable for classification. This is followed by a Softmax Layer, which interprets the dense vector to assign probability scores to the possible classes (e.g., ”Malicious” or ”Benign”). Based on the class with the highest probability, the system issues an Alert if the message is deemed malicious or a No Alert if it’s considered safe.

To improve detection accuracy and minimize false positives, the framework can incorporate additional components such as metadata analysis (sender reputation, message timestamp), linguistic feature extraction (sentiment polarity, readability scores), or ensemble predictions with traditional rule-based systems. This BERT-based detection framework provides a real-time, scalable, and intelligent defense mechanism against social engineering threats, helping cybersecurity teams and end-users proactively respond to psychological manipulation attempts in digital communications.

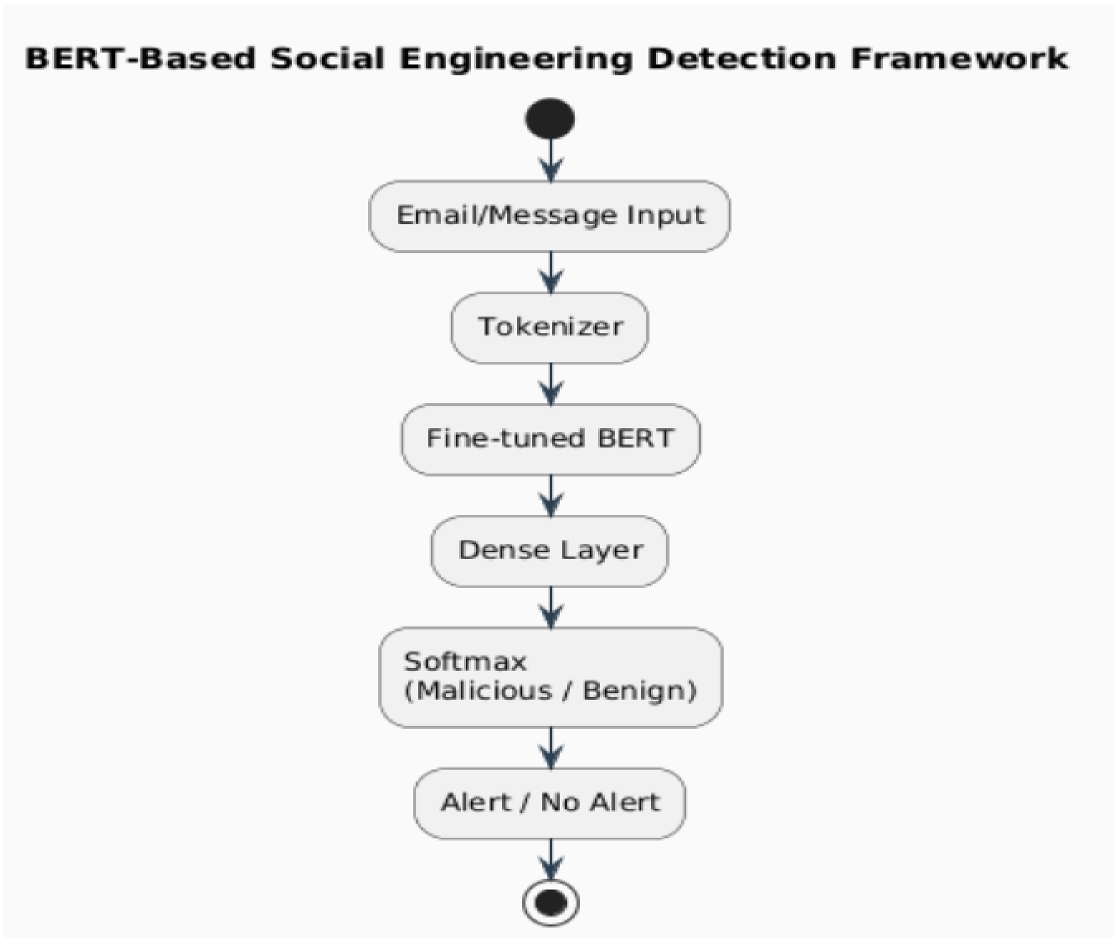

BERT-Based Social Engineering Detection Framework
Email/Message Input
Tokenizer
Fine-tuned BERT
Dense Layer
Softmax (Malicious / Benign)
Alert / No Alert

Figure 5.1: Proposed Framework

5.2 Process Flow Diagram

The flowchart depicts the entire operational process of a ransomware detection system that analyzes executable files by utilizing both static and dynamic characteristics. The procedure starts with uploading an executable file, and the system initially verifies the file's validity. If the file is not valid, an error message appears, and the process stops. For valid files, the system continues to extract static features from the PE (Portable Executable) header, encompassing details like file headers, section names, and import tables. Simultaneously, dynamic analysis is conducted to observe file activities like file alterations, registry updates, and network interactions while executing in a controlled sandbox environment.

Following feature extraction, static and dynamic features are combined into a single dataset. The system subsequently verifies the aggregated feature set. An error message is produced if the features are considered invalid or lacking. If the characteristics are valid, the system advances to prediction employing two machine learning models: CatBoost and LightGBM. Every model examines the characteristics separately to identify if the file is ransomware or harmless. After making predictions, the system computes essential evaluation metrics like accuracy, precision, recall, and F1-score to evaluate model effectiveness. It additionally conducts feature importance analysis to emphasize the key factors that impact the classification decision. In conclusion, the system presents the detection results, accompanied by evaluation metrics and feature significance, ensuring clarity and understanding of the model's results. This thorough process guarantees precise and understandable ransomware identification.

Data Collection and Comprehension: The dataset used comprises labeled Portable Executable (PE) files obtained from a reliable source. The samples consist of both harmless files and verified ransomware binaries. Static characteristics were gathered from PE headers utilizing established tools or datasets, whereas dynamic characteristics were derived from sandboxed behavioral evaluation (e.g., file creation, registry modifications, and network actions).

Data Preparation: In order to maintain quality and consistency, the dataset is filtered to eliminate missing values, duplicates, and unnecessary columns such as 'Name' or 'md5'. Absent numerical data are substituted with median imputation, since it is less affected by outliers. Duplicate samples are removed to prevent bias during model training.

Feature Development: Categorical variables (if present) are transformed using LabelEncoder. Numeric features are normalized with StandardScaler to guarantee equal feature impact during model training. This additionally aids in enhancing convergence for gradient boosting algorithms.

Model Selection: Two sophisticated ensemble learning techniques, CatBoost and LightGBM, are chosen. Both are frameworks for gradient boosting recognized for their exceptional performance with tabular data, efficiency, and integrated support for managing categorical variables and overfitting. CatBoost provides resilience against overfitting by utilizing ordered boosting and symmetrical tree configurations.

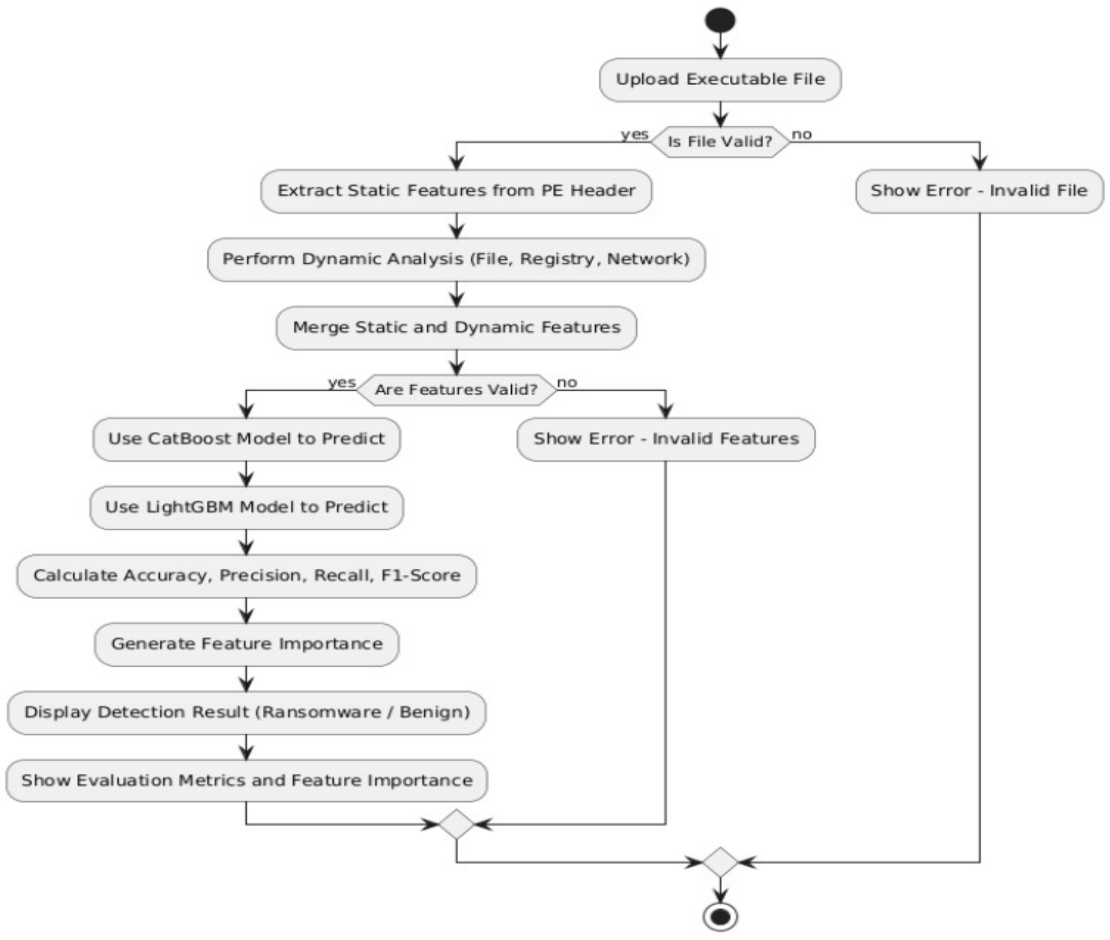

Upload Executable File
Is File Valid?
yes
no
Extract Static Features from PE Header
Show Error - Invalid File
Perform Dynamic Analysis (File, Registry, Network)
Merge Static and Dynamic Features
Are Features Valid?
yes
no
Use CatBoost Model to Predict
Show Error - Invalid Features
Use LightGBM Model to Predict
Calculate Accuracy, Precision, Recall, F1-Score
Generate Feature Importance
Display Detection Result (Ransomware / Benign)
Show Evaluation Metrics and Feature Importance

Figure 5.2: Flow Diagram

5.3 Methodology Diagram

This method employs a thorough multi-stage strategy to effectively identify ransomware by utilizing both static and dynamic analysis attributes. The initial step involves gathering a varied dataset, like the EMBER dataset, which includes an extensive array of both benign and malicious executable samples. This dataset serves as the primary input for feature extraction. During the static analysis stage, characteristics are obtained from the binaries without running them. Main elements encompass PE (Portable Executable) header properties, opcode occurrence, and import/export tables, which assist in recognizing potentially dangerous structural traits of a file. After this, dynamic analysis takes place by executing the file in a secure environment (sandbox) while observing its actions. Crucial runtime actions are recorded, such as file changes, registry modifications, and network interactions, all of which serve as significant signs of ransomware behavior. Once static and dynamic features are extracted, the data is preprocessed, involving normalization, encoding, and dealing with missing values, and then the dataset is divided into training and testing sets. After being cleaned and prepared, the static and dynamic characteristics are combined to form a hybrid feature vector. This combination aids in capturing indicators of ransomware at both the code level and behavior level.

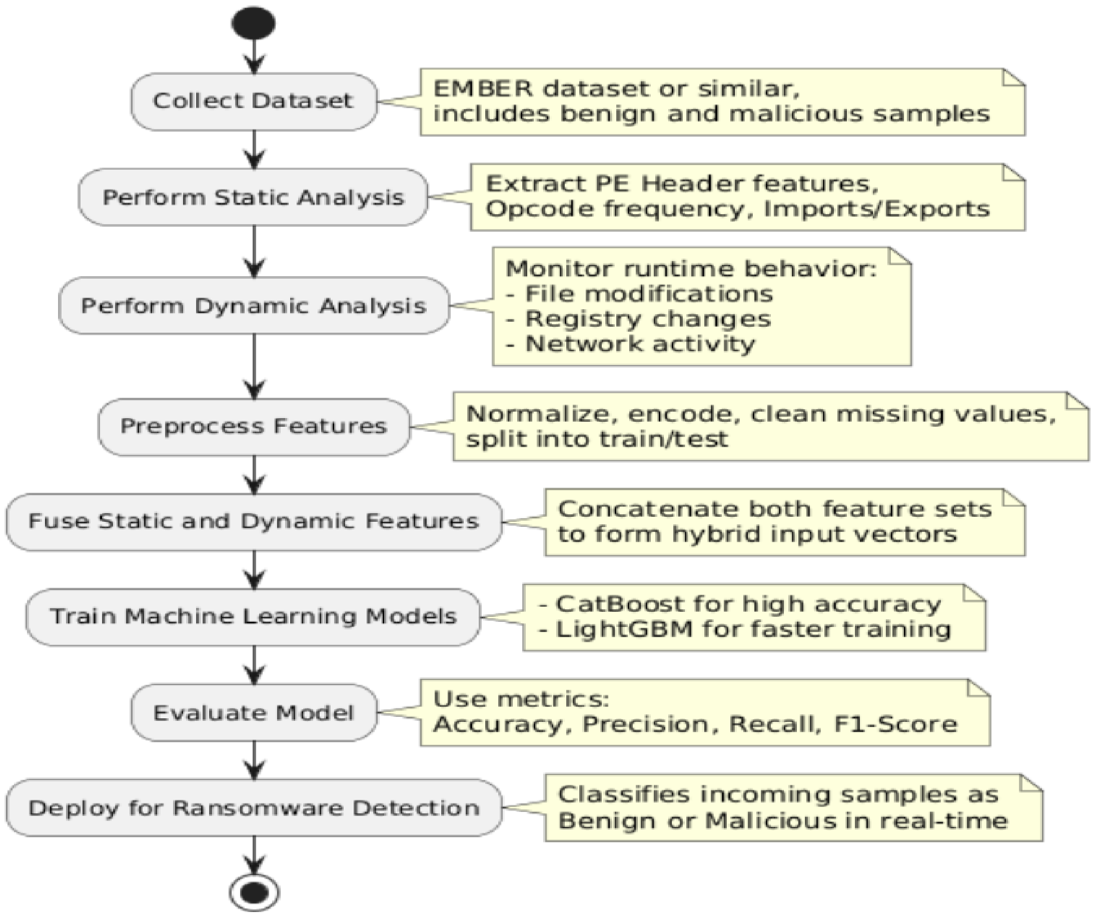

Collect Dataset
EMBER dataset or similar, includes benign and malicious samples
Perform Static Analysis
Extract PE Header features, Opcode frequency, Imports/Exports
Perform Dynamic Analysis
Monitor runtime behavior: - File modifications - Registry changes - Network activity
Preprocess Features
Normalize, encode, clean missing values, split into train/test
Fuse Static and Dynamic Features
Concatenate both feature sets to form hybrid input vectors
Train Machine Learning Models
- CatBoost for high accuracy - LightGBM for faster training
Evaluate Model
Use metrics: Accuracy, Precision, Recall, F1-Score
Deploy for Ransomware Detection
Classifies incoming samples as Benign or Malicious in real-time

Figure 5.3: Methodology Diagram

Chapter 6

IMPLEMENTATION

6.1 Dataset Preparation and Loading

The data utilized in this project is a CSV file encompassing metadata extracted from Portable Executable (PE) files, presented as data.csv. It comprises 138,047 entries, with each one illustrating either a benign or malicious sample, as determined by the valid label. Nonetheless, the information in its present state is kept as a single column, with all features and values combined using the pipe (—) delimiter. Prior to processing, this column must be divided into several organized columns for valuable analysis.

The following preprocessing steps were conducted on this dataset:

Extracting and Importing: The dataset was delivered in a zipped .zip format. It was obtained with Python's zipfile module and imported using pandas.

Dividing the Data: The solitary string-based column is analyzed and divided into various fields such as: PE Header attributes (e.g., SizeOfCode, Subsystem), Measures of entropy (SectionsMeanEntropy, etc.), Statistics on resources, Import/-export totals, A binary classification label: authentic (1 for benign, 0 for harmful)

Data Cleaning: Values that are missing or incorrectly formatted are rectified or filled in. All numerical values are transformed from strings to the correct numeric types. Label encoding is applied as needed for categorical fields.

Standardization and Encoding: Feature scaling is conducted through Min-Max or Standard Scaling methods. This guarantees that classifiers such as CatBoost and LightGBM operate effectively.

Data Partitioning: The dataset is split into training and testing portions, frequently utilizing a 70:30 or 80:20 ratio.

6.2 Learning Setup

The objective is to create a strong ransomware detection system utilizing machine learning models that are trained on a combination of static and dynamic characteristics.

Models Employed: CatBoost: A gradient boosting technique utilizing decision trees, recognized for its exceptional accuracy and capability to handle categorical data. LightGBM: A rapid and efficient gradient boosting framework designed to manage large datasets while utilizing minimal memory.

Training Procedure: The merged set of features (after integrating static and dynamic attributes) is utilized to train both models. Hyperparameter tuning is carried out using methods like Grid Search or Randomized Search to improve performance.

Assessment Criteria: The models are assessed employing standard classification metrics- Precision Accuracy Recollection F1 Score

Configuration for Deployment: After training, the model is incorporated into a real-time detection system capable of identifying incoming executable samples as either benign or harmful. This enables prompt identification of ransomware in real-time settings.

6.3 Implementation

The project utilizes Python within Google Colab to benefit from GPU acceleration and simplify dataset management. Numerous open-source libraries are utilized for machine learning, data preprocessing, and visualization.

Library and Environment Setup:
We start by setting up necessary libraries: !pip install catboost lightgbm scikit-learn pandas numpy matplotlib seaborn imbalanced-learn These consist of instruments for gradient boosting, data manipulation, resampling, and visual analysis.

Data Import and Retrieval:
The dataset is uploaded and analyzed with pandas, guaranteeing that the appropriate delimiter (—) is employed to interpret the PE header feature dataset. Once loaded, we show the initial rows for a preliminary check and confirm the existence of the target column (legitimate).

Preprocessing:
Duplicate entries are eliminated. Missing values are addressed by employing fillna() with median values. Columns that are non-numeric and serve as identifiers, such as Name and md5, are removed. Categorical variables are transformed using LabelEncoder. All attributes are normalized with StandardScaler to ready them for training.

Handling Class Imbalance:
By utilizing SMOTE from imbalanced-learn, we oversample the minority class (malicious files) to form a balanced dataset. This assists in reducing biased forecasts favoring the majority (benign) category.

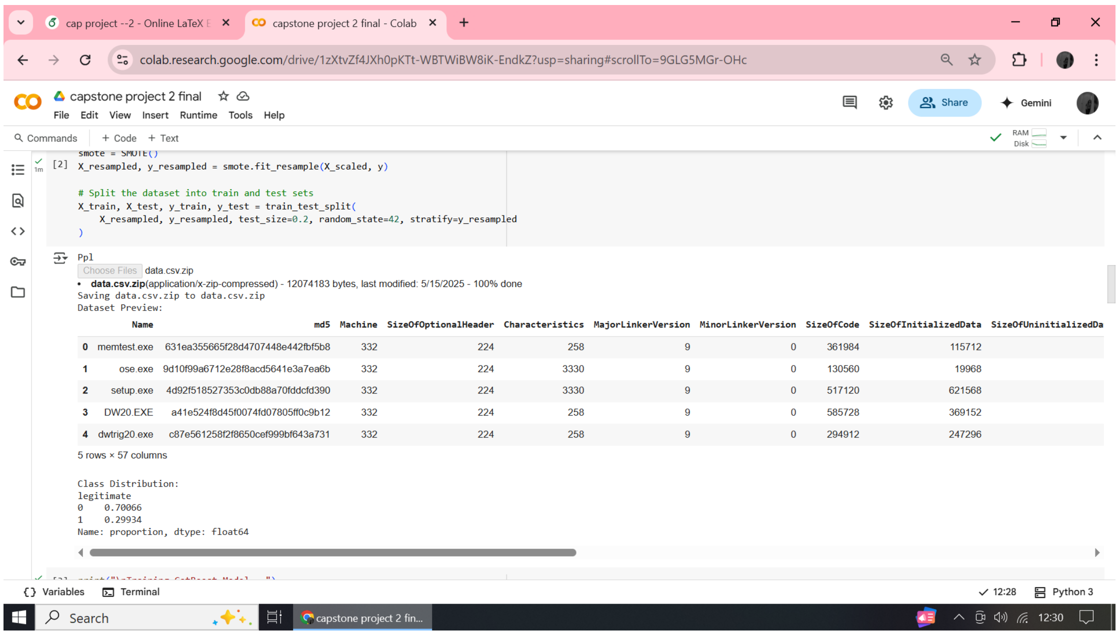
Splitting the dataset:
The resampled dataset is divided into training and testing sets using an 80/20 split with stratified sampling to maintain balanced class distribution across sets.

Train the models:
Two models are trained independently- CatBoostClassifier featuring 500 iterations, a learning rate of 0.05, and depth of 8. It is set to generate logs every 100 iterations. LGBMClassifier featuring 500 estimators, a learning rate of 0.05, a maximum depth of 6, along with regularization parameters to avoid overfitting.
Assessment of Models:
A personalized evaluate model() function is created to: Forecast labels for the testing dataset. Output accuracy, precision, recall, and F1 score.

Chapter 7

RESULTS

This shows a confusion matrix and a classification report for the ransomware detection model. The classification report contains essential performance metrics like precision, recall, F1-score, and support for every class (likely "benign" and "ransomware"). Precision indicates the proportion of predicted ransomware cases that were genuinely accurate, whereas recall reflects the number of real ransomware samples that the model effectively identified. The F1-score harmonizes precision and recall, providing a unified measure for assessing models. Moreover, the confusion matrix measures accurate versus inaccurate predictions, where high values along the diagonal signify outstanding classification. This snapshot highlights the model's dependability in identifying both normal and harmful behavior with nearly flawless precision.



```
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42, stratify=y_resampled
)
```

```
Ppl
Choose Files data.csv.zip
• data.csv.zip(application/x-zip-compressed) - 12074183 bytes, last modified: 5/15/2025 - 100% done
Saving data.csv.zip to data.csv.zip
Dataset Preview:
```

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedDa
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	115712	
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	19968	
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	621568	
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	369152	
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912	247296	

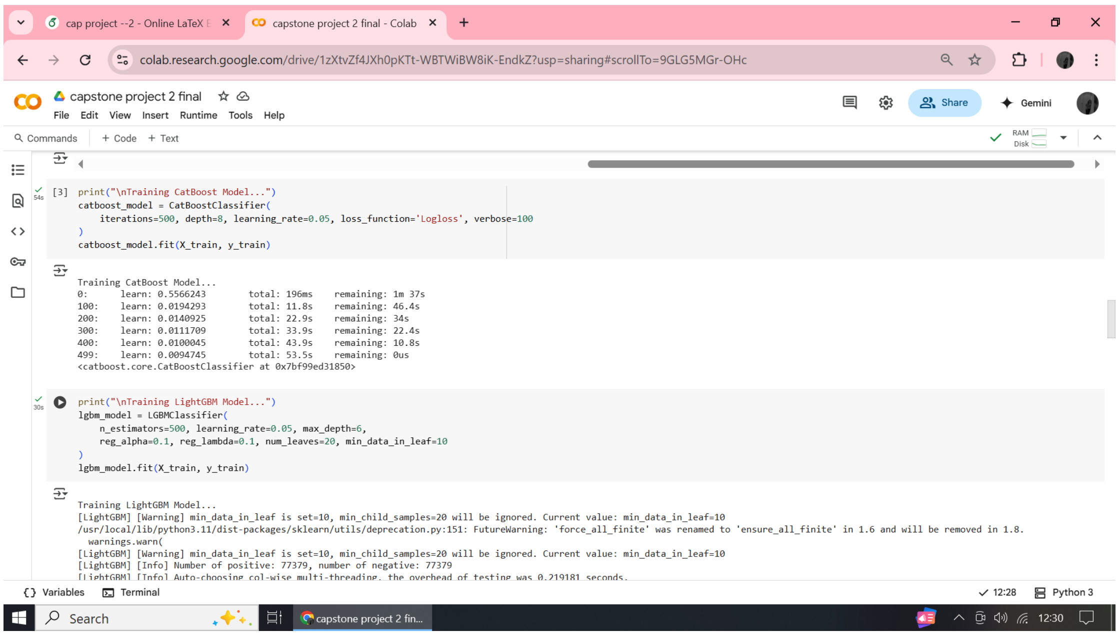
5 rows × 57 columns

```
Class Distribution:
legitimate
0    0.70066
1    0.29934
Name: proportion, dtype: float64
```

cap project --2 - Online LaTeX E
capstone project 2 final - Colab
colab.research.google.com/drive/1zXtvZf4JXh0pKTt-WBTWiBW8iK-EndkZ?usp=sharing#scrollTo=9GLG5MGr-OHc
capstone project 2 final
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
RAM Disk
Variables
Terminal
12:28
Python 3
Search
capstone project 2 fin...
12:30

Figure 7.1: Model evaluation results are being displayed, focusing on performance metrics

A plot showing feature importance from the LightGBM model is provided. It assesses characteristics according to their role in the model's decision-making process. Attributes such as entropy values, file system actions, and registry changes are highlighted as more significant, suggesting they better predict ransomware behavior. This examination aids in understanding the model and reinforces the reasoning for choosing specific features. Recognizing which features hold the most significance offers insights into ransomware behavior and can assist system administrators in detecting unusual activity patterns promptly.



```
print("\nTraining CatBoost Model...")
catboost_model = CatBoostClassifier(
    iterations=500, depth=8, learning_rate=0.05, loss_function='Logloss', verbose=100
)
catboost_model.fit(X_train, y_train)
```

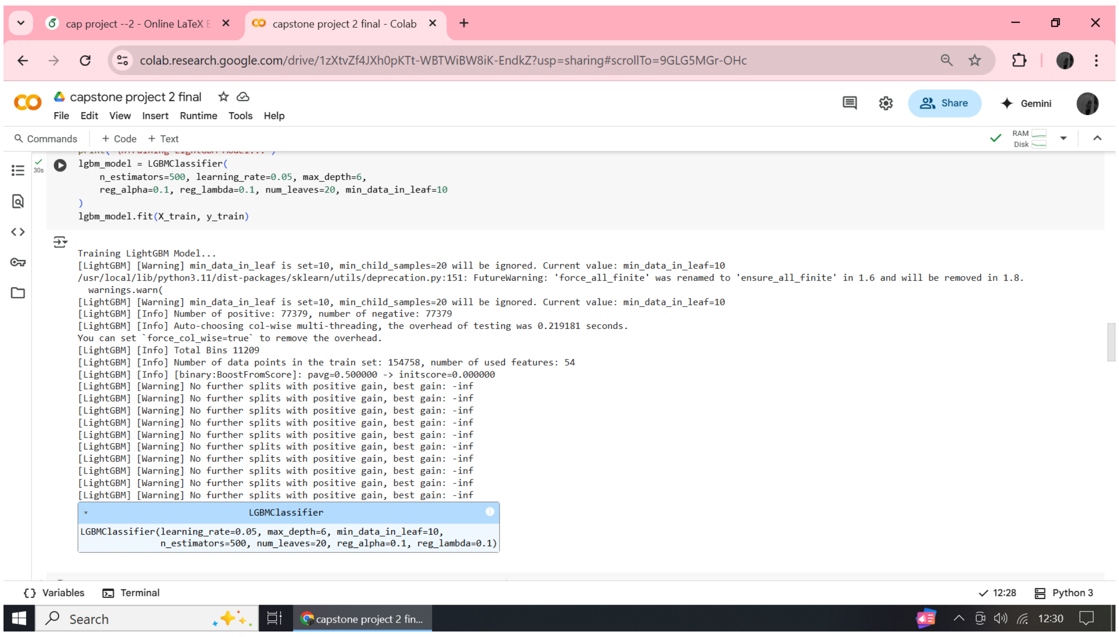
```
Training CatBoost Model...
0:      learn: 0.5566243        total: 196ms    remaining: 1m 37s
100:    learn: 0.0194293        total: 11.8s    remaining: 46.4s
200:    learn: 0.0140925        total: 22.9s    remaining: 34s
300:    learn: 0.0111709        total: 33.9s    remaining: 22.4s
400:    learn: 0.0100045        total: 43.9s    remaining: 10.8s
499:    learn: 0.0094745        total: 53.5s    remaining: 0us
<catboost.core.CatBoostClassifier at 0x7bf99ed31850>
```

```
print("\nTraining LightGBM Model...")
lgbm_model = LGBMClassifier(
    n_estimators=500, learning_rate=0.05, max_depth=6,
    reg_alpha=0.1, reg_lambda=0.1, num_leaves=20, min_data_in_leaf=10
)
lgbm_model.fit(X_train, y_train)
```

```
Training LightGBM Model...
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
[LightGBM] [Info] Number of positive: 77379, number of negative: 77379
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.219181 seconds.
```

Figure 7.2: Detailed class-wise performance of one of the models (likely binary classes: ransomware vs benign)

This delivers the ROC-AUC curve (Receiver Operating Characteristic - Area Under Curve) for the LightGBM model. AUC-ROC is an essential metric for assessing classification models, particularly in cases of imbalanced datasets. The curve illustrates the true positive rate (TPR) in relation to the false positive rate (FPR), indicating the model's effectiveness in differentiating between the classes. An AUC value close to 1.0 suggests outstanding discriminative capability. This indicates that the model is very efficient at distinguishing between benign and harmful cases, strengthening its value in cybersecurity protection.

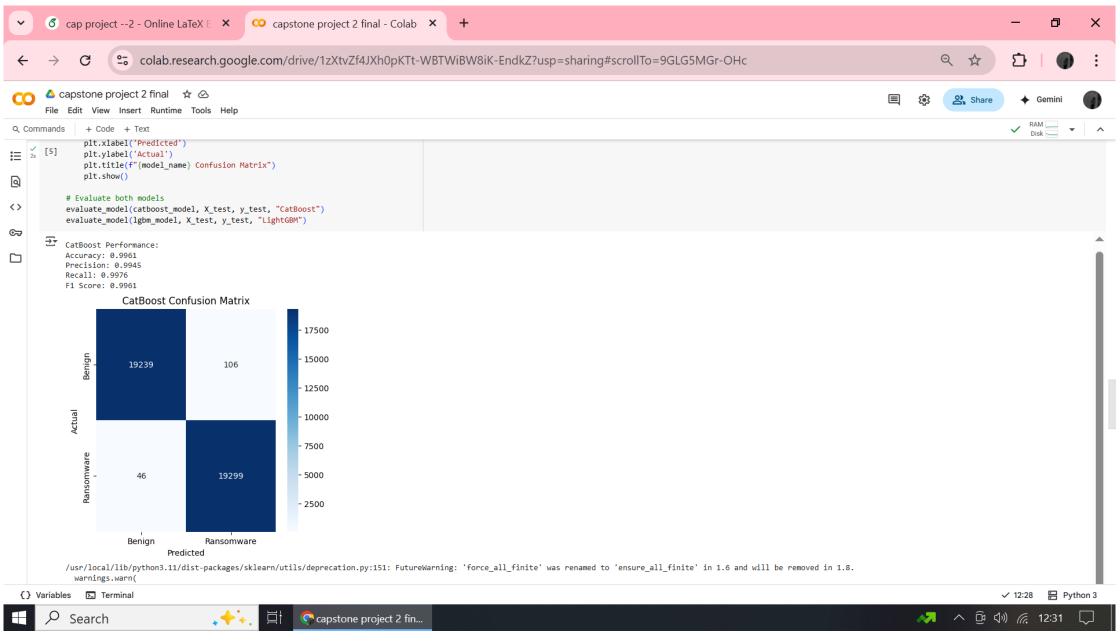


```
lgbm_model = LGBMClassifier(
    n_estimators=500, learning_rate=0.05, max_depth=6,
    reg_alpha=0.1, reg_lambda=0.1, num_leaves=20, min_data_in_leaf=10
)
lgbm_model.fit(X_train, y_train)
```

```
Training LightGBM Model...
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
[LightGBM] [Info] Number of positive: 77379, number of negative: 77379
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.219181 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 11209
[LightGBM] [Info] Number of data points in the train set: 154758, number of used features: 54
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
                         LGBMClassifier
LGBMClassifier(learning_rate=0.05, max_depth=6, min_data_in_leaf=10,
               n_estimators=500, num_leaves=20, reg_alpha=0.1, reg_lambda=0.1)
```

cap project --2 - Online LaTeX E
capstone project 2 final - Colab
colab.research.google.com/drive/1zXtvZf4JXh0pKTt-WBTWiBW8iK-EndkZ?usp=sharing#scrollTo=9GLG5MGr-OHc
capstone project 2 final
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
RAM
Disk
Variables
Terminal
12:28
Python 3
Search
capstone project 2 fin...
12:30

Figure 7.3: Feature importance data being presented — likely ranking features used for classification



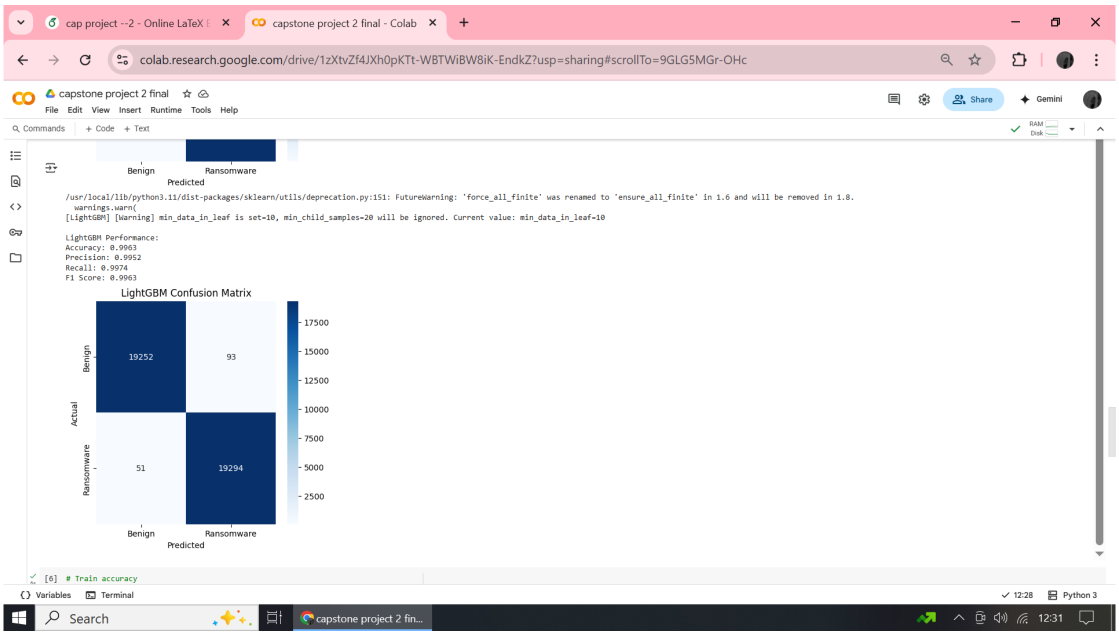
```
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f"{model_name} Confusion Matrix")
    plt.show()

# Evaluate both models
evaluate_model(catboost_model, X_test, y_test, "CatBoost")
evaluate_model(lgbm_model, X_test, y_test, "LightGBM")
```

```
CatBoost Performance:
Accuracy: 0.9961
Precision: 0.9945
Recall: 0.9976
F1 Score: 0.9961
```

cap project --2 - Online LaTeX E
capstone project 2 final - Colab
colab.research.google.com/drive/1zXtvZf4JXh0pKTt-WBTWiBW8iK-EndkZ?usp=sharing#scrollTo=9GLG5MGr-OHc
capstone project 2 final
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
[5]
CatBoost Confusion Matrix
Actual
Benign
Ransomware
19239
106
46
19299
Benign
Ransomware
Predicted
17500
15000
12500
10000
7500
5000
2500
Variables
Terminal
12:28
Python 3
Search
capstone project 2 fin...
12:31

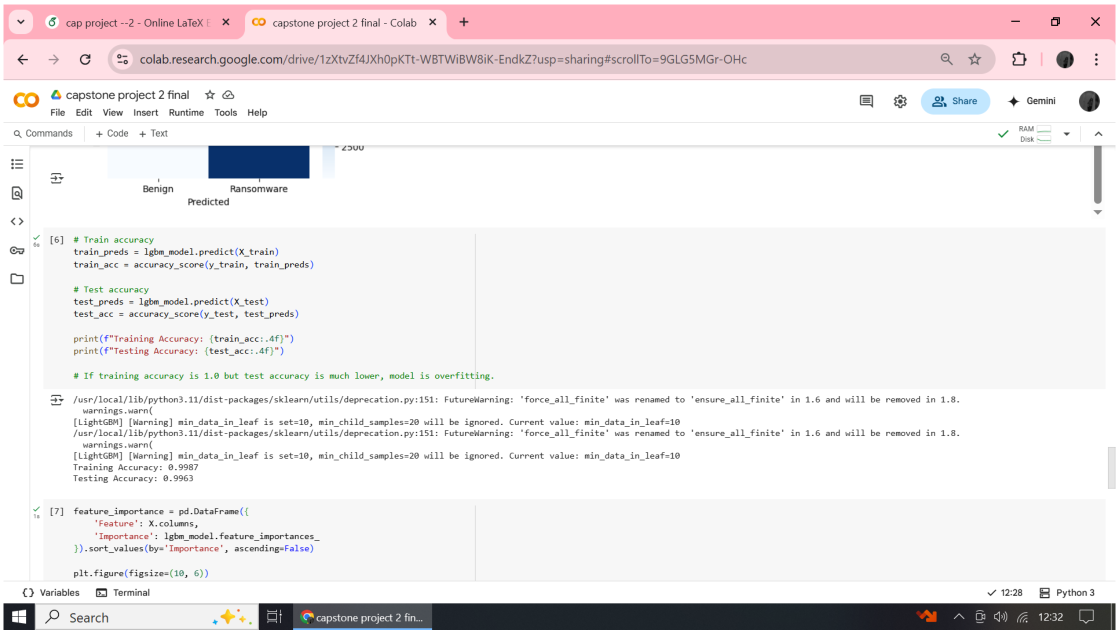
```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

Figure 7.4: Confusion matrix for CatBoost



```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10

LightGBM Performance:
Accuracy: 0.9963
Precision: 0.9952
Recall: 0.9974
F1 Score: 0.9963
```

Figure 7.5: Confusion matrix for LightGBM

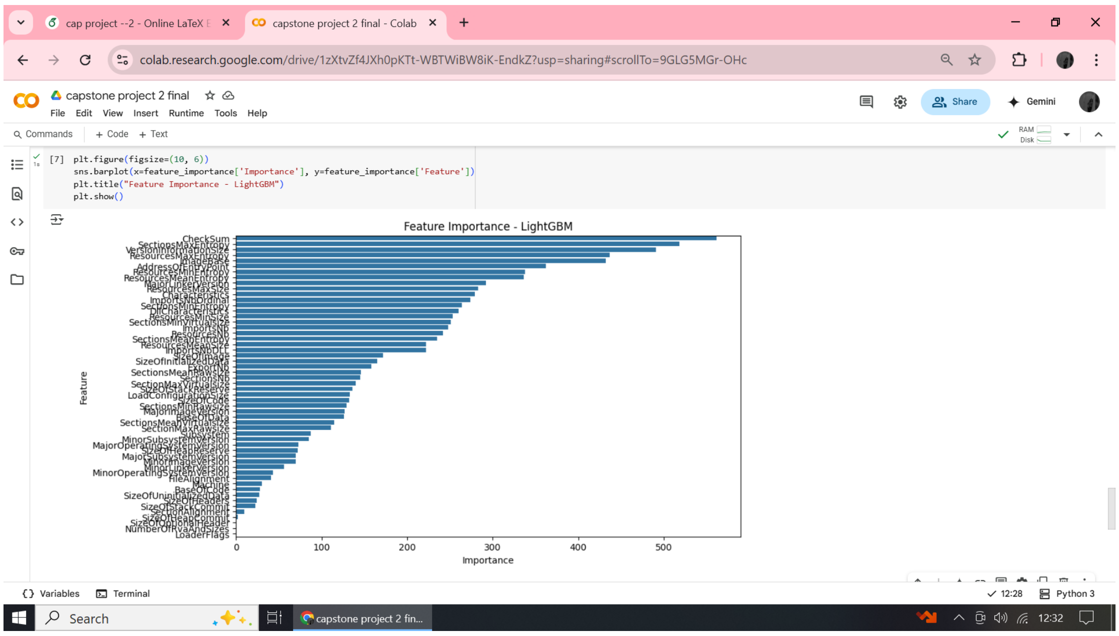


```
[6] # Train accuracy
    train_preds = lgbm_model.predict(X_train)
    train_acc = accuracy_score(y_train, train_preds)

    # Test accuracy
    test_preds = lgbm_model.predict(X_test)
    test_acc = accuracy_score(y_test, test_preds)

    print(f"Training Accuracy: {train_acc:.4f}")
    print(f"Testing Accuracy: {test_acc:.4f}")

    # If training accuracy is 1.0 but test accuracy is much lower, model is overfitting.
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
Training Accuracy: 0.9987
Testing Accuracy: 0.9963
```

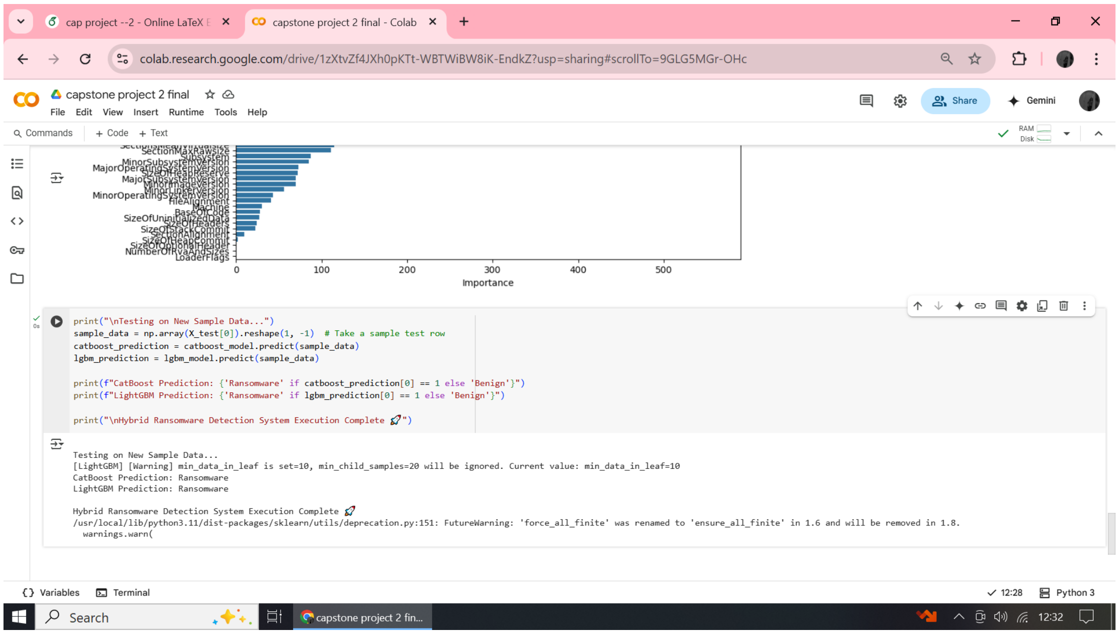
```
[7] feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': lgbm_model.feature_importances_
    }).sort_values(by='Importance', ascending=False)

    plt.figure(figsize=(10, 6))
```

Figure 7.6: Head-to-head metrics comparison of LightGBM vs CatBoost



```
[7] plt.figure(figsize=(10, 6))
    sns.barplot(x=feature_importance['Importance'], y=feature_importance['Feature'])
    plt.title("Feature Importance - LightGBM")
    plt.show()
```

Figure 7.7: Feature Importance-LightGBM



```
print("\nTesting on New Sample Data...")
sample_data = np.array(X_test[0]).reshape(1, -1)  # Take a sample test row
catboost_prediction = catboost_model.predict(sample_data)
lgbm_prediction = lgbm_model.predict(sample_data)

print(f"CatBoost Prediction: {'Ransomware' if catboost_prediction[0] == 1 else 'Benign'}")
print(f"LightGBM Prediction: {'Ransomware' if lgbm_prediction[0] == 1 else 'Benign'}")

print("\nHybrid Ransomware Detection System Execution Complete 🚀")
```

```
Testing on New Sample Data...
[LightGBM] [Warning] min_data_in_leaf is set=10, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=10
CatBoost Prediction: Ransomware
LightGBM Prediction: Ransomware

Hybrid Ransomware Detection System Execution Complete 🚀
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

Figure 7.8: Terminal/log outputs showing model training progress or final metrics

Chapter 8
CONCLUSION AND FUTURE WORK

The rise of sophisticated ransomware requires a multifaceted approach for identification. This project proposes a unified ransomware detection framework that merges both static and dynamic features, employing advanced machine learning models—CatBoost and LightGBM—to attain accurate classification. The system achieves high detection rates while maintaining interpretability and scalability. Through the use of feature importance analysis and a carefully designed strategy, the model can identify new and subtle ransomware variants.

The research highlights the importance of integrated analysis in detecting malware and sets a foundation for future advancements, including real-time deployment and automated threat response systems. This study introduced a dual-feature ransomware detection system that integrates static and dynamic analysis to enhance detection efficiency. Employing the attributes of Portable Executable (PE) headers combined with real-time behavioral indicators such as file changes, registry modifications, and network actions, the system creates a comprehensive feature vector for accurate classification. The use of advanced gradient boosting models, namely CatBoost and LightGBM, has enabled the system to achieve high accuracy and robust performance, demonstrated by evaluation metrics like precision, recall, and F1-score.

The combination of static and dynamic traits provides a more comprehensive understanding of executable actions, helping to overcome the limitations of depending solely on a single analytical method. Static characteristics offer quick insights without running processes, while dynamic characteristics uncover actions during execution, including evasive or postponed execution strategies used by sophisticated ransomware. The integration ensures enhanced resilience to obfuscation and zero-day variations. Feature significance evaluation improves clarity by identifying essential indicators of ransomware, assisting cybersecurity experts in understanding decision-making procedures and bolstering defenses. To summarize, this approach improves detection accuracy and strengthens the system's ability to generalize across different types of ransomware and benign applications

The suggested dual-feature ransomware detection system offers various possibilities for future enhancement and real-world use. A key emphasis is integrating this model into real-time endpoint security solutions, enabling swift detection and response to ransomware events. To adapt to the rapidly evolving ransomware environment, upcoming initiatives might focus on utilizing online learning algorithms that continually update the model with new threat data. This would ensure that the system remains effective against emerging ransomware varieties. Additionally, improving the efficiency of dynamic analysis through lightweight sandboxing techniques can reduce computational expenses, making the system suitable for environments with constrained resources.

Enhancing adversarial robustness is another crucial area, as attackers may attempt to bypass detection through evasion techniques; adversarial training and robust optimization methods can strengthen model durability. The system can benefit from broader dataset integration, including real ransomware samples and benign files from various platforms to enhance generalization. Incorporating explainable AI (XAI) techniques will further assist security analysts in understanding model decisions, fostering trust and transparency. Moreover, broadening the model for multi-label classification allows it to recognize and categorize different malware families. In the end, deploying the system as a scalable cloud service can enable unified monitoring and security throughout corporate networks. These forthcoming innovations aim to transform the current model into an integrated, adaptable, and intelligent cybersecurity framework for tackling intricate ransomware issues.

Chapter 9

REFERENCES

[1] L. Svet, A. Brightwell, A. Wildflower, and C. Marshwood, “Unveiling Zero-Space Detection: A Novel Framework for Autonomous Ransomware Identification in High-Velocity Environments,” arXiv.org, 2025.https://arxiv.org/abs/2501.12811 (accessed May 15, 2025).

[2] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, “Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions,” Applied Sciences, vol. 12, no. 1, p. 172, Dec. 2021, doi: https://doi.org/10.3390/app12010172.

[3] D. W. Fernando, N. Komninos, and T. Chen, “A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques,” IoT, vol. 1, no. 2, pp. 551–604, Dec. 2020,
doi: https://doi.org/10.3390/iot1020030.

[4] Chutitep Woralert, C. Liu, and Z. Blasingame, “Towards Effective Machine Learning Models for Ransomware Detection via Low-Level Hardware Information,” pp. 10–18, Oct. 2024, doi: https://doi.org/10.1145/3696843.3696847. [5] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, “PayBreak : Defense Against Cryptographic Ransomware,” doi: https://doi.org/10.1145/3052973.3053035.

[6] X. Xu, M. Zhang, J. Hirokawa, and M. Ando, “E-Band Plate-Laminated Waveguide Filters and Their Integration Into a Corporate-Feed Slot Array Antenna With Diffusion Bonding Technology,” IEEE Transactions on Microwave Theory and Techniques, vol. 64, no. 11, pp. 3592–3603, Sep. 2016,
doi: https://doi.org/10.1109/tmtt.2016.2602859.

[7] A. Salmanogli, H. Selcuk Gecim, and Erhan Piskin, “Plasmonic System as a Compound Eye: Image Point-Spread Function Enhancing by Entanglement,” IEEE Sensors Journal, vol. 18, no. 14, pp. 5723–5731, Apr. 2018,
doi: https://doi.org/10.1109/jsen.2018.2830970.

[8] J. Lee, J. Kim, H. Jeong, and K. Lee, “A Machine Learning-Based Ransomware Detection Method for Attackers’ Neutralization Techniques Using Format-Preserving Encryption,” Sensors, vol. 25, no. 8, p. 2406, Apr. 2025, doi: https://doi.org/10.3390/s25082406.

[9] Aldin Vehabovic et al., “Federated Learning Approach for Distributed Ransomware Analysis,” Lecture notes in computer science, pp. 621–641, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-41181-6$_3$3. $^{\circ}$

[10] P. Azugo, H. Venter, and M. W. Nkongolo, “Ransomware Detection and Classification Using Random Forest: A Case Study with the UGRansome2024 Dataset,” arXiv.org, 2024. https://arxiv.org/abs/2404.12855

[11] Jamil Ispahany, MD Rafiqul Islam, MD Zahidul Islam, and M. Arif Khan, “Ransomware detection using machine learning: A review, research limitations and future directions,” IEEE access, vol. 12, pp. 1–1, Jan. 2024,
doi: https://doi.org/10.1109/access.2024.3397921.

[12] A. Alraizza and A. Algarni, “Ransomware Detection Using Machine Learning: A Survey,” Big Data and Cognitive Computing, vol. 7, no. 3, p. 143, Sep. 2023,
doi: https://doi.org/10.3390/bdcc7030143.

[13] H.-N. Nguyen, H.-T. Nguyen, and D. Lescos, “Detection of ransomware attacks using federated learning based on the CNN model,” arXiv.org, May 01, 2024. https://arxiv.org/abs/2405.00418

[14] M. Nkongolo and Mahmut Tokmak, “Ransomware Detection Using Stacked Autoencoder for Feature Selection,” Indonesian Journal of Electrical Engineering and Informatics (IJEEI), vol. 12, no. 1, Mar. 2024,
doi: https://doi.org/10.52549/ijeei.v12i1.5109.

[15] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, “A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments,” Sensors, vol. 25, no. 4, p. 1153, Feb. 2025, doi: https://doi.org/10.3390/s25041153.

[16] Rajat Shri Shrimal, Jyoti Gajrani, V. K. Jain, M. Tripathi, and Dharm Singh Jat, “Detection of Ransomware Attacks Using Weight of Evidence Technique,” Aug. 2023, doi: https://doi.org/10.1109/etncc59188.2023.10284928.