PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION

A PROJECT REPORT

Submitted by

ABIRAMI.R (211420104006)

KALAI ARASI.R (211420104117)

MUTHU USHA RANI.M(211420104173)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION"** is the bonafide work of **ABIRAMI.R (211420104006), KALAI ARASI.R (211420104117) and MUTHU USHA RANI M (211420104173)** who carried out the project work under my supervision.

Signature of the HOD with date

DR L.JABASHEELA M.E., Ph.D.,

PROFESSOR AND HEAD,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai – 123

Signature of the Supervisor with date

Dr.K.SANGEETHA M.E.,Ph.D.,
SUPERVISOR
ASSOCIATE PROFESSOR

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Certified that the above candidates were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **ABIRAMI.R(211420104006), KALAIARASI .R (211420104117)** and **MUTHU USHA RANI .M (211420104173)** here by declare that this project report titled **"PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION",** under the guidance of **Dr K. SANGEETHA , M.E., Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ABIRAMI.R

KALAI ARASI.R

MUTHU USHA RANI .M

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., M.Phil., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator, Dr. KAVITHA SUBRAMANI , M.E., Ph.D.,** and **Project Guide**, **Dr.K. SANGEETHA, M.E., Ph.D.,** and all the faculty members of the Department of Computer Science and Engineering for their unwavering support for thesuccessful completion of the project.

ABIRAMI. R

KALAI ARASI. R

MUTHU USHA RANI. M

SPIRO PRIME TECH SERVICES

19.03.2024

To Whomsoever It May Concern

This is to certify that **Ms. ABIRAMI R (Reg No: 211420104006)**, a student of final year B.E. of **"PANIMALAR ENGINEERING COLLEGE "** has completed her major project with great success at our concern, under the Title: **"PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION"** from **JANUARY 2024** to **MARCH 2024.**

Her project is found to be relevant regarding her stream and she had submitted a copy of the project report to us. During her Project period we found her sincere & hard working & possessing a good behaviour and a moral character.

We wish her grand success in future endeavours.

For SPIRO PRIME TECH SERVICES,

SPIRO PRIME TECH SERVICES NO.78, IIIrd Floor, Usman Road, T.Nagar. CHENNAI-600 017

M.SAMPATH KUMAR

MANAGER

Corporate Office : # 78, 3rd Floor, Usman Road, T.Nagar, Chennai - 600 017.
Mail: hr@spiroit.com Mobile: 9791 044 044 / www.spiroprojects.com.

SPIRO PRIME TECH SERVICES

19.03.2024

To Whomsoever It May Concern

This is to certify that **Ms. KALAI ARASI R (Reg No: 211420104117)**, a student of final year **B.E.** of **"PANIMALAR ENGINEERING COLLEGE "** has completed her major project with great success at our concern, under the Title: **"PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION"** from **JANUARY 2024** to **MARCH 2024.**

Her project is found to be relevant regarding her stream and she had submitted a copy of the project report to us. During her Project period we found her sincere & hard working & possessing a good behaviour and a moral character.

We wish her grand success in future endeavours.

For SPIRO PRIME TECH SERVICES,

SPIRO PRIME TECH SERVICES NO.78, IIIrd Floor, Usman Road, T.Nagar. CHENNAI-600 017

M.SAMPATH KUMAR

MANAGER

SPIRO PRIME TECH SERVICES

19.03.2024

To Whomsoever It May Concern

This is to certify that **Ms. MUTHU USHA RANI M (Reg No: 211420104173)**, a student of final year **B.E.** of **"PANIMALAR ENGINEERING COLLEGE "** has completed her major project with great success at our concern, under the Title: **"PREDICTION OF ANDROID RANSOMWARE ATTACK USING CATEGORIAL CLASSIFICATION"** from **JANUARY 2024** to **MARCH 2024.**

Her project is found to be relevant regarding her stream and she had submitted a copy of the project report to us. During her Project period we found her sincere & hard working & possessing a good behaviour and a moral character.

We wish her grand success in future endeavours.

For SPIRO PRIME TECH SERVICES,

SPIRO PRIME TECH SERVICES NO.78, IIIrd Floor, Usman Road, T.Nagar. CHENNAI-600 017

M.SAMPATH KUMAR

MANAGER

ABSTRACT

Android ransomware has become a significant cybersecurity threat, posing serious risks to mobile devices and users' data. To effectively combat this growing menace, predictive models that can identify and thwart ransomware attacks in real-time are crucial. This research presents a novel approach for predicting Android ransomware attacks using categorial classification techniques. Data pre-processing is performed to extract and transform essential features from Android applications. Subsequently, the categorial classification model is trained on a labelled dataset comprising both benign and ransomware applications. The research findings demonstrate the efficacy of the proposed approach in predicting Android ransomware attacks with high accuracy. The model shows promising results in identifying malicious applications, thus providing a proactive defense against ransomware threats on Android devices. By implementing categorial classification techniques, the proposed system enables timely identification and mitigation of potential threats, contributing to a safer and more secure mobile ecosystem. The accuracy of Bernoulli Naïve Bayes Algorithm, Extreme Gradient Boosting Algorithm and Random Forest Algorithm are 67.595, 100 and 100 respectively.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

OVERVIEW

In recent years, the proliferation of mobile devices, particularly those powered by the Android operating system, has brought about unprecedented convenience in our daily lives. However, with this surge in technological advancements comes an alarming increase in cyber threats, with Android ransomware attacks standing out as a pervasiveand evolving menace. The need for a nuanced understanding of these attacks is paramount, as they continue to target individuals, businesses, and even governmental organizations.This project delves into the intricate landscape of Android ransomware attacks, employing a categorial classification framework to unravel the diverse strategies employed by malicious actors. As we embark on this exploration, it is crucialto recognize the gravity of the situation: the potential compromise of personal and sensitive information, financial extortion, and the disruption of essential services. Our objective is not only to comprehend the technical intricacies of these attacks but also to develop a robust defense mechanism against this ever-evolving threat.

Android ransomware has undergone a remarkable evolution since its inception. Inthe early days, these malicious programs were relatively simplistic, relying on basic techniques to trick users into installing infected applications. However, as the Android ecosystem matured and became more secure, cybercriminals adapted, introducing sophisticated strategies to bypass security measures.The evolution of Android ransomware can be traced through key milestones, such as the shift from generic attacks to targeted campaigns. Initially, attackers cast a wide net, attempting tocompromise as many devices as possible. Over time, however, they have become morediscerning, focusing on high-value targets and employing advanced tactics to maximize their impact.

CYBER CRIMINIAL

The impact of Android ransomware extends far beyond the digital realm, affectingboth individuals and organizations on a global scale. Personal users may find themselves locked out of their devices, with precious photos, messages, and personal information held hostage. Financial extortion becomes a pressing concern as victims are coerced into paying ransoms to regain access to their data.In the corporate landscape, Android ransomware poses a severe threat to the confidentiality and integrity of sensitive information. Cybercriminals are increasingly targeting businesses, encrypting critical files and demanding substantial ransoms. The disruption caused by such attacks can lead to financial losses, reputational damage, and even legal consequences.Understanding the motivations behind Android ransomware attacks is crucial for developing effective countermeasures. Cybercriminals are driven by a rangeof motives, and comprehending these motivations provides insight into their modus operandi. Financial gain remains a primary driver, with attackers seeking monetary rewards through extortion. The lure of quick and substantial profits incentivizes the development and deployment of increasingly sophisticated ransomware strains.

To systematically analyze the diverse landscape of Android ransomware attacks, we introduce a categorial classification framework. This framework categorizes attacks based on key criteria, facilitating a comprehensive understanding of their unique characteristics. Ransomware communicates with its command and control servers is a pivotal aspect of classification which is done by the use of communication channels. Analyzing communication channels helps in identifying patterns, allowing for the development of network-based defenses and threat intelligence sharing.Android ransomware relies on a variety of exploitation techniques to compromise the security of mobile devices. One prevalent method involves exploiting vulnerabilities in the Android operating system or third-party applications. Cybercriminals adeptly leverage known

weaknesses, often targeting devices running outdated software. As such, regular system updates and patch management emerge as critical components of robust defensestrategy exploitation.

CRYPTOGRAPHY

The success of Android ransomware hinges on its ability to encrypt user data securely, rendering it inaccessible without the decryption key. The encryption mechanisms employed by ransomware strains vary in complexity, with some utilizingindustry-standard cryptographic algorithms while others employ custom or hybrid methods. Understanding these encryption techniques is crucial for devising effective countermeasures.By comprehending the exploitation techniques and encryption mechanisms employed by Android ransomware, cybersecurity professionals can betteranticipate and respond to emerging threats. Proactive measures, including regular software updates, user education on social engineering tactics, and the development of decryption tools, are essential components of a holistic defense strategy against the evolving technological landscape of Android ransomware.

Therefore our project provides a comprehensive understanding of Android ransomware attacks through a categorial classification framework. By unraveling theintricacies of these threats, we aim to empower individuals, businesses, and cybersecurity professionals to fortify their defenses against this pervasive menace. As the threat landscape continues to evolve, continuous vigilance, adaptive strategies, andcollaborative efforts are imperative to mitigate the impact of Android ransomware attacks on our increasingly interconnected digital world.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the areaand precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinentor relevant of them.Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. Some of the survey's to understand the past and present perspective of loan approval or not.

2.1 Review of Literature Survey

Title: A note on machine learning applied in ransomware detection

Author: Manuela Horduna1 , Simona-Maria L˘az˘arescu1 , and Emil Simion2

Year: 2022

Ransomware is a malware that employs encryption to hold a victim's data,causing

irreparable loss and monetary incentives to individuals or businessorganizations. The occurrence of ransomware attacks has been increasing significantly and as the attackers are investing more creativity and inventiveness into their threats, the struggle of fighting against ill-themed activities has become more difficult and even time and energy-draining.

Therefore, recent researches try to shed some light on combining machine learning with defense mechanisms for detecting this type of malware. Machine learning allows anti-ransomware systems to become more accurate at predicting outcomes or behaviors of the attacks and is vastly used in the advanced research of cybersecurity. In this paper we analyze how machine learning can improve malware recognition in order to stand against critical security issues, giving a brief, yet comprehensive overview of this thriving topic in order to facilitate future research. We also briefly present the most important events of 2022 in terms of ransomware attacks, providing details about the ransoms demanded.

Title: Opportunities for Early Detection and Prediction of Ransomware Attacks against Industrial Control Systems

Author: Mazen Gazzan 1,2 and Frederick T. Sheldon 1,*

Year: 2022

Industrial control systems (ICS) and supervisory control and data acquisition (SCADA) systems, which control critical infrastructure such as power plants andwater treatment facilities, have unique characteristics that make them vulnerable to ransomware attacks. These systems are often outdated and run on proprietary software, making them difficult to protect with traditional cybersecurity measures. The limited visibility into these systems and the lack of effective threat intelligence pose significant challenges to the early detection and prediction of ransomware attacks. Ransomware attacks on ICS and SCADA systems have become a growing concern in recent years. These attacks can causesignificant disruptions to critical infrastructure and result in significant financiallosses. Despite the increasing threat, the prediction of ransomware attacks on ICS remains a significant challenge for the

cybersecurity community. This is dueto the unique characteristics of these systems, including the use of proprietary software and limited visibility into their operations. In this review paper, we willexamine the challenges associated with predicting ransomware attacks on industrial systems and the existing approaches for mitigating these risks. We willalso discuss the need for a multi-disciplinary approach that involves a close collaboration between the cybersecurity and ICS communities. We aim to provide a comprehensive overview of the current state of ransomware predictionon industrial systems and to identify opportunities for future research and development in this area. Keywords: ransomware; industrial control systems; SCADA; ransomware detection and prevention; attack likelihood prediction; situation awareness; security assessment.

Title: Detecting Malicious Accounts in Online Developer Communities Using Deep Learning

Author: Qingyuan Gong, Yushan Liu, Jiayun Zhang, Yang Chen

Year : 2021

Online developer communities like GitHub allow a massive number of developers to collaborate. However, the openness of the communities makes them vulnerable to different types of malicious attacks, since attackers can easilyjoin these communities and interact with legitimate users. In this work, we propose GitSec, a deep learning-based solution for detecting malicious accountsin online developer communities. GitSec distinguishes malicious accounts fromlegitimate ones based on the account profiles, dynamic activity characteristics, as well as social interactions. First, GitSec introduces two user activity sequences and applies a parallel neural network design with an attention mechanism to process the sequences. Second, GitSec constructs two graphs to represent the interactions between users according to their repository operations. Especially, graph neural networks and structural hole theory are employed to deal with the two constructed graphs. Third, GitSec makes use of the descriptivefeatures to enhance the detection performance. The final judgement is made by a decision maker implemented by a supervised machine learning-based

classifier. Based on the real-world data of GitHub users, our comprehensive evaluations show that GitSec achieves a better performance than state-of-the-art solutions, with an AUC value of 0.916. Index Terms—Online Developer Communities, Malicious Account Detection, Social Networks, Deep Learning, Graph Neural Networks, Structural Hole Theory.

Title: Machine Learning Methods For Malware Detection And Classification

Author: Kateryna Chumachenko

Year: 2017

Malware detection is an important factor in the security of the computer systems. However, currently utilized signature-based methods cannot provide accurate detection of zero-day attacks and polymorphic viruses. That is why the need for machine learning-based detection arises. The purpose of this work was to determine the best feature extraction, feature representation, and classification methods that result in the best accuracy when used on the top of Cuckoo Sandbox. Specifically, k-Nearest-Neighbors, Decision Trees, Support Vector Machines, Naive Bayes and Random Forest classifiers were evaluated. The dataset used for this study consistsed of the 1156 malware files of 9 families of different types and 984 benign files of various formats. This work presents recommended methods for machine learning based malware classification and detection, as well as the guidelines for its implementation. Moreover, the study performed can be useful as a base for further research in the field of malware analysis with machine learning methods.

Title: Analysis, Detection, and Classification of Android Malware using System Calls

Author: Shubham Shakya1 and Mayank Dave2

Year: 2020

With the increasing popularity of Android in the last decade, Android is popular

among users as well as attackers. The vast number of android users grabs the attention of attackers on android. Due to the continuous evolution of the varietyand attacking techniques of android malware, our detection methods should needan update too. Most of the researcher's works are based on static features, and very few focus on dynamic features. In this paper, we are filling the literature gap by detecting android malware using System calls. We are running the malicious app in a monitored and controlled environment using an emulator to detect malware. Malicious behavior is activated with some simulated events during its runtime to activate its hostile behavior. Logs collected during the app'sruntime are analyzed and fed to different machine learning models for Detectionand Family classification of Malware. The result indicates that K-Nearest Neighbor and the Decision Tree gave the highest accuracy in malware detectionand Family Classification respectively.

CHAPTER 3
THEORETICAL BACKGROUND

CHAPTER 3

THEORICAL BACKGROUND

Implementation Environment

Software Requirements:

Operating System	: Windows
Tools	: Anaconda with Jupyter Notebook,Visual Studio Code
Backend	: Django(Python Framework)
Frontend	: HTML,CSS,JavaScript,Bootstrap.

Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 100 GB

RAM : minimum 4 GBT

3.2 System Architecture

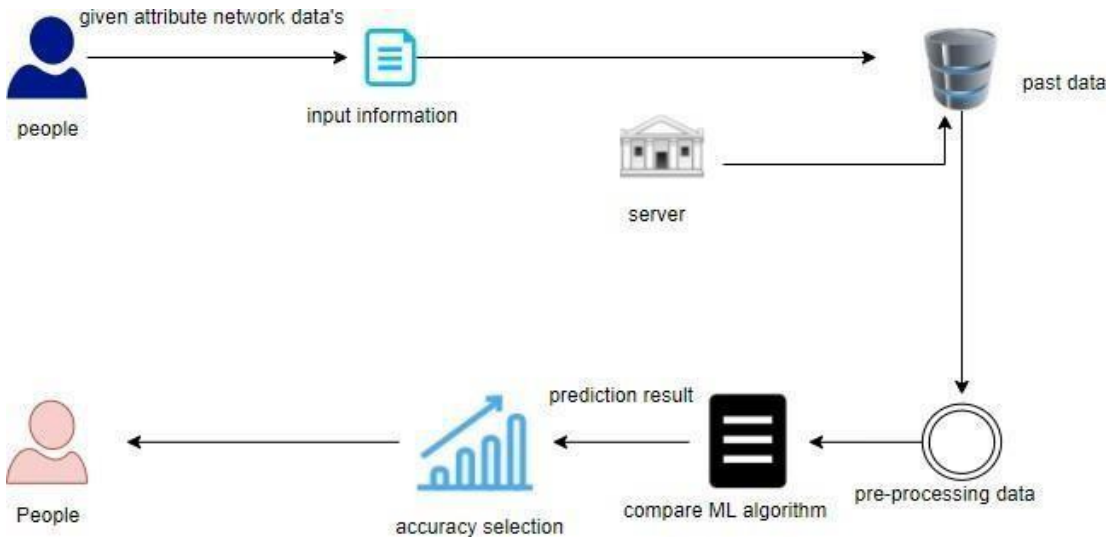


Figure 3.2.1 System Architecture Diagram

The diagram showcases a data processing and prediction flow using machine learning algorithms. It starts with individuals providing input information, which includes given attribute network data. This data is then sent to a server that also receives past data, likely used to enhance prediction accuracy. The server preprocesses the data, cleaning and organizing it for analysis. A machine learning algorithm is then employed to analyze this pre-processed data, leading to a prediction result. An accuracy selection process evaluates the prediction results for their precision, possibly comparing different machine learning models or algorithms. The most of completing a cyclic process that allows for ongoing refinement and improvement of prediction accuracy as more data becomes available and algorithms are fine-tuned.

3.3 Proposed Methodology

Existing System:

The authors introduce msdroid, an innovative Android malware detection system that addresses two critical challenges faced in the field: robustness in real-world scenarios and the need for interpretable explanations. The system takes inspiration from common practices of security analysts by filtering APIs and focusing

on local snippets surrounding sensitive APIs, rather than analyzing the entire program. Each snippet is represented as a graph, incorporating both code attributes and domain knowledge, which is then classified using a Graph Neural Network. This local perspective allows the GNN classifier to concentrate on code segments highly correlated with malicious behaviors, enhancing the system's robustness. Moreover, the information contained in the graphs contributes to a better understanding of the behaviors, making msdroid more interpretable in nature. The efficacy of msdroid is extensively evaluated through a comprehensive comparison with five baseline methods on a dataset of more than 81K apps in various real-world scenarios, including zero-day, evolution, and obfuscation. The experimental results demonstrate that msdroid excels in robustness, outperforming state-of-the-art systems in all cases, with a notable advantage of 5.37% to 49.52% in F1-score. msdroid represents a significant advancement in Android malware detection, offering improved robustness and interpretability, which are crucial for addressing the ever-evolving landscape of cybersecurity threats in the mobile ecosystem.

Disadvantages:

- They did not implement the deployment process.
- Accuracy was low.
- They did just limited records of data.

Proposed System:

The proposed system aims to predict Android ransomware attacks using categorial classification techniques. To achieve this, a diverse dataset of Android applications, encompassing both benign and ransomware samples, is collected. Relevant features, including permissions, API calls, and other attributes, are extracted from each app. Categorial classification algorithms, such as Logistic Regression, Decision Trees, Random Forests, or Support Vector Machines, are employed to train the model to predict whether an application is benign or contains ransomware. The performance of the model is thoroughly evaluated using accuracy, precision, recall, F1-score, and confusion matrix metrics. Additionally, an analysis of feature importance is conducted to identify attributes strongly associated with ransomware

attacks. To enhance interpretability, an explanation mechanism is implemented using techniques like SHapley Additive exPlanations or local interpretable model-agnostic explanations, providing understandable insights into the prediction process. Once the model demonstrates satisfactory performance, it is integrated into an Android application or system, enabling real-time scanning and prediction of ransomware attacks. Throughout the development process, utmost attention is given to ethical practices and security considerations to handle potential risks associated with malware detection projects. Ultimately, the proposed system contributes to enhancing mobile security by providing an effective means of detecting and preventing Android ransomware attacks.

Merits:

- Develop a full-stack application for deployment purposes.
- Accuracy was improved.
- Huge amount of data.

Dataset Description

Ransomware is a type of malicious software that encrypts users' data and demands payment for the decryption key. The effects of ransomware on userscan be devastating, ranging from the loss of important data and disruption of business operations to monetary losses. Additionally, it can cause psychological strain, as victims often find themselves in a helpless situation with their only option being to pay the ransom. This situation can lead to diminished trust in online security solutions, increased risk of further attacks, difficulty recovering data, and the potential for identity theft. To protect against ransomware attacks, one effective option is the use of machine learning. Machine learning algorithms can be used to detect anomalous activity that isassociated with ransomware attacks. This could include things like sudden increases infile access or writing activity, or changes in system resources. AI-based security solutions can then analyze events across the network and alert system administrators to possible attack threats. Finally, machine learning algorithms can be used to analyzethe data from past ransomware attacks and create

models to better predict the behaviorof similar future attacks. This can help system administrators quickly respond to any new attacks. The dataset contains the network monitoring records of android devices to determine the types of ransomware along with benign which have been transacted in the user network.

hash	millisecor	classificat	state	usage_cou	prio	static_pric	normal_pi	policy	vm_pgoff	vm_trunca
42fb5e2ec	0	malware	0	0	3.07E+09	14274	0	0	0	13173
42fb5e2ec	1	malware	0	0	3.07E+09	14274	0	0	0	13173
42fb5e2ec	2	malware	0	0	3.07E+09	14274	0	0	0	13173
42fb5e2ec	3	malware	0	0	3.07E+09	14274	0	0	0	13173
42fb5e2ec	4	malware	0	0	3.07E+09	14274	0	0	0	13173
42fb5e2ec	5	malware	0	0	3.07E+09	14274	0	0	0	13173

Figure 3.3.3.1 Sample of the dataset

Input Design

The dataset contains 203556 rows and 85 columns and the entire data has 10 types of Android Ransomware and Benign traffic types. The types of Ransomware includes SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker,Jisut, Lockerpin and Pletor. The distribution of the data labels are as follows:

- SVpeng Label contains 54161 Records.
- PornDroid Label contains 46082 Records.
- Koler Label contains 44555 Records.
- Benign Label contains 43091 Record.
- RansomBO Label contains 39859 Records.
- Charger Label contains 39551 Records.
- Simplocker Label contains 36340 Records.
- WannaLocker Label contains 32701 Records.
- Jisut Label contains 25672 Records.
- Lockerpin Label contains 25307 Records.
- Pletor Label contains 4715 Records.

UML Diagrams

Use case Diagram

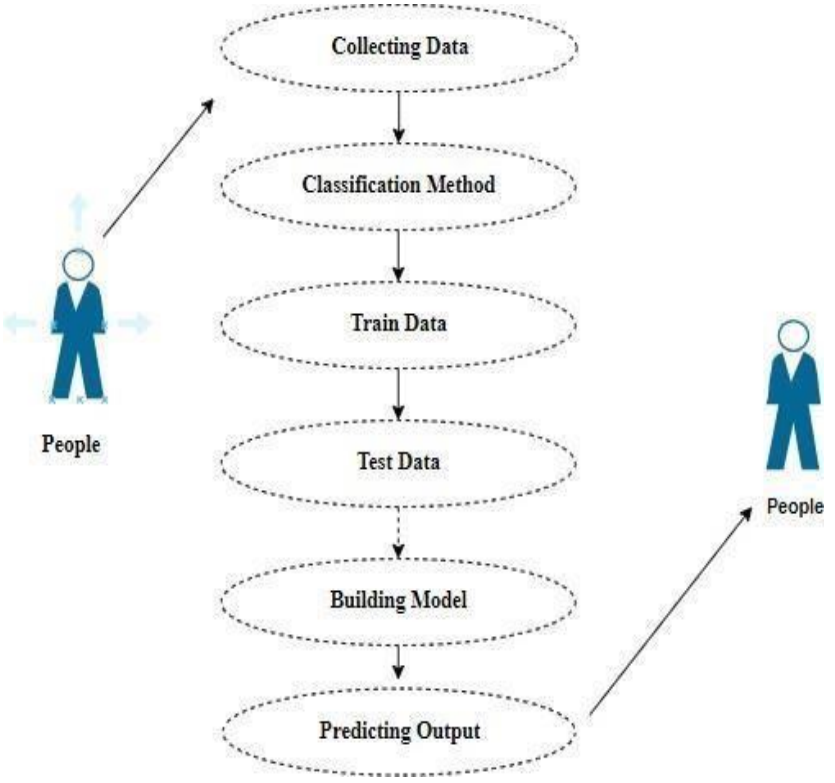


Fig 3.3.5.1 Usecase Diagram

The Usecase diagram represents a flowchart that outlines a data classification process, beginning and ending with human involvement. Initially, individuals gather data, which is then sorted through a classification method. This method is refined by training and testing the data, ensuring the process is accurate and reliable. Subsequently, a model is constructed based on the refined data, which is used to predict outcomes. The final step involves people receiving these predictions, thereby completing the cycle from data collection to result generation.

Class Digram

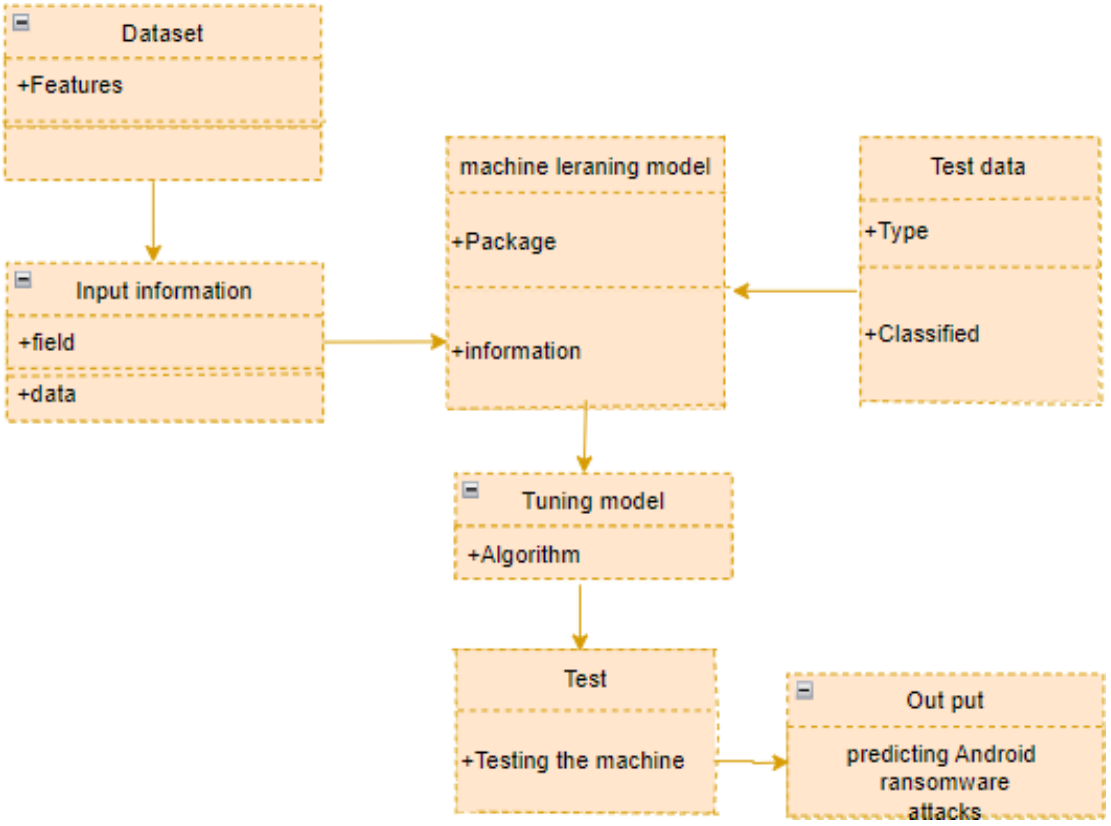


Fig 3.3.5.2 Class Diagram

Class Diagram depicts the classes and their relationships involved in theprediction of Android ransomware. Class diagram starts with a dataset of features, progressing to input information that includes fields and data. This feeds into the machine learning model, which is fine-tuned using specific algorithms. The model is then tested with data characterized by type and classification parameters. Upon successful testing, the model is capable of predicting Android ransomware attacks. The color scheme and design elements further aid in distinguishing each phase of the workflow, emphasizing the structured nature of machine learning in cybersecurity applications. Overall, the image serves as an informative guide to the stages involved in developing a predictive model for Android ransomware threats.

Activity Diagram

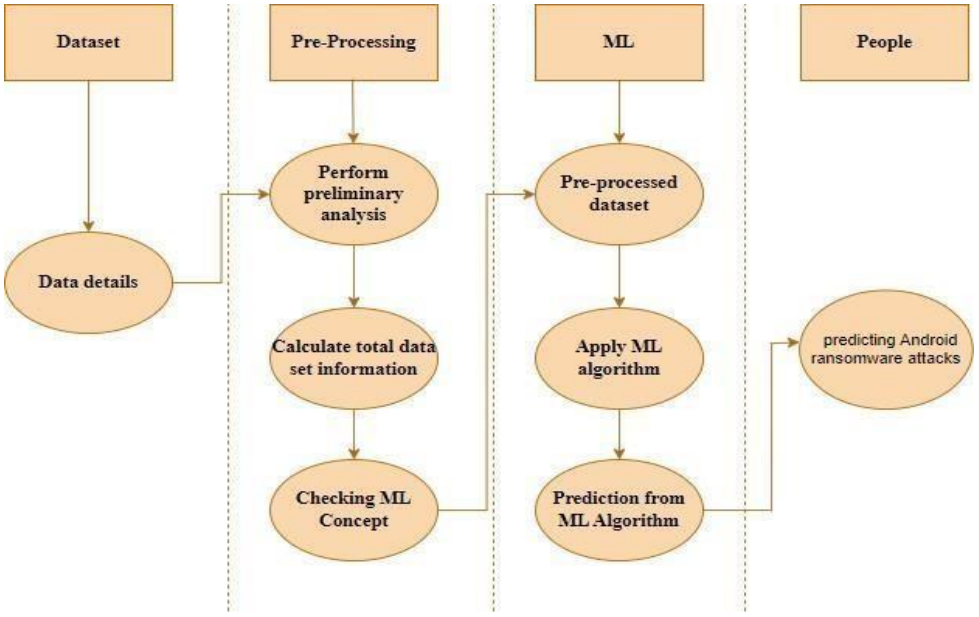


Figure 3.3.5.3 Activity Diagram

Activity Diagram represents the flow of activities involved in predicting Androidransomware using categorial classification. Activity diagram outlines a flowchart for a machine learning process aimed at predicting Android ransomware attacks. It starts with a dataset that includes data details such as preliminary analysis, total dataset information, and ML concept checks. This dataset then goes through a pre-processing stage where it is analyzed and prepared for the ML phase. In the ML phase, an appropriate algorithm is applied to the pre-processed data, leading to predictions. These predictions are ultimately used to forecast Android ransomware attacks, with the final step involving people who can act on these predictions. Overall, the image serves as a visual guide to the stages involved in developing a predictive model for Android ransomware threats.

Sequence Diagram

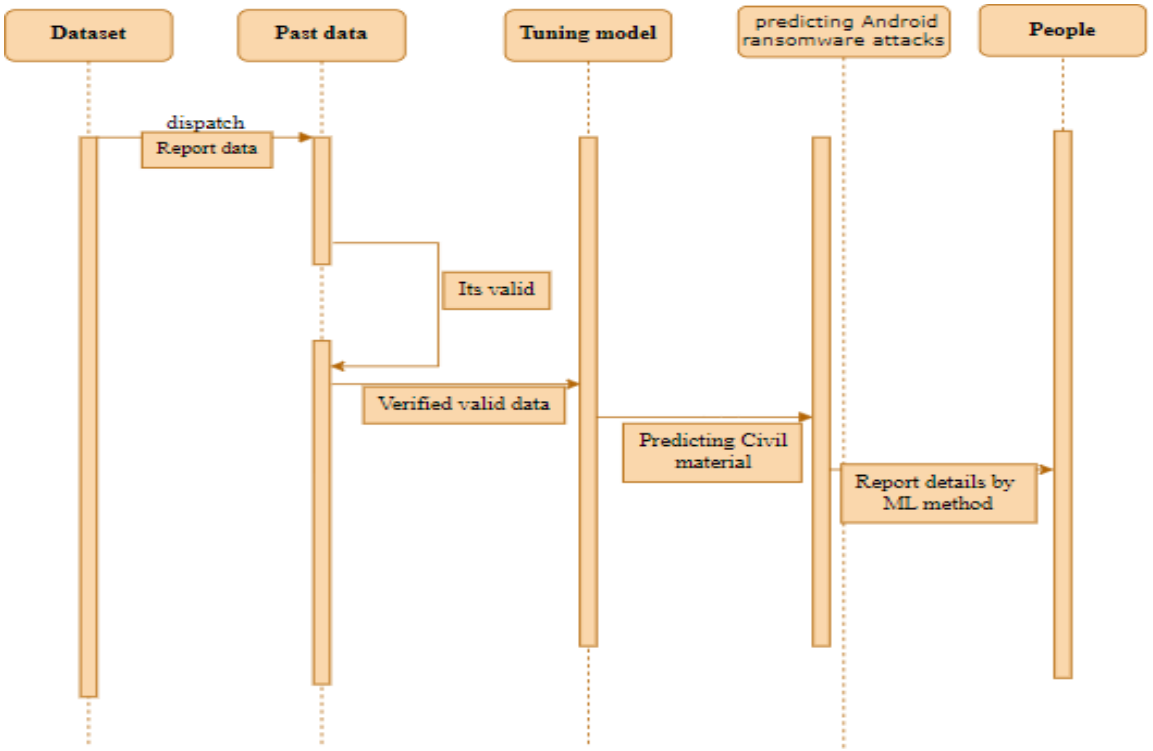


Fig 3.3.5.4 Sequence Diagram

Sequence Diagram illustrates the interactions between various components involved in predicting Android ransomware. The diagram illustrates a machine learning process for predicting Android ransomware attacks. It starts with a dataset, which is validated to ensure accuracy. The validated data is then used in a tuning model to predict civil material related to Android ransomware attacks. The model's predictions are based on the analyzed and processed past data. These predictions are compiled into a report created using machine learning methods. Finally, this detailed report is presented to people, providing insights and information on predicted Android ransomware attacks derived from the analyzed past dataset. Overall, the image serves as a visual guide to the stages involved in developing a predictive model for Android ransomware threats.

CHAPTER 4
SYSTEM IMPLEMENTATION

CHAPTER 4

SYSTEM IMPLMENTATION

Data Pre-processing

The Validation techniques in machine learning are used to get the error rate ofthe Machine Learning (ML) model, which can be considered as close to the true errorrate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data usedto provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a givenmodel, but this is for frequent evaluation. It as machine learning engineers use this datato fine-tune the model hyper parameters. Data collection, data analysis, and the processof addressing data content, quality, and structure can add up to a time-consuming to- do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and itable to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling. The dataset contains 203556 rows and 85 columns and the entire data has 10 types of Android Ransomware and Benign traffic types. The types of Ransomware includes SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker,Jisut, Lockerpin and Pletor. The distribution of the data labels are as follows:

- SVpeng Label contains 54161 Records.
- PornDroid Label contains 46082 Records.
- Koler Label contains 44555 Records.
- Benign Label contains 43091 Record.
- RansomBO Label contains 39859 Records.
- Charger Label contains 39551 Records.
- Simplocker Label contains 36340 Records.
- WannaLocker Label contains 32701 Records.
- Jisut Label contains 25672 Records.
- Lockerpin Label contains 25307 Records.
- Pletor Label contains 4715 Records.

Missing values can have an adverse effect on model performance, and properly treating them is critical for accurate predictions. Several strategies are used to manage missing data, including imputation and elimination. Imputation is the process of fillingin missing numbers with approximated or calculated values based on existing data. Alternatively, removal entails eliminating occurrences with missing values from the dataset. While straightforward, this method is only appropriate when the number of missing values is low and eliminating instances has no substantial influence on the totaldataset. We had no missing values in our data hence there was no neccessity to treat them.

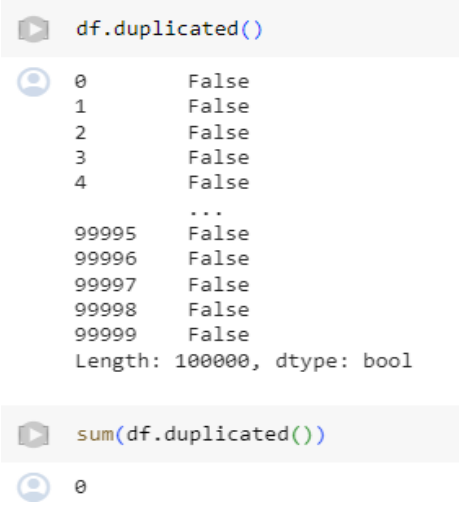


Figure 4.1.1 Checking for duplicate values

Data Visualization

The Importing the library packages with loading given dataset. To analyzing thevariable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training yourmodel that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increasethe value of data in analytics and decision making.

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a littledomain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measuresof association or significance.

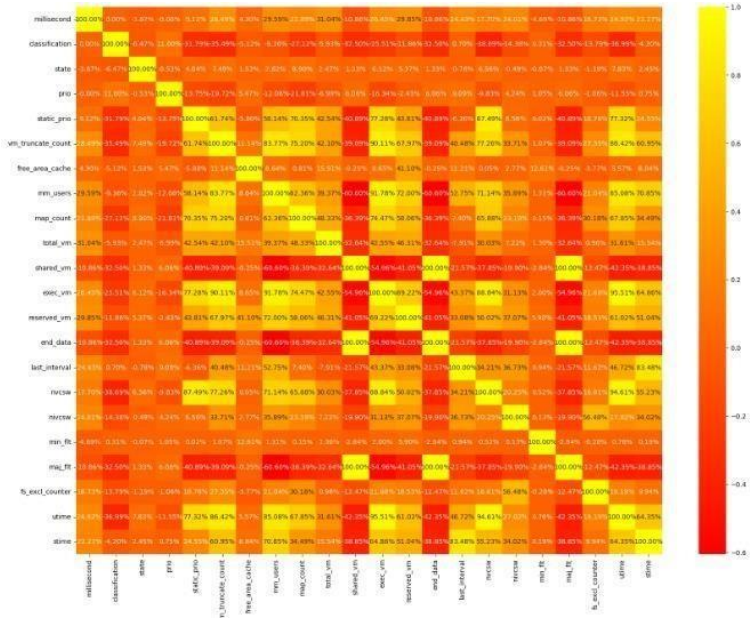


Figure 4.2.1 Heatmap Representation for the fields.

The heatmap representation used in data visualization to represent complex data sets. It displays a matrix of values for different variables, with colors ranging from yellow to dark red indicating the magnitude of each value. The intensity of each cell's color corresponds to the variable's value, allowing for easy identification of patterns and trends. Labels on the axes suggest the heatmap is for technical analysis, possibly related to computer performance or system monitoring. The color scale on the right side helps interpret the values represented by each color.

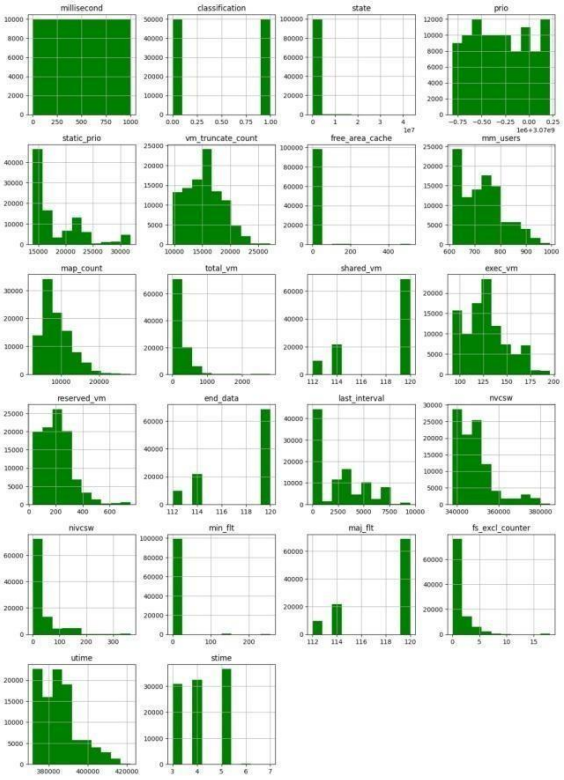


Figure 4.2.2 Histogram Representation for the fields.

This histogram image, each representing a different data set with labels such as "millisecond," "classification," and "state." The histograms vary in height, indicating the distribution of values within each data set. The x-axes are numerically labeled, providing a scale for the data, while the y-axes are unlabeled. This visualization helps in understanding the frequency and distribution of data across various metrics and classifications. Overall, the image provides a clear representation of multiple data sets in a compact and visually accessible format.

Extreme Gradient Boosting Algorithm

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.The mathematical expression for Extreme Gradient boosting algorithm is given as follows:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2$$

$$\mathcal{L}^{(t)} \simeq \sum_{i-1}^{n}[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

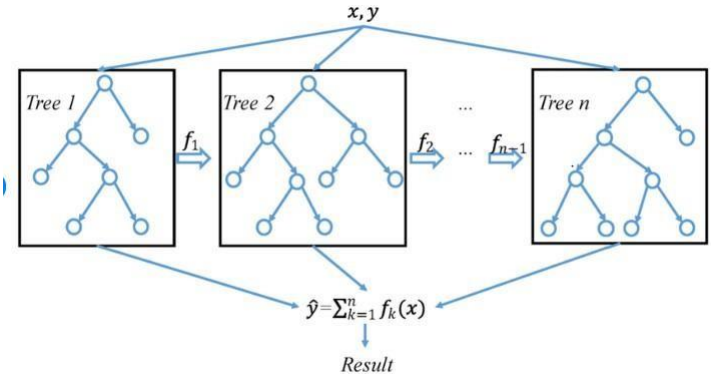


Figure 4.3.1 XG Boost Algorithm Architecture

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of

variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

Random Forest Algorithm

The Random Forest algorithm is a robust machine learning method used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest is an ensemble learning technique, which means it combines the predictions from multiple machine learning algorithms to make more accurate predictions than any individual model.

A key feature of Random Forest is its ability to handle a large number of input variables without variable deletion. It's also resistant to overfitting, thanks to the randomness introduced during the construction of the decision trees. Each tree in the forest is built from a sample drawn with replacement (bootstrap sample) from the training set. Furthermore, when splitting a node during the construction of a tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. This concept is known as the random subspace method.The mathematical expression for Random Forest Algorithm is as follows:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2$$

Where N is the number of data points,
fi is the value returned by the model and
yi is the actual value for data point i.

The algorithm begins with a selection of random samples from a given dataset. It then constructs a decision tree for every sample and gets a prediction result from each tree. When it's time to make a final prediction, the Random Forest takes into account the predictions made by all the individual trees, and the majority vote or

average prediction is considered as the final prediction. This process improves the predictive accuracy and controls over-fitting, making Random Forest a reliable and powerful algorithm for predictive analytics

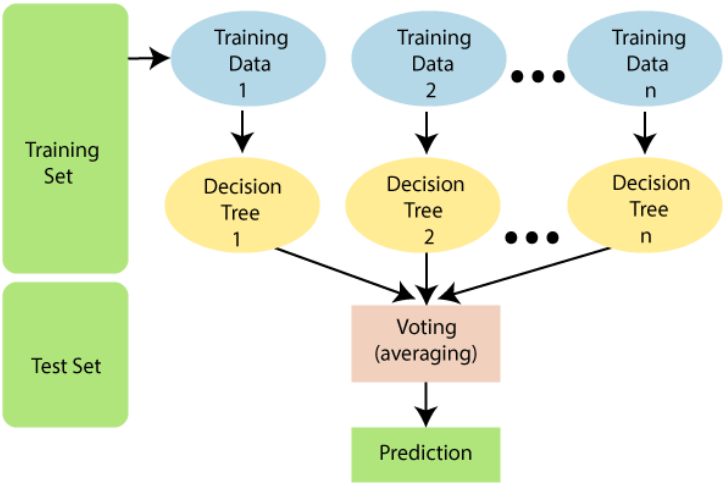


Figure 4.4.1 Random Forest Algorithm Architecture

The Working of the Random Forest Algorithm is quite intuitive. It is implemented in two phases: The first is to combine N decision trees with building the random forest, and the second is to make predictions for each tree created in the first phase.Initially,Pick M data points at random from the training set and create decision trees for your chosen data points (Subsets).Each decision tree will produce a result. Analyze it. For classification and regression, accordingly, the final output is based on Majority Voting or Averaging, accordingly.

4.5. Bernoulli Navie Bayes Algorithm

The Bernoulli Naive Bayes algorithm is a variant of the Naive Bayes algorithm which is particularly suited for binary (boolean) features. It operates under the core principle of Naive Bayes, which is based on Bayes' Theorem, but it assumes that all our features are binary such that they take only two values. Bernoulli Naive Bayes is thus ideal for datasets where features are either present or absent, making it a common choice for text classification tasks where the presence or absence of a word is a feature.The mathematical expression for Bernoulli Navie Bayes Algorithm is given as follows:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

In the Bernoulli Naive Bayes model, the decision rule is based on the Bernoulli distribution. The algorithm calculates the probability of each class for a given sample and the class with the highest probability is considered as the output. It uses a binary occurrence matrix rather than a term frequency count, which is used in multinomial models. This means that it doesn't matter how many times a word appears in a document; rather, it only matters whether the word occurs or not.

When implementing the Bernoulli Naive Bayes algorithm, one typically uses a binary vector to represent the features in a dataset. For example, in text classification, the vector would represent the presence or absence of words within a fixed vocabulary. The algorithm then calculates the parameters for the Bernoulli distribution for each feature independently given the class label. These parameters are then used to make predictions on new samples, providing a simple yet effective classification method.

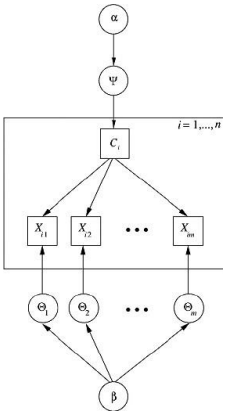


Figure 4.5.1 Bernoulli Naïve Bayes Algorithm Architecture Diagram

4.6 Deployment

In this module the trained deep learning model is converted into hierarchical dataformat file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output whether the given image is CKD / Not CKD.The name of the package is used to resolve resources from inside the package orthe folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an `__init__.py` file inside) or a standard module (just a `.py` file).

CHAPTER 5
RESULTS & DISCUSSIONS

CHAPTER 5
RESULT AND DISCUSSION

ACCURACY

Accuracy in machine learning, accuracy is a frequently used performance metric to assess the accuracy of a model's predictions. It is the measure of the ratio of the number of correct predictions made by the model to the total number of predictions. It is the metric that best captures the performance of the model when the independent variable has a balanced distribution. The formula to calculate accuracy is as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

```
THE CROSS VALIDATION TEST RESULT OF ACCURACY :

[65.15  69.755 67.935 63.19  44.19 ]

THE ACCURACY SCORE OF BERNOULLINB IS : 67.575
```

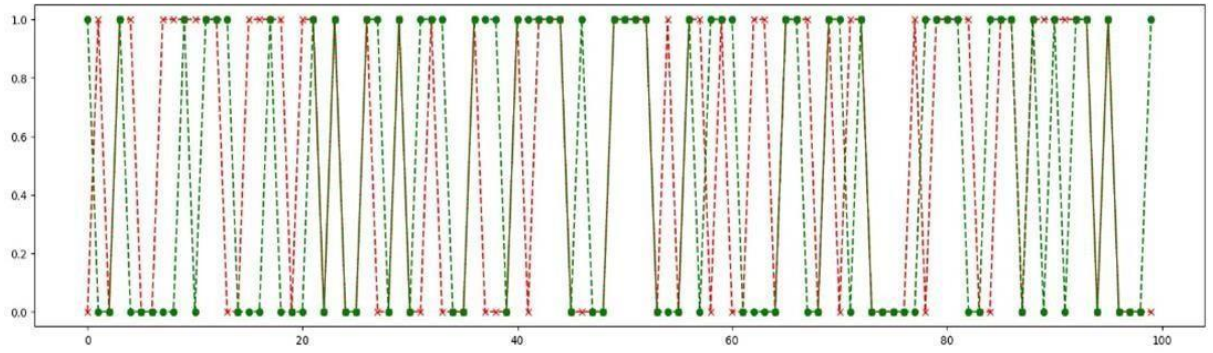


Figure 5.1.1 Accuracy Rate of Bernoullinb

```
THE CROSS VALIDATION TEST RESULT OF ACCURACY :

[86.63  89.405 84.065 79.54  85.775]

THE ACCURACY SCORE OF XGB CLASSIFIER IS : 100.0
```

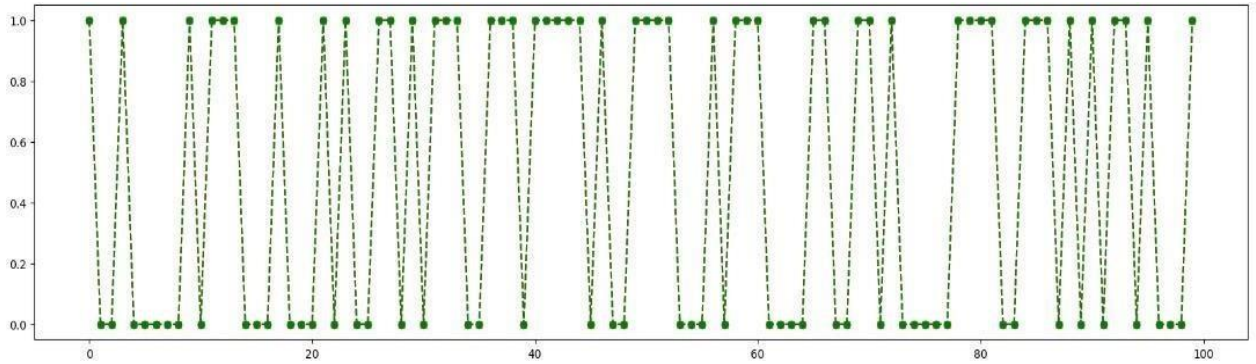


Figure 5.1.2 Accuracy Rate of XGB

```
THE CROSS VALIDATION TEST RESULT OF ACCURACY :

[89.62  89.775 90.705 90.    95.   ]

THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS : 100.0
```

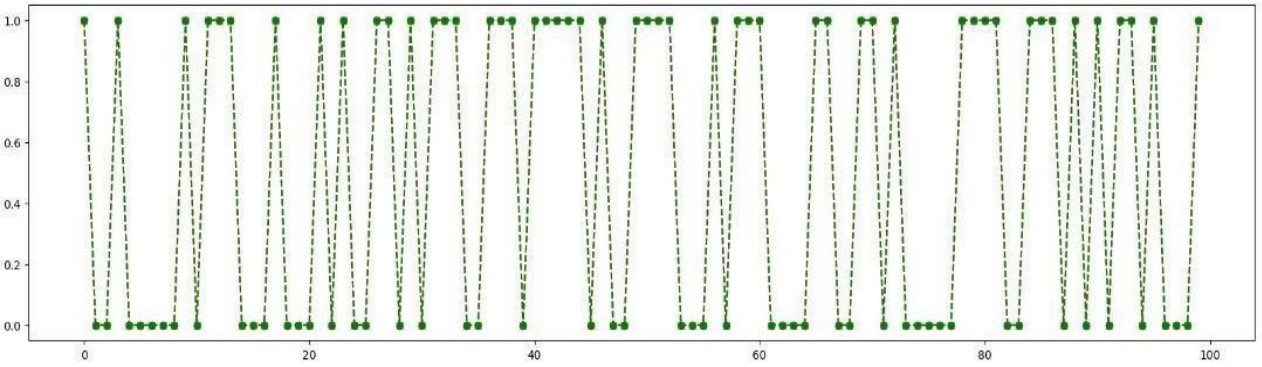


Figure 5.1.3 Accuracy Rate of Random Forest

PRECISION

A performance measure called precision is employed in machine learning to assess how precisely the classification model's positive predictions are achieved. A model with a high precision value is one that makes fewer errors when predicting positive outcomes and has fewer false positive predictions. The precision can be measured using the following formula:

$$\textbf{Precision} = \frac{\textbf{True Positive}}{\textbf{True Positive} + \textbf{True Negative}}$$

RECALL

Along with precision, recall is a performance indicator that quantifies the percentage of positive occurrences that the model correctly recognized as positive. A model with a high recall value will have few incorrect negative predictions and be able to correctly identify the majority of positive events. The recall for a model can be calculated as given below:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

F1 SCORE

It is prevalently used in models to classify data into two groups for assessing the algorithms's efficacy. It is referred to as the harmonic mean of recall and precision.It normally ranges from 0 to 1, with 1 representing the highest possible f1 score. The formula to calculate the f1 score is as follows:

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

THE CLASSIFICATION REPORT OF BERNOULLINB:

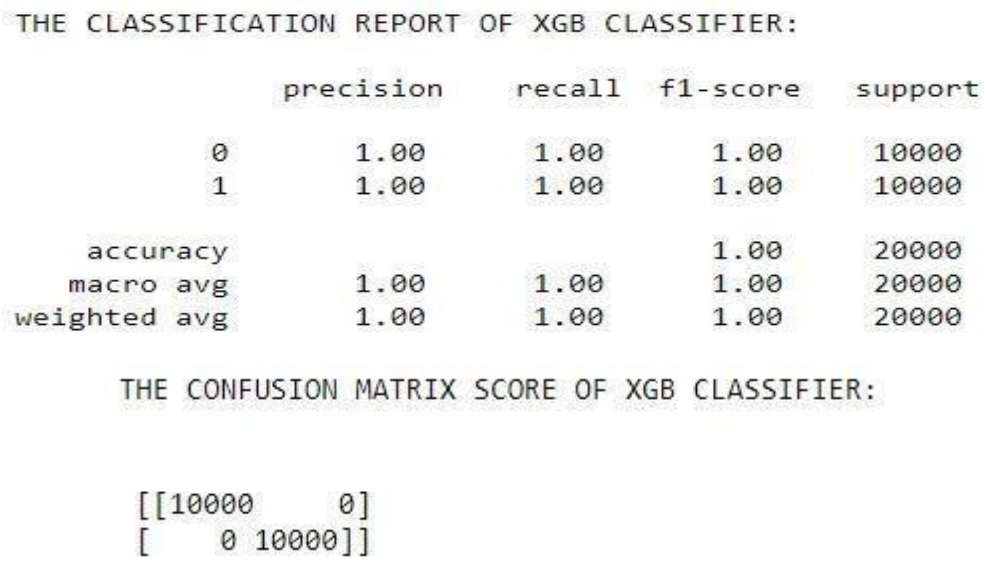
	precision	recall	f1-score	support
0	0.68	0.67	0.67	10000
1	0.67	0.68	0.68	10000
accuracy			0.68	20000
macro avg	0.68	0.68	0.68	20000
weighted avg	0.68	0.68	0.68	20000

THE CONFUSION MATRIX SCORE OF BERNOULLINB:

```
[[6681 3319]
[3166 6834]]
```

Figure 5.4.1 Classification Report and Confusion Matrix of Bernoullinb

The images depicts a classification report for a Bernoulli Naive Bayes model**.** The report includes precision, recall, f1-score, and support metrics for two classes (0 and 1). Overall, the model's performance appears fairly balanced across both classes, with values ranging from 0.67 to 0.68**.** It depicts a confusion matrix score . This matrix provides essential information for evaluating the performance of the classifier. It consists of a 2x2 grid, help assess the model's accuracy, precision, recall, and other metrics. The balanced distribution of values suggests a reasonably effective classification performance across both classes.


THE CLASSIFICATION REPORT OF XGB CLASSIFIER:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10000
1	1.00	1.00	1.00	10000
accuracy			1.00	20000
macro avg	1.00	1.00	1.00	20000
weighted avg	1.00	1.00	1.00	20000

THE CONFUSION MATRIX SCORE OF XGB CLASSIFIER:

```
[[10000     0]
[    0 10000]]
```

Figure 5.4.2 Classification Report and Confusion Matrix of XGB

The image depicts the classification report of an XGB Classifier. This report provides essential metrics for evaluating the classifier's performance on a dataset of 20,000 instances**.** Notably, the classifier achieves perfect scores in precision**,** recall, and F1-score for both classes (0 and 1). Additionally, the overall accuracy**,** macro average**,** and weighted average are all at a maximum value of 1.00**.** The image depicts the confusion matrix score of an XGB Classifier. This matrix provides essential information for evaluating the performance of the classifier. It consists of a 2x2 grid, help assess the model's accuracy, precision, recall, and other metrics.

THE CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10000
1	1.00	1.00	1.00	10000
accuracy			1.00	20000
macro avg	1.00	1.00	1.00	20000
weighted avg	1.00	1.00	1.00	20000

THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:

```
[[10000     0]
[    0 10000]]
```

Figure 5.4.3 Classification Report and Confusion Matrix of Random Forest

The image displays a classification report for a Random Forest Classifier. It includes metrics such as precision**,** recall**,** f1-score**,** and support for two classes (0 and 1). Both classes have perfect scores in precision, recall, and f1-score, indicating excellent model performance. The support for each class is 10,000 instances, resulting in an overall accuracy of 1.00 for 20,000 instances. The image displays the confusion matrix score of a Random Forest Classifier. It consists of a 2x2 grid, help assess the model's accuracy, precision, recall, and other metrics.

CHAPTER 6
CONCLUSION & FUTURE WORK

CHAPTER 6

CONCLUSION AND FUTURE WORK

Conclusion

In conclusion, predicting Android ransomware attacks through categorial classification emerges as a pivotal strategy for bolstering mobile security. By deploying advanced machine learning models, the system achieves early detection and risk assessment, prioritizing user protection against the ever-changing ransomware landscape. The model's adaptability to new threats and seamless integration into the Android security ecosystem underscore its proactive stance. As the mobile app environment evolves, the scalability of this approach proves essential, ensuring a resilient defense against emerging risks. Overall, the application of categorial classification in predicting Android ransomware marks a significant advancement in fortifying the platform against cyber threats, showcasing the efficacy of machine learning in safeguarding mobile ecosystems.

Future Work

Future work in the realm of predicting Android ransomware attacks using categorial classification should focus on refining the model's predictive accuracy through more sophisticated behavioral analysis and feature engineering. Incorporating real-time monitoring capabilities and automated response mechanisms would elevate the system's agility in countering threats promptly. Exploration of ensemble learning approaches, coupled with a commitment to explainability and interpretability, can enhance both the model's robustness and user trust. Continuous updates to the dataset, collaboration with cybersecurity communities for real-time threat intelligence,and user education features remain pivotal for ensuring the model's adaptability to thedynamic landscape of Android applications. Furthermore, cross-platform adaptation to other mobile operating systems would broaden the model's scope and contribute to a more comprehensive defense against ransomware threats across diverse platforms.

APPENDICES

A1. SDG: Goal 8(Decent Work and Economic Growth)

The Sustainable Development Goal 8 (SDG 8) focuses on promoting sustained, inclusive, and sustainable economic growth, full and productive employment, and decent work for all. It primarily addresses issues related to job creation, improving work conditions, and ensuring fair economic opportunities.

The connection between SDG 8 and predicting Android ransomware attacks using categorial classification might not be direct, but there could be indirect links. Effective cybersecurity measures and innovations can contribute to a stable and secure digital environment, which is essential for sustained economic growth. Here's how they might be connected:

1. Job Security in Tech Industry: he tech industry plays a crucial role in economic growth. By addressing cybersecurity threats like ransomware attacks, job security in the technology sector can be enhanced, contributing to the goals of decent work and economic growth.

2. Innovation and Economic Stability: A thriving tech industry fosters innovation, which is vital for economic stability. Predicting and preventing ransomware attacks on Android devices contribute to a secure digital environment, encouraging innovation and economic growth.

3. Preventing Disruptions: Ransomware attacks can disrupt businesses and economic activities. Effective prediction and prevention mechanisms help in avoiding such disruptions, promoting sustained economic growth as businesses can operate smoothly.

4. Tech Workforce Development: Addressing cybersecurity challenges requires a skilled workforce. By investing in training programs and education related to cybersecurity and Android security, SDG 8's goal of promoting full and productive employment can be indirectly supported.

A2.Source Code

Module-1: Data Preprocessing And Data Cleaning

In [ ]:

```
import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')
```

In [ ]:

```
df = pd.read_csv('MALWARE.csv')

del df['hash']

del df['normal_prio']
del df['policy']

del df['vm_pgoff']

del df['usage_counter']

del df['cgtime']

del df['signal_nvcsw']

del df['gtime']

del df['lock']

del df['hiwater_rss']

del df['task_size']

del df['cached_hole_size']

del df['nr_ptes']df.head()
```

In [ ]:

```
df.tail()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df.size
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
var = ['classification']
for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.isnull()
```

In [ ]:

```
df = df.dropna()
```

In [ ]:

```
df['classification'].unique()
```

In [ ]:

```
df.describe()
```

In [ ]:

```
df.corr()
```

In [ ]:

df.info()

In [ ]:

pd.crosstab(df["min_flt"], df["last_interval"])

In [ ]:

df.groupby(["last_interval","shared_vm"]).groups

In [ ]:

df["classification"].value_counts()

In [ ]:

pd.Categorial(df["fs_excl_counter"]).describe()

In [ ]:

df.duplicated()

In [ ]:

sum(df.duplicated())

In [ ]:

In [ ]:

Module-2 Data Visualization And Data Analysis

In [ ]:

import pandas **as** pd

import numpy **as** np

import matplotlib.pyplot **as** plt

import seaborn **as** sns

In [ ]:

```
df = pd.read_csv('MALWARE.csv')

del df['hash']

del df['normal_prio']

del df['policy']

del df['vm_pgoff']

del df['usage_counter']

del df['cgtime']

del df['signal_nvcsw']

del df['gtime']

del df['lock']

del df['hiwater_rss']

del df['task_size']

del df['cached_hole_size']

del df['nr_ptes']

df.head()
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['classification']

for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
plt.figure(figsize=(12,7))
sns.countplot(x='classification',data=df)
```

In [ ]:

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.hist(df['nvcsw'],color='red')
plt.subplot(1,2,2)
plt.hist(df['reserved_vm'],color='blue')
```

In [ ]:

```
df.hist(figsize=(15,55),layout=(15,4), color='green')
plt.show()
```

In [ ]:

```
df['stime'].hist(figsize=(10,5),color='yellow')
```

In [ ]:

```
sns.lineplot(df['end_data'], color='brown') # scatter, plot, triplot, stackplot
```

In [ ]:

```
sns.violinplot(df['total_vm'], color='purple')
```

In [ ]:

```
df['exec_vm'].plot(kind='density')
```

In [ ]:

```
sns.displot(df['total_vm'], color='purple')
```

 # barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot, kdeplot, pointplot, violinplot, stripplot

In [ ]:

```
sns.displot(df['shared_vm'], color='coral') # residplot, scatterplot
```

In [ ]:

```
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
```

In [ ]:

```
def plot(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)
plot(df, 'classification')
```

Module-3 XG BOOST Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
df = pd.read_csv('MALWARE.csv')
```

```
del df['hash']

del df['normal_prio']

del df['policy']

del df['vm_pgoff']

del df['usage_counter']

del df['cgtime']

del df['signal_nvcsw']

del df['gtime']

del df['lock']

del df['hiwater_rss']

del df['task_size']

del df['cached_hole_size']

del df['nr_ptes']

df.head()

In [ ]:

df=df.dropna()

In [ ]:

df.columns

In [ ]:

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['classification']

for i in var:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.head()
```

In [ ]:

```
x1 = df.drop(labels='classification', axis=1)
y1 = df.loc[:,'classification']
```

In [ ]:

```
import imblearn

from imblearn.over_sampling import RandomOverSampler

from collections import Counter

ros =RandomOverSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print("OUR DATASET COUNT         : ", Counter(y1))

print("OVER SAMPLING DATA COUNT : ", Counter(y))
```

In [ ]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42,
stratify=y)

print("NUMBER OF TRAIN DATASET     : ", len(x_train))

print("NUMBER OF TEST DATASET      : ", len(x_test))

print("TOTAL NUMBER OF DATASET     : ", len(x_train)+len(x_test))
```

In [ ]:

```
print("NUMBER OF TRAIN DATASET     : ", len(y_train))

print("NUMBER OF TEST DATASET      : ", len(y_test))
```

```
print("TOTAL NUMBER OF DATASET    : ", len(y_train)+len(y_test))
```

In [ ]:

```
from sklearn.naive_bayes import BernoulliNB
```

In [ ]:

```
BNB = BernoulliNB()
```

```
BNB.fit(x_train,y_train)
```

In [ ]:

```
predicted = BNB.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import classification_report
```

```
cr = classification_report(y_test,predicted)
```

```
print('THE CLASSIFICATION REPORT OF BERNOULLINB:\n\n',cr)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test,predicted)
```

```
print('THE CONFUSION MATRIX SCORE OF BERNOULLINB:\n\n\n',cm)
```

In [ ]:

```
from sklearn.model_selection import cross_val_score
```

```
accuracy = cross_val_score(BNB, x, y, scoring='accuracy')
```

```
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy*100)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
```

```
a = accuracy_score(y_test,predicted)
```

```
print("THE ACCURACY SCORE OF BERNOULLINB IS :",a*100)
```

In [ ]:

```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF BERNOULLINB IS :",hl*100)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF BERNOULLINB\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()


cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF BERNOULLINB:\n\n')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
```

```
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')

plt.plot(df2["y_test"][:100],  marker='o', linestyle='dashed', color='green')

plt.show()
```

Module-4 RandomForest Algorthim

In [ ]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

pd.set_option('display.max_columns', None)

import warnings

warnings.filterwarnings('ignore')
```

In [ ]:

```
df = pd.read_csv('MALWARE.csv')

del df['hash']

del df['normal_prio']

del df['policy']

del df['vm_pgoff']

del df['usage_counter']

del df['cgtime']

del df['signal_nvcsw']

del df['gtime']

del df['lock']
```

```
del df['hiwater_rss']

del df['task_size']

del df['cached_hole_size']

del df['nr_ptes']

del df['state']

df.head()
```

In [ ]:

```
df.columns
```

In [ ]:

```
df=df.dropna()
```

In [ ]:

```
df.columns
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['classification']

for i in var:

    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.head()
```

In [ ]:

```
x1 = df.drop(labels='classification', axis=1)

y1 = df.loc[:,'classification']
```

In [ ]:

x1

In [ ]:

```
# import imblearn
# from imblearn.over_sampling import RandomOverSampler
# from collections import Counter
# ros =RandomOverSampler(random_state=42)
# x,y=ros.fit_resample(x1,y1)
# print("OUR DATASET COUNT        : ", Counter(y1))
# print("OVER SAMPLING DATA COUNT  : ", Counter(y))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=0.20, random_state=42, stratify=y1)
print("NUMBER OF TRAIN DATASET    : ", len(x_train))

print("NUMBER OF TEST DATASET     : ", len(x_test))
print("TOTAL NUMBER OF DATASET    : ", len(x_train)+len(x_test))
```

In [ ]:

```
print("NUMBER OF TRAIN DATASET    : ", len(y_train))
print("NUMBER OF TEST DATASET     : ", len(y_test))
print("TOTAL NUMBER OF DATASET    : ", len(y_train)+len(y_test))
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:

```
RFC = RandomForestClassifier(random_state=42)

RFC.fit(x_train,y_train)
```

In [ ]:

```
predicted = RFC.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import classification_report

cr = classification_report(y_test,predicted)

print('THE CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER:\n\n',cr)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predicted)

print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n\n',cm)
```

In [ ]:

```
from sklearn.model_selection import cross_val_score

accuracy = cross_val_score(RFC, x1, y1, scoring='accuracy')

print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy*100)
```

In [ ]:

```
from sklearn.metrics import accuracy_score

a = accuracy_score(y_test,predicted)

print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS
```

:",a*100)

In [ ]:

```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",hl*100)
```

In [ ]:

```
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
```

```
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')

plt.show()
```

In [ ]:

```
import joblib

joblib.dump(RFC, 'MALWARE.pkl')
```

Module-5 Bernoulin Navie Bayes Algorithm

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')
```

In [ ]:

```
df = pd.read_csv('MALWARE.csv')

del df['hash']

del df['normal_prio']

del df['policy']

del df['vm_pgoff']

del df['usage_counter']

del df['cgtime']

del df['signal_nvcsw']

del df['gtime']

del df['lock']
```

```
del df['hiwater_rss']

del df['task_size']

del df['cached_hole_size']

del df['nr_ptes']

df.head()

In [ ]:

df=df.dropna()

In [ ]:

df.columns

In [ ]:

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['classification']

for i in var:

    df[i] = le.fit_transform(df[i]).astype(int)

In [ ]:

df.head()

In [ ]:

x1 = df.drop(labels='classification', axis=1)

y1 = df.loc[:,'classification']

In [ ]:

import imblearn

from imblearn.over_sampling import RandomOverSampler
```

```
from collections import Counter


ros =RandomOverSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print("OUR DATASET COUNT         : ", Counter(y1))

print("OVER SAMPLING DATA COUNT : ", Counter(y))
```

In [ ]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42,
stratify=y)

print("NUMBER OF TRAIN DATASET     : ", len(x_train))

print("NUMBER OF TEST DATASET      : ", len(x_test))

print("TOTAL NUMBER OF DATASET     : ", len(x_train)+len(x_test))
```

In [ ]:

```
print("NUMBER OF TRAIN DATASET     : ", len(y_train))

print("NUMBER OF TEST DATASET      : ", len(y_test))

print("TOTAL NUMBER OF DATASET     : ", len(y_train)+len(y_test))
```

In [ ]:

```
from xgboost import XGBClassifier
```

In [ ]:

```
XGB = XGBClassifier()

XGB.fit(x_train,y_train)
```

In [ ]:

```
predicted = XGB.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF XGB CLASSIFIER:\n\n',cr)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF XGB CLASSIFIER:\n\n\n',cm)
```

In [ ]:

```
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(XGB, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n', accuracy*100)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF XGB CLASSIFIER IS :",a*100)
```

In [ ]:

```
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF XGB CLASSIFIER IS :",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
```

```
RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')
plt.show()
```

In [ ]:

A3 Sample Screenshots

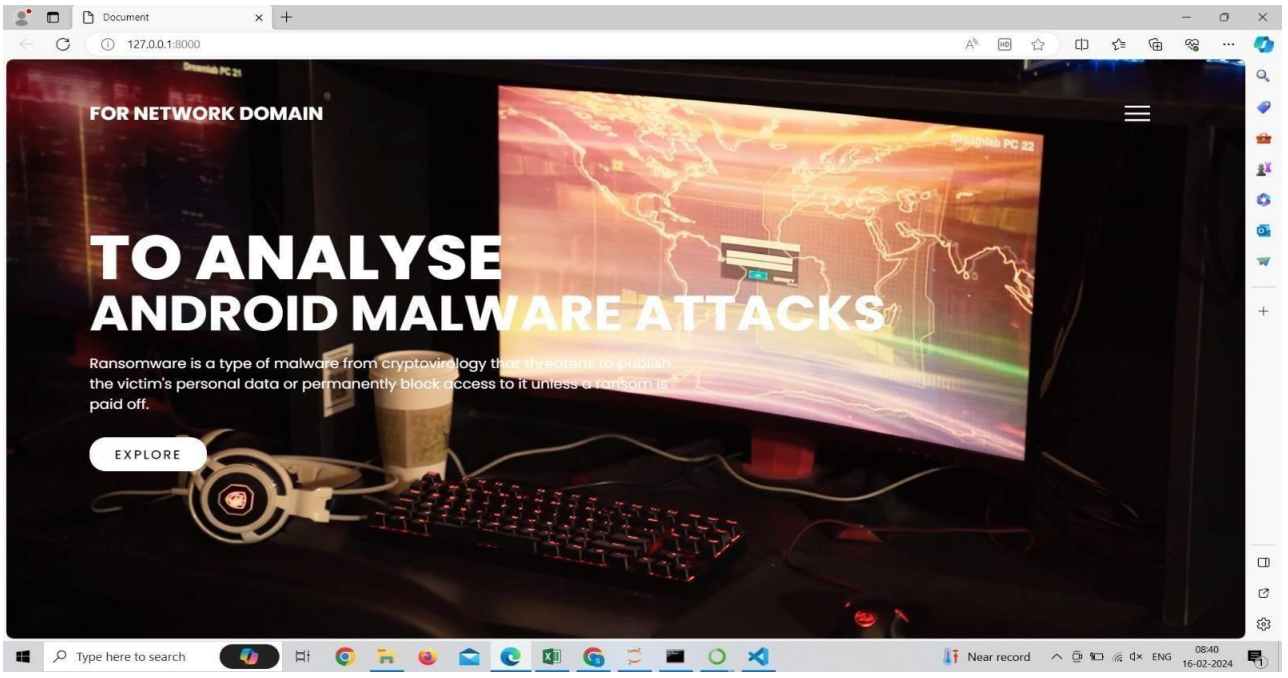


Figure A.1.1 Home Page

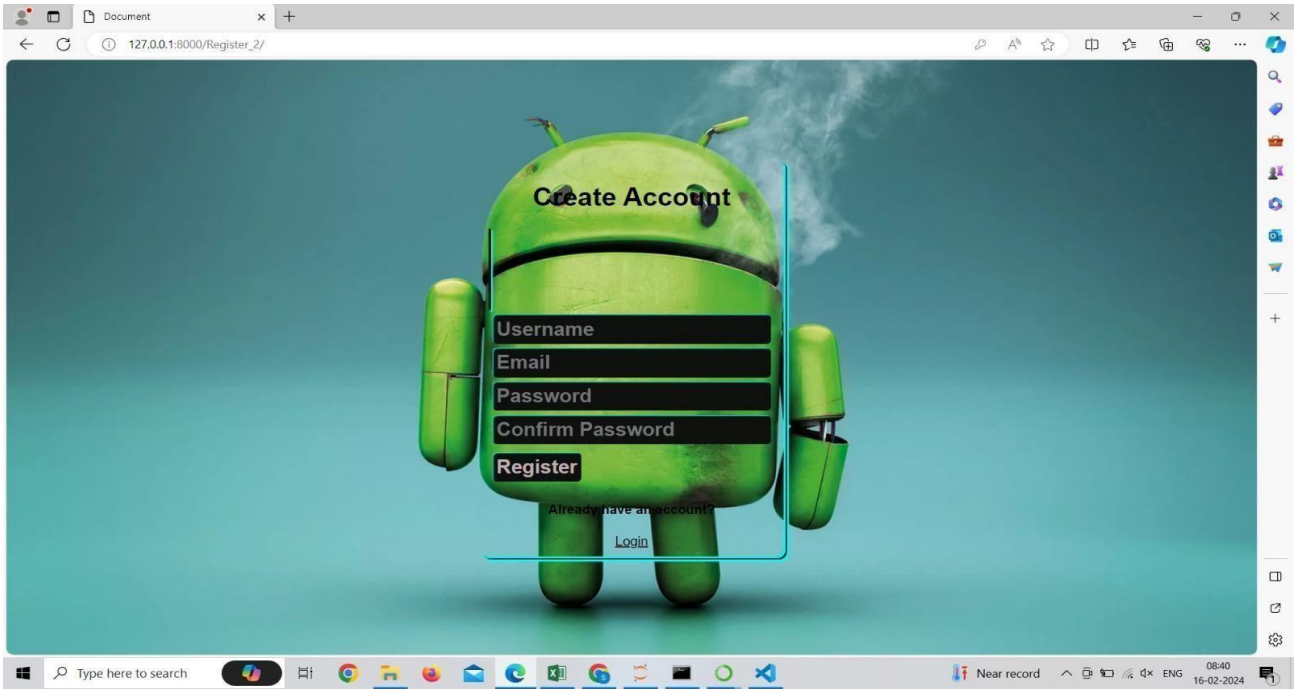


Figure A.1.2 Create Account Page

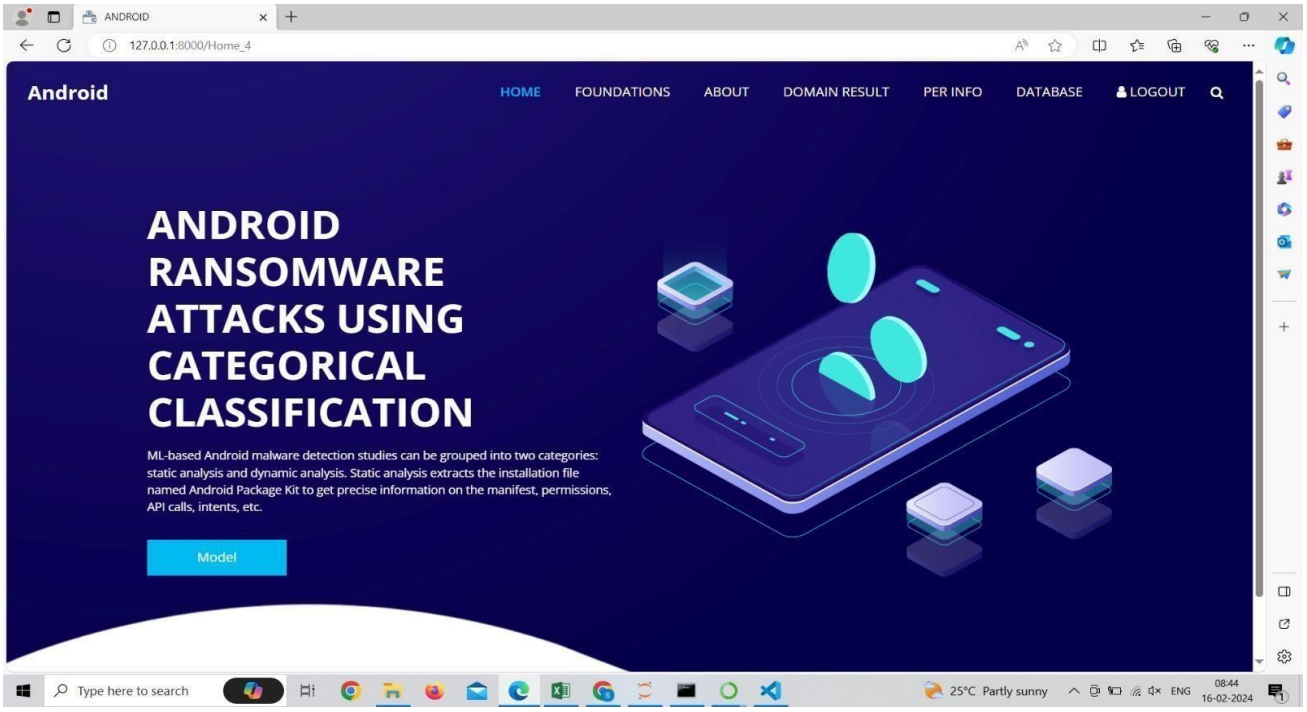


Figure A.1.3 Introduction Page

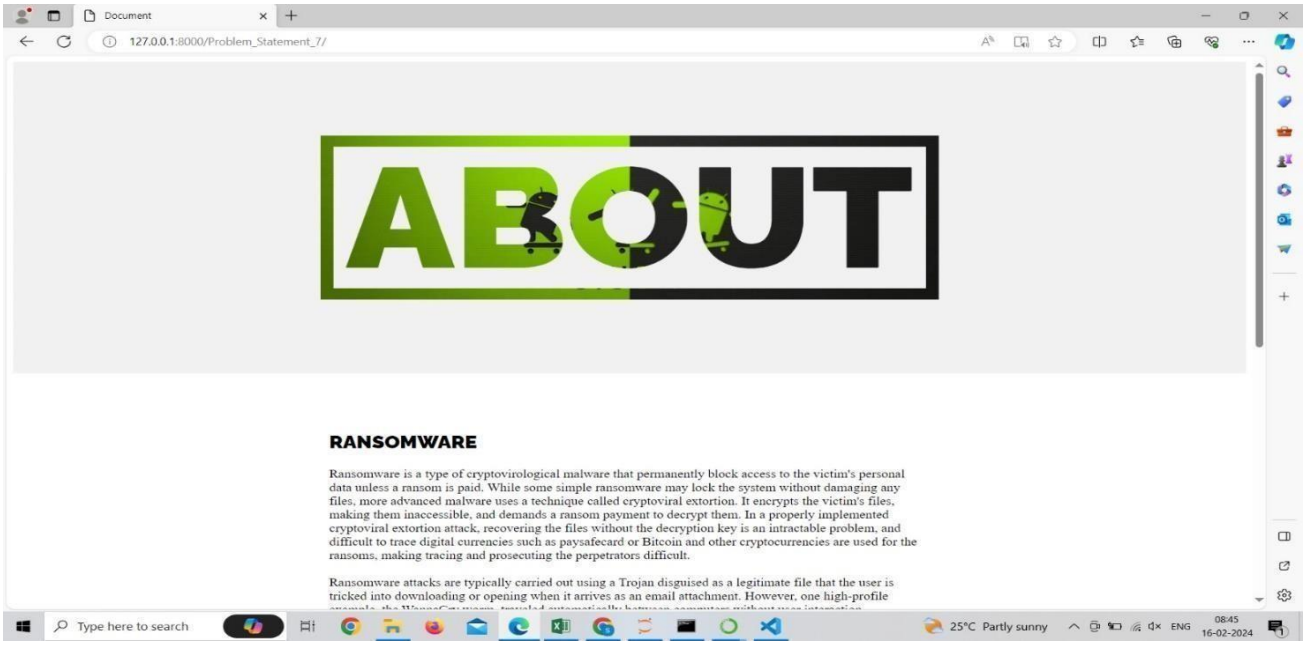


Figure A.1.4 About Page

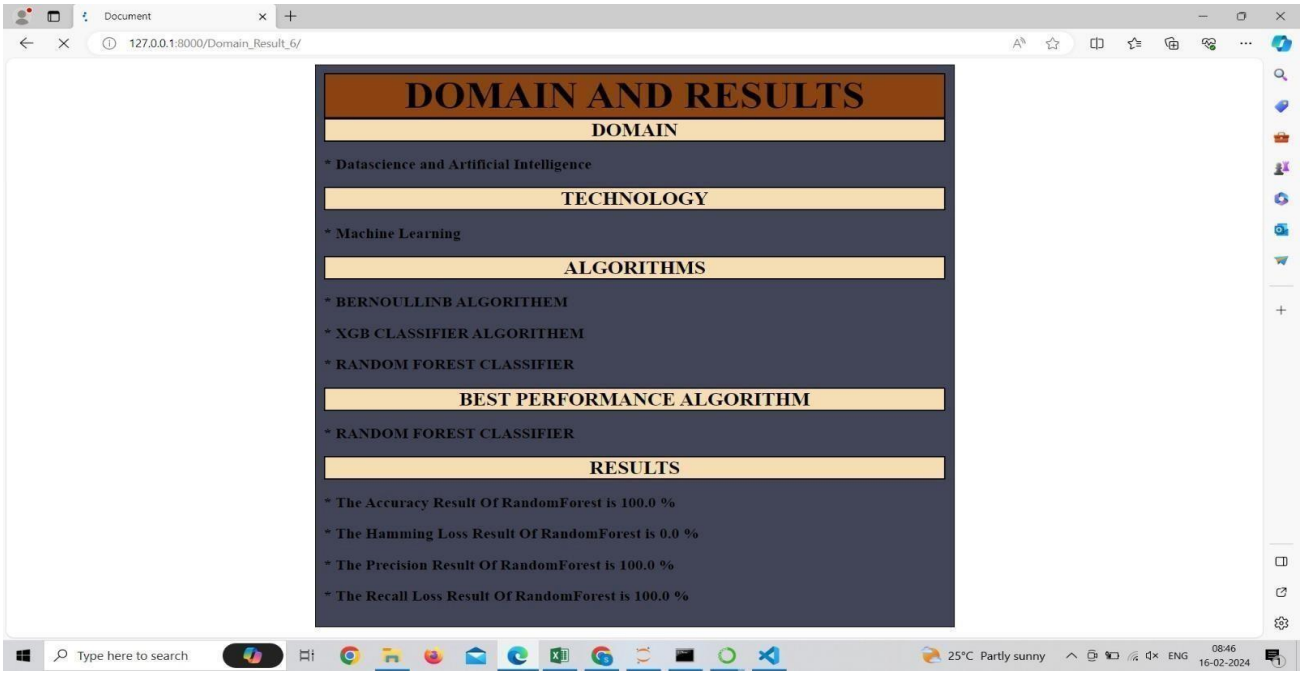


Figure A.1.5 Domain and Results Page

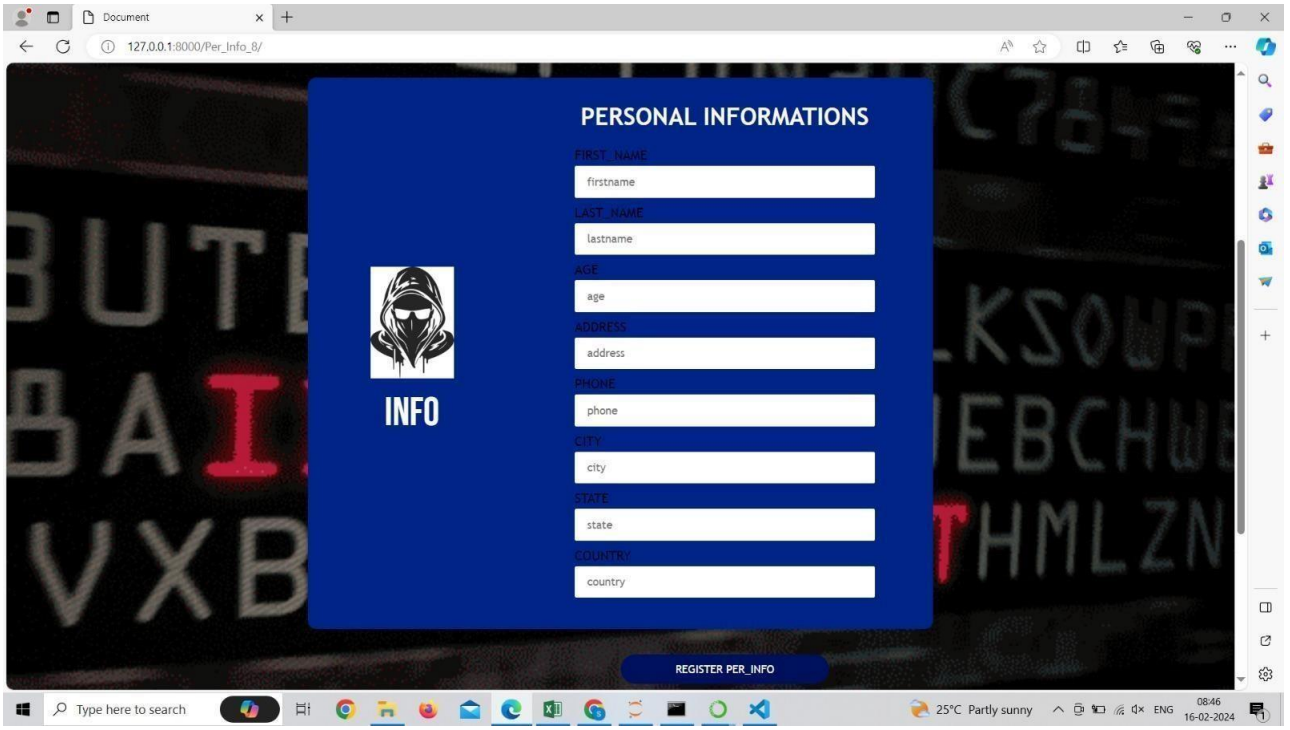


Figure A.1.6 Personal Information Page

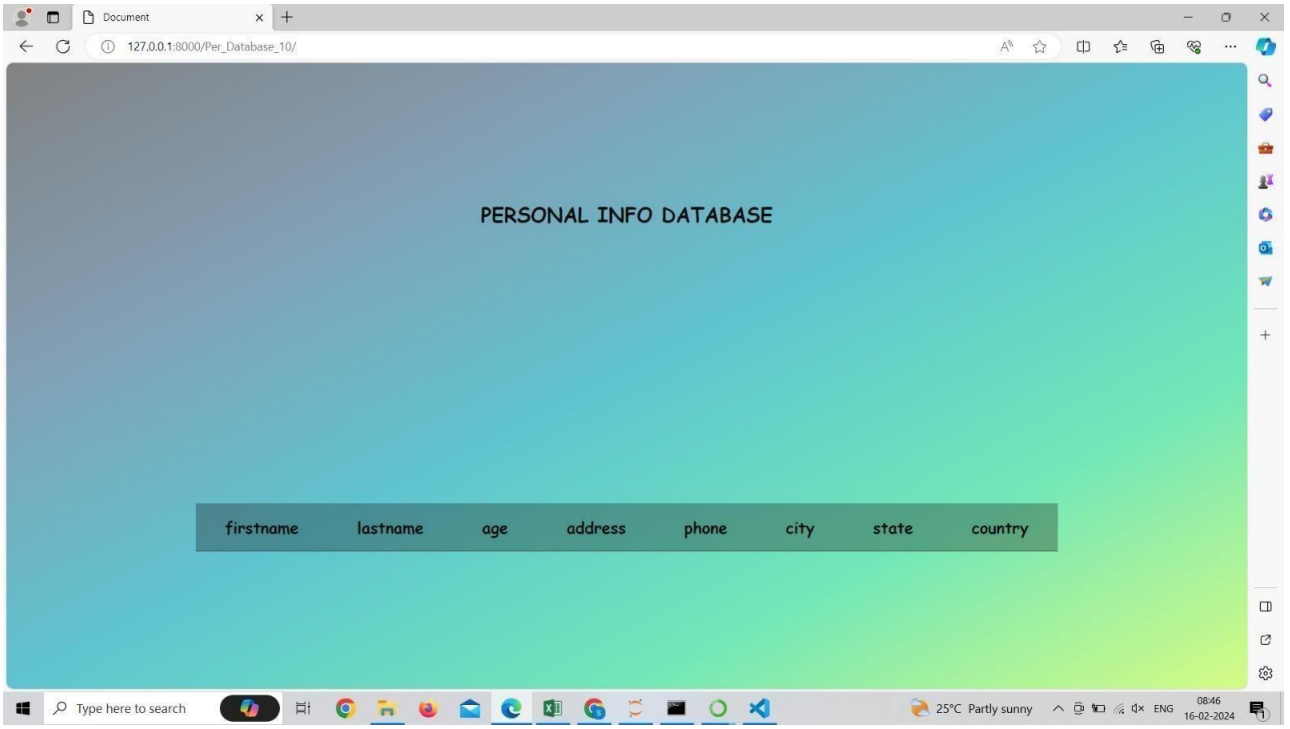


Figure A.1.7 Personal Information Database Page

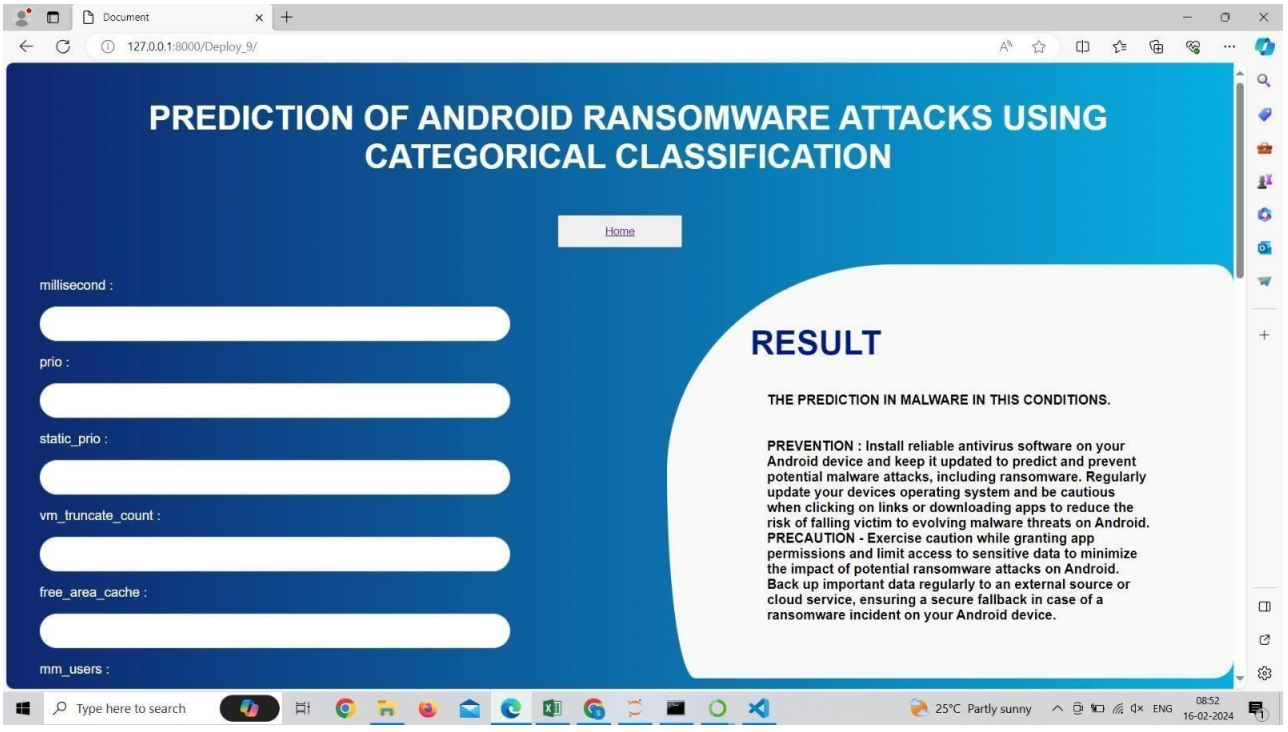


Figure A.1.8 Results Page

A4 Plagiarism Report

Plagiarism Checker X - Report

Originality Assessment

13%

Overall Similarity

Date: Mar 12, 2024
Matches: 615 / 4567 words
Sources: 29

Remarks: Moderate similarity detected, consider enhancing the document if necessary.

Verify Report:
Scan this QR Code

v 9.0.2 - WML 4
FILE - PAPER ARA.PDF

Prediction of Android Ransomeware Attack using Categorical Classification

Dr.K.Sangeetha[1] (Associate Professor) Computer Science and Engineering Panimalar Engineering College Chennai ksangeetha@panimalar.ac.in R.Abirami[2] Computer Science and Engineering Panimalar Engineering College Kancheepuram abiramiravikumar6@gmail.com R.Kalai Arasi[2] 9 Computer Science and Engineering Panimalar Engineering College Chengalpattu kalaiarasiravi24@gmail.com M.Muthu Usha Rani[2] Computer Science and Engineering Panimalar Engineering College Chennai muthuusharani13@gmail.com Abstract—Android ransomware has become a significant cybersecurity threat, posing serious risks to mobile devices and users' data. To effectively combat this growing menace, predictive models that can identify and thwart ransomware attacks in real-time are crucial. This research presents a novel approach for predicting Android ransomware attacks using categorical classification techniques. Data pre-processing is performed to extract and transform essential features from Android applications. Subsequently, the categorical classification model is trained on a labelled dataset comprising both benign and ransomware applications. The research findings demonstrate the efficacy of the proposed approach in predicting Android ransomware attacks with high accuracy. The model shows promising results in identifying malicious applications, thus providing a proactive defense against ransomware threats on Android devices. By implementing categorical classification techniques, the proposed system enables timely identification and mitigation of potential threats, contributing to a safer and more secure mobile ecosystem. Keywords: Android ransomware, Cybersecurity, Predictive modelling, Categorical classification, Mobile devices, Data preprocessing, Android applications, Permissions, API calls, Benign applications, Malicious applications I. INTRODUCTION One of the biggest cybersecurity risks that businesses are currently experiencing is the quick spread of ransomware attacks. Ransomware has grown in

popularity in recent years as a means for hackers to demand money from victims by encrypting their data and requesting payment for a decryption key. Ransomware attacks have affected every sector of the economy, including government, education, healthcare, and finance. Research in this field is extremely important because of the high stakes involved. It is imperative to comprehend the nature of ransomware attacks, how they propagate, and the possible repercussions of becoming a victim of one. As the danger of ransomware attacks increases, researchers and industry professionals must do more to understand the issue and provide practical solutions for mitigation and prevention. This study seeks to support this endeavor by offering a thorough synopsis of the ransomware threat landscape, examining the elements that facilitate malware propagation, and investigating future research directions. By bringing this important topic to light, we seek to reduce the potential harm that these harmful programs could do and assist people and businesses in better defending themselves against ransomware assaults. This essay presents the idea of ransomware and explains how it operates. It is structured as follows. It also covers the many kinds of ransomware attacks, including scareware, locker ransomware, and encrypting ransomware. explains the approach taken to write this study. provides analyses of ransomware detection methods built by researchers using machine learning. It talks about the approaches taken, the results obtained, and the shortcomings of each system. It also covers the difficulties in gathering and preparing data for machine learning-based ransomware detection. It offers a thorough examination of how ransomware has changed over the past twelve years gives a summary of the many ransomware detection methods now in use, such as behavior-based, machine-learning, and signature-based methods. Additionally, it talks about the many assessment criteria that are employed to gauge how well machine learning models identify ransomware. It also emphasizes malware detection through machine learning techniques. It talks about the many machine learning algorithms-like decision trees, random forests, support vector machines, and neural networks-that are employed for this. It talks about the various characteristics and feature selection methods used in machine learning-based ransomware

detection. explains the difficulties in creating malware detection systems based on machine learning. It also outlines potential paths forward in this area, including tackling adversarial attacks, adding real-time detection capabilities, and creating models that are more reliable and accurate. summarizes all the findings of this investigation. Researchers and industry professionals interested in creating efficient machinelearning-based ransomware detection systems will find this research to be a useful resource. Fig. 1. System Architecture.

Fig. 2. Work-Flow Diagram. II. LITERATURE SURVEY The research that used ML and DL algorithms for Android ransomware detection between 2017 and 2022 is presented in this section of the literature. Most of the studies deployed various ML techniques. Firstly, Manuela Horduna proposed that [23] Ransomware is a type of malware that uses encryption to save the data of its victims, resulting in financial incentives for individuals or businesses as well as irreversible loss. Six The frequency of ransomware assaults has been rising sharply, and as attackers are becoming more resourceful and creative in their threats, combating these time- and energy-consuming, malicious actions has gotten more challenging. In order to detect this kind of malware, recent study attempts to provide some insight into the combination of machine learning and defense measures. Machine learning is widely employed in advanced cybersecurity research and helps anti-ransomware systems become more accurate at anticipating the results or behaviours of the attacks. In this paper, we examine how machine learning might strengthen malware detection to fend off serious security threats. We provide a succinct yet thorough summary of this burgeoning field. In addition, we provide a quick overview of the key ransomware attack incidents for 2022 together with information on the requested ransoms. Mazen Gazzan and Frederick T.Sheldon proposed that Industrial control systems (ICS) and supervisory control and data acquisition (SCADA) systems, which control critical infrastructure such as power plants and water treatment facilities, have unique traits that render them susceptible to ransomware assaults. Because they frequently use proprietary software and are out-of-

date, these systems are challenging to secure using conventional cybersecurity techniques. Predicting and detecting ransomware attacks early on is extremely difficult due to restricted access to these systems and insufficient threat intelligence. Attacks using ransomware against SCADA and ICS systems have grown in importance in the last several years. These assaults possess the capacity to drastically damage important infrastructure and cause sizable financial losses. The likelihood 3 of ransomware attacks on industrial control systems is still very high, despite the growing threat. a difficulty for the cybersecurity community. This is because of the special features of these systems, which include their limited operational visibility and usage of proprietary software. This review article will look at the difficulties in anticipating 3 ransomware attacks on industrial systems as well as the methods already in use to reduce the hazards involved. We will also talk about how a multidisciplinary strategy that closely coordinates the efforts of the cybersecurity and ICS communities is necessary. Our goals are to give a thorough assessment of the level of ransomware prediction on industrial systems today and to point out areas that could use more research and improvement in the future. Keywords: attack probability prediction, industrial control systems, SCADA, ransomware detection and prevention, situation awareness, and security evaluation. Qingyuan Gong proposed that Numerous developers can work together thanks to online communities like GitHub. However, because of their open nature, the communities are susceptible to various harmful assaults, as it is simple for attackers to join and engage in conversation with authorized members. Our proposal, called GitSec, is a deep learning-based method for identifying rogue accounts in developer communities on the internet. GitSec uses social interactions, dynamic activity features, and account profiles to differentiate between fraudulent and legitimate accounts. GitSec first presents 19 two user activity sequences, which are then processed using a parallel neural network architecture that incorporates an attention mechanism. Secondly, based on repository operations, GitSec builds two graphs to illustrate how users interact with one another. In particular, structural hole theory and graph neural networks are used to handle the two created graphs Third, GitSec leverages the

descriptive information to improve the performance of detection. A decision maker powered by a classifier based on supervised machine learning makes the ultimate decision. Our thorough analyses, which are based on real-world data from GitHub users, demonstrate that GitSec outperforms cutting-edge alternatives with an AUC value of 0.916. Index Terms— Social networks, Deep Learning, Graph Neural Networks, Online Developer Communities, Structural Hole Theory, and Malicious Account Detection. Shubham Shakya proposed that Android has been more and more popular over the past ten years, both with consumers and with attackers. Attackers on Android are drawn to the large number of Android users. Our detection techniques should require an update as well, given the ongoing expansion of Android malware's variety and assault techniques. Most of the researcher's works are based on static features, and very few focus on dynamic features. By employing System calls to identify Android malware, we are addressing a gap in the literature with our paper. In order to detect malware, we are employing an emulator to execute the harmful app in a watchful and regulated setting. During its execution, malicious activity is triggered by a few simulated events to initiate its hostile conduct. For the purpose of detecting and classifying malware families, logs gathered during the application's runtime are examined and input into many machine learning models. The outcome shows that the Decision Tree and K-Nearest Neighbor

algorithms provided the best accuracy in family classification and malware detection, respectively. Kateryna Chumachenko proposed that An essential component of computer system security is malware detection. Unfortunately, polymorphic viruses and zeroday assaults cannot be accurately detected by the signaturebased techniques that are currently in use. For this reason, machine learning-based detection becomes necessary. The goal of this work was to identify the most effective feature extraction, feature representation, and classification techniques that, when applied on top of In particular, Random Forest, Naive Bayes, Decision Trees, Support Vector Machines, and k-Nearest-neighbors classifiers were assessed. The dataset used for this study consistsed of the

1156 malware files of 9 families of different types and 984 benign files of various formats. This paper offers suggested techniques for malware detection and classification based on machine learning, along with implementation guidance. Additionally, the work conducted can serve as a foundation for future research using machine learning techniques for malware analysis III. METHODOLOGY A. Materials and Methods This study used a very current dataset to analyse and classify network traffic, classifying it as benign or ransomware in order to solve the shortcomings in the existing literature. To attain optimal performance, valuable features were retrieved from the dataset. The following key processes were part of the consistent approach used for this study's execution: data collection, pre-processing, classification, and evaluation. As soon as we obtained the dataset, we examined and analysed it to determine what traits were there. Following that, it was evident what preprocessing measures needed to be taken in order to raise the data's quality and boost the models' dependability and performance. The pre-processing procedures comprised feature selection utilizing forward feature selection and feature importance approaches, under-sampling, binarizing the labels, and transforming categorical features to numerical. Fig. 3. Process of Data-Flow Diagram. B. Dataset Description The dataset used in this study is named Android Ransomware Detection and is publicly accessible on Kaggle. It was gathered to identify ransomware assaults in Android networks through the use of machine learning algorithms. To the best of our knowledge, the dataset is new and has never been used Consequently, there isn't any benchmark work on the dataset to take into account. There are 100,000 records in the collection. Furthermore, there are no missing values in the dataset. About 50,000 of the 1,000 000 data represent benign traffic. Fig. 4. Dataset from Kaggle. C. Data Pre-Processing Pre-processing Data The error rate of the Machine Learning (ML) model is obtained using validation approaches, which are thought to be as near to You might not require the validation approaches if the volume of data is sufficiently large to be representative of the population. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. Identifying

duplicate values, missing values, and the data type-integer or float-must be done. the data sample that is utilized to tune model hyperparameters while offering an objective assessment of a model fit on the training dataset. The 2 evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. This data is used by machine learning developers to fine-tune the model's hyperparameters. A time-consuming to-do list may result from data gathering, analysis, and the process of addressing data content, quality, and structure. Understanding your data and its characteristics is helpful during the data identification process; this knowledge will assist you decide which algorithm to employ to design your a variety of data cleaning tasks with Python's Pandas library; in particular, it focuses on missing values, which is perhaps the largest data cleaning task, and it may more

It would rather spend more time investigating and modeling than cleaning data. Fig. 5. Module diagram for Pre-Processing. Fig. 6. Removing the null values. D. Data Validation Importing the library packages with loading given dataset. Analyzing the variable identification by data type and shape, assessing duplicate and missing values, etc. 2 A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to examine the single-, double-, and multi-variate processes. Data cleaning procedures and methods differ depending on the type of dataset. Finding and eliminating mistakes and abnormalities is the main objective of data cleaning, which aims to improve the value of data for analytics and decision-making. Given Input and Expected Output: Input: Data Output: Removing noisy data E. Data Visualization and Data Analysis 1 Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. An essential set of tools for developing a

qualitative understanding is provided by data visualization. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. Data visualizations, when combined with a little topic expertise, 5 can be utilized to convey and illustrate important correlations in charts and plots that are more visceral and meaningful than measurements of It will be suggested to delve deeper into some of the books indicated at the end, as 1 data visualization and exploratory data analysis are entire topics in and of themselves. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. In applied statistics as well as 2 applied machine learning, the ability to visualize data samples and other types of information rapidly is crucial. It will discover the many 1 types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data. • How to use bar charts to represent categorical values and line plots to represent time series data. • How to use box plots and histograms to summarize data distributions. Fig. 7. Module Diagram for Visualization. Given Input and Expected Output: Input: Data Output: Visualized data Fig. 8. Histogram Graph.

Fig. 9. Classification. Fig. 10. Heat Map. F. Bernoulli NB Algorithm The 14 Bernoulli Naive Bayes (Bernoulli NB) algorithm is a classification algorithm based on the principles of the Naive Bayes theorem. It is specifically designed for binary classification problems, where the target variable has two possible classes (e.g., spam or not, positive or negative sentiment). Here's an explanation of how the Bernoulli Naive Bayes algorithm works: 1) Naive Bayes Classification: Bernoulli NB is a variant 14 of the Naive Bayes algorithm. Naive Bayes methods are based on Bayes' theorem, which is a probabilistic theorem used for classification and other machine learning tasks. It calculates the probability of a data point belonging to a particular class 20 based on its features. 2) Binary Features: Bernoulli NB is 16 well-suited for datasets where the features are binary, meaning they take on values of 0 or 1. These binary features 29 are often used to represent the presence (1) or absence (0) of certain attributes or characteristics in the data. 3) Assumption of

Independence: Like all Naive Bayes algorithms, Bernoulli NB makes a "naive" assumption of feature independence. It assumes that the presence or absence of one feature is independent of the presence or absence of any other feature. This simplifying assumption allows the algorithm to calculate probabilities more efficiently. 4) Decision Rule: The class with the highest posterior probability is the predicted class for the data point. In binary classification, you choose the class with the higher probability as the predicted class (e.g., if P(Y=1|X) > P(Y=0|X), then the prediction is class 1). 5) Laplace Smoothing: To avoid zero probabilities in cases where a feature hasn't been observed in a particular class during training, Laplace smoothing (or add-one smoothing) is often applied. This involves adding a small constant to all the probabilities, ensuring that no probability becomes zero. Fig. 11. Module diagram for BernoulliNB. Given Input and Expected Output: Input: Data Output: Getting Accuracy Fig. 12. Confusion matrix score of Bernoulli. Fig. 13. Data-frame of bernoulli. G. XG Boost Algorithm XG Boost, which stands for eXtreme Gradient Boosting, is a powerful and widely used machine learning algorithm known for its efficiency and effectiveness in various tasks, including classification, regression, and ranking. It belongs to the family of gradient boosting

algorithms and is particularly popular in data science competitions and real-world applications. Here are key features and characteristics of the XG Boost algorithm: 1) Gradient Boosting Framework: XG Boost is based on the gradient boosting framework, which builds an ensemble of weak learners (typically decision trees) sequentially, with each new tree addressing the errors made by the previous ones. 2) Regularization Techniques: It incorporates L1 (LASSO) and L2 (ridge) regularization terms in the objective function, which helps prevent overfitting and improves the model's generalization to new, unseen data. 3) Handling Missing Data: XG Boost has the ability to handle missing data internally. During the training process, it makes decisions on missing values and includes them in the tree construction. 4) Parallel and Distributed Computing: XG Boost is designed for efficient parallel and distributed computing, making it scalable and

capable of handling large datasets. This is particularly beneficial in scenarios where training on a single machine may be impractical. 5) Feature Importance: The algorithm provides a mechanism to assess the importance of each feature in the dataset. This feature importance analysis aids in understanding the contribution of different variables to the predictive performance of the model. 6) Tree Pruning: XG Boost employs a process of tree pruning during the construction of decision trees, which helps prevent the development of overly complex trees that may lead to overfitting. 7) Cross-Validation: Cross-validation is often used in conjunction with XG Boost to assess the model's performance and tune hyperparameters effectively, ensuring robustness and preventing overfitting. 8) Regular Updates and Maintenance: XG Boost is actively maintained and receives updates from the opensource community, ensuring ongoing improvements, bug fixes, and compatibility with the latest technological advancements. 9) Wide Applicability: XG Boost is versatile and can be applied to various machine learning tasks, such as classification, regression, ranking, and anomaly detection, making it a popular choice across different domains. 10) Integration with Other Tools: XG Boost can be easily integrated with popular machine learning libraries and frameworks, such as scikit-learn, making it accessible and convenient for users who are already familiar with these tools. Fig. 14. Module diagram for XGBoost. Given Input and Expected Output: Input: Data Output: Getting Accuracy Fig. 15. Confusion matrix score of XGBoost. Fig. 16. Data-frame of XGBoost. H. Random Forest Classifier Algorithm A Random Forest Classifier is a powerful machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines the predictions of multiple decision trees to produce a more accurate and robust model. Here's an explanation of how the Random Forest Classifier works: 1) Decision Trees: At the heart of the Random Forest Classifier are decision trees. A decision tree is a simple tree-like structure that makes a sequence of binary decisions to classify data points. Each internal node of the tree represents a decision based on a specific feature, and each leaf node represents a class label or a numerical value (in the case of regression). 2) Ensemble

Learning: A Random Forest consists of a collection of decision trees. Instead of relying on a single decision tree for making predictions, the algorithm builds multiple trees, each with a slightly different subset of the training data and features. 3) Random Subsampling: When building each decision tree in the forest, a random subset of the training data is used. The terms "bootstrap sampling" and "bagging" refer to this procedure. It lessens the possibility of overfitting to the training set by introducing variation among the individual trees. 4) Feature Randomization: Another key aspect of Random Forest is feature randomization. When making decisions at each node of a decision tree, the algorithm doesn't consider all available features. Instead, it randomly

selects a subset of features to consider. This introduces randomness and decorrelates the trees, making them less likely to make the same errors. 5) Voting or Averaging: Once all the individual decision trees are constructed, they can be used to make predictions. For classification tasks, the Random Forest combines the predictions of each tree through a majority vote. The final prediction is the class that gets the most votes overall. For regression tasks, the Random Forest averages the predictions of all trees to produce the final prediction. 6) Robustness: Random Forests are known for their robustness and ability to handle noisy or complex datasets. They are less prone to overfitting than individual decision trees because the ensemble of trees helps reduce variance. 7) Hyperparameters: Random Forests have several hyperparameters that can be tuned to optimize their performance, including: • The number of trees in the forest. • The maximum depth of each tree. • The bare minimum of samples needed in order to divide a node. • The maximum number of features to consider at each split. • The criterion for measuring the quality of splits (e.g., Gini impurity or entropy for classification). 8) Feature Importance: Random Forests can provide information about feature importance, which helps identify which features are most influential in making predictions. This can be useful for feature selection or understanding the importance of different variables in your dataset. FIG Given Input and Expected Output: Input: Data Output: Getting Accuracy Fig. 17. Confusion

matrix score of Random Forest Classifier. Fig. 18. Data-frame of Random Forest Classifier. IV. LIMITATIONS AND RECOMMENDATIONS This research was carried out in a simulated setting as opposed to an actual world setting. Evaluating the model's resilience to adversarial attacks in the real world is crucial. Furthermore, the outcomes of the models were not easily explained. Post hoc interpretability can be achieved by methods like Shapley additive explanations (SHAP) or local interpretable model-agnostic explanations (LIME). Because Android smartphones have constrained processing power and resources, machine learning models for ransomware detection must be small and effective in order to function on these devices without degrading performance. Since lightweight machine learning involves creating compressed machine learning models that can be run on Android-based edge devices, it may offer a viable option. Furthermore, it can be difficult to extract relevant aspects from Android applications. The model's performance depends on selecting the right characteristics and extracting them accurately. It calls for subject knowledge as well as familiarity with the traits of Android malware. If feature selection approaches are not applied effectively, they may be required to reduce dimensionality and remove unnecessary features, which could have an impact on the model's performance. Temporally speaking, ransomware is always changing, with new iterations and evasion methods appearing on a regular basis. It's possible that a model designed to identify ransomware strains that have evolved over time won't be able to identify newly created strains. To stay up with the changing threat landscape, the model needs to be continuously updated and retrained. V. FUTURE WORK Future work in the realm of predicting Android ransomware attacks using categorical classification should focus on refining the model's predictive accuracy through more sophisticated behavioral analysis and feature engineering. Incorporating real-time monitoring capabilities and automated response mechanisms would elevate the system's agility in countering emerging threats promptly. Exploration of ensemble learning approaches, coupled with a commitment to explainability and interpretability, can enhance both the model's robustness and user trust. Continuous updates to the dataset, collaboration with cybersecurity

communities for real-time threat intelligence, and user education features remain pivotal for ensuring the model's adaptability to the dynamic landscape of Android applications. Furthermore, considering cross-platform adaptation to other mobile operating systems would broaden the model's scope and

contribute to a more comprehensive defense against ransomware threats across diverse platforms. VI. CONCLUSION In conclusion, predicting Android ransomware attacks through categorical classification emerges as a pivotal strategy for bolstering mobile security. By deploying advanced machine learning models, the system achieves early detection and risk assessment, prioritizing user protection against the ever-changing ransomware landscape. The commitment to minimizing false positives ensures a user-friendly experience without compromising security. The model's adaptability to new threats and seamless integration into the Android security ecosystem underscore its proactive stance. As the mobile app environment evolves, the scalability of this approach proves essential, ensuring a resilient defense against emerging risks. Overall, the application of categorical classification in predicting Android ransomware marks a significant advancement in fortifying the platform against cyber threats, showcasing the efficacy 13 of machine learning in safeguarding mobile ecosystems. REFERENCES [1] 10 Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization Author links open overlay panel Monika, Pavol Zavarsky, Dale Lindskog https://www.sciencedirect.com/science/article/pii/S1877050916318221 [2] On the efficacy 5 of android ransomware detection techniques: a survey https://acadpubl.eu/jsi/2017-115-6-7/articles/8/18.pdf [3] An Android Malware Detection System Based on Cloud Computing Shujuan Cui*, Gengxin Sun, Sheng Bin, Xicheng Zhou https://www.aidic.it/cet/16/51/116.pdf [4] A 18 Survey on Android Ransomware and its Detection Methods Usama Desai https://www.irjet.net/archives/V6/i3/IRJET-V6I3597.pdf [5] Ransomware Detection System for Android Applications Samah Alsoghyer 1 and Iman Almomani 2,3,* https://www.mdpi.com/2079-9292/8/8/868 [6]

Samsung, “Best Samsung Mobiles with 8GB RAM & Above,” 2020. https://www.samsung.com/in/smartphones/8gb-ram-and-abovemobiles/ [7] StatCounter, “Mobile Operating System Market Share Worldwide,” GlobalStats, 2020. https://gs.statcounter.com/os-marketshare/mobile/worldwide [8] M. Nauman and S. Khan, “Design and Implementation of a Fine-Authorized [9] D. Pranav, “Food, Clothing, Shelter and ‘The Internet,’” Team India Blogs, 2018. https://www.investindia.gov.in/team-india-blogs/foodclothing-shelter-and-internet [10] Symantec, “Internet Security Threat Report,” 2019. https://docs.broadcom.com/docs/istr-24-2019-en (accessed Jun. 23, 2020). [11] V. Chebyshev, F. Sinitsyn, D. Parinov, O. Kupreev, E. 22 Lopatin, and A. Kulaev, “IT threat evolution Q1 2020. Statistics,” Kaspersky lab,2020. https://securelist.com/it-threat-evolution-q1-2020statistics/96959/ (accessed Jul. 05, 2020). [12] Mix, “New Android ransomware threatens to ‘expose’ your porn escapades tothe FBI,” Security, 2020. https://thenextweb.com/security/2020/04/28/androidransomware-fbiporn/ (accessed May 13, 2020). [13] M. K. A. Abuthawabeh and K. W. Mahmoud, 6 “Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features,” [14] M. Scalas, D. Maiorca, F. Mercaldo, C. A. Visaggio, F. Martinelli, and G. Giacinto, “On the Effectiveness of System API-Related Information for Android Ransomware Detection,” 55 [15] S. Sharma, N. Kumar, R. Kumar, and R. K. Challa, “The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach,”. [16] B. Pan, “Tools to work with android .dex and java .class files,” Github, 2020. https://github.com/pxb1988/dex2jar (accessed Apr. 19, 2020). [17] Android, “App resources overview,” 2020. https://developer.android.com/guide/topics/resources/providingresources (accessed Apr. 03, 2020).

Sources

1	https://machinelearningmastery.com/data-visualization-methods-in-python/ INTERNET 2%
2	https://machinelearningmastery.com/difference-test-validation-datasets/ INTERNET 2%
3	https://www.mdpi.com/1999-5903/15/4/144 INTERNET 1%
4	https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/ INTERNET 1%
5	https://www.mdpi.com/1424-8220/24/1/189 INTERNET 1%
6	https://iajit.org/paper/2039/An-Ensemble-based-Supervised-Machine-Learning-Framework-for-Android-Ransomware-Detection INTERNET 1%
7	https://medium.com/@dagstaneren/random-forest-f20027466d78 INTERNET 1%
8	https://www.geeksforgeeks.org/decision-tree-introduction-example/#:~:text=A decision tree is a type of supervised,leaf node represents the final decision or prediction. INTERNET 1%
9	https://panimalar.ac.in/be-computer-science-and-engineering.php INTERNET <1%
10	https://researchr.org/publication/MonikaZL16 INTERNET <1%
11	https://medium.com/@deek.sha.rma/step-by-step-machine-learning-random-forest-1bef0d21019e INTERNET <1%
12	https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/ INTERNET <1%
13	https://www.mdpi.com/2073-8994/14/11/2304 INTERNET <1%
14	https://www.geeksforgeeks.org/bernoulli-naive-bayes/ INTERNET <1%

15	https://emeritus.org/in/learn/data-science-random-forest/ INTERNET <1%
16	https://gustavwillig.medium.com/decision-tree-vs-logistic-regression-1a40c58307d0 INTERNET <1%
17	https://www.nature.com/articles/s41598-024-55243-x INTERNET <1%
18	https://www.irjet.net/archives/V6/i3/IRJET-V6I3597.pdf INTERNET <1%
19	https://ieeexplore.ieee.org/document/10018883 INTERNET <1%
20	https://mlcourse.ai/book/topic03/topic03_decision_trees_kNN.html INTERNET <1%
21	https://www.turing.com/kb/random-forest-algorithm INTERNET <1%
22	https://www.researchgate.net/publication/344680244_Vulnerability_Assessment_of_the_Internet_of_Things_IoT_in_the_Irish_Context INTERNET <1%
23	https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-detection/ INTERNET <1%
24	https://towardsdatascience.com/the-only-guide-you-need-to-understand-regression-trees-4964992a07a8 INTERNET <1%
25	https://medium.com/@nerdjock/lesson-18-machine-learning-regularization-techniques-l1-lasso-and-l2-ridge-regularization-b9dc312c71fe INTERNET <1%
26	https://medium.com/aiguys/xgboost-2-0-major-update-on-tree-based-methods-2e4bc4f15baf INTERNET <1%
27	https://www.researchgate.net/figure/Examples-of-Dynamic-Activities-for-GitHub-Legitimate-Malicious-Users_fig1_337017090 INTERNET <1%
28	https://www.sciencedirect.com/science/article/pii/S1084804519303868 INTERNET <1%

29 https://www.geeksforgeeks.org/understanding-data-attribute-types-qualitative-and-quantitative/#:~:text=Binary Attributes: Binary attributes are a type of,yes/no, presence/absence, or true/false conditions within a dataset.
INTERNET
<1%

EXCLUDE CUSTOM MATCHES	ON
EXCLUDE QUOTES	OFF
EXCLUDE BIBLIOGRAPHY	OFF

A5 Paper Publications

REFERENCES

[1] A note on machine learning applied in ransomware detection by Manuela Horduna1 ,Simona-Maria L˘az˘arescu1 , and Emil Simion2 Year 2022 https://eprint.iacr.org/2023/045

[2] Opportunities for Early Detection and Prediction of Ransomware Attacks against Industrial Control Systems by Mazen Gazzan 1,2 and Frederick T. Sheldon 1,* Year 2022 https://www.mdpi.com/1999-5903/15/4/144

[3] Detecting Malicious Accounts in Online Developer Communities Using Deep Learning by Qingyuan Gong, Yushan Liu, Jiayun Zhang, Yang Chen Year 2021 https://ieeexplore.ieee.org/document/10018883#:~:text=In%20this%20work%2C%20we%20propose,as%20well%20as%20social%20interactions.

[4] Machine Learning Methods For Malware Detection And Classification by Kateryna Chumachenko Year 2017

[5] Analysis, Detection, and Classification of Android Malware using System Calls by Shubham Shakya1 and Mayank Dave2 Year 2020 https://arxiv.org/abs/2208.06130

[6] A Systematic Study of Android Non-SDK (Hidden) Service API Security by Yi He, Yacong Gu, Purui Su , Kun Sun , Senior Member, IEEE, Yajin Zhou , Member, IEEE, Zhi Wang, and Qi Li , Senior Member, IEEE year MARCH/APRIL 2023 https://ieeexplore.ieee.org/document/9739878/

[7] Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization Author links open overlay panel Monika, Pavol Zavarsky, Dale Lindskog.

[8] On the efficacy of android ransomware detection techniques: a survey https://acadpubl.eu/jsi/2017-115-6-7/articles/8/18.pdf

[9] An Android Malware Detection System Based on Cloud Computing Shujuan Cui*, Gengxin Sun, ShengBin, Xicheng Zhou https://www.aidic.it/cet/16/51/116.pdf

[10] A Survey on Android Ransomware and its Detection Methods Usama Desai https://www.irjet.net/archives/V6/i3/IRJET-V6I3597.pdf

[11] Ransomware Detection System for Android Applications Samah Alsoghyer 1 and Iman Almomani 2,3,* https://www.mdpi.com/2079-9292/8/8/868

[12] Samsung, “Best Samsung Mobiles with 8GB RAM & Above,” 2020.

https://www.samsung.com/in/smartphones/8gb-ram-and-above-mobiles/

[13] StatCounter, "Mobile Operating System Market Share Worldwide," GlobalStats, 2020. https://gs.statcounter.com/os-market-share/mobile/worldwide

[14] M. Nauman and S. Khan, "Design and Implementation of a Fine-Authorized

[15] D. Pranav, "Food, Clothing, Shelter and 'The Internet,'" Team India Blogs, 2018. https://www.investindia.gov.in/team-india-blogs/food-clothing-shelter-and-internet

[16] Symantec,"Internet SecurityThreat Report," 2019. (accessed Jun. 23, 2020). https://docs.broadcom.com/docs/istr-24-2019-en

[17] V. Chebyshev, F. Sinitsyn, D. Parinov, O. Kupreev, E. Lopatin, and A. Kulaev, "IT threat evolution Q1 2020. Statistics," Kaspersky lab,2020. https://securelist.com/it-threat-evolution-q1-2020-statistics/96959/(accessed Jul.05, 2020).

[18] Mix, "New Android ransomware threatens to 'expose' your porn escapades tothe FBI," Security, 2020. https://thenextweb.com/security/2020/04/28/android ransomware-fbi-porn/ (accessed May 13, 2020).

[19] M. K. A. Abuthawabeh and K. W. Mahmoud, "Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features,"

[20] M. Scalas, D. Maiorca, F. Mercaldo, C. A. Visaggio, F. Martinelli, and G. Giacinto, "On the Effectiveness of System API-Related Information for Android Ransomware Detection, S. Sharma, N. Kumar, R. Kumar, and R. K. Challa, "The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach,".

[21] B. Pan, "Tools to work with android .dex and java .class files," Github, 2020. a. https://github.com/pxb1988/dex2jar (accessed Apr. 19, 2020).

[22] Android, "App resources overview," 2020. a. https://developer.android.com/guide/topics/resources/providing-resources (accessed Apr.03, 2020).