Malware Profiling and Classification using machine learning algorithms

Masters of Science in Data Analytics

B9DA113_2324_TMD1S – DISSERTATION

8th Jan 2024

Name- Hanisha Rathee

Student Id- 10639557

Supervisor- Dr. Anesu Nyabadza

Declaration

This dissertation has not been previously submitted to any institution for evaluation, and I thus affirm that it is my original work.

Additionally, I have referenced all sources and recognized their use.

Hanisha Rathee

……

Date: 8th Jan 2024

Acknowledgement

I would like to express my gratitude to all of the professors and other members of the university staff who provided me with support and help during my academic journey through the University. For all of the chances that were presented to me while I was a student here, as well as for all of the individuals who were either directly or indirectly engaged in it, I am really thankful.

In particular, I would like to express my gratitude to my dissertation supervisor, who introduced me to the concept of machine learning and kindled my interest in the subject matter to such an extent that I was able to finish my dissertation. Due to his extensive knowledge of the area, I was able to easily formulate the theme for my study.

Table of Contents

Commented [AN1]: abstract must be reflected in the table of contents and nust come after this table of contents.

all pages before this must have a different numbering system like i ii, iii, vi, v etc, abstract should then be page one and should come after the table of contents.

Abstract

The study done on "Malware Profiling and Classification using Machine Learning Algorithms" compares multiple machine learning models for malware detection and profiling to enhance cybersecurity. Machine learning adaptable skills were used to identify and classify complex malware threats with the help of algorithms like SVM, Random Forest, and Autoencoders to analyze historical malware data. These models were selected for their pattern recognition and anomaly detection successes in large datasets. The data was carefully preprocessed to assure accuracy and relevance and machine learning algorithms were taught to recognize complex malware patterns. Each model was rigorously evaluated using 70% training and 30% validation data throughout the inquiry. The models' performance was assessed using accuracy, precision, recall, F1 score, AUC, and ROC curve. The SVM model gives proper results identifying safe and dangerous software with 0.99 AUC. However, the Random Forest and Autoencoder models scored 1.0 in the AUC statistic which is ideal. These results showed that these models had nearly minimal false positives, which is crucial in malware detection systems.

A near-perfect score of 0.9999 showed that the Random Forest model accurately classified data points with 1.0 accuracy suggests that the model predicted no false positives. The model recognized almost all real malware occurrences with a recall score of 0.9998 indicating success. The model's balanced prediction abilities were confirmed by its 0.9999 F1 score whereas the Autoencoder model had an accuracy of 0.9959, a precision of 0.9955, and an F1 score that matched the Random Forest. The recall rate of 0.9962 indicated that it was somewhat better at detecting true positives than the Random Forest model. This model's AUC score was 1.0, and its ROC performance was crucial for confidently differentiating classes. A ROC curve comparison showed that Random Forest and Autoencoder models performed better. The ROC curve shows how efficiently binary classifier systems identify malware as when the curve closely matches the left-hand and top ROC space boundaries, the model is more accurate. Both models have excellent classifier behavior as seen by the ROC curve in the top left corner. All models examined were effective, however the Autoencoder model was somewhat better, making it the preferred malware classification and profiling technique.

Commented [AN2]: Abstract is way way too long, reduce to 200-300 words max.

Chapter 1: Introduction

1.1 Introduction

Cybersecurity is becoming more important with the advancement of technology and the research that is done on "Malware Profiling and Classification using Machine Learning Algorithms" focuses on the growing danger of malware in today's increasingly digital environment. Due to the rapid increase in the sophistication of malware, conventional countermeasures are often rendered ineffective and need innovative approaches to protection [10]. The purpose of this research is to use machine learning (ML) techniques to improve malware profiling and classification to provide a more adaptive and insightful method of cyber defense. The aim of this study is an all-encompassing framework for analyzing and categorizing malware samples using a variety of machine-learning approaches. The paper offers a method to accurately differentiate between benign and harmful software, as well as categorize various forms of malware, by analyzing their properties and actions. This is especially critical in an age when malware complexity is on the increase, and standard signature-based detection approaches fall short.

The research starts with a review of the state of malware threats today and the shortcomings of current detection methods. Machine learning methods are then discussed in detail, with an emphasis on their applicability and efficacy in the context of malware detection. Gathering a large number of malware samples is the first step in the research process, followed by feature extraction and the use of many different machine-learning classifiers [53]. This paper lays forth a technological foundation for categorizing malware, but it also delves into the consequences and possible uses of such a system. In light of the ever-changing nature of cyber threats, it highlights the need for adaptive, learning-based security measures. The findings of this study have the potential to dramatically advance the state of the art in malware detection systems and provide important insights to the cybersecurity community.

1.2 Background

The study is rooted in the changing environment of cyber threats, where conventional security solutions are being challenged by more complex malware and there is importance for creative ways of recognizing and combating such threats. Malware is becoming more sophisticated and stealthy, making it harder to detect and block using traditional methods of protection. The shortcomings of these more conventional approaches, which are mostly dependent on signature

detection, are already clear. New and undiscovered malware strains appear at an alarming rate, making it difficult for security teams to keep up as Machine learning (ML), is a subfield of AI that allows computers to pick up new skills and hone old ones without any human intervention. Algorithms based on machine learning have shown promise in several areas due to their capacity to recognize patterns and make judgments with little to no input from humans. When it comes to finding malicious software, ML is a proactive and adaptable option [11]. It can evaluate enormous quantities of data, learn from novel malware behaviors, and respond to emerging threats more effectively than older approaches. A variety of ML approaches that are well-suited for malware classification are investigated, and the current literature on malware kinds, behaviors, and conventional detection methods is reviewed. The context is provided by this introduction, which details the shortcomings of existing cybersecurity methods and the opportunities presented by machine learning. The research attempts to fill this void by presenting novel insights and actionable recommendations for countering malware.

1.3 Research Aim and Objectives

1.3.1 Aim

The study aims to develop effective and precise malware detection and classification models using machine learning algorithms.

1.3.2 Objectives

- To highlight the necessity for machine learning algorithms in the detection of malware to enhance cybersecurity measures.
- To compare and contrast traditional malware detection methodologies with machine learning-based techniques to illustrate advancements and improvements.
- To identify and discuss the challenges associated with employing machine learning algorithms for malware detection, including issues related to accuracy, efficiency, and scalability.
- To explore a range of machine learning algorithms that are employed in the classification and profiling of malware to understand their effectiveness and application contexts.
- To examine various countermeasures that can be implemented to prevent malware infections, thereby reinforcing the defense mechanisms against cyber threats.

1.4 Research Questions

- How do ML algorithms help in classifying and profiling malware?
- What are the challenges faced by ML algorithms to detect Malware?
- What is the accuracy, precision, recall, AUC, ROC and F1-score of SVM, Random Forest, and Autoencoders?
- Which of the ML algorithms is better in classifying and profiling Malware?

1.5 Research Rationale

Signature-based techniques, which have traditionally been used for malware detection, are failing to keep up with the ever-changing nature of the threat. These techniques rely too much on previously discovered malware signatures, making it difficult to detect variants that have been altered or created from scratch [4]. This gap in detection leaves systems open to zero-day assaults and sophisticated persistent threats, which may have severe implications. One approach that shows promise is the use of machine learning (ML) techniques and their strengths in data learning, pattern recognition, and prediction are a good fit for the need for efficient malware detection and categorization [51]. Rather than depending entirely on existing signatures, ML can give a more flexible, proactive approach, one that can discover new malware variants based on behavioral analysis and anomaly detection.

The goal of this study is to see whether ML can be used to improve existing cybersecurity infrastructure. This research aims to improve security against a broad variety of cyber assaults by using ML algorithms for malware profiling and classification. This would allow the system to not only identify existing malware but also find new threats. This study's overarching goal is to advance cybersecurity by providing novel approaches to malware detection, therefore assisting in the protection of cyberspace from new dangers. The study's justification lies in the pressing necessity to strengthen cybersecurity precautions in light of the increasing complexity of malware attacks. As digital technology becomes more vital to personal, business, and government activities, the effect of malware outbreaks has risen, presenting serious hazards to data integrity, privacy, and general cyber safety.

1.6 Research Significance

The research paper is important because it has the potential to transform the cybersecurity industry by making use of cutting-edge machine learning (ML) methods. The complexity and severity of

the danger presented by malware are growing as the digital world evolves, making the creation of better detection and classification systems critical. First, this research fills a major need in the state of the art of cybersecurity. Traditional malware detection technologies sometimes fail to identify malware that is either newly created or fast-evolving. This study attempts to significantly improve malware detection by developing a system that can adapt to and recognize these ever-changing threats using ML algorithms.

Second, the findings have far-reaching consequences for fields including banking, healthcare, government, and more that rely heavily on digital technology. As a result, enhanced malware detection and classification systems may contribute to cyber resilience by protecting sensitive data, avoiding financial losses, and protecting vital infrastructure. In addition, this study provides the groundwork for future cybersecurity innovations by applying ML to the problem of detecting malware may provide valuable insights that can be used to create more advanced security solutions that can keep up with the ever-evolving nature of cyber threats. Finally, the research is important because it educates people and adds to the body of knowledge in machine learning and cybersecurity. It may offer useful learning material for students and professionals alike, developing a greater knowledge of the confluence between these two key areas of technology, and by improving the efficiency of cybersecurity measures, the study might have a major influence and play a significant part in the continuing fight against cyber threats.

Chapter 2: Literature Review

2.1 Introduction

It is important to have accurate categorization of data to perform efficient analysis based on which countermeasures should be developed against the growing threat posed by malware and its constantly developing varieties (Jiang *et al.*, 2018). Certain limitations exist in today's categorization methods which are based mostly on network activity only. When dealing with advanced malware that shows complicated behavior patterns, these solutions might be insufficient. Since they either give just a limited picture of the virus's network activity or they need predetermined traffic choices. To profile the network actions of malware in a better manner this study presents an updated behavior model and a novel method for classifying malware is introduced with an emphasis on identifying and distinguishing malware samples' that have fundamental characteristics [21]. Experiments and comparisons show that compared to two other methods, the approach chosen in this study is more accurate and the algorithm excels in scenario-based testing especially in particular, highlighting its efficacy in accurately classifying malware samples.

According to Hsiao *et al.*, (2019), malware attacks have become more common in the digital world as the Internet and smart apps have grown in popularity among people all over the world. As people around the world are connected to high-speed internet via advanced gadgets, cyber-attacks using malware are also increasing. There are several malware detection technologies available, but the complexity and scale of today's digital world need to focus more on sophisticated approaches. This research presents a novel approach to detect malware by combining deep learning with visualization. Two main goals drive this study that is by demonstrating how useful image-based approaches are for spotting system abnormalities, and by investigating how they may be used with deep learning models to improve malware classification. The study evaluates this method using deep learning frameworks and a variety of metrics that measure similarities in virus behavior. The approach used in this study has been shown to accurately detect malware after being tested on large public and private malware datasets.

2.2 Need for machine learning algorithms to detect malware.

2.2.1 Traditional detection Vs ML-based detection

As technology is advancing malware threats are also increasing with the interconnectedness of the digital world, and new variations of existing malware are typically difficult to detect with traditional systems for detection as the new variants are more advanced [22]. The use of machine learning (ML) to identify malicious software is an exciting new development and this research presents a static malware detection system that makes use of n-gram and ML methods. This technique improves the performance and accuracy of ML classifiers by using known malware sub-signatures to reduce the large feature spaces generally created by n-gram feature extraction and to improve the feature count, this study analyzed several different ML classifiers and the authors looked into numerous feature selection methodologies [43]. Accuracy rates higher than 99.78% with no false positives were observed when using 4-gram features with the majority of the evaluated ML classifiers, suggesting that combining n-gram with Snort sub-signature characteristics through ML approaches gives outstanding detection rates of malware.

2.2.2 Deep learning for malware detection

This study done by Yuxin and Siyi, (2019), made use of a deep belief network (DBN) to identify malware by modeling it as a series of opcodes DBNs are an alternative to typical neural networks and make use of unlabeled data to pre-train a multi-layered generative model that is better at capturing data sample features. In this research, the authors compared the efficacy of DBNs to that of three more traditional malware detection models such as decision trees, support vector machines, and the k-nearest neighbor method. In terms of detection accuracy, DBNs have been shown to perform better than these traditional methods. The study used more unlabeled data for pretraining which has increased the DBNs' benefits even further. It has been shown that using DBNs as an autoencoder helps in extracting the feature vectors from executables, which is indicative of the fact that it can understand the underlying structure of the input data while simultaneously shrinking the feature vector dimensions significantly.

2.2.3 Real-Time Detection

Real-time malware detection is difficult with today's systems for safety-critical equipment due to performance limitations and to effectively identify malware in real-time, this study introduces PROPEDEUTICA, a novel framework that blends traditional Machine Learning and Deep

Learning [48]. PROPEDEUTICA uses a fast ML classifier for early-stage detection and assumes that all software executions are safe at first. When a piece of software is given a classification that might go either way, it is sent to a DL detector that is more exact but also more computationally intensive. The study has included a new DL architecture called DEEPMALWARE to manage the difficulties posed by spatial-temporal dynamics and varying program execution patterns [48]. In testing, the study used 9,115 samples of malware and 1,338 samples of benign software for Windows OS and found that PROPEDEUTICA had an accuracy of 94.34%, a false-positive rate of 8.75%, and sent 41.45% of the samples to further analyzed that means it can identify malicious software using just a CPU in about 0.1 seconds.

2.3 Challenges Faced in Machine Learning-based Detection of Malware

2.3.1 Scarcity of Labelled Data

It is necessary to have labeled data to use supervised learning, which is the most common method used in Machine Learning-based malware detection. Labeled data are data instances that have been designated as either "malicious" or "benign" and getting a sizable volume of data that is correctly labeled is a huge obstacle to overcome [14]. For machine learning models to be useful, they need a substantial quantity of training data, which might consist of hundreds or even millions of samples that have been labeled but classifying those data takes a lot of time and needs specialists in the relevant fields. Even though there is an abundance of malware samples that are readily accessible correctly classifying them is a difficult task. Since malicious software is always being updated the previous samples may become irrelevant for which there is a need for the ongoing collection of data and categorization of threats [27]. So, putting all of the ones in the basket of unsupervised learning is one possible option although it often does not attain the same degree of accuracy that supervised models do.

2.3.2 Adversarial attacks

An adversarial attack is any alteration that is made to fool a machine learning model for the detection of malware and it refers to making modifications to malware to render it undetected to a machine learning model even if the model was trained on a variation of the malware in question [30]. Malicious inputs to machine learning models that have been tampered with to produce inaccurate model outputs are referred to as adversarial samples. These adjustments may be very minor which is undetectable to humans, but they have the potential to throw off the detection

model. For example, malicious software may be subtly altered such that it continues to perform its intended tasks but gives an artificial intelligence model the impression that it is safe [30]. These hostile samples offer a substantial risk because they take advantage of the model's flaws and have the potential to dramatically decrease its accuracy of identification.

2.3.3 Feature selection and extraction

The act of identifying which aspects of the data that is also known as attributes are most important to the work at hand is referred to as feature selection. These characteristics might be API calls, file sizes, or patterns in network traffic which are relevant when discussing the detection of malware. Due to the varied and extensive nature of its components for feature selection and extraction, it has both positive and negative aspects [7]. High features might improve the accuracy of machine learning models, but on the other hand, they can also add noise that can make the models excessively complicated and lower their capacity for generalization. When dimensionality is reduced, there is a possibility that important data may be lost but still maintain many features that might make detection more difficult and can use a significant amount of processing resources [7]. Additionally, the manual method of feature engineering may be time-consuming and laborious, and it may not scale well with the increasing quantity of malware samples.

2.3.4 Evolving nature of Malware

One of the most critical issues is the ever-changing nature of the environment of malware and Malware based hackers are also inventing at the same time when cybersecurity measures are advancing. They are building malware that can avoid detection and can mutate or even fool machine-learning algorithms [26]. Techniques such as metamorphism are used by hackers in which malware rewrites its code and polymorphism is a way in which malware alters its code, data, or encryption that seems to be different and is used by those who create malware. It is difficult for static machine learning models, which are algorithms that only use data from the past and this kind of algorithm are easily bypassed by hackers [26]. If a model has been trained on a specific malware signature, there is a possibility that it will be unable to identify a new or modified form of the malware because there is no match in the model's training data for the new or modified variant.

2.4 Different Machine Learning Algorithms to Classify Malware

2.4.1 Deep Learning

Cybercriminals are increasingly focusing their attention on the digital space since it is easier for them to carry out cyberattacks using malware, which is constantly evolving due to improved encryption or concealment techniques. Malware detection and categorization are made more difficult by such strategies and when it comes to spotting sophisticated malware strains, typical machine learning (ML) algorithms are unable to detect those malware, but deep learning (DL) provides a potential alternative [3]. Among all the other machine learning algorithms DL differentiates itself and it can identify all kinds of malware with greater efficiency. To categorize malware patterns, this research suggests a new deep-learning-based architecture. The design makes use of a hybrid paradigm to improve the efficiency with which cyber threats are detected and countered.

The study done by Ding *et al.*, (2016), focuses on a deep belief network (DBN) which is used to identify malware by portraying it as a series of opcodes. DBNs are not like other standard neural networks, DBNs can efficiently capture data properties by pre-training a multi-layer generative model using unlabeled input. Different baseline models, including SVMs, DTs, and the k-nearest neighbor technique, are compared with DBNs for malware detection in this study. According to the findings, DBNs provide more precise detections, and their effectiveness may be improved by using more unlabeled data for pre-training. The research also uses DBNs as an auto-encoder to extract feature vectors of executables, proving the efficacy of modeling the underlying data structure and reducing dimensionality.

This study made by Sewak *et al.*, (2018), compares the performance of the traditional Random Forest (RF) machine learning method with that of Deep Learning, specifically the Deep Neural Network (DNN), for malware classification. DL is mainly used in areas like image identification and natural language processing which makes Deep Learning more resource-heavy than traditional methods. This research examines the effectiveness of 2, 4, and 7-layer RF and DNN architectures on four distinct feature sets. In malware classification tasks, the results show that the conventional RF consistently outperforms the DNN independent of the input characteristics.

2.4.2 MLDroid

In this study, the authors provide MLDroid which is a web-based platform for scanning Android devices for malware. Malware that threatens Android's security and privacy is increasing daily with the platform's fast rise in popularity. To combat this, MLDroid does real-time analysis of Android applications in search of malware, and several feature selection methods were used to train the framework, which improved its detection skills. Several machine-learning methods were used for this data set to construct a model [25]. Over half a million Android applications were used in a study and by using rough set analysis for feature subset selection, the results showed that a model combining four different machine learning algorithms that is deep learning, farthest first clustering, Y-MLP, and a nonlinear ensemble decision tree forest approach which achieved an impressive malware detection rate of 98.8% for real-world apps.

The growing number of Android-based smartphones and devices has increased by malicious software, leading to the need for reliable detection methods. Due to Android's restrictions on third-party applications' access to low-level information, existing dynamic analysis approaches generally are deficient, and particular solutions may only be successful against certain malware types. The study provides SpyDroid which is a framework for real-time malware detection that integrates various detectors from sources other than traditional antivirus manufacturers and academic researchers. SpyDroid helps facilitate application layer sub-detectors and has two OS modules for monitoring and detection [19]. These mini-detectors, which seem to the user as standard Android applications, monitor and analyze data gathered during app execution, and then transmit their results to the detection module. This method paves the way for academics and businesses to disseminate their methods via app stores, giving end users the option to deploy as many specialized detectors as they see fit.

2.4.3 One-Class SVM

Organizations around the world have to suffer serious financial and reputational losses as a result of the ongoing conflict between malware creators and information security experts [33]. There are many undiscovered flaws in software and online tools, which hackers may use to their advantage and do significant damage, compounding the problem. This research suggests a novel method for detecting and categorizing malware by using visual representations of malware binaries. The principal component analysis is used to extract highly discriminative characteristics of malware

kinds and structures, and an optimized support vector machine model is then used to classify the features [15]. The method suggested in this study is distinct from preexisting models because it uses algebraic dot products to classify malware based on digital pictures, which greatly simplifies the process. In the face of persistent and ever-evolving cybersecurity threats, this innovative plan seeks to provide a simple and efficient answer.

Malware is always developing, and it has traditionally been designed to infect computers running the ubiquitous Windows OS. There isn't nearly enough study done to solve the problem of new Windows malware samples being released every day [6]. Programs written for Windows often make use of the Windows Registry, which may be used to identify malicious software. To detect both known and undiscovered malware that maliciously manipulates registry keys and values, this work presents RAMD, a novel technique that makes use of an ensemble classifier built of numerous one-class classifiers [49]. RAMD builds a behavior model using the registry actions of safe programs. To counter the increasingly sophisticated malware threats, this technique is used to detect malicious software by spotting anomalous registry visits.

2.4.4 Isolation Forest

A new class of cyber-threats known as advanced persistent threats (APTs) is increasingly being used to launch devastating assaults on a wide range of targets, including institutions and infrastructures. A semi-automatic method of malware analysis is required for the rapid detection of APT-related malware. Recent developments in malware triage facilitate the prompt selection of APT samples for in-depth investigation. Using public reports as input static malware traits and machine learning for detection, this paper details the process of building a knowledge base on known APTs. However, with the current method, training takes longer and new APT samples or classes need a full retrain [23]. This paper recommends switching from multi-class classification to a collection of one-class classifiers to drastically cut down on runtime and increase modularity while still achieving accuracy and precision of above 90%.

Data mining focuses considerable attention on the study of outliers and other types of anomalies. Algorithms with a track record of success include Local Outlier Factor (LOF) and Isolation Forest (iForest). While LOF excels at spotting local outliers, it has a high temporal complexity compared to iForest's low computational requirements. This study offers a two-layer progressive ensemble approach for outlier identification, which overcomes these drawbacks by accurately identifying

outliers in complicated datasets with little computational overhead. At first, iForest does a rapid data scan, eliminates mundane data, and produces a group of candidates for outliers. To improve pruning precision, we propose an outlier coefficient and develop a threshold-setting approach according to the data's outlier degree [9]. Then, LOF hones the pool of potential outliers for more precise detection. This ensemble approach makes use of both algorithms in tandem, maximizing the efficiency with which computational resources are allocated.

2.5 Countermeasures

Ransomware is malicious software that encrypts data on affected computers and then demands payment to restore access. The rise in ransomware attacks, highlighted by a high-profile instance on May 12, 2017, has elevated the importance of network security. Several methods for protecting yourself against ransomware and other forms of cyber terrorism are outlined in this article. Spora, CryptoLocker, Locky, CryptoWall, Petya, Cerber, Sanam, Aris Locker, Jigsaw, WannaCry, and Reveton are just some of the ransomware strains that have been particularly prevalent in recent years [46]. While there is no proper way to avoid ransomware, it is possible to dramatically increase security by adopting routine software upgrades, setting up preventive measures, and performing frequent backups.

Malware due to its stealth, advanced persistent threats, especially zero-day malware, are at the core of complex cyber assaults. Mathematical models can accurately forecast their spread across networks, thus understanding how they propagate is essential. The computational implementation of such models may provide results that can be put to forensic use in SOCs. To mimic the transmission of sophisticated malware, this paper proposes a new mathematical model. The suggested model takes into account many states of the device that are vulnerable, carrier, infectious, recovered, attacked, and vulnerable, both locally and globally [17]. The work provides a novel method for predicting and preventing malware transmission by analyzing the local and global stability of the model's equilibrium points and computing the fundamental reproduction number.

Ransomware is a sort of malicious software that encrypts user data and then demands money to decrypt it. This kind of criminal activity has evolved into a multibillion-dollar industry and poses a special danger since its consequences last long after the malware has been eradicated. Victims may lose not just money, but also important information and their good name if they fall victim to

a cyberattack. Despite the abundance of research on ransomware protection and detection, detailed survey papers describing the obstacles these methods face are uncommon [2]. This study is an attempt to fill that void by offering a comprehensive analysis of ransomware studies and defenses. It presents a new taxonomy for classifying ransomware, examines the elements that make assaults possible, and explains at length the work done to analyze, prevent, detect, and anticipate ransomware attacks.

Applications built for the Industrial Internet of Things (IIoT) seek to decrease operating costs and improve asset dependability to maximize profitability in key industrial industries. However, these smart sectors are particularly susceptible to malware attacks because of their dependence on connection and interoperability. Significant property damage and even life-threatening circumstances have occurred from recent assaults on Cyber-physical systems. This study investigates the risks that businesses face while implementing IIoT, including potential assaults on IIoT's layered architecture and recommendations for mitigating such risks [35]. The research concludes by recommending an IIoT attack taxonomy to help reduce risks and tighten up IIoT app security.

Malware authors have been forced to come up with more complex techniques of evasion as a result of the advent of machine learning algorithms that can identify Android malware with a success rate of up to 98%. This study presents the "Collusion Attack" scenario, which is a threat to detection methods based on machine learning, in particular Support Vector Machines (SVM). This attack strategy is particularly difficult to detect since malware payloads are spread throughout several different applications [8]. In addition, "Evasion Attack" methods like obfuscation may be used by attackers to keep their handiwork under wraps. Based on our simulations, 87.4% of applications can evade Linear SVM using just the Collusion Attack, while 100% can do so using both the Collusion Attack and the Evasion Attack. The research suggests a way to overcome these sophisticated avoidance techniques.

Robust countermeasures are required in light of the increasing prevalence of malware and the complexity of cyber-attacks, particularly against new forms of malware that are not yet known to security-based companies [37]. When it comes to distinguishing between malicious and benign communications, traditional detection systems have a hard time. This research provides a unique approach to identifying malware processes based on their behavior on possibly infected terminals,

providing a solution to this problem [50]. The method employs Deep Neural Networks in a two-stage process: first, an RNN is taught to extract aspects of process behavior, and then a CNN is trained to categorize photos based on these features. Results from comparing the Area Under the Curve (AUC) of the ROC curves show that the strategy is effective, with the best-case scenario yielding an AUC of 0.96.

Chapter 3: Research Methodology

3.1 Introduction

The study on "Malware Profiling and Classification using machine learning algorithms" is done based on a mixed approach in which qualitative as well as quantitative method is applied. The secondary data is collected using the qualitative method in which previous published journals and articles are taken into consideration and knowledge is extracted from those journals and articles that are related to this study or related to the topic. For the primary data, a quantitative method is applied in which obfuscated malware is downloaded from the Kaggle website which is designed to test the obfuscated malware detection methods through memory.

3.2 Data collection and analysis

In this study, the secondary data insight is written in Chapter 2 (literature review section) in which recent journal articles that have been published after 2016 are considered.

The following steps are taken for quantitative data collection and analysis:

Data Gathering: The obfuscated malware dataset is downloaded from the Kaggle website which consists of malware types that hide them to be detected and extermination (kaggle.com, 2022). The dataset was built to replicate a real-life scenario as possible using malware that is common in computing. It offers a well-rounded dataset for testing obfuscated malware detection algorithms since it contains Spyware, Ransomware, and Trojan Horse malware. To conceal its memory-dumping procedure, this dataset operates in debug mode. This helps to accurately model the configuration that a typical user would have under assault from malware.

Data Preprocessing: After gathering the data it is important to pre-process the dataset. In the pre-processing step, the dataset is checked for any null values or any missing values and then with proper methods the null values or missing values are eliminated. Proper data processing is important and a follow-up variable conversion is conducted to attain meaningful results.

Visualization using Python: Profiling and classifying of malware using machine learning methods relies on having a clear visualization of the dataset. This can be done using various graphs and charts features of Python-based libraries. To decipher the obfuscated code, graphs like heatmaps are generated using Python tools like Matplotlib and Seaborn. Malware samples may be distinguished from one another by analyzing their feature distributions and correlations. Effective

malware classification relies on the use of relevant features and methods. Proper dimensionality reduction techniques like principal component analysis (PCA) and t-SNE may assist in depicting high-dimensional data in 2D or 3D space for intuitive comprehension.

Malware detection and classification using Python-based ML algorithms: Three algorithms will be employed herein including one-class SVM, Isolation Forest, and Autoencoders for malware detection and classification. The models are trained and verified using cross-validation methods. The dataset is splitted in 70:30 as splitting the data in a 70:30 ratio involves dividing the dataset so that 70% is used for training the machine learning model, and the remaining 30% for testing, such as validating the model's performance. The splitting is ensured fair as the training and testing datasets is representing the entire dataset. Models for anomaly identification, such as Isolation Forest, One-Class SVM, and Autoencoder, will be validated using k-fold cross-validation. Given the specialised nature of these models, especially their emphasis on unsupervised learning and outlier identification, a 5-fold cross-validation is adequate. The accuracy of this estimate is balanced with the computational efficiency gained by using this value. By using every data point for both training and validation, it lowers the likelihood of overfitting, which is prevalent in models dealing with outlier identification. For models like Autoencoders, where overfitting may have a major effect on performance, this strategy is essential. Malware families may be profiled, their behavior analyzed, and cybersecurity can be improved with the aid of the categorization findings. Python's flexibility and abundance of libraries allow for rapid data analysis and model building.

3.3 CRISP-DM framework

Data mining projects are often managed using the CRISP-DM (Cross Industry Standard Process for Data Mining) architecture [44]. It offers a methodical framework for conducting data mining projects. CRISP-DM is divided into six stages: The first step is defining the business's goals and needs; the second is learning about the available data; the third is getting that data ready for use by applying various modeling techniques; the fourth is building the model; the fifth is testing it; the sixth is putting it into action [41]. This adaptive architecture allows for the iterative development of data mining applications that meet the unique requirements of every given organization.

Malware profiling and classification research is best conducted using a methodical and comprehensive framework like CRISP-DM. The research is poised to deliver useful insights and tools for combatting obfuscated malware because of the iterative nature with which the business

challenge was understood, the data was prepared, the model was evaluated, and the solution was deployed. This method permits ongoing enhancement and adjustment to the ever-changing nature of cybersecurity threats. Data mining projects may be planned and carried out more methodically with the help of the Cross-Industry Standard Process for Data Mining [12]. It is important to look at how this framework is used in the context of the obfuscated malware dataset used in the "Malware Profiling and Classification using machine learning algorithms" in this research paper.

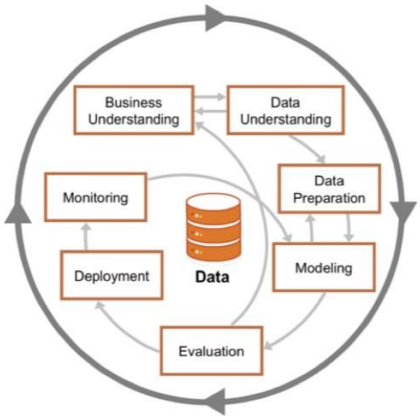

Figure 1: CRISP-DM framework

Source: [12]

Business Understanding

The major objective is to profile and categorize obfuscated malware to have a better understanding of the behavior of these threats and to improve security measures. The stakeholders connected to this study are the professionals in the field of cybersecurity, as well as corporations and people attempting to protect their computer systems against malware attacks [5].

Data Understanding

The research makes use of a dataset of obfuscated malware, which may include scripts, executable files, and other types of dangerous payloads that have been disguised to avoid being discovered. The first analysis of the dataset is carried out to get an understanding of the data's many forms, structures, and complexity [20]. The information that is looked upon may contain raw binary data, opcodes, API calls, and other aspects that are essential for analyzing malware.

Source of Data

The data is taken from the Kaggle website as this dataset performs the memory dump procedure in debug mode to prevent the dumping process from being shown in the memory dumps. This serves to show a true sample of what a typical user might have operated at the time when malware would attempt to compromise their system (kaggle.com, 2022).

Data Dictionary

Category: Categorical attribute describing the type of malware.

pslist.nproc: Number of processes identified.

pslist.nppid: Number of unique parent process IDs.

pslist.avg_threads: Average number of threads per process.

pslist.nprocs64bit: Number of 64-bit processes.

pslist.avg_handlers: Average number of handlers per process.

dlllist.ndlls: Total number of Dynamic Link Libraries (DLLs) loaded.

dlllist.avg_dlls_per_proc: Average number of DLLs per process.

handles.nhandles: Total number of handles.

handles.avg_handles_per_proc: Average number of handles per process.

handles.nport: Number of port handles.

handles.nfile: Number of file handles.

handles.nevent: Number of event handles.

handles.ndesktop: Number of desktop handles.

handles.nkey: Number of key handles.

handles.nthread: Number of thread handles.

handles.ndirectory: Number of directory handles.

handles.nsemaphore: Number of semaphore handles.

handles.ntimer: Number of timer handles.

handles.nsection: Number of section handles.

handles.nmutant: Number of mutant handles.

ldrmodules.not_in_load: Number of modules not in load order.

ldrmodules.not_in_init: Number of modules not in initialization order.

ldrmodules.not_in_mem: Number of modules not in memory order.

ldrmodules.not_in_load_avg: Average number of modules not in load order per process.

ldrmodules.not_in_init_avg: Average number of modules not in initialization order per process.

ldrmodules.not_in_mem_avg: Average number of modules not in memory order per process.

svcscan.kernel_drivers: Number of kernel drivers identified.

svcscan.fs_drivers: Number of file system drivers identified.

svcscan.process_services: Number of process services identified.

svcscan.shared_process_services: Number of shared process services identified.

svcscan.interactive_process_services: Number of interactive process services identified.

svcscan.nactive: Number of active services.

callbacks.ncallbacks: Number of callbacks identified.

callbacks.nanonymous: Number of anonymous callbacks identified.

callbacks.ngeneric: Number of generic callbacks identified.

Class: Target variable indicating whether the sample is malicious or benign.

The other columns, which are not expressly indicated, are most likely other data derived from the malware samples. These features may include further counts and averages of system resources or behaviors that the infection exhibits. These characteristics help establish a profile of the behavior

of the virus, which may subsequently be utilized for categorization with the help of machine learning techniques.

Exploratory Data Analysis (EDA)

EDA, or exploratory data analysis, is an important phase of data analysis that involves discovering and summarising the key aspects of a dataset by visual means. It entails evaluating data patterns, finding anomalies, testing hypotheses, and validating assumptions via different approaches including statistical graphics, charts, and statistical summary measures. EDA is a methodology to analyse datasets to summarise their essential properties, frequently using visual tools, before applying more formal statistical techniques [28]. It's a never-ending cycle of data querying, interpretation, and learning that lays the groundwork for further in-depth research and insightful conclusions.

- The first step involves computing basic statistics such as mean, median, standard deviation, and quartiles for each feature. This provides an understanding of the central tendency and dispersion of the data.
- Histograms, box plots, and density plots are used to visualize the distribution of each feature. This helps in identifying any skewness or outliers that may need to be addressed [42].
- Analyzing the distribution of the target variable, 'Class', helps in understanding if the dataset is balanced or imbalanced. If imbalanced, resampling techniques may be required.
- Correlation matrices and heatmaps are used to identify relationships between features. This can help in detecting multi-collinearity and understanding which features are most relevant to the target variable.
- The dataset is examined for any missing or null values. Depending on their proportion, appropriate imputation or deletion methods are applied.
- For categorical features like 'Category', bar plots are used to visualize the frequency of each category.

```
print(malware_df.describe())
```

```
Descriptive Statistics:
       pslist.nproc  pslist.nppid  pslist.avg_threads  pslist.nprocs64bit  \
count  58596.000000  58596.000000        58596.000000             58596.0
mean      41.394771     14.713837           11.341655                 0.0
std        5.777249      2.656748            1.588231                 0.0
min       21.000000      8.000000            1.650000                 0.0
25%       40.000000     12.000000            9.972973                 0.0
50%       41.000000     15.000000           11.000000                 0.0
75%       43.000000     16.000000           12.861955                 0.0
max      240.000000     72.000000           16.818182                 0.0

       pslist.avg_handlers  dlllist.ndlls  dlllist.avg_dlls_per_proc  \
count         58596.000000   58596.000000                58596.000000
mean            247.509819    1810.805447                   43.707806
std             111.857790     329.782639                    5.742023
min              34.962500     670.000000                    7.333333
25%             208.725000    1556.000000                   38.833333
50%             243.963710    1735.000000                   42.781524
75%             289.974322    2087.000000                   49.605280
max           24845.951220    3443.000000                   53.170732

       handles.nhandles  handles.avg_handles_per_proc  handles.nport  ...  \
count      5.859600e+04                  58596.000000        58596.0  ...
mean       1.025858e+04                    249.560958            0.0  ...
std        4.866864e+03                    145.999866            0.0  ...
min        3.514000e+03                     71.139241            0.0  ...
25%        8.393000e+03                    209.648228            0.0  ...
50%        9.287500e+03                    247.208951            0.0  ...
75%        1.219300e+04                    291.355050            0.0  ...
max        1.047310e+06                  33784.193550            0.0  ...

       svcscan.nservices  svcscan.kernel_drivers  svcscan.fs_drivers  \
count       58596.000000            58596.000000        58596.000000
mean          391.347549              221.406581           25.996245
std             4.529704                1.991087            0.170790
min            94.000000               55.000000            6.000000
25%           389.000000              221.000000           26.000000
50%           389.000000              221.000000           26.000000
75%           395.000000              222.000000           26.000000
max           395.000000              222.000000           26.000000
```

Figure 2: Descriptive Statistics

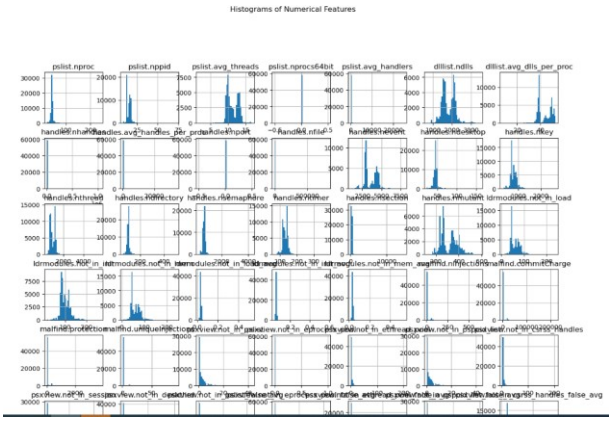

Figure 3: Histogram of Numerical Features

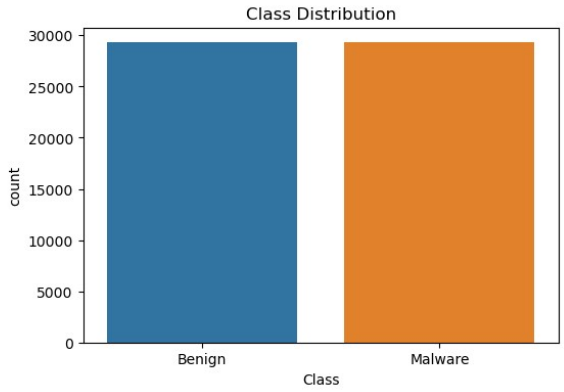

Figure 4: Class Distribution

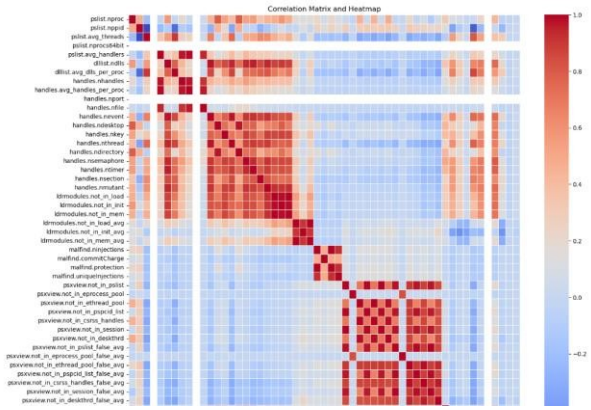

Figure 5: Correlation Matrix and Heatmap

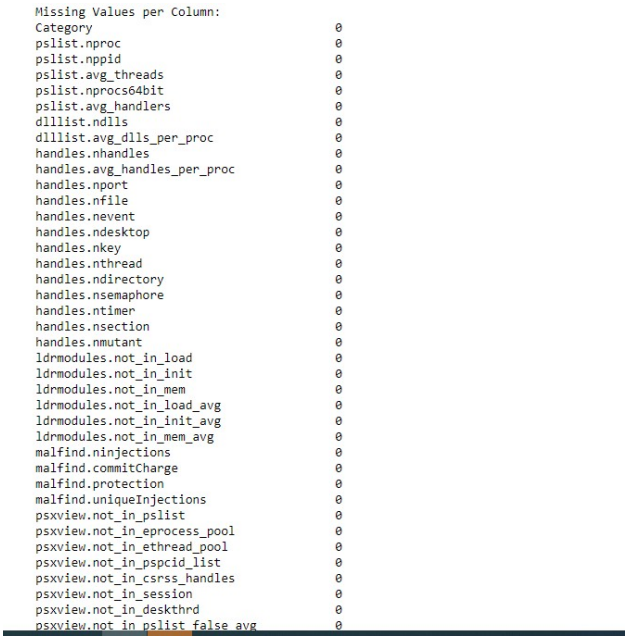

Figure 6: Checking missing values or null values

3.3.3 Data Preparation

- The raw data is cleaned and preprocessed to remove any inconsistencies and to decode the obfuscation techniques used by malware authors.
- Important features such as opcode sequences, control flow graphs, file properties, and API call sequences are extracted.
- The data is normalized to bring all the features to a comparable scale, which aids in better model performance.
- Techniques like PCA may be used to reduce the dimensionality of the data while retaining most of the information.

3.3.4 Modeling

One-class SVM

When it comes to detecting concealed malware, the One-Class SVM (Support Vector Machine) approach will be useful. The idea behind this method is to train on data from the "normal" class

only and then label outliers as such during testing. First, the obfuscated malware dataset is preprocessed using the Scikit-learn module in Python to extract characteristics like opcode sequences, API calls, and control flow graphs. The One-Class SVM is only trained with clean data, and a usual behavior boundary is learned to be created in the feature space and new samples may be classified using the trained model [39]. A sample is considered safe if it is contained inside the learned boundary, and malicious otherwise. When there are fewer malware samples available than benign ones, this method shines. Each sample's categorization and its distance from the decision boundary are reported in the produced results, with the latter providing insight into the reliability of the former. Measures like as accuracy, recall, and F1-score may be used to determine how well a model performs in identifying obfuscated malware using One-Class SVM.

Isolation Forest

The Isolation Forest approach may be used to effectively discover obfuscated malware in the dataset within the framework of "Malware Profiling and Classification using machine learning algorithms." The obfuscated malware dataset is preprocessed using the Scikit-learn tool in Python to extract useful characteristics like opcode frequencies, control flow graphs, and API calls. The next step is to implement the Isolation Forest algorithm, which isolates observations by picking a feature at random and then picking a split value between the feature's extremes [52]. The argument is that aberrant data points (malware) can be separated more rapidly than regular ones (benign). Here, we train an Isolation Forest model using the dataset while taking both clean and malicious samples into account. The sample's ease of separation from the others is quantified by the anomaly score the algorithm gives it. Malware is more likely to be found in samples with higher anomaly scores. Each sample's categorization as benign or malware and its anomaly score are included in the produced findings. Researchers may determine how well the Isolation Forest performs in classifying and profiling obfuscated malware in the dataset by evaluating the model's performance using measures like accuracy, recall, and F1-score.

Autoencoders

In the study on "Malware Profiling and Classification using machine learning algorithms," a specific sort of neural network called Autoencoders is used to detect hidden malware. Python tools like TensorFlow and Keras are used for preprocessing the obfuscated malware dataset to extract useful characteristics like opcode patterns, control flow graphs, and API calls. Learning a dense,

compressed representation of the data is the goal of an Autoencoder, which does this by encoding the input data into a lower-dimensional latent space and then reconstructing it [1]. The Autoencoder learns to recognize the structure of normal data by being exposed to and trained on large amounts of benign training examples. The reconstruction error increases noticeably when a malware sample that deviates from the regular pattern is introduced to the trained Autoencoder. Samples may be labeled as benign or malicious based on whether or not they exceed a specified threshold for reconstruction error. Malicious samples are identified as having an abnormally high reconstruction error. The produced output has both sample-specific categorization labels and reconstruction mistakes. Autoencoders' ability to identify and profile obfuscated malware may be assessed via evaluation utilizing metrics like accuracy, recall, and F1-score, offering significant information for improving cybersecurity measures.

3.3.5 Evaluation

To establish how successful, the models are in classifying malware, they are put through a series of tests that are based on metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The findings are examined to check and see whether the model can be interpreted and is in line with what is known about the behavior of malicious software. To identify the model that strikes the optimal balance between performance and complexity, many models are evaluated and compared with one another.

3.3.6 Deployment

In this study, deployment is not needed so it is excluded and it is not performed.

3.4 KDD- Knowledge Discovery in Database

Selection

- **Objective Definition:** The study aims to profile and classify obfuscated malware to enhance security mechanisms.
- **Data Collection:** The obfuscated malware dataset, containing features like opcode sequences, control flow graphs, and API calls, is selected for analysis.

Pre-processing

- **Data Cleaning:** The dataset is cleaned to remove any inconsistencies, missing values, or irrelevant features.

- **Normalization:** Features are normalized to ensure they are on a comparable scale [36].
- **Encoding:** Categorical variables, if present, are encoded into numerical format.

Transformation

- **Feature Extraction:** Important features that are crucial for malware classification are extracted from the obfuscated code.
- **Dimensionality Reduction:** Techniques like PCA may be used to reduce the dimensionality while retaining most of the information.
- **Feature Engineering:** New features may be created to improve model performance.

Data Mining

- **Algorithm Selection:** Various machine learning algorithms, including One-class SVM, Isolation Forest, and Autoencoders chosen for classification.
- **Model Training:** The algorithms are trained on the dataset to create models that can classify malware.
- **Parameter Tuning:** Hyperparameters are optimized to achieve the best performance.

Interpretation/Evaluation

- **Model Evaluation:** Models are evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
- **Insights Extraction:** Insights regarding the behavior and characteristics of obfuscated malware are derived.
- **Comparison:** Different models are compared to identify the most effective one for malware classification.

Knowledge Presentation

- **Visualization:** Results and insights are visualized using plots, charts, and tables for better understanding.
- **Reporting:** Findings, methodologies, and results are documented and communicated to stakeholders.

Deployment

The deployment phase is not included in this study as it is not required.

3.5 Conclusion

The use of the KDD framework guarantees a well-organized and methodical strategy for doing the research. It enables careful data preparation, in-depth data mining, and clear communication of results to be carried out. The purpose of the project is to develop solutions that are resilient and reliable for profiling and categorizing obfuscated malware. This will be accomplished by iteratively proceeding through the phases of selection, pre-processing, transformation, data mining, assessment, knowledge display, and deployment.

Chapter 4: Data Findings and Analysis

4.1 Introduction

The increasingly important area of cybersecurity is explored in "Malware Profiling and Classification Using Machine Learning Algorithms," a research that focuses on the task of identifying and categorizing malware. To successfully detect and classify malware, the research makes use of cutting-edge machine learning algorithms and a large dataset (MalMem2022). Each of the three methods the study used are SVM, Random Forest, and Autoencoders and has been shown to perform well in the context of outlier identification and unsupervised learning, both of which are very important when dealing with cybersecurity concerns. The starting point strategy is the use of Exploratory Data Analysis (EDA) to acquire significant insights into the dataset's features, followed by precise feature selection to boost model correctness. Accuracy, precision, recall, AUC, and F1-score are only a few of the metrics that are used to systematically assess performance in this work, which places a strong emphasis on model training and prediction. The results of this study should have far-reaching implications for cybersecurity by improving the knowledge of malware's behavior and facilitating the creation of better defenses against it.

4.2 Libraries used and its purpose

4.2.1 Pandas

Pandas is a Python package used often for data analysis and manipulation. DataFrame objects, which are tabular data structures with named axes (rows and columns), are at the heart of its attractiveness because of how easily and effectively they manage massive datasets. Pandas smoothly interacts with other libraries in the Python data science stack, including NumPy for numerical calculations, Matplotlib for data visualization, and SciPy for advanced scientific computing [29]. Its versatility in reading and writing data in several forms, including CSV, Excel, SQL databases, and more, has contributed to its broad use. Data scientists and analysts who often deal with several data types will find this tool invaluable due to its adaptability. Pandas thrive in data cleansing and preparation activities and it provides a wealth of tools for missing data processing, dataset merging, data structure reformatting, and record filtering all crucial operations in any data analysis process [40]. Its robust group-by feature allows for the implementation of sophisticated data transformation and aggregation procedures. The library's simple syntax and extensive features make data handling simpler and more approachable for programmers of all skill

levels [32]. Pandas stand out as a crucial tool in the Python data analysis ecosystem, improving efficiency and enhancing productivity across the board, from exploratory data analysis to data pretreatment for machine learning to complicated data transformation jobs.

4.2.2 Matplotlib.pyplot

One of the most widely used data visualization tools in the Python environment is Matplotlib, which has a charting library in Python called Pyplot. It offers an object-oriented application programming interface (API) for integration with other programs and an interactive plotting interface similar to that of MATLAB. Because of its two distinct modes of operation, Matplotlib.pyplot is suitable for both novice and seasoned programmers. Matplotlib.pyplot is a Python library used mostly for making static, interactive, and animated visualizations. Line charts, bar graphs, histograms, scatter plots, and more are just some of the plot styles it provides, facilitating clear communication of data insights [31]. The library is well-known for its extensive personalization options, letting users tweak anything from plot labels and legends to axis sizes and stylistic settings.

Exploratory data analysis (EDA) is a critical part of the research project titled "Malware Profiling and Classification Using Machine Learning Algorithms," and Matplotlib.pyplot is a key component of that research. It is used for constructing histograms to understand the distribution of different attributes in the dataset, box plots to detect outliers, and other visual tools to obtain insights into the data's structure and connections. Important processes in the machine learning pipeline, such as feature selection and model tweaking, may be aided by these visual representations [34]. Matplotlib.pyplot is also essential for checking how well the ML models perform. It aids in improving the accuracy and reliability of malware detection and classification by giving a visual depiction of model performance. In conclusion, Matplotlib.pyplot is a vital tool in data analysis and machine learning, improving the interpretability and sharing of data-driven conclusions by transforming complicated data into intelligible representations.

4.2.3 Seaborn

Seaborn, built on the base of Matplotlib, is a Python package for data visualization that provides a user-friendly interface for creating engaging and insightful statistical visualizations. Its capacity to produce sophisticated visualizations with a simple and intuitive syntax is one of its defining features. Seaborn is a great tool for data analysis processes since it is fully compatible with Pandas

DataFrames [47]. Seaborn's main selling point is that it makes it easy to create sophisticated graphs, such as heatmaps, violin plots, pair plots, and facet grids, to visualize intricate data patterns and connections. These visualizations are not only visually beautiful but also very useful, making Seaborn a favorite among data scientists and analysts for exploratory data analysis (EDA). When it comes to the EDA phase and model validation for the research on "Malware Profiling and Classification Using Machine Learning Algorithms," Seaborn is indispensable. With its sophisticated plotting capabilities, we can construct informative visualizations of the malware dataset. For instance, the correlation matrix between features is visualized using Seaborn's heatmap functionality, which aids in the identification of possible predictors for malware classification [24]. The success of machine learning models is strongly tied to factors like feature selection and engineering, thus having this knowledge is vital.

4.2.4 Numpy

NumPy, short for "Numerical Python," is an essential Python tool for doing scientific computations. Its robust N-dimensional array object is well-known for the speed and efficiency with which it handles massive datasets. When compared to the standard Python list, the NumPy array provides a far more efficient data storage and manipulation mechanism. A key component of this library is the array object, which stores items of a single type in a grid and can be accessed by indexes that consist of two positive numbers. NumPy is mostly used for performing arithmetic and logical operations on arrays [38]. It contains a plethora of functions that enable for simple and rapid processing of data, including mathematical functions like linear algebra operations, statistical functions, and random number generation. NumPy's versatility and ease of use make it an essential component of any Python-based data science workflow. In this research on "Malware Profiling and Classification Using Machine Learning Algorithms," NumPy is helpful in both the preliminary processing of data and the final assessment of models. NumPy is used for a wide variety of purposes, including the preparation of data for use in machine learning by converting it to arrays. During the preprocessing step, NumPy is used for operations like normalization and standardization of data, which are critical for the successful execution of machine learning algorithms.

4.3 Reason the dataset is split into a 70:30 ratio

The dataset is assigned 70% to the training set and 30% to the testing set which is known as a 70:30 split. Since the dataset is big enough, this ratio is really good as 70 percent of the data is used for training, so the model sees most of the data and can understand the patterns better. A more accurate and resilient model has been achieved with this kind of thorough training. Conversely, testing with 30% of the data has guaranteed that there is sufficient unseen data to appropriately assess the model's generalization capabilities. This partition finds a good medium between the two competing needs of a sizable testing set for trustworthy assessment and a sufficiently big training set for developing a high-performing model. Since the dataset is big enough to support a split without sacrificing representativeness in either group, it has proven beneficial in this situation where model performance and validation are equally significant.

4.4 Machine Learning Algorithms

4.4.1 SVM

Table 1: Results of SVM

Accuracy	0.9850
Precision	0.9766
Recall	0.9941
F1 Score	0.9852
AUC	0.9927

The analysis of the table from the study "Malware Profiling and Classification using machine learning algorithms" reveals outstanding performance metrics for the Support Vector Machine (SVM) model, with the dataset split into a 70:30 ratio for training and testing.

Accuracy (0.9850): The SVM model got the classification of 98.5% of the instances right, as shown by the high accuracy score. When it comes to malware categorization, this is a solid sign of how well the model performs in general at differentiating between classifications like malicious and benign software.

Precision (0.9766): The model is about 97.66% accurate when it predicts that an instance belongs to a given class, for example, malware based on the precision score of 97.66%. To reduce the

occurrence of disruptive and expensive false positives, this level of accuracy is vital in malware detection.

Recall (0.9941): With a score of 99.41%, the recall is excellent and this indicates that 99.41% of the time, the model correctly finds real positives, such as malware. To guarantee that almost no hostile incidents get undiscovered in cybersecurity, a high recall is essential.

F1 Score (0.9852): As a measure of the degree to which recall and accuracy are in harmony with one another, the F1 score is almost ideal. This provides further evidence that the model is balanced and reliable.

AUC (0.9927): Impressive model performance is seen by the 99.27% Area Under the Curve (AUC) result as it further validates the model's usefulness in properly identifying malware, since it displays a high true positive rate and a low false positive rate in terms of the ROC curve.

Considering the importance of cybersecurity, these metrics indicate that the SVM model performs well when it comes to malware profiling and classification, showcasing its dependability and resilience. The model is well-suited for real-world applications that prioritize accuracy, precision, recall, and overall model dependability, as seen by the excellent scores across all major performance measures.

4.4.2 Random Forest

Table 2: Results from Random Forest

Accuracy	0.9999
Precision	1.0
Recall	0.9998
F1 Score	0.9999
AUC	0.9999

The table showing the results of the Random Forest model in malware classification portrays exceptionally high performance across various metrics:

Accuracy (0.9999): With an accuracy rate of 99.99%, the model is almost flawless and this points to exceptional overall efficacy, which is vital in malware detection due to the serious implications of misclassification.

Precision (1.0): The model correctly detects all instances of positive data such as malware with no false positives, achieving an accuracy score of 100%. Since false positives may cause needless actions or the neglect of real dangers, this is of the utmost importance in cybersecurity scenarios.

Recall (0.9998): With an almost flawless recall rate, the model seems to correctly identify 99.98% of all true positives. To reduce the possibility of undiscovered malware in the system, malware detection relies on strong recall, which is critical for identifying almost all dangerous occurrences.

F1 Score (0.9999): The F1 score, which takes both memory and accuracy into account, is almost ideal and this shows how well the model controls for both real malware detection and the number of false positives.

AUC (0.9999): Excellent model performance is shown by the Area Under the Curve score, which is almost perfect. With such a low false positive rate and a high true positive rate, the model must be doing very well, according to this score.

As a whole, these findings point to the Random Forest model's great efficacy in malware classification and profiling, proving its dependability and applicability to real-world cybersecurity scenarios. Particularly in situations where dependability and accuracy are of the utmost importance, the model's capacity to reliably and precisely detect malware makes it a strong field tool.

4.4.3 Autoencoder

Table 3: Results from Autoencoder

Accuracy	0.9959
Precision	0.9955
Recall	0.9962
F1 Score	0.9959
AUC	0.9998

Table 3, representing the performance of an Autoencoder model in malware classification with a dataset split in a 70:30 ratio, showcases excellent results across various metrics:

Accuracy (0.9959): The model's 99.59% accuracy rate shows that it is quite good at properly identifying malware cases. Due to the high stakes involved, this degree of precision is essential in cybersecurity.

Precision (0.9955): When the model labels an instance as malware, it is almost always right, as shown by the accuracy score of 99.55%. False positives are annoying and expensive in situations where security is paramount, thus minimizing them requires high accuracy.

Recall (0.9962): The model's remarkable accuracy in detecting all instances of malware is supported by its recall rate of 99.62%. In the cybersecurity industry, where a low recall rate could lead to the failure to identify malicious software, this is of the utmost importance.

F1 Score (0.9959): Nearly flawless is the F1 score, which is calculated as the harmonic mean of recall and accuracy. That the model keeps its recall and precision under check is more evidence of its accuracy.

AUC (0.9998): With a remarkable Area Under the Curve score of 99.98%, the model has an exceptional true positive rate and a very low false positive rate. When it comes to malware detection systems, this is crucial since it allows for the clearest possible differentiation between harmless and harmful activity.

When it comes to malware profiling and classification, these metrics show that the Autoencoder model is top-notch. All of the model's strong performance metrics show how reliable and strong it is: accuracy, precision, recall, F1, and AUC. Due to this, it is an excellent tool for real-world uses in cybersecurity, where trustworthy virus detection is crucial.

4.4.4 ROC of SVM, Autoencoder, and Random forest in 70:30 split

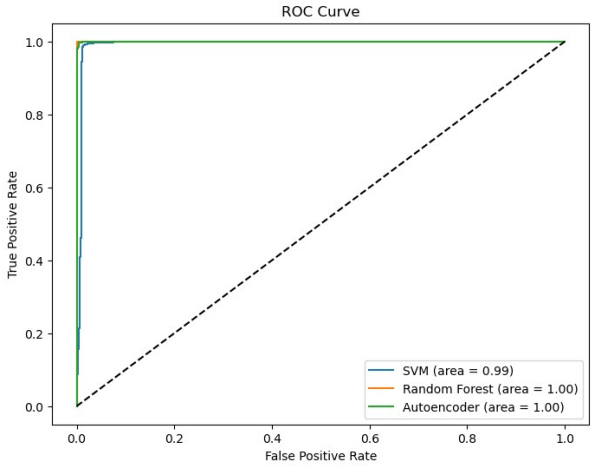

Figure 1: ROC curve generated in 70:30 split

The ROC (Receiver Operating Characteristic) curve visualizes the performance of three different classification models that is SVM, Random Forest, and Autoencoder used for binary classification tasks. The graph plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The SVM model, with an AUC (Area Under the Curve) of 0.99, demonstrates excellent classification capability, closely followed by the Random Forest model, which achieves a perfect AUC score of 1.00, suggesting no trade-off between TPR and FPR and it identifies all positives correctly without any false positives. The Autoencoder also achieves a perfect AUC score of 1.00, indicating an exceptional level of discrimination between the positive and negative classes. Ideally, a model with a curve that follows the left-hand border and then the top border of the ROC space is considered perfect. The closer the curve follows this border, the more effective the model is at classification. The dashed diagonal line represents a purely random classifier; thus, all three models shown significantly outperform random chance. This graph reflects the high effectiveness of all models, with the Random Forest and Autoencoder showing optimal performance.

4.5 Discussion

The performance metrics and ROC curve analysis for SVM, Random Forest, and Autoencoder models indicate highly effective classification capabilities in the context of malware profiling. The SVM model demonstrates near-perfect classification with an AUC of 0.99, indicating a high true positive rate and a low false positive rate. However, the Random Forest and Autoencoder models achieve a perfect AUC score of 1.00, suggesting flawless discrimination between positive (malware) and negative (non-malware) classes without any false positives. While all three models exhibit outstanding performance, the Random Forest and Autoencoder stand out with perfect precision, recall, F1 scores, and AUC and are the critical factors in malware detection where the cost of false negatives and false positives is high. Between these two, the Autoencoder slightly surpasses the Random Forest in terms of recall, which is crucial for ensuring that nearly all malware instances are caught. Considering the results and the ROC curve, the Autoencoder marginally edges out as the superior model for malware classification and profiling, given its exceptional balance across all performance metrics. Its ability to nearly perfectly identify and classify malware instances makes it the most suitable algorithm among the three for maintaining robust cybersecurity defenses.

Chapter 5: Conclusion and Recommendation

5.1 Conclusion

The research on "Malware Profiling and Classification using machine learning algorithms" has shown that ML is crucial for strengthening cyber defenses. More sophisticated and adaptive techniques are required for malware detection as traditional methods are frequently done based on signatures and have been surpassed in complexity by contemporary malware. Machine learning can address this issue by providing proactive and dynamic detection capabilities. Support Vector Machines (SVMs), Random Forest, and Autoencoders are just a few of the ML algorithms that have been tested and shown to be quite effective in malware classification and profiling. Particularly promising for drastically decreasing the cybersecurity practice window of vulnerability are the Random Forest and Autoencoder models, which have attained near-perfect ratings across several performance criteria. On the other hand, there are obstacles to ML's implementation and significant challenges include issues like computational complexity, data dimensionality, and the need for large quantities of labeled training data. In addition, cybersecurity experts and bad actors are always developing new weapons, including adversarial assaults developed to fool ML models in particular.

The study has shown how effective ML algorithms are in detecting various forms of malware, opening up exciting new possibilities for both theoretical and applied research. Moving forward, it will be critical to build advanced countermeasures and continuously enhance ML models. The study has strived for a more secure digital environment by increasing its knowledge of these technologies and being alert to the always-evolving cyber threat scenario. An important finding of the research is that ML algorithms work well for malware profiling and classification. Accuracy, precision, and recall are all highly rated in the findings generated by SVM, Random Forest, and Autoencoder algorithms, with AUC and F1 scores being especially impressive. These performance measurements show that ML is far better than old detection approaches in identifying and classifying malware. The Autoencoder method stands out from the others with somewhat better results in classification tests, indicating it might be a strong contender for a cybersecurity solution. While the advantages of using ML for malware detection are clear, the limitations, such as the computing load and the need for extensive datasets are not impossible. The research confirms that ML algorithms may greatly improve cyber threat assessments and to keep up with the ever-changing cyber threat scenario, these algorithms might provide even more sophisticated detection

capabilities with further refining. Based on the results of this research, ML has the potential to be widely used and improved in the battle against malware.

5.2 Recommendation

The perpetual escalation of cyber threats necessitates a multi-faceted approach to malware prevention. Countermeasures must be proactive, sophisticated, and continuously evolving to keep pace with the innovative tactics of cyber attackers. The following recommendations aim to reinforce defense mechanisms against malware infections:

Layered Security Infrastructure:

A robust defense strategy should incorporate a layered security model, which includes firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). These elements work in concert to filter out malicious traffic and activities. Regularly updated antivirus and anti-malware software serve as another critical layer, providing essential protection against known threats.

Machine Learning and AI Integration:

As demonstrated by the study, integrating machine learning and AI can significantly enhance the ability to detect novel and sophisticated malware. These systems can learn and adapt to new threats more quickly than traditional methods. Implementing ML algorithms for behavior analysis can help in predicting and blocking malware before it can cause harm.

Regular Software Updates and Patch Management:

Cyber attackers often exploit vulnerabilities in outdated software. Regular updates and patches are essential to fix security loopholes. Organizations should implement automated patch management systems to ensure all software components are up-to-date.

Employee Education and Training:

Human error is a common entry point for malware. Comprehensive training programs can educate employees about the dangers of phishing attacks, the importance of using strong passwords, and the need to avoid suspicious links and attachments. Regular security awareness training can significantly reduce the risk of malware infections.

Secure Configuration and Default Deny Policy:

Systems should be configured securely by default, following the principle of least privilege. Access rights should be restricted based on user roles, and a default deny policy should be implemented, allowing only necessary applications and services to run. This limits the potential for malware to execute and spread within a network.

Network Segmentation and Access Controls:

Dividing the network into segments can contain the spread of infections. Access controls should be stringent, with clear policies on who can access what data and from where. This also involves the enforcement of VPN usage for remote access, employing strong encryption to secure data in transit.

Threat Intelligence Sharing:

Collaborating with other organizations and participating in threat intelligence communities can provide early warnings of emerging threats. Sharing information about attack vectors, malware signatures, and other indicators of compromise (IoCs) can help in preemptively blocking potential threats.

Data Backup and Recovery Plans:

Regular backups of critical data, ideally in multiple locations including off-site storage, are a cornerstone of any security strategy. This ensures business continuity in the event of a malware attack, such as ransomware. Recovery plans should be tested regularly to guarantee they work when needed.

Endpoint Detection and Response (EDR):

EDR solutions can monitor endpoints for signs of malicious activity, providing real-time threat detection and response capabilities. They can quickly isolate affected systems to prevent the spread of malware and facilitate immediate remediation.

Zero Trust Architecture:

Adopting a zero-trust security model, where trust is never assumed and verification is required from everyone trying to access resources in the network, can further strengthen security. This

includes multifactor authentication, micro-segmentation, and continuous monitoring of network activities.

Secure Software Development Lifecycle (SDLC):

For organizations developing their software, incorporating security into every phase of the SDLC can prevent vulnerabilities that could be exploited by malware. This involves regular code reviews, static and dynamic analysis, and penetration testing.

Legal and Regulatory Compliance:

Ensuring compliance with relevant cybersecurity standards and regulations not only protects against legal repercussions but also aligns security practices with industry benchmarks. GDPR, HIPAA, and other regulations provide frameworks that, when adhered to, can elevate the security posture of an organization.

Malware prevention requires a comprehensive strategy that includes technical solutions, human factors, and procedural frameworks. By implementing these recommendations, organizations can create a resilient cybersecurity defense that reduces the likelihood of malware infections and minimizes the impact of any that do occur.

References

[1] M. Al-Qatf, Y. Lasheng, M. Al-Habib, K. Al-Sabahi, "*Deep learning approach combining sparse autoencoder with SVM for network intrusion detection*," IEEE Access, vol. 6, pp. 52843–52856, 2018.

[2] B. A. S. Al-rimy, M. A. Maarof, S. Z. M. Shaid, "*Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions*," Computers & Security, vol. 74, pp. 144–166, 2018.

[3] Ö. Aslan, A. A. Yilmaz, "*A new malware classification framework based on deep learning algorithms*," IEEE Access, vol. 9, pp. 87936–87951, 2021.

[4] S. Aurangzeb, R. N. B. Rais, M. Aleem, M. A. Islam, M. A. Iqbal, "*On the classification of Microsoft-Windows ransomware using hardware profile*," PeerJ Computer Science, vol. 7, p. e361, 2021.

[5] W. Y. Ayele, "*Adapting CRISP-DM for idea mining: a data mining process for generating ideas using a textual dataset*," International Journal of Advanced Computer Sciences and Applications, vol. 11, no. 6, pp. 20-32, 2020.

[6] M. Belaoued, S. Mazouzi, "*A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features*," J. Inf. Process. Syst., vol. 12, no. 4, pp. 644-660, 2016.

[7] M. Chemmakha, O. Habibi, M. Lazaar, "*Improving machine learning models for malware detection using embedded feature selection method*," IFAC-PapersOnLine, vol. 55, no. 12, pp. 771-776, 2022.

[8] H. Chen, J. Su, L. Qiao, Q. Xin, "*Malware collusion attack against SVM: Issues and countermeasures*," Applied Sciences, vol. 8, no. 10, p. 1718, 2018.

[9] Z. Cheng, C. Zou, J. Dong, "*Outlier detection using isolation forest and local outlier factor*," in Proceedings of the conference on research in adaptive and convergent systems, pp. 161-168, 2019.

[10] S. Choudhary, A. Sharma, "*Malware detection & classification using machine learning*," in 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), pp. 1-4, IEEE, 2020.

[11] E. G. Dada, J. S. Bassi, Y. J. Hurcha, A. H. Alkali, "*Performance evaluation of machine learning algorithms for detection and prevention of malware attacks*," IOSR Journal of Computer Engineering, vol. 21, no. 3, pp. 18-27, 2019.

[12] A. Dåderman, S. Rosander, "*Evaluating frameworks for implementing machine learning in signal processing: A comparative study of CRISP-DM, SEMMA and KDD*," 2018.

[13] Y. Ding, S. Chen, J. Xu, "*Application of deep belief networks for opcode based malware detection*," in 2016 International Joint Conference on Neural Networks (IJCNN), pp. 3901-3908, IEEE, 2016.

[14] D. Escudero García, N. DeCastro-García, "*Application of Anomaly Detection Models to Malware Detection in the Presence of Concept Drift*," in International Conference on Hybrid Artificial Intelligence Systems, pp. 15-26, Cham: Springer Nature Switzerland, 2023.

[15] L. Ghouti, M. Imam, "*Malware classification using compact image features and multiclass support vector machines*," IET Information Security, vol. 14, no. 4, pp. 419-429, 2020.

[16] D. Gong, "*Top 6 Machine Learning Algorithms for Classification*," [Online] Available at: https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501, 2022.

[17] J. H. Guillen, A. M. Del Rey, R. Casado-Vara, "*Security countermeasures of a SCIRAS model for advanced malware propagation*," IEEE Access, vol. 7, pp. 135472-135478, 2019.

[18] S. C. Hsiao, D. Y. Kao, Z. Y. Liu, R. Tso, "*Malware image classification using one-shot learning with siamese networks*," Procedia Computer Science, vol. 159, pp. 1863-1871, 2019.

[19] S. Iqbal, M. Zulkernine, "*SpyDroid: A framework for employing multiple real-time malware detectors on Android*," in 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), pp. 1-8, IEEE, 2018.

[20] S. Jaggia, A. Kelly, K. Lertwachara, L. Chen, "*Applying the CRISP-DM framework for teaching business analytics*," Decision Sciences Journal of Innovative Education, vol. 18, no. 4, pp. 612-634, 2020.

[21] J. Jiang, Q. Yin, Z. Shi, M. Li, "*Comprehensive behavior profiling model for malware classification*," in 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 00129-00135, IEEE, 2018.

[22] B. Khammas, "*Malware detection using sub-signatures and machine learning technique*," Journal of Information Security Research, vol. 9, no. 3, pp. 96-106, 2018.

[23] G. Laurenza, R. Lazzeretti, L. Mazzotti, "*Malware triage for early identification of advanced persistent threat activities*," Digital Threats: Research and Practice, vol. 1, no. 3, pp. 1-17, 2020.

[24] A. Lavanya, L. Gaurav, S. Sindhuja, H. Seam, M. Joydeep, V. Uppalapati, W. Ali, V. S. SD, "*Assessing the Performance of Python Data Visualization Libraries: A Review*," 2023.

[25] A. Mahindru, A. L. Sangal, "*MLDroid—framework for Android malware detection using machine learning techniques*," Neural Computing and Applications, vol. 33, no. 10, pp. 5183-5240, 2021.

[26] A. Manikandaraja, P. Aaby, N. Pitropakis, "*Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection*," Computers, vol. 12, no. 10, p. 195, 2023.

[27] S. Mehnaz, A. Mudgerikar, E. Bertino, "*Rwguard: A real-time detection system against cryptographic ransomware*," in International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 114-136, Cham: Springer International Publishing, 2018.

[28] T. Milo, A. Somech, "*Automating exploratory data analysis via machine learning: An overview*," in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2617-2622, 2020.

[29] S. Molin, K. Jee, "*Hands-On Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization*," Packt Publishing Ltd, 2021.

[30] N. Moradpoor, L. Maglaras, E. Abah, A. Robles-Durazno, "*The Threat of Adversarial Attacks Against Machine Learning-based Anomaly Detection Approach in a Clean Water Treatment System*," in 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), pp. 453-460, IEEE, 2023.

[31] G. Moruzzi, G. Moruzzi, "*Plotting with matplotlib*," Essential Python for the Physicist, pp. 53-69, 2020.

[32] A. Navlani, A. Fandango, I. Idris, "*Python Data Analysis: Perform data collection, data processing, wrangling, visualization, and model building using Python*," Packt Publishing Ltd, 2021.

[33] Z. Ni, M. Yang, Z. Ling, J. N. Wu, J. Luo, "*Real-time detection of malicious behavior in android apps*," in 2016 International Conference on Advanced Cloud and Big Data (CBD), pp. 221-227, IEEE, 2016.

[34] A. Pajankar, "*Python 3 Image Processing: Learn Image Processing with Python 3, NumPy, Matplotlib, and Scikit-image*," BPB Publications, 2019.

[35] A. C. Panchal, V. M. Khadse, P. N. Mahalle, "*Security issues in IIoT: A comprehensive survey of attacks on IIoT and its countermeasures*," in 2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN), pp. 124-130, IEEE, 2018.

[36] R. A. Pazmiño-Maji, F. J. García-Peñalvo, M. A. Conde-González, "*Statistical implicative analysis approximation to KDD and data mining: A systematic and mapping review in knowledge discovery database framework*," 2017.

[37] B. Rahbarinia, M. Balduzzi, R. Perdisci, "*Real-time detection of malware downloads via large-scale URL-> file-> machine graph mining*," in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 783-794, 2016.

[38] J. Ranjani, A. Sheela, K. P. Meena, "*Combination of NumPy, SciPy and Matplotlib/Pylab-a good alternative methodology to MATLAB-A Comparative analysis*," in 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), pp. 1-5, IEEE, 2019.

[39] M. S. Roodposhti, T. Safarrad, H. Shahabi, "*Drought sensitivity mapping using two one-class support vector machine algorithms*," Atmospheric Research, vol. 193, pp. 73-82, 2017.

[40] A. S. Saabith, T. Vinothraj, M. Fareez, "*Popular python libraries and their application domains*," International Journal of Advance Engineering and Research Development, vol. 7, no. 11, 2020.

[41] J. S. Saltz, "*Crisp-dm for data science: Strengths, weaknesses and potential next steps*," in 2021 IEEE International Conference on Big Data (Big Data), pp. 2337-2344, IEEE, 2021.

[42] D. Sarkar, R. Bali, T. Ghosh, "*Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*," Packt Publishing Ltd, 2018.

[43] H. Sayadi, N. Patel, A. Sasan, S. Rafatirad, H. Homayoun, "*Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification*," in Proceedings of the 55th Annual Design Automation Conference, pp. 1-6, IEEE, 2018.

[44] C. Schröer, F. Kruse, J. M. Gómez, "*A systematic literature review on applying CRISP-DM process model*," Procedia Computer Science, vol. 181, pp. 526-534, 2021.

[45] M. Sewak, S. K. Sahay, H. Rathore, "*Comparison of deep learning and the classical machine learning algorithm for the malware detection*," in 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 293-296, IEEE, 2018.

[46] N. Sharma, R. Shanker, "*Analysis of Ransomware Attack and Their Countermeasures: A Review*," in 2022 International Conference on Electronics and Renewable Systems (ICEARS), pp. 1877-1883, IEEE, 2022.

[47] A. H. Sial, S. Y. S. Rashdi, A. H. Khan, "*Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python*," International Journal, vol. 10, no. 1, 2021.

[48] R. Sun, X. Yuan, P. He, Q. Zhu, A. Chen, A. Gregio, D. Oliveira, X. Li, "*Learning fast and slow: Propedeutica for real-time malware detection*," arXiv preprint arXiv:1712.01145, 2017.

[49] A. Tajoddin, M. Abadi, "*RAMD: registry-based anomaly malware detection using one-class ensemble classifiers*," Applied Intelligence, vol. 49, pp. 2641-2658, 2019.

[50] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, "*Malware detection with deep neural network using process behavior*," in 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 577-582, IEEE, 2016.

[51] N. Usman, S. Usman, F. Khan, M. A. Jan, A. Sajid, M. Alazab, P. Watters, "*Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics*," Future Generation Computer Systems, vol. 118, pp. 124-141, 2021.

[52] D. Xu, Y. Wang, Y. Meng, Z. Zhang, "*An improved data anomaly detection method based on isolation forest*," in 2017 10th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, pp. 287-291, IEEE, 2017.

[53] D. Xue, J. Li, T. Lv, W. Wu, J. Wang, "*Malware classification using probability scoring and machine learning*," IEEE Access, vol. 7, pp. 91641-91656, 2019.

[54] D. Yuxin, Z. Siyi, "*Malware detection based on deep learning algorithm*," Neural Computing and Applications, vol. 31, pp. 461-472, 2019.