1-10-2023

MACHINE LEARNING MODELS INTERPRETABILITY FOR MALWARE DETECTION USING MODEL AGNOSTIC LANGUAGE FOR EXPLORATION AND EXPLANATION

Ikuromor Mabel Ogiriki
Rowan University

MACHINE LEARNING MODELS INTERPRETABILITY FOR MALWARE DETECTION USING MODEL AGNOSTIC LANGUAGE FOR EXPLORATION AND EXPLANATION

by
Ikuromor Mabel Ogiriki

A Thesis

Submitted to the
Department of Computer Science
College of Science and Mathematics
In partial fulfillment of the requirement
For the degree of
Master of Science in Computer Science
at
Rowan University
December 8, 2022

Thesis Chair: Vahid Heydari, Ph.D., Associate Professor, Department of Computer Science

Committee Members:
Shen-Shyang Ho, Ph.D., Associate Professor, Department of Computer Science
Silvija Kokalj-Filipovic, Ph.D., Professor, Department of Computer Science

Dedication

I would like to dedicate this to all black women in STEM, and more specifically, Immigrant Nigerian women. Who work extremely hard and strive to be trailblazers in a foreign country to create a mark and make this world a better place.

Acknowledgements

I would like to thank my advisor, who has been of immense support throughout my entire program. I am especially grateful for his guidance, patience and push for not only the completion of this work but throughout my academic journey. This body of work would not have been possible without his constant input, constructive feedback, and keen patience in the review of many drafts of my papers. Thank you, Dr. Heydari. Your constant support helped boost my confidence and improved my writing tremendously.

I would also like to appreciate Dr. Ho and Dr. Silvija, who were also very key in supporting my journey both as a coordinator of the MS program and as a member of my committee. Your keenness to provide support in any way possible and push to expand the scope of this work is invaluable.

My sincere gratitude goes to my Family, my Partner and my friends whose constant love, encouragement, unwavering support and unshaken belief in my ability challenged me to put my best foot forward every step of the way. This Journey was a lot easier because I had them cheering for me.

Finally, I would like to give all thanks to God, for his provision, mercies and unending favor. Without him, none of this would have been possible.

Abstract

Ikuromor Mabel Ogiriki
MACHINE LEARNING MODELS INTERPRETABILITY FOR MALWARE DETECTION USING MODEL AGNOSTIC LANGUAGE FOR EXPLORATION AND EXPLANATION
2022-2023
Vahid Heydari, Ph.D.
Master of Science in Computer Science

The adoption of the internet as a global platform has birthed a significant rise in cyber-attacks of various forms ranging from Trojans, worms, spyware, ransomware, botnet malware, rootkit, etc. In order to tackle the issue of all these forms of malware, there is a need to understand and detect them. There are various methods of detecting malware which include signature, behavioral, and machine learning. Machine learning methods have proven to be the most efficient of all for malware detection.

In this thesis, a system that utilizes both the signature and dynamic behavior-based detection techniques, with the added layer of the machine learning algorithm with model explainability capability is proposed. This hybrid system provides not only predictions, but also their interpretation and explanation for a malware detection task. The layer of machine learning algorithm can be Logistic Regression, Random Forest, Naive Bayes, Decision Tree, or Support Vector Machine. Empirical performance evaluation results on publicly available datasets and manually acquired samples (both benign and malicious) are used to compare the five machine learning algorithms. DALEX (moDel Agnostic Language for Exploration and explanation) is integrated into the proposed hybrid approach to support the interpretation and understanding of the prediction to improve the trust for cyber security stakeholders on complex machine learning predictive model.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

List of Figures

Figure Page

List of Figures (Continued)

List of Tables

Table Page

Chapter 1

Introduction

In recent years, there has been a growing number of attacks using malware by cyber attackers. This malware can be trojans, viruses, worms, etc., that can attack the computer through emails, various malicious websites, software, and drives that the users have downloaded. Over the years, malware has grown not only in the number or the volume in which they appear but also in numerous types that perform various functionalities. Ransomware is a type of malware where cyber attackers encrypt users' files until a ransom is paid. This has become the most financially beneficial form of malware (Cisco, 2016) because it has successfully recorded at least 40% of ransom payments. Even though these payments have been made, there are few or no guarantees that the file can be removed. These encryptions made to files are not only done to local files. Sometimes it extends to the whole organization. Since many organizations share volumes of documents to enable teamwork, therefore exposing other computers to the risk of being infected by the already infected computer. This is harmful to many organizations because in order to contain and fix these issues, most businesses must stop their operations until they can properly clean all the affected systems and restore their backup image, which can lead to loss of money and valuable work time.

As malware has increased, various research methods have analyzed malicious and non-malicious malware samples using static and dynamic tools. These dynamic tools include Process explorer (Gandotra et al., 2014), ProcMon (Bidoki et al., 2016), Wireshark/ T shark (Ndatinya et al., 2015), TCPDump (Hoque et al., 2014), TCPview (Eilam, 2005), sandboxes (Ali et al., 2018), and many more. One of these very effective methods has been

using machine learning algorithms to set up an analysis environment using these static tools. Examples of static analysis tools include PEView (Sikorski & Honig, 2012), PEid (Wang & Wu, 2011), CFF explorer (Abimannan & Kumaravelu, 2019), disassemblers (Sikorski & Honig, 2012), PsFile (Abdessadki & Lazaar, 2019), and so much more.

We can record how efficiently these systems are used to detect this malware based on their effective ways of extracting features that are generally classified as malicious. One major problem with detecting some of this malware is that most of the recently developed have succeeded in escaping various malware analysis and detection systems used to previously detect and classify these malware (Singh & Singh, 2018; Gao et al., 2014). Another reason is that some of malware use encryption (Alam et al., 2015) and encoding techniques (Singh & Singh, 2018), making detection and analysis increasingly difficult (Hu, 2011).

The two major types of detecting malware are signature-based and behavior based. While signature-based methods of detecting malware are fast and highly efficient, they are easily bypassed by very new or much older types of ransomwares (Mahdavifar & Ghorbani, 2019; Zhang et al., 2020). On the other hand, while behavior-based methods of detecting malware have a higher resistance to this older malware, they are incredibly time-consuming. For the two different methods of the malware detection system, we can discover that while both are good methods, they are not comprehensive methods of malware detection. Machine learning algorithms as malware detection techniques are effective because they are built to handle all the complexities, and the fast-changing natures of new malware strains have been developed. The use of machine learning algorithms as an approach not for malware detection in cyber security but also in healthcare and medicine

has become widespread. Machine learning (ML) has the upper hand in areas of its high computational power and predictive results. However, many of these machine learning algorithms are very hard to understand and give little to no insight into the contributing factors associated with its result. Although these Machine learning algorithms have good predictive performance, the “opaqueness” of these models makes it difficult for decision-makers to create real-world solutions to combat malware threats.

1.1 Problem Statement and Solution

Detection of malware is critical in order to combat the ever-growing threat that is malware. With the rise in more dangerous types of malwares, such as ransomware, significant efforts must be made to detect them effectively. As discussed in chapter 3, past and current research focuses on hybrid and machine learning approaches, while useful, cannot be trusted to be used as a result in malware detection. This is due to the complex nature of the machine learning models, which are often described as "Black Box". Many of these ML models focus only on accepting the input data and providing the output result of their predictions, not bothering about the "How" surrounding the result. The idea of the black box rings true because one has very little understanding as to why a model performs much better, what factors contribute to the result and which factors are more important within a given prediction. While their computational and predictive ability is ranked highly, this limited visibility into what this complex model reasoning comprises makes it extremely difficult for cyber analysts to trust these results. Cyber Security analysts and all the relevant stakeholders deserve the right to know the reason behind a model's prediction of why malware can be classified as malicious or not. The explanation of this highly predicted result would help them evaluate any future decision, judgment, and solution on

combatting various malware attacks based on their insight into the "How" of these predictions.

In this thesis, a system that utilizes both the signature and dynamic behavior-based detection techniques, with the added layer of the machine learning algorithm with model explainability capability is proposed. This hybrid system provides not only predictions, but also their interpretation and explanation for a malware detection task. The layer of machine learning algorithm can be Logistic Regression (LR), Random Forest (RF), Naive Bayes (NB), Decision Tree (DT), and Support Vector Machine (SVM) (Fumo, 2017). Empirical performance evaluation results on publicly available datasets and manually acquired samples (both benign and malicious) are used to compare the five machine learning algorithms. DALEX (moDel Agnostic Language for Exploration and explanation) (Biecek, 2018) is integrated into the proposed hybrid approach to support the interpretation and understanding (e.g., contributing factors from prediction model) of the prediction to improve the trust for cyber security stakeholders on complex machine learning predictive model.

The random forest is empirically shown to be the best classifier with multiple positive variables contributing to the prediction model. Comparing the breakdown contributions to the Weka black box classification result, we see that random forest has the highest number of correct classification because multiple positive variables with high weights were factored into the classification.

Similarly, Naive Bayes, the worse classifier, only focuses on 3 out of 10 features with very low contribution weight. From our empirical results, it is important to note that classifiers that have greater (positive) single contributions have better predictions than classifiers with

multiple low contributions, which gives less accurate predictions. Therefore, the low SVM and Naïve Bayes classification is because SVM and Naive Bayes pay little attention to other attributes.

1.2 Thesis Outline

Chapter one of this thesis will briefly describe malware, the various types, the channels in which this malware enters our system, and the major ways this malware is detected to determine if a sample is malicious or not. Finally, we also described the tools and algorithms used in the detection process, the problem statement, and the proposed solution. In chapter 2, we discussed a brief overview of moDel Agnostic Language for Exploration and explanation (DALEX), along with the importance of Explanatory Model Analysis and Approaches to Model Explainability. Chapter 3 focuses on the literature review of previous malware detection methods, such as Machine learning file system/process monitoring and network traffic analysis, and current methods based on similar interpretable machine learning models using explainable artificial intelligence (XAI). Chapter 4 describes the materials and algorithm design used for the dataset for model explainability and interpretability. Chapter 5 outlines the experiment of the system and the plan for testing the experiment. Chapter 6 gives the result of the experiment carried out. Chapter 7 discusses the analysis and findings that can be deduced from the result of the experiment and possible reasons for the outcome that was shown in the experiment. Chapter 8 focuses on the conclusion of the thesis, the final findings, the direction for future research and provides references that were used for this thesis.

Chapter 2

MoDel Agnostic Language for Exploration and eXplanation (DALEX)

For this chapter, we will provide a general summary of model explainability from the book "Explanatory Model Analysis" by Przemyslaw Biecek (Biecek, 2018), where he created a taxonomy of model-agnostic explanations for machine learning predictive models. This provides us with a bird's eye view of predictive models and the ever-increasing machine learning framework in a language-agnostic manner.

2.1 Overview

In our ever-evolving technological world, machine learning and its complex predictive algorithm have become important in our daily lives and decisions. Some examples include hospitals, data centers, work, and even everyday IoT devices. However, unexplainable predictions can be very harmful (O'Neil, 2016). Typical cases of mechanical failures used for surgical purposes have inflicted injuries on patients (Alemzadeh et al., 2016) and so much more. Recently, this and many other similar predictive model failure cases have increased public concerns and demand for more transparent, fair, and explainable models.

Over the years, various model explanations have been built, such as the following: modelStudio (Baniecki & Biecek, 2019), lime (Ribeiro et al., 2016), SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017), pdpbox (Jiangchuan, 2018), interpret (Nori et al., 2019), alibi (Klaise et al., 2021), and aix360 (Arya et al., 2020). Other solutions developed for model fairness and interactive dashboards support machine learning. However, all these solutions need to be more cohesive. DALEX was developed to unify all

these fragmented solutions from the traditional black box model up to the explainability model while not compromising on its offering interactive explainability and fairness.

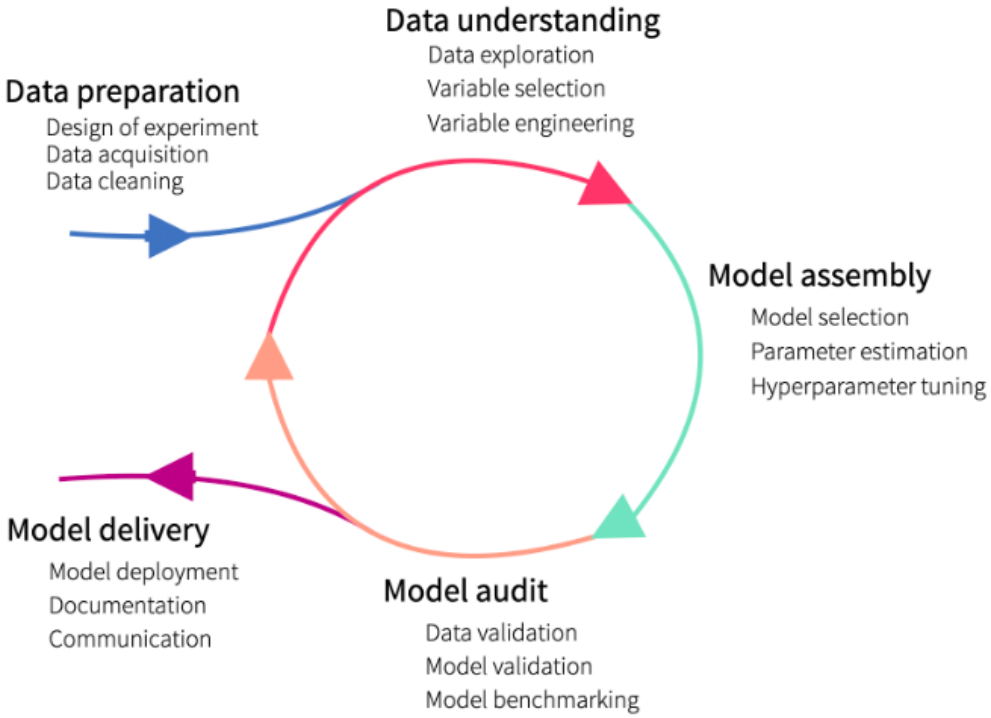
2.2 Importance of Explanatory Model Analysis

Statistical models are divided into two main areas: predictive and explanatory. Predictive models, as part of the two types of statistical models, have existed for years. There is an increasing need for predictive models because it helps give us insight into what the future value of a model would look like and the consequences. For example, we use them to see the likelihood of someone having a certain disease or not.

There are five main reasons why explanatory models are important. These reasons are model complexity, right to explanation, ethical issues, model debugging, and Trust and human interaction. The explanatory model analysis is also very useful because it can be used at any stage in the life cycle model development. Figure 1 shows what a model development cycle for predictive looks like.

Figure 1

Predictive Model Development Lifecycle (Biecek, 2019)



The following five subsections will give us a more in-depth explanation of these five and the roles they play in the importance of explanatory models.

2.2.1 *Model Complexity*

Models have only gotten more and more complex. The availability of fast computers has led to the training of much larger datasets. Therefore, analyzing these models is more challenging by only looking directly at the provided model parameters. Models now possess thousands and millions of these attributes/parameters, so it is essential to have tools that would aid us in analyzing these more complex models.

2.2.2 *Right to Explanation*

Predictive models also play a huge part in our day-to-day lives as they constitute the basis of how many of the shows we watch are recommended, how ads are shown on our social media, how patients are being diagnosed, and so much more. Due to this reason, there have been changes have been made to protect our civil rights. Many countries are now implementing what is known as “The Right to Explanation” as part of their legal systems. Rights to explanation, as defined on Wikipedia, is “a right to be explained an output of the algorithm. Such rights primarily refer to individual rights to be explained decisions that significantly affect an individual, particularly legally or financially.” This protection is put in place to help individuals to gain an understanding of how automated processes work and how these processes affect them and be able to challenge them where necessary.

2.2.3 *Ethical Issues*

In some cases, the analysis of machine learning models might lead to biased decisions. A clear example is models built for recidivism, where the model showed discrimination based on skin color. Another example of biased models was credit score models, which showed biases based on age and skin color. These biases are generated from the learning of historical data. There are many more examples of these types of discrimination daily. Another important factor to note is that it is also very difficult to tell how strongly these biases contribute to the models and to what degree they contribute to these models to help us understand the data and make attempts to correct these biases as we encounter them. The provision of explanatory models helps us by providing those

necessary insights and allowing us to correct factors that may contribute to their bias in each dataset.

2.2.4 *Model Debugging*

As we mentioned earlier in subsection 2.2.1, an increase in model complexity makes it difficult to understand models, making it difficult to fix issues associated with such models. It would only be possible to fix model-related issues when one can understand these models themselves. There is no say that blind changes would not affect the outcome of our models. Therefore, there is a need for tools that provide the right insight to help us with post hoc analysis to enable us to catch issues and fix them to get the best prediction result possible.

2.2.5 Trust and Model Human Interaction

Since machine learning is created to support decision-making by humans, it is important for a level of trust between the models and the humans using them. There is a need for tools that would bridge the gap between the communication of this prediction and the human using them. This transparency would enable humans to trust the models a lot better and, in turn, would enable them to take actionable steps using the information obtained from these models.

2.3 Approaches to Model Explainability

Understanding how a model works is not novel. Over the years, tools for model diagnosis have been developed. However, we can group these approaches of model explainability into three major categories. These categories are interpretable by design, Model Specific Exploration, and Model agnostic Exploration. The following subsection will break down these individual categories with a more detailed explanation.

2.3.1 Interpretable by Design

The idea behind this approach is simple. The overall idea of this approach is that since the structure of the model is simple, the analysis would equally be direct. It requires only the use of models that consist of only simple structures that are very easy to understand. This can consist of numbers that only have very small attributes/variables or decision trees that are not deep or have a lot of rules. However, the drawback to this approach is that it is very time-consuming and requires an expert who has very deep knowledge of how to select and finetune some of the variables to achieve the result.

2.3.2 Model Specific and Exploration

Unlike the previous approach, which was used for simple structures, this is used for more complex model structures. The main objective behind this approach is that since we cannot analyze the parameter individually, we require specific tools to visualize and analyze these complex structures. This approach specializes in exploring how these complex model's function. A few tools used in this approach include diagnostic plots for linear models, node statistics for random forests, or integrated gradients for deep neural networks.

2.3.3 Model Agnostic Exploration

For this research, we will be focusing on this approach. In the model agnostic exploration approach, assumptions are not made about the internal structure of the model. This approach does not care about the simplicity or complexity of the model. Decisions and results for this approach are solely based on the input to this model. This is like the black box model approach, where we analyze only the input. However, the key difference in this approach is the relationship between what the input of the model is to the output.

We believe this is the best approach, and it is used for this research because it is versatile and can be used for any model. It is also very useful in comparing different models.

Chapter 3

Literature Review

This chapter contains material originally published in Technical Analysis of Thanos Ransomware presented by Ogiriki et al. and is used with permission.

This chapter briefly reviews previous malware detection methods classified based on machine learning, file system/process monitoring, and network traffic.

3.1 Classifications Based on Machine Learning

Use machine learning algorithms to group these ransomwares based on their digital genotype and their digital phenotype, this helps to properly identify their various malicious functions, detect the best features of ransomware applications from benign apps, as well as identifying ransomware applications using sequential pattern mining techniques.

3.1.1 DNAAct-Ran

According to Khan et al. (2020), traditional signature-based malware detection is no longer efficient in identifying ransomware. As a result, their answer presented a new and better way of using the ground-breaking Digital DNA sequencing engine. This engine employs a machine-learning algorithm to classify ransomware based on its digital genome and phenotype to identify its numerous destructive functionalities appropriately.

The suggested DNAAct Ran technique uses machine learning to determine if the software is ransomware. This technique achieves this aim by first selecting the significant traits, generating digital sequences for those selected futures, and detecting the ransomware. The algorithms utilized are the Multi-Objective Grey Wolf Optimization (MOGWO, an extension of GWO by Mirjalili, Mirjalili, and Lewis (2014)) and the Binary Cuckoo Search (BCS algorithm) by Yang and Suash (2009). Compared to other ML approaches, the

classifier's performance is used to assess the accuracy of the DNAAct-capacity Ran’s to identify ransomware. Some other active machine learning approaches, in addition to these suggested ones, include naive Bayes, decision stump, and the Adaboost classification algorithm.

3.1.2 Know Abnormal, Find Evil

Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence Homayoun et al. (2020) advocated employing sequential pattern mining algorithms to discover the best attributes of ransomware programs from benign apps and to identify ransomware software. Their detection characteristics' efficiency was examined using them with the J48, random forest, bagging, and MLP classification algorithms. The criteria for this investigation were the usual types of True Positives (for total samples now recognized), False Positives (mistakenly identified samples), True Negatives (number of correctly rejected samples), and False Negatives (number of incorrectly rejected samples) (Incorrectly rejected samples). They begin by identifying and defining detectable patterns and occurrences to identify the appropriate attributes for classification. The sequence pattern mining approach will next be applied to each dataset in order to identify the best sequence pattern. Each sequence in each dataset is then cross-matched based on the maximum sequence pattern to highlight the characteristics of the training classifiers. The following are examples and descriptions of maximum sequence patterns: 1) R (for all events must be registry), 2) D (all events must be DLL), 3) F (all events must be file), 4) RF (multiple transitions, but the first transition is from the registry to the file event), 5) RD (multiple transitions, but the first transition is from the registry to the DLL

event), 6) FR (multiple transitions, but the first transition is from file to registry event), 7) FD (more than one transition, although the initial transition is (more than one transition, but the first transition is from DLL to file event).

3.2 Classifications Based on File System/Process Monitoring

Detecting ransomware based on setting indicators as a way of measuring various ways in which a file changes, if in any case a file all these indications is found to be true then it can be concluded that a file has indeed been corrupted and has an element of malicious characteristics.

3.2.1 UNVEIL

Prior attempts to detect malware have primarily focused on monitoring its low-level file system operations. UNVEIL by Kharraz et al. (2016) is one such technique. UNVEIL detects ransomware by attempting to monitor file activity. To monitor these actions, they were divided into three types based on file system operations (whether a file was read, written/encrypted, deleted, or overwritten). They may be able to detect ransomware assaults as a result of this. The Redemption by Kharraz et al. (2017) method was also used to examine the request pattern of a file I/O to see whether there was any potential ransomware for each process. If this is restored, the processes labeled as dangerous will be terminated. These are good solutions in

general, but their drawback is that many harmless apps, such as encryption and compression of applications, also have such file access characteristic features. If this is the case, these solutions risk producing many false-positive findings since they consider those features to be the same when identifying the activity of various ransomware file systems. Below are a few more characterizations based on file system/process monitoring.

3.2.2 RWGUARD

Mehnaz et al. (2018) presented a decoy-based ransomware technique (RWGUARD) rigorously tested to analyze 14 of the most common ransomwares and detect their operations in real-time. It used both the file change and the process change to identify files encrypted by ransomware.

Three monitoring strategies were used in this approach: file change monitoring, decoy monitoring, and process monitoring. By employing the correct CryptoAPI function and learning characteristics identical to the user's encrypted file, it was possible to distinguish between a benign and an encrypted ransomware file. Using this comprehensive decoy system, it was nearly hard for ransomware to distinguish their fake files.

3.2.3 RANSOMSPECTOR

This method is based on the virtual machine introspection approach presented by Garfinkel and Rosenblum (2003). This solution integrates file operations like opening, renaming, closing, reading, and writing with network operations like connecting, binding, receiving, sending, and disconnecting. They then matched them to discover which corresponded to specific system calls in the operating system's kernel. The virtual machine can also collect this, including context information like the system call's return value, the parameters, and the caller's process. Tang et al. (2020) also discovered that a significant number of crypto-ransomware samples linked to a network create a huge number of network patterns with similar patterns that differ from their file activities. As a result, by analyzing how these ransomware programs interface with the network and file system, they will gain a bit more precision and inform the user if there is evidence of a ransomware assault.

3.2.4 Crypto Ransomware Analysis and Detection Using Process Monitor

Kardile et al. (2017) suggested a method for identifying ransomware attacks based on a process monitor implemented on Cuckoo Sandbox. They intentionally picked sandbox because it removes the danger of data loss. After all, the Cuckoo sandbox returns to its original state after executing the malicious sample. This method builds a genuine and bogus environment to run these ransomware strains. They then capture the file system calls trail and record the I/O access using a process monitor. Their research discovered that when suspected malware attacked the system being targeted, the behaviors and activities of files in the system altered dramatically. They found that the time stamp for the entries in the Master File table was quite close to a ransomware assault occurring on that system by observing the Master File Table.

3.2.5 Cryptodrop

Scaife et al. (2016) presented a solution designed for the Windows operating system, which has been known to be frequently targeted by ransomware. This method of identifying ransomware relies on indicators to track the many ways in which a file changes. If all these indicators are discovered to be true in a file, it may be determined that the file has been corrupted and contains harmful elements. These signs include categorizing ransomware activity based on their actions into three categories. For class one, the ransomware would try to rewrite what was in the original file by opening it, reading it, encrypting it, and closing it. For class two, alter the location of the user's file, read and encrypt the file, and then return it to its original position. The file name may change from the original name by shifting the file back and forth. The ransomware would examine the file, produce an encrypted copy, and then destroy or replace the original for the final class.

3.2.6 SSD-Assisted Ransomware Detection and Data Recovery Techniques

On the storage side, the SSD insider++ method presented by Baek et al. (2020) incorporates sophisticated features like online ransomware detection, flawless data recovery, and sluggish detection. The technique described for online detection is one of the key distinctions between this suggested approach and signature-based alternatives. The algorithm watches and analyzes the host machine's I/O pattern and makes a judgment during run time by analyzing invariant traits that characterize the I/O behavior of ransomware-affected host computers. This is especially significant since it now allows for identifying ransomware attacks in their early phases. The SSD insider++ overcomes the drawbacks of earlier software and hardware in detecting ransomware by combining ransomware detection and a data recovery algorithm onto a single SSD. The SSD-architectural insider++'s architecture comprises ransomware detection and backup/recovery. To identify any abnormal behavior, the SSD Insider++ employs two distinct file operations known as "update-after-read" and "trim-after-read" When ransomware attacks files, its goal is to remain undetected for the longest time feasible by the user. As a result, if many I/O patterns are discovered, we can interpret this as a symptom of a ransomware assault. Baek et al. studied the behavior of six prominent real-world malware to capture ransomware behaviors. Zerber, Locky, Cryptoshield, WannaCry, Mole, and Jaff are examples of malware among them. To identify the traits capable of differentiating this malware by comparing their I/O

footprints to those of common apps. After training and testing with various combinations of this ransomware and programs, the SSD insider++ was able to identify new or undiscovered malware by recognizing their distinctive I/O patterns.

3.3 Classifications Based on the Network Traffic

This solution can be described as a framework that is being used to detect and block various ransomware actions when these malwares are in the process of encrypting files that are being contained in a Network volume from a Network Attached Storage.

3.3.1 The Case of BadRabbit

Alotaibi et al. (2021) developed a technique for detecting efforts to distribute ransomware at the network level rather than preventing the device from being encrypted, which was already addressed in prior studies using BadRabbit as a case study. To accomplish this analysis, they employ two VMs, one Windows 10 and one REMnux, for static analysis. They operated four virtual machines for the dynamic analysis: one with a REMnux acting as a gateway, two with Windows 10 (one infected with BadRabbit), and one with Windows 7. The investigation showed that Bad Rabbit did not need to contact the command-and-control server to exchange an encryption key; instead, it accessed these files using a public key. Because Bad Rabbit is self-propagating ransomware, our solution employs five modules to identify and fight self-propagating malware. Deep packet inspection (dpi) and packet header inspection are examples of these modules (phi), honey pot-based, ARP scanning-based detection, and SMB Packet size checkers.

3.3.2 REDFISH

Morato et al. (2018) presented Ransomware Early Detection from FIle SHaring traffic which is usually referred to as REDFISH. This solution may be regarded as a framework for detecting and blocking different ransomware behaviors when the infection encrypts data on a network volume from a Network Attached Storage. This solution examines the difference in traffic behavior between infected and non-infected hosts. The

characteristics they investigated for these host behaviors include how files on a shared file are opened, read, written, and deleted. Their approach is derived from studying SMB/SMB2 traffic over a single TCP connection.

Chapter 4

System Design

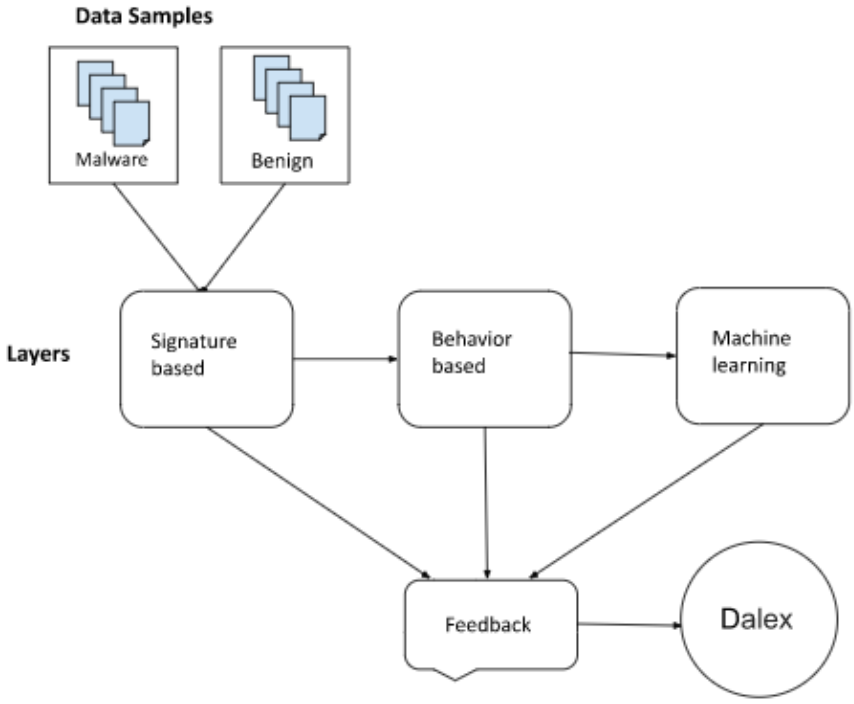
This chapter describes and discusses techniques for detecting malware, machine learning classification, and model interpretability using DALEX. For the system design, we used a hybrid approach that utilizes both the signature and dynamic behavior-based detection techniques, with the added layer of the machine learning algorithm and model explainability. This hybrid system would give a more robust answer to the malware detection challenge. The proposed system's goal is to detect and interpret the various models used for malware classification. The detailed information about system design and the approach is described in detail in the chapter.

4.1 Architectural Layer of the Proposed System

As illustrated in Figure 2, we present a relatively new approach with four layers. Malware samples would be analyzed using signature and behavior-based approaches (layers one and two). These layers would combine the benefits of both strategies. Afterward, we would go on to the next layer, utilizing a machine-learning technique to train malware classifiers.

Figure 2

Diagram of Architectural Layer of the Proposed System



4.2 Artificial User Testing Environment

Malware generally can hide its malicious tendencies when they know they are being monitored. Therefore, it was important to create an environment that would protect the user's information system and information while also testing if malware is malicious or benign. For the signature and behavioral portion of our experiment, we installed Cuckoo Sandbox [Jurriaan, 2013]. Cuckoo Sandbox was our choice because it creates an isolated environment, particularly on Windows 7, where a lot of malwares easily attacks. Cuckoo sandbox is one of the leading open-source automated malware analysis systems. Using cuckoos' sandbox, we can provide any suspicious file, and in a matter of minutes, cuckoos will give us a very detailed report that would enable us to see how a malware file behaves

when it is being carried out inside an environment that looks as realistic, but it is in fact isolated. This software is free and can carry out tasks that can check for files that have malicious behavior in Android, macOS, Linux, and Windows. The features of Cuckoos are.

1. Ability to analyze various malware files, such as emails, executable files, documents, emails, executable files, etc.
2. Ability to perform memory analysis
3. Gather the general behavior of these malicious files, which contains their information and general signatures. It also traces their API calls and compiles them in a simple, readable format.
4. It can analyze and dump network traffic, including when it was encrypted with SSL/TLS.

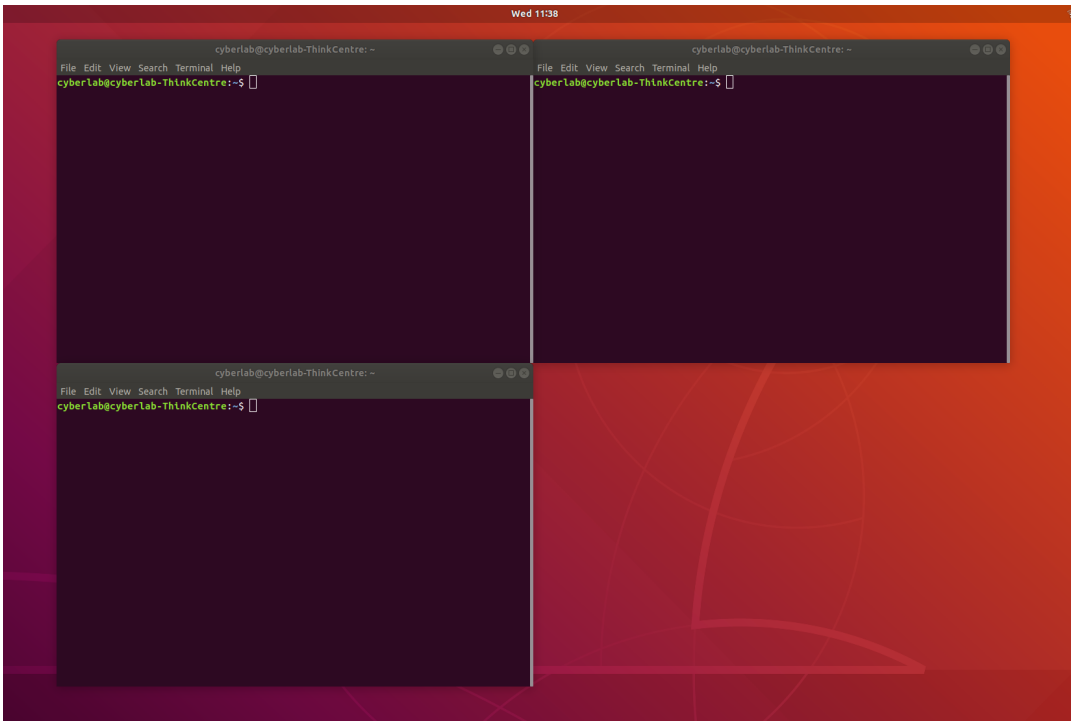
With cuckoo, we can mimic as closely as possible a real windows environment so we can capture a true display of malware samples. Here are a few steps on how we configured the Cuckoo Sandbox.

How to configure Cuckoo for daily use:

- First, open 3 terminal windows:

Figure 3

Terminal Window of Cuckoo Setup



- Then enable the virtual environment in all 3 terminal windows (NOTE: do this before running Cuckoo sandbox in any of the windows or else this will not work, and an error message will be shown)
    - First, run this command in each terminal to set up the virtualenv:

      virtualenv ~/cuckoo
    - Then run this command in each terminal to start the virtualenv:

Figure 4

Terminal Window of Cuckoo Virtual Environment

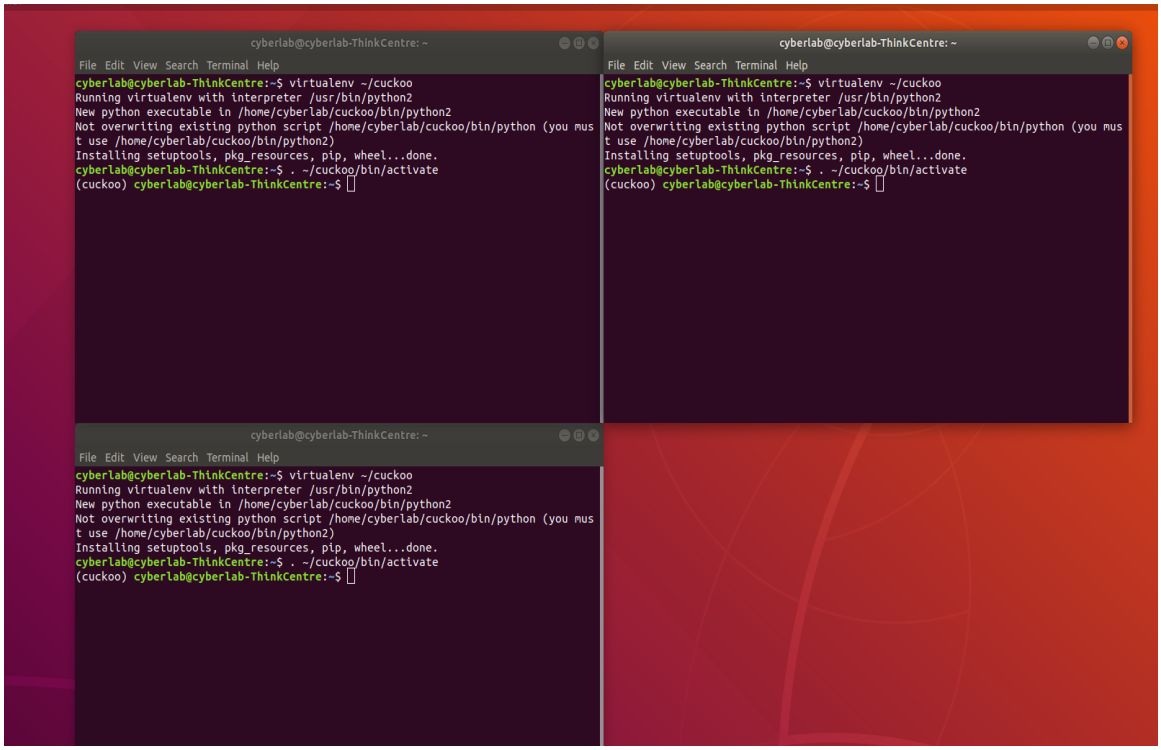
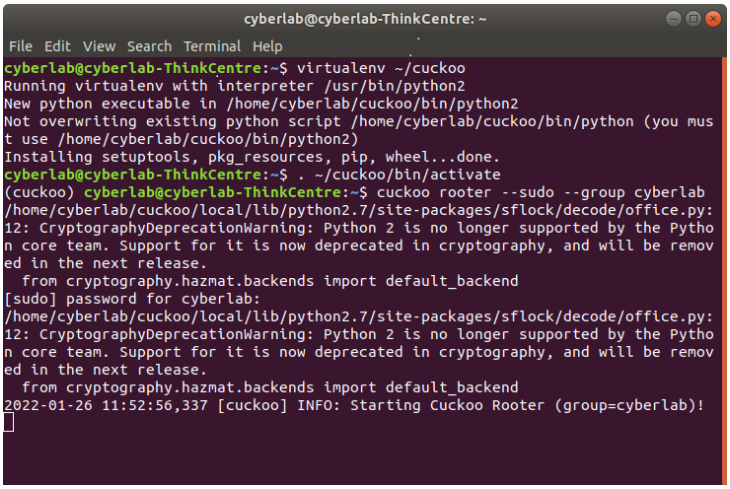


Figure 5

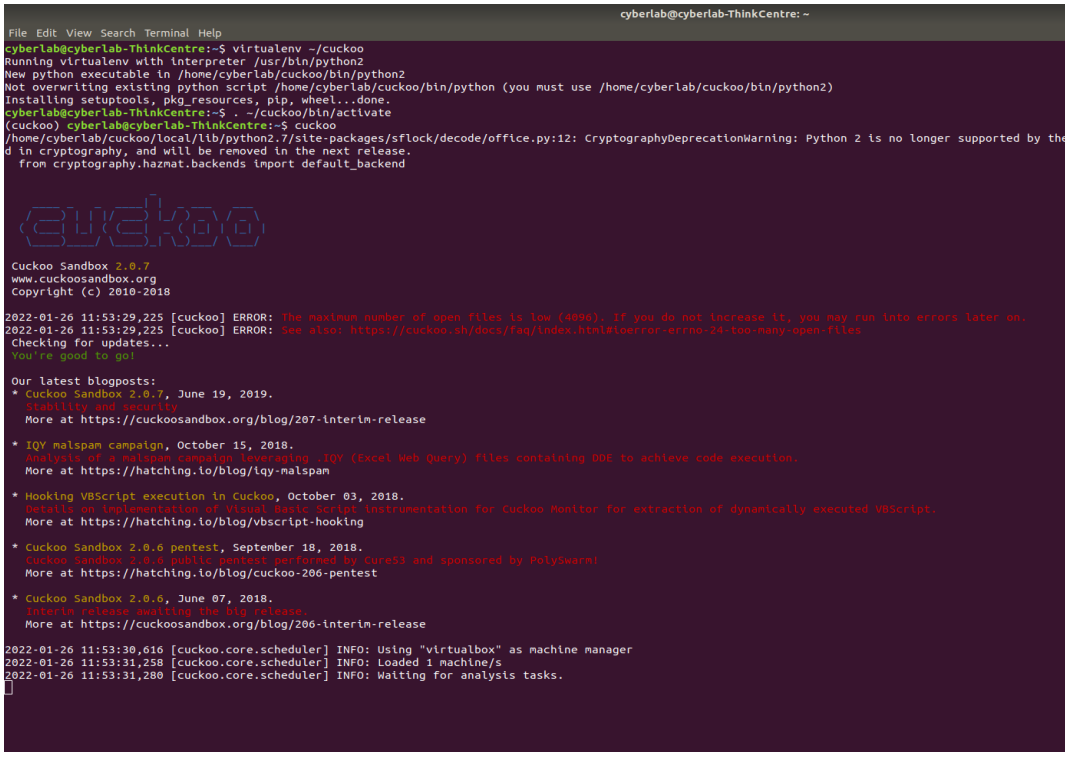
Terminal Window of Cuckoo Router



- Run the Cuckoo Rooter in one of the terminals using this command: cuckoo rooter
- --sudo --group cyberlab:
- Run the command cuckoo in another terminal window:

Figure 6

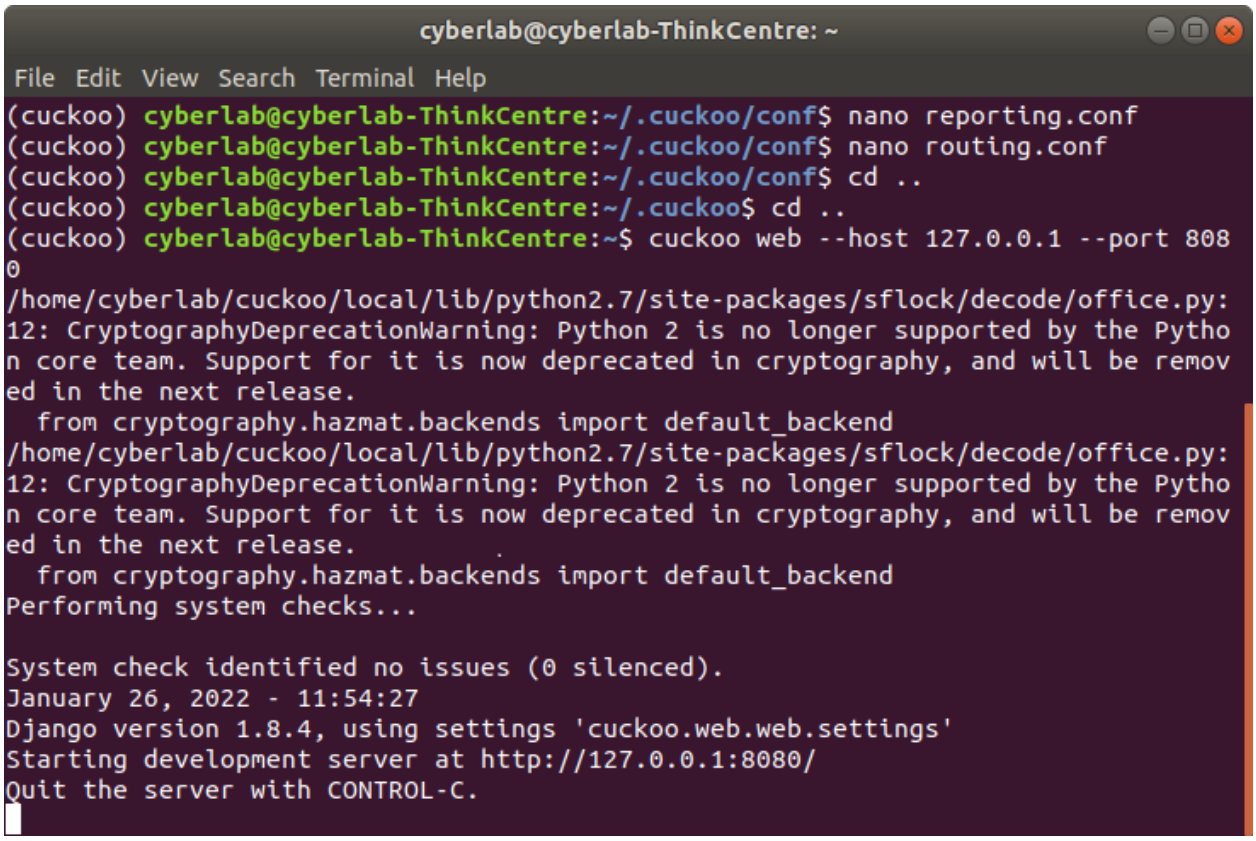
Terminal Window of Successful Cuckoo Setup



- Type in this command in the third terminal to start the web UI: cuckoo web --host 127.0.0.1 --port 8080

Figure 7

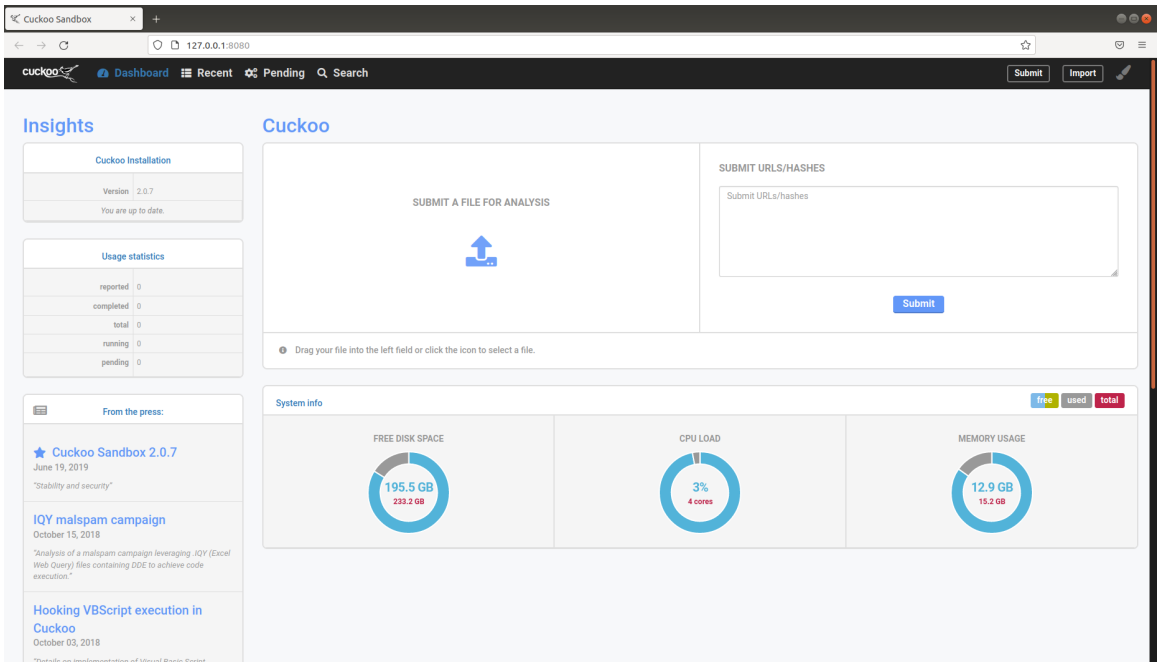
Terminal Window of Cuckoo Web Setup



- Finally, click on the link http://127.0.0.1:8080/ to open Cuckoo Sandbox

Figure 8

Cuckoo Web Application



To use the sandbox test was carried out on both the benign and the malicious samples. For the samples in Figure 9, we used EngRat.0.1.0, a type of ransomware, as our test file and analyzed using Cuckoo in a Windows 7 VM. Here is the summary of the information:

Figure 9

Summary of EngRat Malware Samples in Cuckoo Sandbox

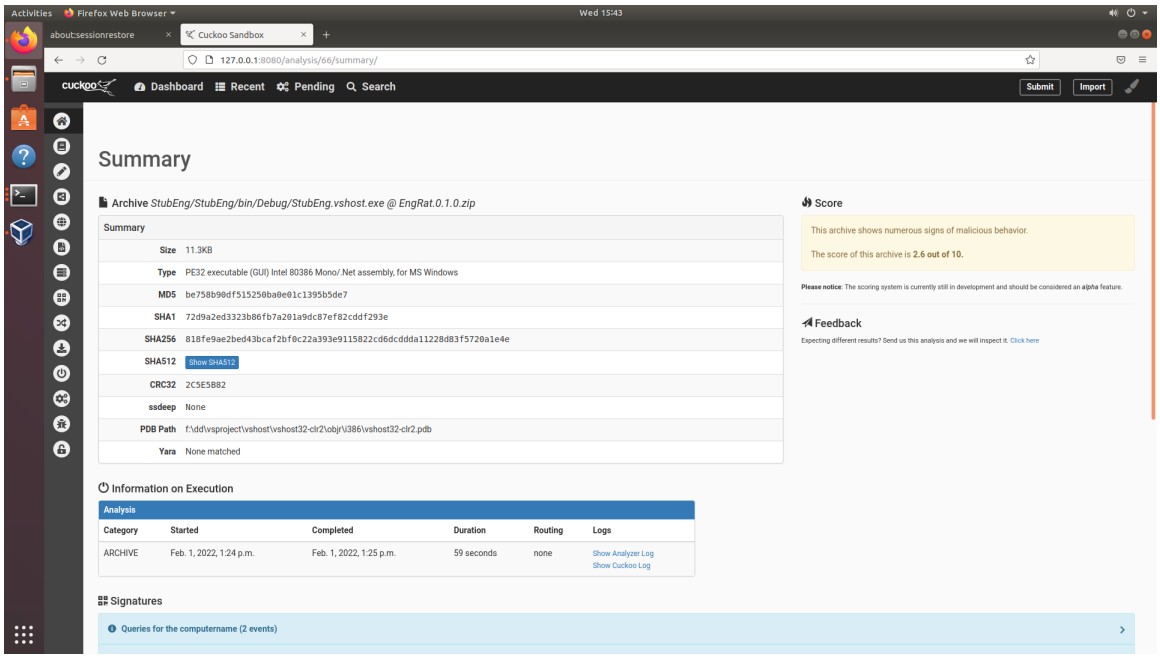
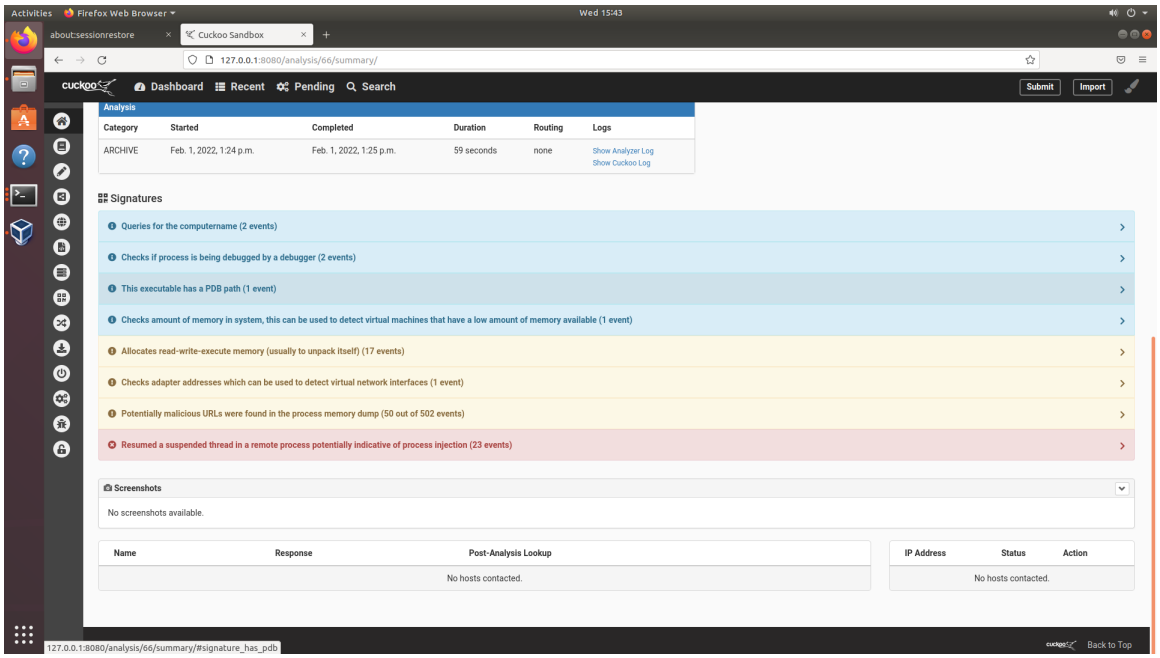


Figure 10

Signatures of EngRat Malware Samples in Cuckoo Sandbox

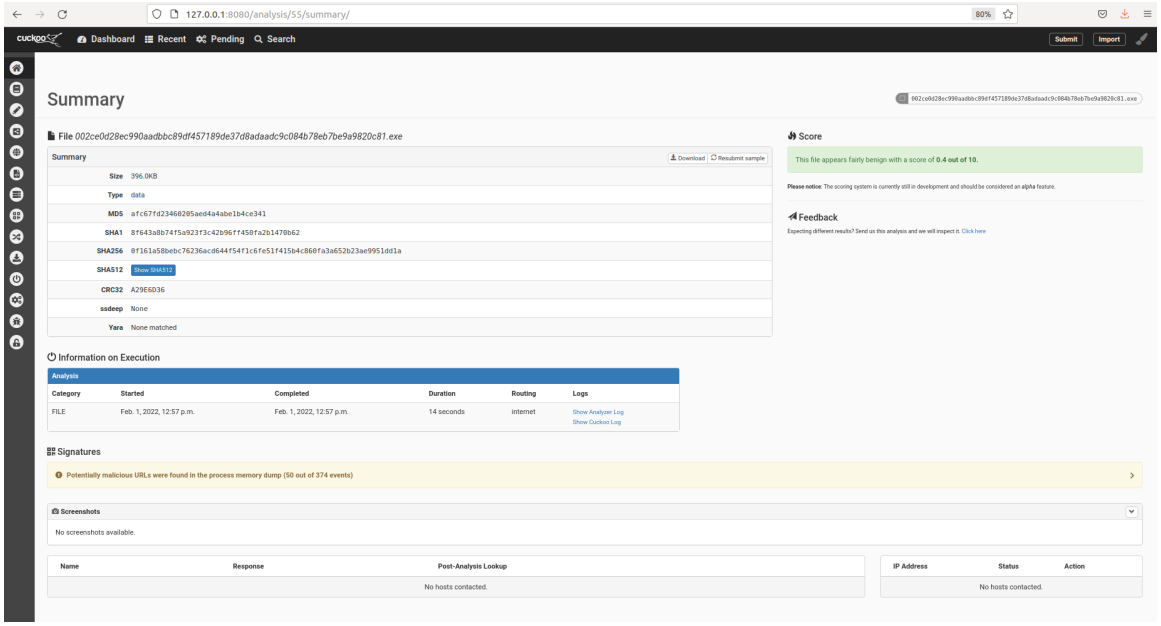


Upon analysis, cuckoo rates the ransomware at 2.6 out of 10. It notes how the ransomware searched potentially malicious URLs, allocated read-write-execute memory to unpack itself, and checked the adapter addresses that can detect virtual network interfaces.

We can compare this to our benign sample, which is like this ransomware:

Figure 11

Summary of Benign Samples in Cuckoo Sandbox



While both released potentially malicious URLs, the benign only yields a result of 0.4. It should be noted that scores lower than 1 out of 10 are considered benign and highlighted green. In contrast, scores higher are highlighted yellow or even red.

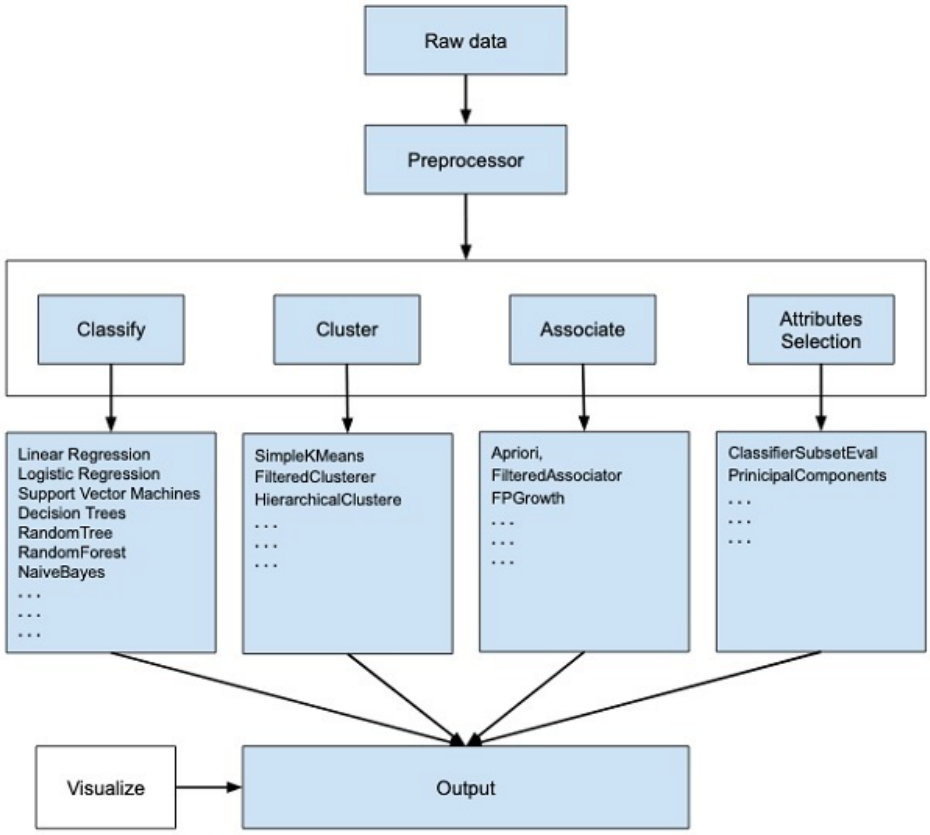
4.3 Machine Learning Using Weka

For the third layer of the system design using machine learning analysis, we downloaded Weka and uploaded the CSV of the samples that have been tested and verified from the cuckoo sandbox. Weka is an open-source software that allows for implementing machine learning algorithms. Although it offers some visualization and output based on the result of the machine-learning classification, it is still a black-box solution. It offers little to no explanation about how machine learning models make their predictions and what factors, and features contribute to their predictions. The full summary of its offerings can be seen below in Figure 12. Weka offers various options such as classify, cluster,

associate, and attribute, which can be very useful for reducing the size of a data set. Weka is also useful for massive data as it offers a simple, easy-to-use GUI interface to handle big data with several ready-to-use algorithms, which leads to quicker development of various machine learning model results in few seconds.

Figure 12

Summarize Architectural Structure of Weka



4.4 Model Explanation

From the above design, we have detected whether malware is benign or malicious. We have also been able to use various classifiers to see which machine learning algorithm performs the best to group this malware into their various types. While we have some results using weka, there needs to be more trust in the result obtained because there is no visibility into the contribution features of these results. This is where the model explanation is useful, as can be shown in Figure 10 below. The concept of the model explanation is in

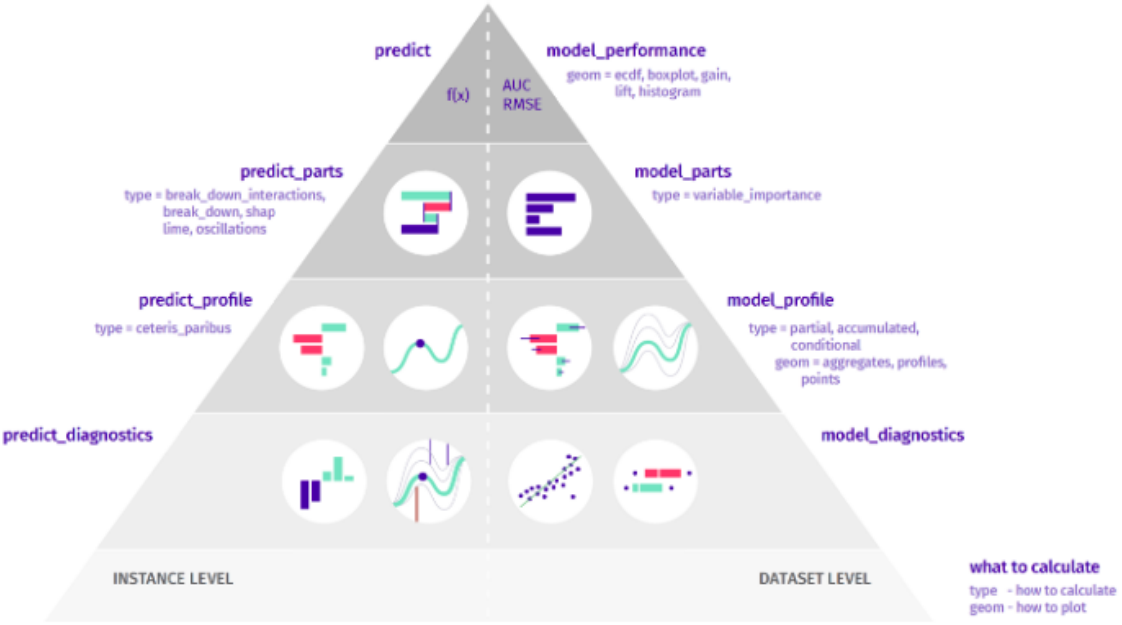
the form of a pyramid. At the very top, it starts with the prediction, which is very similar to the result from weka to show how well the model performed. This assessment can be done in various ways. It can be through the F1 score and the ROC/AUC curve. On this level, the model provides a high-level assessment of the quality of the model.

The second layer of the pyramid provides even more details. On this level, we gain a deeper understanding of what variables can be classified as important and what parts influence the model to work the way it does or not. This level is concerned with showing the strength and influence of the variable.

Further down the pyramid is the explanation of how the model would react if any disturbances or changes occur with the values of its variables. This is also known in the Latin phrase ceteris paribus, which translates to “all things being equal” or “all things being unchanged”. This would help us to understand how model predictions can be affected by variable changes, one at a time. The last layer could help us understand “How good the local fit of the model” is to see if the model performed well in certain situations vs. others. Overall, the entire idea of the pyramid of model explanation is that the further down you go, the deeper the level of detail you can get about a model’s prediction. This is way more useful in solving malware detection problems as it gives a more thorough understanding of black box results in an interactive way.

Figure 13

Diagram of DALEX Explanatory Model



Chapter 5

Experiment Design

This chapter discusses in extensive detail the design of the experiment. This consists of various components, such as the means of acquiring the dataset, the dataset's description, the experiment's implementation, and the classification algorithms used. Finally, the metrics used to determine the model's accuracy to produce the results in chapter 6.

5.1 Data Collection

Previous methods obtained data from various websites such as Virus Total, Virus shares, Zelster, MWanalysis.org, Vxheaven, PCHome Malware Repositories, etc. For research purposes, three major types of datasets are available/ being used and they are:

5.1.1 Publicly Available Datasets

These are currently being offered and provided publicly. They are also being updated and maintained by research enthusiasts for the purpose of research all over the world for free in the field of cyber security.

5.1.2 Artificially Generated Datasets

These are classified as datasets generated manually using special tools or collected from the network traffic by cyber security researchers.

5.1.3 Commercial Datasets

As the name implies, these are datasets that are not freely offered to the public. They are provided as commercial projects and supported by companies for commercial purposes. For this research's sake, we obtained our dataset from publicly available datasets. This is because public datasets are freely available, generate new insights into data

collected by fellow researchers, and possess a larger sample size. To obtain the biggest sample data, we obtained malicious ransomware samples from public sites such as virus total, Vx heavens, NetLux, Anubis, nexginre for malicious data, and Benign samples from portableapps.com.

5.2 Description of the Dataset

The primary dataset used for this experiment was obtained from the University of California, Irvine (Alberto, 2019), which Virus Total donated. This dataset comprises six types of malwares. The samples are not evenly split between each type, with malware types such as trojans, viruses, and adware having the highest numbers, while worms and ransomware have much lower samples. The dataset consists of 2955 samples in total from Virus Total, with over 1000 extracted attributes. With 1901 malware samples and 1054 benign, respectively. These samples consist of both benign and malware samples. Two types of features can be extracted from malware samples. They are static features and dynamic features. Static features are features that can be obtained without running the malicious samples. Whereas Dynamic features are obtained from running the malicious samples from a testing environment. The attribute features for this dataset consist only of dynamic features extracted from the cuckoo sandbox. Cuckoo sandbox is extremely useful for malware analysis because it allows the simulation of an actual computing environment by performing basic human interactions such as opening/closing files, running command line scripts, enabling the submission of malware samples, and much more. This way, we can correctly record the attributes of this malware. Although Cuckoo sandbox is our choice for this experiment, any other sandbox, such as Anubis, NorMan etc., that are used for dynamic analysis could also be used.

Due to the number of attributes, there were a lot of null or empty values, so this dataset required a lot of cleaning to ensure the accuracy of the result. We were able to effectively reduce the attribute size from over 1000 down to 64 relevant features such as dll, file registry information, name, type, import, etc. The dataset only contained its filename using its SHA-256. We also used the SHA-256 value to find the malware name and class of malware for machine learning analysis for all the malware in the dataset.

Table 1

Virus Total Dataset and Classes

No.	**Classes**	**Count**
1	adware	389
2	trojan	750
3	virus	438
4	riskware	45
5	worm	94
6	ransomware	185
7	benign	1054
		Total: 2955

5.3 Implementation

For implementation, we started by testing a lot of samples on the Cuckoo sandbox in order to obtain the result of the samples. The results of the samples are usually stored in a .JSON file. The benefit of Cuckoo is that it allows us to run the malicious samples on that environment like it was a real system while obtaining information about whether the sample was malicious. Since we wanted to test a lot of samples to ascertain whether they were malicious, we decided to use a dataset obtained from Virus Total. The dataset comprised 2955 samples, with features extraction of over 1000. We used this dataset because its features were extracted from the cuckoo sandbox, which is like our already tested samples, and we can add to this dataset to increase the numbers. Out of the 2955 samples, over 62 trojan samples were duplicated. We removed all duplicated trojan samples and replaced them with 62 other ransomware samples. Using cuckoo sandbox, we tested the 62 ransomware samples on both windows 10 and windows 7, respectively, and We extracted the result seen in Figure 14 and Figure 15 below.

Figure 14

Ransomware Test Results on Windows 7

Name	Time Run (s)	Windows 7 Score	Allocates r-w-x	Generates ICMP	Mal. URLs found in mem dump	Corrs w/ host	Resumes sus. thread	Checks amt. of mem	# of samples
Electro Rat file A	75	5.6	y	y	y	n	n	n	12
Electro Rat file B	78	6.8	n	y	y	n	n	n	
Electro Rat file C	107	5.4	y	y	y	n	n	n	
Electro Rat file D	106	3.6	y	y	y	n	n	n	
Electro Rat file E	63	1.4	y	y	n	n	n	n	
Pegasus	77	3	y	y	y	n	y	n	7
Satana File A	134	15.8	n	y	n	y	y	n	2
Satana File B	170	9.2	y	y	y	n	y	n	
Tesla Crypt File A	167	25.8	y	y	y	n	y	y	5
Tesla Crypt File B	167	18.8	y	y	y	n	y	y	
Tesla Crypt File C	197	26.4	y	y	y	y	n	y	
WannaCry	199	21.6	y	y	y	y	y	y	28
Powelliks File A	73	25.8	y	y	y	y	y	y	7
Powelliks File B	144	12.8	y	y	y	y	y	y	
ZeusGameover	155	13.4	y	y	y	y	n	y	4
ZeusBanking	31	8.2	y	y	n	n	y	n	1
WMGhost	51	5	y	y	y	n	n	n	19
Ardamax Keylogger	72	5.4	y	y	y	n	n	y	1
Civil War File A	75	3	y	y	y	n	y	n	3
Civil War File B	149	3	y	y	y	n	y	n	
Cryptowall	21	8.8	y	y	y	n	y	n	1
RedBoot	133	5.4	n	y	n	n	n	n	1
Radamant	89	6	n	y	y	n	n	y	2
Bechiro	106	5.2	y	y	y	n	y	y	1
Blackbird	74	8.2	y	y	y	n	y	y	1
Catapiller	133	3	y	y	y	n	y	n	2
Jigsaw	82	6.2	y	y	y	n	n	y	1
Kazy	18	8	y	y	y	n	n	n	1
LSD File A	135	3	y	y	y	n	y	n	2
LSD File B	139	3	y	y	y	n	y	n	
Matsnu	25	7.8	y	y	y	n	y	n	1
njRat	16	2.2	n	y	y	n	n	n	10
Petya A	194	2.6	y	y	n	n	n	n	2
Petya B	199	2.8	y	y	n	n	n	n	
Quax	76	3.6	y	y	y	n	y	n	2
Rex	19	1.4	n	y	y	n	n	n	1
ShetHas	65	3.2	y	y	n	n	y	n	2
ShadowHammer	133	2.2	y	n	y	n	y	n	1
Infostealer	73	5.6	n	n	y	y	n	n	2
CodoRed Worm A	22	3.2	y	n	y	n	n	n	6
CodoRed Worm B	43	3.8	y	n	y	n	n	n	
Boaxxe	75	10.6	y	n	y	y	n	n	1
ZeroLocker	16	1.4	n	y	n	n	n	n	1
Telefonica	135	2.2	y	n	y	n	y	n	3
Waski	87	7	n	n	y	n	n	n	5
Scansbm	87	2.2	y	n	y	n	y	n	4
Sonoto	36	3.6	y	n	y	n	n	n	1
Nukesped	83	3.8	n	n	y	y	n	n	3
JS Lame	75	2.2	y	n	y	n	y	n	2
Loadmoney	31	2.2	n	n	y	n	n	n	1
Cairxpil	134	3	y	y	y	n	y	n	1
Cryptolocker	132	5.8	n	y	y	y	n	n	4
Anti Exe	136	3	y	y	y	n	y	n	2

Figure 15

Ransomware Test Results on Windows 10

Name	Time Run (s)	Windows 10 Score	Allocates r-w-x	Generates ICMP	Mal. URLs found in mem dump	Coms w/ host	Resumes sus. thread	Checks amt. of mem
Electro Rat file A	147	1.4	n	y	y	n	n	n
Electro Rat file B	287	1.4	n	y	y	n	n	n
Electro Rat file C	702	1.4	n	y	y	n	n	n
Electro Rat file D	560	1.4	n	y	y	n	n	n
Electro Rat file E	416	1	n	y	n	n	n	n
Pegasus	15	0.8	n	y	n	n	n	n
Satana File A	22	3.8	n	y	n	n	n	n
Satana File B	64	6.6	y	y	n	y	y	n
Tesla Crypt File A	140	1.6	n	y	y	n	n	n
Tesla Crypt File B	436	1.8	n	y	y	n	n	n
Tesla Crypt File C	440	1.8	n	y	y	n	n	n
WannaCry	139	2	n	y	y	n	n	n
Poweliks File A	43	2.4	n	y	n	y	n	n
Poweliks File B	89	3	y	y	n	y	n	n
ZeusGameover	37	1.6	n	y	n	n	n	n
ZeusBanking	139	1.4	n	y	y	n	n	n
WMIGhost	30	1.6	n	y	y	n	n	n
Ardamax Keylogger	21	1	n	y	n	n	n	n
Civil War File A	31	0.8	n	y	n	n	n	n
Civil War File B	48	0.8	n	y	n	n	n	n
Cryptowall	139	1.2	n	y	y	n	n	n
RedBoot	18	1.6	n	y	n	n	n	n
Radamant	140	3.4	n	y	y	n	n	n
Bechiro	16	1.4	n	y	n	n	n	n
Bladabindi	295	2.2	n	y	n	n	y	n
Catapillar	31	0.8	n	y	n	n	n	n
Jigsaw	17	1.4	n	y	n	n	n	n
Kazy	17	2.4	n	y	n	n	n	n
LSD File A	17	0.8	n	y	n	n	n	n
LSD File B	32	0.8	n	y	n	n	n	n
Matsnu	23	1.2	n	y	n	n	n	n
njRat	16	1.2	n	y	n	n	n	n
Petya A	17	1.2	n	y	n	n	n	n
Petya B	33	1.4	n	y	n	n	n	n
Quax	16	0.8	n	y	n	n	n	n
Rex	138	0.8	n	y	n	n	n	n
SheHas	36	0.8	n	y	n	n	n	n
ShadowHammer	17	0.8	n	y	n	n	n	n
Infostealer	19	4	n	y	n	n	n	n
CodeRed Worm A	245	0.8	n	y	n	n	n	n
CodeRed Worm B	32	0.8	n	y	n	n	n	n
Boaxxe	139	1.4	n	y	y	n	n	n
ZeroLocker	17	1.4	n	y	n	n	n	n
Telefonica	173	0.8	n	y	n	n	n	n

Steps used to Preprocess the Dataset

1. **Labeling of samples:** The samples only came with the hash values, so we cross checked each hash value and relabeled each of the acquired samples in the dataset to its original virus name. Since the samples were acquired on Virus total, we matched the hash to the virus name on the virustotal.com website.
2. **Clean the dataset:** After acquiring the dataset, we realized that a lot of the extracted features were empty. We decided to delete all the columns of the data of the extracted features where they all had an empty or a zero value. This brought our

dataset from over 1000 features to about 63 relevant features such as dll, file registry information, name, type, import, etc.

5.4 Data Analysis

Classification algorithms are divided into symbolic learning algorithms (CART. C4.5, NewID, AC2, ITrule, Cal5, CN2), statistical algorithms (Naive Bayes, K-Nearest neighbor, kernel density, linear discriminant, quadratic discriminant, logistic regression, projection pursuit, Bayesian networks), neural networks (backpropagation, radial basis functions), and Random Forest. However, for the purpose of this research, we will be comparing Logistic Regression, Naive Bayes Classifier, Random Forest, SVM, and Decision Tree. The aim is to record the performance of these classifiers on the data set and compare the result of these black box models to DALEX. The result will be tabulated and graphed to show the recommended algorithm for classifying data sets.

The experiment will consist of the following stages:

- The data was collected (and cleaned where necessary) from the various open-source malware databases.
- The data was separated into training and test data.
- The dataset trains a model using logistic regression, performs the necessary tests, and records findings.
- The dataset is used to train the model using the Naive Bayes Classifier, perform the necessary tests, and record findings.
- The dataset is used to train the model using the Decision tree, perform the necessary tests, and record findings.

- The dataset is used to train the model using the SVM classifier, perform the necessary tests, and record findings.
- Compare the performance of the algorithms between the black box model and DALEX. Also, provide proof (if any) of the recommended algorithm for the input data.

The goal is to train the five selected classification algorithms to predict whether a malware sample is malicious or benign. The data set used for the training and testing will be the same. The performance of all algorithms will be measured, calculated, and compared based on accuracy, speed, and error recorded.

Chapter 6

Model Performance Comparison and Discussion

In chapter 5, we talked in detail about how we set up our experiment to collect and test benign and malware samples using weka, a black box machine learning software. In this chapter, we will display the black box model (weka) result. The first part of this section starts with a tabulated result of each machine learning classifier using weka. Then we would proceed with discussing the result of the black box model. The experiment's main objective is to conduct an extensive experiment using a black box model and find out the best classification algorithm.

6.1 Machine Learning Classifier Performance on Black Box Model

The tabulated results shown in this subsection in Tables 2, 3, 4, 5, and 6 are from the machine learning classification output using weka. The accuracy result for each machine learning classifier was obtained from the Virus Total dataset discussed in section 5.1 above.

Table 2

Machine Learning Result for Random Forest

Weight split: 80/20								
Random forest Accuracy: 88.49%								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC area	Prc area	Class
0.76	0.05	0.74	0.76	0.75	0.71	0.97	0.80	adware
0.86	0.07	0.83	0.86	0.84	0.84	0.95	0.89	trojan
0.94	0.02	0.89	0.94	0.91	0.91	0.99	0.97	virus
0.22	0.00	0.50	0.20	0.29	0.29	0.74	0.34	riskware
0.77	0.00	0.94	0.77	0.85	0.85	0.97	0.89	worm
0.33	0.00	1.00	0.33	0.50	1.00	0.93	0.39	ransomware
1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	Benign

Table 3

Machine Learning Result for Decision Tree

Weight split: 80/20								
Decision Tree Accuracy: 86.97%								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC area	Prc area	Class
0.66	0.02	0.80	0.66	0.72	0.68	0.91	0.66	adware
0.91	0.12	0.75	0.91	0.82	0.75	0.91	0.73	trojan
0.87	0.02	0.89	0.87	0.88	0.86	0.95	0.85	virus
0.20	0.00	0.67	0.20	0.31	0.36	0.74	0.24	riskware
0.68	0.00	1.00	0.69	0.81	0.82	0.95	0.77	worm
0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.11	ransomware
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	Benign

Table 4

Machine Learning Result for Logistic Regression

Weight: 80 /20								
Logistic Regression Accuracy: 82.06%								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC area	Prc area	Class
0.67	0.06	0.64	0.67	0.66	0.60	0.94	0.64	adware
0.77	0.12	0.73	0.77	0.75	0.64	0.92	0.81	trojan
0.83	0.03	0.84	0.83	0.84	0.81	0.94	0.87	virus
0.00	0.00	0.00	0.00	0.00	-0.01	0.89	0.11	riskware
0.77	0.00	0.77	0.77	0.77	0.76	0.96	0.61	worm
0.17	0.00	0.33	0.17	0.22	0.23	0.75	0.18	ransomware
0.99	0.00	1.00	0.99	0.99	0.99	0.99	0.99	benign

Table 5

Machine Learning Result for SVM

Weight split: 80/20								
SVM Accuracy: 55.33%								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC area	Prc area	Class
0.14	0.00	0.86	0.14	0.25	0.32	0.57	0.24	adware
0.98	0.62	0.39	0.98	0.56	0.37	0.68	0.39	trojan
0.30	0.00	1.00	0.30	0.47	0.52	0.65	0.41	virus
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.01	riskware
0.00	0.00	0.00	0.00	0.00	0.00	0.55	0.04	worm
0.33	0.00	1.00	0.33	0.50	0.58	0.67	0.34	ransomware
0.57	0.00	1.00	0.56	0.72	0.68	0.69	0.72	Benign

Table 6

Machine Learning Result for Naive Bayes

Weight split: 80/20								
Naive Bayes Accuracy: 45.35%								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC area	Prc area	Class
0.27	0.07	0.38	0.27	0.32	0.23	0.87	0.48	adware
0.06	0.02	0.58	0.06	0.12	0.12	0.82	0.58	trojan
0.03	0.02	0.20	0.03	0.06	0.20	0.88	0.46	virus
0.40	0.13	0.05	0.40	0.09	0.10	0.78	0.05	riskware
0.90	0.23	0.13	0.91	0.23	0.30	0.95	0.73	worm
0.33	0.10	0.03	0.33	0.06	0.08	0.76	0.19	ransomware
1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	benign

6.2 Black Box Machine Learning Discussion

To discuss the accuracy results of the machine learning algorithms in section 6.1 above, it is important to understand the metrics used for the comparison that contributed to its overall prediction result shown in Table 7 below. These key metrics are accuracy, precision, recall, F-Measure, MCC, ROC area, and PRC area.

Table 7

Machine Learning Confusion Matrix

	Predicted as Positive	**Predicted as Negative**
Actually Positive	True Positive (TP)	False Negative (FN)
Actually Negative	False Positive (FP)	True Negative (TN)

From Table 7 above, we can define the following evaluation measures below.

a) Accuracy = $(TP + TN) \div (TP + TN + FP + FN)$

b) Recall = $TP \div (TP + FN)$

c) Precision = $TP \div (TP + FP)$

d) F-Measure = 2 x [(Precision x Recall) / (Precision + Recall)]. This can be described as the harmonic mean of recall and precision.

e) MCC = This is abbreviation of Matthew's Correlation Coefficient used in model evaluation to measure what the difference between the predicted value and the actual value is. Ranges between -1 to + 1 with +1 being the perfect model and -1 being a poor model. It is calculated by the formula below.

$$MCC = \frac{(TN \times TP) - (FN \times FP)}{\sqrt{(TP + FP)(FT + FN)(TN + FP)(TN + FN)}}$$

f) ROC area = This is also known as receiver operating characteristics describes the probability that positive instances ranked higher than a negative instances when chosen randomly.

g) PRC area = This measure compares the true positive rate to the predicted positive rate in binary classification tasks. In other words, this can be described as the relationship between recall and precision. For unbalanced datasets, this is a more appropriate measure than the ROC area (Lang et al, 2019).

The model was evaluated on a data split model of the 80/20% approach. Where 80% of the dataset was used for the training of the dataset, and 20 percent was used for the testing of the dataset. It was important to use this method because the model needed to use the training dataset to first learn and then apply whatever it has learned to the rest of the dataset it hasn't tested on before to make the most accurate model predictions. The outcome of this prediction can be seen in chapter 6 above.

On a high level, For the machine learning black box model above, we see that Table 2, which is the random forest had the highest score of 88.49% and the lowest being Naive Bayes with 45.35%. Three other algorithms were also used, and they are as follows:

1. Decision Tree with an accuracy result of approximately 86.97%.
2. Logistic Regression with an accuracy result of approximately 82.06%.
3. Support Vector Machine with an accuracy of approximately 55.33%.

We can clearly see from the above results that 3 out of the classifiers performed well, with very similar results ranging from 82 - 88%. However, SVM and Naive Bayes were the least performing algorithms, with an accuracy of approximately 45 and 55%, respectively. This clearly shows that those models were the poorest and did not classify these samples well. Naive Bayes is known to be one of the faster classifiers compared to logistic regression, but it was the least performer for this data set. The MCC, ROC and PRC area provides weighted ways to understand performance. The most used out of the three is the ROC area under the curve. Higher ROC score of about 0.5 shows that the model is randomly guessing, anything above 0.5 shows the model is performing better than randomly guessing. From the result above asides from SVM model, all the other classifiers have a much higher ROC area score which indicates high performance for the model for each of the various malware classes. While both the ROC and the PRC area look at the predictive score of the classification model, one major difference is that the ROC area looks at the true positive rate and the false positive rate while the PRC area looks at the positive predictive value and the true positive predictive value.

Chapter 7

DALEX Results and Discussion

In this chapter, we will evaluate and compare the experimental results carried out in Chapter 6 above and gain more insights into the contributing factors and weights using DALEX. This section is divided into two parts. The first part discusses the result from the explanation of the models using DALEX. Finally, we would discuss in detail some of the features that contribute to the white box model predictions for the different machine learning models.

7.1 Machine Learning Model Explainability Using DALEX

From section 6.1, we can see that we only get the values of the result but not the explanation/insight of these values. This section explains each of the five machine learning classification algorithms built for the Virus Total dataset. We consider two of the most important explanations: the variable importance and the breakdown. We have thoroughly explained the model breakdown in section 4.3 above. Figure 16 - 20 shows the top 10 most important variables contributing to each machine learning classification prediction. Figure 20 - 25 shows the breakdown, which is the contributing attribute of the variables. This can be indicated as red and green bars, which indicate either negative or positive changes in the mean prediction.

7.1.1 Machine Learning Classifier Variable Importance

Figure 16 to 20 shows the variable importance for all the five different classifiers. The variable importance highlights the 10 most important variables. The values of the variable importance are determined by the drop-out loss function. The dropout loss can be described as a function that quantifies the goodness of fit of a model, aka variable importance. In the

black box model, this can be described as the root mean square error (RMSE) or mean square root (MSE).

Figure 16

Variable Importance for Random Forest Classifier

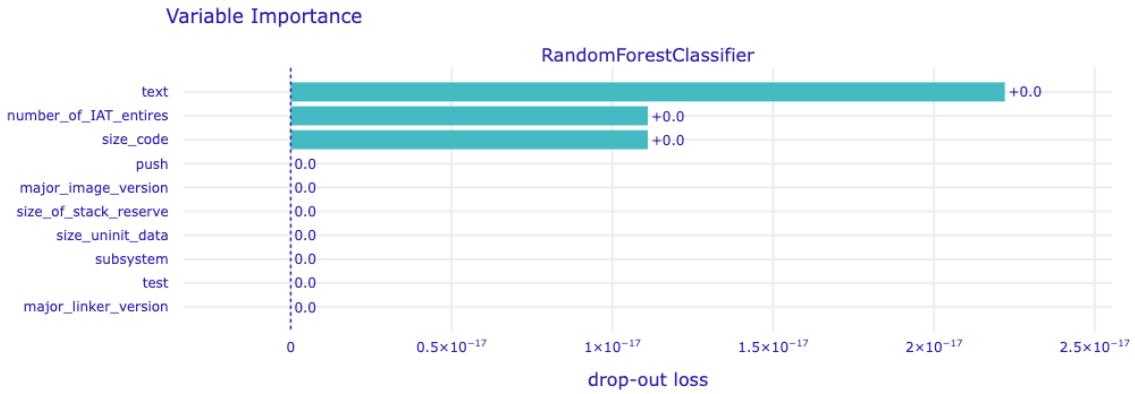


Figure 17

Variable Importance for Decision Tree Classifier

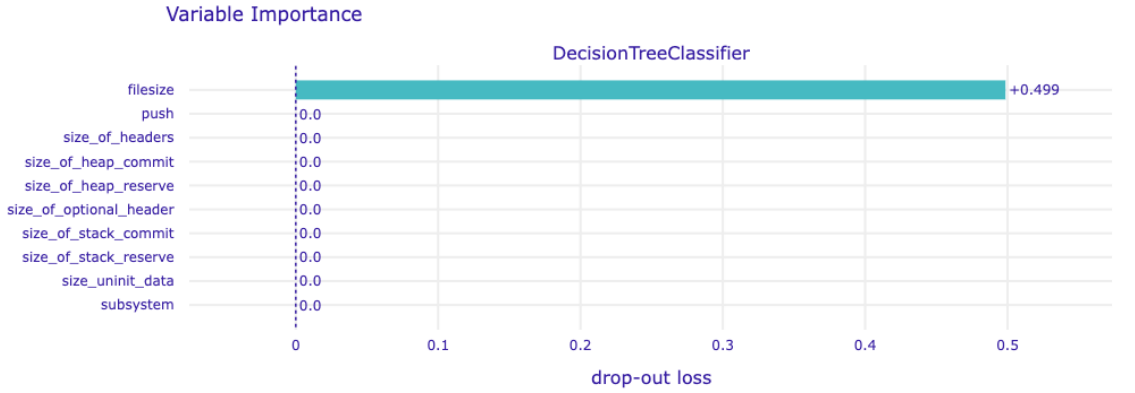


Figure 18

Variable Importance for Logistic Regression Classifier

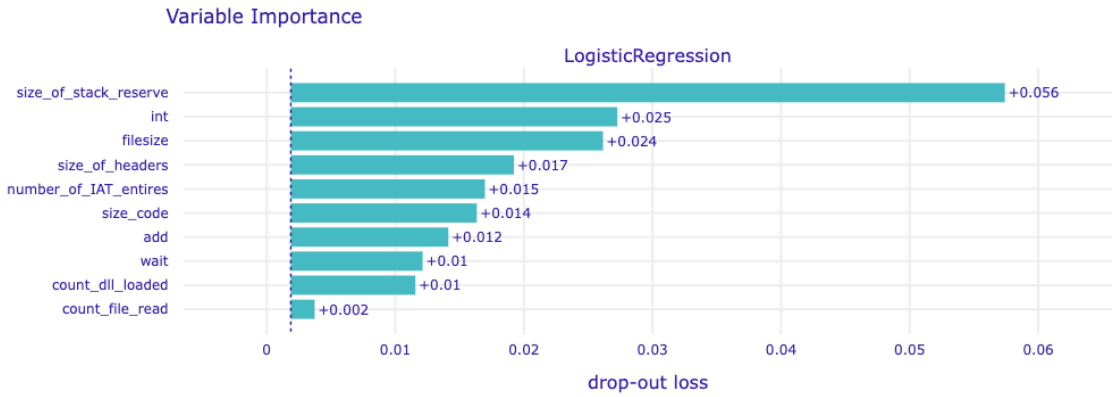


Figure 19

Variable Importance for Support Vector Machine Classifier

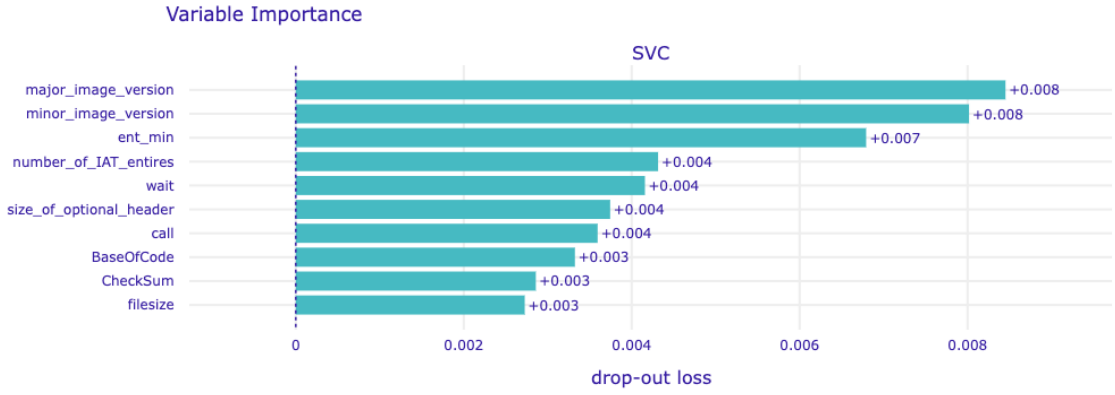
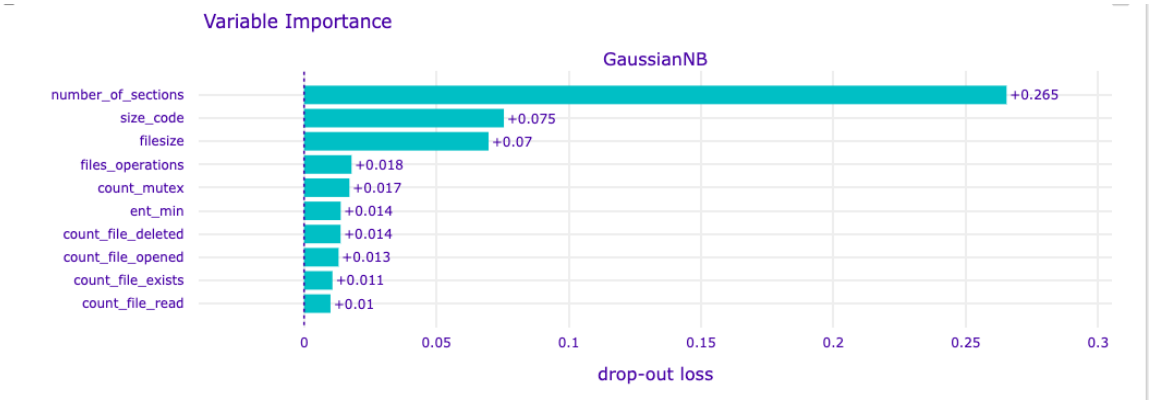


Figure 20

Variable Importance for Naive Bayes Classifier



7.1.2 Machine Learning Classifier Breakdown

Figure 21 to 25 show the breakdown for all the five different classifiers. These graphs show the name of important variables which are arranged from the highest to the lowest contributors. Their contribution values can either be positive (in green bars) or negative (in red bars).

Figure 21

Breakdown for Random Forest Classifier

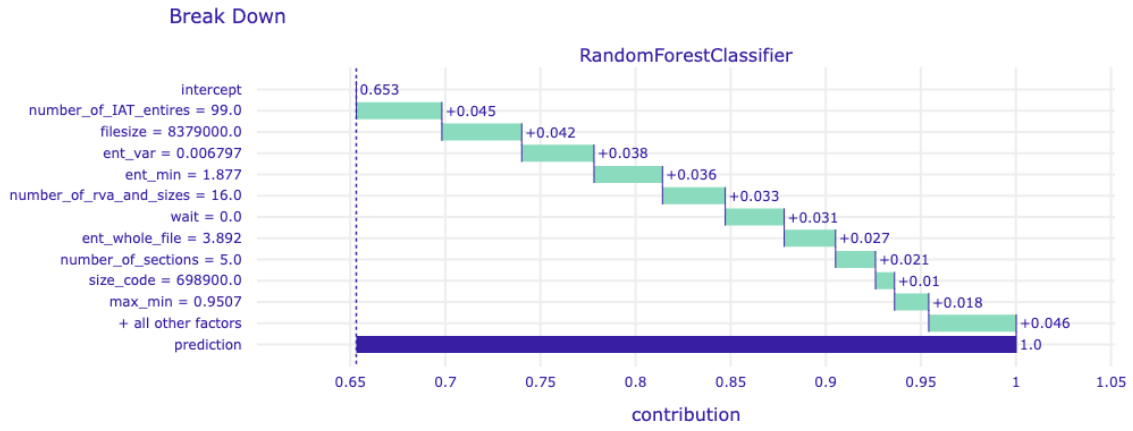


Figure 22

Breakdown for Decision Tree Classifier

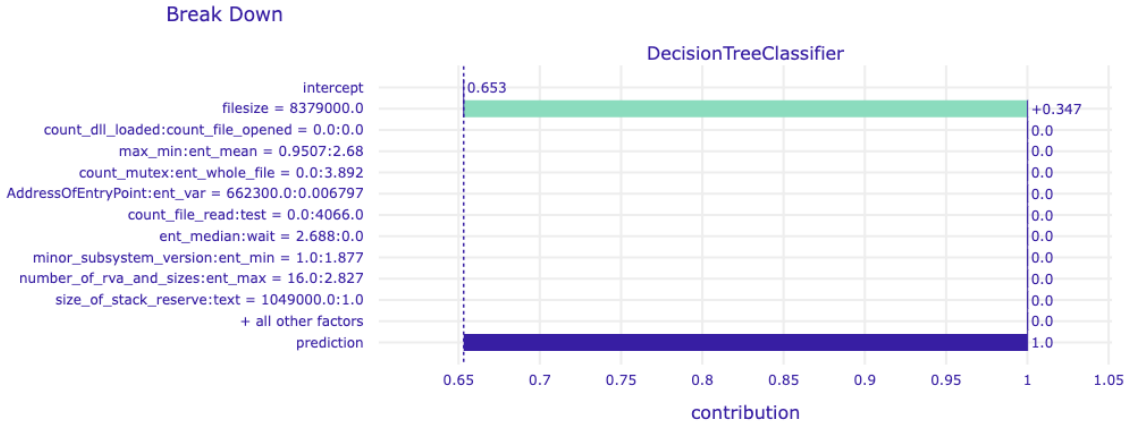


Figure 23

Breakdown for Logistic Regression Classifier

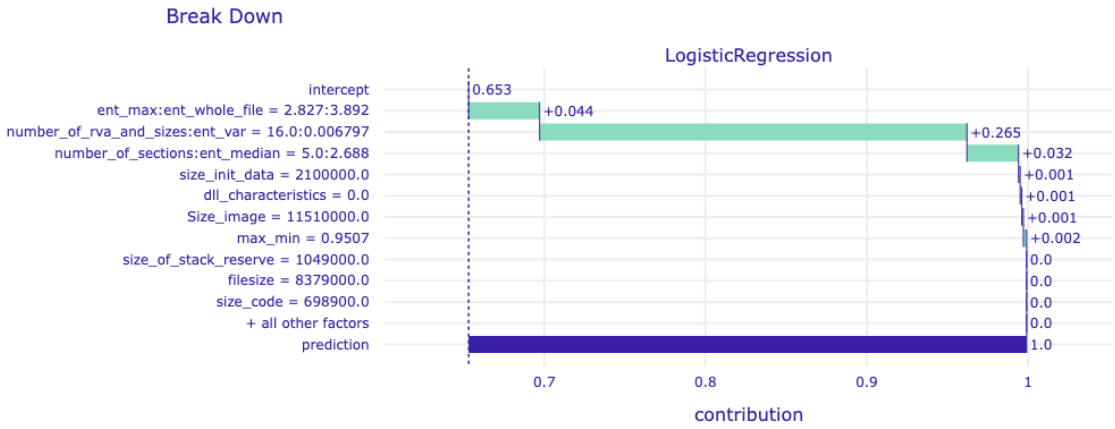


Figure 24

Breakdown for Naive Bayes Classifier

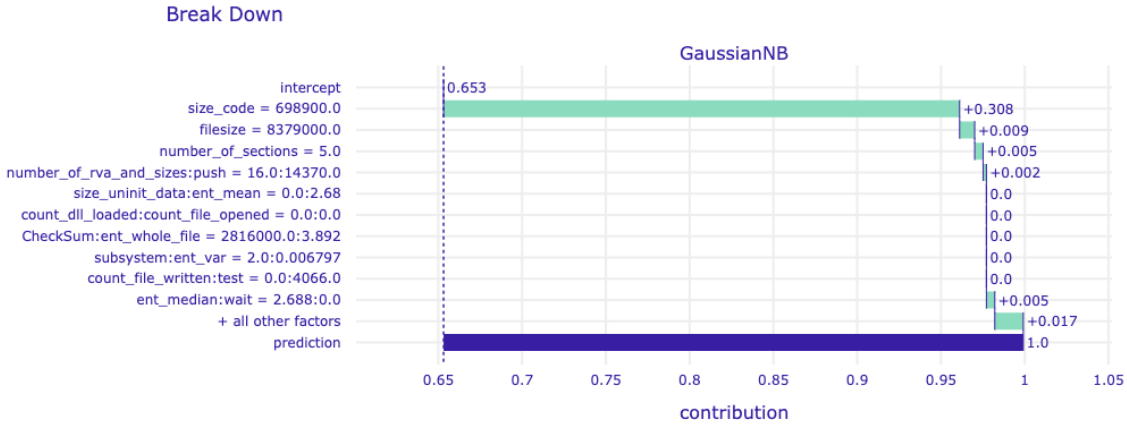
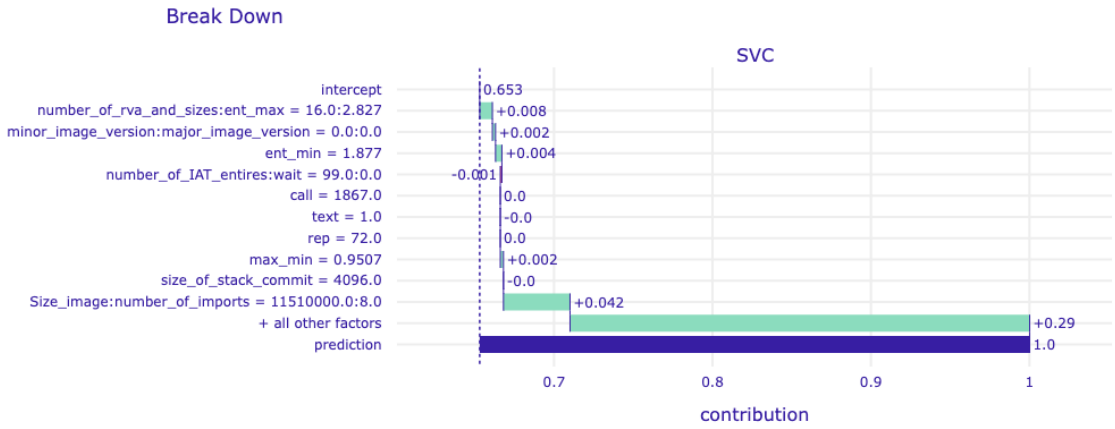


Figure 25

Breakdown for Support Vector Machine Classifier



7.2 Model Explainability on Machine Learning Models

Model explainability offers insight into what factors contribute to the predictive accuracy of a model. From the breakdown of the results, we discussed in section 6.2 above, we can see the high disparity between the results of 2 out of the 5 models. DALEX as a framework helps give us a visual understanding of what those features are and what weight of contribution, they have towards affecting the model's predictive result. For this experiment, we choose two insights, variable importance, and breakdowns, to help explain the model.

Before we do the comparative analysis of the results, it's important to explain some of the variables that are evaluated as contributors to the predictions.

- Number of IAT entries: IAT is an abbreviation of Import address tables. They are part of a dynamic linked library (dll) that helps to keep track of the address of functions that are gotten from other dlls. These dlls contain classes, functions, and resources such as images, icons, files, etc. The IAT is therefore regarded as a table that comprises function pointers. Whenever a system module is loaded, a call is made to a particular function, which then collects the address and stores it in the import address table. This is a honey pot for attackers because having access to multiple import addresses can overwrite these entries, thereby causing vulnerability to the system and making it easy for attackers to manipulate them for their own purposes. Attackers can use the "write-what-where vulnerability" to change the location of the pointer to wherever they want.
- Number of RVA and Size - This is described as the number of data directory items in the optional header.

- Size_of_uninit_data: This is one of the first eight fields that make up the optional header. It is known as the size of the uninitialized data, or where there are multiple sections, this would indicate the total number of sections, usually in bytes.
- Number of Sections: Sections can be described as the basic unit of code or data found in a portable executable (PE) or a common object file format (COFF). The number of sections there refers to the size of the section table.
- Push: This method call is recorded in bytes that saves the data sent to the pipe source. We can describe a pipe as a shared memory that can be used for communication. It is normally divided into two areas: the client and the server, which both write and read information from the pipe.
- File size: This is described as the size of the file, usually recorded in bytes. From our analysis, we can compare the result of the predictive result of our black box model and compare against the factors that contribute to those predictions. According to Table 2 of the Weka analysis, random forest has the highest prediction of 88.49%. Using DALEX as a tool for the model explanation, we can see that Number of IAT entries has the highest contribution. To understand how important this variable is, we must first understand what the variable is.

As shown in Figures 16 through 20, we can see that various dataset attributes accounted for both the variable importance and the breakdown of their contribution to the model. The variable importance of the random forest model was attributed to the amount of text and the number of IAT entries. These were the features that were accounted to be the most important out of the 62 in total. With a dropout loss for random forest ranging from 0.5 x 10^{-17} to 2.5 x 10^{-17} .

The dropout loss can be described as a function that quantifies the goodness of fit of a model, aka variable importance. In the black box model, this can be described as the root mean square error (RMSE) or mean square root (MSE).

The minimum value of what can be qualified as a "perfect" model is 0. The range goes from 0 - 1, with 1 being a failure and 0 being regarded as a success. Any dropout loss that has a value closer to 0.0 is more like to be the perfect model. From the above result, we can see that range for the random forest model given at 0.5×10^{-17} to 2.5×10^{-17} is much closer to zero hence why the prediction was the highest compared to the others. For the decision tree, we can see that its prediction has all 0's except for the variable "file size" which was 0.5. This reduced its overall accuracy. SVM and Naive Bayes, whose predictions are extremely low, have their variables closer to 1, indicating that the model was less successful in its prediction than the other models.

For the least performing classifiers, we can see that they have two similar variable importance, such as file size and ent min, in common with different levels of dropout losses. We can observe that high dropout loss in variable importance can be linked to lower performance in the model. Both Naive Bayes and SVM had dropout losses higher than zero, and they accounted for being the least performers.

The breakdown in Figures 21 through 25 shows the top 10 contributing features to the model predictions. Each of the classifiers has various contributions to the model. The colors are usually divided into two, green and red. The green color represents positive contributors, and the red represents the negative contributors to the model. The random forest is the highest classifier and has a breakdown contribution, as shown in Figure 21, with multiple positive variables contributing to the model. Comparing the breakdown

contributions to the weka black box classification result, we can see that random forest has the highest correct classification because more multiple variables with high variable weight were factored into the classification. SVM overall displayed the most diverse contribution to its classification. Being the only classification that possessed a negative contribution to the model. We can clearly see that the number of IAT entries, which was also one of its most important variables, had a negative impact on the model.

Similarly, Naive Bayes, the worse classifier, only focuses on 3 out of 10 features with very low contribution weight. From the result shown, it is important to note that classifiers that have greater (positive) single contributions have higher predictions than classifiers with multiple low contributions, which gives lower predictions. Therefore, the low classification is because SVM and Naive Bayes pay little attention to other attributes.

Chapter 8

Conclusions and Future Work

Malware such as ransomware has led to the loss of hundreds of millions of dollars yearly. With the invention of various strains and types, finding a long-lasting solution to eradicate these threats in cyberspace has become increasingly more work. Various solutions ranging from signature to behavioral have been developed to assist with not only the detection but the studying of these ransomware behaviors to find ways to mitigate the risk that they present. Machine learning as a tool for malware detection has become increasingly common to help detect malware. However, many machine learning methods can be considered black box models. While their algorithm might provide high results, decision makers in cyberspace have difficulty trusting them because there is no insight into the contributing factors of those model predictions.

In this research, we present a layered approach that is not just the classic signature or behavioral detection method but also introduces model explainability to help us break down all the important elements that contribute to why our model had the prediction results it generated. For this experiment, we started off by creating a test bed which is a virtual artificial environment that mimics a regular operating system and identifies malware interactions with user data. This testbed was made in windows operating systems which are windows 7 and windows 10. More of this was talked about in section 4.1 of this thesis. Using this testbed, we can study how malware generally affects our system, spreads, and continually persists in accessing the user's information. We added these manually tested samples to an already existing Virus Total data to build an entirely new dataset used for this experiment. For the second phase of this experiment, we built five machine learning

models to determine which classification algorithm performed the best. The machine learning classifications were Random Forest, Decision Tree, Naive Bayes, Logistic Regression, and Supervised Vector Machine (SVM). The study was carried out first in a black box model known as weka.

Three of the five observed models have performed very well, with an average of over 80% accuracy in classifying the malware and benign samples. Random forest and Decision tree had the best performance with 88.47% and 86% accuracy. The modes used in this experiment are wider than just predicting if a malicious sample is malicious or not. It can also be used in any area of malware detection, such as class groups, types, filenames, etc.

The study of malware, its behavior and how it spreads with a user's system is always a continuous study. Here, we could only look at the dataset for dynamic attributes alone. In order to have a holistic view of malware behaviors, we must look at static attributes as well. For future research, a comparison of machine learning performance between both static and dynamic attributes would be necessary to see which algorithm performs the best.

Other consideration for future works would include multi-label and multi-class for machine learning for attacks that are a can be classified as two different malware classes.

References

Biecek, P. (2019, July 9). [1907.04461] Model Development Process. arXiv. Retrieved November 17, 2022, from http://arxiv.org/abs/1907.04461

Automated malware analysis. Cuckoo Sandbox - Automated Malware Analysis. (n.d.). Retrieved November 17, 2022, from https://cuckoosandbox.org/

Alemzadeh, H., Raman, J., Leveson, N., Kalbarczyk, Z., & Iyer, R. (2016). Adverse events in robotic surgery: A retrospective study of 14 years of FDA data. 1–20. 10.1371/journal.pone.0151470

Baniecki, H., & Biecek, P. (2019, November). modelStudio: Interactive studio with explanations for ML predictive models. Journal of Open Source Software, 4(43) (1798). oss.theoj.org/papers/10.21105/joss.01798

Saleiro, P., Kuester, B., Hinkson, L., London, J., Stevens, A., Anisfeld, A., Rodolfa, K. T., & Ghani, R. (2018, November 14). Aequitas: A Bias and Fairness Audit Toolkit. arXiv. Retrieved November 18, 2022, from https://arxiv.org/abs/1811.05577

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why Should I Trust You?": Explaining the Predictions of Any Classifier. 1135–1144. 10.1145/2939672.2939778

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems, 4768–4777. https://dl.acm.org/doi/10.5555/3295222.3295230

Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viegas, F., & Wilson, J. (2020). The What-If Tool: Interactive Probing of Machine Learning Models. IEEE Transactions on Visualization and Computer Graphics, 26(1), 56–65. https://ieeexplore.ieee.org/abstract/document/8807255

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019, September 19). InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv. Retrieved November 18, 2022, from https://arxiv.org/abs/1909.09223

Jiangchun, L. (2018). PDPbox: python partial dependence plot toolbox. https:// github.com/SauceCat/PDPbox

Klaise, J., Looveren, A. V., Vacanti, G., & Coca, A. (2021). Alibi Explain: Algorithms for Explaining Machine Learning Models. Journal of Machine Learning Research, 22(181), 1-7. https://www.jmlr.org/papers/volume22/21-0017/21-0017.pdf

Right to explanation. (n.d.). Wikipedia. Retrieved November 17, 2022, from https://en.wikipedia.org/wiki/Right_to_explanation

Rufibach, K. (2010). Use of Brier score to assess binary predictions. Journal of Clinical Epidemiology 63: 938–39.

Cisco System. (2016). Midyear Security Report 2016. Cisco. Retrieved November 18, 2022, from https://www.cisco.com/c/dam/m/en_ca/never-better/assets/files/midyear-security-report-2016.pdf

Gandotra, E. , Bansal, D. and Sofat, S. (2014) Malware Analysis and Classification: A Survey. Journal of Information Security, **5**, 56-64. doi: 10.4236/jis.2014.52006.

Bidoki, S. M., Jalili, S., & Tajoddin, A. (2016, August). PbMMD: A novel policy based multi-process malware detection. Engineering Applications of Artificial Intelligence, 60, 57-70. http://dx.doi.org/10.1016/j.engappai.2016.12.008

Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K., & Xiao, Y. (2015, July 9). Network forensics analysis using wireshark. International Journal of Security Network, 10(2), 91 - 106. http://dx.doi.org/10.1504/IJSN.2015.070421

Hoque, N., Bhuyan, M.H., Baishya, R.C., Bhattacharyya, D.K., & Kalita, J.K. (2014). Network attacks: Taxonomy, tools and systems. journal of network and computer applications., 40, 307-324. https://doi.org/10.1016/j.jnca.2013.08.001

Eilam, E. (2005). Reversing: Secrets of Reverse Engineering. Wiley.

Ali Mirza, Q.K., Awan, I. Younas, M. (2018) Cloudintell: An intelligent malware detection system, Future Generation Computer. System., 86, 1042-1053 http://dx.doi.org/10.1016/j.future.2017.07.016.

Sikorski, M., & Honig, A. (2012). Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press.

T.Y. Wang, C.H. Wu, Detection of packed executables using support vector machines, in: Proceedings - International Conference on Machine Learning and Cybernetics, (2) (2011) pp. 717–722, http://dx.doi.org/10.1109/ICMLC.20116016774.

S. Abimannan, R. Kumaravelu, A mathematical model of HMST model on malware static analysis, Int. J. Inf. Secur. Priv. 13 (2019) 86–103, http://dxdoi.org/10.4018/IJISP.2019040106.

Abdessadki, I., & Lazaar, S. (2019). A New Classification Based Model for Malicious PE Files Detection. International Journal of Computer Network and Information Security(IJCNIS), 11(6), 1-9. 10.5815/ijcnis.2019.06.01

Singh, J., & Singh,, J. (2018). Challenges of malware analysis : Obfuscation techniques. International Journal of Information Security Science, 7(3). https://www.ijiss.org/ijiss/index.php/ijiss/article/view/327

Gao, Y., Lu, Z., Luo, Y. (2014). Survey on malware anti-analysis. Fifth International Conference on Intelligent Control and Information Processing. 270–275.

Alam, S., Horspool, R., Traore, I., Sogukpinar I., (2015, February). A framework for metamorphic malware analysis and real-time detection, Computer Security, 48, 212–233, http://dx.doi.org/10.1016/j.cose.2014.10.011.

Singh, J., & Singh, Dr. J. (2019). Ransomware: An Illustration of Malicious Cryptography. In International Journal of Recent Technology and Engineering. 8(2), 1608–1611. Blue Eyes Intelligence Engineering and Sciences Engineering and Sciences Publication . https://doi.org/10.35940/ijrte.b2327.078219

Hu, X. (2011). Large-scale malware analysis, detection, and signature generation (Order No. 3492834). Available from ProQuest Dissertations & Theses Global. (918832186). Retrieved from http://ezproxy.rowan.edu/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fdissertations-theses%2Flarge-scale-malware-analysis-detection-signature%2Fdocview%2F918832186%2Fse-2%3Faccountid%3D13605

P. Coogan (2010), Spyeye bot versus zeus bot, http://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot.

Mahdavifar, S., Ghorbani, A.A. (2019). Application of deep learning to cybersecurity: A survey, Neurocomputing, 347, 149–176, http://dx.doi.org/ 10.1016/j.neucom.2019.02.056, http://www.sciencedirect.com/science/article/ pii/S0925231219302954.

Zhang,W., Wang, H., He, H., Liu, P. (2020). DAMBA: Detecting android malware by ORGB analysis, IEEE Trans. Reliab. 69 (1), 55–69, http://dx.doi.org/10.1109/TR.2019.2924677.

Mirjalili, S., Mirjalili S. M, and Lewis A. (2014), ‘‘Grey wolf optimizer,’’ Adv. Eng. Softw., Mar, vol. 69, pp. 46–61.

X.Yang and Suash Deb, ‘‘Cuckoo search via Lévy flights,’’ in Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC), 2009, pp. 210–214.

Homayoun, S., Dehghantanha A., M. Ahmadzadeh, S. Hashemi and R. Khayami (2020), "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," in IEEE Transactions on Emerging Topics in Computing, vol. 8, no. 2, 1 April-June pp. 341-351, doi: 10.1109/TETC.2017.2756908.

Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E. (2016): Unveil: a large-scale, automated approach to detecting ransomware. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 757–772. USENIX Association, Austin

Kharraz A, Kirda E (2017) Redemption: real-time protection against ransomware at end-hosts. In: International symposium on research in attacks, intrusions, and defenses, Springer, pp. 98–119

Mehnaz S, Mudgerikar A, Bertino E (2018) Rwguard: a real-time detection system against cryptographic ransomware. In: International symposium on research in attacks, intrusions, and defenses,Springer, pp 114–136

Garfinkel, T., Rosenblum, M., 2003. A virtual machine introspection-based architecture for intrusion detection. In: Proceedings of the 10th Network and Distributed System Security Symposium, pp. 191–206.

Tang, Fei & Ma, Boyang & Li, Jinku & Zhang, Fengwei & Su, Jipeng & Ma, Jianfeng. (2020). RansomSpector: An Introspection-Based Approach to Detect Crypto Ransomware. Computers & Security. 97. 101997. 10.1016/j.cose.2020.101997

Kardile, A.B. (2017), Crypto Ransomware Analysis and Detection Using Process Monitor. Ph.D. Thesis, The University of Texas at Arlington, Arlington, TX, USA.

Scaife, N., Carter, H., Traynor, P., Butler, K.R.B. (2016): Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), June, pp. 303–312. https://doi.org/10.1109/ICDCS.2016.4627.

Baek, Sungha & Jung, Youngdon & Mohaisen, Aziz & Lee, Sungjin & Nyang, Daehun. (2020). SSD-Assisted Ransomware Detection and Data Recovery Techniques. IEEE Transactions on Computers. PP. 1-1. 10.1109/TC.2020.3011214

Alotaibi, Fahad & Vassilakis, Vassilios. (2021). SDN-Based Detection of Self-Propagating Ransomware: The Case of BadRabbit. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3058897.

Morato, Daniel & Berrueta, Eduardo & Magaña, Eduardo & Izal, Mikel. (2018). Ransomware early detection by the analysis of file-sharing traffic. Journal of Network and Computer Applications. 124. 10.1016/j.jnca.2018.09.013.

Khan, Firoz & Ncube, Dr & Ramasamy, Lakshmana & Kadry, Seifedine & Nam, Yunyoung. (2020), "A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning", IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3003785.

Fumo, David. "Types of Machine Learning Algorithms You Should Know." *Towards Data Science*, Towards Data Science, 15 June 2017, https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861. Accessed 15 December 2022.

Jurriaan Bremer. (2013). Blackhat 2013 workshop: Cuckoo sandbox - open source automated malware analysis. http://cuckoosandbox.org/2013-07-27-blackhat-las-vegas-2013.html.

Biecek, Przemyslaw. (2018). "DALEX: Explainers for Complex Predictive Models in R." Journal of Machine Learning Research, vol. 19, no. 84, pp. 1-5, https://jmlr.org/papers/v19/18-416.html.

Ogiriki, Ikuromor, Vahid Heydari, Christopher Beck (2022) “Technical Analysis of Thanos Ransomware.” 17th International Conference on Cyber Warfare and Security, vol. 17, no. 1, pp. 497-504, https://doi.org/10.34190/iccws.17.1.62.

Lang M, Binder M, Richter J, Schratz P, Pfisterer F, Coors S, Au Q, Casalicchio G, Kotthoff L, Bischl B (2019). “mlr3: A modern object-oriented machine learning framework in R.” *Journal of Open Source Software*. doi:10.21105/joss.01903, https://joss.theoj.org/papers/10.21105/joss.01903.