Machine Learning Models For Static Malware Analysis.

Anand Mohan Tripathy

Applied project submitted in partial fulfilment of the requirements for the degree of

MSc in Data Analytics

at

Dublin Business School

Supervisor: David Williams

May 2020

INDEX

Contents

DECLARATION

This page should be a declaration that the completed work in the applied project was completed by you.
‘I declare that this applied project that I have submitted to Dublin Business School for the award of [MSc in Data Analytics] is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.’

Signed: Anand Mohan Tripathy

Student Number: 10527557

Date:24/08/2020

ACKNOWLEDGMENTS

“ I wish to express my sincere appreciation to my supervisor, Professor David Williams , for his time and valuable guidance that have been of valuable help during the writing of this Dissertation.”

I would also like to express my gratitude to Professor Amir Esmaeily for providing guidance in preparing the Research Method Proposal and for making me understand the way research is conducted.

Lastly, I would also like to thank my family and Friends for being always there for me, without their support I could not have taken up this study. The constant support I received from them and also motivating me during this Dissertation time and throughout my master’s course.

ABSTRACT

While conventional malware detection approaches increasingly failed, modern heuristic strategies often performed dynamically, which was not possible in many applications due to related effort and the quantity of files.

The aim for the current work was to analyze the machine leaning models to find out an approach towards static malware detection of PE executables. One benefit could be its simplicity (evaluating 1002 static features), so it might support the application with large file amounts, e.g. network-operators or a posteriori analyses in archival systems. After identifying promising features and their typical values, a traditional machine learning and neural network approach using the python and RapidMiner machine learning tool were generated and evaluated. The results of large-scale classifications showing the statistical were compared analysis to find out the most suitable model. To conclude, malicious samples often have special characteristics which are effectively supported by existing malware-scanners.

Keywords: Static malware detection, Data, Correlation, Decision tree, Neural Net, Support Vector Machines, Dimensionality reduction techniques, Statistics analysis , SMOTE.

1. INTRODUCTION

The usage of advanced analytic techniques that control historic data to uncover instantaneous understanding and to forecast forthcoming actions is known as Predictive analytics. Predictive analytics is the practice of extracting information from existing datasets in order to determine patterns and predict future outcomes and trends. Predictive analytics does not tell what will happen in the future. Instead, it forecasts what might happen in the future with an acceptable level of reliability and includes what-if scenarios and risk assessment.(Shmueli,2011). Predictive analytics and modeling are more accessible than ever before in today's era as it has extraordinary union of instinctive gears, fresh predictive procedures, as well as hybrid cloud deployment prototypes. Predictive analytics can be embedded into business operations for any type of firm. Anything from raw text to more complex data can be extracted, i.e. 'dark data' to useful data.

Internet is similar to a Battlezone, the attackers and the firm have a very short distance , the firm's IT teams and experts act as frontline and play role protecting important assets. A data breach can be predicted well in advance by suggesting detail of time and place of occurrence by using predictive analysis, similar to what radar does when the enemy is approaching in a battlefield. This gives an organization time to raise an alarm, decide the strategy, and ready its team.

Therefore, Predictive analytics in such scenarios has enabled to outplay hackers and come out of the threat safely. Predictive analytics also enabled the firms with the power of dynamical detection of dangers beforehand any damage and ultimately helped the businesses. Organizations with the help of predictive analytics could discover future incidents to maximize prevention, rather than preparing for infection stage. The security procedures taken for this were a combination of humans and bots where bots checked for non-uniformity and humans checked for overall security of the system. Security no doubt has always been a topmost priority for any firm's strategy. Organizations currently are susceptible to advanced attacks and are at risk losing delicate and important data and there seems no end to cyber-attacks, but the modern tools give assurance for tackling such situations.(Xu et al., 2018).

In this research paper , the aim of the study was to find out the predictive machine learning model for malware detection using machine learning life cycle which would help organizations to detect the data breach before it actually happened. The machine learning and deep learning could allow malware detection to discard its present clumsy blacklist policy and could also help find out nearing threats with confidence. If information on a hacking pattern or degree of a detection in the future was predicted in advance to a computer user, similar to a weather forecast, it would help to make ready for and manage or survive with a cyber threat. Presently, there exist technologies like Machine learning algorithms such as Decision Tree (DT),K-Nearest Neighbor (KNN), Naïve Bayes (NB), and Support Vector Machine (SVM) which are commonly used in malware detection .(Fei Xiao and Zhaowen Lin, 2019). The traditional machine learning algorithms have potentially learnt the behavior features from the malware. For the research we are finding performance of the machine learning algorithms' which depends on the accuracy of the extracted features. In addition, it is often difficult to extract meaningful behavior features for improving malware detection performance. Hence , used dimensionality reduction to extract meaningful behavior features. Moreover, feature processing also required expertise and the methodologies. Methodology is not dependent on any technology. Rather, a methodology provided a structure to accumulate outcomes by employing an extensive collection of methods, processes, and heuristics.

For this Research, CRISP-DM was carefully selected, being most widespread method. Also, the combination with commercial products was determined. In chapter 1, an introduction for malware classification was examined, which might also be useful to support and extend existing approaches. In chapter 2, looked back the relevant research's on malware detection to comprehend from different aspect which were considered in our investigations. In Chapter 3, described the methodologies, implementations, and new approaches. Also introduced the selected features chosen for evaluation. The Research-based and machine learning classification models and neural network are also explained. In section 4, the test setup was introduced and the results of the classifications in practical large-scale test were presented and compared. Section 5 concluded this paper and presented a short outlook.

1.1 Research Questions

Based on the Finding it stated, it was important to form research questions that shall lead to the final conclusion.

- To find the best machine learning algorithms while using machine learning life cycle and using CRISP-DM methodologies.
- **Objective:** For malware detection, it is stated that machine learning models like logistic Regression, Decision Tree and random forest are the best suitable models. To find the best accuracy and recall on the machine learning algorithms.

1.2 Research Approach and Assumptions

As stated in the introduction, the chosen approach was to perform a statistical evaluation of structural features of files which are in the form of hash file. This heuristic approach on static features was expected to be a reasonable addition to former static approaches (since heuristic detection strategies today frequently rely on dynamic evaluations). The approach assumed that many malicious code files contained structural indications which would lead to a successful identification in automated classifications. Some potential reasons which led to cause such indications are as below:

- ➢ Structural changes performed by malicious code when bound to an existing (previously non-malicious) binary.
- ➢ Binaries built by malicious code generators having characteristic structural differences compared to programs built in commercial development environments (e.g. non-conformance to the PE specifications in certain issues which are not affecting its ability to be executed).
- ➢ General malicious code to which characteristic packers, encrypts or other obfuscations have been applied. Structural abnormalities caused by low-level programming techniques.
- ➢ Characteristic functional features of malicious code, e.g. the inclusion of critical combinations of API functions.
- ➢ Messages left by the malware authors (usually text strings).

The investigations and practical examinations in the following phases are Based on the above assumptions.

2. LITERATURE REVIEW

In the previous couple of years, the sector of data systems made great advancements in implementing more advanced statistical techniques to support factual or verifiable observation. Parallelly there had been various opportunities for further development as there were particularly large gaps specifically in mainstream empirical IS research due to immediate absence of machine learning . This gap brought an essential opportunity, because machine learning was useful for producing new theory, equating competing theories, developing new measures, improving current theories, assessing the relevance of theories, and evaluating the predictability of pragmatic phenomena. Machine learning comprised of statistical models and empirical methods that were intended toward creating empirical predictions (as in contradiction of predictions that follow from theory only) also it approached for assessing the standard of the predictions in practice . Apart from their practical usefulness, predictive analytics played a crucial role in theory building, theory testing and relevance assessment. Hence, it is was necessary component of research project. Predictive analytics explained the use of statistics and modelling to assess potential success based on current and historical data. Predictive analytics examined trends in data to be practiced if those trends were likely to re-emerge, allowing businesses and investors to determine and control where their money will be used to optimize the potential future events. There have been several sorts of predictive analytics methods available. For instance, data mining contained the examination of large tranches of data to uncover trends therefrom . Text analysis did something similar ,the exception of large blocks of text. Predictive models were analyzing past data to assess the likelihood of certain future outcomes, while descriptive models looked at historical data to assess how a group might respond to a set of variables. (Shmueli and Koppius, 2011)

Predictive analytics had become a policy making tool in many industries. For example, insurance firms examined policy applicants to assess the probability of having to pay out on a potential claim based on the existing risk pool of similar policyholders, as well as past incidents that resulted in payout . Dealers looked at how clients had responded to the overall economy when preparing on a new campaign and used changes in demographics to determine if the present mix of products would entice customers to make a purchase. Active traders looked at a range of metrics based on former events when deciding whether to buy or sell a security. Moving averages, bands and break points were established on historical data and was used to predict future price movements. A usual delusion was that predictive analytics and machine learning were the identical things. Fundamentally, predictive analytics comprised of a series of statistical techniques (which included predictive modelling, machine learning and data mining) and used statistics (both historical and current) to estimate, or forecast, future outcomes. Predictive analytics helped to recognize possible future incidences by analyzing the past. While machine learning, on the other hand, was a subfield of computer science that, as per the 1959 definition by Arthur Samuel—an American pioneer in the field of computer gaming and artificial intelligence, machine learning also gave "computers the ability to learn without being explicitly programmed." The most common predictive models comprised of decision trees, regressions (linear and logistic) and neural networks (which is the developing field of deep learning methods and technologies).(‘Kaspersky-Lab-Whitepaper-Machine-Learning.pdf’, no date)

The internet could be descried as a global network of interconnected systems which served billions of users worldwide. Its popularity and rapid growth had come at an expensive cost, i.e. loss of data and resources due to cyber threats and attacks. The first cybercrime was reported in 2000 and infected almost 45 million internet users(Xu *et al.*, 2018). The quick evolution of the Internet had also brought with it an exponential increase in the variety and occurrence of cyber-attacks. In the past couple years cyber-crimes had increased swiftly with cyber criminals incessantly discovering new ways to circumvent security solutions to acquire illegal access to computer systems and computer networks. Various well-known cybersecurity solutions were in place to check and control these attacks. Analytics could help network admins predominantly in the monitoring and surveillance of real-time network streams and recognition of both malicious and suspicious patterns. Such a conduct was intended to enrich all conventional security techniques The 2019 Internet Crime Study highlighted the efforts made by the IC3 to track trend-setting scams such as Company Email Compromise (BEC), Malware, Elder Fraud and Computer Fraud. IC3 received a complete of 467,361 reports in 2019, with reported losses approaching $3.5 billion, as the study stated.

The most popular types of crime reported have been Phishing / Vishing / Smishing / Pharming, Non-Payment / Non-Delivery, Extortion and Private Data Breach. BEC, Confidence / Romance Fraud and Spoofing (IC3,2019), were the top three forms of crime with the highest recorded losses.

- **Some important cyber-attacks were as follows:**

- **Spamming :**Spamming was described as sending unsolicited bulk messages to multiple recipients. By 2015, the spam frequency was predicted to be 95 percent of whole email traffic. Munging, access filtering and content filtering were some main anti-spam techniques. Munging makes email addresses unusable for spammers, e.g. abc@gmail.com. Access filtering detects spam based on IP and email addresses while content filtering recognizes predefined text patterns in emails to detect spam.
- **Search-Poisoning:** Search poisoning was the dishonest use of Search Engine Optimization techniques to falsely improve the ranking of a webpage. Usually, frequent search keywords were practiced to illegitimately direct users towards short-term websites. The earliest poisoning case was reported in 2007.
- **Denial of Service (DoS) :**A DoS attack made a system or any other network resource inaccessible to its intended users. It would be launched by a big system of distributed hosts, e.g. bot net. Several defensive techniques like intrusion detection systems, puzzle solution, firewalls etc. were built to check DoS attacks.
- **Botnets:** Botnets were networks of malware-infected or compromised computers managed by an adversary. Attackers made use of bot software prepared with integrated command and controlled system to govern those bots and categorized them into a network called the bot net.
- **Malware:** Malware was a software programmed to perform and propagate malicious activities, e.g. viruses, worms, and Trojans. Viruses required human intervention for propagation, worms were self-propagating, while Trojans were non-self-replicating. Destruction from malware comprised of corruption of data/operating system, installation of spyware, stealing personal credentials/hard disk space etc.
- **Website Threats** :Website threats referred to attackers exploiting vulnerabilities in legitimate website, infecting them, and indirectly attacking visitors of these sites. SQL injections, malicious ads, search result redirection were some techniques which was used to infect the legitimate sites(Afzal,2013).

Cyber threats of the future called for cybersecurity– and crucial technologies like predictive analytics would play a very important role to safeguard from threat, as the Internet had become a significant part of people's everyday life as well as of organizations. It also played a vital role as it allowed collaboration to take place between individuals and organization, which enabled them to communicate and carry out business processes. Internet connectivity provided vast benefits in relations of increased access to knowledge and information but using the Internet could become a hazardous experience where users and organizations lacked the appropriate levels of security awareness and knowledge. Anyone that used a computer might had encountered a direct or indirect experience with malware especially if the computer was connected to the internet. Malware was one of the Major cyber-attack which affected the internet over years. Details about Malware have been discussed below.(Mokoena and Zuva, 2017)

2.1 Malware

Malware was referred to the short form for malicious software which referred to software that was designed to trigger harm or damage or even perform other unwanted actions on a computer system. Malware conduct could range from an unwanted advertising pop-up's, to actions which were much more harmful like theft of credentials and sensitive data even infecting other machines on the network. A huge number of malware variations used to automatically generate per day. Recent Symantec report showed that new pieces of malware grew by 36 percent from the year before in 2015 with total samples exceeding 430 million. Exponential progress of malware produced a substantial threat in everyday life. Malware was also designed for financial gain such as ransomware malware and while others were designed to gather sensitive information, gain access to private computer systems, or display unwanted advertising , some were even designed or were used to accomplish civil (e.g. the Anonymous group) or politically motivated attacks. The Malware family included trojan horses, worms, spyware, and adware. Advanced malware like ransomware were made use of to commit financial fraud and extract money from cyber users. Traditional computers used to get a lot of attacks in IoT world. Malware attacked computers and further used the infected machine to attack the other associated devices in IoT environment. For instance, Trojan.Mirai.1 which was the variant of Mirai could corrupt windows hosts and could further exploit those hosts to infect other attached devices. The infected windows would steal confidential information and transform the influenced devices into a botnet to launch a new Distributed Denial of Service (DDoS)attack. Various present traditional computers malware attacks were also spread to other IoT devices. Unfortunately, there was no perfect solutions to evade Mirai and other IoT threats. The method only aimed to weaken those threats by shielding the security of traditional computers in IoT environment. Malware attacks have been around since the first well known malicious software, which was written in the early 1950s, widely known as a “self-reproducing machine”. The evolution of malware over the years has contributed towards major security incidents or breaches which have caused major financial losses as well as reputational damage to many organizations. One of the examples of these malware attacks was the Sony Pictures hack in December 2014. According to a number of reports and security researchers, the malware that infected computers systems at Sony Pictures was named “Wiper” malware. This malware was also involved in vast majority of other similar attacks according to the Verizon’s 2016 Data Breach Investigation Report.

The increase of malware attacks and the increasingly resourceful ways in which it was being used to commit crime such as to conduct espionage, steal personal information, cause destruction to governments and business operations, or deny the user access to information and services as well as defacing websites was potentially a serious threat to the Internet economy.

Malware was one of the most dreadful and major security threats faced on the Internet then. According to a survey, conducted by FireEye in June 2013, 47% of the organizations experienced malware security incidents/network breaches in the past one year. The malwares were constantly increasing in volume (increasing threat landscape), variety (innovative malicious techniques) and velocity (fluidity of threats) . These were evolving, becoming more sophisticated and using new ways to target computers and mobile devices. McAfee catalogued over 100,000 new malware samples every day which meant about 69 new threats every minute or about one threat per second. With the surge in readily available and complex tools, the new generation cyber threats or cyber-attacks became more targeted, persistent, and mysterious.(Gandotra, Bansal and Sofat, 2014) The innovative malwares were targeted, unidentified, stealthy, tailored and zero day as compared to the traditional malwares which were broad, known, open and one time. Once inside, they would hide, duplicate, and deactivate host protections. After getting installed, they executed their command and controlled servers for additional instructions, which would be to steal data, corrupt other machines, and permit reconnaissance.

Attackers misused the vulnerabilities in web services, browsers, and operating systems, to exploit social engineering methods making consumers run the malicious code to spread malwares. Malware authors used complex techniques such as dead code insertion , instruction substitution , register reassignment, code transposition, subroutine reordering and code integration to avoid detection by conventional suspicions like firewalls, antivirus and gateways which normally used signature based systems and were unable to gauge the previously unnoticed malicious executables. Commercial antivirus vendors were not able to propose instant protection for zero-day malwares as they needed to examine these to generate their signatures. To overcome the limitation of signature-based approaches, malware analysis methods were being followed, which could be either static or dynamic. The malware analysis techniques helped the analysts to comprehend the risks and intensions accompanying with a malicious code sample. The insight so gained was used to tackle new trends in malware growth or take defensive measures to cope with the risks coming in future. Features derived from examination of malware were used to group unidentified malwares and categories them into their existing families.

2.2 Malware Analysis

Research into malware detectionvolved to address static detection inadequacies.Behavioral analy sis (dynamic analysis) examined a sample file while it was being executed in a virtual environment.
The approaches for conducting behavioural analysis assumed that malware could not avoid leavi ng a measurable footprint as a result of the actions needed to achieve its goals. Executing the mal ware did, however, enforce a time penalty as compared with static analysis. Although dynamic d ata leaded to more accurate and robust models of detection than static data, due to this time penalty in practise behavioural data was seldom used in commercial endpoint anti-virus systems.

Waiting for several minutes when a single file was being analysed was tedious and wasteful, and in the end the malicious payload was possibly delivered by the end of the review window so the opportunity to stop malicious behaviour was lost. Some approaches monitored "live" activity on the local network or on machine to avoid waiting.Such detection systems managed to either searc

h for features that indicated a specific form of malware (e.g. ransomware) or flag anomalies from a "standard" behaviour base line. Such methods experienced different shortcomings. (Mokoena and Zuva, 2017)
.Searching for different behaviours was similar to conventional methods of comparing incoming files with known variants, which sometimes failed to identify new malwaretypes. Although detec tion of anomalies was vulnerable to a high false positive rate, any behaviors deviating from a "natural"baseline was considered to be malicious. Human analysts often investigated anomalous activity in practice which made the model vulnerable to exploitation.

An attacker could bring lots of anomalous behaviour in such a way that human analysts could be flooded with inquiry requests, reducing the chances of detection of the activity created by the attack itself.
Malware analysis was defined as the process of malware dissecting to understand how it worked by using static and dynamic inspection of various tools, methods, and processes. It was the process to understand how malware could be identified and how malware could be defeated or eliminated.

This method was an important step in developing successful detection techniques to counter malicious code, and the development of software that can remove malware from an infected device was also a significant requirement.

Data was collected by dissecting malware using extraction and monitoring techniques, the methods and processes needed to collect information about the malware successfully differed depending on the malware 's actions and capacity.

Diverse tools, approaches, and processes were used to retrieve information from the malware analyzed without disassembling it, allowing malware to be analyzed in an isolated managed environment for the purpose of gathering and tracking information that can be used Reveal the true

intentions of this malware. The process of breaking down binary into low-level code such as assembly code is known as disassembly.(Zuva, 2017)

The Malware detection method is comprised of two types: static and dynamic.

2.2 Malware Analysis Types

Malware analysis was described previously as that the process of dissecting malware to understand how it worked. Malware analysis was traditionally known as a tedious and manual process.

Time usage and the number of samples that security vendors needed to review on a regular basis was gradually growing. The approach used static and dynamic inspection of different instruments, methods and processes to Understand how malware could be detected , and how malware was defeated or removed. The method of malware analysis consisted of two types of analyses that were static and dynamic analyses.

Static analysis was also the method of evaluating a programmed by reviewing its properties or code. The process of executing a program was known as Dynamic Analysis to analyses it.

❖ Static Analysis

Static malware analysis was described as the process of extracting malware information while it was not running to determine its true purpose by analysing the malware 's code. Extracting infor mation from malware involved inspecting disassembly lists, extracting strings, obtaining a virus signatures, identifying target and compiler architecture that were being used, as well as several ot her malware features. In this method, the malware programme or artefact was not executed whic h then required specialised tools required for thorough analysis of the malware.

Detection patterns used in static analysis included string signature, byte sequence n-grams, syntactic library call, control flow graph and frequency distribution opcode (operational code) etc. Before doing static analysis, the executable were unpacked and decrypted. The disassembler / debugger and memory dumper software were used sometimes to reverse the executables of compiling windows. Disassemble / Debugger tools such as IDA Pro and OllyDbg displayed the code of the malware as instructions for assembling Intel half86, provided a lot of insight into what the malware was doing and patterns for identifying attackers.

Memory dumper tools like LordPE and OllyDump were used to get secure

code in the memory of the device and dump it into a register. This was a useful technique for the analysis of packed executables that were hard to disassemble. Binary obfuscation techniques, which turned malware binaries into self-compressed and uniquely organized binary files, were designed to withstand reverse engineering and thus made the static analysis very costly and ineffective.

In addition, when binary executables (obtained by compiling source code) were used for static analysis, information such as data structure size or variables got lost leading to complicate the analysis of malware code.

The evolving evasion techniques used to thwart static analysis by malware writers led to the development of dynamic analysis. Moser et al., discussed the disadvantages of the technique of static analysis.

They introduced a code obfuscation-based scheme in their work revealed the fact that static analysis alone was not enough to detect or classify malware.

In addition, they indicated that dynamic analysis was a required complement to static analysis, as it was less vulnerable to conversion of code obfuscation.

❖ Dynamic Analysis:

Dynamic analysis was defined as the process performed to extract information from malware. This process involved executing the malware artefact in a secure isolated environment, unlike the process of static analysis that provided only a view of the analyzed malware. Dynamic analysis offered an exhaustive view of the true purpose, features, and work of the malware as knowledge about the malware was collected while the malware was running. Since Dynamic Malware Analysis was conducted during runtime and malware unpacked itself, this method evaded the constraints of static analysis, which was the problems of unpackaging and obfuscation. It was therefore easy to see a programmer's actual actions.

Examination of the actions of a malicious code (system interaction) when performing it in a man aged environment (virtual machine, simulator, emulator, sandbox, etc.) was called dynamic exam ination. Appropriate monitoring tools such as Process Monitor and Catch BAT (for file system a nd registry monitoring), ProcessExplorer and Process Hacker Replace (for process monitoring), Wireshark (for network monitoring), and Regshot (for system change detection) were installed and allowed before executing the malware sample. Different techniques that where be applied to perform dynamic analysis included monitoring of the call function, analysis of task parameters, tracking of information flow, traces of instructions and extensibility points of AutoStart, etc.

Dynamic analysis was more efficient than static analysis and did not require assembling of the executable. It exposed the normal behavior of the malwares which was more resilient to static analysis. Nevertheless, it was time-intensive and resource-consuming, there by elevated the problems of scalability. The virtual environment in which malware was executed varied from the real world, and the malware could behave differently resulting in artificial behavior rather than the same one.

In addition, malware activity was often activated only under certain circumstances(on a particular device date or through a specific command) and could not be detected in virtual environment.

There were many online automated tools available for complex malware analysis, e.g. Norman Sandbox, CWSandbox, Anubis, Ether and Risk Professional , and TTAnalyzer. The analysis reports created by these tools provided a detailed understanding of the malware activity and useful insight into their actions.

The detection method was expected to provide an adequate representation for malware, which was then used either on the basis of a similarity measure or feature vectors for classification. But a large number of new malware samples arriving every day at anti-virus vendors needed an automated approach to reduce the number of samples needing near human analysis. Several Artificial Intelligence techniques had been used in the literature for automated detection and classification of malware, in particular machine-learning based techniques.

Windows Application Programming Interface (API) calling the graph-based method was long been seen as a good prospect for behavior-based malware detection.

For malware detection, machine learning algorithms such as Decision Tree (DT), K-Nearest Neighbor (KNN), Naïve Bayes (NB) , and Support Vector Machine (SVM) were widely used. Current machine learning algorithms could theoretically learn the features of the malware 's actions. Sadly, the output of most machine learning algorithms was based on the precision of the features extracted. In addition, it was often hard to extract useful behavioral features to boost efficiency in malware detection. In addition, the processing of features required know-how. Thus, traditional machine learning algorithms for malware detection somewhat unsatisfactory. Deep learning , a branch of machine learning that seeked to learn features of a high level directly from the original data.

In short, deep learning and traditional machine learning specifically supported end-to-end solution. It removed the entire cycle of the grand and demanding phase of the project.

Deep learning was effective in the study of high-level sample features using multi-layer deep architecture, and was commonly been used in image processing, visual recognition, object detection, etc. This introduced a method of protecting IoT devices against attack by local computers.(Fei Xiao and Zhaowen Lin, 2019)

3. METHOD

3.1 Methodology: Standard Cross-Industry Data Mining Process (CRISP-DM).

CRISP-DM was carefully selected for this analysis as it was one of the most common, but SAS SEMMA was not in the study as its use was substantially reduced over the period of time.

The KDD method was also counted in because it provided the basic and primary information spotting criteria. TDSP was chosen because it was targeted to machine learning or AI technologies that were substantially linked to malware detection.

It consists of a high-level sequence of six levels, which determined the analytical procedure.

CRISP-DM was a good tool, but it needed to improve the information and data for malware detection projects. CRISP-DM was highly well known and applied in numerous case studies. Below diagram shows the six phases in every series of CRISP-DM.

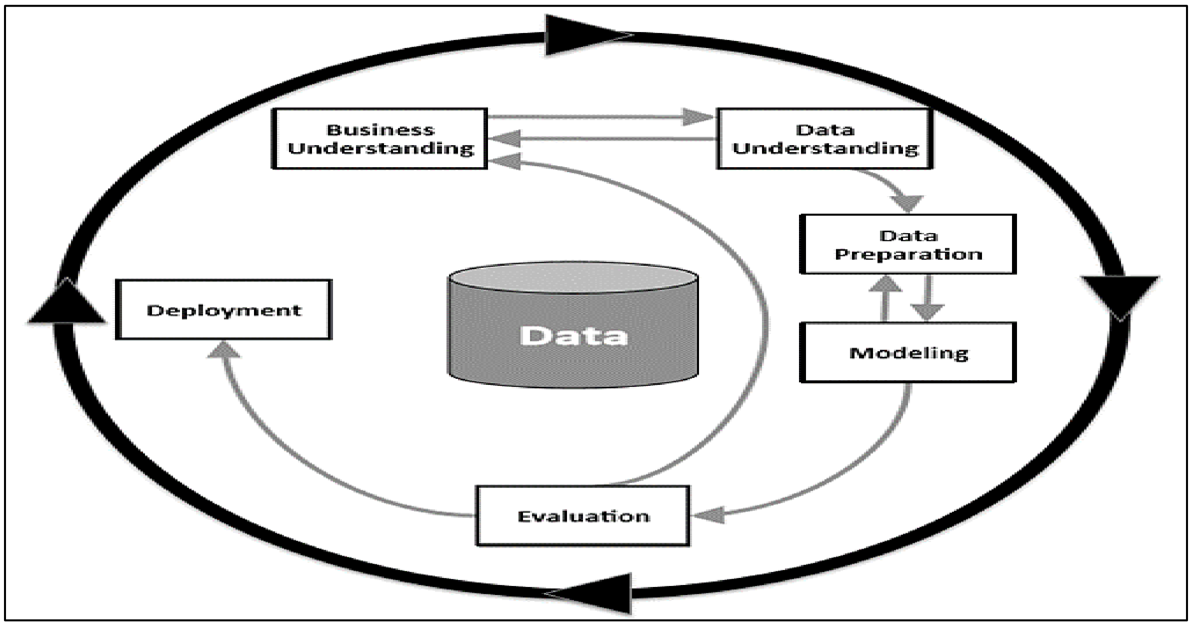


Figure 1: CRISP-DM: Standard Cross-industry Data Mining Process

3.1.1 Business understanding- This initial phase focused on defining project goals and requirements from an entrepreneurial viewpoint and then transformed this knowledge into a summary of data mining challenges and a practical project strategy to achieve the objective.

3.1.2 Data Understanding- This stage initiated with a primary set of data, followed by a process of familiarizing with facts, finding file quality deficiencies, discovering key information understandings, or classifying essential subgroups to hypothesis hidden data. It often Discovered the first insights into the data or found interesting subsets for hidden theories of knowledge. There was a clear connection between understanding the demand and understanding the results. For the formulation of the data mining problem and the project plan at least some understanding of the available data was needed. The process followed was categorized as below.

1) **Describe data** : Description of the data acquired including its format, the quantity (for instance , the amount of records and fields in every table), the field characteristics and all other surface features determined. Evaluation of the data to check if the data acquired meets the requirements.
2) **Explore data:** Described the effects of the data review, including first findings or initial hypothesis, and its impact on the rest of the project. Included even plots and graphs showing data Characteristics suggest further study of relevant subsets of data.
3) **Verify data quality**: Analysis of data characteristics that centered on questions such as:
    1) Was the data comprehensive (if it covered all the cases required)?
    2) Was it correct, or did it contain errors and how frequent were they when there were errors?
    3) Are the values missing in the data? If so, how have they been portrayed, where have they occurred and how popular were they?

4**) Data quality report:** Listed Data Quality Verification tests. If there were quality issues, then suggested possible solutions. Solutions provided to data quality concerns in this step were typically highly dependent on both data and business awareness.

3.1.3 Data preparation- Each process included tasks aimed at finalizing the data collection. Tables, documents, or attributes selected for data modelling were transformed and cleaned. This was further used to construct the final collection of data. This Process consisted of the following steps:

- **Selecting data:** With regards to business priorities , relevant data was picked. This also provided reasons for the data being approved or rejected. The data selection part included both attribute selection (columns) and record selection (rows) in a table.
- **Cleaning data:** This was achieved through implementing more aggressive techniques such as analysts picking clean subsets of the data. This included Choosing clean data subsets, incorporating correct defaults or
  more ambitious approaches such as modelling missing data calculation.
- **Construct Data:** This included creating new records for current attributes from derived attributes or transformed values.
- **Integrate Data:** This involved Combining information using aggregations to construct new records / values.
- **Format Data:** This stage Changed the data format / design. For e.g. elimination or trimming of unauthorized characters.

3.1.4 Data Modelling- Appropriate modelling techniques were chosen and applied to data, and parameters were tested and optimized to optimum values. Modelling steps included below:

- **Selecting modeling technique:** Different methods were chosen for modelling. During the business understanding process there might already be a chosen method, but at this point a particular modelling technique is selected e.g. Creating a decision-tree through C5.0 or creating a neural network using back propagation.. In case of several techniques , this function was carried out separately for each technique.
- **Generate Test Design:** The analyst checked the consistency of the models and their validity. Empirical tests would be conducted to verify model reliability. A process was created, or method was designed to check the consistency and validity of the model before actually constructing a model. For instance- in supervised data mining, tasks like

classification,, error rates were commonly used as measure of quality for data mining models.

Hence normally the dataset was divided into train and test sets and model was created on the train set and the model output was estimated on the divided test set.

• **Build the model:** Analyst ran the model on prepared data set and created a new model.

1) Parameter settings – There were also a large numbers of parameters which could be modified with any modelling method. In this step parameters and their selected values and the rationale for selecting parameter settings were described .
2) .Models – These were the authentic models produced by the modelling tool and not the modelling study.
3) Model descriptions – Here the resulting models were described, the model interpretation was observed, and any problems were recorded with their encountered definitions .

- **Assess the model:** Model 's viability with respect to potential data problem was evaluated by Analyst. The model was ranked at this level and evaluated according to the evaluation criteria.

Business objectives and the business success criteria was considered to its possible extent. A single technique was applied more than once in the projects, and results of models was generated using various different techniques.

3.1.5 Evaluation: It was important to evaluate the importance and competence of an existing model to understand how thoroughly and adequately it addressed cybersecurity issues prior to deployment. This stage involved the estimation by means of a test collection of various different analytical dealings and other outputs as well as tables and graphs.

This test set was completely independent of the training set but followed the corresponding distribution of probabilities and had common outcomes. During this step, the degree to which the model met the business goals was evaluated along with determining the model deficiency.

3.1.6 Deployment: The validation process was implemented in a production environment or likely in a similar test setting post the established model was accepted and certified that it adequately addressed the cybersecurity problem.

It was usually used in a partial and definitive way before all output variables were evaluated in full. Deployment sometimes was as simple as generating a generic report with suitable recommendations or was complicated like implanting the model in an explicit or complex workflow and documenting mechanism handled by a custom application.

3.2 MODELS:

3.2.1 Booster: XGBoost is a distributed gradient boosting library that is optimized to be highly efficient , flexible, and portable. Under the Gradient Boosting Paradigm, it applies machine learning algorithms. XGBoost provides a parallel boost to the tree (also known as GBDT, GBM) that quickly and accurately resolves many data science problems. In a machine learning ensemble, boosting is an algorithm that helps to reduce variance and bias. The algorithm benefits in the transformation of weak learners into strong learners by combining N number of learners.

- **Forms of Boosting:**

1. **Adaptive Boosting (Adaboost)**:

   Created by Freund and Schapire in 1995, the AdaBoost algorithm solved several of the functi onal difficulties of the earlier boosting algorithms. The algorithm takes as its input a training set (x1;y1); ..; (xm;ym) in which each xi belongs to a certain domain or instance space X eac h label yi is in a set of Y. (Yoav Freund, n.d.) .

   Adaboost algorithms are used popularly in regression and classification procedures. An error found in previous models will be modified with weighting until an exact predictor has been made. (Institut Corporate Finance, n.d.)

2. **Gradient Boosting**:

   As with every other ensemble machine learning operation, gradient boosting sequentially add s predictors to the ensemble and follows the sequence in correcting predictors to arrive at an effective predictor at the end of the procedure. Adaboost corrects its previous errors by adjust ing the weights in each iteration for any incorrect observation, but gradient boosting attempts to suit a new predictor in the residual errors committed by the preceding predictor. Gradient boosting uses gradient descent to classify previously encountered problems in the predictions of the learners.

3. **XGBoost (Extreme Gradient Boosting)** :

XGBoosting implements gradient-boosted decision trees, improved efficiency, and speed. Due to the model training that must follow a series, the implementation of gradient boosted machines is fairly slow. XGBoost relies on model efficiency and computational speed. It provides various advantages, such as parallelization, distributed computing, cache optimization and out-of-core computing

- **Advantages and Disadvantages of Boosting:**

As an ensemble model, boosting comes with an algorithm that is easy to read and understand, making interpretations of its predictions easy to manage. By using their clone methods, such as bagging or random forest, and decision trees, the predictive capability is eff icient. Boosting is a robust approach which can easily curb over-fitting.

One disadvantage of boosting is that it is sensitive to outliers, as each classifier is obliged to f ix the predecessor errors. While the system relies too much on outliers. Another downside is t hat the process is virtually difficult to scale up. This is because each estimator bases its accur acy on past predictors, making it difficult to streamline the process.

3.2.2 Decision Tree :

Decision Trees (DTs) are a non-parametric, supervised method of classification and regression learning. The aim is to build a model that predicts the value of a target variable by learning from the data features basic decision rules.(1.10. Decision Trees — scikit-learn 0.23.2 documentation, no date).

For example , decision trees learn from data about approximating a sine curve with a set of if-then-else decision rules. The larger the vine, the more complicated the rules governing the decision and the fitter the model.

DecisionTreeClassifier takes two arrays as its input: an array X, sparse or dense, of size [n samples, n features] holding the training samples, and an array Y of integer values, size [n samples] holding the training sample class labels.

Advantages and Disadvantages of Decision Tree:

- Some **advantages** of decision trees are:

1) Easy to understand and interpret. Trees can be visualized

2) Requires little preparation of the results. Other strategies also involve data normalisati on, generating dummy variables and eliminating blank values. Remember that this m odule does not support missing values, however.

3) The cost of using the tree (i.e., predicting data) in the number of data points used to train the tree is logarithmic.

4) Performs well even if its assumptions are somewhat in breach of the true model from which the data was generated. (1.10. Decision Trees — documentation scikit-learn 0.23.2, no date)

- The **disadvantages** of decision trees include:

1) Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
2) It is known that the problem of learning an optimal decision tree is NP-complete under several aspects of optimality and even for simple conceptions. Practic al decision-tree learning algorithms are also based on heuristic algorithms such as the greedy algo rithm, where optimal decisions are made locally at each node. These algorithms can n ot guarantee that the decision tree which is globally optimal would return. It can be m itigated by training several trees in a learner ensemble, where the features and sample s are sampled randomly with replacement.
3) There are concepts that are difficult to understand, as decision trees can not easily articulate them, such as difficulties with XOR, parity or multiplexer.
4) Learners from decision-making tree build biased trees when other groups dominate. Therefore, it is recommended that the dataset be balanced before integrating with the decision tree.

3.2.3 Random Forest :

In random forests (see RandomForestClassifier and RandomForestRegressor classes), each tree i n the ensemble is constructed from a sample drawn from the training set with replacement (i.e., a bootstrap sample).Moreover, when splitting each node when constructing a tree, the best split is found either from all input features or from a random subset of max features size.

These two sources of randomness have the effect of decreasing the forest estimator variance. Usu ally, individual decision trees show high variance and appear to overfit. The randomness introdu ced into the forests yields decision trees with somewhat decoupled predictive errors. Many mista kes may be eliminated, by taking an average of those predictions. By combining diverse trees, ra ndom forests achieve a reduced variance, sometimes at the cost of a slight increase in bia. In prac tise the reduction in variance is always important, resulting in a stronger overall model.(*3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.23.2 documentation*, no date)

- **Advantages** of Random Forests:
  Random forests have predictions of variable importance, i.e., neural nets. Also, they provide a superior method to work with missing data. Missing values are replaced by the variable that appears the most in each node. Random forests provide the highest accuracy amongst all available classification methods.
  With multiple variables running into thousands, the random forest technique can manage big data as well. This will align data sets automatically when a class is more infrequent in the data than other classes. Also the system handles variables efficiently, making it ideal for complicated tasks.(*Random Forest - Overview, Modeling Predictions, Advantages*, no date)

3.2.4 SVM: Support Vector machines:

Support vector machines (SVMs) are a group of supervised methods of learning used for classification , regression, and identification of outliers. The scikit-learn support vector machines support dense (numpy.ndarray and convertible to that by numpy.asarray) as well as sparse (any scipy.sparse) sample vectors as input. To use an SVM to make pre dictions for sparse data, however, it must have been suitable on these data. For optimal efficiency, using dtype = float64 with C-ordered numpy.ndarray (dense) or scipy.sparse.csr matrix (sparse).

- The **advantages** of support vector machines are:
  1) Efficient in high dimensional spaces.
  2) In cases where the number of measurements is larger than the number of samples, it is still efficient.
  3) Uses a subset of training points in the decision function (known as support vectors), a nd it is also effective in memory.
  4) Versatile: Various functions for the decision function may be defined in the Kernel. Common kernels are provided, but custom kernels can also be specified

- The **disadvantages** of support vector machines include:
  1) If the number of features is much greater than the number of samples, it is crucial to a void over-fitting when selecting Kernel functions and to regularise the term.
  2) SVMs do not have direct estimates of probability, these are determined using expensi ve five-fold cross-validation.(*1.4. Support Vector Machines — scikit-learn 0.23.2 documentation*, no date)

3.2.5 Logistic Regression:

Despite its name, logistic regression is a linear model for classification, rather than regression. In the literature logistic regression is also known as logistic regression, maximum entropy classification (MaxEnt), or log-linear classification. The probabilities representing the potential outcomes of a single trial are modelled using a logistic function in this model. In LogisticRegression functional regression is applied. This implementation will fit in with optional, or Elastic-Net regularization, binary, One-vs-Rest, or multinomial logistic regression..

- **Advantages :**

1) Logistic Regression is one of the simplest algorithms for machine learning, and is easy to implement, but in some cases provides great training performance. For these reasons as w ell, training a model with this algorithm doesn't require high computational power.
2) The parameters predicted (trained weights) give inference on the importance of each func tion. Only the direction of association is given, i.e. positive or negative. So, we can use lo gistic regression to figure out how the features are related.

- **Disadvantages:**

  1) Logistic Regression is a model of statistical analysis which attempts to predict accurate probabilistic results based on independent features. It can lead to the model being overfit on the training set on high- dimensional datasets, which means overestimating the accuracy of predictions on the training set and hence the model will not be able to predict accurate results on the test set. Typically, this occurs in the case where the model is trained with lots of features on little training data. Regularization techniques should also be considered on high dimensional datasets to prevent over-fitting (but this makes the model complex). Very high regularization factors can even lead to under-fitting of the model on the training data.
  2) For logistic regression, nonlinear problems cannot be solved because it has a linear decision surface. In real-world settings linearly separable data is rarely found. It involves the transformation of nonlinear features, which can be achieved by increasing the number of features so that the data is linearly separable in higher dimensions (Advantages and Disadvantages of Logistic Regression, 2020).

3.2.6 K Nearest Neighbors :

Sklearn.neighbors offers tools for learning approaches. Based on unsupervised and supervised neighbors. Uncontrolled nearest neighbors are the basis of many other methods of learning, especially multiple learning, and spectral clustering. Supervised neighbor-based learning comes in two flavors: classification with discrete labels for data, and continuous label regression for data. The idea behind nearest neighboring methods is to locate and predict the mark from these, a predefined number of training samples closest to the new point in space. The sample number may be a user-defined constant (k-nearest neighbor learning) or can differ based on local point density (radius-dependent neighbor learning).

- Some **Advantages** of KNN:
  1) Rapid calculation time
  2) Simple algorithm : to understand
  3) Versatile : suitable for regression and classification
  4) High accuracy : comparisio with better-supervised learning models is not needed
  5) No assumptions about data :
     No need to make additional assumptions, tune several parameters or create a model. Which makes it crucial in the case of nonlinear results.

- Some **Disadvantages** of KNN

1) Accuracy changes with the quality of the data
2) The prediction phase might be slow with large data
3) Sensitive to the scale of the data and inappropriate features
4) Require high memory for store all of the training data
5) It can be computationally expensive as it stores all of the training.(Chatterjee, 2020)

3.2.7 Artificial Neural Network:

The Artificial Neural Network (ANN) is the bit of a computer system designed to re-enact the way the human mind analyses and shapes data. It is the establishment of artificial intelligence (AI) and it takes care of problems / calculations that by human or observable standards may prove difficult or troublesome. ANNs have the ability to self-learn which empowers them to produce better results as more knowledge becomes available. Artificial neural networks are built like the human mind, with hubs of neurons interconnected like a network. An ANN has hundreds or thousands of fake neurons, called processing units, linked by nodes. These handling units consist of modules for input and output. The input units receive different forms and information structures based on an internal weighting system, and the neural system strives to find out about the data introduced to create one output report.

At first, an ANN enters a training process in which it learns to identify patterns in the data, whether visually, orally, or via text. The network contrasts the actual output generated with what it was intended to generate during this supervised stage – the desired output. The comparison between the two results is balanced using backpropagation. It means that the machine operates in reverse, moving from the output unit to the input units to alter the weight of its associations between the units until the difference between the real and wanted result provides the lowest possible error. Using binary numbered yes / no question forms, the ANN is instructed during the training and supervisory phase on what to look for and what its performance should be.

3.3 IMPLEMENTATION:

For this research, we used CRISP-DM methodology. As it contains 6 phase which are suitable for data science and data mining project. We also followed machine learning life cycle which helped us to create this research understandable. We implemented this file in python and as well as in RapidMiner. For python , Anaconda Jupiter notebook(6.0.1) was used for code implementation and for RapidMiner, RapidMiner Studio (9.6.0) was used.

BUSINESS UNDERSTANDING: This was one of the most essential phase of the data science or data mining project for initial understanding . As this phase involved various key steps like Determining the business objective, assessing the situation as well as plan or producing the project.

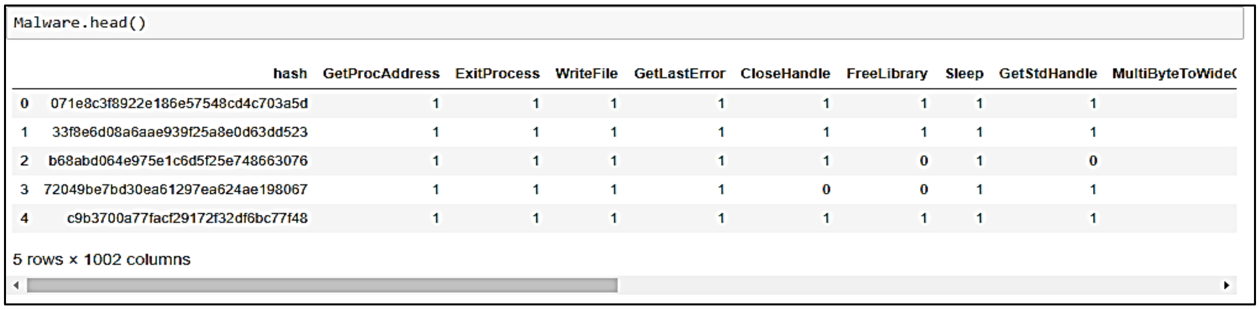
Business Objective:

- To ascertain the most accurate Machine learning algorithm technique with the utilization of the selected data set.

DATA UNDERSTANDING: This phase began with preliminary data collection to find out the earliest insights from the data, to uncover quality problems, or to identify the subsets to form hypothesis about hidden information.

For the research, we took dataset from “ieeedataport.org” which was actually used for malware detection using machine learning and deep learning concepts. The dataset contained 47580 observations and 1002 features. Dataset contained the static analysis data which was extracted cuckoo sandbox report. The “malware” file were downloaded from virushare.com and “good-ware” were downloaded from portableapps.com.

Used head() and describe() methods to find properties like the count of the rows and columns, name of the attributes , mean, median, maximum, and minimum value and standard deviation for



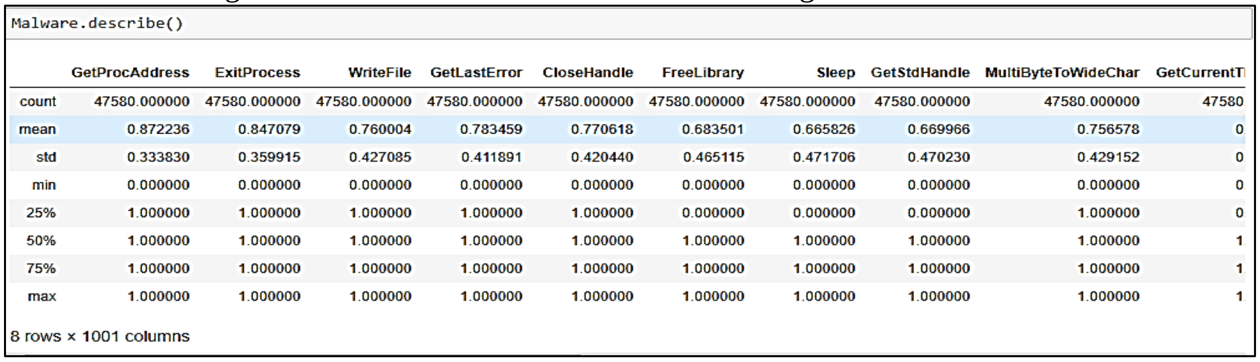
```
Malware.head()
```

	hash	GetProcAddress	ExitProcess	WriteFile	GetLastError	CloseHandle	FreeLibrary	Sleep	GetStdHandle	MultiByteToWide(
0	071e8c3f8922e186e57548cd4c703a5d	1	1	1	1	1	1	1	1	
1	33f8e6d08a6aae939f25a8e0d63dd523	1	1	1	1	1	1	1	1	
2	b68abd064e975e1c6d5f25e748663076	1	1	1	1	1	0	1	0	
3	72049be7bd30ea61297ea624ae198067	1	1	1	1	0	0	1	1	
4	c9b3700a77facf29172f32df6bc77f48	1	1	1	1	1	1	1	1	

5 rows × 1002 columns

each feature. With some initial correlation among the variables also performed exploratory data analysis (EDA) in this step.

Figure 2: Head function for Data understanding.



```
Malware.describe()
```

	GetProcAddress	ExitProcess	WriteFile	GetLastError	CloseHandle	FreeLibrary	Sleep	GetStdHandle	MultiByteToWideChar	GetCurrentTi
count	47580.000000	47580.000000	47580.000000	47580.000000	47580.000000	47580.000000	47580.000000	47580.000000	47580.000000	47580
mean	0.872236	0.847079	0.760004	0.783459	0.770618	0.683501	0.665826	0.669966	0.756578	0
std	0.333830	0.359915	0.427085	0.411891	0.420440	0.465115	0.471706	0.470230	0.429152	0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

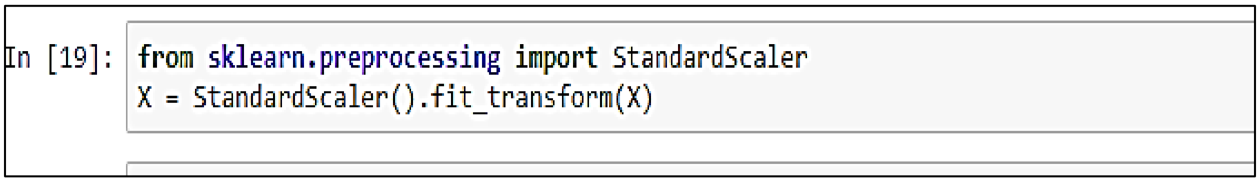
8 rows × 1001 columns

Figure 3: Describe function for Data understanding.

DATA PREPARATION:

This phase included all the processes needed to build the final data set or the data which was fetched into the modeling tools from the untouched or raw data. Steps which were done in research are as below:

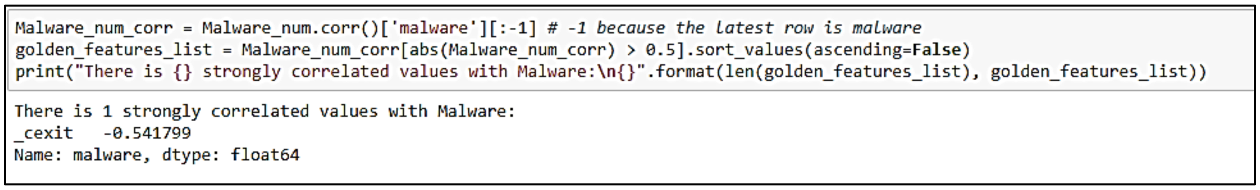
- Initial stage: Converted all the features into standard scaler and checked the null values.



```
In [19]: from sklearn.preprocessing import StandardScaler
         X = StandardScaler().fit_transform(X)
```

Figure 4: Standardizing the dataset .

- Second Stage: Checked the Correlation between the target variable and the independent variable. Found that “cexit” was strongly correlated the malware which was our target variable.

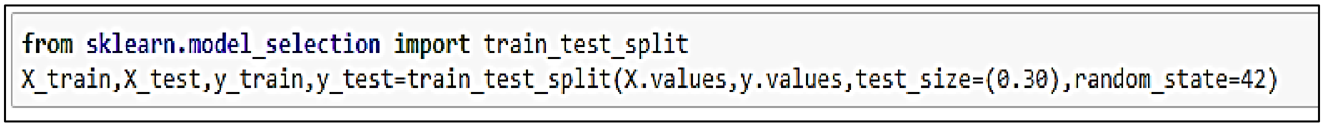


```
Malware_num_corr = Malware_num.corr()['malware'][:-1] # -1 because the latest row is malware
golden_features_list = Malware_num_corr[abs(Malware_num_corr) > 0.5].sort_values(ascending=False)
print("There is {} strongly correlated values with Malware:\n{}".format(len(golden_features_list), golden_features_list))
```

```
There is 1 strongly correlated values with Malware:
_cexit   -0.541799
Name: malware, dtype: float64
```

Figure 5: Finding Correlation between independent and dependent variables.

- Third Stage: Dataset contained 1002 features. Which was actually to many features and as per our business requirement we did not require some features. Hence we used dimension reduction technique principal component analysis (PCA).
- Fourth Stage : After feature selection using PCA , the data was split into train and test set by using Train_Test_split.



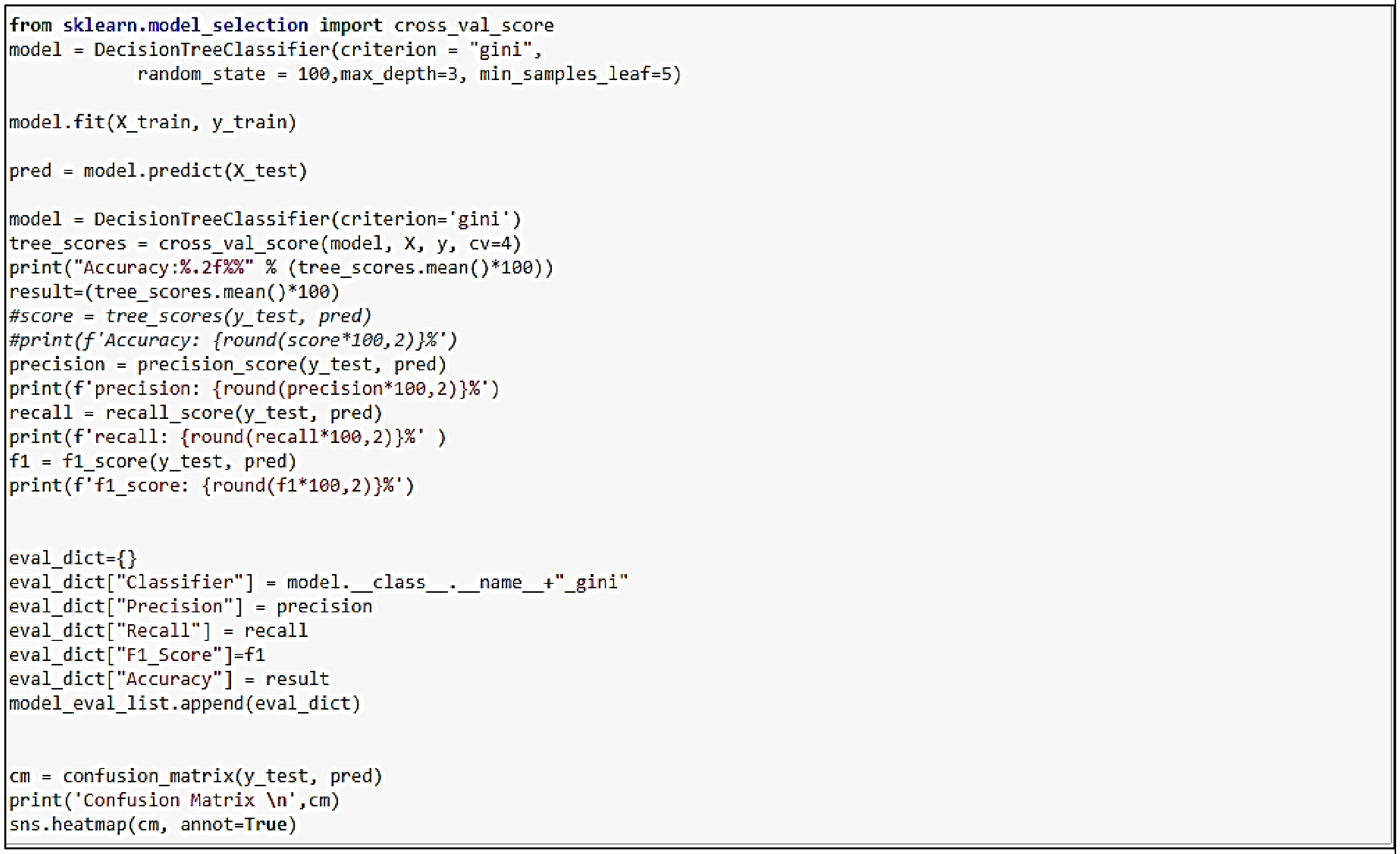
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X.values,y.values,test_size=(0.30),random_state=42)
```

Figure 6: Train_Test_Spilt

- Fifth Stage: As the target variable was unbalanced, used SMOTE function to balance the data.

DATA MODELING: Numerous modeling techniques were considered and implemented along with the calculation of their parameters in order generate optimal values during this phase.
In this phase, As the objective stated that we had to find the best suitable model for malware detection so, we could choose from multiple models like Decision Tree classifier, random forest classifier, KNN classifier, logistic regression, SVM etc. and find the accuracy, precision, f1-score and recall, and confusion matrix of each and every model.

EVALUATION: It has been crucial to thoroughly evaluate the model and review the construction of the model before going for final deployment. It was crucial to understand some important business issues are satisfied at this point of time.
In this phase as many models have been implemented and have been evaluated on the basis of accuracy. Also have calculated F1-score, precision, recall and confusion matrix. Also have done cross validation and hyper parameter tuning to enhance the performance of the models to find the best accuracy.



```
from sklearn.model_selection import cross_val_score
model = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)

model.fit(X_train, y_train)

pred = model.predict(X_test)

model = DecisionTreeClassifier(criterion='gini')
tree_scores = cross_val_score(model, X, y, cv=4)
print("Accuracy:%.2f%%" % (tree_scores.mean()*100))
result=(tree_scores.mean()*100)
#score = tree_scores(y_test, pred)
#print(f'Accuracy: {round(score*100,2)}%')
precision = precision_score(y_test, pred)
print(f'precision: {round(precision*100,2)}%')
recall = recall_score(y_test, pred)
print(f'recall: {round(recall*100,2)}%' )
f1 = f1_score(y_test, pred)
print(f'f1_score: {round(f1*100,2)}%')


eval_dict={}
eval_dict["Classifier"] = model.__class__.__name__+"_gini"
eval_dict["Precision"] = precision
eval_dict["Recall"] = recall
eval_dict["F1_Score"]=f1
eval_dict["Accuracy"] = result
model_eval_list.append(eval_dict)


cm = confusion_matrix(y_test, pred)
print('Confusion Matrix \n',cm)
sns.heatmap(cm, annot=True)
```

Figure 7: Training Decision tree Model and evaluating the performance.

DEPLOYMENT:
The knowledge gained in model creation process was organized and presented in usable format. The key steps here were as below:
1. plan deployment- evaluation of results and development of strategy for deployment.

2. plan monitoring and maintenance- for day to day business this step was important as it provided a reliable experience.

3. production of the final report - at the end of the project a final report was written for customers reference.

4. Review Project- Assessment of failure and success along with possible area of improvements for future references.

4. RESULT

The aim of this chapter was to find out the best suitable models for malware detection after comparing each model from machine learning as well as to check the accuracy and recall for the model. This was achieved by implementing and running the algorithms(Python) and RapidMiner on the dataset

After loading dataset, used dimensionality reduction technique, as the dataset contained 1002 features. Post that used principal component analysis in which selected n=3 which means an estimated number of components. Total variance explained by the n component where n is 3 is (0.56).

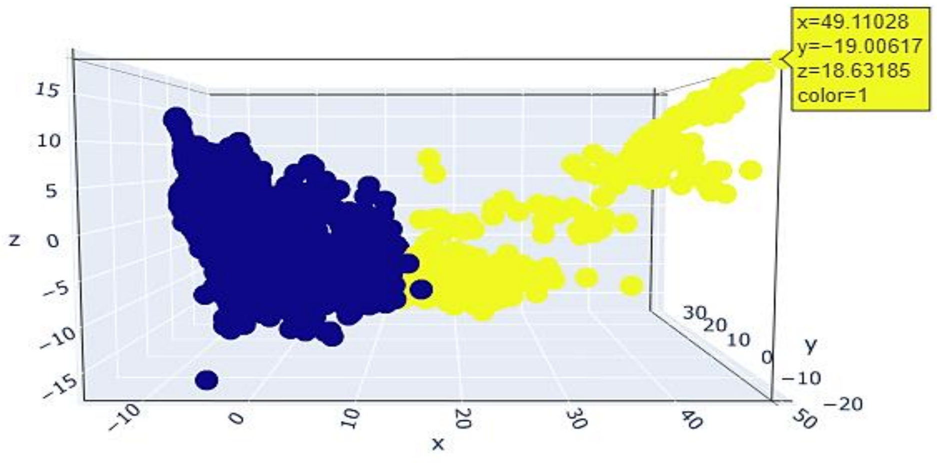


Figure 8 : Principal component Analysis:

After performing principal Component analysis, the dataset was split into train test split by using sklearn.model_selection 70:30 ratio. After splitting the dataset, it was observed that the target variable was not balanced. So, implemented SMOTE (Synthetic Minority Over-sampling Technique) in order to balance it.

After preparation stage the model was implemented, as we were finding the best machine learning model. So, Decision tree with criterion of Gini and entropy, gradient boosting, Random forest, logistics regression, K Nearest Neighbors, Support Vector machine, Ridge classifier, Ada boosting, and neural network was implemented.

After the models were implemented, the calculation of F1-score was performed, followed by precision, recall and accuracy calculation. Post that the cross validation and hyperparameter tuning was done to check the performance of the models.

- The cross Validation and Hyperparameter Tuning was performed using Grid Search.

Random forest hyperparameters included the number of decision trees in the forest and the number of features considered by each tree when splitting a node. Max_features was set on 'auto'.

Used Scikit-Learn's **GridSearchCV method**, defined a grid of hyperparameter ranges, and sample from the grid, performed K-Fold CV with each combination of values.

grid_param = {'n_estimators': [50, 100, 150, 200, 250, 300,350,400]}

GridSearchCV(estimator=rfc, param_grid=grid_param, scoring='accuracy ', cv=5)

The range provided for number of trees was from 50 to 400. 5-fold **Cross Validation** was implemented. So, for each n_estimator value in the range 50 to 400, the first iteration model was trained on the first four folds and evaluated on the fifth. The second time the model was trained on the first, second, third, and fifth fold and evaluated on the fourth. This procedure was repeated 3 more times, each time evaluating on a different fold. At the very end of training, the average the performance on each of the folds came up with final validation metrics for the model.

Cross validation was done to reduce overfitting. More number of folds meant less overfitting of the Random forest classifier. But since it was a repeated process, it was time consuming hence selecting **CV=5.**

Scoring in Grid Search:

- scoring = 'accuracy' when you want to maximize prediction accuracy.
- scoring = 'recall' when you want to minimize false negatives.
- scoring = 'precision' when you want to minimize false positives.
- scoring = 'f1' when you want to balance false positives and false negatives (place equal emphasis on minimizing both)

For this research , scoring for GridSearchCV was set as '**accuracy** '. Reason is explained below.

As mentioned above with reason, 5-fold cross validation was being implemented for the model to reduce overfitting problem.

For this model, defining terms were as below:

- What the model is doing? - based on variables, predicting whether a malware is present or not.

For this research, we created **Confusion matrix** : A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself was relatively simple to understand, below states the related terminology which are used to find result.

True Positive (TP): The model is predicting that a certain malware is detected and in actual the malware is detected .

True Negative (TN): The model is predicting that a certain malware has not detected and in actual the malware has not detected.

False Positive (FP): The model is predicting that a malware has detected but in actual data malware has not detected.

False Negative (FN): The model is predicting that a malware has not detected but in actual malware has detected.

Accuracy = **True Positive + True Negative/ [TP+TN+FP+FN**] , concentrated more on the TP and TN.

The focus for the model should be to check the number of true positives as the research ’s objective was to find the best model for static malware detection., the model should have all the right factors to predict the malware correctly.

Hence, selecting scoring = Accuracy in the GridSearchCV.

Accuracy : Accuracy is how close you are to the true value. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

The accuracy can be defined as the percentage of correctly classified instances

Accuracy = (TP + TN)/(TP + TN + FP + FN)

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

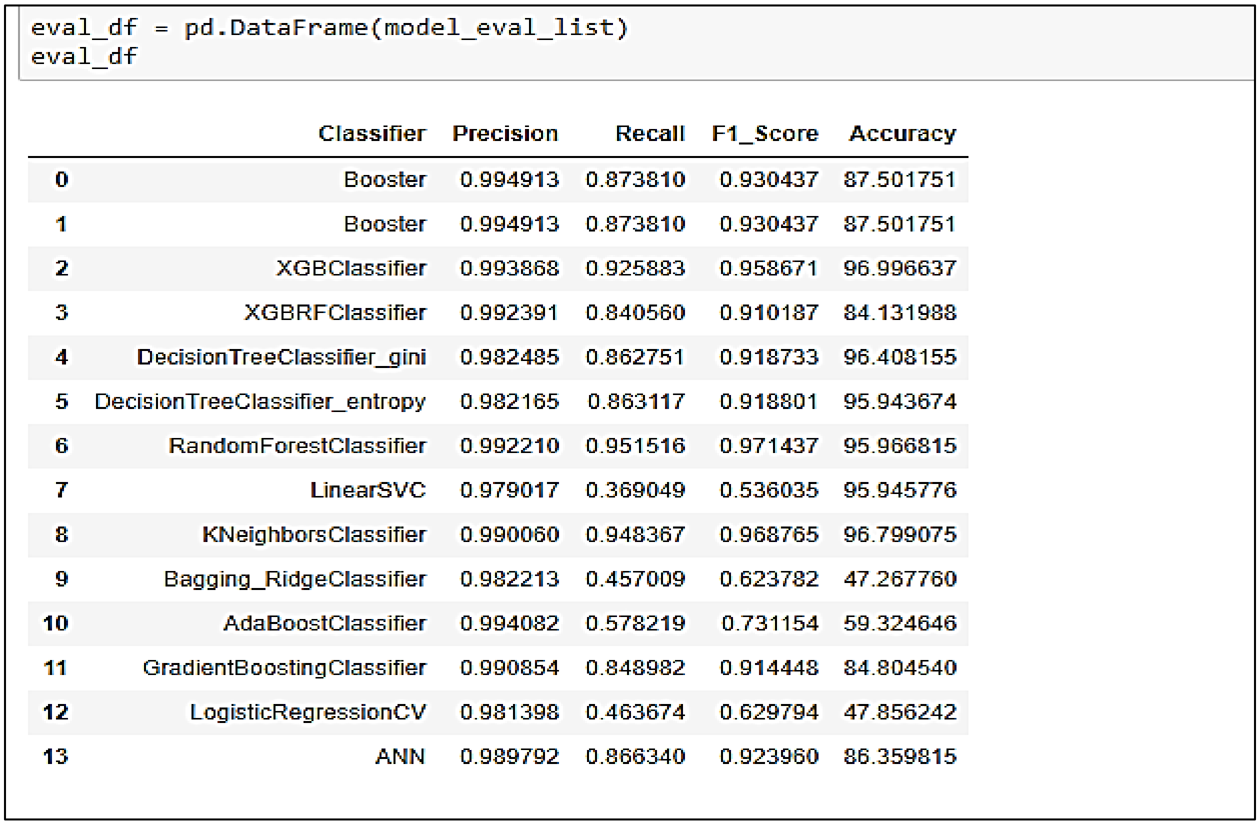
Precision = TP/TP+FP

F1 score : F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Recall : The recall is intuitively the ability of the classifier to find all the positive samples.

Recall=TP/TP+FN.



```
eval_df = pd.DataFrame(model_eval_list)
eval_df
```

	Classifier	Precision	Recall	F1_Score	Accuracy
0	Booster	0.994913	0.873810	0.930437	87.501751
1	Booster	0.994913	0.873810	0.930437	87.501751
2	XGBClassifier	0.993868	0.925883	0.958671	96.996637
3	XGBRFClassifier	0.992391	0.840560	0.910187	84.131988
4	DecisionTreeClassifier_gini	0.982485	0.862751	0.918733	96.408155
5	DecisionTreeClassifier_entropy	0.982165	0.863117	0.918801	95.943674
6	RandomForestClassifier	0.992210	0.951516	0.971437	95.966815
7	LinearSVC	0.979017	0.369049	0.536035	95.945776
8	KNeighborsClassifier	0.990060	0.948367	0.968765	96.799075
9	Bagging_RidgeClassifier	0.982213	0.457009	0.623782	47.267760
10	AdaBoostClassifier	0.994082	0.578219	0.731154	59.324646
11	GradientBoostingClassifier	0.990854	0.848982	0.914448	84.804540
12	LogisticRegressionCV	0.981398	0.463674	0.629794	47.856242
13	ANN	0.989792	0.866340	0.923960	86.359815

Figure 9: Statistical Analysis of Machine learning Model.

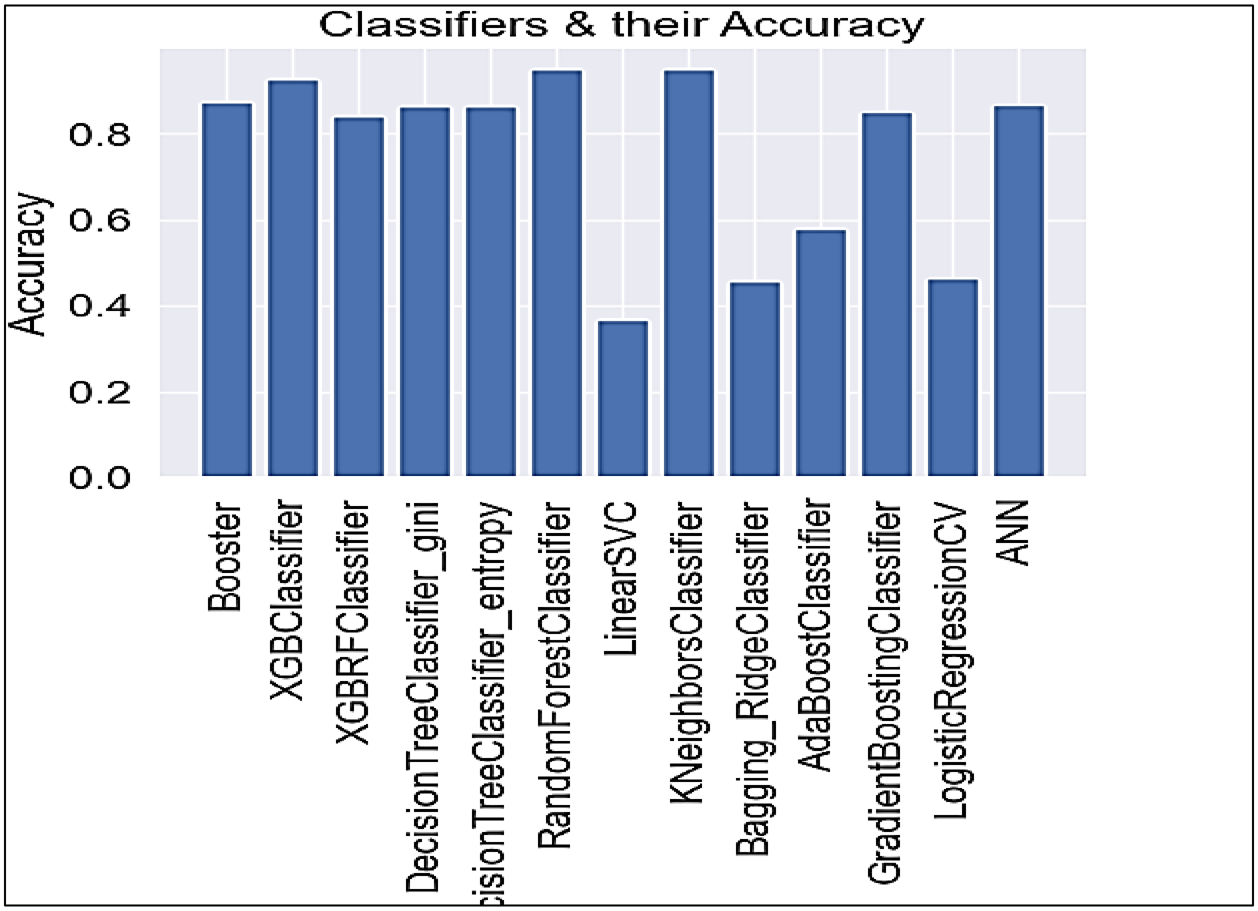


figure 10:Accuracy comparison of machine learning models.

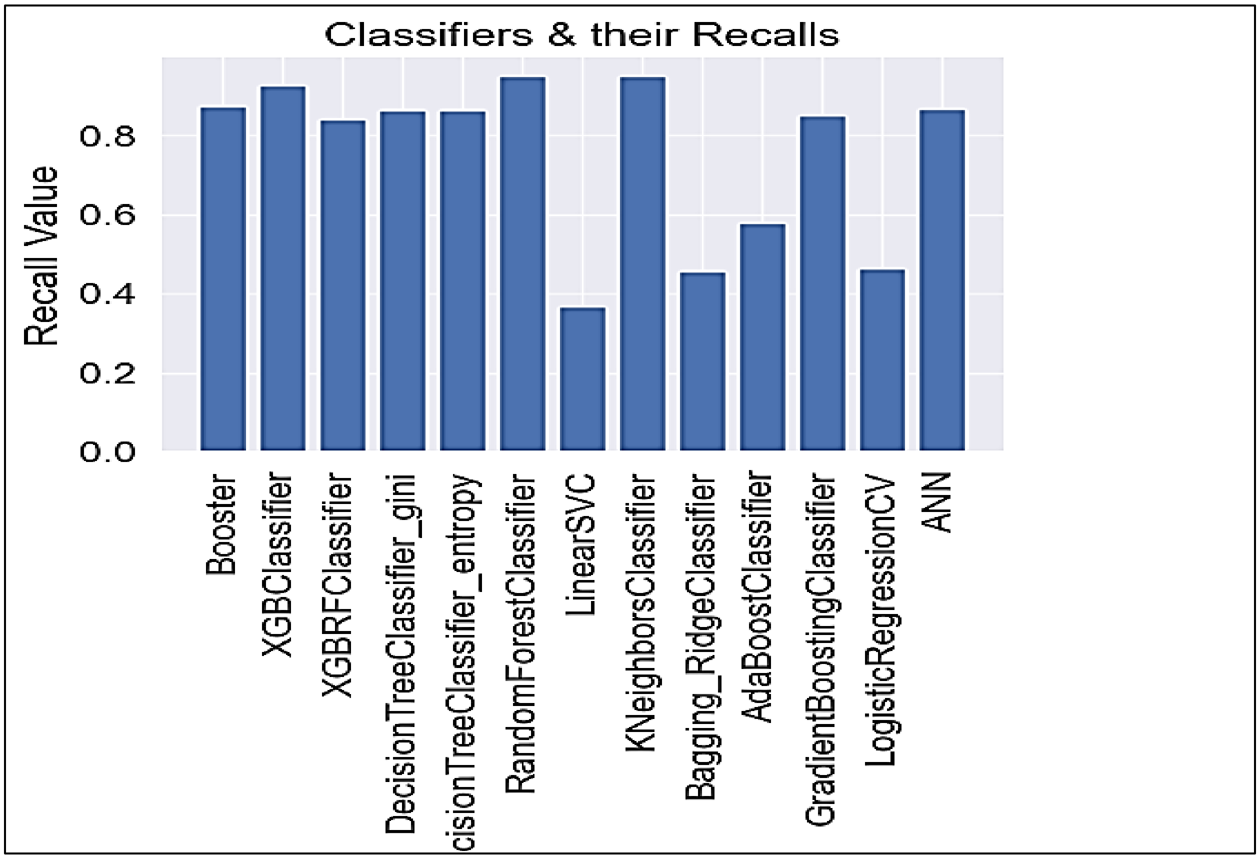


Figure 11: Recall comparison of machine learning models.

For this research, we also used RapidMiner Application. RapidMiner, A data science software platform developed by the company of the same name that provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics. It is used for business and commercial applications as well as for research, education, training, rapid prototyping, and application development and supports all steps of the machine learning process including data preparation, results visualization, model validation and optimization.

The chosen method was auto-model. Auto Model was an extension to RapidMiner Studio that accelerates the process of building and validating models.

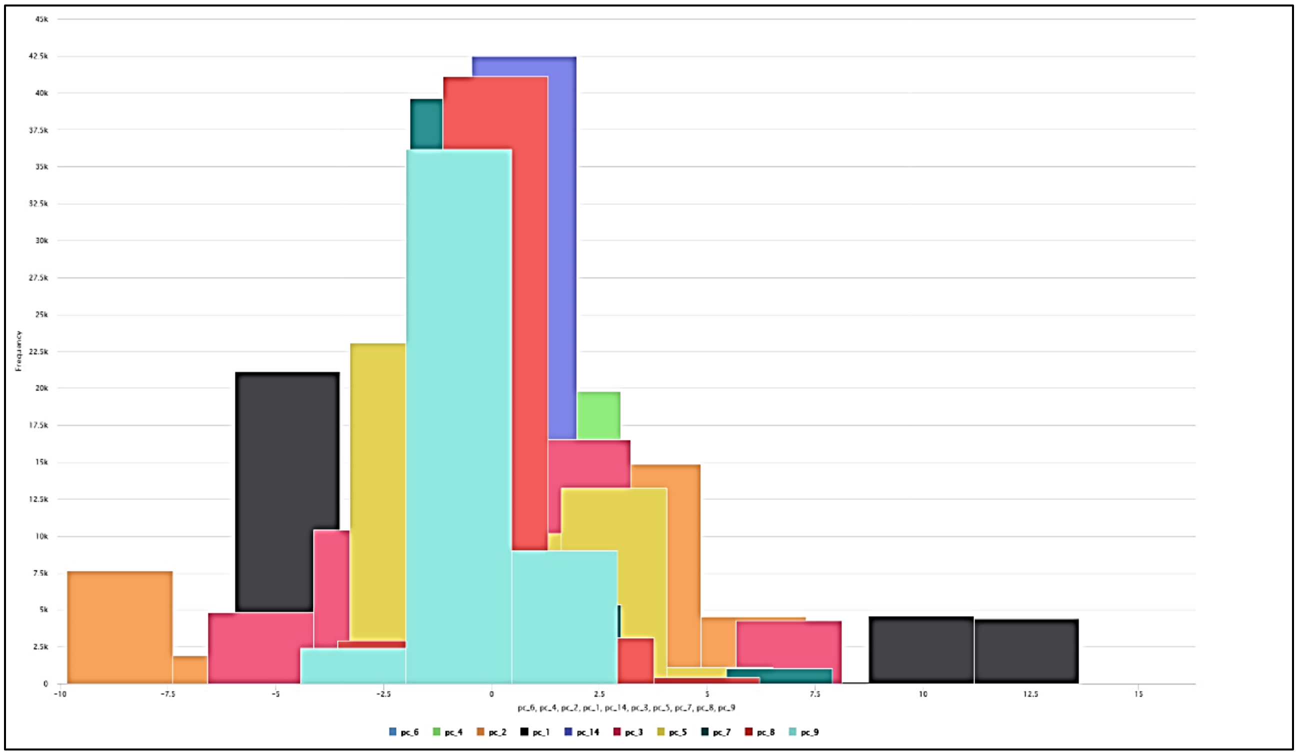
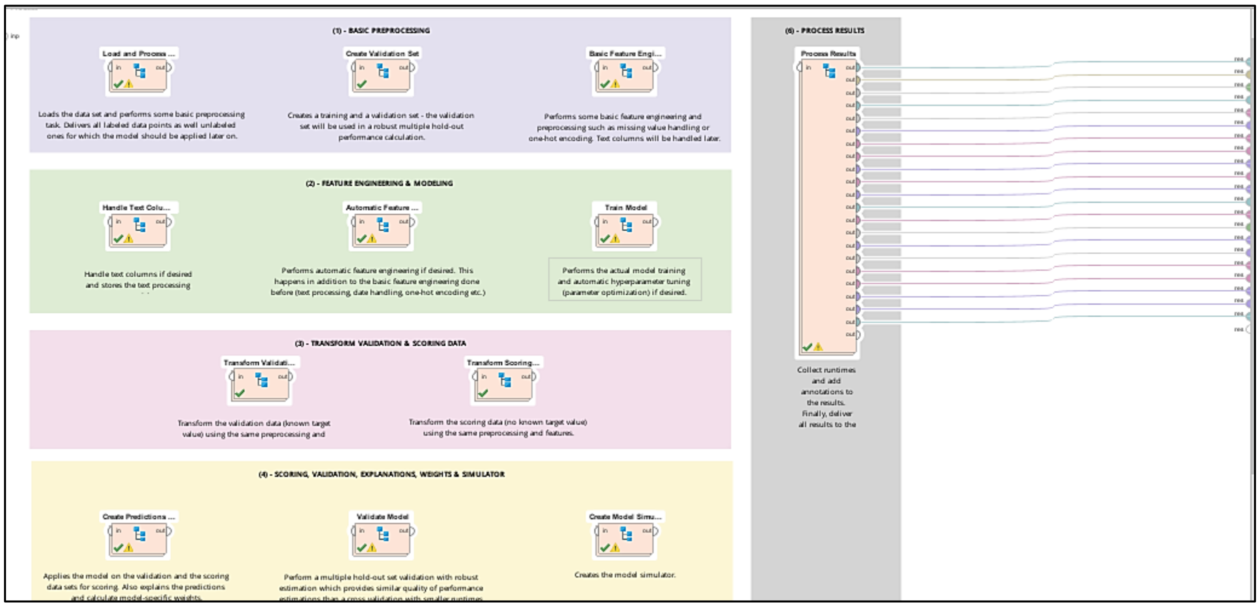


Figure 12: Principle component analysis (n=10) using RapidMiner

- ➢ The implemented steps followed In RapidMiner were the same as implementation steps in Python.



(1) - BASIC PREPROCESSING

Loads the data set and performs some basic preprocessing task. Delivers all labeled data points as well unlabeled ones for which the model should be applied later on.

Creates a training and a validation set - the validation set will be used in a robust multiple hold-out performance calculation.

Performs some basic feature engineering and preprocessing such as missing value handling or one-hot encoding. Text columns will be handled later.

(2) - FEATURE ENGINEERING & MODELING

Handle text columns if desired and stores the text processing

Performs automatic feature engineering if desired. This happens in addition to the basic feature engineering done before (text processing, date handling, one-hot encoding etc.)

Performs the actual model training and automatic hyperparameter tuning (parameter optimization) if desired.

(3) - TRANSFORM VALIDATION & SCORING DATA

Transform the validation data (known target value) using the same preprocessing and

Transform the scoring data (no known target value) using the same preprocessing and features.

(4) - SCORING, VALIDATION, EXPLANATIONS, WEIGHTS & SIMULATOR

Applies the model on the validation and the scoring data sets for scoring. Also explains the predictions and calculate model-specific weights.

Perform a multiple hold-out set validation with robust estimation which provides similar quality of performance estimations than a cross validation with smaller runtimes

Creates the model simulator.

(6) - PROCESS RESULTS

Collect runtimes and add annotations to the results. Finally, deliver all results to the

Figure 13 : Process in RapidMiner.

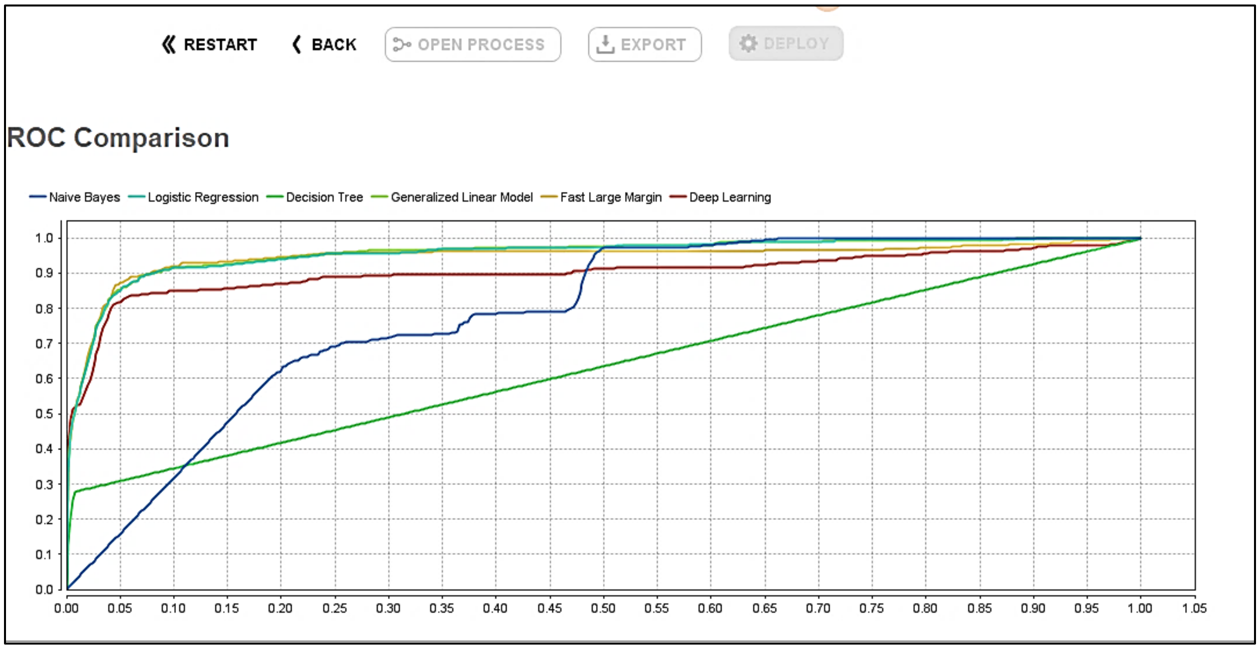
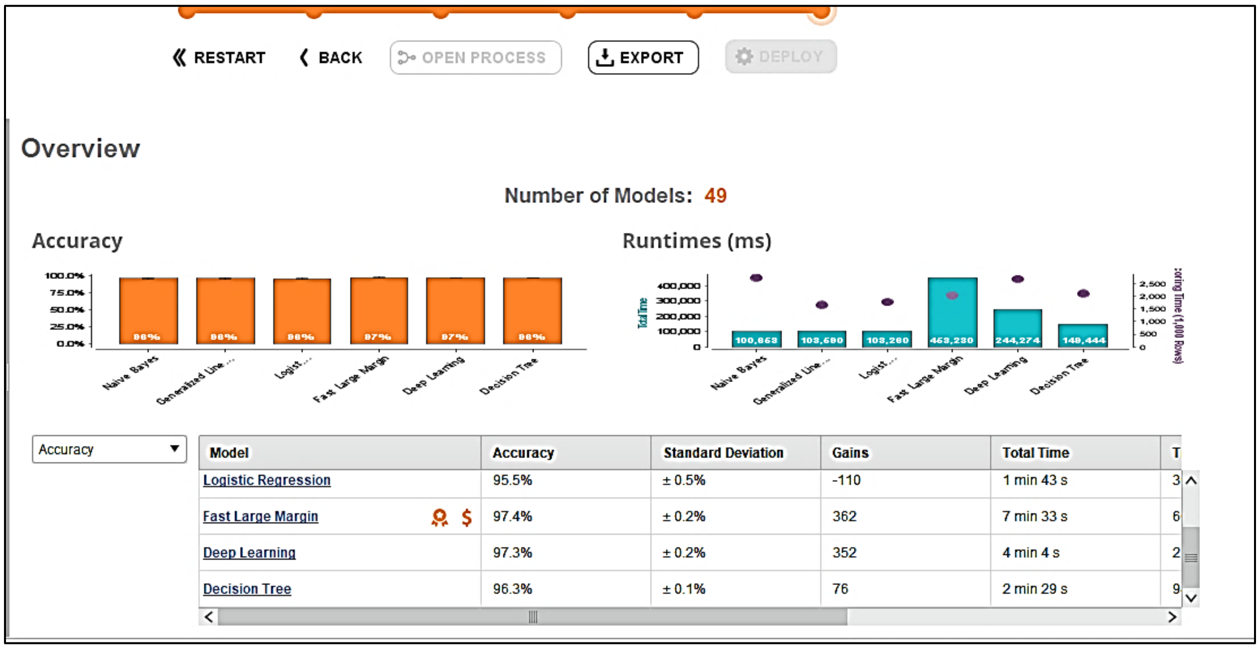


Figure 14: ROC Generated in RapidMiner



Overview

Number of Models: 49

Model	Accuracy	Standard Deviation	Gains	Total Time	T
Logistic Regression	95.5%	± 0.5%	-110	1 min 43 s	3
Fast Large Margin	97.4%	± 0.2%	362	7 min 33 s	6
Deep Learning	97.3%	± 0.2%	352	4 min 4 s	2
Decision Tree	96.3%	± 0.1%	76	2 min 29 s	9

Figure 15: Overview summary of models in RapidMiner.

To conclude, Python and RapidMiner both gave the best models with 97% of accuracy. It can also be stated that **decision tree**, **neural network** and **random forest, K-Nearest Neighbor** are models which give better performance for malware detection.

5. CONCLUSION

Conclusion:

To sum the research, K Nearest Neighbor, Decision tree, neural networks and random forest were the most suitable model for malware detection. The results were checked on the basis of accuracy and recall. The statistical analysis on every model was also done and checked. The previous research and theories also state that these models are suitable for the malware detection, but they were stating on the basis of the accuracy of the models, but our finding is on the basis of the accuracy and recall. The analysis stated that CRISP-DM methodology was best suitable for the research. Tools such as anaconda Jupiter Notebooks and RapidMiner were used to execute or find the result.

5.1 Discussion:

In this research, as our **objective** was to find the best suitable model for malware detection, at the initial stage it was difficult to find the best dataset and a suitable methodology which would fulfill the business requirements for research . In previous research's, many researchers have detected the malware using machine learning models and deep learning models on the basis of the accuracy. Multiple researchers have also stated that the models like decision tree and random forest model are best suitable for finding malware.

In this research, the **decision tree**, **random forest,** and **neural networks** were the models found best suitable for malware detection. Machine learning life cycle was followed and each and every step were implemented for evaluation. Model was evaluated on the basis of accuracy and recall which evaluated the true positive. Also calculated the precision and f1-score which essentially helped to find out model performance. Also, our **Strength** is we performed cross-validation and hyperparameter tuning to enhance the performance of the models. A potential problem during the in-data preparation phase was because the dataset contained 1002 features, hence it was pretty difficult to find the best suitable features, as well as the independent variable, was unbalanced. Therefore, It was not able to give the best result. As a **strength**, we have followed the machine learning cycle life cycle and we used CRISP-DM as methodology, which helped to cover each and every aspects to find the best result. There might be a **drawback**, as the dimensional reduction technique was implemented because of which some features which are important might have missed out. Future research stemming helped to perform a predictive analytics task using the results. To conclude, for this research every aspect to find out the best suitable model for malware detection was covered. The result has been examined in python as well as in RapidMiner to find out best solution for predictive analytics task.

5.2 Future works:

The malware detection arms race involves constant change, malware changes to evade detection and labels change as detection mechanisms react. Recognizing that malware changes over time, prior work has enforced temporally consistent samples by requiring that training binaries predate evaluation binaries.

Predictive analytics can be used for malware detection as research. Models can be trained on the basis of training dataset, evaluated for malware analysis.

The nature of the malware can also be analyzed by using the models. By using this, it will be helpful to create an application for static malware scanner. As these models are the traditional machine learning models so it will be useful. Like Anti-virus, which is also known as anti-malware applications, which is designed to detect, prevent, and take action to disarm or remove malicious software from the computer such as viruses, worms, and Trojan horses. It may also prevent or remove unwanted spyware and adware in addition to other types of malicious programs. As this type of application using the traditional machine algorithm for malware detection.

6. REFERENCE

1.4. Support Vector Machines — scikit-learn 0.23.2 documentation (no date). Available at: https://scikit-learn.org/stable/modules/svm.html#svm-classification (Accessed: 21 August 2020).

1.6. Nearest Neighbors — scikit-learn 0.23.2 documentation (no date). Available at: https://scikit-learn.org/stable/modules/neighbors.html#classification (Accessed: 21 August 2020).

1.10. Decision Trees — scikit-learn 0.23.2 documentation (no date). Available at: https://scikit-learn.org/stable/modules/tree.html (Accessed: 21 August 2020).

1.11. Ensemble methods — scikit-learn 0.23.2 documentation (no date). Available at: https://scikit-learn.org/stable/modules/ensemble.html#forest (Accessed: 21 August 2020).

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.23.2 documentation (no date). Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed: 21 August 2020).

'2019_IC3Report.pdf' (no date). Available at: https://pdf.ic3.gov/2019_IC3Report.pdf (Accessed: 12 August 2020).

Advantages and Disadvantages of Logistic Regression (2020) *OpenGenus IQ: Learn Computer Science*. Available at: https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/ (Accessed: 21 August 2020).

Boosting - Overview, Forms, Pros and Cons, Option Trees (no date) *Corporate Finance Institute*. Available at: https://corporatefinanceinstitute.com/resources/knowledge/other/boosting/ (Accessed: 20 August 2020).

Chatterjee, M. (2020) *The Introduction of KNN Algorithm | What is KNN Algorithm?*, *GreatLearning*. Available at: https://www.mygreatlearning.com/blog/knn-algorithm-introduction/ (Accessed: 21 August 2020).

'Crisp DM methodology' (no date) *Smart Vision Europe*. Available at: https://sv-europe.com/crisp-dm-methodology/ (Accessed: 19 August 2020).

Definition of HYPOTHESIS (no date). Available at: https://www.merriam-webster.com/dictionary/hypothesis (Accessed: 18 August 2020).

Fei Xiao and Zhaowen Lin (2019) *(PDF) Malware Detection Based on Deep Learning of Behavior Graphs*. Available at: https://www.researchgate.net/publication/331027941_Malware_Detection_Based_on_Deep_Learning_of_Behavior_Graphs (Accessed: 24 August 2020).

Freund, Y. and Schapire, R. E. (no date) 'A Short Introduction to Boosting', p. 14.

Gandotra, E., Bansal, D. and Sofat, S. (2014) 'Malware Analysis and Classification: A Survey', *Journal of Information Security*. Scientific Research Publishing, 2014. doi: 10.4236/jis.2014.52006.

Gandotra, E., Bansal, D. and Sofat, S. (2015) 'Computational Techniques for Predicting Cyber Threats', in Jain, L. C., Patnaik, S., and Ichalkaranje, N. (eds) *Intelligent Computing, Communication and Devices*. New Delhi: Springer India (Advances in Intelligent Systems and Computing), pp. 247–253. doi: 10.1007/978-81-322-2012-1_26.

Greitzer, F. L. and Frincke, D. A. (2010) 'Combining Traditional Cyber Security Audit Data with Psychosocial Data: Towards Predictive Modeling for Insider Threat Mitigation', in Probst, C. W. et al. (eds) *Insider Threats in Cyber Security*. Boston, MA: Springer US (Advances in Information Security), pp. 85–113. doi: 10.1007/978-1-4419-7133-3_5.

Halton, C. (no date) *Predictive Analytics Definition*, *Investopedia*. Available at: https://www.investopedia.com/terms/p/predictive-analytics.asp (Accessed: 12 August 2020).

Hypothesis | Definition of Hypothesis by Merriam-Webster (no date). Available at: https://www.merriam-webster.com/dictionary/hypothesis (Accessed: 18 August 2020).

Mahmood, T. and Afzal, U. (2013) 'Security Analytics: Big Data Analytics for cybersecurity: A review of trends, techniques and tools', in *2013 2nd National Conference on Information Assurance (NCIA)*. *2013 2nd National Conference on Information Assurance (NCIA)*, pp. 129–134. doi: 10.1109/NCIA.2013.6725337.

Newm, D. *et al.* (2016) 'Predictive Analytics and the Future of Cybersecurity', *Converge*, 26 October. Available at: https://www.convergetechmedia.com/predictive-analytics-future-cybersecurity/ (Accessed: 15 April 2020).

Oliveira, A. (2019) 'Malware Analysis Datasets: Top-1000 PE Imports'. IEEE. Available at: https://ieee-dataport.org/open-access/malware-analysis-datasets-top-1000-pe-imports (Accessed: 18 August 2020).

(PDF) Data Science Methodology for Cybersecurity Projects (no date) *ResearchGate*. Available at: https://www.researchgate.net/publication/323722668_Data_Science_Methodology_for_Cybersecurity_Projects (Accessed: 4 May 2020).

Probst, C. W. *et al.* (eds) (2010) *Insider Threats in Cyber Security*. Boston, MA: Springer US (Advances in Information Security). doi: 10.1007/978-1-4419-7133-3.

Random Forest - Overview, Modeling Predictions, Advantages (no date) *Corporate Finance Institute*. Available at: https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/ (Accessed: 21 August 2020).

Shmueli, G. and Koppius, O. R. (2011) 'Predictive Analytics in Information Systems Research', *MIS Quarterly*. Management Information Systems Research Center, University of Minnesota, 35(3), pp. 553–572. doi: 10.2307/23042796.

Wirth, R. and Hipp, J. (no date) ‘CRISP-DM: Towards a Standard Process Model for Data Mining’, p. 11.

XGBoost Documentation — xgboost 1.2.0-SNAPSHOT documentation (no date). Available at: https://xgboost.readthedocs.io/en/latest/ (Accessed: 20 August 2020).

Xu, M. *et al*. (2018) ‘Modeling and Predicting Cyber Hacking Breaches’, *IEEE Transactions on Information Forensics and Security*. *IEEE Transactions on Information Forensics and Security*, 13(11), pp. 2856–2871. doi: 10.1109/TIFS.2018.2834227.

Yun, J. B. *et al*. (2012) ‘System and method for predicting cyber threat’. Available at: https://patents.google.com/patent/US8191149B2/en (Accessed: 15 April 2020).