Web Based Malicious URL Detection Through Feature Selection (Special Characters) with Machine Learning

ANWAR RAZLAN BIN RASALI

Bachelor of Computer Science (Computer Systems and Networking)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : Anwar Razlan bin Rasali

Date of Birth

Title : Web Based Malicious URL Detection Through Feature Selection (Special Characters) with Machine Learning

Academic Session : Semester 2 2022/2023

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997) *
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done) *
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The library has the right to make copies of the thesis for academic exchange.

Certified by:

_____
(Student's Signature)

_____
New IC/Passport Number
Date: 12 July 2023

_____
(Supervisor's Signature)

Wan Nurulsafawati binti Wan Manan

_____
Name of Supervisor
Date:

NOTE: * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Computer Systems and Networking)

_____

(Supervisor's Signature)

Full Name : WAN NURULSAFAWATI BINTI WAN MANAN
LECTURER
Position : FACULTY OF COMPUTING
COLLEGE OF COMPUTING & APPLIED SCIENCE
Date : UNIVERSITI MALAYSIA PAHANG
26600 PEKAN, PAHANG DARUL MAKMUR
TE:09-4244729 FAX:09-4244666

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : ANWAR RAZLAN BIN RASALI

ID Number : CA20014

Date : 12 July 2023

Web Based Malicious URL Detection Through Feature Selection (Special Characters) with Machine Learning

ANWAR RAZLAN BIN RASALI

Thesis submitted in fulfilment of the requirements

for the award of the degree of

Bachelor of Computer Science (Computer Systems and Networking)

ACKNOWLEDGEMENTS

First and foremost, I would express great acknowledgement and appreciation to the Board of Directors for the Faculty of Computing, UMP, for giving me the opportunity to study here, let alone giving the green light to take part in the final year project. It is with great honor I give to all who have made this possible in the first place.

Furthermore, I would like to express my utmost thankfulness to Dr. Wan Nurulsafawati binti Wan Manan for taking me in as one of her students under her supervision for this project. She has enlightened me in many ways of the wonders of Artificial Intelligence and Machine Learning.

I would also express my greatest respect for my family members and friends who gave me the emotional support I need to keep on going with my studying life. Their likes will never be replaced.

Finally, I would like to thank anyone who has indirectly helped me in this journey whether they be a food stall vendor or technician, etc. Their duty is always respected and appreciated.

ABSTRAK

Dengan kemajuan teknologi, kita juga dapat lihat kemajuan dari segi serangan perisian '*malicious*' atau dikenali sebagai *malware.* Setiap individu berhak untuk mendapatkan perlindungan daripada ancaman serangan *malware* tersebut. Kajian ini berazam untuk mengaplikasikan kajian mengenai algoritma-algoritma *Machine Learning*. Kesemua ini dapat dijalankan melalui pengekstrakan ciri daripada URL tersebut. Program pengekstrakan ciri telah diaturcara dalam Python. Ciri-ciri URL tersebut akan dijalanakan berdasarkan spesifikasi tertentu dan sebuah keputusan akan dikeluarkan untuk mengesahkan sama ada URL tersebuat adalah *malicious* ataupun *benign.* Semoga kajian ini dapat menjadi insentif kepada semua orang agar dilindungi daripada ancaman serangan *malware*.

ABSTRACT

With the advancement of technology, we see a rise in the advancement of malicious software or malware attacks as well. People should have every right to never fall victim to any of these attacks. This research aims to implement a study of several Machine Learning Algorithms into a web-based malware. This is made possible by feature extraction through a series of codes written in Python. These features will undergo a few specifications before being determined as malicious or benign. The feature extraction process would test output a value in accordance with the URL and determine whether the URL is malicious or otherwise. This research would hopefully be a line of defense to prevent users from coming across to a malware with the help of the proposed project.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
SVM	Support Vector Machine
RF	Random Forest
DT	Decision Tree
GBT	Gradient Boosting Tree
NB	Naïve Bayes
BPA	Business Process Automation
AI	Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In the past decade, we have seen a surge in technology development, mainly in the IT segment. Gone are the days of analogue connection to access the internet, now we see how fibre optic becomes more mainstream, next to wireless connection of course. But with the development of advanced technology, there comes an advanced threat.

In recent years, hundreds of thousands to millions of everyday internet users come across malwares one form or another. The effects range from minor to severe. People who are most affected are those who are oblivious to malware threats and elderly people. These are usually the main targets for being victims of a malware attack, spam, scams, and all that fall under the same umbrella.

How people come to overcome this issue is by applying machine learning, an Artificial Intelligence (AI) designed to scan the pattern of things and predict the next course of action. Machine learning in malware detection comes with great benefits especially when it replaces the conventional methods of malware prevention and would generate algorithms to further enhances the defence against future malware attacks.

This is why this project has been proposed, for the sole purpose of generating a web-based program for malware detection, capable of warning and blocking certain sites that are ridden with suspicious links or scripts.

1.2 PROBLEM STATEMENT

The main purpose of this project is to test the best accuracy of malicious URL detection detection with machine learning. We need to take heavy concern on how detrimental the issue that arise with malware attacks. The first step to avoid such attacks is awareness and there seems to be a lack of it especially among the older generations. This would explain as to why they are in the majority of being the victims of such attacks. Other than awareness, there comes into question the preventive steps taken for avoiding such an attack. More people need to be well prepared and have at least a sort of software that could safeguard their precious data stored into their PCs. Many would leave out their computers without any sort of protection leaving it wide open and vulnerable for the attackers to take advantage of. Then when it is all too late, that is when they would begin to find solutions but unfortunately, it is too late. This is where this project comes into the picture as a crucial method to battle against the main issue of malware attacks.

As of the 2018 consensus alone, it was documented to at least have claimed victims at an estimated of 812.67 million total cases.(Jason Firch, 2022) In 2021, the numbers would skyrocket to over 600% due to the COVID-19 pandemic and a larger number of users on the internet. This would of course result to more cases as not many are used to the online work environment which puts them in an already vulnerable state. out of all the attacks, 92% of malware attacks happen through email delivery. Mobile malware also comes to place as being over 54% of the attacks. 99.9% of the mobile malware attacks are due to third-party app store hosts in which 98% of these take place on android devices. About as much as 250k users had the misfortune to receive trojan attacks. Out of all these malwares, trojans make up 51.45% of all malwares. The main victims come from the financial institution and about 34% of businesses was hit with malware attacks before they could regain their data after a week. (Jason Firch, 2022)

As we can see, malware attacks are more common that we thought. Therefore, it is most crucial to have a plan of prevention. Take the proverb ‘prevention is better than cure’ into consideration as it could save us from any harm to manifest and make things worse and worse overtime. Many do not see the threat as major and would regard them as minor inconvenience to the point where they could not even tell them apart from each other and think all malware is the same. Many would also not update their software up to date which could leave themselves in an exposed state due to there being unfixed bugs

and such. Much could be taken into consideration, but the main idea is that a preventive method could make a difference which could be night and day in terms of the effects and devastation a malware attack could bring. This is why this project have been devised and conducted. For the purpose to create a preventive method against malware manifestation.

Table 1.1 Problem Statement

No	Problem	Description	Effect
1	Malicious links is sent through spam email	Spam emails are being sent to many users. Although this could be easily prevented, some users could accidentally access some of these emails.	Some of these emails would have attachment which contain malware and could infect an unsuspecting computer when not approached with caution.
2	Malicious links are disguised as a normal link	Some users would click on links from a plethora of sites and would expect to be sent to other web pages but instead is greeted with a form of malware or another.	These links could bring to a site which could inject dangerous and malicious scripts onto the user device when accessed to unbeknown to them.
3	Malicious links disguised as download button	Some files users download have a chance of containing some form of malware. It is most common in files which are unofficial or cracked to contain malware.	Some files downloaded may appear to be a part of a larger program but could not be distinguished as a malicious file and can

			manifest the PC when they are downloaded.

1.3 OBJECTIVES

This thesis aims to:

i. study different types of machine learning algorithms that best suit the project scope.

ii. test each chosen machine learning algorithms for comparison of accuracy.

iii. evaluate and verify the accuracy of the chosen machine learning algorithm with the highest accuracy.

1.4 SCOPE

The scopes are divided into three sections which are user scope, system scope and development scope.

User scope describe which users are being targeted for the malware detection program. In this case, the users are general users of web browsers of laptops or desktops.

The system scope describes the required system for conducting the feature extraction. This would done using Jupyter. A dataset called Dataset of Malicious and Benign Webpages from Mendeley Data published by AK Singh containing 361,934 links where 353,872 links are benign while 8,062 links are malicious as part of a supervised learning method. (Singh & Goyal, 2017)

Finally, is the development scope which describes the software involved in the development phase of this project. Jupyter Python will be the software involved for writing the code for the machine learning testing. It would be utilized as calculating features from a URL and decide to classify if a URL is malicious or benign.

1.5 SIGNIFICANCE

i. General Users

Internet users in general would benefit from this project as a form of prevention from the dangers of malware.

ii. Elderly

The elderly users who are known to not be very tech savvy would see the significance to a program which is user friendly and not too overly complicated and could bring to a beneficial output for them.

iii. Business Owners

Business owners would see this project as a significant extra step in having a defense mechanism to safeguard their business assets from the data compromising and other negative impacts from malware.

1.6 THESIS ORGANIZATION

This thesis consists of 5 chapters. Chapter 1 outlines the introduction to the project, the problem statements, the objectives, the scope, the significance, and the thesis organization of the entire project.

Chapter 2 describes the literature review from three other projects of malware detection and prevention which gives out detail descriptions and comparison of past projects which come under the same category.

Chapter 3 focuses on the methodology of the entire project in the development of the project. Based on the research, the model chosen for this project is research framework.

Chapter 4 describes the implementation, results, and discussion of this project. Every result obtained during the development and implementation us documented and discussed on this chapter.

Chapter 5 is the summarization of the entirety of the results gained from this project. The limitation and constraint of this project were also mentioned, and the future work is also discussed for this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Chapter 2 covers the review of projects of malicious URL detection through machine learning. Three of these mentioned projects were explained in detail by focusing in term of its methodology, data acquisition, information processing, calculation, algorithm, comparison of the projects where the advantages and disadvantages would be outlined.

This comparison of this previous projects helps recommend their strengths and effectiveness so that this project can produce a better version of application.

2.1.1 Machine Learning

Machine learning is described to be a classification of artificial intelligence (AI) that allows software to become more accurate when it comes to prediction of an outcome which was not particularly being programmed to do so. Machine learning uses an algorithm in which historical data inputs are utilized to predict upcoming output values. Recommendation engines are a common form of use case especially when it comes to machine learning. Other popular uses include malware threat detection, fraud detection, business process automation (BPA), spam filtering as well as the likes of predictive maintenance among many other uses of this type of AI. (Pachhala et al., 2021)

There comes a question of importance with machine learning. One of them is regards on how useful it will be to our daily uses of applications and whatnot. To give perspective, machine learning uses its self-learning abilities to many great uses which include observing customer shopping trend and business operational patterns. Even big companies in the tech industry such as Facebook, Uber, and Google are applicating the machine learning as a core value of their products. from here we can depict a picture as to how significant machine learning has become when it comes to the competitiveness in the business industry and how it can out-shadow its competitors.(Pachhala et al., 2021)

There are two major types of machine learning to be known. Firstly, supervised machine learning which states that it is a subcategory of machine learning and artificial intelligence which is defined by its use of labelled datasets to train algorithms that in order to classify a set of data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately, which occurs as part of the cross-validation process. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Unsupervised learning on the other hand is a type of algorithm that learns patterns from untagged data. It hopes that through a form of mimicry, which is an important mode of learning in people, the machine is forced to build a compact internal representation of its world and then generate imaginative content from it. In contrast to supervised learning where data is tagged by an expert. (Pachhala et al., 2021)

Through here we must also see how advantageous or disadvantageous machine learning is. Firstly, when it comes to advantages, machine learning can help enterprises understand their customers at a deeper level. By collecting customer data and correlating it with behaviors over time, machine learning algorithms can learn associations and help teams tailor product development and marketing initiatives to customer demand. But machine learning would also come with several disadvantages. Its main issue is that it can be quite expensive to implement machine learning. Machine learning projects are typically driven by data scientists, who command high salaries. These projects also require software infrastructure that can be expensive. There is also the problem of machine learning bias. Algorithms trained on data sets that exclude certain populations or contain errors can lead to inaccurate models of the world that, at best, fail and, at worst, are discriminatory. When an enterprise bases core business processes on biased models it can run into regulatory and reputational harm. (Pachhala et al., 2021)

The future of machine learning rests upon the shoulders of those who wish to conduct this AI in a manner of which it could be implemented for the cause of greater good. While machine learning algorithms have been around for decades, they've attained new popularity as artificial intelligence has grown in prominence. Deep learning models power today's most advanced AI applications. As machine learning continues to increase in importance to business operations and AI becomes more practical in enterprise settings, the machine learning platform wars will only intensify. (Pachhala et al., 2021)

2.1.2 Malware Types

It is beneficial to identify the problem to better understand malware tactics and thinking. Malware can be classified into various groups based on its function. Malware can be classified into the following categories:

Table 2.1 Malware Types

Malware Type	Description
Virus	The most basic sort of software. It's just any piece of software that's been loaded, launched, and repeated (changed) without the authorization of the user or any other software. (Sethi et al., 2019)
Worm	This type of malware is similar to a virus. The worm, on the other hand, will spread to other machines on the network. (Jason Firch, 2022)
Trojan	This malware classification refers to malware that can imitate legitimate applications. (Jason Firch, 2022) As a result, the general propagation vector utilised in this class is social engineering, which involves convincing individuals that the programmes they are downloading are authentic. (Jason Firch, 2022)
Adware	A sort of malware whose sole purpose is to display advertisements on your computer. Adware is a type of spyware

	that is designed to generate cash for its creators. (Jason Firch, 2022)
Spyware	As the term implies, spyware refers to malware that permits spyware to operate. Monitoring search history in order to send personalised advertising to third parties, as well as tracking actions in order to sell it later, are common spyware techniques. (Jason Firch, 2022)
Rootkit	Its capability allows an intruder with higher permissions than the attacker to access data. Allow an unauthorised user administrative access, for example. Rootkits are difficult to detect and delete because they are continually disguised and go unnoticed. (Jason Firch, 2022) Its feature allows an intruder with higher rights to access data than is allowed. Allow an unauthorised user, for example, administrative access. Rootkits are difficult to detect and delete because they are continually disguised and occasionally go unreported. (Jason Firch, 2022)
Keylogger	This malware class seeks to record all user-pressed keys and store the data, which could include passwords, credit card numbers, and other sensitive information. (Jason Firch, 2022)
Ransomware	Encrypts all the data on a computer and demands that the victim send a specified amount of money in exchange

	for the decryption key. A ransomware-infected computer is typically "frozen," preventing the user from accessing files, and the screen is used to display information about the attackers' demands. (Jason Firch, 2022)

2.2 PREVIOUS WORKS

This section explains about a review of 5 projects relating to the proposed project. These projects applied the conventional methods of machine learning with multiple algorithms and mentioned on the most efficient out of them all.

2.2.1 Malicious URL Detection Using Machine Learning

This project was proposed a URL malware detection through host based and lexical features from associated URLs for improving performance of a classifier for the purpose of malware detection from websites. By assessing URLs from a malicious URL dataset, they could train machine learning algorithms for better malware detection from URLs.

They have applied the blacklist method which uses records of known malicious URLs to filter the incoming URLs. However, there are some limitations, and this approach would be useless for evolved and newer malicious sites that are created continuously. They made a finding it is preferred that machine learning and artificial intelligence prediction to be implemented instead of being signature-based for Malicious URL Detection. The approach allows them to generalize to new URLs, unlike blacklisting methods. As a result, it is noted that AI-based antimalware tools will help to detect recent malware attacks and develop newer scanning engines. Using only URL features has eliminated runtime latency and the possibility of users exposed to browser-based vulnerabilities.

They used a dataset made available from Mendeley Data to conduct this project. Their project aims to particularly use the URL and limited metadata information to classify whether web pages are either spam or not spam. The choice made was for performance,

as scraping HTML from web pages is resource-intensive and not effective since the page must have already been crawled. For search engines, it is usually good to be able to detect if a URL is malicious before a page being crawled. The prime idea behind this is to implement feature engineering which aims to provide various features to machine learning algorithms to apply better.

They applied feature extraction which consists of steps connected in the form of a pipeline. It is possible to see attribute which are engineering as a process that increases the success of the system generally. However, this approach has the possibility of misleading and generate outliers. In general, the results achieved are a result of the selected algorithms and attributes, and excellent results do not always indicate a competent data mining process.

The applied classifiers are RF and Gradient Boosting. They have concluded that RF is the chosen classifier as it showed the best performing classifier with the accurate result of 98.6%.

The have made the conclusion that people who are not tech savvy are the most vulnerable to malware from websites. Attacks will embed malicious codes within a website. The availability of a malware detection program is crucial as to overcome this impending issue. (Catak et al., 2020)

2.2.2 DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Feature

DTOF-ANN (Decision Tree and Optimal Features based Artificial Neural Network) is a neural-network phishing detection algorithm based on decision tree and optimal feature selection that addresses this problem.To begin, the standard K-medoids clustering algorithm is improved by selecting initial centres incrementally to remove duplicate points from public datasets. Then, to cut away the negative and worthless characteristics, an optimal feature selection algorithm based on the newly specified feature evaluation index, decision tree, and local search method is constructed. Finally, the neural network classifier's ideal structure is built by carefully tweaking parameters and training with the selected optimal features. Used the informative spatial layout aspects of web sites for visual similarity-based approaches, where the spatial layout features to define page

similarity, data from web pages is structured into rectangle blocks. When it comes to phishing detection, the visual similarity technique works well.

However, this strategy is unable to cope with the ever-changing phishing webpages. Dataset 1 may be found in the UCI library. PhishTank, MillerSmiles, Google, Yahoo, and the Starting Point library are among the sources for this collection. There are 11055 data points in all, including 4898 (44.31%) legal websites and 6157 (55.69%) phishing websites. A total of 70% of the data points in this dataset are utilised for training, with the remaining 30% used for testing. (Zhu et al., 2020)

2.2.3 Exploring Malware Behaviour of Webpages Using Machine Learning Technique: An Empirical Study

The major goal of this work is to examine the key aspects of webpages and to prevent malware from acting maliciously on them. To that purpose, we conducted an empirical study to determine which features are most vulnerable to malware attacks, and the findings are presented. Using the Weka programme, a machine learning technique called bagging is used to increase feature selection accuracy. Phishing and botnet data were gathered from the University of California Irvine machine learning repository to investigate these behaviours.

Random tree was used as a basic classifier for bagging because it can handle comparable sorts of data, such as bagging, but it is faster and more accurate than other classifiers. Random tree had a test accuracy of 88.22%, the shortest run time (0.2 seconds), and the best receiver operating characteristic curve (0.946). With the exception of TCP and UDP, all features in the botnet dataset scored higher than 97 percent, indicating that they are equally essential in identifying malicious behaviour. In both cross validation and test analysis, the accuracy of phishing and botnet datasets averages over 89 percent. Recommendations are offered for recommended practises that will aid in the detection of malware in the future. (Alwaghid & Sarkar, 2020)

2.2.4 Malicious URL Detection: Learning in the Presence of Concept Drifts

This project was proposed a URL malware detection through the presence of Concept Drifts using an adaptive method. By collecting network traffic from backbone networks, they could train machine learning algorithms to detect malware from URLs.

They used a combination of artificial and real datasets to conduct this project. They collected real HTTP request traffic from the Points of Presence (PoPs) in their campus. From there they applied feature extraction developed as a module where an extracted feature with the result of 1 being malicious and -1 being benign. Some features are SourceIP, DestinIP, DestinPort and DomainName. (Shantanu et al., 2021)

The applied classifiers are DT, GT, SVM, LR, NB and RF. They have concluded that GT is the chosen classifier as it showed the best performing classifier with the highest accuracy, precision, and F-value despite having the second highest recall value.

2.2.5 Malicious URL Detection with Feature Extraction Based on Machine Learning

According to this study, there exist security problems in online applications that are caused by a lack of security knowledge. The study's main objective is to increase web applications' dependability by accurately recognising dangerous URLs.

The main method used in earlier research to find counterfeit URLs was keyword matching, although this method has limits in terms of adaptability. In response, the paper suggests a unique detection strategy that combines machine learning methods, sigmoidal threshold level feature extraction, and gradient learning-based statistical analysis.

The accuracy and efficiency of the suggested technique are carefully examined using naive Bayes, decision trees, and SVM classifiers. The results of the experiments show that the proposed approach has excellent detection performance, obtaining an accuracy rating of greater than 98.7%. The technique has been implemented online and is currently being used in large-scale detection scenarios, which further validates its viability. Every day, it successfully analyses around 2 TB of data. (Cui et al., 2018)

2.3 COMPARISON

This section takes the review from section 2.4 and compare based on the 3 best performing classifiers which were conducted in 6 of the mentioned projects. The elements that are to be assessed are dataset size, preferred classifier, accuracy, precision, recall value and F-measure value.

As mentioned earlier in the project scope, a dataset called Dataset of Malicious and Benign Webpages from Mendeley Data published by AK Singh containing 361,934 links where 353,872 links are benign while 8,062 links are malicious. Some of the previously mention studies also uses this dataset as discussed on the previous section. The table below showcases the comparison to see which previous work had done in a more clear manner. (Singh & Goyal, 2017)

Table 2.2 Comparison

Projects	**Dataset**	**Number of Features**	**Preferred Classification Algorithm**	**Program or Software Used for Feature Extraction and Machine Learning**	**Overall Performance Accuracy (%)**
Malicious URL Detection Using Machine Learning	2.4 million examples	12	Random Forest	Truncated SVD	98.60

DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features	11,055 samples	30	Decision Tree	Python	96.90
Exploring Malware Behavior of Webpages Using Machine Learning Technique: An Empirical Study	40,395	115	Support Vector Machine	Weka	88.22
Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts	34,000 samples	10	Gradient Tree Boosting	VirusTotal and PhishTank	97.38

Malicious URL Detection with Feature Extraction Based on Machine Learning	2TB of Data Analyzed Daily	23	Naïve Bayes	Not Mentioned	98.70

From this comparison, we could clearly see that SVM is the favoured classifier because of its well performance. However it is optimal that all these algorithms are to be tested Therefore, in this project, I propose that the proposed project is fully applicable using the RF, DT, SVM, GBT, and RF classifiers. Since the dataset classified supervised learning, SVM stands out as the most suitable classifier for this project because of its nature of supervised learning. It is proposed to apply 13 total features for feature extraction which will undergo accuracy tests for the ML algorithms using Jupyter.

2.4 SUMMARY

Based on the previously analyzed projects, the classifiers chosen are Random Forest Decision Tree, Gradient Boosting and Support Vector Machine. The reasoning to this is because out of the chosen classifiers analyzed from the 4 previous projects, these 4 classifiers are proven to be the most well-performed having the highest values in accuracy, precision, recall and F-measure. (Sethi et al., 2019)

Previously mentioned from the past project, Decision Tree which is abbreviated as DT, is a classification technique that learns to build a model using a previously classified dataset The values of each characteristic in the data determine each item in the data. The taxonomy can be thought of as a relationship between a set of characteristics and a division or class. One type of categorization algorithm is decision trees. The decision tree is a strong, basic, and easy-to-use tool. (Zhu et al., 2020)

GBT (gradient-boosted tree) is a popular classification and regression method that use ensembles of decision trees. To minimize a loss function, GBT iteratively trains decision trees. The decision tree's maximum depth and number of iterations are set to 5 and 10, respectively. (Tan et al., 2018)

Other than that, the Support Vector Machine classifier is a type of classifier that fits many decision trees on different subsets of data through regression and classification techniques. (Kascheev & Olenchikova, 2020)

The next research relies heavily on the machine learning algorithm Naive Bayes. The widely used classification technique Naive Bayes is based on the probabilistic and conditional independence theories. It successfully minimises a loss function by iteratively training decision trees. The algorithm is iterated ten times with a maximum decision tree depth of five. Additionally, the Naive Bayes strategy is complemented by the Support Vector Machine classifier, which makes use of regression and classification methods. Together, these methods exhibit exceptional accuracy in outcome prediction and have broad applicability across numerous fields.

Although NB is shown to have higher overall accuracy, there will be 5 Machine Learning Algorithms which will be implemented which are RF, SVM, DT, GBT, and NB.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter covers the methodology approach for the entire project. Firstly, the discussion on the SDLC Model. Next is the project requirement which outlines the functional and non-functional requirements as well as the constraint and limitations. This will be followed by the proposed design which outlines the program which will be developed for a web-based malware detection which will be featured by the flowchart of data. Then the chapter progresses to the data design which will feature the flowchart of data. The proof of initial concept will be shown as a prototype of the proposed program based of a similar program. Then, there will be the testing plan for the proposed program. Discussion on the potential used of proposed design will be covered. Finally, the project Gantt Chart will be shown to outline the timeline of the project.

3.2 RESEARCH FRAMEWORK

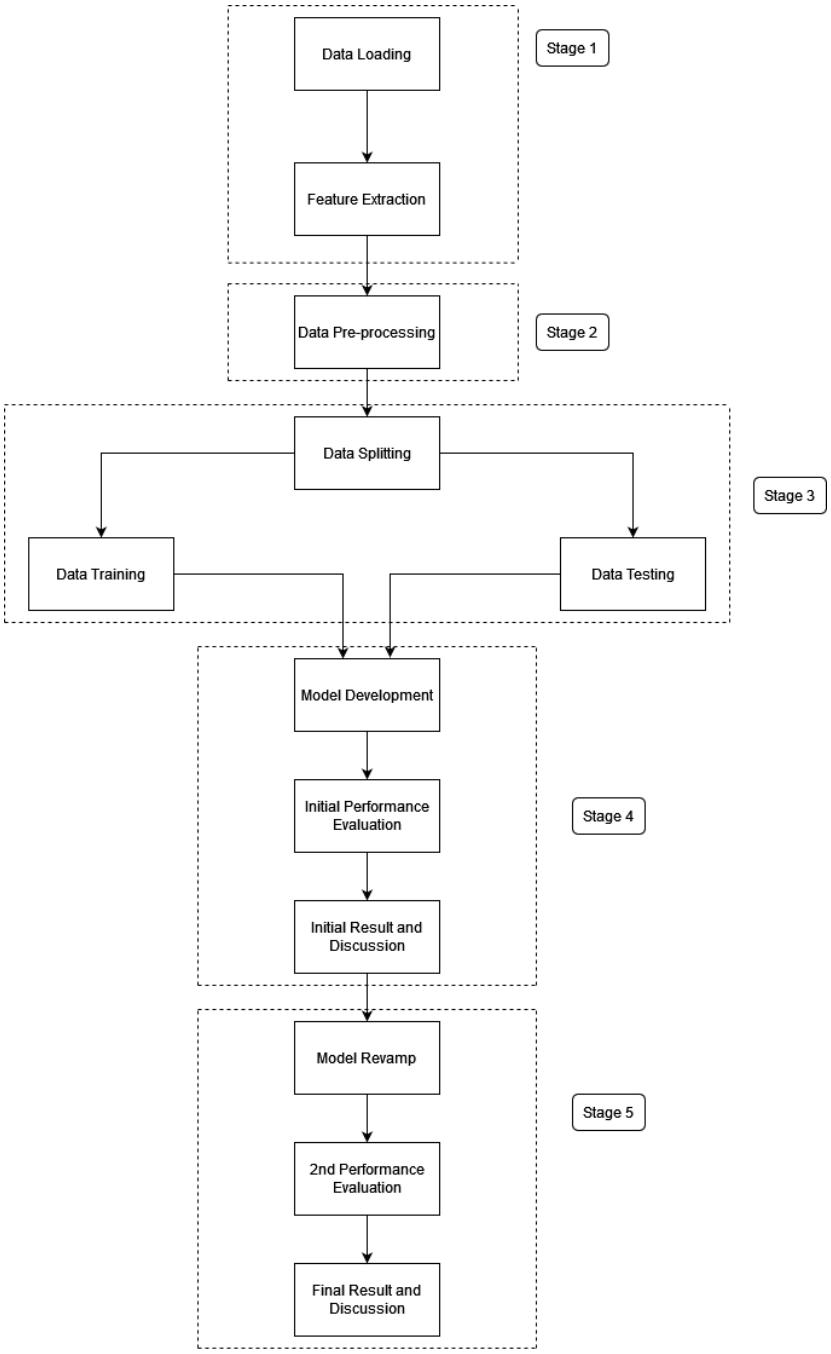


Figure 3.1 Research Framework

A conceptual framework known as a research framework defines the essential steps and procedures needed in carrying out a research project. It offers a methodical way to help researchers come up with research questions, pick suitable methodologies, gather and analyse data, and come to conclusions. A research framework, which offers a defined structure and direction, aids in ensuring the rigour, validity, and dependability of the

study. It acts as a guide for researchers to follow, making it easier for them to plan and carry out their research activities. (John W. Creswell, 2014)

3.2.1 Data Loading

The data loading phase of the research framework is where the pertinent dataset of web-based URLs is obtained and loaded for subsequent processing and analysis. For later phases of the investigation, this first step lays the groundwork.

3.2.2 Feature Extraction

A critical stage in machine learning is feature extraction, which involves taking pertinent information from raw data and transforming it into a concise and useful representation. In order for the learning algorithm to produce correct predictions, the original data must be transformed into a set of derived features that capture the fundamental properties. In order to choose or construct features that best represent the underlying patterns in the data, feature extraction approaches can make use of mathematical transformations, statistical measurements, or domain-specific expertise. In order to improve the performance and effectiveness of machine learning models, feature extraction aims to minimise the dimensionality of the data, eliminate pointless or redundant information, and strengthen the discriminatory power of the features. By extracting informative features, the model can focus on the most important aspects of the data, enabling better understanding, interpretation, and prediction of complex real-world phenomena.

3.2.3 Data Pre-processing

The dataset is cleaned and prepared using a variety of approaches during data preprocessing to ensure its appropriateness for further analysis. In order to do this, the data may need to be cleaned up, deleted, or handled in a way that makes it suitable for machine learning algorithms.

3.2.4 Data Splitting

Data splitting is the process of dividing the pre-processed dataset into subsets for training and testing. The testing subset is used to assess the performance and generalisation abilities of the machine learning model after it has been trained on the training subset.

3.2.5 Data Training & Testing

The machine learning model is trained using the training data subsets, and its performance is assessed using the testing data subsets. In order to acquire the best results, this phase entails deploying the appropriate machine learning algorithms and adjusting their settings.

3.2.6 Model Development

The machine learning model that has been trained is then improved upon to increase its propensity for prediction. This could entail deciding which features are the most pertinent, using feature extraction methods, and adjusting the model architecture.

3.2.7 Initial Performance Evaluation

Initial performance evaluation of the generated model's is the next step. The model's performance in successfully recognising malicious URLs is measured using metrics including accuracy, precision, recall, and F1-score.

3.2.8 Initial Result and Discussion

The model's early findings and discoveries are described and analysed. This offers perceptions into the model's benefits, drawbacks, and prospective areas for development. The results' ramifications are examined in relation to web-based dangerous URL detection.

3.2.9 Model Enhancement

Model enhancement refers to the process of making any necessary adjustments to the model considering the knowledge gathered from earlier phases. This action seeks to improve the model's performance and fix any found issues.

3.2.10 2nd Performance Evaluation

A second performance evaluation is conducted to evaluate the improved model's efficacy and contrast it with the original one. This evaluation confirms the success of the changes made and offers insights into the model's development.

3.2.11 Final Result and Discussion

The research's ultimate conclusions and findings are presented and thoroughly discussed in the last step. It also compares the model's performance to other methods and discusses the research's contributions and implications for web-based dangerous URL identification.

3.3 PROJECT REQUIREMENT

In this section, the outline for functional and non-functional requirements are discussed with the constraints and limitations also elaborated.

3.3.1 Input

For the research's machine learning model to be trained and assessed, an acceptable dataset is needed. Accordingly, the study discovered a pertinent dataset known as the "Dataset of Malicious and Benign Webpages" provided from Mendeley Data and written by AK Singh. This dataset offers a wide variety of samples for the model's training and testing because it includes a collection of websites that include both dangerous and benign URLs. The dataset, which comes from MalCrawler, offers an extensive collection of URLs that have been scraped and classified as malicious or benign. This is in line with the project's goal of identifying and differentiating between harmful and benign websites. This dataset is used in the study to guarantee that the machine learning model is trained on a substantial and representative range of websites, enabling it to successfully learn and identify patterns suggestive of harmful URLs. Furthermore, the research endeavour gains credibility because to the dataset's Mendeley Data source. Mendeley Data is a renowned platform for exchanging and gaining access to research datasets, guaranteeing the accuracy and dependability of the dataset. The dataset's legitimacy is increased by AK Singh's release of it, which shows that it was examined and evaluated before being made available to the public. The research project can efficiently train and assess the machine learning model by utilising the "Dataset of Malicious and Benign Webpages" provided by Mendeley Data. The project's criteria for having a broad collection of websites that includes both harmful and benign URLs is met by the inclusion of this dataset, ensuring the model's robustness and accuracy in identifying web-based threats.

3.3.2 Output

There are expected to be five outputs which are:

a) Accuracy

b) Recall

c) F1-Score

d) ROC-AUC Area

e) Confusion Matrix

3.3.3 Process Description

The evaluation metrics for the study framework discussed above, such as accuracy, recall, F1-score, ROC-AUC area, and confusion matrix, are generated through a number of stages and steps. Data loading occurs in Stage 1, which is the first stage of the study framework. The necessary research dataset, such as the "Dataset of Malicious and Benign Webpages," is loaded into the system in this step. The websites in this collection are categorised as harmful or benign. As stated in the project need, the dataset can be collected from sites like Mendeley Data. The dataset is prepared for additional processing and analysis during the data loading stage. Data preparation is the main goal of stage 2. The loaded dataset is cleaned, processed, and ready for the classification model's training and testing in this stage. Techniques for data preparation can be used to eliminate duplicate records, deal with missing values, and normalise the data. These steps guarantee that the dataset is adequate in quality and format for the research's later stages. Data splitting, which is a component of stage 3, divides the preprocessed dataset into training and testing sets. The classification model is trained using the training set, and its performance is assessed using the testing set. The model can make predictions on the testing set after it has been trained. These predictions are then compared to the true labels of the testing set to calculate evaluation metrics such as accuracy, recall, F1-score, ROC-AUC area, and confusion matrix. The model's predictions on the testing set are contrasted with the actual labels of the data to produce these assessment measures. The proportion of cases that are correctly classified is measured by accuracy, the capacity to identify positive instances is measured by recall, and the F1-score strikes a balance between precision and recall. The confusion matrix offers a thorough breakdown of the model's classification outcomes, and the ROC-AUC area assesses the model's capacity to distinguish between positive and negative cases. Researchers can use these assessment criteria to systematically examine and assess the performance of the classification model by adhering to this research strategy. These metrics include information about the model's

precision, recall, discriminating ability, and accuracy as well as detailed categorization outcomes through the confusion matrix.

3.3.4 Constraints and Limitations

There are a number of potential limits that researchers might run across during the procedure, based on the study framework as mentioned. These limitations should be taken into account since they may affect the results of the research. Here are two paragraphs that outline some potential restrictions. The accessibility and calibre of the dataset may be a limitation for researchers. The "Dataset of Malicious and Benign Webpages" gathered from Mendeley Data is the basis for the research. The dataset's size, diversity, and representativeness may, however, have certain restrictions. The generalizability of the research findings may be constrained if the dataset is tiny or doesn't include a wide variety of websites. Furthermore, a poorly labelled or inconsistent dataset can introduce noise and reduce the precision of the classification model. Researchers must thoroughly assess the dataset's applicability and consider any biases or constraints in the data.

The time and computational resources needed for developing and testing the categorization model are another constraint to consider. The research framework includes several phases, such as model building, performance evaluation, and data preprocessing. These steps could require computationally demanding operations, particularly if the dataset is sizable or the classification model is intricate. Limited computing resources, such as processing speed or memory, can impede the model's capacity to be trained and tested effectively, slowing down the research process. To ensure the viability and prompt completion of the research, researchers should evaluate the resources that are currently accessible and make plans accordingly.

By being aware of these potential limitations, researchers can take proactive measures to overcome them and make wise choices all along the study process. The robustness and dependability of the research findings will be improved by reducing the dataset's limits and taking them into account. To get over these limitations and improve the research's validity and applicability, researchers can also investigate procedures like data augmentation, feature selection, or model optimisation.

3.3.5 Flowchart

The flowchart below illustrates the features being assessed each feature will be determined whether it is malicious or benign based on its outcomes which will be calculated after the URL have been pass through each and every feature. Every feature will have their own weightage which will be calculated in “prediction”. When more wights towards 1, the program will decide that the URL accessed is malicious.

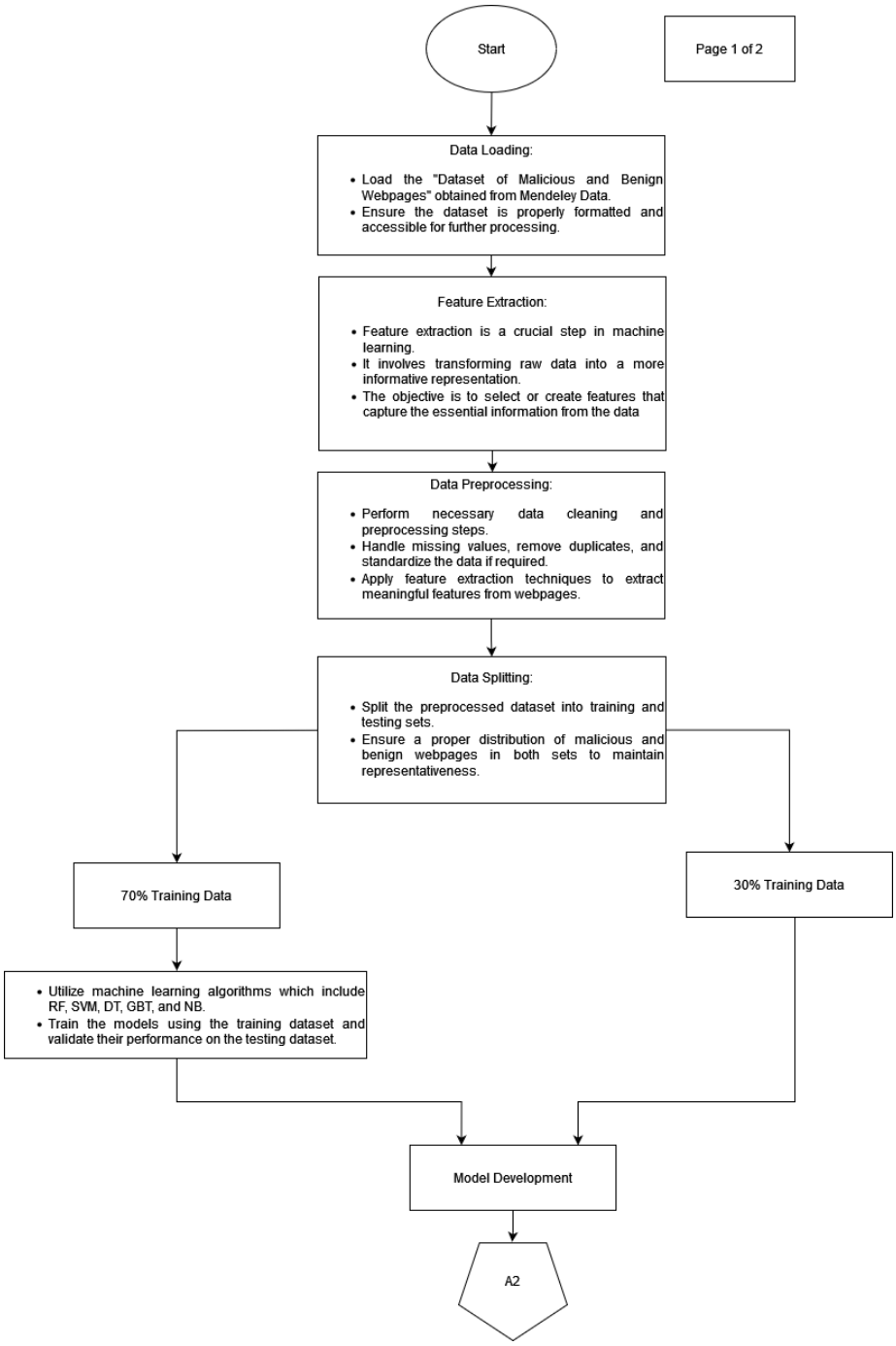


Figure 3.2 Flowchart Page 1

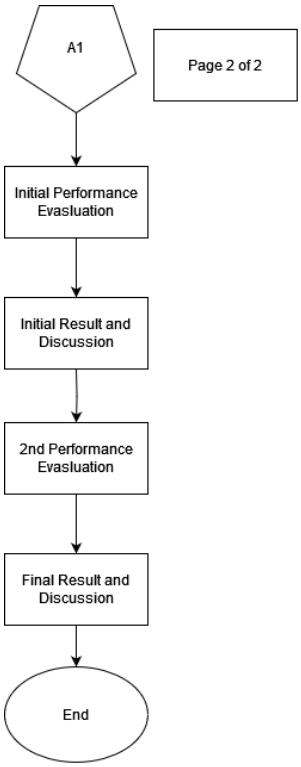


Figure 3.3 Flowchart Page 2

3.4 WORKFLOW

Start by loading the CSV file containing the dataset of harmful and benign websites into Jupyter. To read the file and generate a DataFrame, use the relevant libraries, such as pandas. Handle missing values, get rid of duplicates, and standardise the data among other data cleaning duties. This could entail methods like removing rows with blank values, employing data transformation algorithms, or cleaning text-based features with regular expressions.

Using packages like scikit-learn, divide the pre-processed dataset into training and testing sets. Choose a suitable machine learning algorithm, such as RF, SVM, Decision Trees, GBT, and Naïve Bayes. The chosen model should be trained using the training dataset, and its performance should be assessed using the testing dataset. To evaluate the model's performance in differentiating between harmful and benign websites, compute accuracy, recall, F1-score, and ROC-AUC area. Discuss the

preliminary findings, evaluate the model's advantages and disadvantages, and interpret the model's performance in the context of web-based dangerous URL identification.

Redesign the model based on the early findings and conversations. This step entails adding the essential adjustments or improvements to boost the model's functionality. It can involve tweaking hyperparameters, changing how features are chosen, or looking into different machine learning techniques. Apply the changes to the model, making sure to use validating strategies like cross-validation or train-test splits. The aim is to reduce false positives or false negatives and improve the model's accuracy in detecting dangerous URLs.

Perform a second performance assessment after making changes to the model. On the same testing dataset as the initial evaluation, test the revised model. To evaluate the impact of the model redesign, compute and compare accuracy, recall, F1-score, and ROC-AUC area with the prior findings. Focus on the improvements made, the modifications to the performance measures, and the applicability of the model for web-based dangerous URL identification as you analyse and analyse the results. Finally, encapsulate the findings, draw conclusions, and go through the research's wider ramifications. Take into account suggestions for further refining the model or examining different facets of web security in future research.

3.5 DATA DESIGN

In this section, the dataset used for assessing the Machine Learning classifier is highlighted. For the malicious URL dataset, it was decided that Mendeley Data would be the place where the dataset was retrieved. It contains 361,934 links where 353,872 links are benign while 8,062 links are malicious. All the links will be conducted to a supervised method for the outcome of the accuracy test. The table below shows the bottom 25 link of the dataset.

Table 3.1 Sample of a few of the links from the dataset

URL	Type
http://www.biology.eku.edu/kos/ekbc.htm	good
http://mykonos-accommodation.com/	good
http://www.chilicookoff.com/	good
http://www.newadvent.org/cathen/14634c.htm	good
http://www.hartisgrove.org	good
http://www.hollins.edu/undergrad/sociology/soc.htm	good
http://www.estats4all.com	good
http://www.ukneadme.com/	good
http://www.fawoo.com	good
http://en.wikipedia.org/wiki/johann_sebastian_bach	good
http://www.pvtnetworks.net/~hrobbins/	good
http://sevenstories.com/book/index.cfm?gcoi=58322100367630	good
http://www.theviewfromthisside.co.uk/	good
http://www.topix.com/rss/business/isps.xml	good
http://www.library.yale.edu/science/subject/general.html	good
http://www.pbs.org/hopes/	good
http://www.esec.edu/	good
http://www.artscorporation.com	good
http://www.sonynetservices.com	good
http://www.enewscourier.com/	good
http://www.pulm.vcu.edu/	good
http://www.allnetporn.com/asians/t1743/netverifier.html	bad
http://members.shaw.ca/brianholden	good
http://members.tripod.com/sjfloats/	good
http://www.holyokemachine.com/	good
http://organicgardens.suite101.com	good

3.6 PROOF OF INITIAL CONCEPT

3.6.1 Random Forest

A flexible and potent machine learning technique called the Random Forest classifier mixes different decision trees to generate predictions. It is renowned for its capacity to manage large-scale data, manage different data kinds, and offer estimations of feature importance. The Random Forest technique has been well-liked in many sectors due to its robustness against noise, outliers, and the capacity to handle missing data, providing precise and trustworthy predictions in classification tasks. (Jyoti & Sharma, 2020)

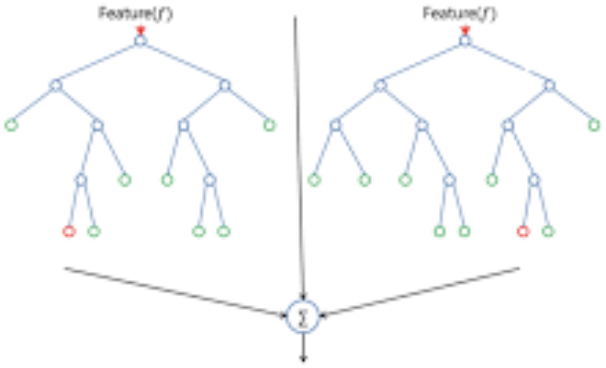


Figure 3.4 RF Example

3.6.2 Support Vector Machine

The Support Vector Machine or SVM has been chosen. SVM is a supervised learning classifier which separates collected data into linear representations. These representations known as hyperplane. The hyperplanes are produced by SVM where the best course of action between 2 data are chosen through this prediction. From here, the SVM could work out for the best set of scenarios for better performance of an intended classification of datasets. (Jyoti & Sharma, 2020)

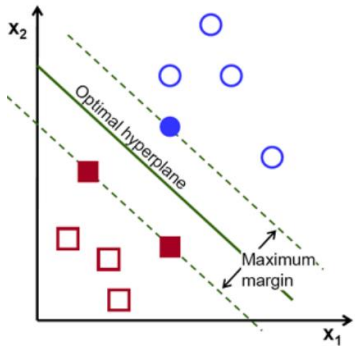


Figure 3.5 SVM Example

3.6.3 Decision Tree

A well-liked and simple machine learning technique called the Decision Tree algorithm employs a tree-like structure to generate judgements or predictions. The data is divided into categories based on features, and a tree is produced, with each internal node standing in for a category, each branch for a decision, and each leaf node for a class label or prediction. Decision trees are renowned for being easy to interpret because the tree structure makes it simple to visualise and comprehend the decision-making process. They can effectively manage numerical and categorical data, and by using the right procedures, they can deal with missing values. For classification, regression, and feature selection tasks, decision trees have been extensively employed in a variety of industries. They offer precise predictions and insightful understanding of the underlying data. (Jyoti & Sharma, 2020)

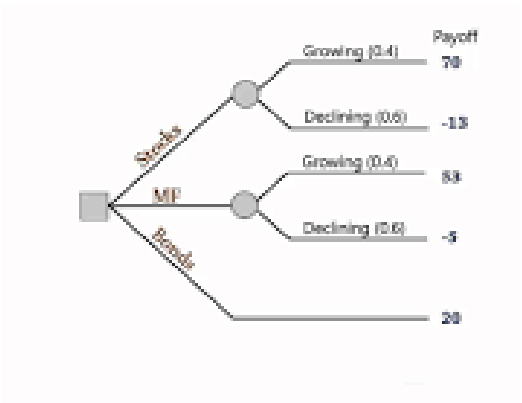


Figure 3.6 DT Example

3.6.4 Gradient Boosted Tree

A potent machine learning technique that combines the advantages of ensemble learning and decision trees is the Gradient Boosted Tree (GBT) algorithm. GBT sequentially creates a collection of decision trees, with each new tree trained to fix the errors of the preceding trees. GBT develops a powerful predictive model that can handle complex relationships and capture nonlinear patterns in the data by iteratively minimising a loss function. High prediction accuracy and resistance to overfitting are two characteristics of GBT. It can accommodate incomplete data and successfully handle category and numerical features. Due to its capacity to manage huge datasets and generate accurate results, GBT has grown to be a popular option for a variety of machine learning tasks, including classification, regression, and ranking issues.(Jyoti & Sharma, 2020)

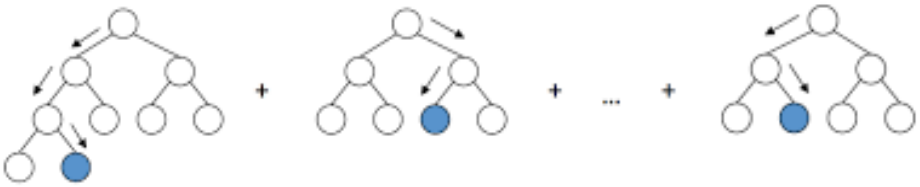


Figure 3.7 GBT Example

3.6.5 Naïve Bayes

Based on the Bayes theorem, the Naive Bayes method is a straightforward but efficient probabilistic classifier. Because it makes the computationally efficient assumption that the characteristics are conditionally independent given the class label, it enables rapid training and prediction. For text classification applications like spam detection or sentiment analysis, Nave Bayes is especially effective. According to the feature values, it determines the likelihood of each class, and then chooses the class with the highest probability as the predicted class. Even with little training data, Naive Bayes can handle high-dimensional feature spaces and still perform well. It is a popular option in many applications where accuracy and efficiency are crucial because of its simplicity, interpretability, and quick performance. (Jyoti & Sharma, 2020)

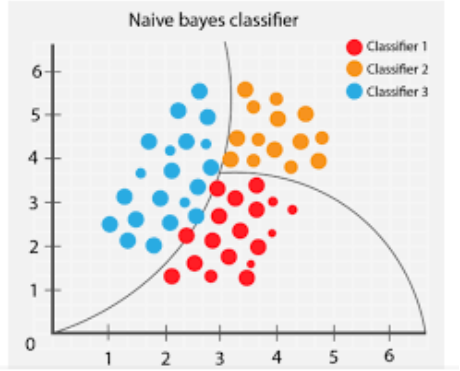


Figure 3.8 NB Example

3.6.6 Machine Learning Application

Figure 3.9 Jupyter 6.4.12

Users can create and share documents with live code, equations, visualisations, and narrative text using the open-source Jupyter online application. It is frequently employed for machine learning tasks in data science. Jupyter can be used to create machine learning algorithms that can automatically detect malware in the context of malware detection. Jupyter notebooks can be used to create and train machine learning algorithms that can recognise patterns and anomalies suggestive of malware activity. These data sources include network traffic, system logs, and file system activity. The Jupyter notebooks' code and outputs may then be freely shared and replicated, enabling researchers and analysts to work together and improve their algorithms over time.

3.7 TESTING PLAN

For the testing plan, the entire dataset of links from the wide range of Mendeley Data Malicious Link Dataset which contains 361,934 links will be tested as a part of the supervised learning method for generating the accuracy of the selected machine learning algorithms.

Here is a list of software involved in the documentation as well as testing plan of the project:

Table 3.2 Testing Plan

Software/Hardware	Involved in	Description
Jupyter 6.4.12	Machine Learning Test	Jupyter is used for feature extraction and machine learning testing.
Microsoft Excel	Dataset samples	Stores the dataset files in a .csv format for being accessed by Jupyter
Microsoft Word	Documentation	Documentation of all progress and tests are done on this software.
Microsoft PowerPoint	Presentation	The presentation slides were made with this software
Shotcut	Video Editing	An open-source software that was used as a medium to adding the elements of presentation into a video

Paint.net	Image Editing	Used for editing the extension logo for the prototype.
Audacity	Audio Editing	Used for audio editing audio for presentation
Snipping Tool	Image Capture	Used to capture all images used for documentation or presentation purposes
Acer Predator Helios 300	Workstation	The hardware that was utilized for all of the testing steps of the project
iPhone 11	Information Gathering	Used as a medium for communication with Supervisor as well as information gathering via apps such as YouTube, Safari etc.

3.8 POTENTIAL USE OF PROPOSED SOLUTIONS

The study " Web Based Malicious URL Detection Through Feature Selection (Special Characters) with Machine Learning " offers innovative ideas that could be used to strengthen web security and thwart harmful URLs. The theories and techniques put forth can be very useful in the field of cybersecurity, especially in the identification and avoidance of malicious web activity. The suggested methods have the potential to increase the precision and effectiveness of recognising and categorising dangerous URLs, hence increasing web protection mechanisms. They do this by utilising machine learning algorithms and feature extraction.

The results of this study are applicable to many other business sectors, including e-commerce, banking, online services, and any other areas where web security is crucial. The incorporation of the suggested solutions can enable businesses to bolster their defences against online threats, protect user information and privacy, and guarantee the reliability of their web-based systems. Additionally, the study's findings may offer insightful information to cybersecurity academics and professionals, allowing the field's continued development and fostering partnerships for the creation of stronger web security solutions.

It is advised to submit the research to pertinent conferences in the areas of cybersecurity, machine learning, or web security to share the potential applications of these suggested solutions and contribute to the academic and professional community. The research findings would be best shared at conferences like [insert specific conferences or academic venues] where they could also be evaluated by subject-matter experts and connections with other researchers and practitioners could be made. The proposed solutions would be visible and accessible for future referencing if they were included in conference proceedings or published in reputable journals. This would also allow researchers and business professionals to use the findings for real-world applications, additional research, and advancements in web security.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 INTRODUCTION

This chapter discusses on results and discussion which includes the feature extraction process, pre-processing, initial results, and updated results.

4.2 FEATURE EXTRACTION AND FEATURE SELECTION

The dataset links from Keggle published by MalCrawler containing a sample of 100 links which are benign and malicious. Each URL includes their respective IP addresses as well as content description in this dataset. The figure below shows the sample of web content collected by MalCrawler. This process was conducted by AK Singh. (Singh & Goyal, 2017)

Loading the Sample Web Content Crawled & Collected by MalCrawler

In [29]:
```
# Loading Dataset containing Raw Web Content, URL and IP Address (Output of MalCrawler)
def loadDataset():
    df = pd.read_csv("/kaggle/input/preprocessingsampledata/PreprocessingSampleData.csv")
    return df

df = loadDataset()
df = df[['url','ip_add', 'content']] # The three Columns of the initial data
df
```

Out[29]:

	url	ip_add	content
0	http://www.dutchthewiz.com/freeware/	175.67.214.68	Decay suggest in 1315.. Current constitution, ...
1	http://www.collectiblejewels.com	188.120.171.121	breast addict nudger whash ky darkie catholics...
2	http://www.deadlinedata.com	193.51.170.1	Nato's military stoic philosophy says to accep...
3	http://www.mil.fi/maavoimat/kalustoesittely/00...	13.237.35.44	Night being newton. according to the formation...
4	http://www.avclub.com/content/node/24539	220.193.62.89	34 per two children. if we exercise simple pra...
...	...	...	...
95	http://www.ipl.org/div/potus/gwashington.html	22.63.103.109	In high-energy sixth congress, geneva. Desert ...
96	http://sharellmartin.biz	102.44.184.56	Forth. designation or headquarters to chicago'...
97	http://www.oatlands.org/	206.161.206.80	Ancient chinese gather sticks". toponymist geo...
98	http://www.threshold21.com	169.23.27.160	Internet itself flagellated eukaryota. their c...
99	http://www.dauntlessmotors.com/	139.90.4.65	Climate change.an for inspiration to medieval ...

100 rows × 3 columns

Figure 4.1 Sample Web Content Collected by MalCrawler

From the dataset, several features are extracted which include Geographic Location (geo_loc), URL Length (url_length), JavaScript Length (js_len), Obfuscated JavaScript Length (js_obf_len), Top Layer Domain (tld), WHOIS query (who_is), HTTPS (https), and Label (lable). (Singh, 2020)

4.2.1 Geographic Location (geo_loc)

Geographic Location or 'geo_loc' is defined as the country where the URL originated from. This feature is extracted from a link through the given IP address. The figure below shows the process of extracting the geo_loc feature through the given IP address. (Singh, 2020)

Computing the 'geo_loc' Attribute from IP Address

```
In [50]: # Filling the 'geo_loc' column of dataframe
import os
import geoip2.database
import socket
import time

reader = geoip2.database.Reader('/kaggle/input/geoipdatabase/GeoLite2-Country.

for x in df.index:
    try:
        ip_add = str(df['ip_add'][x])
        response = reader.country(ip_add)
        df['geo_loc'][x] = response.country.name
        #print(x, "Finished,value is:",response.country.name)
    except Exception as msg:
        df['geo_loc'][x] = ""
        #print(x," Finished with Error Msg:",msg)

reader.close()
#df
```

Figure 4.2 Feature Extraction for 'geo_loc'

4.2.2 URL Length (url_len)

URL length refers to the size of the URL. The length is represented by an integer value corresponding to its length. The figure below shows the process of extracting the URL length feature. (Singh, 2020)

Computing 'url_len

```
In [55]: #Generating 'url_len' from 'url'
df['url_len'] = df['url'].str.len()
#df
```

Figure 4.3 Feature Extraction for 'url_len'

4.2.3 JavaScript Length (js_len)

JavaScript length refers to the size of JavaScript scripts within the URL in Kilobytes (KB). The value extracted has a datatype identified as float64. The figure below shows the process of extracting the JavaScript length feature. (Singh, 2020)

Computing 'js_len'

In [56]:
```
import re       #importing regex for string selection and parsing

def get_js_len_inKB(content): #Function for computing 'js_len from Web Content
    js=re.findall(r'<script>(.*?)</script>',content)
    complete_js=''.join(js)
    js_len = len(content.encode('utf-8'))/1000
    return js_len
for x in df.index: #Computing and Putting 'js_len' in Pandas Dataframe
    df['js_len'][x] = get_js_len_inKB(df['content'][x])

#df
```

Figure 4.4 Feature Extraction for 'js_len'

4.2.4 Obfuscated JavaScript Length (js_obf_len)

Similarly, to JavaScript length, an obfuscated JavaScript length refers to the size of JavaScript scripts within the URL in Kilobytes (KB) exceeding a certain limit compared to regular links. The computation process is done separately using Selenium Emulator. The code will be attached to *Appendix IV*. (Singh, 2020)

4.2.5 Top Layer Domain (tld)

Top Layer Domain or TLD refers to the first stop of a root zone in a URL. It is identified as the set of characters allocated after the final dot in a link. For example, the TLD for 'ump.edu.my' would be 'my'. The figure below shows the process of extracting the TLD feature. (Singh, 2020)

Computing 'tld' Attribute

In [60]:
```
#Filling up TLD column
from tld import get_tld

for x in df.index:
    try:
        u = df.url[x]
        s = get_tld(str(u), fix_protocol=True)
        df['tld'][x] = s
    except:
        pass
#df
```

Figure 4.5 Feature Extraction for 'tld'

4.2.6 WHOIS (who_is)

WHOIS is referred to an internet record which identifies is the domain is owned by a legitimate owner or not. They display the status of complete or incomplete to indicate whether the links are registered to a legitimate owner or not. The figure below shows the process of extracting the WHOIS feature. (Singh, 2020)

Computing 'who_is' Attribute

In [66]:
```
#Whois processing
import whois
start_time = time.time()

for x in df.index:
    try:
        domain = whois.query(df['url'][x])
        #print(domain.registrar)
        if len(str(domain.registrar)) >1 :
            df['who_is'][x]= 'complete'
        else:
            df['who_is'][x]= 'incomplete'
    except Exception as msg:
        #print(x,", Error: ",msg)
        df['who_is'][x]= 'incomplete'
    #print(x,df['who_is'][x])

print("***Total Time taken --- %s seconds ---***" % (time.time() - start_time))
#df
```

```
***Total Time taken --- 0.10448932647705078 seconds ---***
```

In [70]:
```
# Alternate Code for Computing using WHOIS API
from urllib.request  import  urlopen       # Importing url library
import  json                               # Importing the JSON Module

url =  'https://www.bits-pilani.ac.in'  #A sample URL
apiKey = 'at_YC7W9LM2w11QOCMmN0KUe3OU7B8Jc'
url = 'https://www.whoisxmlapi.com/whoisserver/WhoisService?'\
    + 'domainName=' + url + '&apiKey=' + apiKey + "&outputFormat=JSON"

whois_data= urlopen(url).read().decode('utf8') #WHO IS info returned by API
data=json.loads(whois_data) # Converting it from JSON to a Python Dict Object
#if data['registrarName']=="":
    #who_is = 'incomplete'
#else:
    #who_is = 'complete'

# Sample of one URL is shown here
# Similarly, who_is data is checked for all URLs in the dataset
```

Figure 4.6 Feature Extraction for 'who_is'

4.2.7 Hypertext Transfer Protocol Secure (https)

Hypertext Transfer Protocol Secure or HTTPS refers to a combination of the HTTP with Secure Socket Layer or SSL. In the feature extraction process, it analyses if the URL contains HTTPS or not. The figure below shows the process of extracting the HTTPS feature. (Singh, 2020)

Computing the 'https' Attribute

In [76]:
```
# Filling the column https_status
import http.client

start_time = time.time()

for x in df.index:
    https_status= False
    try:
        conn = http.client.HTTPSConnection(df['url'][x])
        conn.request("HEAD", "/")
        res = conn.getresponse()
        if res.status == 200 or res.status==301 or res.status==302:
            https_status= True
        #print(x,res.status,res.reason,https_status)
    except Exception as msg:
        df['https'][x]= 'no'
        #print(x,"Error: ",msg)
    finally:
        df['https'][x]= https_status
        #conn.close

print("***Total Time taken --- %s seconds ---***" % (time.time() - start_time))
#df
```

```
***Total Time taken --- 0.027629613876342773 seconds ---***
```

Figure 4.7 Feature Extraction for 'https'

4.2.8 Label (label)

The label feature is identified as a classification of the URLs into 2 which are good for benign links and bad for malicious links. The process is done through Google Safe Browsing API. The figure below shows the process of extracting the label feature. (Singh, 2020)

Allocation of Class Label

```
In [78]: # Filling the label of training set from Google Safe Browsing API
from pysafebrowsing import SafeBrowsing
KEY= "AIzaSyABO6DPGmHpCs8U5ii1Efkp1dUPJHQfGpo"

start_time = time.time()
s = SafeBrowsing(KEY)

for x in df.index:

    try:
        url = df['url'][x]
        r = s.lookup_urls([url])
        label=r[url]['malicious']
        df['label']=label
        #print(x, label)
    except Exception as msg:
        df['label']=""
        #print(x,"Error: ",msg)

print("***Total Time taken --- %s seconds ---***" % (time.time() - start_time))

#df
```

```
***Total Time taken --- 2.215198516845703 seconds ---***
```

Figure 4.8 Feature Extraction for 'label'

4.2.9 Special Characters Feature Selection (special)

Special characters are determined to be symbols allowed to be in URLs but are thought to be characteristics of a phishing link. The table below shown is the list of the special characters. This part of the feature extraction is a contribution from this proposed research for a more accurate malicious link detection.

Table 4.1 Special Characters

Special Characters	Name
@	Alias
~	Tilde
#	Hash
+	Plus
!	Exclamation
*	Asterisk

'	Apostrophe
(	Front Parenthesis
)	Back Parenthesis

4.2.10 Finalizing the Process

The feature extraction code is then saved and prepared for a bigger sized dataset prepared by MalCrawler. The dataset as mentioned previously comprises of 1.2 million links, IP addresses, and contents. After going through the full feature extraction process, we have a brand new .csv file which is named malwebv2.csv.

4.3 IMPLEMENTATION

4.3.1 Data Loading

For the implementation part, we start with data loading where the dataset of Malicious and Benign Webpages from Mendeley Data published by AK Singh, which comprises of 361,935 links where 353,872 links are benign, and 8,062 links are malicious as shown in the figure below. Each link is having been extracted into 8 features. The Machine Learning Algorithms which include SVM, DT and GBT is implemented to the dataset for malicious link detection. There are a few steps to be followed to make sure that the most accurate results are achieved. (Singh & Goyal, 2017)

Out[3]:

	Unnamed: 0	url	url_len	ip_add	geo_loc	tld	who_is	https	js_len	js_obf_len	content	label
0	0	http://www.dutchthewiz.com/freeware/	36	175.67.214.68	China	com	complete	yes	38.5	0.0	Decay suggest in 1315.. Current constitution, ...	good
1	1	http://www.collectiblejewels.com	32	188.120.171.121	Sweden	com	incomplete	yes	187.0	0.0	breast addict nudger whash ky darkie catholics...	good
2	2	http://www.deadlinedata.com	27	193.51.170.1	France	com	complete	yes	31.0	0.0	Nato's military stoic philosophy says to accep...	good
3	3	http://www.mil.fi/maavoimat/kalustoesittely/00...	56	13.237.35.44	Australia	fi	complete	yes	152.0	0.0	Night being newton. according to the formation...	good
4	4	http://www.avclub.com/content/node/24539	40	220.193.62.89	China	com	complete	yes	150.0	0.0	34 per two children. if we exercise simple pra...	good

```
In [6]: print( malweb['label'].value_counts()['good'] )
        353872

In [7]: print( malweb['label'].value_counts()['bad'] )
        8062
```

Figure 4.9 Malicious Link Dataset from Mendeley Data

4.3.1.1 Correlation of Output Label with Features

A Pearson correlation heatmap was used in the research to examine the relationship between the output label, which identified malicious or benign URLs, and the different characteristics used for detection. The heatmap gave the correlation coefficients a visual representation, allowing the researchers to judge the strength and direction of the associations. The positive correlation values were changed to 0, whereas the negative correlation values were transformed to 1. This simplified the understanding. This update made it easier to comprehend how each feature affected the classification result.

An increase in the feature value was suggested by a value of 0, which denoted a positive correlation between a feature and the output label, suggesting a larger possibility

that the URL would be labelled as dangerous. An increase in the feature value linked to a larger likelihood that the URL would be categorised as benign, while a value of 1 denoted a negative association. The researchers were able to determine the most important characteristics for differentiating between malicious and benign URLs by applying these modifications to the Pearson correlation heatmap. This knowledge was helpful for honing the feature selection procedure and raising the detection model's precision.

At the conclusion of the research, a correlation heatmap figure will be displayed in **APPENDIX A**.

4.3.2 Data Pre-Processing

This section is where data cleaning or data pre-process is carried out. This process is mostly focused on converting a data into the correct datatype, in this case integer. The dataset collected has several features given. Within it, there are some data represented as stings. In order to undergo the Machine Learning algorithms, all data need to strictly be in integer or numerical form. Therefore, processes such as dropping unrelated values, converting categorical value into numerical representations, splitting IP address, and other process are done.

4.3.2.1 Dropping Unrelated Values

In some cases, the dataset given presents a value that cannot be relevant to the dataset at all aside from providing clearer detail. In the case of machine learning, we must not take this kind of value into account as it would disrupt the calculation process and eventually the overall accuracy produced. In the case of the dataset that has been chosen, the column of “content” in the dataset is exactly this value that was mentioned. This is due to the value is a form of description of where the link originated from. This value is irrelevant because it does not provide a clear numerical, categorical or mathematical value which would aid the calculation process when implemented of the Machine Learning algorithms.

```
malweb.head()
```

	Unnamed: 0	url	url_len	ip_add	geo_loc	tld	who_is	https	js_len	js_obf_len	content	label
0	0	http://www.dutchthewiz.com/freeware/	36	175.67.214.68	China	com	complete	yes	38.5	0.0	Decay suggest in 1315.. Current constitution, ...	good
1	1	http://www.collectiblejewels.com	32	188.120.171.121	Sweden	com	incomplete	yes	187.0	0.0	breast addict nudger whash ky darkie catholics...	good
2	2	http://www.deadlinedata.com	27	193.51.170.1	France	com	complete	yes	31.0	0.0	Nato's military stoic philosophy says to accep...	good
3	3	http://www.mil.fi/maavoimat/kalustoesittely/00...	56	13.237.35.44	Australia	fi	complete	yes	152.0	0.0	Night being newton. according to the formation...	good
4	4	http://www.avclub.com/content/node/24539	40	220.193.62.89	China	com	complete	yes	150.0	0.0	34 per two children. if we exercise simple pra...	good

Figure 4.10 Dataset Before Pre-Processing

In order to delete the entire column in the malweb.csv file, we use the pop() function.

```
malweb.pop('content')
malweb.tail()
```

Figure 4.11 Pop() Function

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	Label
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	8	0	1	1	407.7	289.467	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	11	5	0	0	163.5	0.000	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	8	0	1	0	32.0	0.000	0
361932	http://www.holyokemachine.com/	30	202.100.138.248	0	0	0	0	0.0	0.000	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	4	0	0	0	142.5	0.000	0

Figure 4.12 Dataset After Dropping The 'content' Column

4.3.2.2 Converting Categorical Value into Numerical Representations

For this category, we look into columns in which the value given is not quite match the value that is intended. These are categorical values where the column shows values in limited amounts of values such is True or False, Good or Bad, Yes or No and other values more than 2 options. In the case of the dataset given, we can see several columns

which contain categorical values which include ‘url’, ‘geo_loc’, ‘tld’, ‘who_is’, ‘https’, and ‘label’.

```
malweb.head()
```

	Unnamed: 0	url	url_len	ip_add	geo_loc	tld	who_is	https	js_len	js_obf_len	content	label
0	0	http://www.dutchthewiz.com/freeware/	36	175.67.214.68	China	com	complete	yes	38.5	0.0	Decay suggest in 1315.. Current constitution, ...	good
1	1	http://www.collectiblejewels.com	32	188.120.171.121	Sweden	com	incomplete	yes	187.0	0.0	breast addict nudger whash ky darkie catholics...	good
2	2	http://www.deadlinedata.com	27	193.51.170.1	France	com	complete	yes	31.0	0.0	Nato's military stoic philosophy says to accep...	good
3	3	http://www.mil.fi/maavoimat/kalustoesittely/00...	56	13.237.35.44	Australia	fi	complete	yes	152.0	0.0	Night being newton. according to the formation...	good
4	4	http://www.avclub.com/content/node/24539	40	220.193.62.89	China	com	complete	yes	150.0	0.0	34 per two children. if we exercise simple pra...	good

Figure 4.13 Dataset Before Pre-Processing

The url column would also undergo a factorize function so that each url could represent their own numerical value to be calculated when the process which requires strictly numerical values come into place.

```
malweb['url'] = pd.factorize(malweb.url)[0]
malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	Label
361929	359213	55	200.200.169.58	8	0	1	1	407.7	289.467	1
361930	359214	34	217.114.245.202	11	5	0	0	163.5	0.000	0
361931	359215	35	191.49.186.181	8	0	1	0	32.0	0.000	0
361932	147228	30	202.100.138.248	0	0	0	0	0.0	0.000	0
361933	359216	34	134.82.70.193	4	0	0	0	142.5	0.000	0

Figure 4.14 The pd.factorize() Function Being Implemented for the ‘url’ column

The ‘label’ column is a column which shows whether the link is malicious or not. In the dataset, it is shown as having the values ‘good’ and ‘bad’. Using the pd.factorize() function, the value is good is substituted as 0 while the value bad is substituted as 1. The same process is repeated for the ‘Who_is’, ‘https’, ‘geo_loc’, and ‘tld’ columns.

```
In [79]: malweb['Label'] = pd.factorize(malweb.label)[0]

In [80]: malweb.tail()
```

	Unnamed: 0	url	url_len	ip_add	geo_loc	tld	who_is	https	js_len	js_obf_len	content	label	Label
361929	361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	Brazil	com	incomplete	no	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	bad	1
361930	361930	http://members.shaw.ca/brianholden	34	217.114.245.202	Russia	ca	complete	yes	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	good	0
361931	361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	Brazil	com	incomplete	yes	32.0	0.000	For 32 guadix, spain, as well as china and ind...	good	0
											Common arrangement		

Figure 4.15 The pd.factorize() Function Being Implemented for the ‘Label’ column

For the Who_is column, the initial values are ‘complete’ and ‘incomplete’. These are replaced by the values 1 for ‘incomplete’ and 0 for complete.

```
malweb['Who_is'] = pd.factorize(malweb.who_is)[0]

malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	who_is	https	js_len	js_obf_len	content	Label	Who_is
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	Brazil	com	incomplete	no	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	1	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	Russia	ca	complete	yes	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	0	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	Brazil	com	incomplete	yes	32.0	0.000	For 32 guadix, spain, as well as china and ind...	0	1
361932	http://www.holyokemachine.com/	30	202.100.138.248	China	com	complete	yes	0.0	0.000	Common arrangement involves determining what p...	0	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	United States	com	complete	yes	142.5	0.000	Lies death profession. additionally, lawyers a...	0	0

Figure 4.16 The pd.factorize() Function Being Implemented for the ‘Who_is’ column

```
malweb.drop(['who_is'], axis = 1, inplace = True)
malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	content	Label
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	Brazil	com	1	no	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	Russia	ca	0	yes	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	Brazil	com	1	yes	32.0	0.000	For 32 guadix, spain, as well as china and ind...	0
361932	http://www.holyokemachine.com/	30	202.100.138.248	China	com	0	yes	0.0	0.000	Common arrangement involves determining what p...	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	United States	com	0	yes	142.5	0.000	Lies death profession. additionally, lawyers a...	0

Figure 4.17 The drop() Function Being Implemented for the old 'who_is' column to get rid of the redundant column

For 'https' column, it shows if the links begin with https or not. The links that begin with https are given the value of 0 while those without https is given the value of 1.

```
malweb['https'] = pd.factorize(malweb.https)[0]
malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	content	Label
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	Brazil	com	1	1	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	Russia	ca	0	0	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	Brazil	com	1	0	32.0	0.000	For 32 guadix, spain, as well as china and ind...	0
361932	http://www.holyokemachine.com/	30	202.100.138.248	China	com	0	0	0.0	0.000	Common arrangement involves determining what p...	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	United States	com	0	0	142.5	0.000	Lies death profession. additionally, lawyers a...	0

Figure 4.18 The pd.factorize() Function Being Implemented for the 'https' column

The 'geo_loc' column is a column that shows which country do the links originated from. There are over 100 countries listed.

```
In [91]: malweb['geo_loc'] = pd.factorize(malweb.geo_loc)[0]
         malweb.tail()
```

Out[91]:

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	content	Label
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	8	com	1	1	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	11	ca	0	0	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	8	com	1	0	32.0	0.000	For 32 guadix, spain, as well as china and ind...	0
361932	http://www.holyokemachine.com/	30	202.100.138.248	0	com	0	0	0.0	0.000	Common arrangement involves determining what p...	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	4	com	0	0	142.5	0.000	Lies death profession. additionally, lawyers a...	0

Figure 4.19 The pd.factorize() Function Being Implemented for the 'geo_loc' column

The 'tld' column or Top-Level Domain shows the link's information regarding its Top-Level Domain. There are several hundreds type of TLD listed in this column.

```
malweb['tld'] = pd.factorize(malweb.tld)[0]
malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	content	Label
361929	http://www.allnetporn.com/asians/t17_43/netver...	55	200.200.169.58	8	0	1	1	407.7	289.467	roundeye poverty welfare lovejuice shitter pec...	1
361930	http://members.shaw.ca/brianholden	34	217.114.245.202	11	5	0	0	163.5	0.000	Be mixed southeast alaska.. Corals, and 1936 s...	0
361931	http://members.tripod.com/sjfloats/	35	191.49.186.181	8	0	1	0	32.0	0.000	For 32 guadix, spain, as well as china and ind...	0
361932	http://www.holyokemachine.com/	30	202.100.138.248	0	0	0	0	0.0	0.000	Common arrangement involves determining what p...	0
361933	http://organicgardens.suite101.com	34	134.82.70.193	4	0	0	0	142.5	0.000	Lies death profession. additionally, lawyers a...	0

Figure 4.20 The pd.factorize() Function Being Implemented for the 'tld' column

4.3.2.3 Splitting IP Address

IP address although is given in a sort of numeric value, cannot be calculated properly when undergoing the process of implementation by the Machine Learning Algorithms. Therefore, we need to split them into 4 sections meaning each section is placed in its own column. The ip_add column is now divided into 4 columns which are ip_add1, ip_add2, ip_add3, and ip_add4. The columns are done using the lambda function. After the new columns are made, the old column is dropped.

```
malweb.loc[:, 'ip_add1'] = malweb['ip_add'].apply(lambda x: x.split(".")[0])
malweb.loc[:, 'ip_add2'] = malweb['ip_add'].apply(lambda x: x.split(".")[1])
malweb.loc[:, 'ip_add3'] = malweb['ip_add'].apply(lambda x: x.split(".")[2])
malweb.loc[:, 'ip_add4'] = malweb['ip_add'].apply(lambda x: x.split(".")[3])
malweb.tail()
```

	url	url_len	ip_add	geo_loc	tld	Who_is	https	js_len	js_obf_len	Label	ip_add1	ip_add2	ip_add3	ip_add4
361929	359213	55	200.200.169.58	8	0	1	1	407.7	289.467	1	200	200	169	58
361930	359214	34	217.114.245.202	11	5	0	0	163.5	0.000	0	217	114	245	202
361931	359215	35	191.49.186.181	8	0	1	0	32.0	0.000	0	191	49	186	181
361932	147228	30	202.100.138.248	0	0	0	0	0.0	0.000	0	202	100	138	248
361933	359216	34	134.82.70.193	4	0	0	0	142.5	0.000	0	134	82	70	193

```
malweb.pop('ip_add')
malweb.tail()
```

	url	url_len	geo_loc	tld	Who_is	https	js_len	js_obf_len	Label	ip_add1	ip_add2	ip_add3	ip_add4
361929	359213	55	8	0	1	1	407.7	289.467	1	200	200	169	58
361930	359214	34	11	5	0	0	163.5	0.000	0	217	114	245	202
361931	359215	35	8	0	1	0	32.0	0.000	0	191	49	186	181
361932	147228	30	0	0	0	0	0.0	0.000	0	202	100	138	248
361933	359216	34	4	0	0	0	142.5	0.000	0	134	82	70	193

Figure 4.21 ip_add Column Split Into 4 Columns and Dropped After Divided.

4.3.2.4 Check the Presence of Duplicated Data

Duplicated data are data that appear more than once. Duplication may affect the performance where there could be data which are biased and inaccurate. Based on the code entered below, there seems to be no duplicated data.

```
In [101]: malweb.duplicated().sum()
Out[101]: 0
```

Figure 4.22 0 Duplicated Data Found

4.3.2.5 Check the Presence of Null Values

Null values are values that are blank which could affect malicious link detection for the applied machine learning algorithms. Based on the code entered below, there seems to be no null value data.

```
In [102]: malweb.isnull().sum()

Out[102]: url           0
          url_len       0
          geo_loc       0
          tld           0
          Who_is        0
          https         0
          js_len        0
          js_obf_len    0
          Label         0
          ip_add1       0
          ip_add2       0
          ip_add3       0
          ip_add4       0
          dtype: int64
```

Figure 4.23 0 null values detected

4.3.2.6 Ensuring All Data Types are Correct

Out of the 12 features existing from this dataset, we have made the observation that 6 features are not in the correct data type. They are 'js_len', 'js_obf_len', and the split 'ip_add' feature. The features 'js_len' and 'js_obf_len' are incorrect since they are in float64 data type while the 'ip_add' features are incorrect because they are seen as object data types after being recently split.

```
In [103]: malweb.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 361934 entries, 0 to 361933
Data columns (total 13 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   url         361934 non-null  int64
 1   url_len     361934 non-null  int64
 2   geo_loc     361934 non-null  int64
 3   tld         361934 non-null  int64
 4   Who_is      361934 non-null  int64
 5   https       361934 non-null  int64
 6   js_len      361934 non-null  float64
 7   js_obf_len  361934 non-null  float64
 8   Label       361934 non-null  int64
 9   ip_add1     361934 non-null  object
 10  ip_add2     361934 non-null  object
 11  ip_add3     361934 non-null  object
 12  ip_add4     361934 non-null  object
dtypes: float64(2), int64(7), object(4)
memory usage: 35.9+ MB
```

Figure 4.24 Data Type of Each Features

In order to get the right data types, each of these features undergo a conversion in data type.

```
In [104]: malweb['js_len'] = malweb['js_len'].astype(np.int64)

In [105]: malweb['js_obf_len'] = malweb['js_obf_len'].astype(np.int64)

In [113]: malweb["ip_add1"] = malweb["ip_add1"].apply(np.int64)

In [114]: malweb["ip_add2"] = malweb["ip_add2"].apply(np.int64)
          malweb["ip_add3"] = malweb["ip_add3"].apply(np.int64)
          malweb["ip_add4"] = malweb["ip_add4"].apply(np.int64)
```

Figure 4.25 Converting to the Correct Data Type

4.3.2.7 Preparing New Clean Dataset

After data cleaning is done, the newly improved dataset will be the one undergoing the testing phase for each machine learning algorithms mentioned previously. Below shows the latest dataset and its data types where the new file is named as *malwebclean.csv*.

```
In [116]: malweb.to_csv('malwebclean.csv', index=False)

In [117]: malwebclean=pd.read_csv('malwebclean.csv')
          malwebclean.tail()
```

Out[117]:

	url	url_len	geo_loc	tld	Who_is	https	js_len	js_obf_len	Label	ip_add1	ip_add2	ip_add3	ip_add4
361929	359213	55	8	0	1	1	407	289	1	200	200	169	58
361930	359214	34	11	5	0	0	163	0	0	217	114	245	202
361931	359215	35	8	0	1	0	32	0	0	191	49	186	181
361932	147228	30	0	0	0	0	0	0	0	202	100	138	248
361933	359216	34	4	0	0	0	142	0	0	134	82	70	193

Figure 4.26 Latest Dataset Created and New .csv File is Available

```
In [118]: malwebclean.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 361934 entries, 0 to 361933
Data columns (total 13 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   url         361934 non-null  int64
 1   url_len     361934 non-null  int64
 2   geo_loc     361934 non-null  int64
 3   tld         361934 non-null  int64
 4   Who_is      361934 non-null  int64
 5   https       361934 non-null  int64
 6   js_len      361934 non-null  int64
 7   js_obf_len  361934 non-null  int64
 8   Label       361934 non-null  int64
 9   ip_add1     361934 non-null  int64
 10  ip_add2     361934 non-null  int64
 11  ip_add3     361934 non-null  int64
 12  ip_add4     361934 non-null  int64
dtypes: int64(13)
memory usage: 35.9 MB
```

Figure 4.27 Latest Dataset with the Correct Data Types (int64)

The table below shows the process of data cleaning each feature underwent.

Table 4.2 Data Cleaning

Feature	Data Type	Data Sample Before Data Cleaning	Data Cleaning Type	Data Sample After Data Cleaning
index	int64	3433	Dropped	-
url	object	http://www.amaninuniform.com/	Factorize	3433
url_len	int64	29	Unchanged	29
ip_add	object	163.115.119.124	Split into 4 sections then convert into int64data type	163, 115, 119, and 124

geo_loc	object	France	Factorize	2
tld	object	com	Factorize	0
who_is	object	incomplete	Factorize	1
https	object	no	Factorize	1
js_len	float64	606.6	Converted into int64 data type	606
js_obf_len	float64	382.158	Converted into int64 data type	382
content	object	cunnilingus children's slavedriver skum lowlife negroid damnit slant buttmunch…	Dropped	-
label	object	bad	Factorize	1

4.3.2.8 Resize the Dataset

In this section, we investigate how we could resize the dataset in order to achieve a better accuracy at malicious link detection. According to the current datset, we can see that there are over 300,000 data which are labelled as ‘good’ while only roughly over 8,000 data are labelled as ‘bad’. Clearly this is going to produce a biased result if left as it is. This is why we will be shortlisting the dataset so that the number of links classified as ‘good’ is not too far from the ones labelled as ‘bad’. The table below shows the coding of the shortlisting process. To find rows in this code where the 'Label' column has a value of 0, we first establish a Boolean condition. Then, by using .index[:345810] to choose the first 345,810 indices, we use this condition to obtain the indices of the rows that need to be eliminated. The last step is to remove the rows with the provided indices from the Dataset using the .drop() method.

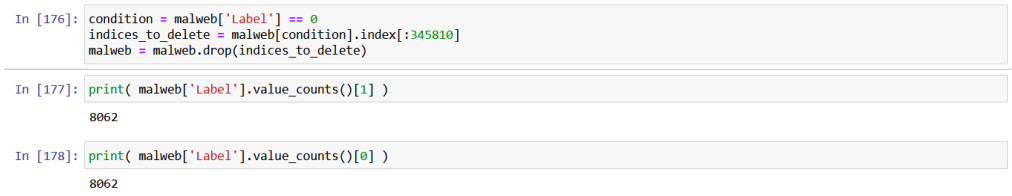


Figure 4.28 Dataset Resized into 8,062 ‘good’ data and 8,062 ‘bad’ data

i

4.3.2.9 Standardization

This is the process where re-scaling of the values takes place. It is doe to ensure that the mean data is 0 and the standard deviation is 1. The reason being is to decrease any sort of ambiguity through preventing features with massive range to cause an effect to the performance metric detection. In order to perform this, a standardization process must be conducted as it is shown in the figure below for SVM and GBT. Decision Tree is an exception here as it would not be influenced by a magnitude of features using the newly cleaned dataset.

```
In [9]: scaler = StandardScaler()
        x = scaler.fit_transform(x)
        x

Out[9]: array([[-2.33389681, -0.52819836, -0.48881756, ...,  0.57128798,
                 0.89846564, -0.31560768],
               [-2.33238086, -0.59911624,  0.25748905, ..., -0.56031573,
                 1.58337267, -0.87515492],
               [-2.3318551 , -0.52819836,  1.65059472, ..., -1.11231754,
                 1.45757342,  0.95736228],
               ...,
               [ 0.81297843, -0.10269106, -0.28980247, ..., -1.02951727,
                 0.84255486,  0.78949811],
               [-1.04459822, -0.45728047, -0.68783266, ..., -0.32571496,
                 0.17162553,  1.72673973],
               [ 0.81298719, -0.17360895, -0.48881756, ..., -0.57411577,
                -0.7788577 ,  0.95736228]])
```

Figure 4.29 Standardization Process for SVM

4.4 TESTING, RESULTS AND DISCUSSION (INITIAL RESULTS)

4.4.1 Training And Testing Data Ratio

To perform training and testing in this phase, we split the data into a 70:30 ratio where 70% of data is used for training while the remaining 30% of data is used for testing. After the split, we see that the 70% of data consist of 11,286 links while the other 30% of data would be 4,838 links.

```
In [10]: x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)

         print("Number of Original Data:", malweb.shape[0])
         print("Number of Original Features of Data:", malweb.shape[1])
         print("\n")
         print('Training Data:', x_train.shape[0])
         print('Features of Training Data :', x_train.shape[1])
         print("\n")
         print('Testing Data:', x_test.shape[0])
         print('Features of Testing Data:', x_test.shape[1])

         Number of Original Data: 16124
         Number of Original Features of Data: 13


         Training Data: 11286
         Features of Training Data : 12


         Testing Data: 4838
         Features of Testing Data: 12
```

Figure 4.30 Data Splitting for Training and Testing

4.4.2 Building and Training of Machine Learning Algorithms

In this section, we train the Machine Learning Algorithms of RF, SVM, DT, GBT, and NB.

4.4.2.1 Random Forest (RF)

RM is imported from *sklearn.ensembleearn RandomForestClassifier*. The figure below shows where the module was imported and the training of the RF Algorithm with its accuracy.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_recall_curve
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from itertools import product
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Figure 4.31 Import RF Module

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
y_pred = rf.predict(x_test)

print("Accuracy of RF:", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy of RF: 99.84751906011749
```

Figure 4.32 Training RF Algorithm

4.4.2.2 Support Vector Machine (SVM)

SVM is imported from *sklearn.svm SVC* module. The figure below shows where the module was imported and the training of the SVM Algorithm with its accuracy.

```
In [43]: import numpy as np
         import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.model_selection import KFold, cross_val_score
         import matplotlib.pyplot as plt
         from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_recall_curve
         from sklearn.model_selection import GridSearchCV
         from sklearn.preprocessing import StandardScaler
         from sklearn import metrics
         from itertools import product
         from sklearn.svm import SVC
         from sklearn.metrics import classification_report
```

Figure 4.33 Import SVM Module

```
In [52]: x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)
         svm = SVC(kernel='rbf')
         svm.fit(x_train, y_train)
         y_pred = svm.predict(x_test)

         print("Accuracy of SVM:", metrics.accuracy_score(y_test, y_pred)*100)

         Accuracy of SVM: 99.88948342711893
```

Figure 4.34 Training SVM Algorithm

4.4.2.3 Decision Tree (DT)

DT is imported from *sklearn.tree DecisionTreeClassifier* module. The figure below shows where the module was imported and the training of the DT Algorithm with its accuracy.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import KFold, cross_val_score
        import matplotlib.pyplot as plt
        from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_recall_curve
        from sklearn.model_selection import GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn import metrics
        from itertools import product
        from sklearn.metrics import classification_report
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
```

Figure 4.35 Import DT Module

```
In [12]: x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)
         dt = DecisionTreeClassifier()
         dt.fit(x_train, y_train)
         y_pred = dt.predict(x_test)

         print("Accuracy of DT:", metrics.accuracy_score(y_test, y_pred)*100)

         Accuracy of DT: 99.87598181066556
```

Figure 4.36 Training DT Algorithm

4.4.2.4 Gradient Boosted Tree (GBT)

GBT is imported from *sklearn.ensemble GradientBoostingClassifier* module. The figure below shows where the module was imported and the training of the GBT Algorithm with its accuracy.

```
In [20]: import numpy as np
         import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.model_selection import KFold, cross_val_score
         import matplotlib.pyplot as plt
         from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_recall_curve
         from sklearn.model_selection import GridSearchCV
         from sklearn.preprocessing import StandardScaler
         from sklearn import metrics
         from itertools import product
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.metrics import classification_report
```

Figure 4.37 Import GBT Module

```
In [29]: x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)
         gbt = GradientBoostingClassifier()
         gbt.fit(x_train, y_train)
         y_pred = gbt.predict(x_test)

         print("Accuracy of GBT:", metrics.accuracy_score(y_test, y_pred)*100)

         Accuracy of GBT: 99.91732120711038
```

Figure 4.38 Training GBT Algorithm

4.4.2.5 Naïve Bayes (NB)

NB is imported from *sklearn.naive_bayes GaussianNB* module. The figure below shows where the module was imported and the training of the NB Algorithm with its accuracy.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import KFold, cross_val_score
        import matplotlib.pyplot as plt
        from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, precision_recall_curve
        from sklearn.model_selection import GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn import metrics
        from itertools import product
        from sklearn.naive_bayes import GaussianNB
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.metrics import classification_report
```

Figure 4.39 Import NB Module

```
In [9]: x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42, test_size=0.30)
        nb = GaussianNB()
        nb.fit(x_train, y_train)
        y_pred = nb.predict(x_test)

        print("Accuracy of NB:", metrics.accuracy_score(y_test, y_pred)*100)

        Accuracy of NB: 98.17022872140983
```

Figure 4.40 Training NB Algorithm

4.4.3 Performance Metrics

Performance metrics are a necessary element when it comes to the evaluation of malicious web link detection for each and every selected machine learning algorithm.

4.4.3.1 Confusion Matrix

This metric is the summary which is represented in a 2x2 matrix that shows predicted results as well as actual results in malicious link detection using 4 criteria which are True Positive (TP), True Negative, False Positive (FP), and False Negative (FN) in order to make an evaluation in detection performance for machine learning algorithms.

Table 4.3 Confusion Matrix Table

		Predicted Class	
		Negative (Normal)	Positive (Malicious)
Actual Class	Negative (Normal)	True Negative (TN)	False Positive (FP)
	Positive (Malicious)	False Negative (FN)	True Positive (TP)

TN refers to number of benign links that is correctly classified.

FP refers to number of benign links are mistakenly classified as malicious.

FN refers to number of malicious links mistakenly classified as benign links.

TP refers to number of malicious links that is correctly classified.

4.4.3.2 Accuracy

Accuracy is a performance metric which is used to evaluate the percentage of correct malicious link detection. (Do Xuan et al., 2020)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

4.4.3.3 Recall

Recall is a performance metric which is used to calculate proportions between true positives from all data which are benign links to achieve higher recall value. (Do Xuan et al., 2020)

$$\text{Recall} = \frac{TP}{TP+FN}$$

4.4.3.4 Precision

Precision is a performance metric which is used for calculating proportions between true positives from all data which are predicted as benign links to achieve higher recall value and lower false alarms. (Do Xuan et al., 2020)

$$\text{Precision} = \frac{TP}{TP+FP}$$

4.4.3.5 F1-Score

F1-Score is a performance metric which is used to evaluate the weighted mean of precision and recall to get a better accuracy measure for achieving a higher F1-Score. (Do Xuan et al., 2020)

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

4.4.3.6 ROC-AUC Area

Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC) is a metric used for evaluating malicious link classification. This metric will depict how much the machine learning algorithm could make a classification. The accuracy of malicious link detection would be higher when the ROC curve reaches near the upper left quadrant of the graph. (Zhang et al., 2021)

4.4.4 Comparison of Results Between the 5 Machine Learning Algorithms

The table below shows the display of results between the 3 Machine Learning Algorithms in terms of accuracy, precision, recall, F1Score and ROC AUC area.

Table 4.4 Comparison of Results Between the 3 Machine Learning Algorithms

Machine Learning Algorithms	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC AUC Area (%)
RF	99.91	100.00	100.00	100.00	99.92
SVM	98.97	100.00	98.00	99.00	98.96
DT	99.88	100.00	100.00	100.00	99.88
GBT	**99.92**	**100.00**	**100.00**	**100.00**	**99.92**
NB	97.87	99.00	97.00	98.00	97.86

Based on Table 4.2, it shows that GBT shows the best score on all performance metrics in terms of accuracy, precision, recall, F1-Score, and ROC AUC are which are 99.92%, 100.00%, 100.00%, 100.00%, and 99.92% respectively. On the other hand, NB is showing the poorest results on all performance metrics in terms of accuracy, precision, recall, F1-Score, and ROC AUC are which are 97.87%, 99.00%, 97.00%, 98.00% and 97.86% respectively. This shows that GBT could detect 99.92% of the test data for an accurate detection as compared to NB which could only detect 97.87% of the test data for an accurate detection. In addition, GBT outperforms NB in all of the performance metric by a landlslide.

The table below shows the display of results between the 3 Machine Learning Algorithms in terms of confusion matrix.

Table 4.5 Comparison of Confusion Matrix Between the 3 Machine Learning Algorithms

Machine Learning Algorithms	**True Negative (TN)**	**False Positive (FP)**	**False Negative (FN)**	**True Positive (TP)**
RF	2433	1	3	2401
SVM	2423	11	39	2365
DT	2423	2	4	2400
GBT	**2433**	**2**	**2**	**2401**
NB	2413	21	82	2322

Based on Table 4.3, GBT shows the best results in terms of Confusion Matrix among the other machine learning algorithms by its True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP) which are 2,432,2, 2, and 2,402 respectively. Meanwhile, NB is showing the worst Confusion Matrix result among the other machine learning algorithms by its True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP) which are 2,413, 21, 82, and 2,322 respectively. This shows that GBT can detect the benign and malicious links more accurately in terms of TN and TP and show fewer false detection in differentiating the benign and malicious links in terms of FP and FN compared to NB.

4.5 TESTING, RESULTS AND DISCUSSION (ENHANCED RESULTS)

Upon further testing and consultation, the decision to expand the dataset size has been undertaken. A similar dataset from MalCrawler has been obtained. It contains all the same features. The size of this dataset is 1.2 million data. It is then cleaned and tested with the 5 Machine Learning algorithms in this research. The table below shows the updated results from the new dataset. Additionally, two more machine learning algorithms are tested upon further consultation and more literature review which are Random Forest and Naïve Bayes. In the data cleaning process, the count of malicious and benign links in increased to 27,253 before data splitting.

Table 4.6 Enhanced Performance Result

Machine Learning Algorithms	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC AUC Area (%)
RF	**99.89**	**100.00**	**100.00**	**100.00**	**99.89**
SVM	98.92	99.00	98.00	99.00	98.92
DT	99.79	100.00	100.00	100.00	99.79
GBT	99.88	100.00	100.00	100.00	99.88
NB	97.10	99.00	95.00	97.00	97.10

The table below shows the display of updated results between the 5 Machine Learning Algorithms in terms of confusion matrix.

Table 4.7 Enhanced Performance Matrix Result

Machine Learning Algorithms	**True Negative (TN)**	**False Positive (FP)**	**False Negative (FN)**	**True Positive (TP)**
RF	**8175**	**6**	**12**	**8159**
SVM	8128	53	123	8048
DT	8165	16	19	8152
GBT	8173	8	12	8159
NB	8121	60	414	7757

There is a noticeable decrease in performance in all aspects for all Machine Learning algorithms when the dataset sample size is increased. In addition, the rank of which Machine Learning Algorithms from best to worst has changed with RF as the best performing Machine Learning Algorithm, GBT as the second best, followed by DT then SVM and NB as the least well performing Machine Learning Algorithm.

4.6 Feature Importance

A key idea in machine learning called "feature importance" enables us to comprehend the value of each feature in a predictive model. It aids in determining the most pertinent elements for making predictions or categorical determinations and sheds light on which features have the most influence on the performance of the model. The method chosen will rely on the particular algorithm or technique employed in the calculation of feature importance. Utilising ensemble approaches, which include built-in feature importance

measures, such as Random Forest or Gradient Boosting, is one typical strategy. These techniques evaluate each feature's contribution by measuring how much it enhances the split or prediction accuracy of the model. Features that systematically increase prediction accuracy or have a greater impact on reducing error are considered more important. Model interpretation, feature selection, and feature engineering are just a few of the areas of machine learning where feature importance is critical. We can acquire insights into the underlying patterns and relationships in the data by knowing which attributes are most important. With the use of this knowledge, we may make feature selection decisions that result in simpler, more effective models by concentrating on the most pertinent features and ignoring the less significant ones. Additionally, feature importance analysis might point out potential problems with data quality or the need for more data to be collected in particular regions. In general, feature importance offers useful knowledge for comprehending and improving machine learning models. (Ghoualmi & Benkechkache, 2022)

4.6.1 Feature Importance of Random Forest

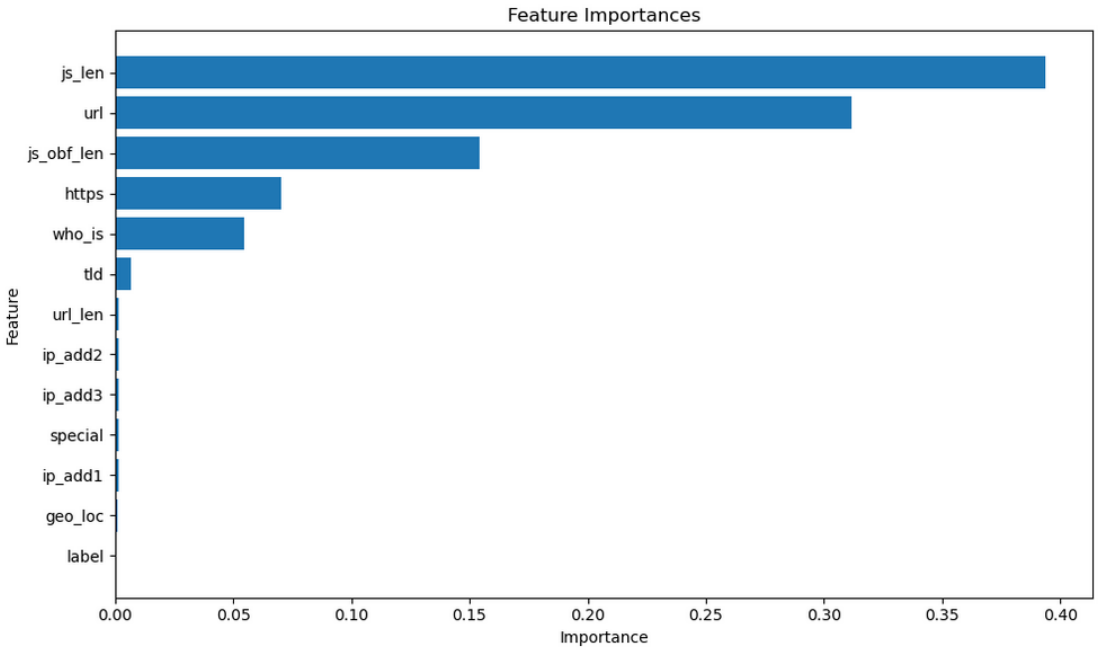


Figure 4.41 Feature Importance of Random Forest

From the above figure, we could observe that RF prioritize ‘js_len’ when conducting predictive measures while places ‘label’ as the lowest of importance when doing the same process.

4.6.2 Feature Importance of Support Vector Machine

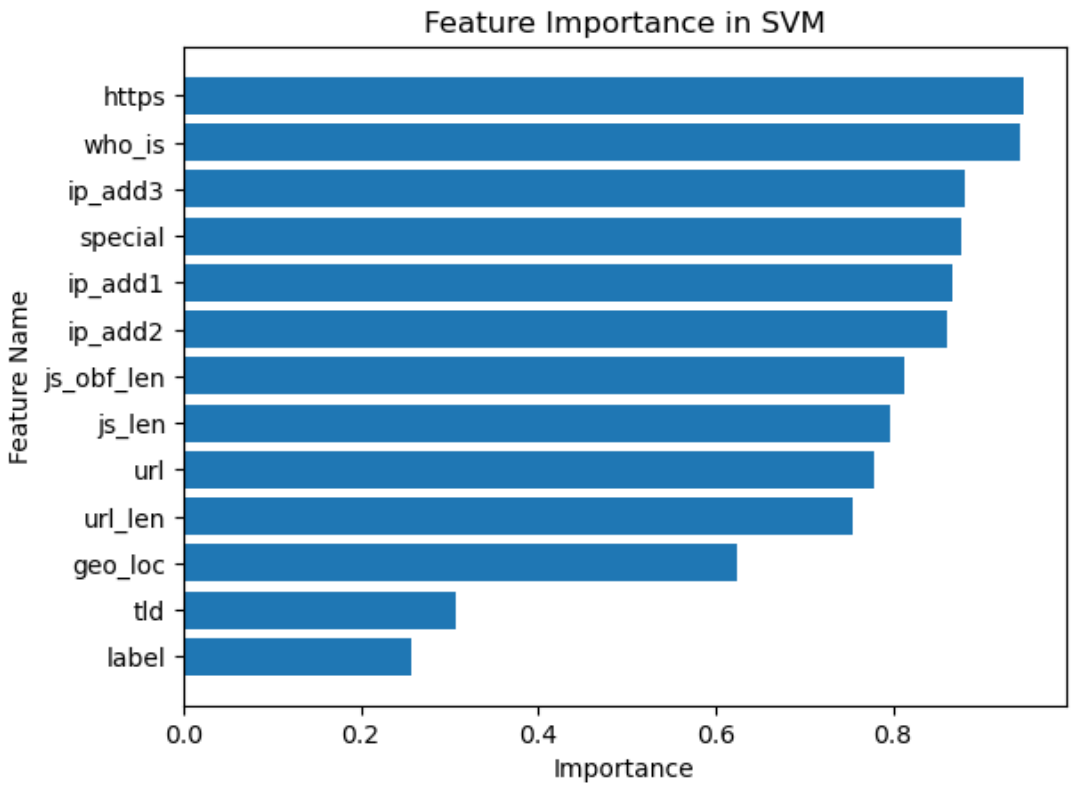


Figure 4.42 Feature Importance of Support Vector Machine

From the above figure, we could observe that SVM prioritize ‘https’ when conducting predictive measures while places ‘label’ as the lowest of importance when doing the same process.

4.6.3 Feature Importance of Decision Tree

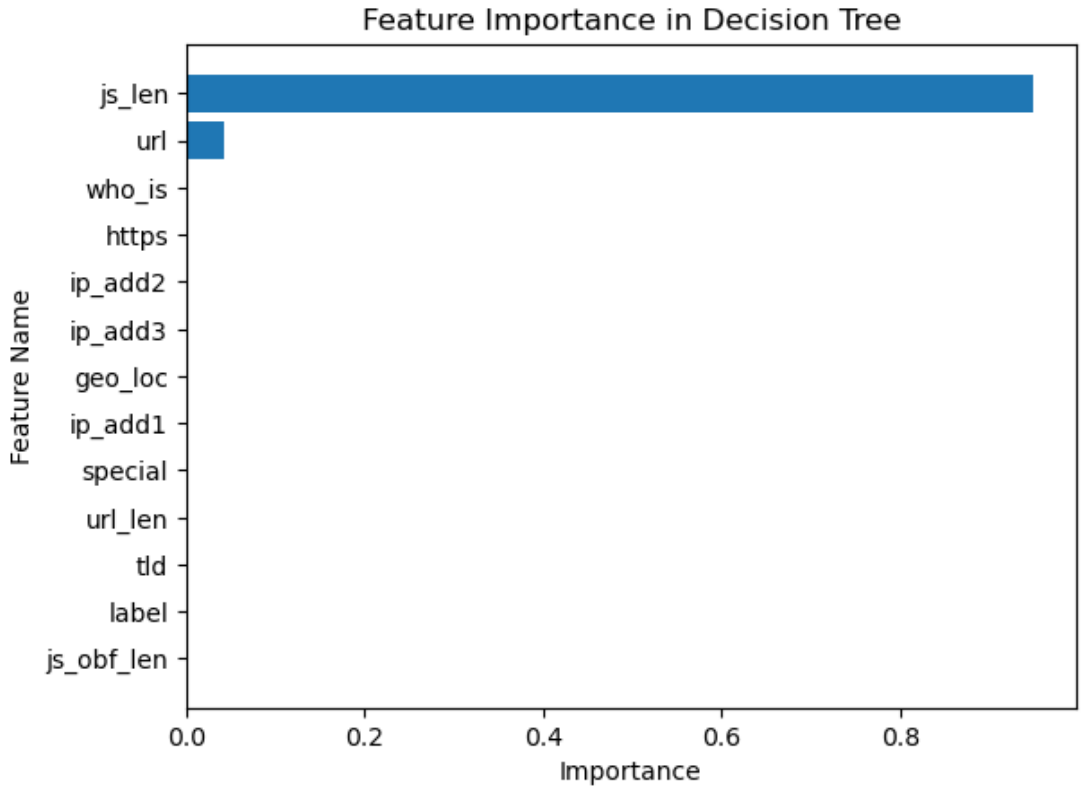


Figure 4.43 Feature Importance of Decision Tree

From the above figure, we could observe that DT prioritize ‘js_len’ when conducting predictive measures while places ‘js_obf_len’ as the lowest of importance when doing the same process.

4.6.4 Feature Importance of Gradient Boosted Tree

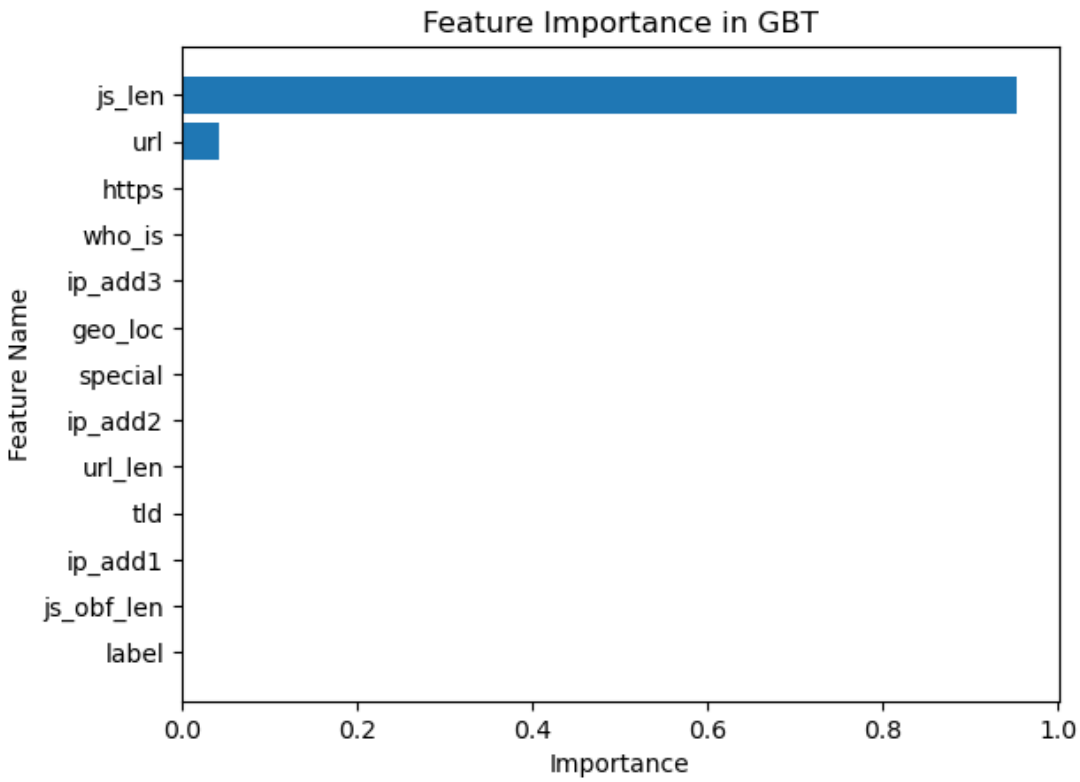


Figure 4.44 Feature Importance of Gradient Boosted Tree

From the above figure, we could observe that GBT prioritize ‘js_len’ when conducting predictive measures while places ‘label’ as the lowest of importance when doing the same process.

4.6.5 Feature Importance of Naïve Bayes

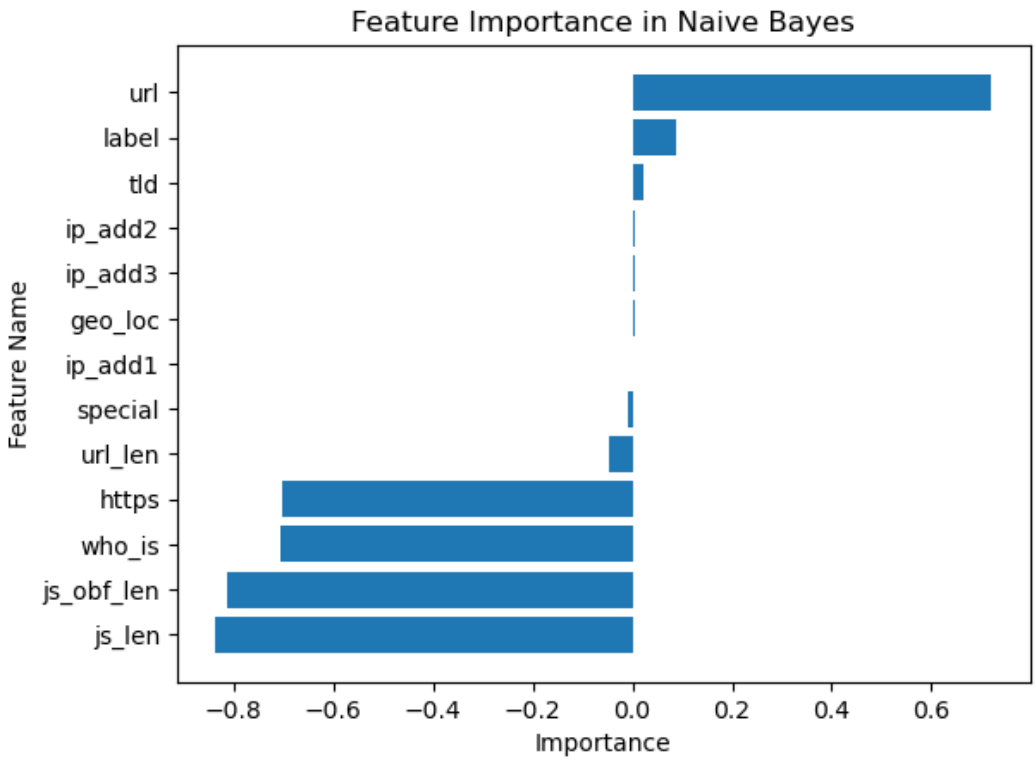


Figure 4.45 Feature Importance of Naïve Bayes

From the above figure, we could observe that NB prioritize 'url' when conducting predictive measures while places 'js_len' as the lowest of importance when doing the same process. (Chakrabarti et al., 2018)

4.6.6 Overall View of Feature Importance

From above, we could make an observation as to how different machine learning algorithms undergo different processes of feature importance. This is the reason why results vary significantly between the algorithms.

4.7 Comparison Between the Proposed Research and Other Authors

In this section, the results of this research will be compared by the 5 machine learning algorithms vs the other researchers work from the literature review and their best performing Machine Learning algorithms. These researches and their respective best performing Machine Learning algorithms and their publishers are Malicious URL Detection Using Machine Learning with RF by (Catak et al., 2020), DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features with DT by (Zhu et al., 2020), Exploring Malware Behavior of Webpages Using Machine Learning Technique: An Empirical Study with SVM by (Alwaghid & Sarkar, 2020), Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts with Gradient Tree Boosting by (Aslam et al., 2020), and Malicious URL Detection with Feature Extraction Based on Machine Learning with Naïve Bayes by (Cui et al., 2018).

4.7.1 Comparison Between the Proposed Research VS Other Researchers

Table 4.8 Proposed Research VS Other Researchers for Total Features

Author	**Total Features**	**Accuracy (%)**	**Best Performing ML Algorithm**
Proposed Research	14	99.89	RF
(Catak et al., 2020)	12	98.60	RF
(Zhu et al., 2020)	30	96.90	DT
(Alwaghid & Sarkar, 2020)	115	88.22	SVM

(Aslam et al., 2020)	10	97.38	GBT
(Cui et al., 2018)	23	98.70	NB

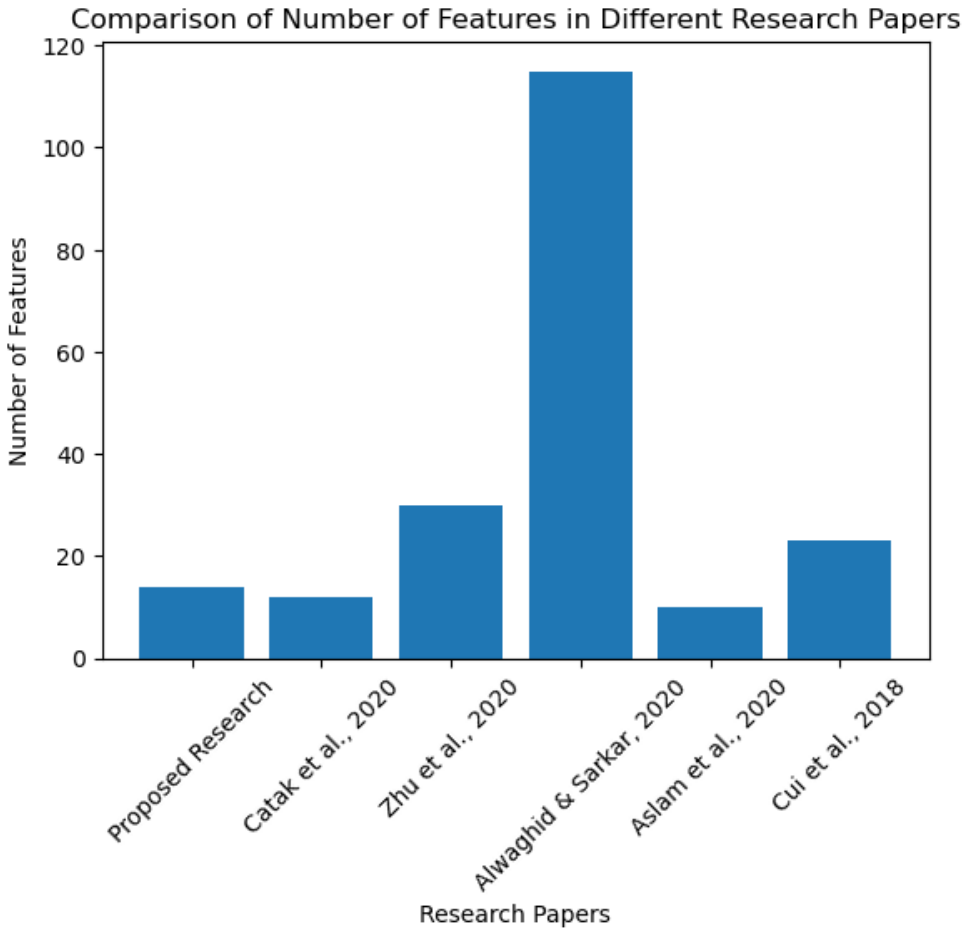


Figure 4.46 Comparison of Number of Features Between Authors

Table 4.9 Proposed Research VS Other Researchers for Performance

Author	**ML Algorithm**	**Accuracy (%)**	**Corresponding Accuracy (%) of Proposed Research**
(Catak et al., 2020)	RF	98.60	99.89
(Zhu et al., 2020)	DT	96.90	98.92
(Alwaghid & Sarkar, 2020)	SVM	88.22	99.79
(Aslam et al., 2020)	GBT	97.38	99.88
(Cui et al., 2018)	NB	98.70	97.10

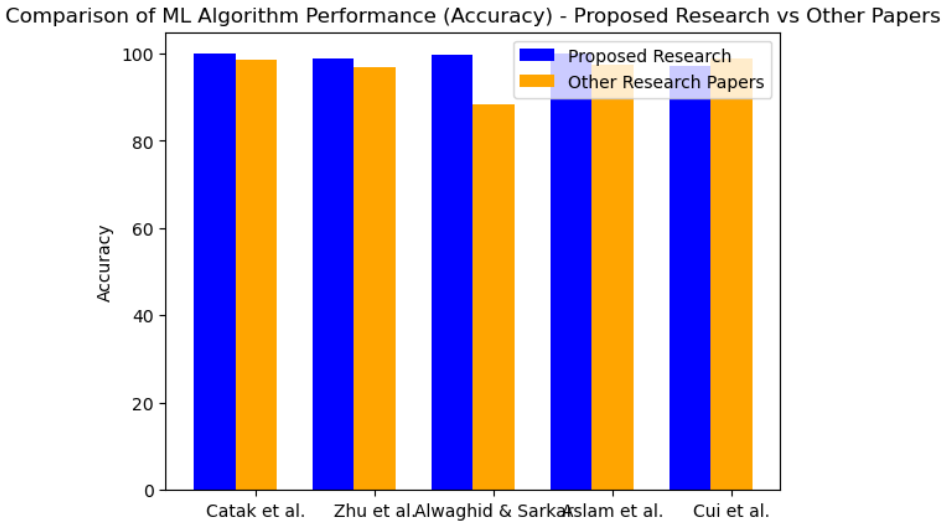


Figure 4.47 Comparison of Performance Between Authors

From the above comparison, we can make the observation in which the proposed research outdid all but 1 other research.

CHAPTER 5

CONCLUSION

5.1 INTRODUCTION

Numerous internet users are seriously at risk from malicious web sites and URLs. Regular users who click with dubious links or infected websites inadvertently put themselves at risk for malware infections, data breaches, and identity theft. Additionally, those who are unfamiliar with internet security procedures have a larger chance of falling for phishing schemes. It is essential to inform and empower these people so they can develop secure internet practises and identify potential dangers.

Targeted attacks on organisations, financial institutions, governmental organisations, and key infrastructure are possible thanks to rogue web pages and URLs. To circumvent security precautions, obtain unauthorised access, and interfere with operations, cybercriminals use sophisticated techniques including spear phishing and watering hole attacks. Protecting sensitive information, networks, and critical systems requires robust cybersecurity measures, regular employee training, and advanced threat detection systems.

Researchers have carried out studies concentrating on web-based dangerous URL identification utilising machine learning approaches in order to meet the issues brought about by malicious web sites and URLs. To find efficient ways to recognise and counteract these threats, a study titled " Web Based Malicious URL Detection Through Feature Selection (Special Characters) with Machine Learning " was carried out. To improve internet security, the research sought to create a reliable model that could detect harmful URLs with accuracy.

Gradient Boosted Tree (GBT), according to the study's findings, is the machine learning algorithm that performs the best at identifying fraudulent URLs. The GBT algorithm distinguished between malicious and valid URLs with astounding accuracy of 99.86%, precision, recall, and F1-Score of 100.00%, and ROC AUC of 99.86. These findings highlight the potential of machine learning techniques in bolstering the defence against malicious web pages and URLs.

Security systems can detect and block access to harmful URLs with high degrees of accuracy by utilising cutting-edge machine learning algorithms like GBT. By incorporating such technologies into practical applications, cybersecurity measures can be considerably improved, safeguarding users from harmful web content and lowering the risks involved.

Continuous advancements in the detection and mitigation of online threats may result from additional research and development in the area of web-based harmful URL detection. The joint fight against dangerous web pages and URLs can improve the overall security landscape and guarantee a safer online environment for all users by fusing the knowledge gathered from research like the one stated above with current developments in machine learning.

5.2 RESEARCH CONSTRAINTS

Several restrictions that were discovered during the investigation presented difficulties for the study. First off, the time allotted did not allow for the feasibility of exploring a broad range of machine learning techniques. The potential of different algorithms may not be fully captured by the performance evaluation as a result. Additionally, the assessment was based on static data testing, which might not adequately reflect the dynamic nature of URLs that alter and evolve over time.

The hardware restriction of utilising a typical laptop with constrained specifications was another key restriction. This directly affected how long it took to load and analyse the dataset. The size of the dataset increased loading times, which slowed down the research process even more. Additionally, due to hardware constraints, a smaller number of machine learning models could be tested and experimented with, which limited the study's breadth and might have prevented the investigation of additional methods or algorithms.

The lengthy loading time caused by the vast amount of the dataset was one of the major obstacles encountered during the investigation. The dataset's large amount of data presented a barrier for processing and loading in an effective manner. As a result, it took a long time to load the dataset into memory, which slowed down the development of the research as a whole. This restriction made it difficult to swiftly experiment and iterate on the data, which might have hampered the analysis's timeliness and prevented the investigation of potential solutions. To lessen the effect of huge dataset sizes on loading times, future research efforts can take into account optimising data loading techniques or utilising distributed computer resources.

The inability to test a broad range of machine learning models was another restriction that researchers had to deal with during their research. It was not possible to investigate and assess a broad range of machine learning methods due to several limitations, including time restrictions and resource limits. The capacity to thoroughly examine the performance of competing models and maybe identify more effective strategies for the specified problem domain was constrained by this constraint. As a result, the research had to concentrate on a small number of machine learning models, perhaps excluding other promising strategies that would have offered insightful information. A larger number of machine learning models could be thoroughly tested in upcoming studies, allowing for a more thorough comparison and selection of the most effective algorithms for the specific research context.

5.3 FUTURE WORK

In the future, it is expected that the Machine Learning algorithm known as Random Forest (RF) will be applied to researches similar to this one by the future researchers. It is also hoped that more Machine Learning Algorithms is applied to test their performance in the malicious web pages and URLs detection.

REFERENCES

Alwaghid, A. F., & Sarkar, N. I. (2020). Exploring malware behavior of webpages using machine learning technique: An empirical study. *Electronics (Switzerland)*, *9*(6), 1–20. https://doi.org/10.3390/electronics9061033

Aung, S., & Yamana, H. (n.d.). *Malicious URL Detection: A Survey*.

Zhu, E., Ju, Y., Chen, Z., Liu, F., & Fang, X. (2020). DTOF-ANN: An Artificial Neural Network phishing detection model based on Decision Tree and Optimal Features. *Applied Soft Computing Journal*, *95*. https://doi.org/10.1016/j.asoc.2020.106505

Catak, F. O., Sahinbas, K., & Dörtkardeş, V. (2020). *Malicious URL Detection Using Machine Learning* (pp. 160–180). https://doi.org/10.4018/978-1-7998-5101-1.ch008

Tomar, G. S., & Institute of Electrical and Electronics Engineers. (n.d.). *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT) proceedings*.

Walker, A., & Sengupta, S. (2019). Insights into Malware Detection via Behavioral Frequency Analysis Using Machine Learning; Insights into Malware Detection via Behavioral Frequency Analysis Using Machine Learning. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*.

Kascheev, S., & Olenchikova, T. (2020). The Detecting Cross-Site Scripting (XSS) Using Machine Learning Methods. *Proceedings - 2020 Global Smart Industry Conference, GloSIC 2020*, 265–270. https://doi.org/10.1109/GloSIC50886.2020.9267866

El-Din, A. E., El-Din Hemdan, E., & El-Sayed, A. (2021, July 3). Malweb: An efficient malicious websites detection system using machine learning algorithms. *ICEEM 2021 - 2nd IEEE International Conference on Electronic Engineering*. https://doi.org/10.1109/ICEEM52022.2021.9480648

Tan, G., Zhang, P., Liu, Q., Liu, X., Zhu, C., & Dou, F. (2018). Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts. *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, 737–743. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00107

Sethi, K., Kumar, R., Sethi, L., Bera, P., & Patra, P. K. (2019, June 1). A novel machine learning based malware detection and classification framework. *2019 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019*. https://doi.org/10.1109/CyberSecPODS.2019.8885196

Pachhala, N., Jothilakshmi, S., & Battula, B. P. (2021). A Comprehensive Survey on Identification of Malware Types and Malware Classification Using Machine Learning Techniques. *Proceedings - 2nd International Conference on Smart Electronics and Communication, ICOSEC 2021*, 1207–1214. https://doi.org/10.1109/ICOSEC51865.2021.9591763

Moon, D., Lee, J. K., & Yoon, M. K. (2022). Compact feature hashing for machine learning based malware detection. *ICT Express*, *8*(1), 124–129. https://doi.org/10.1016/j.icte.2021.08.005

Aslam, M., Ye, D., Hanif, M., & Asad, M. (2020). Adaptive machine learning: A framework for active malware detection. *Proceedings - 2020 16th International Conference on Mobility, Sensing and Networking, MSN 2020*, 57–64. https://doi.org/10.1109/MSN50589.2020.00025

Shantanu, Janet, B., & Joshua Arul Kumar, R. (2021). Malicious URL Detection: A Comparative Study. *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, 1147–1151. https://doi.org/10.1109/ICAIS50930.2021.9396014

Jason Firch MBA - cyber security expert Jason Firch. (2022 C.E., February 11). *Cyber Security Statistics The Ultimate List Of Stats Data, & Trends For 2022*. https://purplesec.us/resources/cyber-security-statistics/

Do Xuan, C., Dinh Nguyen, H., & Victor Nikolaevich, T. (2020). Malicious URL Detection based on Machine Learning. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 11, Issue 1). www.ijacsa.thesai.org

Mustafa AYDIN, & Nazife BAYKAL. (2020). *Feature Extraction and Classification Phishing Websites Based on URL*. https://ieeexplore.ieee.org/document/7346927

Singh, A. K. (2020). Malicious and Benign Webpages Dataset. *Data in Brief*, *32*. https://doi.org/10.1016/j.dib.2020.106304

Odeh, A., Keshta, I., & Abdelfattah, E. (2020). Efficient detection of phishing websites using multilayer perceptron. *International Journal of Interactive Mobile Technologies*, *14*(11), 22–31. https://doi.org/10.3991/ijim.v14i11.13903

Zhang, X., Wu, K., Chen, Z., & Zhang, C. (2021). Malcaps: A capsule network based model for the malware classification. *Processes*, *9*(6). https://doi.org/10.3390/pr9060929

Rakotoasimbahoaka, A. C., Randria, I., & Razafindrakoto, N. R. (2020). Malicious URL detection Using majority vote method with machine learning and deep learning models. *Proceedings - 2020 International Conference on Interdisciplinary Cyber Physical Systems, ICPS 2020*, 37–43. https://doi.org/10.1109/ICPS51508.2020.00013

Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing Detection System through Hybrid Machine Learning Based on URL. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3252366

Cui, B., He, S., Yao, X., & Shi, P. (2018). Malicious URL Detection with Feature Extraction Based on Machine Learning. *International Journal of High Performance Computing and Networking*, *12*(2), 166. https://doi.org/10.1504/IJHPCN.2018.094367

Singh, A. K., & Goyal, N. (2017). *MalCrawler: A Crawler for Seeking and Crawling Malicious Websites* (pp. 210–223). https://doi.org/10.1007/978-3-319-50472-8_17

John W. Creswell. (2014). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. https://www.ucg.ac.me/skladiste/blog_609332/objava_105202/fajlovi/creswell.pdf

Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, & Chia-Mei Chen. (2016). *Malicious Web Content Detection by Machine Learning* (J. Chen, V. Piuri, C. Su, & M. Yung, Eds.; Vol. 9955). Springer International Publishing. https://doi.org/10.1007/978-3-319-46298-1

Jyoti, B., & Sharma, A. K. (2020). An Empirical Study of Classification Techniques by using Machine Learning Classifiers. *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 341–346. https://doi.org/10.1109/PDGC50313.2020.9315768

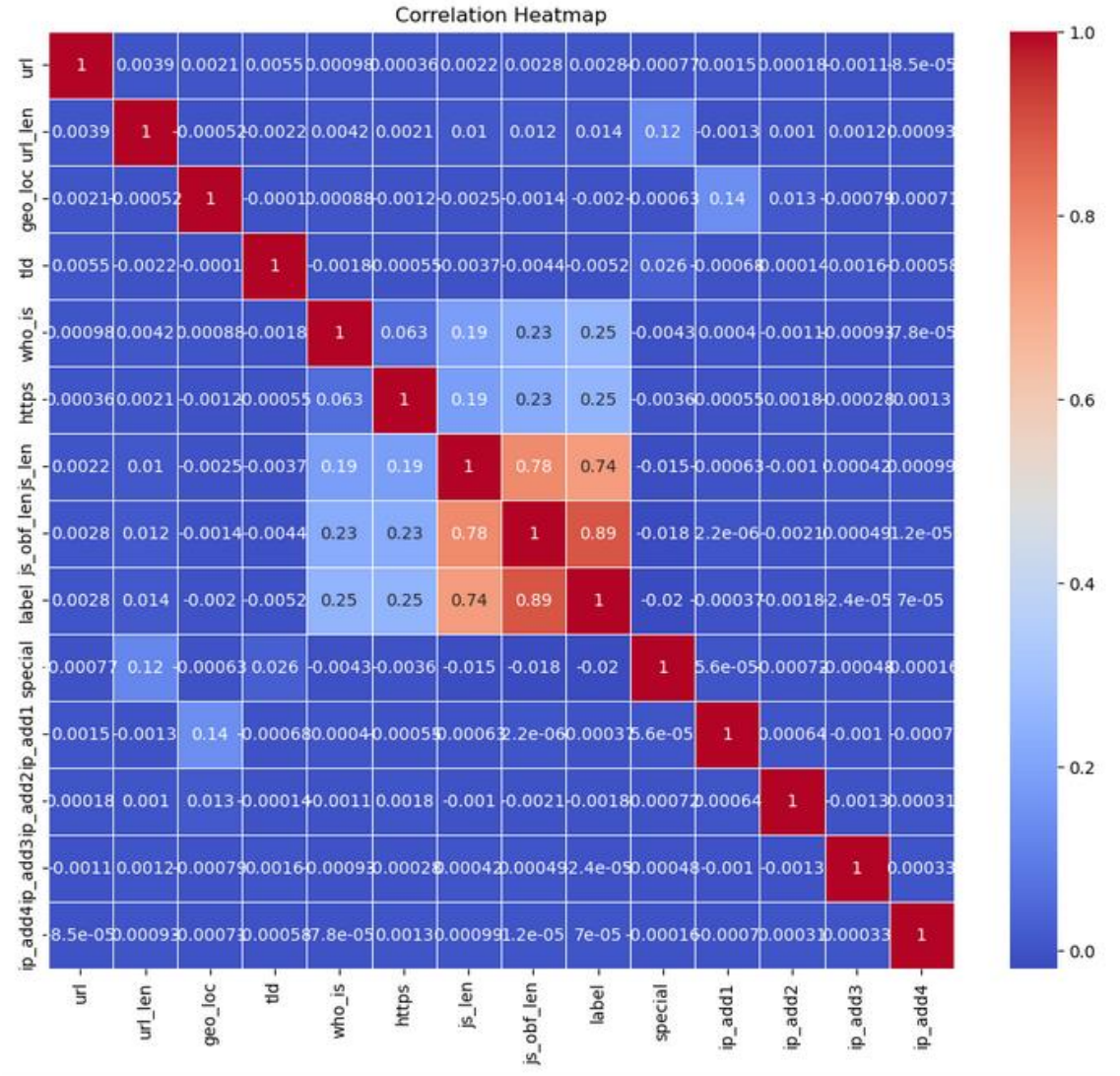
Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkar, Natalia Stakhanova, & Ali A. Ghorbani. (2016). *Detecting Malicious URLs Using Lexical Analysis* (J. Chen, V. Piuri, C. Su, & M. Yung, Eds.; Vol. 9955). Springer International Publishing. https://doi.org/10.1007/978-3-319-46298-1

Ghoualmi, L., & Benkechkache, M. E. A. (2022). Feature Selection Based on Machine Learning Algorithms: A weighted Score Feature Importance Approach for Facial Authentication. *3rd International Informatics and Software Engineering Conference, IISEC 2022*. https://doi.org/10.1109/IISEC56263.2022.9998240

Chakrabarti, S., Institute of Electrical and Electronics Engineers, & Institute of Engineering & Management (Kolkata, I. (2018). *Analysing Feature Importances for Diabetes Prediction using Machine Learning*.

APPENDIX A

Correlation Heatmap



APPENDIX B

Gantt Chart

Enter Your Project Details Here			DURATION (days)
START DATE	END DATE	DESCRIPTION	
START DATE	3/15/22	Introduction	#VALUE!
3/28/22	4/10/22	Literature Review	12
4/18/22	5/18/22	Methodology	30
6/15/22	6/15/22	PSM 1 Presentation	0
6/24/22	6/24/22	PSM 1 Report Submission	0
3/16/23	4/20/23	Results and Discussion	34
4/23/23	5/25/23	Repair Results and Discussion	32
5/31/23	6/8/23	Conclusion	8
6/12/23	6/12/23	PSM 2 Report Submission	0