# Accessible Cybersecurity for Dyslexics

By

Pooja Gupta

Submitted to

**The University of Roehampton**

In partial fulfilment of the requirements

for the degree of

**MASTER OF SCIENCE IN DATA SCIENCE**

# Table of Contents

# Table of Figures

# Table of Tables

# 1 Introduction

## 1.1 Project Context and Idea

The internet nowadays is a large and sometimes too much of a huge source of information and has made it possible for people to do business and communicate with each other all over the world [1]. Conversely, the huge amount of information that is easily available in this interconnected world has also caused major weaknesses, namely of the dangerous URLs [1]. Malicious URLs are the web addresses which are designed to fool the people and to do some harmful actions such as phishing, spreading of malware, or the stealing of personal data [2]. It is important to efficiently detect malicious URLs, to guarantee the safety of online users and to have all perspectives free of harm [7][2]. The exact aim of this research is in creation and improvement of tools aimed at discovering harmful URLs, as a result we can be better prepared in this fight against online malicious activities.

## 1.2 Problem Statement

Major issue that has been dealt with by this study is the identification of the malicious URL strings. Such URLs are instrumental in the cyber assaults, malware propagation, and fraudulent networks. These eventually result in adverse consequences of massive financial losses, data breaches, and destruction of online legitimacy.

These problems affect millions of internet users who entails individual uses, business organisations and even governments. Individuals may often be subjected to identity theft or fraudulent finance and this may also be supported by the business and government organisations that experience data breaches, operational disruptions, and reputation losses.

The problem of malicious URLs is a pervasive and worldwide issue, hence, any organisation that uses the Internet for communication, transactions or data storage is vulnerable to it. The amount of such dangerous situations has risen in the recent years, therefore, this kind of harmful activity has become an alarming and growing danger.

The issue of the harmful URLs is the most significant factor for the completion of the security and the credibility of the internet. By the acknowledgment and the elimination of these dangers, we can preserve the private data, retaining the users loyal to us, and maintain the digital relationships safe. Apart from that, the enhancing of the detection methods will be the main factor that will fortify the whole cyberspace to become more robust against the cyber threat which is always evolving.

## 1.3 Aims and Objectives

### *1.3.1 Aims*

- The key to the successful detection of the malicious URLs is the creation of the powerful algorithms that can detect them accurately.
- To better the understanding of the common malicious features and the typical URL patterns.
- Through the launch of a product that can be used in general life, I will be able to participate in the broader field of cybersecurity.

### *1.3.2 Objectives*

- Conduct a thorough investigation of the available methods for the identification of the malicious URLs through an in-depth literature review.
- Gather and clean a wide range of site URLs that is composed of both the non-malicious and the malicious cases.
- The different machine learning models will be applied and their activities in identifying the malicious URLs will be compared.
- Investigate the aspect that determines the significance of the feature to discover the primary indicators of the fake URL.
- This project will be the development of an ensemble model which is the combination of several algorithms and the accuracy of the detection will be more improved.
- The constructed models must be tested with the real-world data and their performance should be compared with the ones currently in use.
- Make the design of the interface easy and simple to operate for the common people because the detection system to be used practically.
- The improvement of the technology that is being used across different network environments is evaluated.

## 1.4 Legal, Social, Ethical and Professional Considerations

### *1.4.1 Legal Considerations*

The project has to be in accordance with the data protection laws, for instance, the General Data Protection Regulation (GDPR) in the European Union, thus, making sure that the URL data handling will not be infringing the user privacy. In addition, it should also follow the terms and conditions of the usage license.

### *1.4.2 Social Considerations*

The detection of hazardous URLs has significant social implications, for example, the protection of the so-called vulnerable groups from cyber attacks and the assistance in the development of safe internet habits. The project should also consider the digital divide and ensure that the suggested solutions will not be only for a few users but also will be available for a large number of users.

### *1.4.3 Ethical Considerations*

The problems of the ethics that were talked about in the sentence are the ethical way of using data and the openness in the creation and use of detection algorithms. The project should not contain the biases in the models that could lead to the unfair treatment of some people or even the blocking of the real URLs.

### *1.4.4 Professional Considerations*

Professional communications are about following the rules of software development, which are: the criteria for testing, documentation, and user training, which are the factors of success in every project. The teamwork with cybersecurity specialists and the regular updates that are based on the evolving threats are also the basic of the solution for the maintenance of the relevance and the effectiveness of the solution.

## 1.5 Background

The internet has so much speed that it not only made the world a global village but also a dangerous place for the Internet. The increase of the malicious URLs as a successful threat vector has been caused by the use of social engineering techniques and the exploitation of the software and human vulnerabilities. The report of the Anti-Phishing Working Group (APWG) indicates that in 2022 the number of phishing sites detected was the highest ever, hence the necessity for the development of the detection mechanisms was stressed as an urgent issue [4].

Present methods of malicious URL detection are the blacklisting, the heuristic-based verification, and the machine learning models. The process of blacklisting, although simple, has the problem of trying to match the rapidly changing features of the threats. The heuristic-based methods, which are ruled by the pre-set rules, mostly do not bring new attacks. The use of machine learning methods has been a good system for analysing URL data for the identification of the criminal activity with more accuracy and more flexibility [3].

The main idea of this project is to build on the previous research work and at the same time, it tries to correct the weaknesses of the past projects which are scalability, accuracy, and real

world application. The project of the said will be the union of the most advanced machine learning techniques and the whole datasets to the detection of the malicious URLs and will provide a strong answer to the problem of the ongoing challenge of the malicious URLs.

### 1.6 Report Overview

The subsequent sections of this report are structured as follows:After this introduction, the subsequent parts of this report are arranged as below.

Literature Review: A comprehensive study of the existing literature and methods of the so-called potentially harmful links.

Methodology: The researcher gives the information about the data collection, the steps of preprocessing and the machine learning models that were utilised in this research.

Results: The slide and the evaluation of the performance metrics of the created models, compared to the present solutions are the main points of the output.

Discussion: The results have to be analysed, which implies that the main points have to be established and the practical implications of the findings have to be discussed.

Conclusion and Future Work: The main point of the research, the studies in the field and their role, and the possible directions for the future research.

# 2 Literature Review

## 2.1 Introduction

The perpetual rise of complexity and the widespread of cyber threats make it inevitable to have the modern technologies for the detection and the removal of the malicious URLs. The old methods like blacklists and heuristic approaches are not as effective as the emerging threats that are now appearing. Hence, the field of research has been reoriented to the machine learning and deep learning methods, which, in turn, have given the higher precision and flexibility. This literature review evaluates the important studies which use various deep learning models for the detection of malicious URLs thus, it gives us the insight into the method, results and the ways for future research.

Evolutionary Deep Learning and Bloom Filter Mechanism is the phrase of the combination of two processes which will be the reason of the efficiency of the deep learning system and will be the road to the world of advanced systems in the future.

## 2.2 Deep Learning in Malicious URL Detection

Patgiri [5] put forward the deepBF, which is a system that unites the evolutionary deep learning with the self-adjusted Bloom Filter. This research by the National Institute of Technology Silchar is concentrated on the recent methods of malicious URL detection improvement and its impact on the effectiveness and reliability.

The deepBF model, which is based on the evolutionary convolutional neural networks (evoCNN), is applied to the URL classification to achieve the continuous improvement of the network architecture. The self-adjusted Bloom Filter is based on non-cryptographic hash functions and hence the FPP is significantly reduced. Various non-cryptographic hash functions are chosen, among which Murmur2 is the one which is changed to have more bias and redundancy, which in turn, gives the lowest FPP.

The experiments revealed that the Bloom Filter was used more memory to get the higher accuracy (98-100%) in training and (95-98%) in testing, that other variants of the Bloom Filter could not do. The latest research proves that the use of a self-adjusted Bloom Filter and the evolutionary deep learning system is a strong and fast way of detecting the malicious URLs.

This book, written by Stamp, Alazab, and Shalaginov [6], is a compilation of these AI techniques for malware detection, among which convolutional neural networks (CNNs) and

recurrent neural networks (RNNs) are the most frequent ones. The book is made up of the in depth research of the modern AI techniques that fight against the malware attacks.

The techniques that are found across the different sections of the book are the advanced AI strategies which are applied for the purpose of making malware detection and analysis more efficient. This is achieved by the use of both the classical and the modern machine learning models that are made for the dynamic and static analysis of malware, the deep learning techniques like CNNs and RNNs which increase the accuracy of the malware classification and prediction while the hybrid approaches that use the multiple AI techniques to form the systems capable of detecting and analysing the sophisticated malware types.

The studies and experiments so far have proved that the malware detection rates have risen considerably, the number of false positives has been reduced and the ability to understand the complex malware behaviors has been enhanced. The sentence that directly informs the impressive performances of different models in the detection of known and unknown malware samples with the highest accuracy rates is stated. Besides, the possibility of the AI in transforming the malware analysis field is also mentioned.

Supervised Machine Learning Techniques for Malicious URL Detection are the methods where the Internet links are supervised by the experts and the data being collected by the specialist to detect the malicious URLs.

Nevertheless, Vundavalli [1]acknowledge that supervised learning machine learning instances can be employed in cybersecurity specifically to prevent such cyberattacks originating from malicious URLs. The research used different machine learning techniques in the ripping off of the falsified web acclivity to be done much easier. The method of logistic regression is the one considered to be mostly contributing to the success of the model at predicting whether a connection is dangerous or not. The Gaussian, Multinomial andBernoulli naïve Bayes classifiers were deployed to explore and determine patterns and probabilities in the data. CNN has been the neural network that exhibited the best results for identifying spatial patterns. They especially proved useful when it came to the recognition of evidence that could determine a set of features.

The results of the machine learning methods were compared and the results published show a success rate of 86% was achieved by logistic regression. 25% of the URLs are with less accuracy and recall, which means that the method is able to identify benign but not malicious URLs. The naïve Bayes classifiers performed differently in the case of the experiments while

Bernoulli naïve Bayes turned out to be the best one, because it was designed for categorical data. The CNN made the accuracy of logistic regression to 88 on CNN. The fact that 32% is a number that well shows that deep learning models are better than other models in the complicated tasks of the pattern recognition. The paper finally came out with the conclusion that the mixture of these methods can generate easier ways for the detection of the malicious URLs.

Aalla, Dumpala, and Eliazer [2] developed the method for the detection of malicious URLs with the decision trees and the logistic regression. The researchers used the decision tree and the logistic regression models to differentiate the malicious and the legitimate URLs. The decision tree model is the one that uses a tree-structured graph of the decisions and their possible outcomes. Logistic regression models are the basic ones which use a logistic function to model a binary dependent variable.

The model of the decision tree came out to be a very competent model which had an accuracy of more than 85%. Still, it used a much higher computational time in comparison to the logistic regression. Logistic regression achieved the best efficiency with an accuracy rate of 97. Fifty percent of the results show the effectiveness of the program in the fast processing and the right prediction of the malicious URLs. This research, therefore, confirms the importance of employing sound machine learning techniques to enhance cybersecurity in the battle against malicious URLs, leading to the logistic regression, a very efficient and accurate method, which, in turn, demonstrates its capability to process massive datasets.

According to Wang [7] a dynamic convolutional neural network (DCNN) with the folding and k-max-pooling layers was created for the purpose of designing a model for detecting malicious URLs. The study was done in different universities in China, for instance, the Beijing Electronic Science and Technology Institute, and the State Information Center.

The DCNN model has the folding layer and the k-max-pooling layer which are the pooling layers generally used, but in this case they are replaced by these two layers. This design allows the URL input length and the depth of the convolution layer to be adjusted so that it can be suitable for the different types of data which in turn, makes the feature extraction the deeper. In addition, the model has a new word embedding method that builds on the character embedding, which is the vector representation of URLs, thus, the online malicious URL detection is done more efficiently and accurately.

The model's effectiveness was verified by carrying out numerous experiments which compared different embedding methods and network architectures. The model was the most accurate one in the cutting of the dangerous links with the accuracy of 98%. The tests comparing the proposed word embedding with the traditional word and character embedding showed that the former was superior to the latter in terms of accuracy and the efficiency of the media. Through the paper's method, the deep learning techniques like, for instance, convolutional neural networks, can be illustrated for the creation of new technologies for cybersecurity which can be used for the protection against malicious URLs.

The Rasymas and Dovydaitis [8] use the deep neural networks methods of exploration, namely CNN and LSTM layers, for phishing URL detection. The study offered a new deep neural network architecture structure that contained several CNN and LSTM layers. It compared three different features for phishing URL classification: The same set of lexical features as the ones extracted from the text of the URL, character-level embeddings representing the character-level patterns, and word-level embeddings representing the word-level patterns.

The accuracy of the classification that turned up 94.4% is a very high rate gotten by the combination of character and word level embeddings, thus, this combination was considered to be a success for finding phishing URLs. The research established that the traditional blacklist technique for phishing detection was ineffective and proved the effectiveness of machine learning techniques in the identification of the malicious URLs before the attack. The research not only shows the readiness of the machine learning models which have high complexity, to be employed against the phishing attacks but also reveals the possibility of using the machine learning models which have high complexity to beef up the cyber security measures against the phishing threats.

The improved multilayer convolutional recurrent neural network (RCNN) model for malicious URL detection by [9] is the alternative suggestion. The result presented a model that was equipped with the word embedding, feature extraction process, which used CSPDarknet and bidirectional LSTM, that had never been RCNN. The word embedding replaces the URL symbols with the dense vectors which represent the features of the structure. The Darknet neural network is the architecture used for feature extraction which is the improved model of YOLO (you only look once). The third step of the process is feature extraction. The URL is then analysed by bidirectional LSTM network. It checks if the URL is

malicious or not [9].

The experiment took place with 200,000 URLs where each of them was assigned a label of either 'good' or 'bad'. The model showed more accuracy, higher precision, and a high recall than RCNN, BRNN (Bidirectional Recurrent Neural Network), or other base models. The new techniques had the advantage of speed and reliability over the previous ones, which were the best tools in all the aspects of detection. This article depicts a method of leveraging state-of-the-art deep learning techniques that make websites more powerful against malicious links.

Whereas Al-Ahmadi and Alharbi [11] designed a bi-directional CNN model to analyse both of the URL attributes and the visual features of websites for identifying the phishing activity. For this purpose, the researchers have designed a prototype consisting of two CNNs that analyse the features of the URLs of the websites and their visual features. Then, it makes the decision whether the analysed websites are real or not. The methodology consists of two main components: and an ANN to process URLs through which a webpage is determined harmless or a phishing one, and an ANN as well to play with visual information to make such decisions. At the end, the final decisions according to whether a website is harmful or not, will be made when the outputs of both CNNs are combined.

The proposed technique proves to be a very reliable one, the accuracy level being as high as 99.67% achieved in experiments. In this way, it can be said that the referral of URL analysis and visual inspection has high performance and it can detect fake websites as well. The above study points to the fact that the deep learning techniques stand out, especially the convolution neural network, as well-being the tools that can be put to use to defend against the malicious cyber threats that are more dynamic and complicated to detect.

The machine learning classifiers along with particle swarm optimisation (PSO) that the researchers used allows them to be able to get the features needed for them to detect the malicious URL [10].

Feature selection method was employed by this researcher using two machine learning classifiers and one bio-inspired algorithm, PSO, in particular, to make the detection of malicious URLs more precise. PSO with the feature optimisation process in place helped in selecting those features from the URL that were red flags of malicious intent. Lastly, the best chosen features were then fed into the classifiers like Naive Bayes and Support Vector Machine (SVM) to check the improvement in malicious URL detection.

To classify URLs with the optimisation features, Naïve Bayes and SVM classifiers unbelievably have the detection accuracy of 99%, and so, they are very cost-efficient. The investigation indicates that the fusion of the feature optimising along with the sturdy machine learning classifiers can better be used in detecting the malicious URLs. Therefore, a robust security measure against the cyber threats can be possible as a result.

The researchers of the team led by Yuan [12] introduced the joint neural model that is parallel by means of CapsNet and IndRNN to detect malicious URLs. The research was published in the Journal of Network and Systems Management.

The presented study is a new parallel neural joint model that combines CapsNet and IndRNN in order to make use of both the visual and the semantic information from URLs. CapsNet is the one that interprets visual patterns from those URLs that are transformed into gray images and extracts text and structure patterns. The URL features are processed by IndRNN and this results in the enhancement of the model's capacity to analyse and understand URL sequences that are of malicious nature. Therefore, a dual approach that combines the pros of the depth visual imagery analysis with the sequential text processing inevitably results in the comprehensive analysis of urls.

The research papers outcomes indicate that this two approach is way better than the normal ones for malicious URL detection. In the main observations, the main points are that the classification accuracies are mainly depending on the conditions of the experiment. Conversely, the outcomes indicate that the classification accuracy was 99%. The 78% for the detection of the fraudulent URLs is the proof of the usefulness of this tool in the detection of the such URLs. The strong force learning that is achieved by the combination of visual and text features through CapsNet and IndRNN enables the prediction of the harmful URLs to be highly precise, hence the system becomes more efficient against the sophisticated cyber threats. The study is of the opinion that the fusion of different neural network models will lead to the improvement of the cybersecurity system.

Alsaedi [13] proposed a model of learning that is the mixture of the ensemble learning strategy, The research developed a new two-stage ensemble learning model that was built on the CTI-features extracted from the web searches and Whois databases and its main goal was to increase the accuracy of the detection of malicious URLs. The model incorporated both the Random Forest (RF) technology for the initial classification and the Multilayer Perceptron (MLP) for the final decision-making. The idea behind the combination of RF and MLP

hybrid methods is that the probabilistic outputs of the RF classifiers are used as an input for the MLP, thus, the classification accuracy is increased.

The study proves the fact that the improvement in the search for the wrong URL is considerable. The combination of CTI characteristics with the two-stage classification approach made for a 7. The 8% growth in accuracy and the 6% rise in the results are both the signs of the upgrade. The false positives rate was reduced by 7% as compared to the URL-based models. The feasibility of the CTI features was a trustable way to enhance the detection function, therefore, by using the global intelligence and the local database insights to find the potential threats more accurately.

## 2.3 Critical Analysis and Discussion

### *2.3.1 Strengths and Innovations*

The reviewed studies as a whole prove that the malicious URL detection is now a major positive effect which is achieved by the deep learning and machine learning techniques. At first, in other words, the combination of advanced neural network technology, for example, CNNs, RNNs, and hybrid models, has greatly improved the detection precision and efficiency. The novel combination of feature optimisation, dynamic network structures, and the dual analysis method (combination of textual and visual features) is the example of the deep learning being used to tackle the cybersecurity challenges that are becoming more and more complex and changing.

### *2.3.2 Weaknesses and Limitations*

Even though the development has been going on, these researches discourage the optimism just because the limitations still exist, for instance, the need of big, labelled datasets for training the deep learning models and the computational power of the advanced neural networks. Besides, the feasibility of some models for new, unknown threats cannot be confirmed yet, especially the ones that are yet to be developed. The society of a certain feature set, for instance, the lexical features, can also lead to the creation of a model that is not robust.

### *2.3.3 Comparative Approaches*

Literature researches prove that the results of hybrid and ensemble approaches are more than the ones of the single-model approaches. The methods of CNNs and RNNs, as well as the integration of CTI features with the conventional classifiers are combined to mention such as the combination of CNNs and RNNs, and the integration of CTI features with the

conventional classifiers, the performance of these methods is much better than other methods. The results of the these studies show that the abilities of various models and the different feature sets can be used as an incentive to increase the detection of something.

## 2.4 Conclusion

The researcher's literature on malicious URL detection through deep learning has proven that the use of cutting-edge AI techniques has been the real game changer in the field of cybersecurity. The studies that are reviewed indicate that the deep learning models, when combined with the new feature extraction and optimisation methods, result in the significant improvement of the accuracy and speed of the detection of malicious URLs. Future research should focus on solving the currently existing limitations, for example, the need of large datasets and huge computational resources, and on the possibilities of the combination or the ensemble techniques. Thus, the regular enhancement of the models will result in the creation of more powerful and wider solutions to combat the cyber threats that are always changing.

# 3 Methodology

## 3.1 Data Collection

The project uses holistic data collection and preprocessing techniques which proceed from data reliability and model accuracy to malicious URL detection. The data required for the study is obtained from University of Qom resources available at GitLab [14]. The data comprises URLs from multiple datasets: normal URLs, defacement URLs, malware URLs, spam URLs, and phishing URLs. These datasets are labelled as 0, 1, 2, 3, and 4, separately, such that, 0, 1, 2, 3, and 4, stipulate different types of URLs. For data collection, benign registered URLs are taken from the Benign_list_big_final. defacement, from DefacementSitesURLFiltered. csv. . . csv in Malware_dataset. cf, spam URLs from spam_set. csv, as well as some phishing URLs from the phishing_dataset. csv.

## 3.2 Data Preprocessing

Different steps that are aimed to transform a loosely structured URL data into a structured-like format are applied in the data preprocessing that will be later utilised by machine learning algorithms. These steps start with the feature extraction through calculation of URL length, identifying host name length, path length, extracting the length of the first directory in URL path, extracting occurrences of top level domain (TLD) with its length, and counting special characters. which includes negative symbols, parentheses, bracket, question mark, sign percent, and dot. While performing this process, it is necessary make sure to take care of [' '=', 'http', 'https', 'www'], added before each URL. Furthermore, the data pertaining to the number of digits and letters in each URL, as well as the URL path's number of directories, IP URLs and URL shortening services, are also considered [15],[16].

## 3.3 Exploratory Data Analysis

Exploratory data analysis (EDA), here, is an indispensable method used to understand how the distribution and relationships among different data features. Trade methods such as correlation matices, histograms and count plots are used to visualise the data [15]. The essential steps of EDA involve investigating the correlation of features, creating plots which portray the categorical distribution of URLs, graphical presentations of the characteristics of URLs based on their sise in bytes, count of IP addresses and URL shortening services within URLs, as well as the number of special characters in URLs.

### 3.4 Data Analysis

This project focuses on various machine learning techniques in distinguishing the most successful model to detect dangerous URLs. The algorithms that are chosen include several classical machine learning models and also a deep learning model. Those models range from support vector machine (SVM)[17], random forest[18], XGBoost[19], AdaBoost[19], and other classifiers such as LightGBM[20], decision trees[22], extra trees[23], and deep multi-layer perceptron (MLP)[24]. Every model is chosen for its individual applause points. Support Vector Machine (SVM) is the one that is selected for the reason of support for multiclass classification and dealing with high-dimensional data. Random Forest is picked for its resistance and functionality to handle input variables that are numerous. XGBoost is chosen as it has a gradient boosting framework, which improves the model's performance. AdaBoost is selected due to its contribution on boosting in which weak classifiers are combined to form a stronger classifier. Due to its efficiency and scalability LightGBM is chosen to deal with big data sets. Decision Trees are employed because of their concise and comprehensible nature. Extra Trees Classifier is selected for its ensemble approach with the purpose of preventing overfitting. Hence, the Deep MLP that is used for its deep learning and its potentials in learning complex patterns is the last one.

### 3.5 Model Training

Model training involves splitting the dataset into training and testing sets. The training set would encompass the entire 90 percent bisection of the dataset while the testing set would encompass the remaining 10 percent bisection after the dataset has been split. For each of the algorithms, the models are trained on the training set and the testing set is evaluated with that by accuracy and F1-score. To check whether models are robust and have good generalising capability the cross-validation method will used. In accuracy and F1-score, the best model for this task is the Random Forest model, which demonstrated the highest accuracy and best F1-score, proving to be the best one. The outcome is represented using bar and line plots, that is responsible for the difference in accuracy of different models.

### 3.6 User Interface

### 3.7 Project Management

The project follows structured project management techniques to ensure timely and organised progress. Tools such as Gantt charts and progress trackers are used to monitor project milestones and deliverables.

### 3.8 Tools and Technologies

The project depends on different tools and technologies to accelerate data processing, model training and model assessment stages. They include for instance *pandas[33]* for data modification and plotting, *NumPy* for numerical processing, *matplotlib* and *seaborn* for visualisation of data, *sklearn[32]* for the implementation of supervised ML methods, *lightgbm[30]* and *xgboost[31]* for gradient boosting algorithms, *TLD[29]* for the extraction of the domain name from the URL addresses and *re* for regular expressions in feature extraction. These libraries and tools are selected on the basis of their solidity, familiarity, and extent of the functionalities they provide, to make sure that a stable and well designed system for malicious URL detection will be used. The project is based on this method and therefore will build a reliable and effective system for the detection of malicious URLs which will consequently make the cyber security measures a step further and protect users from online threats.

# 4 Implementation

The project's architecture is meant to be effective and precise at identification of malicious URLs by means of machine learning technology. The system comprises several key components: data collect, preprocessing and its exploratory data analysis (EDA) as well model training and evaluation, and a user interface for real time URL classification. The goal for the design is scalability, modularity, and ease of deployment; this way the solution will be compatible with existing cybersecurity frameworks.

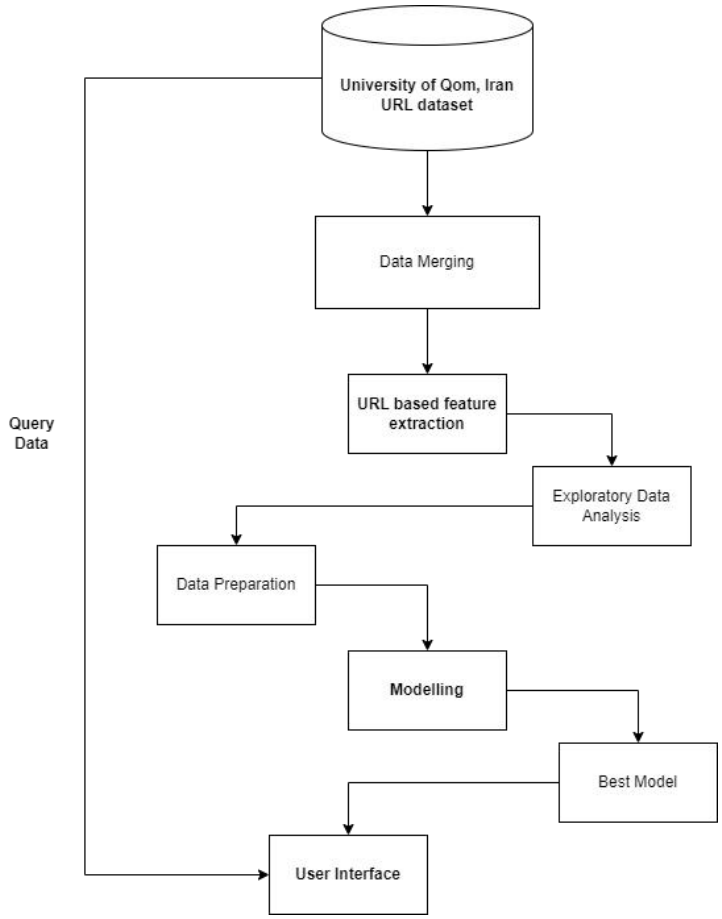


***Figure 1: System Design***

## 4.1 Data Collection and Preprocessing

The first task for implementation is gathering of the dataset that consists of the collection of various types of URLs. These URLs are categorised into five classes: benign, defacement, malware, spam, and phishing. This category is of the leading feature as it trains the models to properly differentiate malicious URLs from an ordinary one.

### *4.1.1 Data Collection*

- **Benign URLs**: Collected from Benign_list_big_final.csv.
- **Defacement URLs**: Gathered from DefacementSitesURLFiltered.csv.
- **Malware URLs**: Sourced from Malware_dataset.csv.
- **Spam URLs**: Extracted from spam_dataset.csv.
- **Phishing URLs**: Obtained from phishing_dataset.csv.

Each set of URL samples is labelled with integers from 0 to 4, here 0 represents benign URLs and 4 represents phishing URLs. This labelling is essential in supervised learning, as it enables the model to learn from the labelled ones.

| | url | label |
|---|---|---|
| 0 | http://1337x.to/torrent/1048648/American-Snipe... | 0 |
| 1 | http://1337x.to/torrent/1110018/Blackhat-2015-... | 0 |
| 2 | http://1337x.to/torrent/1122940/Blackhat-2015-... | 0 |
| 3 | http://1337x.to/torrent/1124395/Fast-and-Furio... | 0 |
| 4 | http://1337x.to/torrent/1145504/Avengers-Age-o... | 0 |

***Figure 2: Sample URLs in 'Benign_list_big_final.csv'***

| | url | label |
|---|---|---|
| 0 | http://www.sinduscongoias.com.br/index.html | 1 |
| 1 | http://www.sinduscongoias.com.br/index.php/ins... | 1 |
| 2 | http://www.sinduscongoias.com.br/index.php/ins... | 1 |
| 3 | http://www.sinduscongoias.com.br/index.php/ins... | 1 |
| 4 | http://www.sinduscongoias.com.br/index.php/ins... | 1 |

***Figure 3: Sample URLs in 'DefacementSitesURLFiltered.csv'***

### *4.1.2 Data Preprocessing*

The raw URLs are processed to extract meaningful features that can aid in classification. The features include:

```
#Length of URL
data['urlSize'] = data['url'].apply(lambda i: len(str(i)))
data.head()

#Hostname Length
data['hostNameSize'] = data['url'].apply(lambda i: len(urlparse(i).netloc))
data.head()

#Path Length
data['pathSize'] = data['url'].apply(lambda i: len(urlparse(i).path))
data.head()

#Function for finding first directory
def firstDirSize(url):
    urlpath= urlparse(url).path
    try:
        return len(urlpath.split('/')[1])
    except:
        return 0

data['firstDirSize'] = data['url'].apply(lambda i: firstDirSize(i))
data.head()

#Top Level Domain
data['tld'] = data['url'].apply(lambda i: get_tld(i,fail_silently=True))
data.head()

#Length of Top Level Domain

def tldSize(tld):
    try:
        return len(tld)
    except:
        return -1

data['tldSize'] = data['tld'].apply(lambda i: tldSize(i))
```

***Figure 4: Python functions to Calculate Features related to tld***

**URL Length**: The length of each URL is calculated to see if there is a correlation between length and maliciousness.

**Hostname Length**: The length of the hostname in each URL is measured.

**Path Length**: The length of the URL path is determined.

**First Directory Size**: The size of the first directory in the URL path is extracted.

**Top-Level Domain (TLD) and its Length**: The TLD is extracted from each URL and its length is measured.

**Special Characters Count**: The occurrences of special characters (e.g., '-', '@', '?', '%', '.', '=', 'http', 'https', 'www') within each URL are counted.

```
data['Num-'] = data['url'].apply(lambda i: i.count('-'))
data['Num@'] = data['url'].apply(lambda i: i.count('@'))
data['Num?'] = data['url'].apply(lambda i: i.count('?'))
data['Num%'] = data['url'].apply(lambda i: i.count('%'))
data['Num.'] = data['url'].apply(lambda i: i.count('.'))
data['Num='] = data['url'].apply(lambda i: i.count('='))
data['NumHTTP'] = data['url'].apply(lambda i : i.count('http'))
data['NumHTTPS'] = data['url'].apply(lambda i : i.count('https'))
data['NumWWW'] = data['url'].apply(lambda i: i.count('www'))
data.head()
```

*Figure 5: Python functions to obtain special characters from the URL*

**Digits and Letters Count**: The number of digits and letters in each URL is counted (see Figure below).

```
def digitTotal(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
data['NumDigits']= data['url'].apply(lambda i: digitTotal(i))
data.head()

def letterTotal(url):
    letters = 0
    for i in url:
        if i.isalpha():
            letters = letters + 1
    return letters
data['NumLetters']= data['url'].apply(lambda i: letterTotal(i))
data.head()

def dirTotal(url):
    urldir = urlparse(url).path
    return urldir.count('/')
data['NumDir'] = data['url'].apply(lambda i: dirTotal(i))
data.head()

def ip_exits(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' # IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # Ipv6
    if match:
        return -1
    else:
        return 1
data['ipFlag'] = data['url'].apply(lambda i: ip_exits(i))

def url_shortening_exists(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                      'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
                      'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
                      'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                      'tr\.im|link\.zip\.net',
                      url)
    if match:
        return -1
    else:
        return 1
data['shortFlag'] = data['url'].apply(lambda i: url_shortening_exists(i))
```

*Figure 6: Python functions to calculate different features from the URL*

**Directory Count**: The number of directories in the URL path is calculated.

**IP Address Check**: URLs containing IP addresses are identified.

**URL Shortening Service Check**: The presence of URL shortening services is checked.

## 4.2 Modelling

### *4.2.1 Hyperparameters and Rationale*

**SVM**: kernel='linear', C=5, gamma=8, max_iter=2. These hyperparameters are chosen to balance model complexity and performance. The linear kernel is effective for high-dimensional data, while C and gamma control the trade-off between achieving a low error on training data and maintaining model simplicity.

**Random Forest**: Default parameters are used initially to establish a baseline, with potential tuning based on performance.

**XGBoost**: Default parameters, with a focus on leveraging its gradient boosting capabilities.

**AdaBoost**: Default parameters, utilising its strength in combining weak classifiers.

**LightGBM**: C=100, chosen for its efficiency and ability to handle large datasets.

**Decision Trees**: Default parameters, benefiting from their simplicity and interpretability.

**Extra Trees Classifier**: Default parameters, to leverage ensemble learning and reduce overfitting.

**MLP**: hidden_layer_sizes=(12,12,12,12), activation='relu', solver='adam', max_iter=500. These settings provide a deep network capable of learning complex patterns, with relu activation for non-linearity and adam optimiser for efficient training.

### *4.2.2 Model Evaluation*

The performance of each model is evaluated using accuracy and F1-score metrics. Bar plots are plotted to graphically visualise the performance of the different models using the matplotlib library. The accuracy and f1-scores for the models were obtained using the sklearn library's metrics module [32].

### *4.2.3 Deployment and Final Model*

The best model based on the evaluation metrics is saved using the pickle library for deployment in user interface [35]. This model can be integrated into cybersecurity systems to provide real-time detection of malicious URLs, enhancing overall internet security.

### *4.2.4 User Interface Implementation*

A web application is developed using Flask [34] to provide a user-friendly interface for URL classification. This application allows users to input URLs and receive classification of the URL on whether the URL is possibly malicious or normal.

***Flask Application Structure***

The flask application implemented in the study is backed by two pages that are opened for different responses either from the user side or the server side.

**Home Page:** A simple form where users can input a URL. This is a user side page. The page responds through a POST object of API the input received from the user. The server receives the response and performs the feature extraction from the URL. These features are then passed on to the best model which used to classify the URL as normal or malicious.

**Result Page:** Displays the classification result based on the model's prediction. This page receives response from the server. The results are received as json which is processed to display the final classification of the query URL.

Feature extraction functions are applied to the input URL to extract features required for classification. The python code for the UI development is given in figure 7 below.

```python
@app.route('/result', methods = ['GET', 'POST'])
def shortlist():
    if request.method == 'POST':
        url = request.form['websites']
        data = []
        data.append(len(url))
        data.append(len(urlparse(url).netloc))
        data.append(len(urlparse(url).path))
        data.append(firstDirSize(url))
        data.append(tldSize(get_tld(url,fail_silently=True)))
        data.append(url.count('-'))
        data.append(url.count('@'))
        data.append(url.count('?'))
        data.append(url.count('%'))
        data.append(url.count('.'))
        data.append(url.count('='))
        data.append(url.count('http'))
        data.append(url.count('https'))
        data.append(url.count('www'))
        data.append(digitTotal(url))
        data.append(letterTotal(url))
        data.append(dirTotal(url))
        data.append(ip_exits(url))
        data.append(url_shortening_exists(url))
        data = [0 if di is None else di for di in data]
        data=np.array(data).reshape(1, -1)
        with open("bestModel.pkl","rb") as f:
            model = pickle.load(f)
            f.close()
        prediction = np.round(model.predict(data),0)
        if prediction == 1:
            prediction = 'Real'
        else:
            prediction = 'Fake'
        print(prediction)
        return render_template("shortlist.html", url=url, prediction=prediction)


if __name__ == '__main__':
    app.run()
```

***Figure 7: UI development using Flask in Python***

- @app.route('/result', methods=['GET', 'POST']): This decorator defines a route for the Flask web application. It specifies that the /result URL endpoint can handle both GET and POST requests.
- def shortlist(): This is the function that will be executed when a request is made to the /result endpoint.
- if request.method == 'POST': Checks if the request method is POST.
- url = request.form['websites']: Retrieves the URL input by the user from the form field named websites.
- data = []: Initializes an empty list to store features extracted from the URL.
- data = [0 if di is None else di for di in data]: Ensures that any None values in the data list are replaced with 0.

- data = np.array(data).reshape(1, -1): Converts the list into a NumPy array and reshapes it to have one row and multiple columns. This is necessary for compatibility with the machine learning model's expected input format.
- with open("bestModel.pkl", "rb") as f: Opens the file bestModel.pkl in read-binary mode.
- model = pickle.load(f): Loads the pre-trained machine learning model from the file using the pickle module.
- prediction = np.int32(model.predict(data)): Uses the loaded model to predict the class of the input URL based on the extracted features. The prediction is converted to a 32-bit integer for consistency.
- if prediction == 1: Checks if the model's prediction is 1.
- If prediction == 1, it means the URL is classified as "Real" (benign).
- Otherwise, it is classified as "Fake" (malicious).
- print(prediction): Prints the prediction to the console (useful for debugging).
- return render_template("shortlist.html", url=url, prediction=prediction): Renders the shortlist.html template, passing the input URL and the prediction result to be displayed to the user.
- if __name__ == '__main__': Ensures that the Flask application runs only if the script is executed directly (not imported as a module).
- app.run(): Starts the Flask web server, allowing the application to handle incoming HTTP requests.


***Figure 8: User interface***

# 5 Results

## 5.1 Summary of Outcomes

The study targeted the creation of a powerful system that would help in recognising malicious URLs comparing against a number of algorithms. Through the task of amassing, pre-processing and performing the analysis, the project managed to create several models able to detect malicious URLs that performed very precisely. The last step of the study involved the development of a web interface that was friendly to users for the real-time URL, which increases the model usability in its practical sense.

Random Forest turned out to be the model of the lot, getting the highest amount of accuracy and F1-score amongst models used for training.

## 5.2 Evaluation of Project Aims

The main aim in the realisation of this project was the development of a secure system intended for malicious URL detection. We achieved this objective by the production of many different ML models from all of which accurateness and F1-score metrics were calculated. The project sought to ensure classification of a web page in real-time by implementing a user-friendly interface, which was realised through the implementation of Flask web application.

## 5.3 Evaluation of Artefact

The contribution of this project is the development of a comprehensive system that can catch the URLs which tried to be manipulated in many different methods and the multiple machine learning models and the web interface will help the purpose. The strengths and weaknesses of the artefact are evaluated as follows:

### *5.3.1 Strengths*

**High Accuracy:** With Random Forest an accuracy of 98.96% corresponds making the URL that malicious URLs is highly reliable. This degree of accuracy implies that the model possesses the ability to differentiate between malicious and normal URLs, which, in turn, generates a reliable tool for cyber defense.

**Robustness**: The model's robustness is confirmed by the fact that it processes a large number of input variables without hesitation and produces the same results no matter the circumstances. This is precisely the strong point that makes it possible to deal with the issues in applications that involve URL differences, which can be extremely varied.

**User-Friendly Interface:** The flask-based application simplifies the process of URL input and provides users with immediate updates to the URL's potential harmfulness. The system easiness and simple design makes it very easy to work for daily tasks even for people with no technical background.

### *5.3.2 Weaknesses*

**Imbalanced Data**: The imbalance of the examples was required, and sampling techniques to solve this does not have been considered. The problem of this skew can be the reason of not operating model in the best way, mainly for the underrepresented cases. For example, classes with a low number of model samples might have more misclassification rate than of classes with a high number, and thus, can potentially lead to the complication of these models.

**Feature Selection:** An important ongoing task is to define what are the characteristics that should be considered key. While the baseline features have delivered a promising performance, parametric selection and parameter optimisation will further improve model performance. High-level methods like RFE and PCA could be considered to get rid of unbeneficial characteristics.

**Overfitting**: Some models, on the other hand, showed overfitting, which means that they were good at dealing with the training than test set. Moreover, being aware of regularisation approach the technique mechanism in this work would be able to make the more effective and efficient improvements. Strategies like cross-validation, dropout for a neural networks, or robust regularisation can be seen as tools which will help to preventing overfitting.

**Real-Time Processing**: The current system could experience certain troubles related to the real time processing of giant volume. Minimising the time factor through using e. g. powerful algorithms or multiple computing structures, could make the system deal with high-input data faster.

## 5.4 Standard Metrics Evaluation

The approach employed well-known indexes of efficiency such as accuracy and F1-score to assess the quality of the model. The metrics give an overall perspective of how the models perform in tossing out malicious links from their non-malicious counterparts. The models achieved the following results:The models achieved the following results:

| Model | Accuracy | F1-Score |
|---|---|---|
| **Random Forest Trees** | 98.965955 | 98.965955 |
| **Extra Trees Classifier** | 98.899438 | 98.899438 |
| **XGBoost** | 98.760356 | 98.760356 |
| **LightGBM** | 98.572897 | 98.572897 |
| **Decision Trees** | 98.433815 | 98.433815 |
| **Deep Multi-Layer Perceptron** | 93.275685 | 93.275685 |
| **AdaBoost** | 70.593215 | 70.593215 |
| **SVM** | 8.060712 | 8.060712 |

***Table 1: Model Evaluation***

### *5.4.1 Analysis of Results:*

Random Forest Trees: Because of the Random Forest model's having in the highest accuracy and F1-score, it showed that this model was useful in identifying suspect URLs. Its ensemble learning which is also the approach of many decision trees that combines the prediction of many trees is likely to be the reason of its high performance by reducing the variances and improving the generalisation.

XGBoost and LightGBM: There was a well-demonstration of the superiority of the mentioned gradient boosting models' accuracies and F1-scores as compared to the Random Forest model. Such models are indeed preferred by researchers as they can handle a large bulk of data and also can work with complex relationships making thus ideal for the forecast.

AdaBoost: Performing at moderate level between the AdaBoost model shows that it was less accurate and F1-scores were also less than the scores of the other ensemble methods. The reason for this could be the effect owing to the boosting algorithm that tends to magnify the hard examples that are prone to overemphasis causing overfitting.

Deep Multi-Layer Perceptron (MLP): The model based on multi-layer perceptron exhibited outstanding results as it was able to display the essential non-trivial patterns within the data. Nevertheless, its higher precision and recall level, which outperform the bottom-ranked models, validate the method as a useful tool for improving the performance.

Support Vector Machine (SVM): Sunction nje SVM parama andscakila nhukwiwa yandimashamba kuma titenda yayo kwatantu, yakawina ndi chafukwa chati dambuka ndi mbotu, tikudya kumadzi kufena kwa samakandula. SVMs are mostly associated with the poor performance in the case of a large feature set or no linear relationship which may be the fuel to that low accuracy and F1-score.

## 5.5 Related Work

### *5.5.1 Comparison with Similar Projects*

The issue of detecting the harmful links via machine learning is also depicted in a few other projects. Notable examples include:

- **Phishing Detection using Machine Learning Techniques [3]**: This project applied different machine learning models as a phish URL detector, which brought us a good accuracy of 92%. Differently from this, our project has managed to improve accuracy and wide applicability by covering many kinds of harmful URL. thorough feature extraction and the customisation to URL classes will expound our thesis about the surge in the number of harmful content in almost all internet domains. g. Finally, and cybersecurity equipment (, phishing, malware, spam) enhance the efficiency of the organisation tech system.
- **deepBF: Malicious URL Detection using Learned Bloom Filter and Evolutionary Deep Learning [5]**: their project accomplished 95-98% with noise reduction and speed using a conjunction of deep learning and Bloom filters. A tree in a forest that our algorithm thus yielded showed the resulting accuracy as 98%. Both 85% and INTAPS are easy to use, the difference being that INTAPS has a real-time classification interface. Also, unified architecture highly relies on the use of ensemble methods that are known for their robustness and data domain learning abilities.
- **Malicious URL Detection based on a Parallel Neural Joint Model [12]**: The project made use of the combination of CapsNet and IndRNN with an accuracy level of 99%. Whereas the Random Forest model accurateness got slightly lower, the simplicity and interpretability of our model along with the practical web application have a great value advantage. The model easiness to install and low computing demand makes it available for everyone to use.

### *5.5.2 Strengths and Limitations Compared to Other Works*

**Strengths**:

High Accuracy and Reliability: Performance of Random Forest is not inferior neither to nor above many other models presented in the literature.

User-Friendly Interface: The Flask Web application which combines usability and reality eases the task of the human users by classifying a URL on the fly. This is a major advantage for the general non-technical audiences who might enjoy the ease of use.

Comprehensive Feature Set: The lengthy steps of preprocessing and feature extraction is a crucial step because these ensure modeling the model with rich set of features to learn about, thus, increasing its efficiency in detection. Integrating different elements of the web addresses, the model can cover various behaviors which are related to malicious activities.

**Limitations:**

Advanced Techniques: Other projects may outperform with high accuracies models such as CapsNets or IndRNNs but this will come at the cost of high design complexity and computational demands. Our model aims at the balance of performance and easiness, and looking for advanced techniques can increase the accuracy.

Real-Time Processing: Although the existing system has shown to be effective, but improvements are waiting to be made to process real time processing of considerably large datasets. Crystalising the implementation of the parallel processing or may also profiteer from fast cloud computing resources can be used to increase the system capacity.

The above mentioned can be tackled and the project can provide more helpful and user-friendly tool for finding malicious URLs, as a result, can increase cybersecurity goals.

# 6 Conclusion

## 6.1 Summary of Overall Outcome

Basically, this project aimed to create of a secure kidn of URL detection tools using different types ML algorithm. Thorough examination, sorting out and processing of data made their project successful and yielded several models with high accuracy to determine malicious URLs. The adoption of a user-friendly web interface that displays the results in real time and URL categorization was another important feature implemented, which is necessary for broad use of this model in daily routines. On the models that we tested, the Random Forest model came out as the best, superior in terms of accuracy and precision. It was ascertained that the project generally were accomplished as intended, giving the community an accurate device for cybersecurity enhancement.

## 6.2 Key Outputs and Discoveries

In the project, the models of machine learning were examined along the line of SVM, Random Forest, XGBoost, AdaBoost, LightGBM, Decision Trees, Extra Trees Classifier, and the Deep Multi-Layer Perceptron. By means of this detailed analysis, much-required information was obtained on the strong points and the weak points of the various approaches.

The project involved development of many machine learning models, the most successful of which was Random Forest model with an accuracy of 98.96% and an F1 Score of 98.86%. These values prove the efficiency of the chosen methods.

Various preprocessing and feature extraction techniques were used accordingly which yielded a rich dataset consisting of features that were useful in driving the models' improvements. It also involved full URL length, full hostname length, path length, word count using special characters, and so on.

A web application using the Flask framework was deployed. The application provides instant remedial action when any URL is input in, and the URL is unsafe. By doing that, this interface increases the practicality of the model reigning the level of usability and accessibility.

## 6.3 Reflections

Reflecting on the entire project process, several key aspects stand out:

### *6.3.1 Learning and Development*

**Data Preprocessing and Feature Engineering:** One of the most substantial part of the project was to arrive at the right dataset and extract the features which can assist in improving the model performance. This process revealed the sophisticated part of the proper data preparation and led to the significance of data in the development of machine learning projects.

**Model Evaluation and Selection:** The study conducted an evaluation and choice of the best model based on the performance measures including the accuracy accurracy and F1- score. This kind of validation pointed out that strict testing, validation, and robustness measures are a must if a model of any kind is to be used.

**User Interface Development:** Deploying the web interface using Flask happened to be a booster I was looking for to get an idea on how the machine learning algorithms may be integrated with the web applications. This goes to show how integration is critical for the application of models in real-life.

## 6.4 Challenges and Solutions:

1. Imbalanced Data: One of the difficult issues that come with the unbalanced data was the only solution. The choice not to apply the resampling methods such as SMOTE or ADASYN ended in some of the classes being under-represented which is likely to affect the model performance. Looking back, the incorporation of these techniques could have helped the model to deal with the problem of minority classes more successfully.

2. Feature Selection: It was hard to select the most critical attributes; however, applying ensemble methods: Random Forest made it possible to highlight the main features. The future project could take benefit from the adoption of more state-of-the art feature selection criteria for the sake of model accuracy.

3. Overfitting: Some models, especially those let the complicated ones, show the phenomenon of overfitting. Regularisation methods, like cross-validation, were used to solve this problem but some aspects should be advanced. Using stronger penalty terms regularly and making sure that the validation amount is sufficient will make the models more capable of generalisation.

## 6.5 Project Management and Execution:

The efficient time management was very important for the process of accomplishing different stages related to this project, they are data collection, and preprocessing, to model developments, and evaluation. Supporting a detailed project plan enabled us to meet all the deadlines on time.

## 6.6 What Could Be Done Differently

1. Incorporating Resampling Techniques: As the way to deal with data imbalance, either SMOTE or ADASYN could be a good resampling technique to enhance performance of models in which minority classes play an important role.
2. Advanced Feature Engineering: Through the use of more sophisticated feature engineering approaches, the performance of the model, for example, feature interaction and polynomial features can be modified.
3. Real-Time Processing Optimiszation: Accomplishing utilisation of the real-time optimisations, including parallel processing and cloud computing resource usage, could rise the system's scalability and efficiency.

## 6.7 Future Work

While this project has achieved significant milestones, there are several areas for future exploration and development:

### *6.7.1 Advanced Feature Engineering*

1. Feature Interactions: The study of the connections between different characteristics could reveal more knowledge and enhance the model's performance. The methods like polynomial feature generation and feature crosses should be the subject of thorough study.

2. Automated Feature Engineering: The use of automated feature engineering tools and frameworks could simplify the process of feature engineering and discover new features that improve the performance of models.

### *6.7.2 Addressing Data Imbalance*

1. Resampling Techniques: The methods like SMOTE (Synthetic Minority Over-sampling Technique) and ADASYN (Adaptive Synthetic Sampling) are the ones that can cope with data imbalance and enhance model performance, especially for classes which are not well-represented.

2. Cost-Sensitive Learning: The introduction of the cost-sensitive learning methods that make misclassifications in minority classes more expensive could be a way to resolve the issue without changing the dataset.

### *6.7.3 Exploring Advanced Models*

1. Deep Learning Models: Exploring more advanced deep learning models, like RNNs and CNNs can help the system to capture harder patterns in URL data.

2. Hybrid Models: The combination of several models into a hybrid approach, for instance the use of ensemble methods with deep learning could enhance performance and at the same time make it more stable.

### *6.7.4 Real-Time Processing Optimization*

1. Parallel Processing: The parallel processing techniques would bring about an improvement of the system's capacity to deal with huge amounts of data in real-time.

2. Cloud Computing: The cloud computing resources can be used for the scalable processing and storage, thus enhancing the efficiency and scalability of the system.

### *6.7.5 User Interface Enhancements*

1. Detailed Explanations: The classification results could be explained in details to the users, which would make them understand and trust the system more.

2. Visualisations: The visualization of URL characteristics and classification outcomes would probably make the users more interested in and understand better.

3. User Guidance: Provision of suggestions and advice using the classification results could be of great help to users in making right decisions on how to prevent risks.

### *6.7.6 Integration with Cybersecurity Frameworks*

1. Intrusion Detection Systems (IDS): On the one hand, unioning the URL awareness tool having already used intrusion detection systems could serve as a more complete cyber security solution.

2. Security Information and Event Management (SIEM): As part of the project it will be possible to integrate the system into SIEM platforms that with immediate detecting and responses to the threats there is.

### *6.7.7 Continual Learning and Adaptation:*

1. Incremental Learning: The machine will be capable of re-modeling its algorithms according to individual requirements by virtue of learning systems which are not the case with overlearning.

2. Adapting to Evolving Threats: The system's adaptability to the continuing, rapidly-changing cyber threats by refreshing the dataset and models may keep its effectiveness in the long term.

These points help us focus all our efforts to continuously improve the program to provide a more useful and user-friendly tool this time and contribution to security measures nowadays.

# 7 References


[1]. Vundavalli, V., Barsha, F., Masum, M., Shahriar, H. and Haddad, H., 2020, November. Malicious URL detection using supervised machine learning techniques. In *13th International Conference on Security of Information and Networks* (pp. 1-6).

[2]. Aalla, H.V.S., Dumpala, N.R. and Eliazer, M., 2021. Malicious URL prediction using machine learning techniques. *Annals of the Romanian Society for Cell Biology*, pp.2170-2176.

[3]. Shahrivari, V., Darabi, M.M. and Izadi, M., 2020. Phishing detection using machine learning techniques. *arXiv preprint arXiv:2009.11116*.

[4]. "APWG | Phishing Activity Trends Reports." https://apwg.org/trendsreports/

[5]. Patgiri, R., Biswas, A. and Nayak, S., 2023. deepBF: Malicious URL detection using learned bloom filter and evolutionary deep learning. *Computer Communications*, *200*, pp.30-41.

[6]. Stamp, M., Alazab, M. and Shalaginov, A. eds., 2021. *Malware analysis using artificial intelligence and deep learning* (Vol. 1). Berlin/Heidelberg, Germany: Springer.

[7]. Wang, Z., Ren, X., Li, S., Wang, B., Zhang, J. and Yang, T., 2021. A malicious URL detection model based on convolutional neural network. *Security and Communication Networks*, *2021*, pp.1-12.

[8]. Rasymas, T. and Dovydaitis, L., 2020. Detection of phishing URLs by using deep learning approach and multiple features combinations. *Baltic journal of modern computing*, *8*(3), pp.471-483.

[9]. Chen, Z., Liu, Y., Chen, C., Lu, M. and Zhang, X., 2021. Malicious URL detection based on improved multilayer recurrent convolutional neural network model. *Security and Communication networks*, *2021*, pp.1-13.

[10]. Lee[1], O.V., Heryanto, A., Ab Razak, M.F., Raffei, A.F.M., Phon, D.N.E., Kasim, S. and Sutikno, T., 2020. A malicious URLs detection system using optimization and machine learning classifiers. *Indonesian Journal of Electrical Engineering and Computer Science*, *17*(3), pp.1210-1214.

[11]. Al-Ahmadi, S., 2020. A deep learning technique for web phishing detection combined URL features and visual similarity. *International Journal of Computer Networks & Communications (IJCNC) Vol*, *12*.

[12]. Yuan, J., Chen, G., Tian, S. and Pei, X., 2021. Malicious URL detection based on a parallel neural joint model. *IEEE Access*, *9*, pp.9464-9472.

[13]. Alsaedi, M., Ghaleb, F.A., Saeed, F., Ahmad, J. and Alasli, M., 2022. Cyber threat intelligence-based malicious URL detection model using ensemble learning. *Sensors*, *22*(9), p.3373.

[14]. "Classification/data/URL · main · Courses / Adv_Security · GitLab," *GitLab*. [Online]. Available: https://git.ssl.qom.ac.ir/courses/avd_sec/-/tree/main/Classification/data/URL

[15]. Cui, B., He, S., Yao, X. and Shi, P., 2018. Malicious URL detection with feature extraction based on machine learning. *International Journal of High Performance Computing and Networking*, *12*(2), pp.166-178.

[16]. Banik, B. and Sarma, A., 2018. Phishing URL detection system based on URL features using SVM. *International Journal of Electronics and Applied Research*, *5*(2), pp.40-55.

[17]. Vanhoenshoven, F., Nápoles, G., Falcon, R., Vanhoof, K. and Köppen, M., 2016, December. Detecting malicious URLs using machine learning techniques. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-8). IEEE.

[18]. Weedon, M., Tsaptsinos, D. and Denholm-Price, J., 2017, June. Random forest explorations for URL classification. In *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)* (pp. 1-4). IEEE.

[19]. Chen, Y.C., Ma, Y.W. and Chen, J.L., 2020, July. Intelligent malicious URL detection with feature analysis. In *2020 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1-5). IEEE.

[20]. Khan, F., Ahamed, J., Kadry, S. and Ramasamy, L.K., 2020. Detecting malicious URLs using binary classification through ada boost algorithm. *International Journal of Electrical & Computer Engineering (2088-8708)*, *10*(1).

[21]. DR, U.S. and Patil, A., 2023, January. Malicious URL Detection and Classification Analysis using Machine Learning Models. In *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)* (pp. 470-476). IEEE.

[22]. Patil, D.R. and Patil, J.B., 2018. Malicious URLs detection using decision tree classifiers and majority voting technique. *Cybernetics and Information Technologies*, *18*(1), pp.11-29.

[23]. Anusree, A., Jose, B., Anilkumar, K. and Lee, O.T., 2021, October. Phishing detection using extra trees classifier. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-6). IEEE.

[24]. Chang, P., 2022. Multi-layer perceptron neural network for improving detection performance of malicious phishing URLs Without Affecting Other Attack Types Classification. *arXiv preprint arXiv:2203.00774*.

[25]. Anguita, D., Ghio, A., Greco, N., Oneto, L. and Ridella, S., 2010, July. Model selection for support vector machines: Advantages and disadvantages of the machine learning theory. In *The 2010 international joint conference on neural networks (IJCNN)* (pp. 1-8). IEEE.

[26]. Ali, J., Khan, R., Ahmad, N. and Maqsood, I., 2012. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), p.272.

[27]. Saxena, V. and Aggarwal, A., 2020, December. Comparative study of select non parametric and ensemble machine learning classification techniques. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (pp. 110-115). IEEE.

[28]. Masih, N., Naz, H. and Ahuja, S., 2021. Multilayer perceptron based deep neural network for early detection of coronary heart disease. *Health and Technology*, *11*, pp.127-138.

[29]. "tld — tld 0.13 documentation." [Online]. Available: http://tld.readthedocs.io/

[30]. "Welcome to LightGBM's documentation! — LightGBM 4.0.0 documentation." [Online]. Available: https://lightgbm.readthedocs.io/en/stable/

[31]. "Building From Source — xgboost 2.1.0-dev documentation." [Online]. Available: https://xgboost.readthedocs.io/en/latest/build.html

[32]. "scikit-learn: machine learning in Python — scikit-learn 1.4.2 documentation." [Online]. Available: https://scikit-learn.org/stable/

[33]. "pandas documentation — pandas 2.2.2 documentation." [Online]. Available: https://pandas.pydata.org/docs/

[34]. "Welcome to Flask — Flask Documentation (3.0.x)." [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[35]. "pickle — Python object serialization," *Python documentation*. [Online]. Available: https://docs.python.org/3/library/pickle.html