International Journal of Scientific and Management Research
Volume 07 Issue 04 (April) 2024
ISSN: 2581-6888
Page: 27-63

Phishing Detection Using Machine Learning: A Model Development and Integration

Valentine Adeyemi Onih
Department of Cybersecurity, University of Hertfordshire, Hertfordshire, **United Kingdom**
DOI - http://doi.org/10.37502/IJSMR.2024.7403

Abstract

This study aimed to develop a robust machine learning-based phishing detection system using algorithms such as K-nearest neighbour (KNN), artificial neural network (ANN), and random forest (RF). It utilised datasets from Ariyadasa et al. (2021) and UNB (2016) to discern patterns distinguishing legitimate from phishing websites. Furthermore, an objective was to integrate the optimal model into a Django-based web application, facilitating real-time phishing detection. A comprehensive literature review on phishing detection techniques was also undertaken.

Datasets chosen underwent rigorous pre-processing to address missing values and imbalance. Feature selection was achieved manually and automatically using mutual information classification. Three machine learning algorithms, RF, KNN, and ANN, were explored. Their hyper-parameters were optimised using GridSearchCV.

Performance results highlighted RF's accuracy at 99.78%, KNN's at 99.67%, and ANN's at 99.11%. While RF and KNN models perfectly identified legitimate websites, ANN showcased an impeccable detection of phishing websites. The RF model, with the highest accuracy, was integrated into a Django application, providing a user interface for real-time phishing detection.

All models exhibited high accuracy rates, demonstrating their efficacy in phishing detection. While RF was integrated into the web application for this study, the choice between models depends on specific user or business requirements and priorities. Feedback mechanisms within the Django application further promise refinement in future recommendations. The study provides a foundational step toward enhancing web safety through effective phishing detection.

Keywords: RF, ANN, KNN, datasets, web, Django.

1. Introduction

In today's digital environment, phishing attacks continue to present a serious and increasing threat to people and organisations (figure 1.1). These attacks involve cybercriminals attempting to trick consumers into submitting sensitive information by establishing fake websites that look authentic [3]. Phishing attacks can have serious repercussions, including financial losses, data breaches, and reputational harm [4]. Therefore, it is essential to provide efficient techniques for recognising and combating these threats.

The latest 2023 phishing reports from reputable sources shed light on the alarming surge in phishing attacks and the need for robust countermeasures. According to Zscaler ThreatLabz's

phishing report, there has been a staggering 47.2% increase in phishing attacks in 2022 compared to the previous year. These attacks are becoming increasingly sophisticated, exploiting vulnerabilities in email, SMS, and voice communications. The education sector witnessed a dramatic surge of 576% in phishing attacks, while the retail and wholesale sector experienced a 67% drop [27, 61].

The State of the Phish report by Proofpoint further emphasizes the evolving tactics employed by threat actors. It reveals that more than 30 million malicious messages in 2022 involved Microsoft branding or products, exploiting the trust associated with familiar branding (Figure 1.2). Additionally, there was a 76% increase in direct financial losses resulting from successful phishing attacks [62]. These statistics highlight the need for continuous advancements in phishing detection and prevention techniques.

With the proliferation of phishing attacks, it is imperative to understand cybercriminals' underlying motivations and methods. The [61] identifies several targeted industries and brands. Education emerged as the most targeted industry, while Microsoft brands, such as OneDrive and Sharepoint, along with the crypto exchange Binance, were among the most frequently targeted entities [61]. By examining these patterns, organizations can enhance their cybersecurity strategies and strengthen their defenses against phishing attacks.

The emergence of machine learning as a powerful tool for detecting patterns in data has opened new possibilities for phishing detection. Although phishing attacks exhibit unique features, they share common traits and patterns [28, 29]. By utilising the power of machine learning, organisations can detect phishing websites based on their characteristics and shield users from falling victim to these trickery schemes. Machine learning algorithms, such as the K-Nearest Neighbours (KNN), Artificial neural network (ANN) and the Random Forest (RF), have shown promise in identifying and classifying phishing websites [29].

As a result, there is an urgent need to create better detection and prevention techniques given the rise in phishing attempts. The numbers and trends in the 2023 phishing reports (figure 1.1), demonstrate the demand for ongoing study and development in the cybersecurity sector. Organisations can improve their defence and reduce the dangers brought on by phishing attempts by utilising machine learning techniques and studying the characteristics of phishing websites.

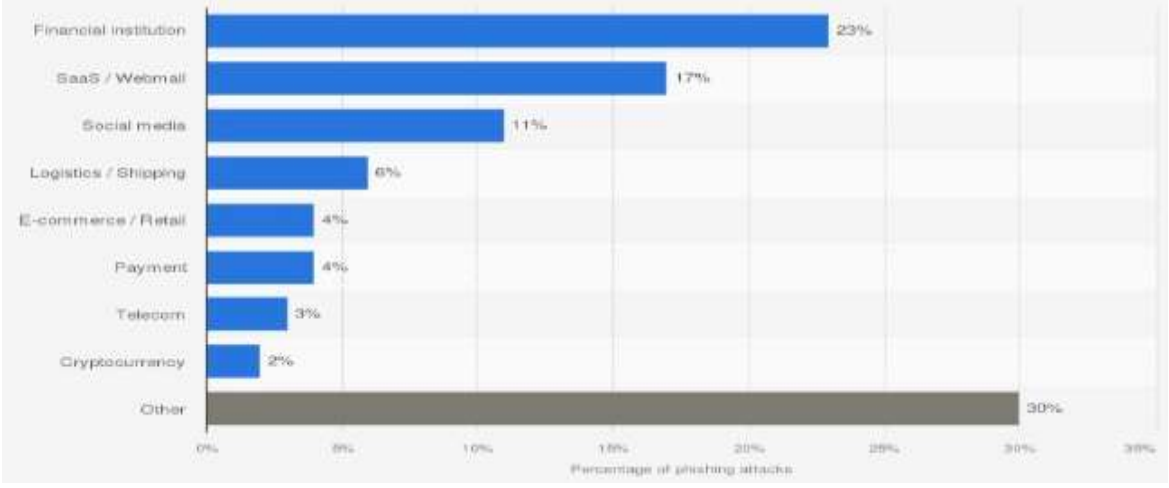


Figure 1.1: Trends of phishing attacks per industry ([61])

This research paper aims to provide a comprehensive and analytical evaluation of different machine learning and deep learning methods for detecting phishing websites. The primary focus will be on the KNN, ANN and the RF algorithm. By comparing and analyzing these models and integrating the best performing model into the Django web application, this study seeks to contribute to the advancement of phishing detection methods and provide insights for developing robust machine learning models to combat phishing attacks effectively.

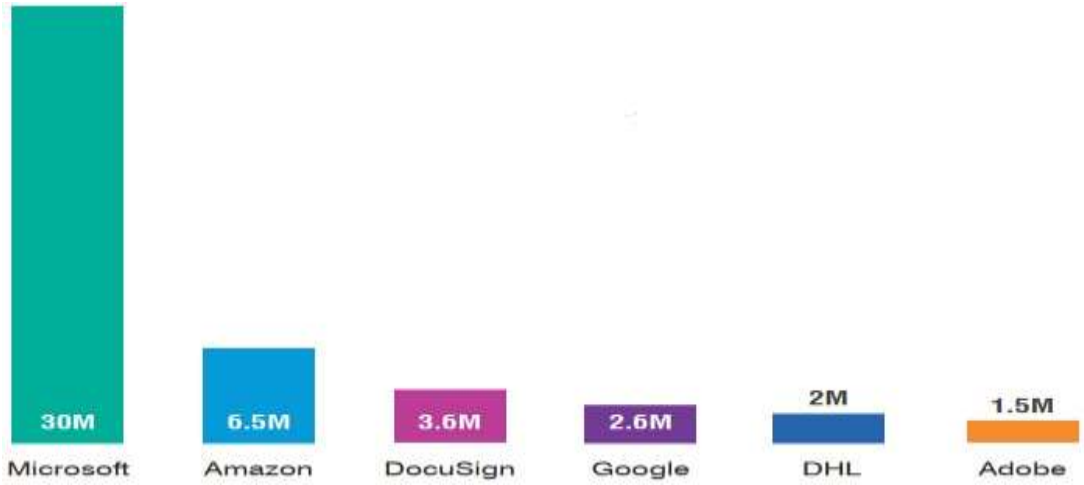


Figure 1.2: Cyber-attacks which involve Brand Abuse in 2022 ([62])

1.1 Background of The Study

Phishing attacks have become a significant threat in the digital landscape, targeting individuals, organizations, and even governments. These attacks involve the creation of deceptive websites that imitate legitimate ones, with the aim of tricking users into disclosing sensitive information such as login credentials, financial details, or personal data. The prevalence of phishing attacks is evident from the alarming statistics reported by reputable sources. The U.S. Federal Bureau of Investigations Internet Crime Complaint Center (IC3) highlighted in their 2020 Internet Crime Report that internet-based theft, fraud, and exploitation resulted in substantial financial losses [63]. Furthermore, Symantec reported a significant rise in phishing attempts worldwide, underscoring the urgency to develop effective detection mechanisms.

To combat the ever-evolving nature of phishing attacks, various techniques have been employed, including network-level protection, authentication, client-side tools, user education, server-side filters, and classifiers. However, these approaches often encounter challenges in accurately identifying and distinguishing phishing websites from legitimate ones. The advent of machine learning (ML) algorithms has opened new possibilities for detecting patterns and traits associated with phishing attacks. ML models have demonstrated their potential in identifying common characteristics of phishing websites, contributing to the overall detection accuracy.

1.2 Rationale Behind the Study

The increasing threat of phishing attacks poses significant risks to individuals and organizations worldwide. While previous research has explored the use of machine learning techniques for phishing detection, there is still a pressing need for further investigation and improvement in the accuracy and effectiveness of these models. Therefore, the rationale behind

this study is to address the following key aspects in order to advance the field of phishing detection:

1. Enhancing Detection Accuracy: The primary objective of this research is to develop an improved detection model for phishing websites by utilizing the K-Nearest Neighbors (KNN), artificial neural network (ANN) and the Random Forest (RF) algorithm. By focusing on these specific machine learning techniques, the study aims to enhance the accuracy of phishing detection then integrate the best performing model into the Django web application. This improvement will facilitate the timely identification and mitigation of potential threats, thereby reducing the common damages associated with phishing attacks.
2. Dataset for Comprehensive Analysis: To achieve a more comprehensive analysis, this study utilizes the datasets developed by [33] and [34] which will help to classify as either legitimate or phishing websites. By leveraging this dataset, the study aims to capture a wider range of phishing website characteristics, including various deceptive tactics, structural patterns, and content features. This comprehensive approach will enable the ML model to learn from a more diverse set of examples, thereby improving its effectiveness in detecting previously unseen and evolving phishing attacks.

By examining the existing approaches employed by security experts in addressing phishing detection challenges, as well as optimizing the KNN, ANN and the RF algorithms, then followed by the integration of the best performing model into the Django web application, this research seeks to make significant contributions to the advancement of phishing detection methods. The findings of this study will provide valuable insights into the development of robust machine learning models specifically designed to identify and combat phishing attacks.

The implications of this research are far-reaching. The improved detection model resulting from this study will not only benefit individuals and organizations by providing them with enhanced protection against phishing attacks but will also contribute to the broader field of cybersecurity. The knowledge gained from this research can be leveraged to inform the development of more sophisticated and proactive defense mechanisms against evolving cyber threats.

Hence, the rationale behind this study lies in the pressing need to enhance the accuracy and effectiveness of phishing detection models and to integrate the best performing model in the Django web application. By focusing on the optimization of the developed model, and utilizing the datasets for comprehensive analysis, this research aims to make significant contributions to the field of phishing detection. The outcomes of this study will provide valuable insights and tools for effectively identifying and combating phishing attacks, thereby contributing to the advancement of cybersecurity.

SCOPE OF THIS RESEARCH

It is important to mention that there are many forms of phishing attacks that exists on the digital world. However, this research is streamlined to address phishing attacks related to websites, hence, two datasets one developed by [33] and the other by the [34] will be used for the development of machine learning models for phishing attacks detection and although there are different web application technologies that can be used for integration, this research will use the Django framework to integrate the best performing model into the web application.

AIMS AND OBJECTIVES

1. Develop a robust machine learning-based phishing detection system: The primary objective of this project is to create a sophisticated machine learning model using KNN, ANN, and RF algorithms to accurately classify and detect phishing websites. By analyzing a diverse set of features extracted from legitimate and phishing websites using the dataset provided by [33] and [34], all the models aim to identify patterns and characteristics that differentiate between legitimate and phishing websites.
2. Implement an intuitive web interface for real-time phishing detection: The second objective is to integrate the best performing machine learning model into a Django-based web application. The application will provide users with a user-friendly interface for input and receive real-time information on the likelihood of phishing nature of the input.
3. Conduct a comprehensive literature review on phishing detection techniques: In addition to the model development and implementation, this project aims to conduct a literature review to explore existing techniques, algorithms, and methodologies for phishing detection. By analyzing relevant academic papers, articles, and industry reports, the research will provide valuable insights into the current state-of-the-art in phishing detection and identify potential areas for further improvement.

RESEARCH QUESTIONS

- How can a machine learning model be trained and optimized to effectively detect phishing attacks?
- How can the phishing detection model be integrated into existing web browsers for real-time protection?
- What are the performance metrics and evaluation techniques suitable for assessing the effectiveness and efficiency of the developed phishing detection model?

2. Literature Review

The literature review aims to thoroughly understand the existing ideas, approaches, and research in relation to this research topic. To identify knowledge gaps and comprehend, the chapter will study and analyse different novelty research papers from different academic databases. The hazards and repercussions of phishing attacks are highlighted, helping to show the significance and relevance of phishing website detection. It gives background information on the various phishing methods and the web application framework related to this research.

The literature review also examines numerous methods and procedures used in phishing website detection, including machine learning algorithms, data mining, feature extraction, and classification techniques. It examines the benefits and drawbacks of various detection models and assesses how well they perform. This research chapter is concluded with the proper identification of the gaps of research.

2.1 Conceptual Review

The ideas of phishing detection, machine learning, the Django Framework, and Django web security are fundamental to building secure web applications [12]. To thwart phishing assaults and create safe web applications, it is crucial to comprehend the fundamental ideas behind phishing detection, machine learning, the Django Framework, and Django web security.

Phishing detection

[36] defined phishing as a social engineering assault by cybercriminals to gain a user's personal information, including usernames, passwords, and credit card numbers. There are many different kinds of phishing assaults, including but not limited to spoofing, malware-based phishing, DNS-based phishing, data theft, email/spam, web-based delivery, and phone phishing.

[37] defined phishing detection as seeing and stopping phishing assaults, which use deceptive methods to coerce people into disclosing personal information or carrying out destructive deeds. [38] noted that detection methods aim at Phishing detection techniques, such as email filtering, URL analysis, machine learning algorithms, and reputation-based systems, using different strategies.

2.2.1 Machine Learning

Machine learning is a branch of artificial intelligence that focuses on creating models and algorithms to learn from data and make predictions or judgements without being explicitly programmed [39]. Machine learning algorithms are trained on big datasets.

comprising attributes taken from well-known phishing and legitimate websites in the context of phishing detection. These algorithms develop the ability to recognise trends, oddities, and phishing attack signs, allowing for the precise classification of dubious websites. The goal of the learning system is to derive a description of a given concept from a collection of examples that the teacher has provided and from examples of concepts that can be advantageous. The vocabulary used to describe examples and concepts is described in background knowledge. For instance, it might comprise predicates, auxiliary syntactic rules, subjective preferences, and possible variables (attributes) values hierarchy. The learning algorithm then builds upon the type of instances, the volume and importance of the background knowledge, the representational issues, the assumed nature of the concept to be learned, and the designer's experience. Figure 2.1 shows different forms of machine learning with their respective domains.

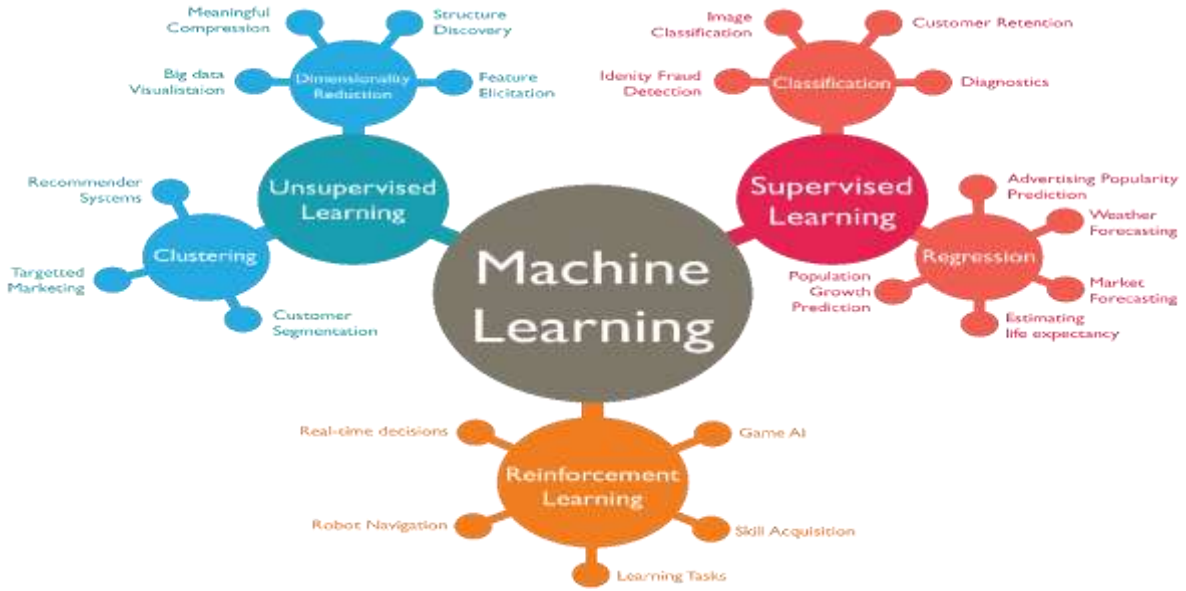


Fig 2.1 Different types of ML (Source: [30])

Django Framework

According to [40], the Django Framework is a high-level Python web framework that streamlines and accelerates the creation of web applications. It offers various tools, libraries, and functions and adheres to the model-view-controller (MVC) architectural pattern. Django provides tools that help developers create scalable and safe web applications, including URL routing, template rendering, database administration, and user authentication. It encourages robustness and maintainability because of its focus on clean, reusable code.

Django Web Security

The Django Framework's practises, processes, and features that assist in defending online applications against various security risks collectively make up Django Web Security [41]. Cross-site scripting (XSS) and cross-site request forgery (CSRF) threats are guarded against by security features included in Django. It offers secure session management, password hashing, access control techniques, user authentication, and authorisation. Additionally, Django makes input validation, secure form processing, and defence against SQL injection attacks possible. Mastery of Django's settings, customisation, and proper application of all offered methods and approaches elevates backend engineers to the next level of proficiency. SQL queries from web frameworks to databases are a thing of the past.

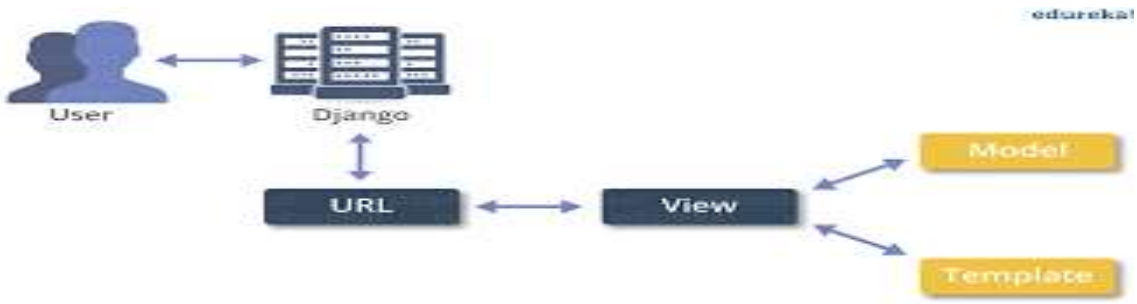


Figure 2.2 General security structure of Django technology (Source: [41])

Fundamental principles and theories underpinning phishing detection techniques and machine learning algorithms.

In order to recognise and stop phishing assaults, machine learning algorithms and phishing detection techniques are essential. Comprehending the fundamental ideas and principles underlying these approaches and algorithms is crucial to develop an efficient and precise phishing detection system. Social engineering is a critical component of phishing assaults, which uses psychological tricks to trick people into taking particular activities [42]. Designing efficient detection techniques for phishing attempts requires an understanding of the psychological and behavioural components of social engineering. [43] explained that phishing attacks typically occur through emails. Phishing detection methods examine email headers, content, and metadata to spot suspect patterns like misspellings, polite greetings, dubious URLs, or suspicious email addresses. These algorithms also consider the sender's reputation, email structure, and contextual meaning to distinguish between genuine and phishing emails. Phishing websites sometimes resemble trustworthy websites to deceive users into disclosing personal information. Techniques for analysing web pages look at their text, HTML structure, visual components, and URL properties to spot phishing red flags such as dubious redirects,

shady designs, and false login forms. This study aids in identifying legitimate websites from fraudulent ones.

[44] posited that supervised learning algorithms are the cornerstone of many phishing detection systems and are a type of machine learning algorithm. These algorithms are trained on labelled datasets containing features from known legitimate and phishing events. In order to accurately classify unknown instances, the model learns patterns, relationships, and decision limits that distinguish between actual and phishing occurrences. Feature engineering is choosing and extracting pertinent properties or traits from data that can distinguish between phishing and genuine activity [45]. This process includes analysing URL attributes, webpage content, email headers, metadata, and other pertinent elements. Feature engineering ensures that the machine learning system receives accurate detection by creating informative input. Ensemble approaches mix various machine learning models or algorithms to increase detection accuracy. An ensemble of models is built using methods like bagging, boosting, and stacking to make judgements as a group [46]. Ensemble approaches improve phishing detection systems' performance by reducing individual algorithms' drawbacks. Online learning algorithms adjust and update in real-time as new data becomes available. Systems detecting phishing frequently work in dynamic contexts where new phishing methods and patterns are constantly emerging. Online learning algorithms enable continuously updating and modifying the model, keeping the detection system current and effective against changing phishing attempts.

Algorithms for anomaly detection find cases that drastically depart from predicted patterns or the norm [47]. Anomaly detection techniques can find anomalous behaviour, trends, or traits that point to a possible phishing attack in phishing detection. These algorithms add new and previously undiscovered phishing instances, typically supervised learning methods. Building reliable and effective systems requires understanding the ideas and principles underlying machine learning algorithms and phishing detection approaches. By utilising these concepts, researchers and professionals can create complex detection models and systems capable of quickly recognising and reducing the hazards presented by phishing attempts.

Challenges and limitations of existing approaches in detecting phishing attacks.

The effectiveness of current methods for detecting phishing assaults is hampered by several issues and limitations that increase the hazards involved with phishing. [48] noted that attackers who use phishing constantly modify their strategies to get around detection measures. They use sophisticated methods like spear phishing, smishing, and vishing, which make it difficult for conventional detection tools to keep up with the changing attack vectors. Systems for detecting phishing must regularly upgrade their algorithms and feature sets to recognise new and advanced phishing techniques.

Phishing attempts frequently employ polymorphic behaviour, in which the emails' or websites' content and structure alter on the fly to avoid detection [47]. Because of this, it is challenging for static rule-based or signature-based techniques to identify phishing attacks correctly. More sophisticated methods, such as machine learning algorithms that can learn and adapt to changing patterns, are needed for identifying and categorising polymorphic asses. Zero-day attacks are newly identified vulnerabilities or attack strategies that have not yet been patched. [49] argued that phishing attackers might use zero-day vulnerabilities to perform sophisticated, covert attacks. Until the vulnerabilities are found and fixed, existing detection techniques might not be able to recognise and counteract such assaults.

[50] contended that phishing attempts frequently employ contextual data to trick users successfully. In order to make their attacks seem more credible, attackers may include personal information, current affairs, or data acquired from social media. Traditional detection methods could have trouble capturing and analysing this contextual data, limiting their capacity to distinguish between legitimate and phishing attacks. However, according to [51], impersonation is the most common tactic used in phishing attempts by attackers to fool consumers into believing they are dealing with legitimate businesses, brands, or people. It can be challenging to identify sophisticated impersonation schemes, especially when attackers carefully mimic reliable communication channels, email addresses, or websites. It may be challenging for current detection techniques to distinguish between real and fake entities.

A balanced and representative dataset with phishing and legal occurrences is needed to build efficient machine-learning models for phishing detection. However, given the scarcity of actual phishing events, gathering enough labelled phishing data can take time and effort. This data imbalance may impact the effectiveness and precision of machine learning-based detection methods. Even if detection tools are available, user awareness and education are essential to avoid successful phishing attempts.

2.2 Theoretical Review

This section discuss the theoretical review of this research. In this section, existing therories, models relevant to the research domain are documented.

Existing theories and models relevant to phishing detection and machine learning.

Creating efficient detection systems is aided by the ideas and models already in existence in machine learning and phishing detection [52]. However, a well-liked machine learning model for phishing detection uses decision trees [53]. They base their decisions on feature values using a hierarchical structure of nodes and branches. Decision tree models help spot trends suggestive of phishing attacks because they can capture complex decision boundaries and are interpretable. Ensemble approaches like Random Forests, and Gradient Boosting use many decision trees to increase detection accuracy. Decision trees can be used to categorise cases as phishing or legitimate in the context of phishing detection based on characteristics like URL properties, webpage content, or email attributes. Users can comprehend the decision-making process and recognise the factors that contribute to the classification thanks to their transparency and interpretability. However, decision trees may have drawbacks include overfitting, sensitivity to minute data changes, and challenges capturing complicated relationships. Pruning, ensemble approaches, and feature engineering are a few ways that can be used to overcome these restrictions. Researchers can create accurate, more understandable phishing detection systems by utilising decision trees and the associated approaches.

The application of machine learning algorithms in phishing detection

K-Nearest Neighbours (KNN) is a straightforward and understandable machine-learning technique that may be utilised for phishing detection and other classification-related tasks [55]. By comparing examples to their k-nearest neighbours in the training dataset using a distance metric, it categorises them. By considering elements like URL properties, webpage content, or email characteristics, KNN has been used to detect phishing. The algorithm determines how far off the features of the unlabeled examples in the training set are from those of the unlabeled instances. The majority class of the unknown instance's k-nearest neighbours is then used to

classify it. KNN is appropriate for beginning investigation or as a baseline model in phishing detection research since it is reasonably simple to develop and interpret.

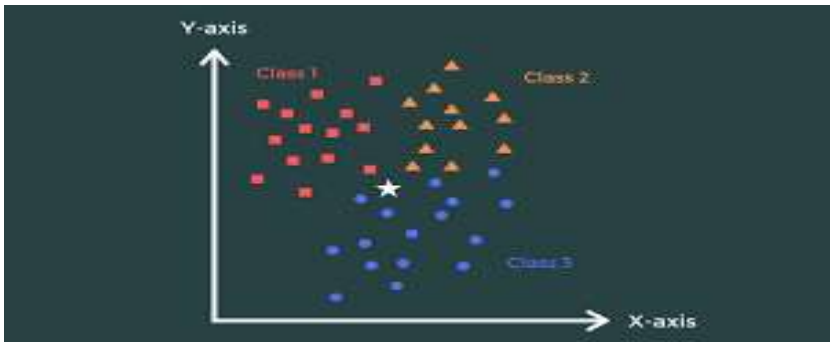


Figure 2.3: KNN in space (Source: [55])

Artificial Neural Networks (ANNs): Deep learning models of ANNs, in particular, have attracted much attention and success in several fields, including phishing detection. The interwoven layers of artificial neurons that make up the human brain inspire the design and operation of ANNs [56]. ANN models efficiently capture complicated linkages in phishing attempts because they can automatically learn elaborate patterns and representations from data. In phishing identification, ANNs have been used to analyse factors like URL attributes, email text, and webpage content. In order to accurately classify cases of phishing, the models learn hierarchical representations of these features. Within the ANN framework for phishing detection, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are often utilised architectures.

Multiple decision trees are used in Random Forest (RF), an ensemble learning technique, to increase classification accuracy. Each tree is trained using a separate portion of the training data, and the trees' predictions are combined to get the final prediction [57]. Due to its capacity to manage high-dimensional feature spaces and record complex decision boundaries, RF has been widely used to detect phishing. To spot trends suggestive of phishing assaults, the system might examine factors like URL attributes, webpage content, or email characteristics. RF can handle skewed datasets and is resistant to overfitting, making it a good choice for phishing detection tasks.

The strengths and the weaknesses of the K-nearest neighbours (KNN), rrtificial neural networks (ANN) and the random forest (RF) are documented in the table 2.1.

Table 2.1 Strengths and the weaknesses of the KNN, ANN and the RF

Algorithms	Strength	Weakness	Suitability for Research Objectives
K-Nearest Neighbours (KNN) [18, 19, 31]	Intuitive and straightforward to comprehend and apply. It does not need training time	Computation-intensive during testing since it necessitates figuring the distances to every training instance.	KNN may be appropriate for a research project if the goals emphasise clarity, interpretability, and the availability of a

	because the complete training dataset is stored. May handle classification into many classes. Works effectively with modestly large to small datasets. Robust in the face of extremes.	Responsiveness to the selection of the distance metric. It is difficult to explain the decision-making process because of a lack of interpretability. When feature spaces have several dimensions, performance may suffer. It has to choose the correct value for k carefully.	moderately substantial dataset. It can be used as a starting point for research or in situations where explainability is essential. KNN might not be the best option if the dataset is huge or has high dimensions or if computing efficiency is an issue.
Artificial Neural Networks (ANN) [21, 22, 25]	The capacity to recognise intricate patterns and relationships in data. Can handle datasets with many dimensions. Flexibility across a variety of issue arenas. Can learn from unprocessed data, eliminating the need for substantial feature engineering. It gives network architecture and activation features flexibility.	To generalise successfully, a sizable amount of labelled training data is necessary. Overfitting is a risk, especially with little training data. The high computational complexity for deep networks during training and inference. Difficulty in expressing the learnt representations and need for interpretability. Hyperparameter tweaking and model	ANNs are appropriate when the research goals call for identifying intricate linkages and patterns in the dataset. ANNs can perform well in identifying phishing websites if there is a significant amount of labelled training data and computational resources. However, it is vital to consider their complexity, high computational resource requirements, and potential interpretability issues.

		selection can be complex tasks.	
Random Forest (RF) [13, 14]	Robust handling of feature spaces with high dimensions. Robust against data noise and overfitting. Can manage datasets with imbalances. Provides feature importance rankings that make it easier to comprehend the relevance of a feature. Parallelisable inference and training.	It could be challenging to capture fine-grained decision boundaries. Uninterpretable at the level of the individual trees. The computational cost of training can be high, especially when there are many trees and complicated datasets. Random Forests may favour higher-level features or categorical variables with more categories. Random Forests	is a good fit if handling high-dimensional feature spaces, controlling unbalanced datasets, and gaining insights into feature relevance are the main research goals. While resolving problems like overfitting, RF can do well in detecting phishing websites. Alternative strategies, however, can be worth considering if interpretability at the individual tree level is essential if computational efficiency is an issue.

2.3 Empirical Review

In the research by [58], the authors worked on the evaluation of various machine learning algorithms to detect phishing websites. As part of there research, the dataset of 11,000 entries were used which were gathered from different sources including Phishtank. The primary aim was to compare machine learning methodologies with their conventional counterparts in the phishing detection, so that the clear benchmark can be established.

To fulfil this objective, they methodically appraised several machine learning algorithms, inclusive of eDRI, RIDOR, Bayes Net, SVM, and Boosting. These were subjected to the detailed data pre-processing, and their respective accuracies were documented. Notwithstanding the competitive performance of the algorithms, eDRI emerged as the most effective model, registering an impressive accuracy rate of 83%. Despite this achievement, the study fell short of offering a comprehensive solution for end-user integration.

In similar research, [59] harnessed the power of machine learning techniques to draw a distinction between legitimate websites and phishing websites. They undertook the painstaking task of prioritising features based on their contributions to the classification of URL links. The motive behind such ranking was to decipher the optimal configuration in which these features should be presented.

2.4 Gap in Literature

Upon close examination of the empirical literature, there is a noticeable lack of provision for end-user integration in the studies conducted by [58] and [59]. Even though their research demonstrated significant success in using machine learning for phishing detection, they failed to present mechanisms for incorporating their developed models into applications for the benefit of the end-users. This limitation curtails the practical applicability of these innovative solutions as they cannot be readily deployed for user protection.

Furthermore, there seems to be an overemphasis on accuracy as the principal metric for evaluating model performance. Specifically, [58] and [59] appeared to focus extensively on accuracy to the detriment of other performance indicators. In the realm of fraud detection, where datasets are frequently imbalanced, other metrics such as precision, recall, and F1-score can provide a more holistic understanding of model performance. Thus, the lack of attention given to these metrics can be considered a significant gap in these studies.

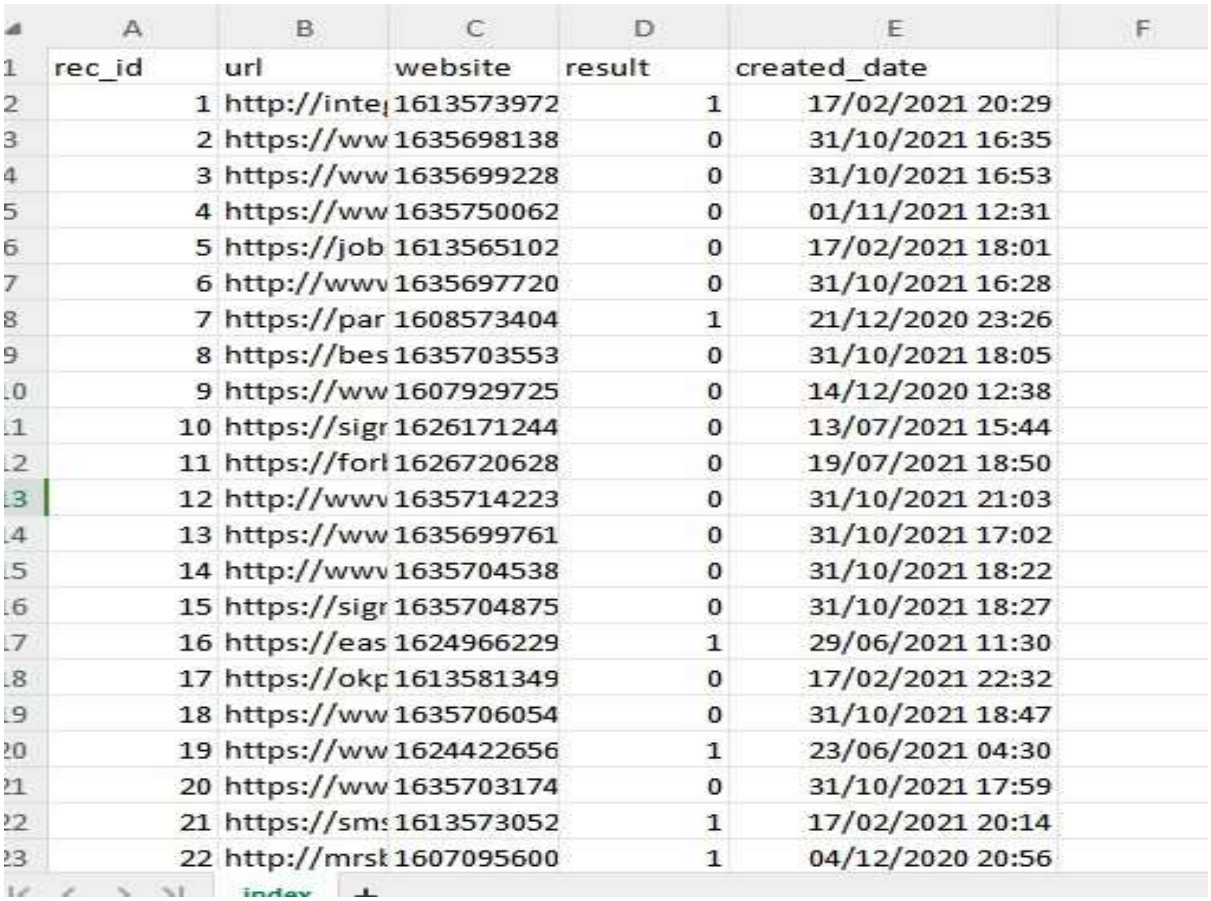
Regarding the types of machine learning techniques explored, while multiple models were examined in the works of [59] and [60], the range was still somewhat limited. Potential candidates such as K-Nearest Neighbors (KNN) and Artificial Neural Networks (ANN) were not investigated. The effectiveness of these and other algorithms in the context of phishing detection therefore remains largely untouched and under-explored in existing research.

3. Methodology

The methodology of the research provides a structured framework that serves as a guideline for the entire research process. It details the methods and techniques employed to collect, analyse, and interpret data to address the primary objectives of the study. This chapter explicates the methods and tools used to conduct this research, from data collection and pre-processing to modelling and integration.

3.1 Data Collection

In this research, two datasets were utilised. The first dataset, which is developed by [34] and from the jupyter notebook, is called with variable name "dataset". This dataset was imported from a file named "index.csv". The second dataset, with the variable named "canin", was used to load the "CanadianInstitute.csv" file which is the [33] dataset. Both datasets were read into the system using the "read_csv" method from the Python-based Pandas library. After loading, the initial entries of the datasets were visualized to have an overview of the contained data. Figure 3.1a&b shows the few screenshots of the used datasets.



	A	B	C	D	E	F
1	rec_id	url	website	result	created_date	
2	1	http://inte	1613573972	1	17/02/2021 20:29	
3	2	https://ww	1635698138	0	31/10/2021 16:35	
4	3	https://ww	1635699228	0	31/10/2021 16:53	
5	4	https://ww	1635750062	0	01/11/2021 12:31	
6	5	https://job	1613565102	0	17/02/2021 18:01	
7	6	http://wwv	1635697720	0	31/10/2021 16:28	
8	7	https://par	1608573404	1	21/12/2020 23:26	
9	8	https://bes	1635703553	0	31/10/2021 18:05	
10	9	https://ww	1607929725	0	14/12/2020 12:38	
11	10	https://sigr	1626171244	0	13/07/2021 15:44	
12	11	https://forl	1626720628	0	19/07/2021 18:50	
13	12	http://wwv	1635714223	0	31/10/2021 21:03	
14	13	https://ww	1635699761	0	31/10/2021 17:02	
15	14	http://wwv	1635704538	0	31/10/2021 18:22	
16	15	https://sigr	1635704875	0	31/10/2021 18:27	
17	16	https://eas	1624966229	1	29/06/2021 11:30	
18	17	https://okp	1613581349	0	17/02/2021 22:32	
19	18	https://ww	1635706054	0	31/10/2021 18:47	
20	19	https://ww	1624422656	1	23/06/2021 04:30	
21	20	https://ww	1635703174	0	31/10/2021 17:59	
22	21	https://sms	1613573052	1	17/02/2021 20:14	
23	22	http://mrsk	1607095600	1	04/12/2020 20:56	

Figure 3.1a The Brief Screenshot of [34] Dataset

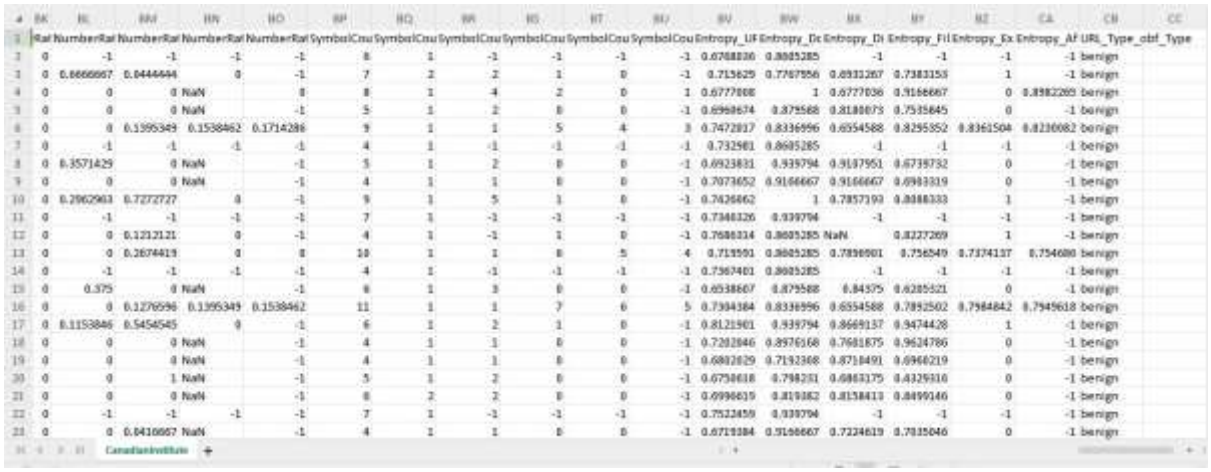


Figure 3.1b The Brief Screenshot of University of New Brunswick ([33])

3.2 Data Inspection and Pre-processing

To understand the nature of the data, the "info" method was employed on both datasets. This method revealed the number of entries, the types of data they contain, and the presence or absence of null or missing values. Visualizing missing values was further facilitated using the Seaborn library's heatmap functionality, providing a clear overview of the data's completeness.

Subsequently, the datasets were combined to derive a comprehensive data-frame. Non-essential columns such as 'rec_id', 'created_date', 'website', and 'result' were dropped to streamline the

data for further processing. The research also took into consideration that datasets might sometimes be imbalanced, which can impact the performance of many machine learning algorithms. In this context, the Seaborn library was utilised once again to visualize the balance of the result' variable, highlighting the class distribution.

Handling missing data is a vital step in pre-processing. The research adopted a strategy of replacing missing values with the mean of the respective column. This was achieved using Pandas' "fillna" method in conjunction with the "mean" function.

3.3 Feature Engineering

To rectify the disproportion noted in the dependent variable illustrated, if the dataset is imbalanced (as shown in the next chapter) the random oversampling technique were applied [9]. This approach seeks to harmonize the dataset by randomly duplicating instances of the underrepresented class until it equates the number of instances in the dominant class. Through this, it is assured that the model's expose to a balanced representation of all classes during the training phase.

Similarly, if the dataset contains entries that have very different numeric values, normalisation is applied to make the entries fall within the stipulated range. In this research, since the dataset is imbalanced, then the over sampling techniques were used to balance the dataset, similarly, StandardScaler were used to normalise the input.

3.4 Train Test Split

The development of ML models calls for the splitting of the dataset into separate training and testing sets. This practice is globally acknowledged as a robust strategy for measuring a model's performance. Typically, a considerable portion of the dataset is earmarked for training, with a lesser share allocated for testing [1]. While there are no inflexible rules for this partitioning in the ML model development, it is widely recommended to apportion a greater percentage to training dataset in relation to the testing dataset. In this research, the dataset is split such that 90% is employed for the training of the model, with the remaining 10% reserved for testing [2]. This method permits the model to acquire knowledge from an extensive volume of data, after which it is evaluated on a separate subset to affirm its capacity to generalize and deliver reliable performance on unfamiliar data [2].

3.5 Random Forest

Random Forest (RF) is a flexible, easy to use, and highly accurate machine learning algorithm that builds multiple decision trees and merges them together [13]. It is an ensemble learning method, where the predictions from multiple models are combined to improve the overall performance [14].

The scikit-learn library [15], was used to construct and train an RF model. The Random Forest Classifier function is initiated with a random seed to ensure that the model's performance is replicable across different runs [16].

The hyperparameters for the RF model are defined in the param_grid [15]. The n_estimators refer to the number of trees in the forest. Increasing the number of trees can lead to better performance but also higher computational cost. The max_depth is the maximum depth of the tree. If not defined, nodes are expanded until all leaves are pure or until all leaves contain less

than the min_samples_split. The min_samples_split represents the minimum number of samples required to split an internal node. These hyperparameters can have a substantial impact on the performance of a RF model and tuning them can lead to more accurate predictions [14].

To find the optimal combination of these hyperparameters, the GridSearchCV function [17], is used which performs an exhaustive search over the specified parameter grid. This function also performs cross-validation, dividing the dataset into a specified number of folds (in this case, 5), training the model on some folds, and validating it on the remaining fold [2]. This process is repeated until every fold has been used for validation.

The best RF model is determined by the best score achieved during the grid search [17]. This model is then used to make predictions on the test set.

3.5 K-Nearest Neighbour

The k-Nearest Neighbours (KNN) is a type of instance-based learning algorithm that predicts the class of a new sample based on the classes of its nearest neighbours in the feature space (Guo et al., 2003). The algorithm computes the distance between the new sample and every sample in the training set, selects the k nearest samples, and assigns the most common class among these k neighbours to the new sample.

The next step fine-tunes a KNN model with GridSearchCV [17]. The hyperparameters in param_grid include n_neighbors (the k in KNN), weights (the weight function used in prediction, with uniform giving equal weights to all neighbours and distance giving more weight to the closer neighbours [15].

For each combination of hyperparameters, GridSearchCV [17], performs cross-validation [2], by dividing the dataset into a specified number of folds (in this case, 5), training the model on some folds, and validating it on the remaining fold. This process is repeated until every fold has been used for validation.

The best KNN model is determined by the highest score achieved during the grid search.

3.6 Artificial Neural Network (ANN)

The research methodology employs the use of an Artificial Neural Network (ANN), a computational model inspired by the human brain's neural network. The ANN is composed of layers of interconnected nodes, also known as neurons [21]. Each neuron performs specific computations, and the results are passed onto other neurons. The weightings associated with these connections are iteratively updated in a process called learning, which takes place when the network is trained [21, 22].

The ANN in this research was constructed using [23], a high-level neural networks API. Keras provides an interface for the TensorFlow library [24], enabling the efficient building and training of neural networks.

The create_model function is used defines the architecture of the neural network, which consists of an input layer, a hidden layer, and an output layer. The input layer has a number of neurons equal to the number of features in the input data. The activation function for the neurons in the input and hidden layers is Rectified Linear Unit (ReLU), chosen for its efficiency and performance in reducing the likelihood of the vanishing gradient problem. The output layer

consists of a single neuron with a sigmoid activation function, making it suitable for binary classification problems [23].

The model is compiled with the Adam optimizer, a popular choice for training deep learning models due to its efficiency [25]. The learning rate, which controls how much to change the model in response to the estimated error each time the model weights are updated, is also specified at this point [25].

A KerasClassifier wrapper is then used to wrap the ANN model to be used in scikit-learn's GridSearchCV for hyper-parameter tuning. The parameters tuned include the number of neurons in the layers (8, 16, or 32) and the learning rate (0.0001, 0.001, or 0.01).

GridSearchCV (Belete & Huchaiah, 2021), performs an exhaustive search over the specified parameter values, with cross-validation [2]. In this case, it uses a 5-fold cross-validation, splitting the training data into 5 subsets. It trains on four of these subsets and validates on the remaining one. This process repeats five times.

Once GridSearchCV has evaluated all combinations of parameters, it identifies the set of parameters that achieved the highest score during cross-validation. After training, the best model is retrieved from the grid search results and evaluated on the test data.

3.6 Evaluation Metrics

To draw a meaningful comparison between this study and earlier research, we need to assess performance using standard metrics. This section will provide insights into various metrics used for classification, such as recall, precision, f1-score, support, and accuracy [26]. The evaluation carried out in this study is built on these key metrics:

1. Recall: This metric, reflects the ratio of correctly predicted positive instances (True Positives, TP) to the total actual positive cases, which is the sum of True Positives (TP) and False Negatives (FN). Essentially, it measures the ability of the classifier to correctly identify all the positive instances. Formula:

$$Recall = TP / (TP + FN) \qquad \text{eqn. 3.1}$$

2. Precision: Precision is a metric that quantifies the proportion of accurate positive predictions from all positive predictions made by the classifier. It is obtained by dividing the true positives (TP) by the sum of true positives and false positives (FP). Formula:

$$Precision = TP / (TP + FP) \qquad \text{eqn.3.2}$$

3. F1-score: The F1-score represents the harmonic mean of precision and recall, providing a comprehensive measure of these two aspects. Higher F1-scores suggest superior overall performance of the model. Formula:

$$F1-score = (2 * Precision * Recall) / (Precision + Recall) \qquad \text{eqn.3.3}$$

4. Support: Support denotes the total count of actual instances for each class in the test dataset. It signifies the real number of observations belonging to a specific class. Formula:

$$Support = TP + FN \qquad \text{eqn.3.4}$$

5. Accuracy: The Accuracy metric is universally employed for evaluating classification tasks. It determines the proportion of accurate predictions, including both true positives and true negatives, in relation to all predictions made by the classifier. Formula:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad \text{eqn.3.5}$$

Moreover, the Confusion Matrix is an essential tool for evaluating the efficacy of the model development [26]. Serving as a performance measurement for classification tasks in machine learning (figure 3.4), it offers a tabular depiction of actual versus predicted class labels, which enables to evaluate the model's success and identify instances of misclassification.

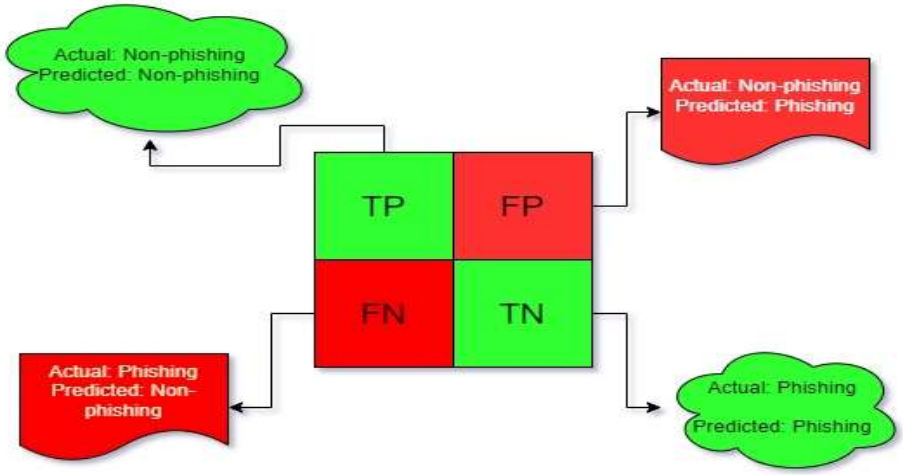


Figure 3.4 Graphical representation of confusion matrix (Source: Author).

WEB INTEGRATION

Django, a high-level Python web framework, simplifies the process of building complex web applications. It adheres to a variant of the Model-View-Controller (MVC) design pattern, called the Model-View-Template (MVT) pattern [12]. This pattern promotes a clear separation of concerns, enhancing the reusability and maintainability of the application's components [12]. Figure 3.2 shows the detailed steps from the acceptance of the URL to the prediction based on the integrated model.

In this research, Django was used to create a web-based application that integrates the best-performing machine learning model.

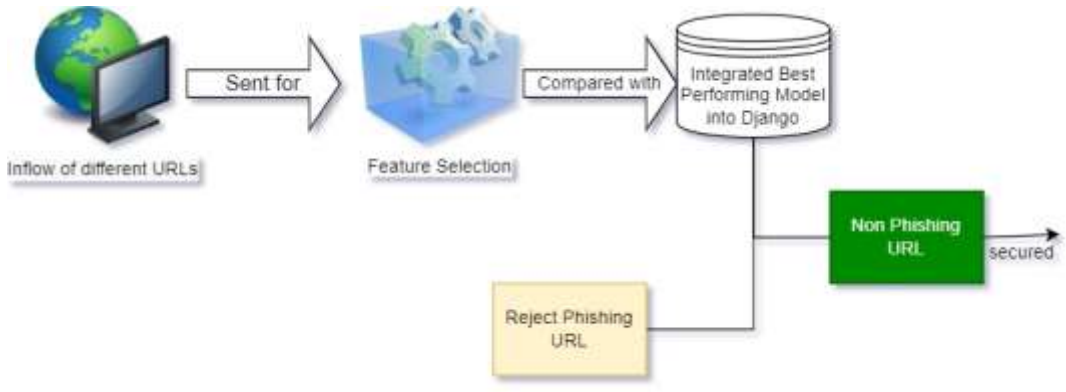


Figure 3.2. Model development to detect life cycle.

Programming and framework

Modern research, especially data-driven and computational modelling, requires choosing relevant programming languages and frameworks. It affects data intake, manipulation, analysis, and presentation, which affects research efficiency, scalability, and reproducibility. This section describes this research's programming language and frameworks.

Python: Research's Backbone

This research focuses on Python. Python is the top data science and AI language due to its ease and readability. Its rich ecosystem of libraries and frameworks makes it more versatile and efficient than most languages.

This investigation showed Python's prowess in various stages:

Python's Pandas package made data loading, cleaning, and transformation easy. DataFrames made tabular data handling easy and efficient in Pandas.

The Python-based Matplotlib and Seaborn tools were used for visual data exploration. These tools created insightful charts and heatmaps to explain data distributions and patterns.

Data Pre-processing: Scikit-learn and Imbalanced-learn helped this research with feature selection, data splitting, and class imbalances.

Machine Learning: Scikit-learn's methods, hyper-parameter tuning tools, and evaluation metrics again helped model development.

Saving Model: Save the best-performing model before using it in this research or web application. This is done with Pickle, another Python package.

Django: Used for Building the Web Application

Integrating and implementing the machine learning model into an interactive web application was the next difficulty. Django was used for this investigation.

Python-based Django's "batteries-included" approach is famous. Django provides a complete framework for building sophisticated web apps. Its ORM, user authentication, and templating engine make it ideal for incorporating machine learning models.

Django enabled this research by turning the static machine learning model into a dynamic, user-friendly programme. Figure 3.3 shows the field that allows the end user to add the URL via Django technology.

Figure 3.3 Input Field from Django Application

Jupyter Notebook: The Model Development Environment

The data visualisation and the model development environment are crucial in shaping the coding experience, and for this research, Jupyter notebook was the chosen Integrated Development Environment (IDE). This IDE comes with the anaconda navigator software which is popularly used among data scientists.

VSCode: The Web Development Environment

The web development environment is crucial in shaping the coding experience, and for this research, Visual Studio Code (VSCode) was the chosen Integrated Development Environment (IDE). VSCode, a lightweight yet powerful source-code editor developed by Microsoft, offers features like syntax highlighting, intelligent code completion, and an embedded terminal, making it an ideal choice for both Python development and Django web application creation.

Furthermore, VSCode's rich ecosystem of extensions, including those tailored for Python and Django, enhances productivity, making it easier to write, debug, and deploy code. It also facilitates a unified environment where both the data processing scripts and web application codes reside, streamlining the development workflow.

Ethical Consideration

The research's central ethos was to maintain the highest standards of ethical integrity throughout its methodology and findings. Primarily, all data utilized in the study were sourced responsibly, ensuring that no personal, sensitive, or confidential information was accessed or disseminated. The data pre-processing steps, especially the handling of missing or erroneous values, were executed transparently to avoid introducing biases. The machine learning models were trained and evaluated with fairness in mind, ensuring that no group or characteristic was unfairly represented or disadvantaged. Furthermore, while deploying the model on the Django platform, measures were taken to protect user data and privacy. During the feedback, users are informed about the data they submit, ensuring they understand its use and implications. Lastly, the research was conducted independently, free from external influence or bias, ensuring that the findings and conclusions drawn are both objective and replicable.

4. Implementation of The Practical Work and Answers to The Research Questions

In this chapter, the procedures and techniques used in the implementation of the research are elaborated upon, then answers to the identified research questions are documented. The implementation process consists of data manipulation, exploratory data analysis, model training, and the eventual integration of the developed model into a Django web application.

Implementation of the Practical Work

This section is split into the implementation of the steps which were used to handle challenges related to the dataset, this involves missing, null, entries, challenges with the choice of choosing the best features, fixing normalization, etc.

Data Pre-processing

Starting with the datasets, two primary sources were used. These datasets, sourced from the [33] and the [34], offered valuable insights into the characteristics of phishing and non-phishing websites. The dataset from [34] shows that none of the entries has missing or null value, while

that of the [33] has several missing or null values, such as Entropy_DirectoryName, Entropy_FileName, etc (Figure 4.1a). Hence, the mean value of the affected feature were used to address the challenges on the [33] dataset.

```
# total number of null value
dataset.isnull().sum()
```

```
rec_id          0
url             0
website         0
result          0
created_date    0
dtype: int64
```

```
# total number of null value
canin.isnull().sum()
```

```
Querylength                  0
domain_token_count           0
path_token_count             0
avgdomaintokenlen            0
longdomaintokenlen           0
                          ...
Entropy_DirectoryName     1826
Entropy_Filename           190
Entropy_Extension            3
Entropy_Afterpath            3
URL_Type_obf_Type            0
Length: 80, dtype: int64
```

Figure 4.1a Showing if any of the entries has missing or null value.

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80000 entries, 0 to 79999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   rec_id        80000 non-null  int64
 1   url           80000 non-null  object
 2   website       80000 non-null  object
 3   result        80000 non-null  int64
 4   created_date  80000 non-null  object
dtypes: int64(2), object(3)
memory usage: 3.1+ MB
```

```
canin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15367 entries, 0 to 15366
Data columns (total 80 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Querylength                 15367 non-null  int64
 1   domain_token_count          15367 non-null  int64
 2   path_token_count            15367 non-null  int64
 3   avgdomaintokenlen           15367 non-null  float64
 4   longdomaintokenlen          15367 non-null  int64
 5   avgpathtokenlen             15096 non-null  float64
```

Figure 4.1b General information of the [34] and [33] dataset

Figure 4.1b shows that the entries of the [34] dataset is 80000 with only 5 features (columns) while that of the [33] dataset is 15367 entries with 80 features. This explains more evidence for the imbalanced nature of the dataset when combined (figure 4.2).

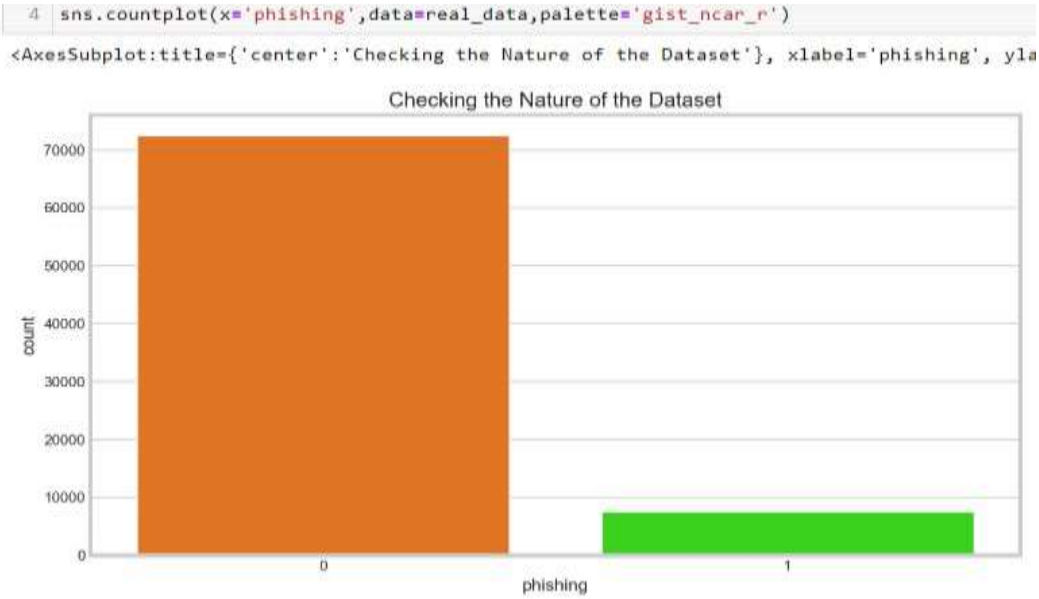


Figure 4.2 checking if the combined dataset is balanced or not balanced to know the next step to perform on them

Examining the balanced nature of a dataset is crucial because an imbalanced dataset, where one class significantly outnumbers others, can lead to a machine learning model that performs poorly, particularly on the minority class [5, 6]. The figure 4.2 shows that the combined dataset is imbalanced. In this research, oversampling was employed (see section 4.1.3), as it enhances the model's performance by replicating instances from the minority class, thereby balancing the class distribution and allowing the model to learn more about the minority class characteristics [7, 8, 11].

FEATURE SELECTION

In this research, manual and automatic feature selection were applied.

Manual Feature Selection

Based on the two datasets provided, it was determined that certain features are common to both phishing and non-phishing websites. As a result, these common features were removed from the dataset, as they do not provide any discriminatory power in distinguishing between the two classes of websites. The features that were removed include rec_id, website, created_date and result. The features such as rec_id and created_date are deleted because while rec_id only represents the serial number of the entries, the created_date documents when each entries were created, this explains why these features are irrelevant to the model development. However, result is the target variable for the [34] dataset, but since the features of the [33] dataset are far more than that of the [34] datasets (figure 4.1b), this explains that the target variable of the [33] dataset will cover wider range of prediction, hence phishing, which is the target variable of [33] were used and the result of the [34] dataset were dropped. On the similar note, the dataset author [34], explains that the website has some html page of each websites which makes this feature unnecessary because two different websites (real and clone websites) can have the same html page [9, 10]. On the contrary, the url were used because every websites have their own unique URL.

Figure 4.3a shows the remaining features when manual selection were applied.

```
new_data=combined.iloc[:,:].drop(['rec_id', 'created_date', 'website', 'result'], axis=1)
new_data.head()
```

	url	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen	avgpathtokenl
0	http://intego3.info/EXEL/index.php	0.0	2.0	12.0	5.5	8.0	4.0833
1	https://www.mathopenref.com/segment.html	0.0	3.0	12.0	5.0	10.0	3.5833
2	https://www.computerhope.com/issues/ch000254.htm	2.0	2.0	11.0	4.0	5.0	4.7500
3	https://www.investopedia.com/terms/n/next-elev...	0.0	2.0	7.0	4.5	7.0	5.7142
4	https://jobs.emss.org.uk/lcc.aspx	19.0	2.0	10.0	6.0	9.0	2.2500

5 rows × 81 columns

Figure 4.3a Remaining features after manual feature selection.

Automatic Feature Selection

Since it is not clear what other remaining 81 features represent, we used the concept of mutual information classification algorithm [26]. Using this algorithm, we were able to get relationship between our dependent variable and independent. Figure 4.4 below shows the diagram of the result.

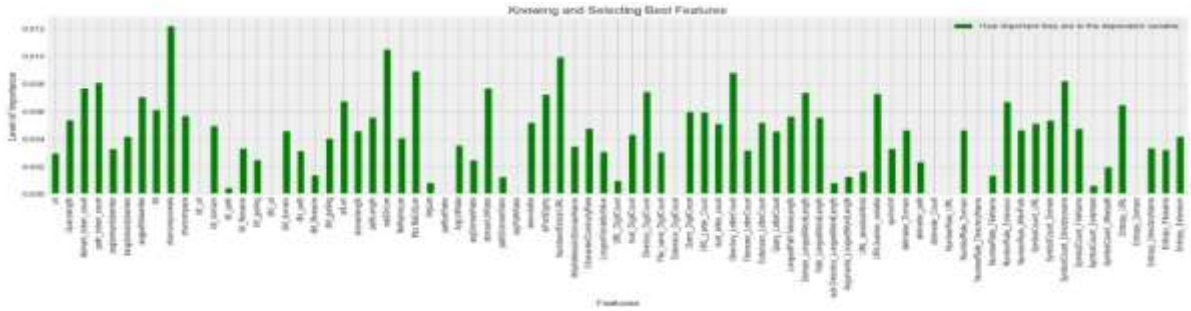


Figure. 4.4 Feature selection using the mutual information which helps to rank the features according to importance

After the removal of the unwanted features, it is important to check if the remaining features have the string (or character) representation. As shown in figure 4.5, URL entries are represented in character and the figure 4.5 also confirms that the only feature in the dataset that is represented with characters is URL

```
cols = new_data.columns[new_data.dtypes.eq('object')]
cols
```

```
Index(['url'], dtype='object')
```

Figure 4.5 Columns with character

FEATURE ENGINEERING

After determining the presence of an imbalanced dataset (figure 4.2), the Random Over Sampler function were employed to equalize the representation of all classes in the training set (figure 4.6). This ensured a fair chance of accurately predicting each class by preventing the model from being biased towards the dominant class [52]. During this phase, the X_train and y_train datasets were subjected to oversampling, wherein the minority class's data points were randomly selected and duplicated until a balance was achieved, thereby fostering a more equitable learning environment during model training.

Following the balanced nature of the dataset achieved through random oversampling, using the StandardScaler from the sklearn.preprocessing module, normalization were achieved since

many entries of the dataset as shown from the figure 4.3a largely varies. This procedure was pivotal in standardising the numerical values in the dataset to fall within a specific range, enhancing the model's predictive accuracy. It works by deducting the mean and dividing by the standard deviation for each data point in the features set [26]. This approach is revered for fostering speed and efficiency in the model training process, as it mitigates the challenges associated with disparate data values, which could potentially skew the results.

As shown in figure, 4.2, the imbalanced nature of the dataset is reasonable since legitimate websites (represented by 0) are expected to be higher than phishing websites (represented by 1).

```
#implementation of RandomOverSampler for our imbalanced dataset
rand_oversampler = RandomOverSampler()
X_train, y_train = rand_oversampler.fit_resample(X_train, y_train)
```

```
#feature normalisation
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 4.6 Fix imbalanced dataset and Normalisation.

With the combined use of random oversampling and normalization techniques, the intention was to tackle the class imbalance problem and affirm that the input features maintain a consistent scale [56]. These steps were crucial in enhancing the impartiality and efficiency of the machine learning models utilized in our study [56].

K-Value In K-Nearest Neighbour

Finding the best K-value of the KNN model contributes to the outcome of the KNN model. This research involves creating a KNN model with 'k' value 3 which is obtained from the elbow method (Figure 4.7) [32], predicting the classes of the test set samples, and calculating the error rate (the proportion of incorrect predictions). The error rates are then plotted against the 'k' values (Figure 4.7) [18].

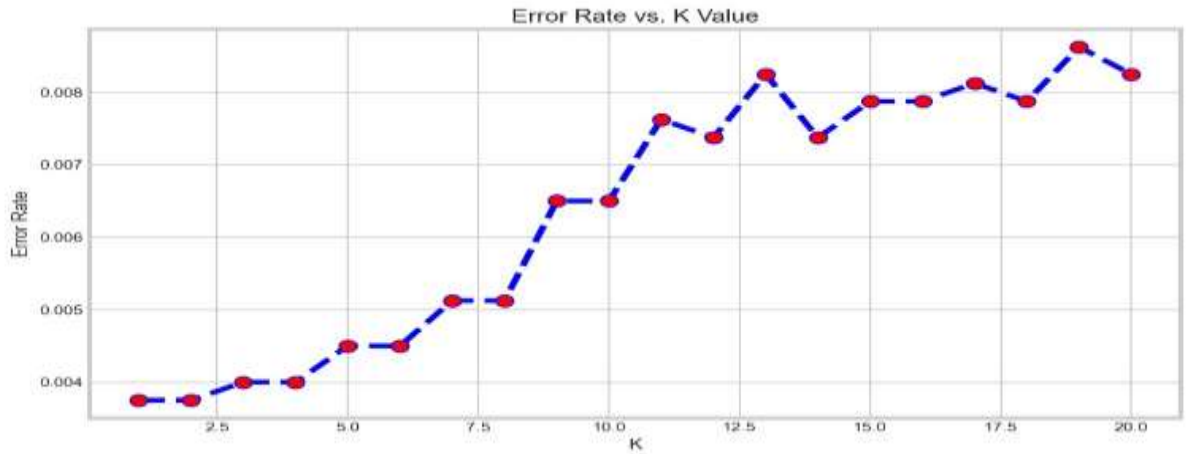


Figure 4.7 Finding K in KNN

4.2 Answers to The Research Questions

This section provides answers to the supporting research questions. Drawing from the research aims and objectives, each research question is discussed in detail.

4.2.1. How can a machine learning model be trained and optimised to effectively detect phishing attacks?

To effectively detect phishing attacks using a machine learning model, selecting an appropriate dataset is the first step. In this research, two significant datasets were chosen: [33] and [34]. The former has 5 distinct features, while the latter is more comprehensive with 80 features.

The quality of data is paramount. Ensuring a clean dataset devoid of missing or null values paves the way for more accurate predictions. The [34] dataset was free of any missing values, whereas the [33] dataset contained them. To address this discrepancy, the mean value of the affected columns was employed, ensuring consistency and integrity.

Considering the balance in the dataset is of equal importance. A skewed or imbalanced dataset can lead to inaccurate predictions, especially for the underrepresented class. The datasets used were found to be imbalanced, favouring the majority class. To address this, oversampling, a technique that replicates instances from the minority class, was implemented. This ensures a balanced representation of all classes, preventing the model from having biases.

Feature selection played a critical role. Both manual and automatic approaches were adopted. Manually, non-discriminatory features were eliminated. Subsequently, an automatic approach using the mutual information classification algorithm assessed relationships between dependent and independent variables, further refining feature selection.

2. How can the phishing detection model be integrated into existing web browsers for real-time protection?

While the research extensively described the model creation, training, and evaluation process, integration into web browsers for real-time protection would typically involve the development of a browser extension or plugin. Such an extension would utilize the trained model's API to evaluate web pages in real-time, providing instant feedback to users. A comprehensive integration strategy would encompass considerations like low latency, ensuring that the model returns predictions quickly without disrupting the user's browsing experience.

3. What are the performance metrics and evaluation techniques suitable for assessing the effectiveness and efficiency of the developed phishing detection model?

For model assessment, a splitting strategy was employed: 90% of the dataset for training and 10% for testing. This approach offers a clear demarcation between training and validation sets, ensuring that the model's performance is evaluated on unseen data, thus enhancing its credibility.

Three primary machine learning algorithms were explored: Random Forest (RF), K-Nearest Neighbour (KNN), and Artificial Neural Network (ANN).

For the RF model, hyper-parameter tuning was a focal point. Utilising the GridSearchCV function, an exhaustive search was conducted over a defined parameter grid, optimising the

model's performance. This method not only identified the best parameters but also ensured the model's robustness by integrating cross-validation.

	Precision (%)	Recall (%)	F1-Score (%)	Support
Legitimate	100.00	100.00	100.00	7212
Phishing	98.00	99.00	99.00	788
Accuracy			100.00	8000
Macro Avg.	99.00	100.00	99.00	8000
Weighted Avg	100.00	100.00	100.00	8000

KNN model's effectiveness often hinges on the right choice of 'k'. The elbow method was employed to ascertain the optimal 'k' value. Further, GridSearchCV facilitated a meticulous hyper-parameter tuning process, iterating through various combinations to find the best performing model.

The ANN, a more intricate model, was constructed using Keras [23]. The network's architecture included input, hidden, and output layers, with hyper-parameters like the number of neurons in layers and the learning rate being optimised using GridSearchCV.

In essence, this research's approach to each machine learning model was thorough, ensuring each model was optimally trained and evaluated. Through a combination of meticulous data preparation, feature selection, and model training, the research offers substantial insights into the development of a robust phishing detection system.

5. Evaluation

This chapter starts with the documentation of the result of this research then where relevant, compared with the result of the existing research works. The result section will start by documenting the result of all the models used in this research. then proceed with the result of all the integrated Django technology [12].

5.1 Result

Classification Report

Table 5.1 shows the classification report which are recall, accuracy, f1-score, precision and support gotten from each model.

Table 5.1a Result Based on RF model.

	Precision (%)	Recall (%)	F1-Score (%)	Support
Legitimate	100.00	100.00	100.00	7212
Phishing	98.00	99.00	99.00	788

			100.00	8000
Macro Avg.	99.00	100.00	99.00	8000
Weighted Avg	100.00	100.00	100.00	8000

The accuracy of the RF model is 99.78%.

Table 5.1B Result Based on KNN Model

	Precision (%)	Recall (%)	F1-Score (%)	Support
Legitimate	100.00	100.00	100.00	7212
Phishing	98.00	99.00	98.00	788
			100.00	8000
Macro Avg.	99.00	100.00	99.00	8000
Weighted Avg.	100.00	100.00	100.00	8000

The accuracy of the KNN model is 99.67%.

Table 5.1C Result Based on ANN Model

	Precision (%)	Recall (%)	F1-Score (%)	Support
Legitimate	100.00	99.00	100.00	7212
Phishing	92.00	100.00	96.00	788
			99.00	8000
Macro Avg.	96.00	99.00	98.00	8000
Weighted Avg.	99.00	99.00	99.00	8000

The accuracy of the ANN model is 99.11%.

RF = Random Forest

KNN = K-Nearest Neighbour

ANN=Artificial Neural Network

CONFUSION MATRIX

The figure 5.1 below shows the confusion matrix of all the models.

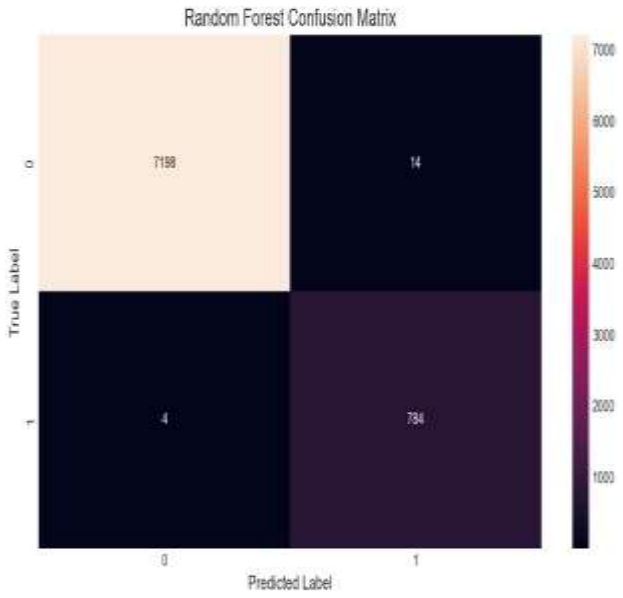


Fig 5.1a Confusion matrix of RF Model

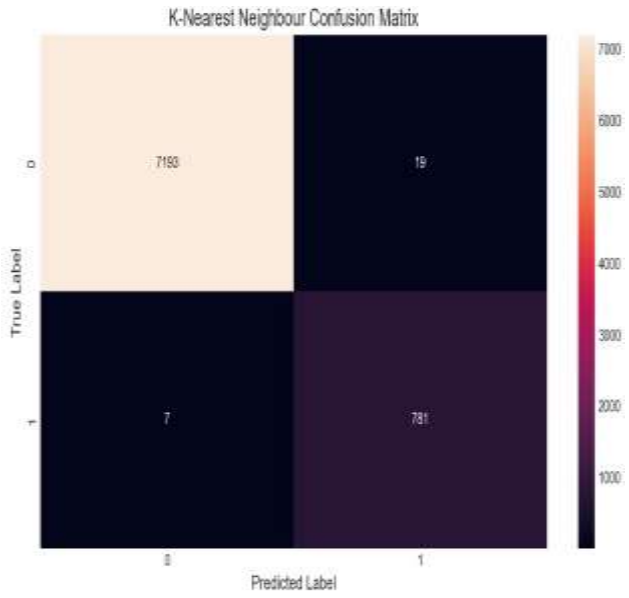


Fig 5.1b Confusion matrix of KNN Model

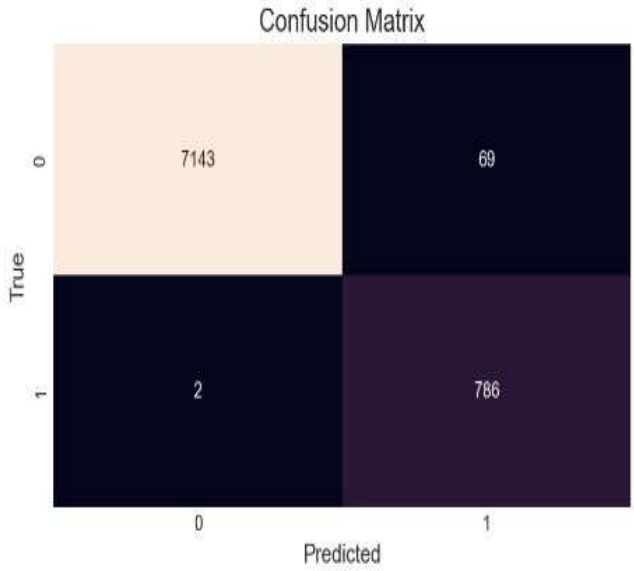


Fig 5.1c Confusion matrix of ANN Model

The table 5.2 summarise the graphical result of the confusion Matrix.

Model	**Metrics**			
	TP	FP	TN	FN
RF	7198	14	784	4
KNN	7193	19	781	7
ANN	7143	69	786	2

TP = True positive
TN = True negative
FP = False negative
FN = False negative

The performance of a machine learning model is best gauged through metrics that measure various aspects of its predictions, particularly in cases where the consequences of misclassification can be dangerous, such as in phishing detection. Tables 5.1a, 5.1b, and 5.1c, along with the summarized results of the confusion matrix in table 5.2, offer profound insights into the effectiveness of three prominent algorithms: Random Forest (RF), K-Nearest Neighbour (KNN), and Artificial Neural Network (ANN).

Insights From the Classification Report and The Confusion Matrix

Random Forest Model (RF)

From table 5.1a, the Random Forest (RF) model showcases an exceptionally high performance. Achieving an accuracy rate of 99.78%, it appears to be the most accurate among the three. The recall for legitimate websites is at a perfect 100%, meaning that all legitimate websites were correctly identified, a crucial factor for businesses and users to avoid false alarms. On the other hand, the recall for phishing websites stands at 99%, indicating that the model caught 99% of the malicious websites but missed 1%. Given the implications of failing to detect a phishing site, this 1% could be significant. The confusion matrix in figure 5.1a and its summarised results in table 5.2 reflect this, showing that only 4 phishing sites were misclassified, while a small number of legitimate sites (14) were wrongly identified as phishing.

K-Nearest Neighbour (KNN)

The KNN model, as presented in table 5.1b, yielded an accuracy of 99.67%. Though slightly less accurate than the RF model, its performance remains commendable. Like the RF model, the recall for legitimate websites stands at 100%, while for phishing websites, it's at 99%. The difference between the

RF and KNN in this context is minimal but still essential. The figure 5.1b and table 5.2 reinforce this, revealing a slightly higher misclassification with 7 phishing sites going undetected and 19 legitimate sites being erroneously flagged.

Artificial Neural Network (ANN)

The ANN model, often praised for its capability to handle complex datasets, exhibits an accuracy of 99.11% as shown in table 5.1c. Though still high, it is the least accurate among the three models. Its recall for legitimate sites is at 99%, slightly lower than the other two models. Interestingly, the recall for phishing sites is perfect at 100%. This suggests that while the ANN might occasionally misclassify a legitimate site, it does not miss any phishing site. The confusion matrix in figure 5.1c and table 5.2 accentuates this fact. A total of 69 legitimate sites were wrongly classified as phishing, but only 2 phishing sites were missed.

In summary, when one delves into the results, specific nuances become evident. If the priority is to avoid any false alarms and ensure that legitimate sites are not flagged wrongly, the RF model might be the most appropriate. However, if the utmost priority is ensuring not a single phishing site goes undetected, even at the risk of occasionally misclassifying legitimate sites, the ANN might be more suitable.

All three models have displayed impressive results, with each having its strengths. The decision on which model to deploy in a real-world setting would hinge on the specific priorities and risk appetites of the users or businesses in question. What remains undeniable is that with such high

levels of accuracy and recall, these models offer a robust line of defense against the ever-present threat of phishing attacks.

However, in this research, the RF model was integrated into the Django web application as shown in section 5.3.

Django Integration Result

Figure 5.2 shows the testing result of the RF model into the Django framework [12].

Figure 5.2a The website field page

The figure 5.2a shows the look of the interface the consuming user will see when visited the home page (in this case localhost:8000). The user must insert the complete URL name, otherwise server error (500) will be shown.

Figure 5.2b Testing for the Legitimate website

The figure 5.2b shows the example of the known legitimate URL while figure 5.2c shows the result page of the legitimate URL. From the figure 5.2c, user can click on the website URL and visit the page.

Figure 5.2c The Legitimate website page

Figure 5.2d Testing for phishing website

Figure 5.2d shows the example of the phishing URL while figure 5.2e shows how the result page for the phishing URL looks like.

Figure 5.2e The Phishing Website Page

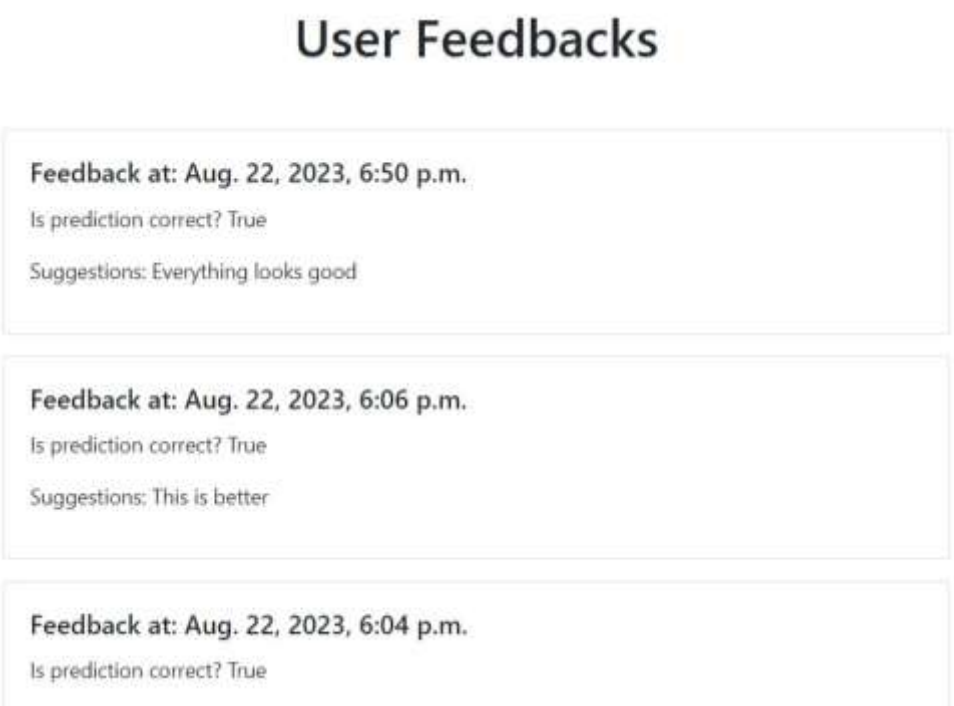


Figure 5.2f The User Feedback Page

Figure 5.2f shows the user feedback which is directly gotten from the result page. The feedback asked if the model made correct prediction and asked the end-user which area can be improved.

Testing this model and getting the feedback from the end-user can help to improve the model for future purpose. This can be helpful in knowing the recommendation for the future research.

5.3 COMPARISON WITH OTHER RESEARCH

In the field of machine learning, comparing the current study with previous research necessitates a similar choice of datasets. This comparison is crucial as it allows for the examination of the effectiveness and efficiency of different methodologies on the same data, thereby offering a fair and balanced analysis.

Upon conducting an exhaustive search on Google Scholar, the researcher can confidently assert that, to the best of his knowledge, no prior research has amalgamated the dataset from [33] with that of the [34] dataset. This unique combination of datasets not only sets this study apart but also contributes to its novelty.

Furthermore, the use of benchmark datasets has been instrumental in guiding the goals, values, and research agendas in the machine learning community [51]. In this context, this study's unique combination of the [33] and [34] datasets will serve as a benchmark for future researchers interested in using these combined datasets. This study will thereby contribute to the ongoing evolution and advancement of machine learning research methodologies.

Moreover, this study will provide valuable insights into the performance improvements that can be achieved through dataset combination. This will inform future research efforts and could potentially lead to the development of more effective machine learning models.

In conclusion, this research not only fills a gap in existing literature by combining two distinct datasets but also sets a precedent for future studies. This pioneering approach underscores the significance of innovation in machine learning research and further cements this study's contribution to the field.

6. Conclusion

The driving force behind this research was the increasing sophistication of phishing attacks and the subsequent need for an equally sophisticated detection system. Through the application of three machine learning models - Random Forest (RF), K-Nearest Neighbour (KNN), and Artificial Neural Network (ANN) - this research has provided significant insights into the landscape of phishing detection.

1. Model Evaluation: Among the three models, RF showcased the highest accuracy at 99.78%, followed closely by KNN at 99.67%, and then ANN at 99.11%. Each model displayed its strengths and peculiarities. For instance, while RF and KNN boasted 100% recall for legitimate sites, ANN managed to detect all phishing sites, with a recall rate of 100% for such sites.
2. Django Integration: The integration of the RF model into the Django web application marked the practical application of this research. Through this interface, users can quickly ascertain the legitimacy of a given website, and, by extension, avoid potential phishing threats. The feedback feature added another layer of engagement, giving users an avenue to critique and provide insights on the model's performance.
3. Benchmarking: The unique combination of datasets from [33] and [34] makes this research a pioneering effort in its right. By establishing this combination, the research

not only provides a broader perspective on phishing detection but also creates a new benchmark for subsequent researchers.

Recommendations

1. Model Refinement: Despite the high performance of the models, especially RF, there is always room for improvement. Future work could focus on reducing the 1% missed phishing sites in the RF model, ensuring a more foolproof detection system.
2. Algorithm Exploration: While RF, KNN, and ANN were the chosen models for this study, the vast world of machine learning offers numerous other algorithms and techniques. Research can be expanded to explore and benchmark the performance of other models against the current findings.
3. Integration with Browsers: Given the increasing risk of phishing attacks, there's an evident need for such detection systems to be more accessible. Integrating the model, especially the RF model, directly into web browsers can offer users real-time protection, addressing one of the research questions.
4. User Feedback Utilization: The feedback mechanism integrated into the Django application can be a gold mine for refining the model. Accumulated feedback should be analyzed periodically to enhance the model's accuracy and overall user experience.
5. Dataset Expansion: For a more comprehensive and robust model, future research should consider expanding the dataset, combining more datasets from various sources or even introducing real-time data collection mechanisms.
6. Mitigating Errors: The research mentioned potential server errors (like the 500 error) when URLs are inputted incorrectly. Future iterations of the Django application should aim to address such issues, providing users with a more seamless experience.

In conclusion, this research represents a significant stride in phishing detection. Through rigorous testing and evaluation, it not only presents a robust detection system but also lays the groundwork for further innovations in the field. The ever-evolving nature of cyber threats demands continuous refinement and adaptation, and this research provides a promising direction for such endeavors.

References

1) R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, “A comparative evaluation of outlier detection algorithms: Experiments and analyses,” Pattern Recognition, vol. 74, pp. 406–421, Feb. 2018, doi: https://doi.org/10.1016/j.patcog.2017.09.037.
2) Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, “Machine learning algorithm validation with a limited sample size,” PLOS ONE, vol. 14, no. 11, p. e0224365, Nov. 2019, doi: https://doi.org/10.1371/journal.pone.0224365.
3) Kulkarni and Leonard Brown, “Phishing Websites Detection using Machine Learning,” Computer Science Faculty Publications and Presentations, vol. 20, Aug. 2019, Available: https://scholarworks.uttyler.edu/compsci_fac/20/
4) M. Flath and N. Stein, “Towards a data science toolbox for industrial analytics applications,” Computers in Industry, vol. 94, pp. 16–25, Jan. 2018, doi: https://doi.org/10.1016/j.compind.2017.09.003.
5) F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, “Data imbalance in classification: Experimental evaluation,” Information Sciences, vol. 513, pp. 429–441, Mar. 2020, doi: https://doi.org/10.1016/j.ins.2019.11.004.

6) Namvar, M. Siami, F. Rabhi, and M. Naderpour, "Credit risk prediction in an imbalanced social lending environment," arxiv.org, Apr. 2018, Available: https://arxiv.org/abs/1805.00801
7) R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," BMC Bioinformatics, vol. 14, no. 1, Mar. 2013, doi: https://doi.org/10.1186/1471-2105-14-106.
8) J. Brownlee, "Random Oversampling and Undersampling for Imbalanced Classification," Machine Learning Mastery, Jan. 14, 2020. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
9) J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," Neurocomputing, vol. 300, pp. 70–79, Jul. 2018, doi: https://doi.org/10.1016/j.neucom.2017.11.077.
10) Srilatha , A. Ajith, and T. Johnson P. , "Feature deduction and ensemble design of intrusion detection systems," Computers & Security, vol. 24, no. 4, pp. 295–307, Jun. 2005, doi: https://doi.org/10.1016/j.cose.2004.09.008.
11) O. Abiodun, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, and R. S. Alkhawaldeh, "A systematic review of emerging feature selection optimization methods for optimal text classification: the present state and prospective opportunities," Neural Computing and Applications, vol. 33, no. 22, pp. 15091–15118, Aug. 2021, doi: https://doi.org/10.1007/s00521-021-06406-8.
12) Django, "The Web framework for perfectionists with deadlines | Django," Djangoproject.com, 2019. https://www.djangoproject.com/
13) [13]S. J. Rigatti, "Random Forest," Journal of Insurance Medicine, vol. 47, no. 1, pp. 31–39, Jan. 2017, doi: https://doi.org/10.17849/insm-47-01-31-39.1.
14) G. Biau and E. Scornet, "A random forest guided tour," TEST, vol. 25, no. 2, pp. 197–227, Apr. 2016, doi: https://doi.org/10.1007/s11749-016-0481-7.
15) Scikit-Learn, "scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation," Scikit-learn.org, 2019. https://scikit-learn.org/
16) C. Angione, E. S. Silverman, and E. Yaneske, "Using machine learning as a surrogate model for agent-based simulations," Using machine learning as a surrogate model for agent-based simulations, vol. 17, no. 2, pp. e0263150–e0263150, Feb. 2022, doi: https://doi.org/10.1371/journal.pone.0263150.
17) M. Belete and M. D. Huchaiah, "Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results," International Journal of Computers and Applications, pp. 1–12, Sep. 2021, doi: https://doi.org/10.1080/1206212x.2021.1974663.
18) L. Eko , B. Achmad , and B. Fitra , "Optimization of K Value in KNN Algorithm for Spam and Ham Email Classification | Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)," www.jurnal.iaii.or.id, vol. 4, no. 2, Apr. 2020, Accessed: Jul. 24, 2023. [Online]. Available: http://www.jurnal.iaii.or.id/index.php/RESTI/article/view/1845
19) Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN Model-Based Approach in Classification," On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, vol. 2888, pp. 986–996, 2003, doi: https://doi.org/10.1007/978-3-540-39964-3_62.

20) Scikit-Learn, "sklearn.neighbors.KNeighborsClassifier," scikit-learn. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed Jul. 24, 2023).
21) J. Zou, Y. Han, and S.-S. So, "Overview of Artificial Neural Networks," Methods in Molecular BiologyTM, vol. 458, pp. 14–22, 2008, doi: https://doi.org/10.1007/978-1-60327-101-1_2.
22) M. Mishra and M. Srivastava, "A view of Artificial Neural Network," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), Aug. 2014, doi: https://doi.org/10.1109/icaetr.2014.7012785.
23) Keras, "Home - Keras Documentation," Keras.io, 2019. https://keras.io/
24) Google, "TensorFlow," TensorFlow, 2019. https://www.tensorflow.org/
25) Y. Liu, J. Bernstein, M. Meister, and Y. Yue, "Learning by Turning: Neural Architecture Aware Optimisation," proceedings.mlr.press, Jul. 01, 2021. http://proceedings.mlr.press/v139/liu21c.html
26) J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics," Electronics, vol. 10, no. 5, p. 593, Mar. 2021, doi: https://doi.org/10.3390/electronics10050593.
27) R. Paturi, L. Swathi, K. Sai. Pavithra, R. Mounika, and Ch. Alekhya, "Detection of Phishing Attacks using Visual Similarity Model," IEEE Xplore, May 01, 2022. https://ieeexplore.ieee.org/document/9793231 (accessed Jul. 22, 2023).
28) L. Cheng, F. Liu, and D. D. Yao, "Enterprise data breach: causes, challenges, prevention, and future directions," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 7, no. 5, p. e1211, 2017, doi: https://doi.org/10.1002/widm.1211.
29) L. Tang and Q. H. Mahmoud, "A Survey of Machine Learning-Based Solutions for Phishing Website Detection," Machine Learning and Knowledge Extraction, vol. 3, no. 3, pp. 672–694, Aug. 2021, doi: https://doi.org/10.3390/make3030034.
30) D. Shewan, "10 Companies Using Machine Learning in Cool Ways," Wordstream.com, 2017. https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications
31) Hinneburg, C. C. Aggarwal, and D. A. Keim, "hat is the nearest neighbor in high dimensional spaces?," 2000. http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-70224
32) M. Cui, "Introduction to the K-Means Clustering Algorithm Based on the Elbow Method," Introduction to the k-means clustering algorithm based on the elbow method, 2020, doi: https://doi.org/10.23977/accaf.2020.010102.
33) UNB, "URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB," www.unb.ca, 2016. https://www.unb.ca/cic/datasets/url-2016.html
34) Subhash, S. Fernando, and S. Fernando, "Phishing websites dataset," 2021.
35) V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing Detection Using Machine Learning Techniques," arXiv:2009.11116 [cs, stat], Sep. 2020, Available: https://arxiv.org/abs/2009.11116
36) O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Systems with Applications, vol. 117, pp. 345–357, Mar. 2019, doi: https://doi.org/10.1016/j.eswa.2018.09.029.
37) K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," Journal of Ambient Intelligence and Humanized

Computing, vol. 10, no. 5, pp. 2015–2028, Apr. 2018, doi: https://doi.org/10.1007/s12652-018-0798-z.

38) E. Gandotra and D. Gupta, "An Efficient Approach for Phishing Detection using Machine Learning," Multimedia Security, pp. 239–253, 2021, doi: https://doi.org/10.1007/978-981-15-8711-5_12.
39) Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," Telecommunication Systems, vol. 76, no. 1, Oct. 2020, doi: https://doi.org/10.1007/s11235-020-00733-2.
40) ZScaler, "2023 Phishing Report Reveals 47.2% Surge in Phishing Attacks Last Year," Zscaler, 2023. https://www.zscaler.com/blogs/security-research/2023-phishing-report-reveals-472-surge-phishing-attacks-last-year
41) K. Jain and B. B. Gupta, "PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning," Advances in Intelligent Systems and Computing, pp. 467–474, 2018, doi: https://doi.org/10.1007/978-981-10-8536-9_44.
42) Safi and S. Singh, "A systematic literature review on phishing website detection techniques," Journal of King Saud University - Computer and Information Sciences, Jan. 2023, doi: https://doi.org/10.1016/j.jksuci.2023.01.004.
43) K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," Information Sciences, vol. 484, pp. 153–166, May 2019, doi: https://doi.org/10.1016/j.ins.2019.01.064.
44) F. Atlam and O. Oluwatimilehin, "Business Email Compromise Phishing Detection Based on Machine Learning: A Systematic Literature Review," Electronics, vol. 12, no. 1, p. 42, Dec. 2022, doi: https://doi.org/10.3390/electronics12010042.
45) C.-Y. Wu, C.-C. Kuo, and C.-S. Yang, "A Phishing Detection System based on Machine Learning," IEEE Xplore, Aug. 01, 2019. https://ieeexplore.ieee.org/document/8858325 (accessed Mar. 17, 2022).
46) D. A. P. Delzell, S. Magnuson, T. Peter, M. Smith, and B. J. Smith, "Machine Learning and Feature Selection Methods for Disease Classification With Application to Lung Cancer Screening Image Data," Frontiers in Oncology, vol. 9, Dec. 2019, doi: https://doi.org/10.3389/fonc.2019.01393.
47) Hannousse and S. Yahiouche, "Towards benchmark datasets for machine learning based website phishing detection: An experimental study," Engineering Applications of Artificial Intelligence, vol. 104, p. 104347, Sep. 2021, doi: https://doi.org/10.1016/j.engappai.2021.104347.
48) M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani, and H. Gharaee, "An Adaptive Machine Learning Based Approach for Phishing Detection Using Hybrid Features," IEEE Xplore, Apr. 01, 2019. https://ieeexplore.ieee.org/document/8765265
49) M. Almseidin, M. Alkasassbeh, M. Alzubi, and J. Al-Sawwa, "Cyber-Phishing Website Detection Using Fuzzy Rule Interpolation," Cryptography, vol. 6, no. 2, p. 24, May 2022, doi: https://doi.org/10.3390/cryptography6020024.
50) O. Khadidos, S. Shitharth, A. O. Khadidos, K. Sangeetha, and K. H. Alyoubi, "Healthcare Data Security Using IoT Sensors Based on Random Hashing Mechanism," Journal of Sensors, vol. 2022, pp. 1–17, Jun. 2022, doi: https://doi.org/10.1155/2022/8457116.

51) U. Ozker and O. K. Sahingoz, "Content Based Phishing Detection with Machine Learning," 2020 International Conference on Electrical Engineering (ICEE), Sep. 2020, doi: https://doi.org/10.1109/icee49691.2020.9249892.
52) N. Puri, P. Saggar, A. Kaur, and P. Garg, "Application of ensemble Machine Learning models for phishing detection on web networks," IEEE Xplore, Jul. 01, 2022. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9913599 (accessed May 11, 2023).
53) N. S. Zaini et al., "Phishing detection system using nachine learning classifiers," Indonesian Journal of Electrical Engineering and Computer Science, vol. 17, no. 3, p. 1165, Mar. 2020, doi: https://doi.org/10.11591/ijeecs.v17.i3.pp1165-1171.
54) A. Cuzzocrea, F. Martinelli, and F. Mercaldo, "A machine-learning framework for supporting intelligent web-phishing detection and analysis," Proceedings of the 23rd International Database Applications & Engineering Symposium on - IDEAS '19, 2019, doi: https://doi.org/10.1145/3331076.3331087.
55) A. Orunsolu, A. S. Sodiya, and A. T. Akinwale, "A predictive model for phishing detection," Journal of King Saud University - Computer and Information Sciences, Dec. 2019, doi: https://doi.org/10.1016/j.jksuci.2019.12.005.
56) Md. M. Uddin, K. Arfatul Islam, M. Mamun, V. K. Tiwari, and J. Park, "A Comparative Analysis of Machine Learning-Based Website Phishing Detection Using URL Information," IEEE Xplore, Aug. 01, 2022. https://ieeexplore.ieee.org/document/9904055 (accessed Dec. 14, 2022)
57) Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, "An intelligent cyber security phishing detection system using deep learning techniques," Cluster Computing, May 2022, doi: https://doi.org/10.1007/s10586-022-03604-4.
58) N. Abdelhamid, F. Thabtah, and H. Abdel-jaber, "Phishing detection: A recent intelligent machine learning comparison based on models content and features," 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Jul. 2017, doi: https://doi.org/10.1109/isi.2017.8004877.
59) Singh, I. Al-Mahmood, and M. Al-Tahsin, "Novel Approach to Secure Websites with Machine Learning Classifiers," Asian Journal of Social Science and Management Technology Asian Journal of Social Science and Management Technology, vol. 4, no. 2, pp. 2313–7410, 2022, Accessed: Mar. 04, 2024. [Online]. Available: http://www.ajssmt.com/Papers/42152176.pdf
60) N. K. Gyamfi and J.-D. Abdulai, "Bank Fraud Detection Using Support Vector Machine," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Nov. 2018, doi: https://doi.org/10.1109/iemcon.2018.8614994.
61) Zscaler, "2023 Zscaler ThreatLabz State of Phishing Report | Zscaler," info.zscaler.com, 2023. https://info.zscaler.com/resources-industry-reports-threatlabz-phishing-report
62) Proofpoint, "2023 State of the Phish Report - Stats, Trends & More | Proofpoint AU," Proofpoint, Feb. 22, 2021. https://www.proofpoint.com/au/resources/threat-reports/state-of-phish
63) Internet Crime Complaint Center, "2020 Internet Crime Report," FBI, 2020. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf