National College of Ireland

Using Genetic Algorithms for Optimized Feature Selection in Machine Learning and Deep Learning Models to Detect Phishing Websites

MSc Research Project
Msc Cyber Security

Akinola David Omotola
Student ID: x22133755

School of Computing
National College of Ireland

Supervisor: Raza ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. Akinola David Omotola……

Student ID: …… x22133755……

Programme: ……Msc Cybersecurity…… **Year:** …2024..

Module: MSc Research Project

Supervisor: ……Raza ul Mustafa……

Submission Due Date: ……27/5/2024……..……

Project Title: ……Using Genetic Algorithms for Optimized Feature Selection in Machine Learning and Deep Learning Models to Detect Phishing Websites……

Word Count: ……8116…… **Page Count**……21…….

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …… Akinola David Omotola…

Date: ……27/5/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Using Genetic Algorithms for Optimized Feature Selection in Machine Learning and Deep Learning Models to Detect Phishing Websites

Akinola David Omotola
X22133755

Abstract

Given the advancement of network technology and the global rise in internet usage, cybersecurity concern has also risen. Phishing attacks which use misleading tactics to leak sensitive information bring about significant financial losses, reputational harm, system disruptions and legal consequences for both victim and organisation. This growing threat demonstrates the critical need for more robust detection systems, especially when accounting for evolving attack tactics. This study addresses the issue by investigating the use of a Genetic Algorithm (GA) combined with Support Vector Machine (SVM), Random Forest Classifier (RFC), Gradient Boost Classifier (GBC) and Graph Neural Network (GNN) to identify phishing websites. The findings of the experiment show that hybrid models perform better than single-base models, providing higher accuracy, F1-score, and AUC on a small dataset. According to this research, the GA-GNN and GA-RFC were the best and most improved model with accuracy of 95%, F1-score of 94.74% and AUC of 95.45%. The paper does admit several limitations such as, performance drop with larger sample size. Nonetheless, the study highlights the promise of hybrid models in tackling phishing detection problems and raising detection precision.

1 Introduction

This chapter is an introduction to phishing website. It highlights the impact and significance of phishing websites. This chapter provides defines the problem and the motivation behind this study. It discusses the objectives needed to answer the question posed by the problem. This chapter gives a background of what this research study about and why it is important.

1.1 Background

The rapid evolution of computer technology has greatly transformed the modern world with the global internet user population increasing from 60% to 66% in the last half decade (Petrosyan, 2024). This increased dependence on internet platforms across several industries has been made clearer by the COVID-19 pandemic, exposing large amounts of data to security risks such as DDoS, XSS scripting, SQL injection, phishing and many more attacks (Tang & Mahmoud, 2021).

Amidst this digital expansion, cybersecurity threats have also increased making it easier for nefarious actions to be carried out by persons with the intent to damage computing

systems, jeopardise data integrity, or cause harm. Cybercriminals exploit opportunities to perpetrate phishing attacks, a significant cybersecurity challenge (Ali, 2017; Harinahalli Lokesh & BoreGowda, 2021). Phishing attacks use deceptive communication channels such as emails and internet advertisements that target people to make them give sensitive information by impersonating reputable entities (Safi & Singh, 2023; Zhuang et al., 2012).

The impact of phishing attacks can be both devastating to individuals and organizations. Phishing attacks cause huge monetary loss for organisations. In 2019, hackers used approaches including credential theft and fake invoicing to inflict $1.7 billion worth damages. Further, these attacks harm organizations' reputation since hackers can spread spam pretending to be the company which undermines the trust of partners and customers and lead to reduced sales. According to a survey, 41% of UK customers never come back to a brand after a breach, and 44% of consumers stop shopping with it for months. Phishing attacks cause system failures and lost productivity by infecting systems with malware or ransomware. For UK organisations, they constitute the most disruptive type of cyberattack; in the last year, two-thirds of them experienced interruption. Furthermore, there are harsh legal penalties for improper treatment of data; under UK GDPR legislation, fines can amount to £17.5 million, or 4% of annual worldwide revenue. For example, Equifax was forced to pay up to $700 million for their data breaches, while British Airways was fined of up to £20 million (CybSafe, 2023). Figure 1 below shows the industries most targeted by phishing attacks.

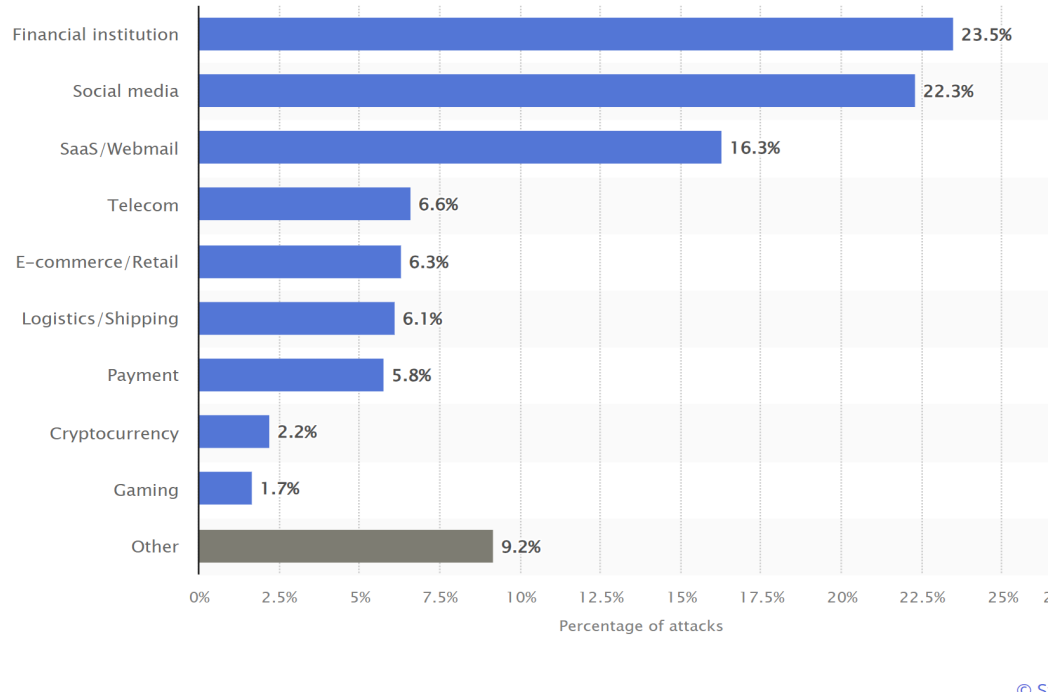

Figure 1. Percentage of Attacks Ranked by Industry (Petrosyan, 2024)

1.2 Motivation

The rising threat of phishing attacks pose a significant challenge for vulnerable users within information systems making it difficult to implement effective mitigation strategies. Business Email Compromise (BEC) assaults surged with a 57% increase in average requested wire transfer that amounted to $293,359. The financial sector and social media make up almost half of the all targets boasting of a combined 46% of all phishing attacks, while online payment providers face 5.8%. With phishing related data breaches rising to 82% and human intelligence-driven tactics dominating cyber threats, the existing gaps in literature denote that classifier considerations are limited and imbalanced datasets impact how efficient current machine learning (ML) and deep learning (DL) techniques are. To improve phishing prevention techniques against emerging attacks requires addressing these issues. Combating phishing attempts that target individuals and financial institutions require early identification. Intelligent detection systems are essential, especially those that use supervised ML and DL algorithms, as traditional solutions that rely on static databases find it difficult to keep up with new threats (He et al., 2011; Nguyen & Nguyen, 2016).

1.3 Problem Statement

There are unmet research needs related to the inefficiencies of traditional detection frameworks and the robustness of hybrid models as the remedy for such problems such as sophisticated evasion techniques and adversarial attacks. These attacks have been proven to bypass base models inadequacies. Single-base models are sometimes not capable of handling complex multi-faceted feature set (Saravanan and Subramanian 2020; van Geest et al., 2024).

1.4 Research Question

This study aims to address the critical research question:
"How does combining Genetic Algorithm with Supervised ML, and DL algorithms improve phishing website detection?"
The study aims to assess the effectiveness of combining various algorithms for phishing website detection, including Genetic Algorithm, supervised machine learning and deep machine learning. By conducting experiments, it seeks to evaluate the performance of this combination in accurately identifying phishing websites and compare it with individual algorithms. Through empirical evidence, the research aims to contribute insights into enhancing cybersecurity against phishing attacks.

1.5 Research Objectives

To address the problem and answer the research question, this study aims to offer an alternative framework to efficiently detect phishing websites. This study integrates Genetic Algorithm with base machine learning and deep learning models in order mitigate the drawbacks of stand-alone models. Using extensive assessment criteria, such as false positives and detection rates, this study seeks to improve cybersecurity defenses against phishing attempts. To accomplish this task, the following objectives will be met.

- Critically review works done by previous researchers in phishing website detection using hybrid models. Additionally, the literature around the use of the GA in the detection of phishing websites will be explored.

- Implement a hybrid model using a combination of GA with Graph Neural Network (GNN), Gradient Boost Classifier (GBC), Support Vector Machine (SVM) and Random Forest Classifier (RFC) using phishing website dataset from a publicly available source – Mendeley.
- Evaluate the implemented models based on their Accuracy, AUC, F-measure, supervised learning models – Evaluate the efficacy of the hybridized GNN, GBC, SVM and RFC on the same datasets.

The chapters are arranged as follows: The key ideas of the phishing life cycle, types of phishing attacks, technical methods for identifying phishing websites, and a critical evaluation of the state of the art research are presented in Chapter 2. The methodology used in this study, including the research plan, data-gathering techniques, and analytical techniques is covered in Chapter 3. Chapter 4 describes how the proposed model was implemented and gives the study's findings. Chapter 5 provides a complete examination of the study's findings, including conclusions based on the findings, constraints discovered during the research, and prospective directions for further research.

2 Literature Review

This chapter discusses the relevant literature pertinent to phishing website detection. It highlights the lifecycle, different phishing attack types and detection methods. This chapter provides critical analyses of the existing body of work, identifies the gaps and key contributions. This chapter lays the groundwork for this research study and methodological approach adopted.

To avoid data breaches and protect sensitive information from exposure during the reconnaissance, weaponization, distribution, exploitation, and exfiltration stages of the phishing lifecycle, strong cybersecurity measures must be implemented. Understanding these stages and implementing appropriate defence systems can help organisations to minimize risk (Goel et al., 2017; Ding et al., 2019).

2.1 Types of Phishing

The two most common types of phishing assaults are malware-based and deceptive. Malware-based phishing or "pharming" is perpetrated by luring unsuspecting individuals or organisations into installing malicious software (malware) on their systems. Attackers frequently employ phishing websites that appear authentic, sometimes from trusted sources such as financial institutions or government agencies, to trick victims into updating account information or opening attachments. In contrast, misleading phishing includes hackers impersonating legitimate businesses or organisations in order to trick victims into providing critical information. Attackers establish a false feeling of urgency or trust, frequently mimicking banks, social media sites, or coworkers, to trick victims into compromising their security (Bergholz et al., 2010; Almomani et al., 2013b). This study focuses on the latter.

2.2 Phishing Website Detection Method

Various technologies have been developed to address the issue of phishing websites. These include anti-phishing solutions such as List-based techniques, Heuristic-based techniques, Visual Similarity-based techniques, and Machine learning-based techniques.

List-based approaches use web browser whitelists and blacklists. Whitelists include trusted URLs, which increase false positives while blacklists exclude known phishing URLs, which increases false negatives. The drawback of this method is that it is unable to identify phishing websites that are not on the blacklist or whitelist, as well as websites that have content that is identical to that of phishing websites that have been restricted (Rao et al., 2020).

Heuristic methods extract characteristics from source code and third parties which make them good at identifying novel phishing attacks. However, they suffer when it comes to differences in website features (Rao et al., 2015). Another drawback is that phishing sites located on hacked servers may go undetected if third-party services are used, leading to the mistaken classification of these websites as legitimate due to their inclusion in search results (Rao et al., 2020).

The two types of existing similarity-based phishing detection techniques are screenshot similarity and HTML code similarity. In order to identify phishing, researchers first analysed HTML codes. However, these techniques may not be successful if attackers produce pages that appear similar but use different HTML codes, or if they substitute pictures or embedded objects for HTML. Because of this, some researchers can now determine similarity from snapshots of displayed webpages (Wang et al., 2024).

Machine learning-based methods have become more widely adopted because of the limitations of the above methods. ML algorithms rely on training data size and feature set, they provide great efficacy in recognizing phishing attempts with huge datasets by utilizing machine learning algorithms such as RF, LR, DT and NB (Kumar et al., 2020). Combining heuristic and ML or DL techniques (hybrid models) show potential in effectively tackling computational issues by using the advantages of both approaches (Saravanan and Subramanian, 2020).

2.3 State of the Art

Several studies have conducted comprehensive analyses that have compared the effectiveness of several machine learning models in detecting phishing websites using different benchmark datasets. Many studies such as (Subasi et al., 2020; Zouina and Outtaj, 2017; Huang et al., 2019; and Somesha et al., 2020) focused on URL-based phishing website detection using

machine learning and deep learning approaches with high detection rates. However, the limitation with these approaches is that they disregard the HTML content which contains relevant information that could make the website phishing detection models more effective. Because the HTML feature has not been used in the training of these models, these models could falsely classify phishing websites as legitimate.

Fewer studies such as (Li et al., 2019and and guptta2024) have used hybrid (i.e. URL-based and HTML-based) features to build more robust phishing website detection models. These models have demonstrated very high detection rates. However, most of these studies implement single-model methods which are stifled by the models' inherent cons.

Vijayalakshmi et al. (2020) conducted an in-depth exploration of website phishing detection methodologies, categorizing them into URL-based, content-based, and hybrid-based approaches. Through an extensive analysis, they assessed detection techniques such as list-based, heuristic rule, and learning-based methods, aiming to offer a comprehensive overview of the field. By evaluating these methods across dimensions such as performance, limitations, reliance on third-party services, and language independence, the study provides valuable insights into their efficacy and applicability. However, while the survey includes some discussion of deep learning (DL) models, it lacks detailed elucidation of their feature extraction mechanisms from HTML data, indicating scope for further investigation into this aspect of phishing detection strategies.

Saravanan and Subramanian 2020 opined that the current research was limited by poor hyperparameter tuning and inadequate feature selection methods. The authors then proposed a hybrid model that used GA to select the best features given that irrelevant features impact the model's building time and accuracy. The ARTMAP module utilizes a supervised neural network for classification, automatically categorizing input patterns into recognized classes based on predictive capabilities made of two self-organized ART modules joined by a self-associative memory and an internal controller. The network aims to optimize accuracy and reduce error. The proposed model showed improved performance in accuracy, error rate and detection time although stymied by using only URL-based features.

Rao et al. (2018) proposed a new classification methodology that utilizes heuristic-based feature extraction techniques. This approach categorized the selected features into three main groups. Despite achieving an impressive accuracy rate of 99.55%, the methodology is limited by its reliance on third-party services, which introduces variability in the speed of website classification. Additionally, the model's effectiveness heavily relies on good quality and ample quantity of the training dataset used. Furthermore, the extraction of broken link features can lead to increased computational overhead, especially noticeable in websites with complex hyperlink structures. These limitations underscore the need for further refinement and optimization of the proposed methodology.

Table 1. Summary Table for State of the Art

Reference Paper	Strengths	Drawbacks
Subasi et al., (2020)	- Investigates ensemble methods. Introduces boosting + bagging model.	- Comparison limited to UCI dataset - Focuses on URL-based features

	- Measures time complexity	
Zouina and Outtaj, (2017)	- Introduces lightweight SVM model suitable for mobile devices	- Only URL-based features are applied in the model, disregarding the HTML content
Huang et al., (2019)	- Introduce PhishingNet for timely phishing URL detection that extract character-level spatial features and word-level temporal features using CNN and RNN.	- Only URL-based features are applied in training the model, disregarding the HTML content
Somesha et al., (2020)	- Investigates deep learning method. - Testing done with different number of layers	- Use of third-party feature - May miss phishing sites utilizing embedded objects like Flash, JavaScript, and HTML files to substitute textual content.
Li et al., (2019)	- Designed lightweight features for URLs and HTML, and introduce HTML string embedding. - Used stacking model to enhance phishing webpage detection. - Used multiple datasets for evaluation.	- May not address real-time threat detection
Guptta et al., (2022)	- Introduces a machine learning-based method for real-time phishing website detection - Focusing on hybrid features derived from URLs and hyperlinks.	- May not address real-time threat detection
Saravanan and Subramanian (2020)	- Used reduced feature vector - Introduced a stacking model using ARTMAP	- May not address real-time threat detection - Used only URL-based features
Rao et al. (2018)	- Proposes a novel classification model leveraging heuristic features. - Evaluation of various types of Random Forest (RF) algorithms.	- Use of third-party services. - Use of only URL-based features - Increased computational overhead due to broken link features

The adoption of deep learning techniques underscores the significance of leveraging advanced computational methods capable of discerning intricate patterns within URL structures, content features and domain-specific features associated with phishing attempts. This highlights the potential of sophisticated neural network architectures in bolstering cybersecurity measures by accurately identifying malicious online activities. The utilization of deep learning techniques, coupled with domain-specific features, underscores the potential for enhancing detection accuracy and robustness in identifying fraudulent online activities.

3 Research Methodology

This chapter outlines the suitable processes for executing the research experiment, following a minor modification of the traditional CRISP-DM methodology to make a suitable standard data science framework for this research study (see Figure 1 below). It covers the definition of objectives, data gathering, data preparation, modelling and the evaluation methodology. The chapter describes the research processes from data collection and processing to the ML algorithms, and evaluating their performance in detection of phishing websites.

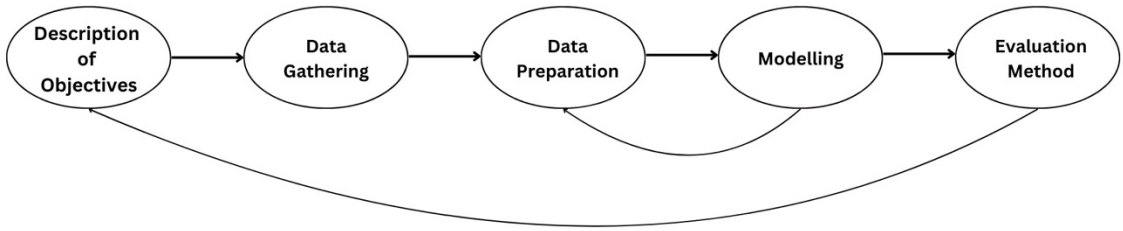

Figure 2. Phishing Website Detection Implementation Process

3.1 Definition of Objectives

Protecting people and companies from phishing attacks, mitigating the drawbacks of single-base models, and improving the overall detection rate of phishing website detectors is the main justification for using a hybrid ML to identify phishing websites. Phishing websites try to trick its victims into revealing personal information or committing dangerous activities. Businesses may examine and categorise website URLs using machine learning, identifying possible phishing attempts based on unique characteristics and patterns. Sensitive data is protected, cybersecurity measures are strengthened, data breach risks are reduced, and consumer trust is increased by implementing an efficient hybrid ML model for phishing website detection. Businesses may deploy resources more effectively and prevent cybercrime by automating the detection process.

3.2 Data Gathering

This study is aimed at the detection of phishing websites using hybrid ML and DL approaches, the website datasets used were obtained from Mendeley. The dataset consists of 80,000 combine legitimate and phishing website records built for the training and testing of the implemented models, collected from Google, Ebbu2017 Phishing Dataset, PhishTank, OpenPhish and PhishRepo. The URL and the HTML page are included in each instance. The root file, index.sql, is used to map URLs to the appropriate HTML pages. The index.sql file contains five attributes – rec_id, url, website, result and created_date.

3.3 Data Preparation

The process of refining raw data, or data cleaning, is a necessary first step in order to effectively gather information and carry out accurate data mining. Gathering, organising, and classifying the websites was the initial stage in this procedure. First, the data frame is analysed and checked for missing values which are then removed. The features which do not contribute to the classification of websites are also removed from the dataset.

3.4 Modelling

To implement the phishing website detection system models that will satisfy the aim of this research, four machine learning models are selected and are described as follows:

3.4.1 Random Forest Classifier (RFC)

In many phishing detection systems, random forest is a widely supervised method for classification. A random subset of the training data and features is used to train each of the decision trees that the algorithm creates. Each tree evaluates the fresh sample during classification, and the final prediction is made using the majority vote. To maximise information gain, decision trees are constructed recursively by dividing the data based on input characteristics and continuing the process until subsets have homogenous class labels or a stopping requirement is satisfied.

3.4.2 Support Vector Machine (SVM)

Robust machine learning methods called support vector machines (SVM) are used in classification. Their primary objective is to determine which hyperplane is best for classifying the data and maximising the space between the hyperplane and the closest data points. SVM is useful for detecting phishing websites since it can distinguish between authentic and phishing webpages using characteristics gleaned from the HTML content and URL. SVM can detect abnormalities in future data and learn patterns from labelled data, which enables it to initiate mitigation measures.

3.4.3 Gradient Boost Classifier (GBC)

Phishing website detection can be aided by Gradient Boosting technique. The fundamental principle of boosting is to combine weak learners into a powerful model. For this purpose, Gradient Boosting is a very dependable technique that improves performance on a range of risk functions and removes multicollinearity, allowing for the development of accurate prediction models. Suitable for categorization tasks, a strong learner is generated from an ensemble of weak learners.

3.4.4 Graph Neural Network (GNN)

Graph neural networks (GNNs) are specialised deep learning models designed to handle and analyse graph-structured data. The graphs represent entity relationships which make up the nodes and edges. They provide an adaptable framework for describing and understanding diverse patterns in nature, and they function as a fundamental instrument for gathering data from the natural world. By successfully identifying structural information and obtaining meaningful high-level representations from graph-structured data, GNNs are superior representation learners.

The justification for choosing these models is that they perform well, especially when high-dimensional data and substantial class differences are present. They mimic non-linear decision boundaries well, are noise-resistant, and perform well with both big and small feature sets. Notably, these models provide good accuracy without overfitting, even with a greater number of characteristics. They exhibiting enhanced explainability, scalability,

flexibility, and resource efficiency, they offer significant insights into crucial variables for categorization, making them useful instruments for phishing website detection applications.

3.5 Evaluation Methodology

To assess the performance of the hybrid phishing website detectors, the models utilize a dataset that hold legit and phishing webpages. The ratio of webpages that are correctly grouped serves as an additional metric to assess the algorithm's effectiveness, representing the portion of URLs correctly categorized.

The accuracy, F1-score and AUC are the chosen evaluation metrics for this research. Selecting the right assessment metric for classification tasks depends on the class distribution of the dataset and whether the focus is on positive or negative predictions. Accuracy suits balanced problems where the importance of positive and negative predictions is equal. On the other hand, as the F1 score combines accuracy and recall into a single value, accurately expressing model performance, it is favoured in unbalanced circumstances where positive examples are prioritised. Furthermore, the F1 score assesses performance at a particular threshold, but the AUC measure sheds light on a model's capacity to distinguish between true and false occurrences across a range of thresholds, independent of class imbalance.

4 Design Specification

This chapter discusses the underlining architecture to facilitate the implementation of the proposed phishing website detection system. It describes the GA and how it fits into the proposed hybrid model.

4.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) emulates natural selection to solve optimization problems, using a population of potential solutions represented as individuals (chromosomes), evolving through selection, crossover, and mutation (usually referred to as the GA operations or steps). The steps or operations involved in the GA-based feature selection are as follows:

- Initial Population: In GA, the initial population comprises individuals representing feature combinations from the dataset. Each feature acts as a binary switch within the chromosome, denoted as either on (1) or off (0).
- Fitness Function: Evaluates each individual's performance relative to others, assigning a score based on its ability to predict phishing or legitimate websites. Higher scores indicate better-performing individuals, guiding the selection of the top candidates for the next iteration.
- Selection: After obtaining scores, the best chromosomes are chosen for the next generation using an efficient method called tournament selection.
- Crossover: In crossover, genes from two parent chromosomes are mixed to produce new offspring, akin to combining traits from parents to create offspring. This process is known as a two-point crossover, which is one of the several types of crossover operators used to recombine the genetic material of two-parent individuals to produce new offspring.

- Mutation: This ensures population diversity and prevents premature convergence of the algorithm by randomly altering parts of a chromosome to explore potentially better solutions using an efficient method called bit-flip mutation.

By taking these stages repeatedly, the population has evolved towards ever-more-optimal solutions.

4.2 Proposed Model

The model we propose uses the base models to perform the classification. Based on the results of the predictions, the GA is used to perform dimensionality reduction i.e. the GA applies the fitness function to evaluate which genomes (features) in the population are optimal for analysing phishing websites.

Hyperparameter tuning is used to select optimal parameters for each model that will be used for the analysis. The proposed model uses the optimal feature selection from the GA and the hyperparameter tuned models to perform analysis for each model built for phishing website detection. A schematic overview of the proposed framework for automated phishing detection is shown below.

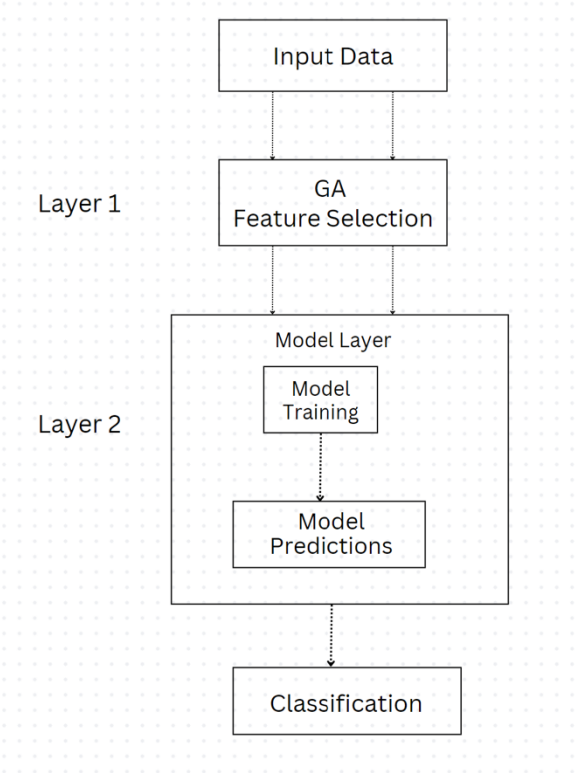

Figure 2. Model Architecture

5 Implementation

This chapter describes each machine learning model's implementation of the previously defined system design and how the models function in relation to the selected evaluation metrics. Additionally, the main conclusions and the outcomes of each machine learning model's system design implementation are examined.

A comprehensive overview of the insights gathered throughout the investigation is provided by the EDAs carried out on the dataset. Studying the dataset in depth leads to a better understanding of its statistical significance. A pie chart representing the class distribution of phishing to legitimate websites was used to get this data shown in figure 3 below.

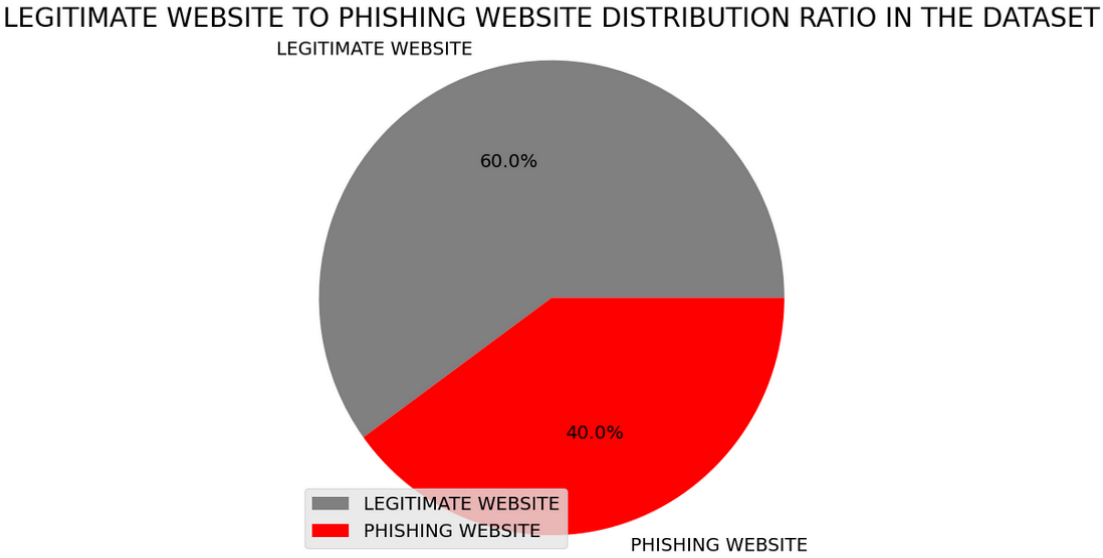

Figure 3. Class Distribution of Websites

5.1 Feature Extraction

To extract features from the dataset, two python modules were created, namely, features_to_extract.py and features_extraction.py. the features_to_extract.py defined functions used to generate and extract a feature from the raw dataset. On the other hand, the features_extraction.py is used to assemble the features that have been extracted from the raw dataset. After extracting these features, they are converted to Pandas data frame and stores as a csv file containing the new dataset used for analysing the phishing websites. The table x in the appendix shows a summary of the extracted features and their description.

5.2 Experiment Setup

This study implemented four ML and DL models including SVM, RFC, GBC and GNN. Two experiments were carried out, experiment 1 was designed to use the default parameters while

experiment 2 was designed with GA feature selection and hyperparameter tuning. These models were built using ***run_svm_model***, ***run_rfc_model***, ***run_gbc_model*** and ***run_gnn_model*** function respectively. These functions create an instance of the model being analysed using the default parameters or hyperparameters based on the value of the ***grid_params*** arguments given to the function. The function then invokes the ***run-_model_analysis*** function which performs model training by calling the ***train_model*** function, generate model predictions by calling the ***generate_model_prediction*** function, generate model evaluation summary by calling the ***generate_summary_result*** function. Lastly, displays analysis summary by calling the ***generate_analysis_summary*** function. The output of the ***generate_summary_result*** function is the stored in the ***analysis_result*** global variable.

5.3 Implementation of Genetic Algorithm (GA)

To implement the GA for feature selection, the **creator** module from deap library is used to define the representation of an individual (aka chromosome or genome) in the population. Next, the instance of the ***Toolbox*** (***toolbox***) is created from the ***base*** module and the created instance of the ***toolbox*** to register and configure GA operations. The ***toolbox*** instance is used to register the fitness function used to evaluate the fitness of an individual using the provided model and fitness function ***get_model_fitness***. Then, the ***toolbox*** instance registers other GA operations namely – ***mate*** (also known as the Crossover operation with the attribute "cxTwoPoint" is used to swap segments between two parents to create new offspring), ***mutate*** (also known as the Mutation operation with the attribute "mutFlipBit" is used to flip a bit in the chromosome with a probability of 0.05) and ***select*** (also known as the Selection operation with the attribute "selTournament" is used to select the best individuals out of a randomly chosen subset of the population) operations respectively. To execute the GA, the initial population is created by calling the ***toolbox.population(n=num_of_population)*** method with ***num_of_population*** given as the number of individuals in the initial population, and the evolutionary process is initiated by calling the ***eaMuPlusLambda*** module in the ***algorithms*** module. The selection, crossover, and mutation are done using the ***`algorithms.eaMuPlusLambda`*** function. This function evolves the population over 10 generations, employing the specified crossover and mutation operators with given probabilities ***`cxpb`*** and ***`mutpb`***, respectively. The model tracks statistics such as the ***avg***, ***min*** and ***max*** fitness values of the population using the ***`stats`*** object. Additionally, a ***`HallOfFame`*** object is used to maintain the best individuals across generations. After evolution, the best individual from the final population is selected using ***`tools.selBest`***, and it is added to the ***`hall_of_fame`*** to ensure that the best solution is preserved. Finally, the ***`hall_of_fame`*** holds the best individuals from all generations and is returned as the output of the genetic algorithm.

For the GA implementation to rank and select the best-performing individual in the population for a given generation, the ***get_model_fitness was created.*** The ***get_model_fitness*** function accepts five arguments namely – ***individual*** (binary string for the selected features), ***predictors*** (the independent variables in the dataset), ***target*** (the dependent variable in the dataset), ***model*** (the model used to analyse the individual), and ***is_func*** (a Boolean variation

use to build neural network model should the model need to be compiled).To calculate the fitness of the individual, the fitness function uses ***cross_val_score*** from the sci-kit learn library to evaluate the performance of a model using cross-validation. The cross-validation method is a statistical tool used to evaluate a model's efficiency on unseen data. The generalization function is particularly useful for evaluating how well your model generalizes to independent data. The cross_val_score splits the dataset into k folds, the model is trained on k-1 folds, and the model is validated on the remaining fold, repeating this process k times. Each iteration returns a list of scores, which can be used to estimate the model's performance by calculating the mean from the list of scores.

5.4 Implementation of Support Vector Machine (SVM)

To implement the SVM model, the ***run_svm_model*** function was used to assemble the SVM model. The function accepts four parameters ***(processed_analysis_dataset, analysis_result, model_name, grid_params)***. The ***processed_analysis_dataset*** parameter contains the analysis dataset which includes both the training and testing datasets for the dependent and independent variables. The ***analysis_result*** parameter is used to track analysis summary for SVM model. The ***model_name*** is a dictionary with 3 keys – (name which tracks the full model's name, short which tracks the abbreviated model name, and analysis which tracks the descriptive title for SVM model analysis). The ***grid_params*** parameter with a default value of None is used to determine if the implementation is for the baseline experiment or for the hyperparameter tuned model. The ***grid_params*** parameter had the following variables defined as follows for the hyperparameter tuned model: kernel was set to ['linear', 'rbf'], C is [0.1, 1, 10, 100, 1000], and gamma [1, 0.1, 0.01, 0.001, 0.0001].

The **SupportVectorMachine** module was invoked to run the SVM model from the scikit-learn library. "123" was assigned to ***random_state*** parameter when creating an instance of the class. Next, the value of the ***grid_params*** is checked, if it contains a value not equal to None, the **GridSearchCV** module with the SVM model instance and the ***grid_params*** are used to create a new instance of a hyperparameter tuned SVM model. To build and evaluate the SVM model, the ***run_model_analysis*** function was called with the (model, model_name, processed_analysis_dataset) parameters. After the analysis has been completed, the ***run_model_analysis*** function returns the evaluation estimates stored in the ***analysis_result*** dictionary.

5.5 Implementation of Random Forest Classifier (RFC)

To implement the RFC model, the ***run_rfc_model*** function was used to assemble the RFC model. The function accepts four parameters ***(processed_analysis_dataset, analysis_result, model_name, grid_params)***. The ***processed_analysis_dataset*** parameter contains the analysis dataset which includes both the training and testing datasets for the dependent and independent variables. The ***analysis_result*** parameter is used to track analysis summary for RFC model. The ***model_name*** is a dictionary with 3 keys same as above but for RFC model analysis). The ***grid_params*** parameter with a default value of None is used to determine if the implementation is for the baseline experiment or for the hyperparameter tuned model. The ***grid_params*** parameter had the following variables defined as follows for the hyperparameter tuned model: n_estimators was set as [25, 50, 100, 150], ['sqrt', 'log2', None]

were assigned variables for ***max_features***, ***max_depth*** was assigned [3, 6, 9], [3, 6, 9] for ***max_leaf_nodes***, criterion ['gini', 'entropy'], bootstrap [True].
The **RandomForestClassifier** module was invoked to run the RFC model from the scikit-learn library. The ***random_state*** parameter was set to "123" when an instance of the class was created. Next, the value of the ***grid_params*** is checked, if it contains a value not equal to None, the **GridSearchCV** module with the RFC model instance and the ***grid_params*** are used to create a new instance of a hyperparameter tuned RFC model. The same functions and steps are followed from the SVM implementation to create and assess the RFC model.

5.6 Implementation of Gradient Boost Classifier (GBC)

To implement the GBC model, the ***run_gbc_model*** function was used to assemble the GBC model. The function accepts four parameters ***(processed_analysis_dataset, analysis_result, model_name, grid_params)***. The ***processed_analysis_dataset*** parameter contains the analysis dataset which includes both the training and testing datasets for the dependent and independent variables. The ***analysis_result*** parameter is used to track analysis summary for GBC model. The ***model_name*** is the same as the above. The ***grid_params*** parameter with a default value of None is used to determine if the implementation is for the baseline experiment or for the hyperparameter tuned model. The ***grid_params*** parameter had the following variables defined as follows for the hyperparameter tuned model: n_estimators was set as [50, 100, 200], learning_rate [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3], and max_depth [3, 5, 7].
The **GradientBoostClassifier** module was invoked to run the GBC model from the scikit-learn library. The ***random_state*** parameter was set to "123" when an instance of the class was created. Next, the value of the ***grid_params*** is checked, if it contains a value not equal to None, the **GridSearchCV** module with the GBC model instance and the ***grid_params*** are used to create a new instance of a hyperparameter tuned GBC model. The same functions and steps are followed to create and assess the GBC model.

5.7 Implementation of Graph Neural Network (GNN)

The Keras Model class was imported from the model module of Tensorflow's Keras library, and was used to create the GNN model. To construct the graph convolutional layers, the layers module additionally imported the Dense class from the Tensorflow Keras library. An instance of the Keras Model class is included in the graph neural network model, which is assembled using the ***build_gnn_model*** function. To assemble the neural network with the input, graph convolutional layers as the hidden layers, and output layers, the input dimension is used in the construction of the first convolutional layer (gc1) and is fed to a dense layer to create convolutional layer (gc2) using the ***"relu"*** activation function. The output layer was assigned ***"sigmoid"***. "binary_crossentropy" was the defined loss function, "adam" as the optimizer value, and "accuracy" as the metrics parameter when the model was constructed. During training, the Adam optimizer dynamically modifies weights and biases to accelerate convergence. A three-parameter neural network model instance was created using the

KerasClassifier wrapper. The parameters are the ***build_gnn_model*** function, ***epochs*** parameter was set to "10" and the ***batch_size*** was set to "32". The model_name parameter is a dictionary with three keys same as the models above. The same functions and steps are followed to build and evaluate the GNN model.

6 Evaluation

The objective of this section is to give a complete analysis of the research study's results and primary conclusions, as well as the ramifications of these findings. The aim of this study was to assess the effectiveness of hybrid machine learning models in the detection of malicious websites. The SVM, RFC, GBC and GNN were selected as the base models to be enhanced using GA to select the best features for this reason. Two experiments were carried out, experiment 1 was done with just the base models and experiment 2 was done with feature extraction using the genetic algorithm. To quantify how well the ML and DL models were implemented, the evaluation metrics described in chapter 3 were used.

6.1 Experiment 1

Experiment 1 was carried out to evaluate the performance of SVM, RFC, GBC and GNN. When the accuracy, F1-score and AUC was calculated, the GNN demonstrated to be the least effective model across recording 75%, 76.19%, and 76.26% respectively. GBC performed best without any hyperparameter tuning or GA feature selection with records of 95%, 94.12%, and 94.44% on the accuracy, F1-score and AUC respectively. RFC measured good performance with above 80% across all metrics. SVM only recorded 80% in accuracy and below 80% on F1-score and AUC. The table below clearly shows the summary of the base models' performance.

Table 2. Performance Score for Base Models

	Accuracy (%)	F1-Score (%)	AUC (%)
SVM	80	77.78	79.80
RFC	85	82.35	84.34
GBC	**95**	**94.12**	**94.44**
GNN	75	76.19	76.26

6.2 Experiment / Case Study 2

Experiment 2 was carried out to evaluate the performance of SVM, RFC, GBC and GNN when hybridized using GA and hyperparameter tuning. The accuracy, F1-score and AUC was quantified, the GA-GBC demonstrated to be the least effective model across with a reduction in the performance. There was a 5% drop in performance in the accuracy, F1-score and AUC. GA-RFC and GA-SVM had almost identical improvements of about 10% across all metrics. The most staggering improvement using GA and hyperparameter tuning was in the GA-GNN. The GA-GNN demonstrated very good performance with a huge jump in accuracy from 75% to 95%, F1-score and AUC skyrocketed from 76.19% to 94.74% and 76.26% to

95.45%, respectively. GA-RFC obtained identical evaluation to GA-GNN. The table below clearly depicts the performance of the GA and hyperparameter tuned models.

Table 3. Performance Score for Improved Models

	Accuracy (%)	F1-Score (%)	AUC (%)
GA-SVM	90	90	90.91
GA-RFC	95	94.74	95.45
GA-GBC	90	87.50	88.89
GA-GNN	**95**	**94.74**	**95.45**

6.3 Discussion

Two experiments were undertaken to determine the effectiveness of optimizing single base SVM, RFC, GBC and GNN models for the purpose of identifying phishing websites. In experiment 1, the models were built as single base without hyperparameter tuning or feature extraction using the GA described in chapter 4. The findings of experiment 1 showed that GNN performed the poorest across all evaluation metrics. When the SVM was coupled with the GA to make the GA-SVM, there was significant improvement across all metrics due to the selection of features using the GA. The GA-SVM recorded 10% increase in accuracy, 12% increase in the F1-score and almost 11% increase in the AUC.

Interestingly, there was a change in the performance of GBC as its performance declined even with the use of GA and hyperparameter tuning across all quantification measures. However, the GNN was the most improved model when combined with GA across all measures. There was a 18%, 19% and 20% surge in F1-score, AUC and accuracy, respectively. The experiment demonstrates that hybrid models could effectively detect phishing websites better than some single base models. The bar chart in figure 4 below shows a comparative summary between the base models and hybrid models. It was revealed in this experiment that the more data which was used in training testing, the lower the performance of the hybrid models.

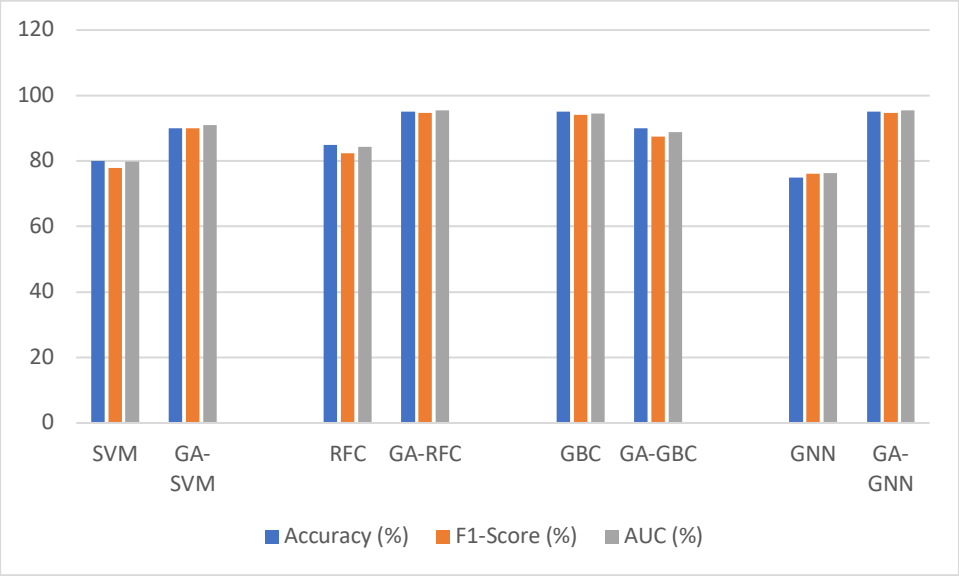

Figure 4. Comparative Summary between the Base Models and Hybrid Models

6.4 GA Feature Selection: Addressing Overfitting and Underfitting

With the combination of the genetic algorithm and hyperparameter tuning model implementation, the efficiency of the model dropped significantly when the number of records in the dataset became large enough to run the models. Further investigation suggests that feature selection can result in overfitting or underfitting the model. By "overfitting", it means minimisation of the feature selection beyond the point at which generalisation performance ceases to improve and subsequently begins to decline suggesting that selecting features that are beneficial only to the training data. Underfitting means the minimisation of the feature selection beyond the point the implemented where the implemented model is too simple to capture the underlying patterns in the data. This means that the model performs poorly on both the training data and unseen test data. Essentially, the model is not complex enough to learn the relationships within the dataset.

To mitigate the effect of overfitting the GA feature selection, robust cross-validation during feature selection to prevent overfitting is adopted and this approach uses a separate test set to validate the final selected feature.

7 Conclusion and Future Work

To conclude, the goal of this study was to determine if phishing website detection might be improved by integrating Genetic Algorithm (GA) with supervised machine learning and deep learning techniques. Based on the critical review of the outcome of the experiments, the phishing detection systems performed far better when hyperparameter tuning and GA were integrated with them; more notable with the GNN than the base SVM, RFC and GBC machine learning models. The experiment indicated that hybrid models offer significant improvements in accuracy, F1-score, and AUC over single-base models. This implies that

using GA for feature selection and optimisation might improve phishing detection systems' effectiveness. This finding has important ramifications for preventing phishing scams on websites. By creating more robust and effective detection systems, organisations may better defend against the exposure of their sensitive data and avoid financial losses and reputational harm. The results underline the significance of GA feature selection in enhancing detection accuracy and demonstrate the potential of hybrid models in tackling the problems related to phishing website identification. Despite the positive findings, it is imperative to acknowledge the study's limitations. One of the limitations is that with more data used for training and testing the lower the performance of the hybrid models; which subsequently makes the single-base models perform better.

Further studies may include the use of different datasets to mitigate the effects leading to the performance decline. Applying more models to the modular architecture might improve system adaptability by making it simpler to switch out models and customise the system for more sophisticated attack scenarios. As cyber threats change, this research provides useful information for enhancing phishing detection systems and strengthening cybersecurity posture.

References

Al-Ahmadi, S. and Lasloum, T. (2020) 'PDMLP: Phishing Detection using Multilayer Perceptron,' International Journal of Network Security and Applications, 12(3), pp. 59–72. https://doi.org/10.5121/ijnsa.2020.12304.

Ali, W. (2017). Phishing Website Detection based on Supervised Machine Learning with Wrapper Features Selection. International Journal of Advanced Computer Science and Applications, 8(9). https://doi.org/10.14569/ijacsa.2017.080910

CybSafe (2023). The ripple effect: How one phishing attack can cause disaster across your organization, CybSafe. Available at: https://www.cybsafe.com/blog/how-can-phishing-affect-a-business/ (Accessed: 21 April 2024).

Das Guptta, S., Shahriar, K. T., Alqahtani, H., Alsalman, D., & Sarker, I. H. (2022a). Modeling hybrid feature-based phishing websites detection using Machine Learning Techniques. Annals of Data Science, 11(1), 217–242. doi:10.1007/s40745-022-00379-8

Ding, Y., Luktarhan, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. Computers & Security, 84, 256–275. doi:10.1016/j.cose.2019.03.018

Elsadig, M. et al. (2022) 'Intelligent deep machine learning cyber phishing URL detection based on BERT features extraction,' Electronics, 11(22), p. 3647. https://doi.org/10.3390/electronics11223647.

Goel, D., & Jain, A. K. (2018). Mobile phishing attacks and Defence Mechanisms: State of Art and Open Research challenges. Computers & Security, 73, 519–544. doi:10.1016/j.cose.2017.12.006

Das Guptta, S. et al. (2022) 'Modeling hybrid feature-based phishing websites detection using Machine Learning Techniques', Annals of Data Science, 11(1), pp. 217–242. doi:10.1007/s40745-022-00379-8.

Harinahalli Lokesh, G., & BoreGowda, G. (2021). Phishing website detection based on effective machine learning approach. Journal of Cyber Security Technology, 5(1), 1–14. https://doi.org/10.1080/23742917.2020.1813396

He, M., Horng, S. J., Fan, P., Khan, M. K., Run, R. S., Lai, J. L., Chen, R. J., & Sutanto, A. (2011). An efficient phishing webpage detector. Expert Systems with Applications, 38(10), 12018–12027. https://doi.org/10.1016/j.eswa.2011.01.046

Huang, Y. et al. (2019) 'Phishing URL Detection via CNN and Attention-Based Hierarchical RNN,' IEEE [Preprint]. https://doi.org/10.1109/trustcom/bigdatase.2019.00024.

Khan, Md.F. (2021) 'Detection of phishing websites using deep learning techniques,' Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(10), pp. 3880–3892. https://doi.org/10.17762/turcomat.v12i10.5094.

Kumar, J. et al. (2020) 'Phishing website classification and Detection Using Machine Learning', 2020 International Conference on Computer Communication and Informatics (ICCCI) [Preprint]. doi:10.1109/iccci48352.2020.9104161.
Li, Y., Yang, Z., Chen, X., Yuan, H., & Liu, W. (2019). A stacking model using URL and HTML features for phishing webpage detection. Future Generation Computer Systems, 94, 27–39. doi:10.1016/j.future.2018.11.004

Nguyen, H. H., & Nguyen, D. T. (2016). Machine learning based phishing web sites detection. Lecture Notes in Electrical Engineering, 371, 123–131. https://doi.org/10.1007/978-3-319-27247-4_11

Prakash, P. et al. (2010) 'PhishNet: Predictive Blacklisting to Detect Phishing Attacks,' IEEE [Preprint]. https://doi.org/10.1109/infcom.2010.5462216.

Rao, R.S. and Ali, S.T. (2015) 'PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach,' Procedia Computer Science, 54, pp. 147–156. https://doi.org/10.1016/j.procs.2015.06.017.

Rao, R.S. and Pais, A.R. (2018) 'Detection of phishing websites using an efficient feature-based machine learning framework,' Neural Computing and Applications, 31(8), pp. 3851–3873. https://doi.org/10.1007/s00521-017-3305-0.

Rao, R. S., Vaishnavi, T., & Pais, A. R. (2020). CatchPhish: detection of phishing websites by inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*, *11*(2), 813–825. https://doi.org/10.1007/s12652-019-01311-4

Safi, A., & Singh, S. (2023). A systematic literature review on phishing website detection techniques. Journal of King Saud University - Computer and Information Sciences, 35(2), 590–611. https://doi.org/10.1016/j.jksuci.2023.01.004

Şahingöz, Ö.K. et al. (2019) 'Machine learning based phishing detection from URLs,' Expert Systems With Applications, 117, pp. 345–357. https://doi.org/10.1016/j.eswa.2018.09.029.

Shahriar, H. and Zulkernine, M. (2012) 'Trustworthiness testing of phishing websites: A behavior model-based approach,' Future Generation Computer Systems, 28(8), pp. 1258–1271. https://doi.org/10.1016/j.future.2011.02.001.

Somesha, M. et al. (2020) 'Efficient deep learning techniques for the detection of phishing websites,' Sādhanā, 45(1). https://doi.org/10.1007/s12046-020-01392-4.

Subasi, A., & Kremic, E. (2020). Leveraging AI and machine learning for societal challenges, cas 2019 comparison of adaboost with multiboosting for phishing website detection. Procedia Computer Science, 168, 272–278. https://doi.org/10.1016/j.procs.2020.02.251

Tang, L., & Mahmoud, Q. H. (2021). A Survey of Machine Learning-Based Solutions for Phishing Website Detection. In Machine Learning and Knowledge Extraction (Vol. 3, Issue 3, pp. 672–694). MDPI. https://doi.org/10.3390/make3030034

Truta, F. (no date) Businesses can lose half of customers after a data breach, research shows, Bitdefender Blog. Available at: https://www.bitdefender.co.uk/blog/businessinsights/businesses-can-lose-up-to-58-of-customers-after-a-data-breach-research-shows/ (Accessed: 26 May 2024).

van Geest, R. J., Cascavilla, G., Hulstijn, J., & Zannone, N. (2024). The applicability of a hybrid framework for automated phishing detection. Computers & Security, 139, 103736. doi:10.1016/j.cose.2024.103736

Vijayalakshmi, M. et al. (2020) 'Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions,' IET Networks, 9(5), pp. 235–246. https://doi.org/10.1049/iet-net.2020.0078.

Wang, M. et al. (2024) 'Phishing webpage detection based on global and local visual similarity', Expert Systems with Applications, 252, p. 124120. doi:10.1016/j.eswa.2024.124120.

Yang, P., Zhao, G. and Zeng, P. (2019) 'Phishing website detection based on multidimensional features driven by deep learning,' IEEE Access, 7, pp. 15196–15209. https://doi.org/10.1109/access.2019.2892066.

Zhuang, W., Jiang, Q., & Xiong, T. (2012). An intelligent anti-phishing strategy model for phishing website detection. Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012, 51–56. https://doi.org/10.1109/ICDCSW.2012.66

Zouina, M. and Outtaj, B. (2017) 'A novel lightweight URL phishing detection system using SVM and similarity index,' Human-centric Computing and Information Sciences, 7(1). https://doi.org/10.1186/s13673-017-0098-1.