ROBERT GORDON UNIVERSITY ABERDEEN

School of Computing, Engineering and Technology

MSci Project Report

Think Before You Click: A Machine Learning Approach to Phishing URL Detection

Navya Shiju

This report is submitted as part of the requirements for the degree of

MSci in Computing Science

I confirm that the work contained in this project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Abstract

Phishing attacks continue to present a significant threat to cybersecurity, exploiting vulnerabilities through deceptive URLs. This project proposes the development of an efficient and accurate phishing URL detection system using machine learning techniques. Focusing only on URL-based features, three individual models were implemented, validated, and optimised. These included Decision Tree, Random Forest, and Support Vector Machine. Further investigation of machine learning model effectiveness led to the development of a hybrid model which was implemented by stacking the optimised Random Forest and SVM models, with XGBoost acting as the meta-classifier. Thorough testing demonstrated that the hybrid model outperformed the individual models in both classification accuracy, with a score of 96.76%, and real-time inference efficiency, making it suitable for further deployment within corporate environments. Stratified K-Fold Cross-Validation was utilised throughout to ensure model reliability and generalisability. In addition, a semi-formal interview was conducted with an industry professional to assess the system's practical applications, confirming its ability to align with real-world needs in enterprise cybersecurity operations. Overall, this project successfully demonstrates that hybrid machine learning models provide a promising solution for high-accuracy, low-latency phishing detection systems within large-scale organisations.

TABLE OF CONTENTS

1.Introduction

Phishing URL detection has become a critical area of research as attackers increasingly exploit deceptive links to compromise systems and steal sensitive information. Traditional machine learning approaches have made strong progress in detecting phishing URLs by analysing URL structures, webpage content, and external reputation services. However, many existing solutions either focus on single-model architectures or are not optimised for real-time application in large corporate environments where speed and accuracy are equally essential.

This project aims to develop a machine learning-based phishing URL detection system that operates solely on URL features, ensuring rapid classification without external queries. It evaluates the effectiveness of Decision Tree, Random Forest, and Support Vector Machine models individually, and further explores the benefits of combining models through a hybrid stacking approach. While previous work has proven the success of ensemble methods in general settings, there is limited research which assesses their practical advantage in a large-scale organisational environment. This project aims to address that gap by comparing individual and hybrid models in terms of predictive accuracy and real-time inference efficiency, to provide insights into the feasibility and advantages of hybrid systems for large-scale, corporate phishing defence.

2. Project Scope

2.1 Introduction

In today's digital climate, where an increasing amount of people are using the Internet as a platform to make online transactions, network, etc. (Ahammad et al., 2022), cybersecurity has risen to the forefront as a critical issue for individuals, businesses, and governments alike. Among the vast variety of cyber threats, phishing remains one of the most prevalent and damaging attack vectors, as agreed by (Information Commissioner's Office, 2024), and while anyone can be the victim of phishing, attackers are looking for high value returns, and therefore, large corporations are often the most targeted and they account for 75% of all attacks, in 2022, which led to $2.7 billion in losses as shown in the FBI's Internet Crime Report, (Federal Bureau of Investigation, 2024).

Phishing is a form of cyberattack in which attackers attempt to mislead individuals into providing sensitive information, such as usernames, passwords, or financial details, by posing as a trustworthy individual or organisation and attacking the "target vulnerabilities that exist in systems due to the human factor", (Khonji et al., 2013). This definition is further agreed upon by (Sapphire, 2023), but the latter goes a step further by clarifying that these attacks typically occur through email, messaging platforms, or malicious websites that closely mimic legitimate services. Phishing can be broadly categorized into several types, including spear phishing (targeted attacks), whaling (high-profile targets), and clone phishing (replicating legitimate communications), (Nadeem et al., 2023).

Figure 1 – Evolution of Phishing Attacks from 1996 to 2020, (Do et al., 2022)

The detection of phishing URLs has become an important area of research within the cybersecurity community**,** as "traditional methods for detecting phishing websites, such as blacklist-based approaches, heuristic analysis, and rule-based systems, have become

increasingly ineffective in combating these evolving threats", (Chy & Hasan, 2024). Machine learning (ML) and deep learning (DL) have surfaced as promising solutions within cybersecurity, offering the capability to detect phishing attempts with increased accuracy and adaptability. These technologies enable the analysis of vast datasets to identify patterns and anomalies in URLs, (Alshingiti et al., 2023).

This section of the dissertation will focus on reviewing the literature around the principal aspects of phishing URL detection, such as, the importance of this research, characteristics of a malicious URL, the various machine learning and deep learning models employed to develop phishing URL detection software, and the possibilities for further enhancement as the research and field grows more robust.

2.2 Phishing

2.2.1 Real World Implications on Large Corporations –

Phishing attacks pose a substantial risk to large corporations across industries, with significant financial, operational, reputational and regulatory consequences, (Yoganandham, 2024). As businesses progressively rely on digital infrastructure for financial transactions, data storage and communication, they become appealing targets for attackers seeking to abuse vulnerabilities through phishing attacks.

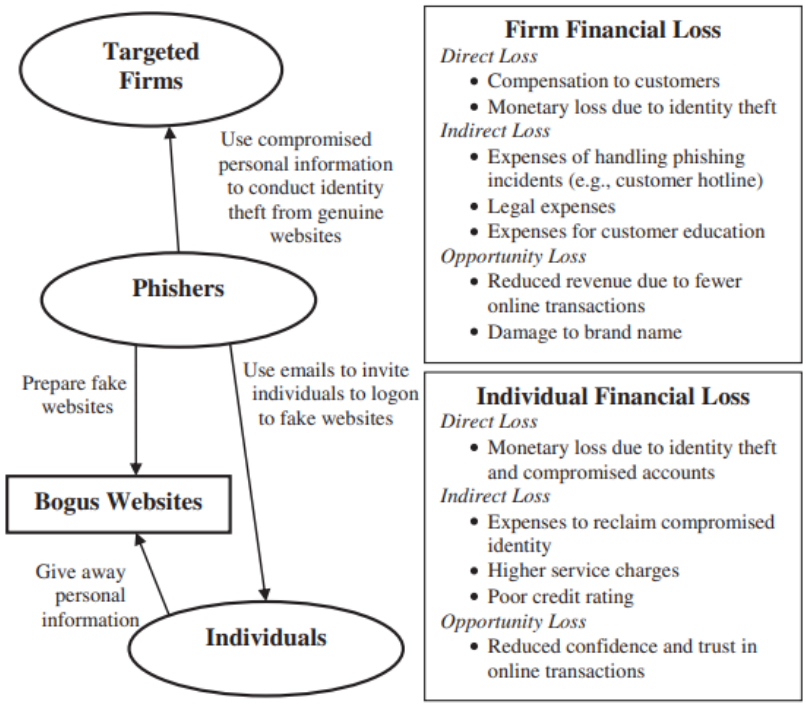

Figure 2 - Phishing attacks and their impacts on firms and individuals, (Bose & Leung, 2014)

2.2.1.1 FINANCIAL LOSSES AND DIRECT COSTS –

Employees are often deceived into disclosing confidential financial information by phishing attacks as they are specifically designed to exploit human weaknesses, and this leads to substantial fiscal losses. These losses can be immense for big corporations. Billions of dollars have been lost, globally, each year, because of phishing attacks, with organisations facing not only the direct cost of stolen finances, but also associated recovery expenses which can

include legal fees and compensation for affected clients and customers, (Ragucci & Robila, 2006). Theft of intellectual property, illegal wire transfers, and unauthorised access to corporate financial accounts can all stem from phishing attacks – once an attacker is within the company's networks, the repercussions can be vast, (Alkhalil et al., 2021).

When exploring the literature around this topic, a paper by Bose & Leung (2014) peaked the researcher's curiosity as it is extremely relevant to the subject matter. The study focuses on the financial impact of phishing alerts on global corporations, specifically analysing how these alerts impact the value of the company. The study used a dataset of 1,942 phishing alerts related to 259 firms from 32 countries, collected from anti-phishing organizations and databases. It aimed to measure the loss in market capitalisation, following phishing alerts using methodologies such as sub-sampling analysis and cross-sectional regression. The analysis showed that phishing alerts led to a significant drop in market capitalisation, with each alert causing a loss of at least $411 million for affected companies.

The solution applied theoretical frameworks such as the Capital Asset Pricing Model (CAPM) and the Fama-French model to evaluate the impact on stock prices and trading volumes. Results indicated that phishing alerts caused significant negative market reactions, especially for financial holding companies and alerts released in 2006-2007. The extent of the impact was also influenced by factors like country of listing, ownership and industry type.

While the study effectively highlights the financial consequences of phishing, its reliance on historical data limits its ability to predict the impact of future phishing incidents. Additionally, the findings may not be generalisable to all industries or firms outside the sample, and the rapid evolution of phishing tactics suggests that continuous adaptation of detection and prevention strategies is necessary for long-term effectiveness. The study provides valuable insights into the economic damage caused by phishing, but further research with more dynamic models and updated data is needed to address evolving threats.

2.2.1.2 Operational Disruption –

Beyond financial losses, phishing attacks can severely disrupt critical operations. When phishing attempts lead to the installation of malware or ransomware, entire networks can be compromised, resulting in downtime that disrupts essential business functions, (Ali et al., 2024). For instance, when phishing attacks target key employees within a company, attackers can gain access to internal systems, without being detected straight away, and prevent employees from using critical databases and tools. Disruptions like these can decrease productivity, hinder operations, and delay a company's ability to meet regulatory obligations or client demands, (Nadeem et al., 2023). To mitigate the consequences of these attacks, substantial resources are often required, and the long-term operational setbacks are not acceptable, especially in industries such as healthcare where continuous access to data and systems are vital, (Arapi, 2018).

2.2.1.3 Reputational Damage –

The fundamental aspect to success for a company is the reputation they build, and a breach of security can damage the public's trust in a company's ability to safeguard sensitive client information, which leads to long-term damage to the organisation's image, (Frank, 2024). Reputational damage can also present itself in the form of lost business proposals, as clients

often become unwilling to collaborate with a company that does not comply with standard security measures and prove themselves to be vulnerable to digital threats, (Sopna & Ahmad, 2021). In highly competitive markets, the perceived inability to protect data can erode market share and lead to declining revenues, (Perera et al., 2022).

2.2.1.4 Regulatory and Compliance Challenges –

Large companies who operate under strict regulations, like the General Data Protection Regulation (GDPR) in the EU, can face considerable compliance challenges due to phishing attacks. GDPR requires organisations to implement thorough data protection practices and necessitates the company to release a notification of the breach within 72 hours, (intersoft consulting, n.d.). Failure to meet this requirement, or demonstrate adequate preventative measures, in light of a breach of personal data due to a phishing attack, can result in immense fines of $20 million, or 4% of the annual global turnover, whichever is higher, (IT Governance Ltd, n.d.). Acceptable preventative measures could include encryption, access control, and routine security audits, to mitigate data breach risks.

Companies in specific sectors must also follow industry-specific data protection standards, that extend beyond GDPR. The Health Insurance Portability and Accountability Act (HIPPA) is enforced within healthcare, with severe data protection requirements with non-compliance in the light of phishing attacks leading to significant penalties and loss of trust, (Nahai, 2019). The Payment Card Industry Data Security Standard (PCI DSS) enforces thorough security measures within payment processing, and failure to comply can result in fines and in some instances, the suspension of processing privileges, if a cyberattack did take place, (Boese, 2020). Companies are incentivised to continuously strengthen their cyber security frameworks as phishing attacks become more complex, (Verma & Shri, 2022).

In order to maintain operational integrity, uphold public trust, and comply with various regulatory guidelines, investments in advanced detection systems and comprehensive employee training have become essential, (Arachchilage & Love, 2014).

2.2.2 Feature Extraction within Phishing URL Characteristics –

The ability to distinguish between legitimate and malicious URLs is the basis of phishing detection research, as researched by Sankhwar et al. (2019). Phishing URLs are difficult to separate from genuine sites, as they are designed to imitate the legitimate URL and the subtle changes embedded within the phishing URL are not easily identifiable, which leads to users disclosing private information and rendering themselves vulnerable. However, patterns arise, and these fraudulent URLs typically display certain characteristics that can be analysed and used to detect incoming phishing attacks, (Verma & Das, 2017).

The features that will be studied within this dissertation fall under the category of “lexical features” and they refer to the textual structure of the URL itself. These attributes are among the most recognisable indicators in distinguishing phishing URLs from legitimate ones, as phishing URLs often rely on linguistic techniques to deceive users, (Singh et al., 2024).

A paper by Gupta et al. (2021) presents some of the key lexical features:

- URL Length: Attackers often use extended URLs to obscure the actual domain name or insert misleading subdomains and directories.
- Number of Subdomains: Attackers often create URLs with several subdomains to confuse users into believing they are visiting a legitimate website (e.g., “secure.login.bank.com” instead of “bank.com”).
- Presence of Suspicious Characters: Phishing URLs often contain special characters such as hyphens, numbers, and special symbols (e.g., @, %, &, and others) that are less common in legitimate URLs. These characters are used to create URLs that resemble legitimate domains while hiding the malicious intent.
- Use of IP Addresses in URLs: Instead of a recognizable domain name, some phishing URLs use IP addresses directly (e.g., “http://192.168.0.1”) to avoid detection and confuse users.
- Misspelled Domains: Attackers frequently register domains with slight misspellings of legitimate websites (e.g., “g00gle.com” instead of “google.com”) or use visually similar characters from different alphabets (e.g., using Cyrillic characters in place of Latin characters).

2.3 Machine Learning and Deep Learning Models for Phishing URL Detection

The field of Phishing URL detection has been revolutionised by leveraging both feature-based machine learning (ML) and more advanced deep learning (DL) models as they provide increasingly adaptive solutions to increasingly complicated phishing approaches, in comparison to more traditional approaches, (Divakaran & Oest, 2022). This is evident as the models can analyse large datasets, to detect subtle patterns and anomalies embedded within the URLs, which leads to increased accuracy and a scalable detection of phishing attacks, (Kayode-Ajala, 2022).

Machine learning algorithms, such as Random Forests, Decision Trees and Support Vector Machines, provide solid frameworks for feature-based classification, (Chen *et al.*, 2020). Meanwhile, deep learning architectures, including neural networks and convolutional neural networks (CNNs), offer superior capabilities in capturing complex, non-linear relationships within the data, (Ahmed *et al.*, 2023).

This section investigates various ML and DL models currently employed in phishing URL detection, examining their underlying principles, comparative strengths, and limitations.

2.3.1 Machine Learning for Phishing URL Detection

2.3.1.1 Decision Tree

Decision Trees are a well-established algorithm in machine learning, often used for tasks involving classification and regression. These models work by creating a flowchart-like representation of data, obtained from the dataset features, and they are known for their intuitive and interpretable structure, (Suthaharan, 2016). The algorithm forms a hierarchical structure, by dividing the dataset recursively, where the internal nodes link to specific

features, branches signify possible feature values and the leaf nodes signify the predicted outcomes or classes, (Taeho, 2021).

Decision Trees have a great aptitude for learning complex decision boundaries and adapting to missing values and outliers as they are capable of handling both numerical and categorical features, (Suthaharan, 2016). In addition to that, these models are widely used in feature selection, enabling the categorisation of significant features within the dataset. However, they do have a tendency to overfit, particularly in instances involving very deep and intricate trees. To prevent this flaw, pruning and a maximum tree depth are used, (Taeho, 2021).

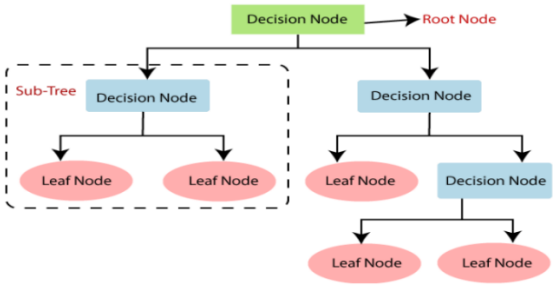

Figure 3 – Decision Tree Algorithm, (Omari, 2023)

2.3.1.2 Random Forest

Random Forest is a popular algorithm, widely recognised for its ability to integrate several Decisions Trees to increase the accuracy of its predictions. This model is robust and adaptable, which makes it well suited for regression and diverse classification tasks, (Biau, 2012). In essence, a Random Forest creates a selection of Decision Trees, and each of them is trained on a different subset of the training data, via bootstrapping. Additionally, at each node of a tree, only a random subset of features is considered for splitting, introducing controlled randomness that prevents overfitting and enhances generalisation capabilities, (Rigatti, 2017). The final prediction in Random Forest is determined by combining the outputs of individual trees, using voting for classification tasks or averaging for regression tasks, (Omari, 2023). This methodology effectively reduces variance and enhances the model's overall performance.

There are many benefits to using a Random Forest model. These include maintaining robustness in the presence of outliers, automatic feature selection, estimation of feature importance, and management of high dimensional data, which combine to provide insightful interpretations of the patterns within the data, (Rigatti, 2017). However, there are limitations such as being extremely resource dependent and reduced interpretability, compared to single Decision Trees. To lessen the impact of these limitations, hyperparameter tuning, such as adjusting the number of trees and setting the maximum depth of each tree is typically implemented, (Omari, 2023).

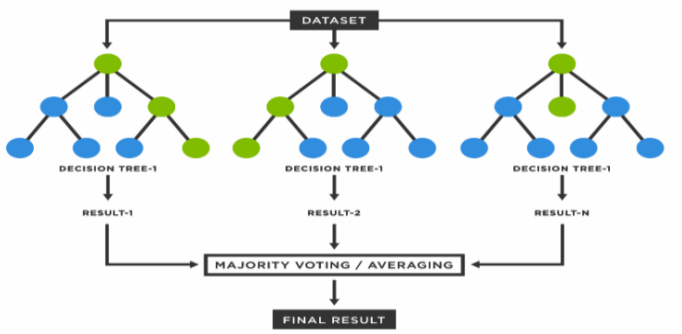

Figure 4 – Random Forest Algorithm, (Omari, 2023)

2.3.1.3 Support Vector Machine

Support Vector Machine (SVM) is widely used for both regression and classification tasks as it is a powerful supervised learning model. The main goal of SVM is to establish an optimal hyperplane which successfully divides the data points into distinct classes within the feature space, (Gholami and Fakhari, 2017). A hyperplane "serves to separate (i.e., "classify") observations belonging to one class from another based-on patterns of information about those observations called features" as stated by Pisner and Schnyer (2020).

By manipulating the data in this way, SVM can identify a hyperplane that maximises the margin between classes, thereby enhancing its capacity for generalization, (Cervantes *et al.*, 2020). A key feature of SVM is its focus on identifying the hyperplane, which not only separates classes but also maximizes the distance to the nearest data points. This characteristic makes SVM robust to outliers and enables it to model non-linear decision boundaries effectively through various kernel functions. SVM is versatile as it supports both binary and multiclass classification tasks. For binary classification, SVM seeks a decision boundary that efficiently differentiates between the two classes, (Pisner and Schnyer, 2020).

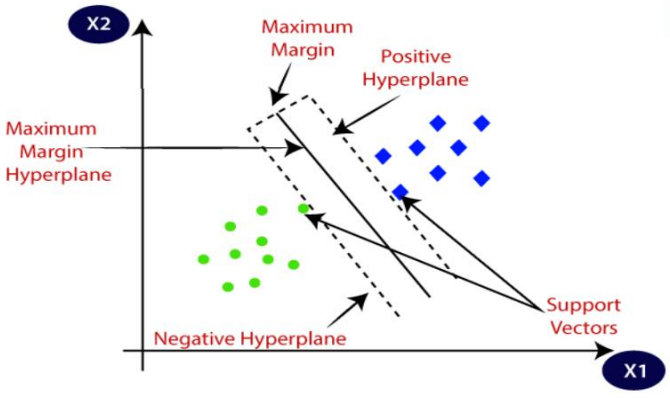

Figure 5 – Support Vector Machine Algorithm, (Omari, 2023)

When exploring the literature around this topic, a paper by Mahajan & Siddavatam (2018) caught the attention of the researcher as the models used within it mirror the models explored within this dissertation exactly. The paper naturally addresses the problem of phishing website detection using machine learning algorithms. The dataset used in this study was sourced from Alexa for benign URLs and PhishTank for phishing URLs, consisting of 36,711 URLs (17,058 benign and 19,653 phishing URLs).

The solution involves extracting various URL-based features, such as the presence of IP addresses, @ symbols, the number of dots in the URL, and the use of HTTPS tokens, which are then analysed by machine learning algorithms. The study compares the performance of the Decision Tree, Random Forest and SVM classifiers, focusing on accuracy, false positive, and false negative rates. Random Forest achieved the highest accuracy of 97.14% with the lowest false negative rate, outperforming the other models.

While the proposed solution effectively detects phishing websites with high accuracy, the study is limited using a relatively small dataset and the reliance on static features. As phishing techniques evolve, these features may become less effective, and the system might struggle to detect new, sophisticated attacks. The conclusions drawn were promising, but future work could explore hybrid approaches that combine machine learning with other detection methods, such as blacklists, to further improve performance and adaptability and combine established technology with innovative methodologies.

After continuous research, a paper by Karim et al. (2023) further demonstrated that the increasing sophistication of phishing attacks requires more robust detection techniques. This paper employs a hybrid machine learning approach based on URL features and the dataset used in this study was sourced from Kaggle and consists of over 11,000 phishing and legitimate URLs, with 33 extracted attributes such as domain length, presence of IP addresses, and subdomains.

The solution proposed involves using several machine learning models, including Decision Tree, Random Forest, Naïve Bayes, Gradient Boosting, and Support Vector Machine (SVM), along with a hybrid ensemble model (LSD: Logistic Regression, SVM, and Decision Tree). The model's performance was evaluated using precision, recall, accuracy, F1-score, and specificity, with the hybrid model achieving the highest accuracy of 98.12%. Techniques like canopy feature selection, cross-validation, and grid search hyperparameter optimization were applied to enhance the model's performance.

While the hybrid model showed strong results, a critical limitation is the potential overfitting of the model to the dataset, as well as the reliance on URL-based features, which may not always generalise well to evolving phishing strategies. The authors suggest that future work should integrate more diverse data and possibly combine this approach with real-time detection systems. Although the model performs well in controlled experiments, real-world applicability could face challenges in dealing with more sophisticated phishing tactics.

2.3.2 Deep Learning for Phishing URL Detection

2.3.2.1 NEURAL NETWORK

Neural Networks are ideal for building complex, nonlinear models with large datasets. As computer power continues to increase and GPUs are also helping with intense computation, neural networks are performing very well, (Siddiq et al., 2022). This is further agreed by (Islam et al., 2019), but the latter goes further by saying that these models are particularly effective for identifying patterns and predicting events when there is a large database of prior examples to learn from.

2.3.2.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) provide more promising and robust results as they reduce the network complexity and hasten the learning process due to them being ideally suited for efficient and fast feature extraction from raw and complex data, (Do et al., 2022). They can extract key features in tokens or sequences of tokens, regardless of the minor positional differences across diverse textual content, (Altwaijry et al., 2024), making them ideal for phishing URL detection.

As can be seen from this research, deep learning models, with their ability to uncover complex patterns in large datasets, offer ever evolving applications in cybersecurity, particularly by enhancing the accuracy and adaptability of threat detection systems.

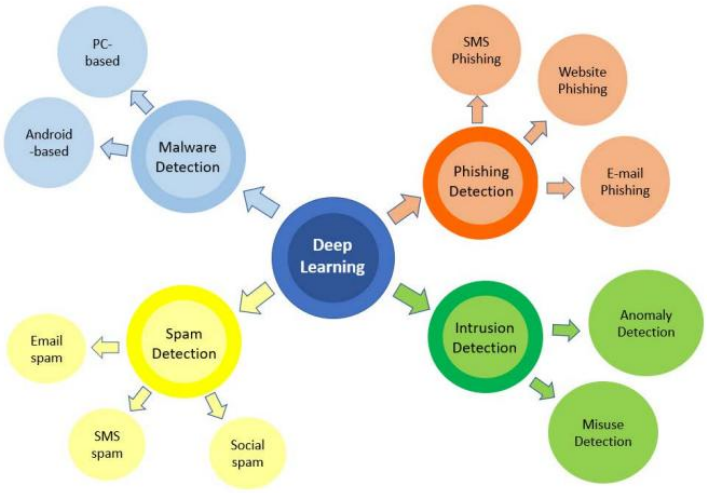

Figure 6 – Main Branches of Applying Deep Learning within Cybersecurity, (Do et al., 2022)

When exploring the literature around this topic, a paper by Siddiq et al. (2022) caught the attention of the researcher as it was an excellent comparative study of various deep learning models. The paper naturally focuses on the problem of detecting phishing websites using deep learning techniques. The dataset used in this study was sourced from the UCI Machine Learning Repository, containing 10,055 samples with 30 categorical features.

The proposed solution involves the implementation of two models: a standard neural network and a Convolutional Neural Network (CNN). Both models aim to detect phishing websites based on the extracted features, which include characteristics related to URL structure, domain information, and HTML features. The models were trained using an 80%-20% train-test split, and various hyperparameters were tuned to optimize performance. The standard neural network achieved 94.8% accuracy, while the CNN model achieved 93.6% accuracy with 98.4% recall.

Critically, while the deep learning approach showed promising results in phishing detection, the study lacks a discussion on real-time application, especially regarding the feature extraction process in live environments, which could be time-consuming. The conclusions drawn are optimistic, but future work could include improvements in model regularisation and scalability, along with real-time testing, to make the detection system more practical for real-world use. Additionally, incorporating larger and more diverse datasets would enhance the model's robustness and effectiveness.

2.4 Conclusion

The purpose of the literature investigation was to conceptualise the key factors within Phishing URL detection and that was accomplished with a thorough critical review of the existing literature. It was discovered that phishing URL detection has become a critical area of research within cybersecurity, with both ML and DL models offering promising approaches to identifying malicious URLs. This literature review has explored key models—such as Decision Trees, Random Forests, SVMs, CNN's, etc.—and evaluated their effectiveness in distinguishing malicious URLs from legitimate ones, through a range of lexical features.
While phishing detection applications have advanced further with these models, significant gaps do remain, and these can be investigated further within the confines of this project through practical implementation.

One of the most significant limitations that is highlighted within the literature shows that there is a real focus on individual models, with minimal exploration of hybrid approaches that could utilise multiple ML or DL algorithms. Hybrid models that leverage the strengths of multiple models to isolate a broad range of Phishing URL characteristics could improve the accuracy of the models by a substantial amount. This project could contribute to this research further by implementing and evaluating a hybrid model, alongside singular models, to determine their effectiveness in Phishing URL detection.

Another notable challenge within the existing literature is the real-time application of these models, particularly in environments where low-latency detection is critical. Many of the existing approaches that utilise ML/DL models are heavily resource dependent and could be challenging to deploy in high-traffic, commercial settings where a prompt response is vital. Within the confines of this project, optimising models for real-time processing could be explored further to allow for better insights in balancing accuracy with speed, and increasing practicality in a real-world setting.

In conclusion, while machine learning and deep learning has greatly enhanced phishing URL detection, the scope of this project could help to address the lack of hybrid modelling and real-time efficiency in most of the studies reviewed. By focusing on these areas, this project aims to advance the accuracy, and practicality of ML-based phishing URL detection within large corporations, supporting the broader objective of creating a more secure digital environment.

As a result of the extensive research conducted within this phase of the project, the main research questions that will be taken forward and explored further are:

1. How will a hybrid machine learning model perform in terms of real-time efficiency when compared to individually trained models?
2. Which Machine Learning models will have the highest accuracy in predicting the authenticity of a Phishing URL?

3. Design and Methodology

This section outlines the design choices and methodological approach used within this project for building a phishing URL detection system using machine learning. The aim of the project was to develop and evaluate an effective classification pipeline, capable of accurately detecting phishing URLs with minimal latency for real-time application within a corporate cybersecurity environment.

3.1 Tools and Environment

The project was implemented using Python in the Google Colab environment. Python is the most commonly used language in the machine learning community (Hillier, 2020), as it offers a wide range of libraries which were used extensively throughout the model development, optimisation, and performance visualisation phases of this project. Google Colab was chosen as the development platform due to its ease of use and GPU acceleration (Van Den Reym, 2020), which would be beneficial when testing the implemented models to save the researcher time. Additionally, both the language and platform were tools the researcher was already familiar with, enabling efficient and confident development throughout the project lifecycle.

3.2 Dataset Selection

The dataset used for this study is the Web Page Phishing Detection Dataset, sourced from Kaggle (Tiwari, 2021)**.** This dataset includes 11,430 URLs, each with 87 extracted features, that is evenly distributed between phishing and legitimate classes (50% each), making it suitable for binary classification tasks. The dataset includes features extracted from three sources. 56 features are based on the structure and syntax of URLs, 24 features are extracted from the content of corresponding webpages and 7 features come from querying external services.

As the objective was to build a URL-based phishing detection system, only features directly related to the URL structure and syntax were kept. This decision ensured the model could make predictions without relying on content crawling or external service queries, which reduces latency and aligns the system with potential real-world deployment where immediate classification is critical.

3.3 Choice of Models

The project used three machine learning models as a foundation to build upon for classification:

- Decision Tree - Chosen for its interpretability and fast inference time. It serves as a strong baseline and provides insight into how individual features influence classification decisions (Suthaharan, 2016).
- Random Forest - Selected for its ensemble-based approach, which improves generalisation by averaging multiple decision trees to reduce overfitting (Biau, 2012).
- Support Vector Machine (SVM) - Included for its effectiveness in binary classification tasks, particularly in high-dimensional spaces (Cervantes *et al.*, 2020).

These models were chosen due to their performance in similar cybersecurity applications, ease of implementation, and compatibility with datasets like the one used in this study.

3.4 Evaluation Metrics

To thoroughly assess model performance, several standard machine learning evaluation metrics were employed (Naidu, Zuva and Sibanda, 2023).

The model was evaluated in terms of Accuracy as it measures the proportion of correct predictions out of the total number of samples.

Precision was also included as a metric as it indicates the proportion of URLs classified as Phishing that were indeed malicious. A high precision score minimises the chance for false positives which is an important factor to note when considering both the authenticity of a URL, and future development as an end-user could potentially be negatively impacted by a false-positive assignment.

The Recall metric was incorporated as it measures the proportion of actual phishing URLs correctly identified by the model. A high recall minimises the chance for false negatives.

F1-Score represents the mean of precision and recall, ensuring a balanced assessment as both metrics are important.

Inference Time (ms) evaluates the real-time efficiency of each model by measuring how quickly it can classify new inputs (Dominguez, 2025). The inclusion of inference time as a metric was particularly relevant to the project's aim of deploying a phishing detection system in a corporate environment, where the ability to respond to threats in real time is essential to prevent malicious attacks.

3.5 Project Timeline and Milestones

The implementation phase of the project was completed over approximately two and a half months, followed by a one-month period for documentation and report dissemination. Figure 7 illustrates the project timeline.

	Duration											
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Research and Planning												
Literature Review on Phishing Detection Techniques												
Identification of Suitable Models and Dataset												
Data Acquisition and Pre-Processing												
Exploring of Dataset												
Feature Filtering and Selection												
Baseline Model Implementation												
Training and Evaluation of Baseline Decision Tree, Random Forest and SVM Classifiers												
Model Optimisation and Validation												
Hyper-Parameter Tuning using GridSearchCV												
Stratified 10-Fold Cross-Validation												
Hybrid Model Development												
Optimised Random Forest + SVM + XGBoost Classifier												
Final Validation and Evaluation												
Cross Validation and Inference Time Calculations												
Report Dissemination												

Figure 7 – Gantt Chart Detailing Project Timeline

The key milestones identified and successfully achieved during the project are:

- Development and evaluation of baseline models
- Optimisation through hyperparameter tuning and validation

- Implementation of an optimised hybrid model
- Comparative analysis and performance benchmarking of all models

3.6 Requirements Analysis

To be considered successful, this project must achieve the following requirements, both functional and non-functional.

3.6.1 Functional Requirements –

- Dataset Pre-Processing - The project must be tailored towards phishing URL detection by filtering and using only URL-based features from the dataset, excluding webpage content and external service features.
- Model Development - The project must implement three baseline machine learning models: Decision Tree, Random Forest, and Support Vector Machine (SVM).
- Cross-Validation - Each model must undergo Stratified K-Fold Cross-Validation to evaluate its generalisability and avoid overfitting.
- Model Optimisation - Hyperparameter tuning must be implemented for each baseline model using GridSearchCV to improve predictive performance.
- Hybrid Model Implementation - A hybrid model must be developed by stacking the optimised Random Forest and SVM models, with XGBoost acting as the meta-classifier.
- Performance Evaluation - The project must evaluate all models using key metrics: Accuracy, Precision, Recall, and F1-Score.
- Real-Time Efficiency Measurement - The project must measure and report the inference time (in milliseconds) of each model to assess real-time deployment suitability.

3.6.2 Non-Functional Requirements –

- Efficiency - The models must be able to make predictions quickly enough (ideally within a few milliseconds) to support real-time phishing detection.
- Accuracy – The models must achieve high accuracy when predicting the legitimacy of phishing URLs to minimise the likelihood of false positives.
- Scalability - The system should be designed to handle larger URL datasets in future applications without significant performance degradation.
- Reliability - The models must produce consistent results across different subsets of data, with minimal variance during cross-validation.
- Reproducibility - The project must allow results to be reproducible by others under the same conditions by using fixed random states and clear pre-processing steps.

4. Implementation

4.1 Pre-Processing

The initial stage of the implementation involved preprocessing the dataset, which consisted of 11,430 labelled URLs with 87 features. Since the aim of the project was to develop a phishing URL detection system, only features that directly corresponded to the structure and syntax of URLs were selected for training. These accounted for 56 of the 87 features and were extracted from the original dataset using column filtering techniques, explicitly excluding the 24 features extracted from the content of the corresponding webpages and the 7 features obtained by querying external services. This design decision ensured that the detection model could operate without relying on additional web content or network requests.

To prepare the dataset for training, labels indicating phishing or legitimate URLs were assigned in binary format, where phishing was represented as 1 and legitimate as 0 (Wong, 2023). The dataset was then split into training and testing subsets using the *train_test_split()* function, with 80% of the data allocated for training and 20% reserved for testing. The split was stratified using the *stratify=y* parameter to ensure the 50:50 class balance remained in both subsets. *StandardScaler()* was used to normalise the feature values to preventing models sensitive to feature scaling, like SVMs, from being biased by differing value ranges (Mulani, 2022).

4.2 Implementation, Validation and Evaluation of Baseline Models

After the pre-processing phase, three baseline machine learning models were implemented: Decision Tree, Random Forest, and Support Vector Machine. The Decision Tree was implemented using *DecisionTreeClassifier(random_state=42)*, ensuring a simple structure that allows for early benchmarking (Pramod, 2023). Setting the random state to 42 ensured reproducibility of results across multiple runs. The Random Forest was initialised as *RandomForestClassifier(n_estimators=100, random_state=42)*, as it utilised a collection of 100 decision trees to reduce variance and improve predictive stability (Shafi, 2024). The SVM model was built using *SVC(kernel='linear', probability=True, random_state=42)*. The use of a linear kernel ensured computational feasibility given the dimensionality of the data, while *probability=True* enabled the extraction of class probability estimates for following stacking operations (Ansari, 2017).

Each model was trained using *the .fit()* method on the training dataset and evaluated on the test set using *the .predict()* method. The evaluation metrics calculated for each included accuracy, precision, recall, and F1-score. The results of these initial evaluations showed that, while all models achieved reasonably high performance, Random Forest and SVM outperformed the Decision Tree model in terms of both precision and recall. In particular, the Random Forest baseline model achieved an accuracy of 92.83%, a precision of 92.23%, a recall of 93.52%, and an F1-score of 92.87%, indicating a strong balance between correctly identifying phishing URLs and minimising false positives. The SVM demonstrated slightly lower performance but remained effective, whereas the Decision Tree displayed noticeably weaker results across all metrics, particularly in precision and recall, which are critical when identifying phishing threats. These results suggested that although all models were reasonably effective, Random Forest and SVM demonstrated a stronger ability to accurately

identify phishing URLs, whilst maintaining a lower rate of false positives, making them strong candidates for further optimisation and hybrid model development.

To ensure fairness and to reduce overfitting, each baseline model was validated using Stratified K-Fold Cross-Validation (*StratifiedKFold*, no date), configured with 10 folds through the *StratifiedKFold(n_splits=10)* class. The *cross_val_score()* function was used to calculate the mean accuracy and standard deviation across all folds. This approach ensured that the class balance was maintained in each training-validation split and that the model's performance was consistent across multiple subsets of the data. It was observed that both Random Forest and SVM produced relatively low standard deviations, indicating stable performance, while Decision Tree displayed greater variance.

4.3 Optimisation and Evaluation of Baseline Models

After validating the individual models, hyperparameter tuning was performed using GridSearchCV to find the optimal configurations for each model (Shah, 2021). For Decision Tree, the grid included parameters such as *max_depth* and *min_samples_split* (Badrinarayan, 2024). For Random Forest, the search considered combinations of *n_estimators*, *max_depth*, and *min_samples_leaf* (Koehrsen, 2018). For SVM, the tuning stage confirmed the use of a linear kernel (vaibhavkumar, 2023). Each search was performed with 5-fold cross-validation, and the best parameters were applied to retrain the optimised models. Performance metrics were re-evaluated after tuning, which showed improvement across all models, especially in the case of Random Forest and SVM.

4.4 Implementation and Evaluation of Hybrid Model

Now that the optimised individual models have been developed, the next stage involved implementing a hybrid model using a stacking method (Kharwal, 2024), (Kaabar, 2024). The hybrid model was built using Random Forest and SVM as these were the best performing models, and XGBoost as the meta-classifier. XGBoost was selected due to its high predictive accuracy, and its ability to perform regularisation well, which helps prevent overfitting in hybrid models (Tuychiev, 2023). This was implemented by generating class probability predictions from both base models using the *.predict_proba()* method. The predicted probability of the phishing class was extracted using slicing *rf_preds = best_rf.predict_proba(X_test)[:, 1]* and *svm_preds = best_svm.predict_proba(X_test)[:, 1]*. These predictions were then joined to form a new feature matrix: *stacked_features = np.column_stack((rf_preds, svm_preds))*. This stacked feature matrix acts as the input to the hybrid model, allowing it to learn higher-level patterns by using the combined strengths of the individual models to make more accurate final predictions (Kavlakoglu and Russi, 2025).

```
#generating stacked predictions from the optimised models - random forest and SVM as they were the best performing models

#predicting the class probabilities for the test set using the optimisied random forest model - only the probability of the legitimate class is extracted here
random_forest_preds = best_random_forest.predict_proba(X_test)[:, 1]
#predicting the class probabilities for the test set using the optimisied SVM model - only the probability of the legitimate class is extracted here
svm_preds = best_svm.predict_proba(X_test)[:, 1]

#stacking the predictions as new feature set - combinied two arrays to create a two-feature matrix
stacked_features = np.column_stack((random_forest_preds, svm_preds))

#training the classifier - XGBoost
meta_model = XGBClassifier( eval_metric="logloss", random_state=42)     #logloss is used as this is a binary classification task
meta_model.fit(stacked_features, y_test)      #fitting the XGBoost classifier using the stacked preditions as features

#making final predictions with hybrid model to determine URL authenticity
stacked_preds = meta_model.predict(stacked_features)

print("Hybrid model trained successfully!")
```

Figure 8 – Hybrid Model Implementation with XGBoost Classifier

```
#creating a function to cross-validate the hybrid model to evaluate how well it generalises
def cross_validate_hybrid_model():
    hybrid_scores = []      #initialising an empty list to store the accuracy scores for each fold

    #iterating over the training and validation indices
    for train_index, test_index in kf.split(X_train, y_train):
        #training the optimised models on current training fold
        best_random_forest.fit(X_train.iloc[train_index], y_train.iloc[train_index])
        best_svm.fit(X_train.iloc[train_index], y_train.iloc[train_index])

        random_forest_preds_fold = best_random_forest.predict_proba(X_train.iloc[test_index])[:, 1]   #generating the predictions for the validation fold using the random forest
        svm_preds_fold = best_svm.predict_proba(X_train.iloc[test_index])[:, 1]                    #generating the same by usinf the SVM
        stacked_features_fold = np.column_stack((random_forest_preds_fold, svm_preds_fold))      #stacking the predictions into a new feature set for the hybrid model

        meta_model.fit(stacked_features_fold, y_train.iloc[test_index])   #training the XGBoost on stacked features
        stacked_preds_fold = meta_model.predict(stacked_features_fold)    #making predictions with the model using the stacked features

        #calculating the accuracy for this fold
        hybrid_scores.append(accuracy_score(y_train.iloc[test_index], stacked_preds_fold))

    #printing the overall cross-validation results for the hybrid model
    print("----- Optimised Hybrid Model Cross-Validation Results -----")
    print(f"Mean Accuracy: {np.mean(hybrid_scores):.4f}")
    print(f"Standard Deviation: {np.std(hybrid_scores):.4f}")

#running the cross-validation function for hybrid model
cross_validate_hybrid_model()
```

Figure 9 – Function to Cross-Validate the Hybrid Model to Ensure Effective Generalisability

The XGBoost classifier was initialised using *XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42*. The *logloss* metric was selected as it is particularly well-suited for binary classification tasks, as it measures the uncertainty of the model's probability outputs, and penalises confident but incorrect predictions more heavily, which encourages the model to produce reliable probabilities (Saha, 2018). It was then trained on the new stacked feature set. The hybrid model's performance was evaluated again using the standard classification metrics. A specific cross-validation method was also implemented to validate the hybrid model using the same 10-fold stratified approach. In each fold, the base models were retrained on the training set, their predictions were stacked, and the XGBoost classifier was trained and validated. This process helped ensure that the hybrid model was not overfitted to the test set and could generalise well across different data partitions.

4.5 Evaluation of Real-Time Efficiency

Finally, the optimised individual models and the hybrid model were compared in terms of classification performance and inference time. To evaluate real-time efficiency, inference time was measured using Python's *time* module (Code, 2024), (Dominguez, 2025).

To conclude the implementation, a thorough performance comparison was conducted across all models, which will be explored further in the next section.

5. Testing and Results

The evaluation phase of this project was carried out in several stages to thoroughly assess the performance of the machine learning models by looking at the performance metrics and cross-validation results. This section outlines the progression of model performance, from initial baseline implementations to the final optimised and validated hybrid model, including both accuracy and real-time efficiency metrics.

Accuracy measures the overall proportion of correctly classified URLs, while precision focuses on how many URLs predicted as phishing were actually phishing. Recall measures the model's ability to correctly identify all phishing URLs, and the F1-score provides a balanced combination of both precision and recall (Naidu, Zuva and Sibanda, 2023). Together, these metrics provide a more complete understanding of a model's strengths and weaknesses, which is particularly important in a cybersecurity context where false positives and false negatives carry different risks.

The initial evaluation of the baseline models, Decision Tree, Random Forest, and Support Vector Machine, showed a significant variation in classification performance. The baseline Random Forest model outperformed the others, with an accuracy of 92.83%, a precision of 92.23%, a recall of 93.53%, and an F1-score of 92.87%. In contrast, the Decision Tree model, produced a lower accuracy of 87.93% with slightly reduced precision and recall. The Support Vector Machine performed reasonably, but significantly worse than the others, achieving an accuracy of 84.08% and an F1-score of 84.26%. These results set the initial benchmark for further model optimisation.

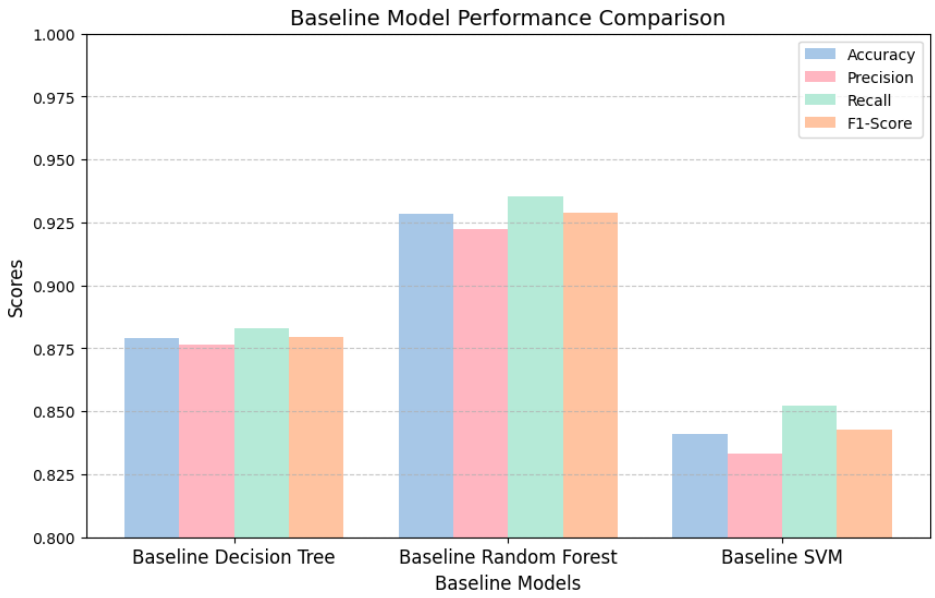

Figure 10 – Bar Graph to Visualise and Compare Baseline Model Performance

As shown in Figure 10, the baseline performance comparison highlights the clear superiority of the Random Forest model over Decision Tree and SVM across all key evaluation metrics, particularly in terms of recall and F1-score. This initial benchmarking was important in establishing that ensemble-based approaches, like a Random Forest, were more effective at

identifying the distinct patterns of phishing URLs, which built a strong foundation for optimisation in the next stages of the project.

To assess the consistency and generalisability of the models, 10-fold stratified cross-validation was implemented. The Random Forest demonstrated strong performance again, with a mean accuracy of 91.82% and a low standard deviation of 0.0085, indicating minimal variance across folds. The Decision Tree showed a slightly lower mean accuracy of 88.12%, with a similar standard deviation of 0.0100. The SVM model, while still reasonably consistent, had the lowest cross-validated accuracy at 84.06% and the highest standard deviation among the three, with 0.0107, reflecting slightly less stability across the data sets.

In the next phase, hyperparameter tuning was applied to all three models to improve their predictive performance. The optimisation process led to slight improvements across all models. The Optimised Random Forest model achieved an improved accuracy of 92.52%, with precision and recall increasing to 91.61% and 93.61% respectively. The F1-score rose accordingly to 92.60%. The Support Vector Machine also showed performance improvement, with accuracy increasing to 90.03% and F1-score reaching 90.07%. The Decision Tree model demonstrated a slight improvement, with an accuracy of 88.88% and an F1-score of 88.93%. These results confirmed that the models benefited from hyperparameter tuning, particularly Random Forest and SVM, which remained the two strongest classifiers.

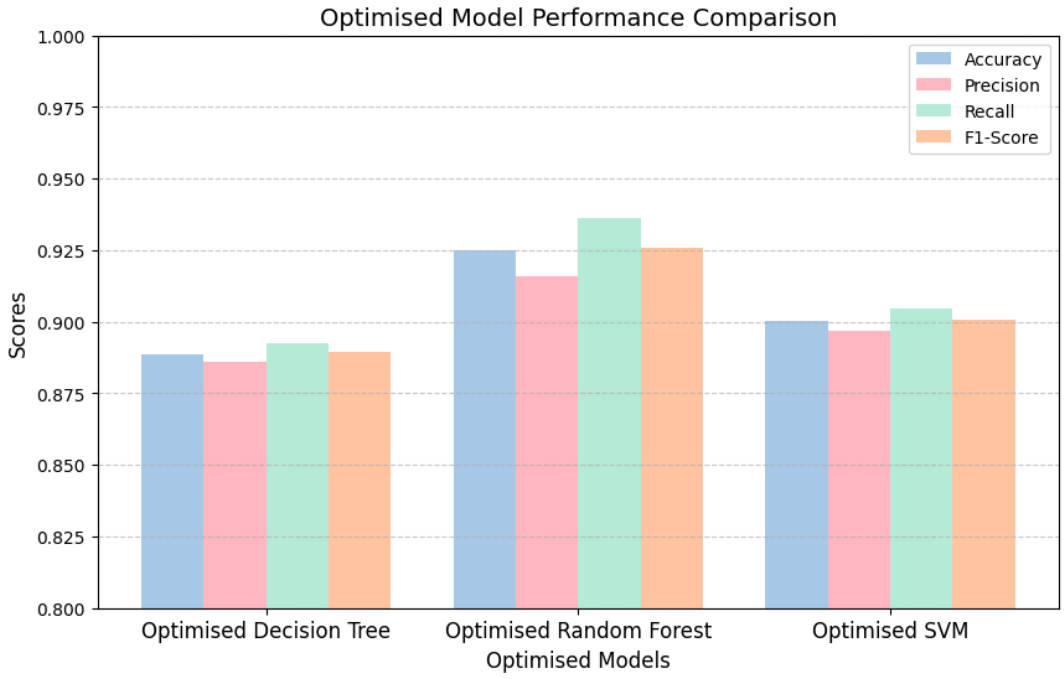

Figure 11 – Bar Graph to Visualise and Compare Optimised Model Performance

As shown in Figure 11, the performance comparison of the optimised models demonstrates that hyperparameter tuning led to noticeable improvements across all models, with Random Forest still performing the best, particularly in recall and F1-score. This improvement in model performance after optimisation highlights the effectiveness of targeted hyperparameter tuning in refining model decision boundaries and enhancing phishing URL detection accuracy.

Based on the performance of the individual models, a hybrid model was constructed using a stacked ensemble method and it resulted in the highest classification performance of all the models tested. It achieved an accuracy of 96.76%, precision of 96.60%, recall of 96.94%, and an F1-score of 96.77%. Cross-validation confirmed the model's effectiveness, with a mean accuracy of 98.91% and a very low standard deviation of 0.0039, suggesting excellent consistency across different data splits. The improvement in both performance and stability over the individual models highlights both the benefit of combining the strengths of Random Forest and SVM through stacked generalisation, and hybrid models themselves.

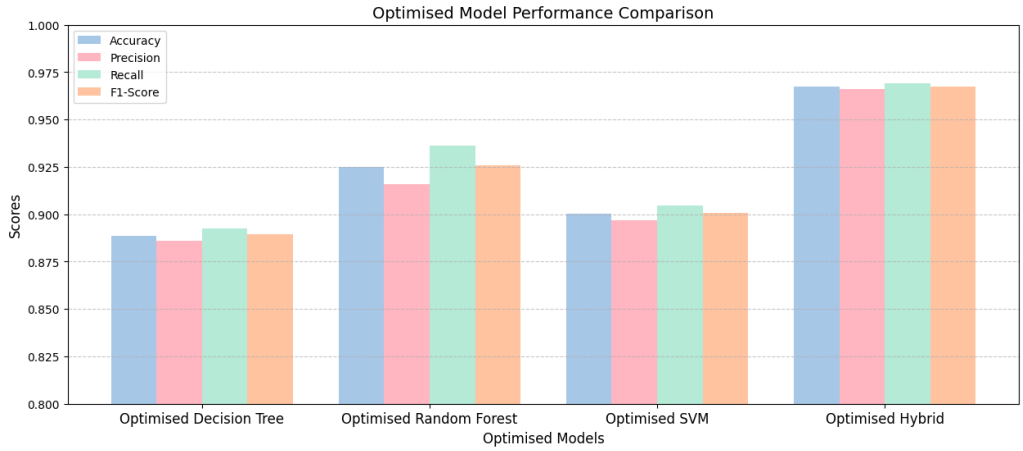

Figure 12 – Bar Graph to Visualise and Compare Performance Amongst All Models

Figure 12 shows the comparative performance of the optimised individual models alongside the hybrid model, which clearly shows that the hybrid model significantly outperformed all the other models across every evaluation metric. This significant performance gain reinforces the advantage of combining multiple base models through stacking, leveraging the complementary strengths of Random Forest and SVM to achieve a higher level of predictive accuracy and robustness in phishing URL detection.

In addition to classification performance, the models were evaluated in terms of real-time inference efficiency, which is a critical requirement for phishing URL detection systems deployed in real-world enterprise environments (Kapoor, 2024). The Decision Tree was the fastest, with an average inference time of 3.453 ms. The Hybrid Model followed closely at 5.174 ms, despite involving multiple layers of computation. Random Forest was slower with 93.796 ms, due to the ensemble of trees, while the SVM was the slowest, requiring 396.777 ms for predictions. These results demonstrated that the hybrid model not only resulted in superior accuracy in comparison to the individually trained models but did so with almost immediate response times which makes it highly suitable for real-world deployment in environments where speed is essential to mitigate possible phishing threats.

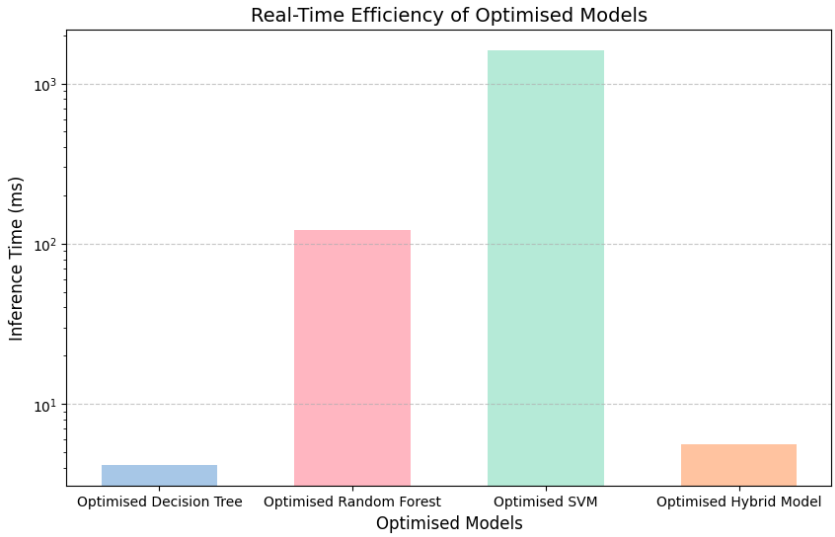

Figure 13 – Bar Graph to Visualise and Compare Inference Speed Among All Models

Figure 13 shows the real-time inference efficiency of the optimised models, highlighting that the Decision Tree and Hybrid models achieved the fastest prediction times, both remaining under 10 milliseconds. In contrast, the Random Forest and SVM models presented significantly longer inference times, particularly the SVM, which required over one second to make predictions. These results highlight the hybrid model's high predictive performance while maintaining near-instantaneous response times, making it highly suitable for real-world deployment in time-sensitive environments such as corporate cybersecurity systems.

Overall, the testing and results phase demonstrated a clear progression in model performance, from baseline benchmarks to a highly effective and efficient hybrid model. The results validate the design choices made throughout the implementation and highlight the potential of ensemble learning techniques for real-time, high-accuracy phishing URL detection.

6. Evaluation

6.1 Comparison to Existing Work Within Research Field

To effectively evaluate the performance of this project, the performance metrics achieved by the models were compared to similar works addressing the problem of phishing URL detection. Initially, the high accuracy scores achieved, particularly by the optimised Random Forest and the hybrid model, raised concerns regarding the potential for overfitting. Models that consistently produce near-perfect results can sometimes be too finely tuned to a specific dataset, limiting their generalisability (Bu and Zhang, 2020). To investigate this, the researcher consulted with an academic supervisor and reviewed recent literature and public implementations of similar phishing URL detection systems. Through this comparative analysis, it was found that many peer-reviewed studies and Kaggle-based solutions have reported similarly high levels of predictive performance, often exceeding 90% accuracy (Islam, 2024), which is similar to the results obtained within this project. However, the hybrid model seems to outperform the results found in existing solutions, which validates this project's contribution to the field.

One contributing factor to these consistently high scores across the field is the nature of phishing URLs themselves. Malicious URLs often exhibit distinct lexical and structural patterns, such as excessive use of special characters, suspicious domain names, or unusually long query strings. These patterns are significantly different from those found in legitimate URLs, making it easier for machine learning algorithms to separate the two classes effectively (Burbela,2023). As a result, the strong performance of the models is well-supported, and the low variance observed during cross-validation further reinforces the conclusion that the models are generalising well and not simply memorising the data.

6.2 Semi-Structured Interview

The performance evaluation of the phishing URL detection system demonstrated excellent results in terms of classification accuracy and precision. However, given the practical motivation behind this project, which stemmed from the researcher's professional experience within a cybersecurity team at a large Oil & Gas company, it was important to assess the system not only through technical metrics, but also in the context of its real-world applicability. To support this, a semi-structured interview was conducted with the researcher's manager, a Senior Cyber Security Manager, whose insights offered valuable operational perspective on the viability, deployment, and potential future development of the system within an enterprise environment. As part of his role involves overseeing the management of a large-scale Security Operations Centre (SOC) and issuing company-wide training to all end-users through the of simulated Phishing tests, he is ideally suited to discuss the impact a tool like this would have within an operational context.

A key point that emerged during this conversation was the importance of aligning the proposed detection solution with existing workflows in the SOC. Currently, phishing alert triage within the company is a largely manual process, where alerts are generated by users clicking the "Report Phish" button in Outlook, which is then queued for analysts to review. This is both time-consuming and resource-intensive, often resulting in delays in incident response. The researcher's manager identified a significant opportunity for this model to be integrated into that workflow by automating the initial triage phase and delivering near-

instant classification of reported emails, based on the initial click of the ReportPhish button. An integration like this would improve SOC operations, reduce analyst workload, and improve the time to detect and respond to phishing threats.

Additionally, a second option was proposed by the researcher which involved the development of a browser-based tool embedded within Microsoft Outlook or Edge, as part of potential future works. This would allow users to manually input suspicious URLs they have come across and receive immediate feedback on their legitimacy. While this approach aligns well with user engagement strategies and complements the existing phishing awareness training within the company, the manager highlighted that the majority of users are unlikely to independently and proactively take such steps, unless they come from a technical background. Therefore, automating this process is more likely to lead to better results.

From an operational perspective, security teams prioritise more than just accuracy and F1-score, with detection speed, explainability, and ease of integration being equally important. During the evaluation discussion, the researcher's manager noted that although the hybrid model was not the fastest, its high accuracy of 96% and acceptable inference time made it the most promising candidate for integration into the organisation's SOC workflow. The balance between performance and efficiency was seen as particularly valuable for real-time triaging of phishing emails, where swift decision-making is essential. However, the manager also stressed the importance of model explainability, as analysts require clear insights into why a URL is classified as malicious or benign to support auditing, validate decisions, and refine detection strategies. These operational needs validate the design of the hybrid model, suggesting that with the addition of an explainability component, it could meet the performance and accountability standards expected in a modern enterprise SOC, and not be treated as a simple black box (Rahman *et al.*, 2024).

6.3 Evaluation of Task Undertaken

Finally, when evaluating the extent to which this project answered the original research questions outlined in the project scope, it is clear that the initial goals have been achieved through the design, implementation, and analysis of multiple machine learning models. However, further reflection under the guiding influence of the researcher's manager and project supervisor, revealed opportunities for a much bigger project. While the current project provides effective classification and real-time efficiency and answers the question of the effectiveness of a hybrid model, there are several advanced components that could have enhanced the project's practical application, such as real-time deployment mechanisms, integrated explainability, and the use of deep learning methods for improved feature classification. These discussions highlighted both the complexity of this field and the problem itself, leading to a deeper understanding of what is required of an effective cyber threat prevention tool.

7. Legal, Social, Ethical, and Professional Issues

Developing a machine learning system for phishing URL detection raised several legal, social, ethical, and professional considerations. From a legal perspective, the dataset was sourced publicly from Kaggle and used in accordance with its open license, ensuring responsible data usage. No personal or sensitive data was involved, which helped to mitigate concerns around GDPR (*General Data Protection Regulation (GDPR) compliance guidelines*, 2018) and the Data Protection Act (*Data Protection Act*, 2018). However, future real-world deployments would require continued compliance, particularly if the system were to process live organisational data.

Ethically, the project aims to enhance corporate cybersecurity by proactively identifying phishing threats, offering clear societal benefits in reducing fraud and financial theft. However, careful testing is necessary to minimise false positives that could inadvertently block legitimate resources and disrupt business operations. Transparency and explainability in model decisions are also critical, especially in professional environments where security teams must justify automated actions. Throughout the project, professional standards such as the BCS Code of Conduct (BCS Code of Conduct, no date) and ACM Code of Ethics and Professional Conduct (ACM Code of Ethics and Professional Conduct, 2018) were considered to ensure accountability and responsible system development.

Socially, the increasing reliance on automated decision-making in cybersecurity raises concerns around trust. Systems like this must support, rather than replace, human analysts to maintain balanced and effective threat detection within organisations.

8. Conclusion and Future Work

In conclusion, this project aimed to investigate the effectiveness of machine learning techniques, specifically with the implementation of a hybrid model, for accurate phishing URLs detection.

Two key research questions guided the work:
1. How will a hybrid machine learning model perform in terms of real-time efficiency when compared to individually trained models?
2. Which machine learning models will have the highest accuracy in predicting the authenticity of a phishing URL?

Both questions were successfully addressed. Three baseline models—Decision Tree, Random Forest, and SVM—were implemented and evaluated using accuracy, precision, recall, F1-score, and inference time. Among them, Random Forest consistently achieved the best predictive accuracy, followed by SVM. The Decision Tree, while fastest in inference time, performed weakest overall.

Developing the hybrid model helped answer the first research question. Although unfamiliar to the researcher initially, a structured testing phase led to the successful development of a hybrid model that achieved over 96% accuracy and inference speeds of approximately 5ms, confirming its potential for real-time deployment. These results support the value of hybrid approaches in combining the strengths of multiple classifiers.

In terms of lessons learned, this project provided insight into the end-to-end pipeline of designing, training, validating, and evaluating machine learning models for a cybersecurity application. It highlighted the importance of thorough validation methods to prevent overfitting, and demonstrated how domain-specific characteristics, such as the syntactic patterns of phishing URLs, can significantly aid in classification performance.

While the project met its defined goals, there were additional avenues that, due to time constraints, could not be fully explored. Future work could involve experimenting with deep learning models such as Recurrent Neural Networks (RNNs), which are particularly effective at handling sequential data like URLs. Further testing on alternative or real-world datasets would also be valuable to assess how well the model generalises beyond the current data. Additionally, there is strong potential for integrating this detection system into real-world Security Operations Centre (SOC) workflows. Enhancements such as real-time deployment, model explainability, and integration with analyst dashboards could increase the practical value of the system and provide security teams with clearer, data-focused insights to act upon.

In conclusion, this project demonstrated that hybrid machine learning models can more effectively and efficiently determine the authenticity of phishing URLs, when compared to individually trained models. The research not only answered its initial research questions but also laid the foundation for future development and real-world applications.

9. Acknowledgements

I would first like to express my sincere gratitude to my supervisor, Dr. Mark Zarb, whose unwavering optimism throughout this year helped me maintain perspective and stay motivated even during the most challenging periods. His support and encouragement is unlike any other. I would also like to thank my supervisor group, The Zarbers, for creating such a supportive and motivating environment.

Most of all, I am deeply grateful to my family and friends. Their unwavering belief in me, and the solidarity and support they showed me throughout the tumultuous journey that has been my final year, has been immeasurable. I could not have made it through this year without them.

10. References

Aastha, V. and Charu, S. (2022) 'Cyber security: A review of cyber crimes, security challenges and measures to control', *Vision The Journal of Business Perspective* [Preprint]. Available at: https://doi.org/10.1177/09722629221074760.

ACM Code of Ethics and Professional Conduct (2018) *Acm.org*. Available at: https://www.acm.org/code-of-ethics (Accessed: 29 April 2025).

Ahammad, S.K.H. *et al.* (2022) 'Phishing URL detection using machine learning methods', *Advances in engineering software (Barking, London, England: 1992)*, 173(103288), p. 103288. Available at: https://doi.org/10.1016/j.advengsoft.2022.103288.

Ahmed, S.F. *et al.* (2023) 'Deep learning modelling techniques: current progress, applications, advantages, and challenges', *Artificial intelligence review*, 56(11), pp. 13521–13617. Available at: https://doi.org/10.1007/s10462-023-10466-8.

Ali *et al.* (2024) 'Phishing—A cyber fraud: The types, implications and governance', *International journal of educational reform*, 33(1), pp. 101–121. Available at: https://doi.org/10.1177/10567879221082966.

Alkhalil, Z. *et al.* (2021) 'Phishing attacks: A recent comprehensive study and a new anatomy', *Frontiers in Computer Science*, 3. Available at: https://doi.org/10.3389/fcomp.2021.563060.

Alshingiti, Z. *et al.* (2023) 'A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN', *Electronics*, 12(1), p. 232. Available at: https://doi.org/10.3390/electronics12010232.

Altwaijry, N. *et al.* (2024) 'Advancing phishing email detection: A comparative study of deep learning models', *Sensors (Basel, Switzerland)*, 24(7), p. 2077. Available at: https://doi.org/10.3390/s24072077.

Ansari, A. (2017) *Classifying data using Support Vector Machines(SVMs) in Python*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/ (Accessed: 29 April 2025).

Arachchilage, N.A.G. and Love, S. (2014) 'Security awareness of computer users: A phishing threat avoidance perspective', *Computers in human behavior*, 38, pp. 304–312. Available at: https://doi.org/10.1016/j.chb.2014.05.046.

Arapi, K. (2018) 'The healthcare industry: Evolving cyber threats and risks', *The healthcare industry: Evolving cyber threats and risks* [Preprint]. Available at: https://www.proquest.com/openview/6fb8d8f9984e83b682b5499fb1d36194/1?cbl=18750&pq-origsite=gscholar&parentSessionId=Tjzz7YVRy8ZKjMlWX0vay3AdTyvW35JPcEuUc6mRVY8%3D.

Badrinarayan, M. (2024) *Decision trees: Split Methods & Hyperparameter Tuning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2024/03/decision-trees-split-methods-hyperparameter-tuning/ (Accessed: 29 April 2025).

BCS Code of Conduct (no date) *Bcs.org*. Available at: https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/ (Accessed: 29 April 2025).

Biau, G. (ed.) (2012) *Analysis of a random forests model*. Journal of Machine Learning Research 13 (2012) 1063-1095. Available at: https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf (Accessed: 30 October 2024).

Boese, R.F., IV (ed.) (2020) *PCI DSS Compliance Challenges for Small Businesses*. ProQuest. Available at: https://www.proquest.com/openview/87f36c48c6a22f922bbeccec9069ca12/1?cbl=18750&diss=y&pq-origsite=gscholar.

Bose, I. and Leung, A.C.M. (2014) 'Do phishing alerts impact global corporations? A firm value analysis', *Decision support systems*, 64, pp. 67–78. Available at: https://doi.org/10.1016/j.dss.2014.04.006.

Bu, C. and Zhang, Z. (2020) 'Research on overfitting problem and correction in machine learning', *Journal of physics. Conference series*, 1693(1), p. 012100. Available at: https://doi.org/10.1088/1742-6596/1693/1/012100.

Burbela, K. (no date) *Model of detection of phishing URLs based on machine learning*, *Diva-portal.org*. Available at: https://www.diva-portal.org/smash/get/diva2:1773760/FULLTEXT02#:~:text=Machine%20learning%20algorithms%3A%20Machine%20learning,URLs%20and%20known%20phishing%20websites. (Accessed: 29 April 2025).

Cervantes, J. *et al.* (2020) 'A comprehensive survey on support vector machine classification: Applications, challenges and trends', *Neurocomputing*, 408, pp. 189–215. Available at: https://doi.org/10.1016/j.neucom.2019.10.118.

Chen, R.-C. *et al.* (2020) 'Selecting critical features for data classification based on machine learning methods', *Journal of big data*, 7(1). Available at: https://doi.org/10.1186/s40537-020-00327-4.

Chy and Hasan, M.K. (2024) 'Securing the web: Machine learning's role in predicting and preventing phishing attacks', *International Journal of Science and Research Archive*, 13(1), pp. 1004–1011. Available at: https://doi.org/10.30574/ijsra.2024.13.1.1770.

Code, M.A. (2024) *Mastering PyTorch inference time measurement - Mark Ai code*, *Medium*. Available at: https://medium.com/@MarkAiCode/mastering-pytorch-inference-time-measurement-22da0eaebab7 (Accessed: 29 April 2025).

Data Protection Act 2018 (2018) *Data Protection Act 2018*. Available at: https://www.legislation.gov.uk/ukpga/2018/12/contents (Accessed: 29 April 2025).

Divakaran, D.M. and Oest, A. (eds) (2022) *Phishing Detection Leveraging Machine Learning and Deep Learning: A Review*. IEEE. Available at: https://doi.org/10.1109/MSEC.2022.3175225.

Do, N.Q. *et al.* (2022) 'Deep learning for phishing detection: Taxonomy, current challenges and future directions', *IEEE access: practical innovations, open solutions*, 10, pp. 36429–36463. Available at: https://doi.org/10.1109/access.2022.3151903.

Dominguez, D. (2025) *Understanding inference-time compute - Daniel Dominguez*, *Medium*. Available at: https://dominguezdaniel.medium.com/understanding-inference-time-compute-c6c6ca2e17a8 (Accessed: 28 April 2025).

Federal Bureau of Investigation (2024) *FBI Releases Internet Crime Report*, *Fbi.gov*. Available at: https://www.fbi.gov/contact-us/field-offices/sanfrancisco/news/fbi-releases-internet-crime-report (Accessed: 29 October 2024).

Frank, E. (2024) 'Navigating cyber threats: Mitigating fraud risks in modern business operations', *Navigating cyber threats: Mitigating fraud risks in modern business operations* [Preprint]. Available at: https://easychair.org/publications/preprint/JBrJ/open (Accessed: 29 October 2024).

General Data Protection Regulation (GDPR) compliance guidelines (2018) *GDPR.eu*. Available at: https://gdpr.eu/ (Accessed: 28 April 2025).

Gholami, R. and Fakhari, N. (eds) (2017) *Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications*. ScienceDirect. Available at: https://www.sciencedirect.com/science/article/pii/B9780128113189000272 (Accessed: 30 October 2024).

Gupta, B.B. *et al.* (2021) 'A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment', *Computer communications*, 175, pp. 47–57. Available at: https://doi.org/10.1016/j.comcom.2021.04.023.

Hillier, W. (2020) 'What's the best language for machine learning?', *CareerFoundry*, 15 December. Available at: https://careerfoundry.com/en/blog/data-analytics/best-machine-learning-languages/ (Accessed: 28 April 2025).

Information Commissioner's Office (2024) 'Phishing'. Available at: https://ico.org.uk/about-the-ico/research-reports-impact-and-evaluation/research-and-reports/learning-from-the-mistakes-of-others-a-retrospective-review/phishing/ (Accessed: 19 October 2024).

intersoft consulting (no date) *Art. 33 GDPR – Notification of a personal data breach to the supervisory authority - General Data Protection Regulation (GDPR)*, *intersoft consulting*. Available at: https://gdpr-info.eu/art-33-gdpr/ (Accessed: 29 October 2024).

Islam, A. (2024) *Phishing URL Detection 96% Accuracy*, *Kaggle.com*. Available at: https://www.kaggle.com/code/ahmedislam0/phishing-url-detection-96-accuracy (Accessed: 29 April 2025).

IT Governance Ltd (no date) *GDPR penalties & fines*, *Itgovernance.co.uk*. Available at: https://www.itgovernance.co.uk/dpa-and-gdpr-penalties (Accessed: 29 October 2024).

Kaabar, S. (2024) *Combining two machine learning models for enhanced predictions*, *All About Trading!* Available at: https://abouttrading.substack.com/p/combining-two-machine-learning-models-e30 (Accessed: 29 April 2025).

Kapoor, M. (2024) 'Comparative analysis of AI algorithms for enhancing phishing detection in real-time email security', *Aitoz Multidisciplinary Review*, 3(1), pp. 338–352. Available at: https://aitozresearch.com/index.php/amr/article/view/97 (Accessed: 29 April 2025).

Karim, A. *et al.* (2023) 'Phishing Detection System Through Hybrid Machine Learning Based on URL', *Phishing Detection System Through Hybrid Machine Learning Based on URL*, p. 18. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10058201&tag=1 (Accessed: 24 October 2024).

Kavlakoglu, E. and Russi, E. (2025) 'What is XGBoost?', *Ibm.com*, 16 April. Available at: https://www.ibm.com/think/topics/xgboost (Accessed: 29 April 2025).

Kayode-Ajala, O. (ed.) (2022) *Applying Machine Learning Algorithms for Detecting Phishing Websites: Applications of SVM, KNN, Decision Trees, and Random Forests*. International Journal of Information and Cybersecurity. Available at: https://publications.dlpress.org/index.php/ijic/article/view/41.

Kharwal, A. (2024) *Hybrid Machine Learning Model with Python*, *thecleverprogrammer*. Aman Kharwal. Available at: https://thecleverprogrammer.com/2024/11/04/hybrid-machine-learning-model-with-python/ (Accessed: 29 April 2025).

Khonji, M., Iraqi, Y. and Jones, A. (2013) 'Phishing detection: A literature survey', *IEEE Communications Surveys & Tutorials*, 15(4), pp. 2091–2121. Available at: https://doi.org/10.1109/surv.2013.032213.00009.

Koehrsen, W. (2018) *Hyperparameter tuning the random forest in python - TDS archive - medium*, *TDS Archive*. Available at: https://medium.com/data-science/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 (Accessed: 29 April 2025).

Mahajan, R. and Siddavatam, I.A. (2018) 'Phishing website detection using machine learning algorithms', *International journal of computer applications* [Preprint]. Available at: https://doi.org/10.5120/IJCA2018918026.

Mohaiminul, I., Guorong, C. and Shangzhu, J. (eds) (2019) *An overview of neural network*. American Journal of Neural Networks and Applications. Available at: https://doi.org/10.11648/j.aajnnaa.20190501.12.

Mulani, S. (2022) *Using StandardScaler() function to standardize python data*, *DigitalOcean*. DigitalOcean. Available at: https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python (Accessed: 29 April 2025).

Nadeem, {muhammad *et al.* (2023) *Phishing Attack, Its Detections and Prevention Techniques*, *Researchgate.net*. Available at: https://www.researchgate.net/publication/374848676_Phishing_Attack_Its_Detections_and_Prevention_Techniques#fullTextFileContent (Accessed: 23 October 2024).

Nahai, F. (2019) 'General data protection regulation (GDPR) and data breaches: What you should know', *Aesthetic surgery journal*, 39(2), pp. 238–240. Available at: https://doi.org/10.1093/asj/sjy296.

Naidu, G., Zuva, T. and Sibanda, E.M. (2023) 'A review of evaluation metrics in machine learning algorithms', in *Lecture Notes in Networks and Systems*. Cham: Springer International Publishing, pp. 15–25.

Omari, K. (2023) 'Comparative study of machine learning algorithms for phishing website detection', *International journal of advanced computer science and applications : IJACSA* [Preprint]. Available at: https://doi.org/10.14569/ijacsa.2023.0140945.

Perera, S. *et al.* (2022) 'Factors affecting reputational damage to organisations due to cyberattacks', *Informatics (MDPI)*, 9(1), p. 28. Available at: https://doi.org/10.3390/informatics9010028.

Pisner, D.A. and Schnyer, D.M. (eds) (2020) *Support vector machine*. ScienceDirect. Available at: https://www.sciencedirect.com/science/article/pii/B9780128157398000067 (Accessed: 30 October 2024).

Pramod, O. (2023) *Decision Trees*, *Medium*. Available at: https://medium.com/@ompramod9921/decision-trees-8e2391f93fa7 (Accessed: 29 April 2025).

Ragucci, J.W. and Robila, S.A. (eds) (2006) *Societal Aspects of Phishing*. IEEE. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4375893 (Accessed: 29 October 2024).

Rahman, Mashfiquer *et al.* (2024) 'The Role of Explainable AI in cyber threat intelligence: Enhancing transparency and trust in security systems', *World Journal of Advanced Research and Reviews*, 23(2), pp. 2897–2907. Available at: https://doi.org/10.30574/wjarr.2024.23.2.2404.

Rigatti, S.J. (ed.) (2017) *Random Forest*. JOURNAL OF INSURANCE MEDICINE. Available at: https://meridian.allenpress.com/jim/article/47/1/31/131479/Random-Forest (Accessed: 30 October 2024).

Saha, S. (2018) *Understanding the log loss function of XGBoost*, *DataDrivenInvestor*. Available at: https://medium.datadriveninvestor.com/understanding-the-log-loss-function-of-xgboost-8842e99d975d (Accessed: 29 April 2025).

Sankhwar, S., Pandey, D. and Khan, R.A. (eds) (2019) *View of email phishing: An enhanced classification model to detect malicious URLs*. Available at: https://doi.org/10.4108/eai.13-7-2018.158529.

Sapphire (2023) 'What does phishing mean? Understanding the basics of online security', *wordpress-331244-3913986.cloudwaysapps.com*. Sapphire.net, 18 October. Available at: https://www.sapphire.net/blogs-press-releases/what-phishing-mean/ (Accessed: 23 October 2024).

Shafi, A. (2024) *Random Forest Classification with Scikit-Learn*, *Datacamp*. Available at: https://www.datacamp.com/tutorial/random-forests-classifier-python?dc_referrer=https%3A%2F%2Fwww.google.com%2F (Accessed: 29 April 2025).

Shah, R. (2021) *Tune hyperparameters with GridSearchCV*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/ (Accessed: 29 April 2025).

Siddiq, A.A., Mohammad, A. and Islam, M.S. (2022) 'Phishing Website Detection using Deep Learning', in *Proceedings of the 2nd International Conference on Computing Advancements*. New York, NY, USA: ACM, pp. 83–88.

Sopna, S.S. and Ahmad, Z.Z. (2021) 'Online identity theft, security issues, and reputational damage', *Preprints*. Available at: https://doi.org/10.20944/preprints202102.0082.v1.

StratifiedKFold (no date) *scikit-learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (Accessed: 29 April 2025).

Suthaharan, S. (2016) 'Decision tree learning', in *Integrated Series in Information Systems*. Boston, MA: Springer US, pp. 237–269. Available at: https://doi.org/10.1007/978-1-4899-7641-3_10.

Taeho, J. (2021) 'Decision tree', in *Machine Learning Foundations*. Cham: Springer International Publishing, pp. 141–165. Available at: https://doi.org/10.1007/978-3-030-65900-4_7.

Tejveer, S., Manoj, K. and Santosh, K. (2024) 'Walkthrough phishing detection techniques', *Computers & electrical engineering: an international journal*, 118(109374), p. 109374. Available at: https://doi.org/10.1016/j.compeleceng.2024.109374.

Tiwari, S. (2021) *Web page Phishing Detection Dataset*, *Kaggle.com*. Available at: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset?resource=download (Accessed: 26 March 2025).

Tuychiev, B. (2023) *Using XGBoost in Python Tutorial*, *Datacamp*. Available at: https://www.datacamp.com/tutorial/xgboost-in-python (Accessed: 29 April 2025).

vaibhavkumar (2023) *RBF SVM parameters in scikit learn*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/rbf-svm-parameters-in-scikit-learn/ (Accessed: 29 April 2025).

Van Den Reym, M. (2020) *7 advantages of using Google colab for Python*, *Python in Plain English*. Available at: https://python.plainenglish.io/7-advantages-of-using-google-colab-for-python-82ac5166fd4b (Accessed: 28 April 2025).

Verma, R. and Das, A. (2017) 'What's in a URL: Fast feature extraction and malicious URL detection', in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. New York, NY, USA: ACM, pp. 55–63.

Wong, K.J. (2023) *Feature Encoding Techniques in Machine Learning with Python Implementation*, *towardsdatascience*. Available at: https://towardsdatascience.com/feature-encoding-techniques-in-machine-learning-with-python-implementation-dbf933e64aa/ (Accessed: 29 April 2025).

Yoganandham, G. (2024) *Researchgate.net*. Available at: https://www.researchgate.net/profile/Yoganandham-Govindharaj-2/publication/384246916_THE_ECONOMIC_IMPACT_OF_PHISHING_VISHING_ONLINE_MARKETPLACES_AND_EMERGING_CYBERCRIMES_EXPOSING_THE_CYBERCRIME_ECONOMY_AND_SOCIAL_COSTS_IN_THE_MODERN_ERA_OF_DIGITAL_FRAUD_-AN_ASSESSMENT/links/66f0fd50fc6cc46489704a89/THE-ECONOMIC-IMPACT-OF-PHISHING-VISHING-ONLINE-MARKETPLACES-AND-EMERGING-CYBERCRIMES-EXPOSING-THE-CYBERCRIME-ECONOMY-AND-SOCIAL-COSTS-IN-THE-MODERN-ERA-OF-DIGITAL-FRAUD-AN-ASSESSMENT.pdf (Accessed: 29 October 2024).

11. Appendices

11.1 Project Log –

23/09/2024 – Having to rethink my proposed project idea. I can't seem to find datasets that suit my initial plan of analysing character traits, amongst key demographics, of those who fall victim to a Phishing scam, and I don't think a dataset like that will be easy to find either. Also, not sure of the practical applications of this idea so pivoting slightly to phishing URL detection. This still falls under the initial umbrella of what I wanted to do because it's still inspired by my placement, but this seems more feasible as I have already found a relevant dataset that categorises the key features of a phishing URL which I can then classify with "malicious" or "legitimate". Links in really nicely with the work done at my placement and there are a lot of practical applications to this too when considering future development.

03/10/2024 – Submitted Ethics Form and Project Proposal.

04/10/2024 – Started reviewing the literature around the topic of Phishing URL detection. Aiming to start quite broad and then narrow down the research as per the advice given by Mark.

10/10/2024 – Supervisor Meeting. Received feedback on the project proposal's we submitted at the start of the month – Mark says that mine is all good to go and that I should start building upon what I proposed. Asked Mark about the "Requirements Analysis" section but was told that I didn't need it for the Project Scope – it'll come later in the Design and Methodology section of the second half of the report. Will continue working on writing the project scope.

22/10/2024 – Asked for feedback on what it means to "critically review" a paper – I have 20 papers to support my project scope so far, but not sure how in-depth the "critical review" for each paper has to be. Feedback given was that Mark cares more about the "coherent narrative" of the project scope, backed with evidence throughout, so not every paper has to be reviewed in depth, some can just be supporting references for stats and figures.

24/10/2024 – Supervisor Meeting with Haribos. Feedback for our initial draft project scopes were given, will spend some time working on the advice given, but luckily nothing major was pointed out so far. Also spent a large chunk of the day to find supporting evidence that I foolishly put placeholders for. Will not be making that mistake again. Found out our second markers, I have Mark Bartlett. Continue working on project scope for the rest of the week.

27/10/2024 – Continuation of Project Scope – refining sections and adding details. Clarified with Mark that I can reuse the same reference multiple times but what he doesn't like to see is if two pages rely solely on the same reference to support the text. Additionally clarified what is meant by the "future implementation plans" – it's just referring to the research questions so will make sure to clearly define what the project aims to achieve. Will use two research questions, max, so that my project/dissertation has a clear focus and I don't get too ambitious and under-deliver.

28/10/2024 – Submitted final draft of the project scope to Mark for feedback. It's not completely finished, still need to add references and tidy it up, etc.

30/10/2024 – Submitted finished Project Scope.

01/11/2024 to 17/01/2025 – Inactivity period due to other coursework and holidays. Had a meeting on the 7th of November to close out the semester and discuss next steps but main priority for the remainder of the semester is the other modules. Received feedback on Project Scopes on the 26th of November – got an A so very happy with that.

01/02/2025 – Started researching best machine learning models to utilise within the project. Revisited my project scope to remind myself of what knowledge I had gathered from my initial research – used that as a stepping stone to narrow down what models would be great to implement within my project, pretty sure I will go with Decision Tree, Random Forest and SVM but would like to solidify my choices before I start the implementation so that I don't have to backtrack on any progress.

2/02/2025 to 14/02/2025 – Brief inactivity period due to some personal issues.

15/02/2025 – Started the implementation within Google Colab. Loaded in the dataset and began some initial exploration of the data.

17/02/2025 – Submitted Degree Show Booklet Form. Continued initial pre-processing of data by extracting relevant URL features and disregarding the rest. Continued with some feature encoding.

01/03/2025 – This week I started the implementation of my baseline models: Decision Tree, Random Forest and SVM and visualised the differences between the performance metrics.

08/03/2025 – This week I continued with the code by optimising, validating and evaluating my baseline models. This was done through Stratified K-Fold validation with 10 splits as I did some research to see which validation model would be the best suited for my project and this was the most popular answer. 10 splits didn't seem to have an effect on the duration of the models run time so continued with this as it makes the models more reliable. I also used GridSearch to find the optimal configurations for each model and this proved to be very effective. Standard evaluation metrics were applied like accuracy, recall, etc. Inference time doesn't come into play yet as I'll use that when comparing my optimised models against the hybrid model. Made sure to visualise the improvements in the models so that it was clear to see the effects of the optimisation and validation and also makes my colab file a bit more visual which is good for the report, poster, demo, etc.

15/03/2025 – This week I focused on implementing my hybrid model. As the Random Forest and SVM were the best performing models, it made sense to carry these two forward. With some research, I found that XGBoost was a good choice for the meta-classifier because it's known for it's high accuracy and ability to generalise well and I can't argue with that. So far, the implementation has been relatively straightforward, but the hybrid model is causing some issues – like I knew it would. I haven't ever built a hybrid model before so

understanding the theory behind the stacked features is taking a little time and I'm encountering a couple of errors, will continue to work on this throughout the next week.

17/03/2025 – Started working on my poster! Clarified a few questions with the Zarbers, but have a layout planned.

18/03/2025 – Continued working on the poster – started adding some text and images and created some figures to demonstrate the methodology pipeline – used the text from the project scope as it basically summed up everything I needed.

22/03/2025 – This week I made progress with the hybrid model, and I managed to get it working and I also created a function to cross-validate the hybrid model at each fold. I then compared the performance of all the optimised models with the hybrid model and the results clearly highlight that an ensemble-approach vastly outperform the other models with over 96% accuracy. Next step is to work on the Inference Speed comparisons.

24/03/2025 – The last bit of the code is the calculation of the inference speed for each of these models and I need this done soon so that I can include the results in my poster as the real-world applicability to this project is basically one of the main things about this whole project. Continued working on this aspect but encountered a couple of errors that I need to figure out how to fix, will continue tomorrow.

25/03/2025 – Finished calculating the inference speed! And I have also visualised this to clearly indicate the comparison between the different models. This basically proved my hypothesis as the hybrid model, whilst not the fastest, it did give almost instantaneous results, and with the added benefit of the highest accuracy score, it just shows that hybrid models are ideally suited to be deployed within an enterprise-level setting.

26/03/2025 – Finalised project code within Google Colab. Needed to make sure this was done as I need some of the graphs for my poster. Continued making final touches to my poster in terms of wording, layout, etc. with feedback from the Zarbers. Will aim to continue adding to the code to improve generalisability by testing with other datasets and calculating a couple of other metrics – this is stemming from my conversations with Mark Zarb and Pam on how to improve my project development.

27/03/2025 – Submitted Final Poster after making a couple of last minute changes to make it more visual by including a figure in the Future Works section, cited the relevant paper.

01/04/2025 – April marks the beginning of the report dissemination (fancy term I learnt from Pam). Progress will likely be slow with this as there are many other deadlines within this month so time must be spread out accordingly.

10/04/2025 – Due to time constraints, I will not be able to continue adding to the complexity of the project with the before mentioned approaches – will discuss this in the Future Works section of the report when I get to that section.

18/04/2025 – Had a call with my manager today to discuss the project, the practical implications of it and potential improvements that can be made to further optimise the project for enterprise-level deployment. Was originally meant to be a quick 5-minute call but it lasted for 45 minutes as he was very interested in this project and the discussion, we had about how it could be incorporated into the existing workflows at work gave me a lot to talk about in the evaluation section which is great. Over the next few days, I will be spending the evenings working on the dissertation report, whilst balancing other deadlines, so that I am ready to submit an almost-finished draft to Mark on Tuesday as it would be greatly beneficial to have as much feedback as possible.

22/04/2025 – Submitted dissertation draft to Mark for feedback. At this point, the report is mostly done – the only sections that are missing is the Introduction, Abstract, Acknowledgments, etc. Still need to format it properly but will do that once I receive feedback.

23/04/2025 – Got feedback from Mark – thankfully it wasn't anything too major! Will work on getting these changes made and the final report completed over the next few days.

28/04/2025 – Nearing the end of the project dissemination. All sections are completed. Now I just need to cut down a little on the word count, push my code to GitHub and find a bunch of references because I evidently did not learn my lesson from the project scope, and I left it to the end again and now I have a bunch of placeholders saying "go find that reference you found before".

29/04/2025 – Tidied report, checked formatting and references, and then submitted to dropbox and report was pushed to GitHub so that the repo contains all the relevant files for this dissertation.

11.2 Source Code -

The full code and utilised dataset can be found through GitHub:
https://github.com/navyashiju/CM300_Phishing-URL-Detection