PHISHING BASED SOCIAL ENGINEERING DETECTION IN AI

NURSYUHADAH BINTI AHMAD SUDERMAN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BORANG PENGESAHAN STATUS LAPORAN

JUDUL: [PHISHING BASED SOCIAL ENGINEERING DETECTION IN AI]

SESI PENGAJIAN: [2024 / 2025]

Saya: ____[NURSYUHADAH BINTI AHMAD SUDERMAN]_____

mengaku membenarkan tesis Projek Sarjana Muda ini disimpan di Perpustakaan Universiti Teknikal Malaysia Melaka dengan syarat-syarat kegunaan seperti berikut:

1. Tesis dan projek adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan unituk tujuan pengajian sahaja.
3. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. * Sila tandakan (✓)

_____ SULIT (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

_____ TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi / badan di mana penyelidikan dijalankan)

____✓____ TIDAK TERHAD

_____   _____
(TANDATANGAN PELAJAR)   (TANDATANGAN PENYELIA)

Alamat tetap: E-7-9 P/A DESA REJANG
PERSIARAN REJANG, TAMAN
SETAPAK JAYA 53300, WILAYAH
PERSEKUTUAN KUALA LUMPUR

Dr. ZAHEERA BINTI ZAINAL ABIDIN
PENSYARAH KANAN
FAKULTI KECERDASAN BUATAN DAN KESELAMATAN SIBER
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DR ZAHEERA ZAINAL ABIDIN

Tarikh: ____08-09-2025_____   Tarikh: _____

CATATAN: * Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak

PHISHING BASED SOCIAL ENGINEERING DETECTION IN AI

NURSYUHADAH BINTI AHMAD SUDERMAN

This report is submitted in partial fulfillment of the requirements for the
Bachelor of [Computer Science (Computer Security)] with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

DECLARATION

I hereby declare that this project report entitled

[PHISHING BASED SOCIAL ENGINEERING DETECTION IN AI]

is written by me and is my own effort and that no part has been plagiarized

without citations.

STUDENT   : NURSYUHADAH BINTI AHMAD SUDERMAN   Date : <u>08-09-2025</u>

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of [Computer Science (Computer Security)] with Honours.

SUPERVISOR   : DR ZAHEERA BINTI ZAINAL ABIDIN   Date :

DEDICATION

This work is dedicated to Allah Almighty, my creator, pillar, and source of wisdom, knowledge, and insight. Throughout this journey, He has been my source of strength, and I have only been able to fly on His wings.

To my beloved parents, Ahmad Suderman Bin Mohd Yaacob and Norayzuazlin Binti Mohd Ayob whom I wholeheartedly dedicate this study to. Thank you for being my source of inspiration and for your unwavering support and trust. To my brothers, whom I hold very dear to my heart.

To my supervisor, Dr Zaheera binti Zainal Abidin, for supporting and believing in me to accomplish this project. Finally, I want to thank my friends for being there for me during my bachelor's degree, sharing words of advice and support to help me accomplish this study.

ACKNOWLEDGEMENTS

First and foremost, First and foremost, I would like to praise and thank Allah the Almighty, the Most Gracious, and the Most Merciful for His blessing given to me in completing this project. Peace and salutations to the Prophet Muhammad S.A.W, who has led all humanity from darkness to light.

I would like to express my heartfelt appreciation to Dr. Zaheera binti Zainal Abidin for her assistance in completing this assignment successfully. Her insightful thoughts have provided various inspirations and enlightenment for this project.

I'd also want to thank my dear parents and sisters for their encouragement and support during this effort. Their help and concern had given me confidence while I worked on this project. Finally, I am grateful to my friends who are ready to share their wisdom and have directly or indirectly guided me during this project.

ABSTRACT

Social engineering attacks pose a significant and growing threat in the cybersecurity landscape, exploiting human psychology rather than technical vulnerabilities to manipulate individuals into compromising sensitive information. Among various social engineering techniques, phishing has emerged as the most prevalent and damaging, driven by its scalability, low execution cost, and evolving tactics such as spear phishing and smishing. As traditional rule-based detection methods struggle to keep up with these dynamic threats, Artificial Intelligence (AI) offers a robust and adaptive solution. This study explores AI-based approaches—specifically Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL)—for the detection of phishing attacks. It provides a comprehensive literature review, analysis of current research trends, and detailed examination of relevant datasets and algorithms including Support Vector Machines (SVM), Long Short-Term Memory (LSTM), and Deep Q-Networks (DQN). The research highlights the advantages of AI in automating threat detection, enhancing accuracy, and enabling systems to adapt to new attack vectors. By focusing on phishing as a primary social engineering threat, this work aims to contribute to the development of intelligent, scalable, and real-time defenses against increasingly sophisticated cyber threats.

ABSTRAK

Serangan kejuruteraan sosial merupakan ancaman yang semakin ketara dalam landskap keselamatan siber, di mana ia mengeksploitasi psikologi manusia dan bukannya kelemahan teknikal untuk memanipulasi individu agar mendedahkan maklumat sensitif. Dalam pelbagai teknik kejuruteraan sosial, serangan *phishing* telah muncul sebagai bentuk yang paling meluas dan merosakkan, didorong oleh keupayaannya untuk berkembang secara besar-besaran, kos pelaksanaan yang rendah, serta taktik yang sentiasa berubah seperti *spear phishing* dan *smishing*. Oleh kerana kaedah pengesanan berasaskan peraturan tradisional semakin sukar menandingi ancaman yang dinamik ini, Kecerdasan Buatan (AI) menawarkan penyelesaian yang kukuh dan adaptif. Kajian ini meneroka pendekatan berasaskan AI—khususnya Pembelajaran Mesin (ML), Pembelajaran Mendalam (DL), dan Pembelajaran Penguatan (RL)—dalam pengesanan serangan *phishing*. Ia merangkumi ulasan literatur yang komprehensif, analisis *trend* penyelidikan semasa, serta pemeriksaan terperinci terhadap set data dan algoritma yang relevan termasuk *Support Vector Machines* (SVM), *Long Short-Term Memory* (LSTM), dan *Deep Q-Networks* (DQN). Kajian ini menekankan kelebihan AI dalam mengautomasikan pengesanan ancaman, meningkatkan ketepatan, dan membolehkan sistem menyesuaikan diri dengan vektor serangan baharu. Dengan memberi tumpuan kepada *phishing* sebagai ancaman utama kejuruteraan sosial, kajian ini bertujuan menyumbang kepada pembangunan sistem pertahanan yang pintar, berskala, dan masa nyata terhadap ancaman siber yang semakin canggih.

TABLE OF CONTENTS

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

LIST OF ABBREVIATIONS

ML	**-**	**Machine Learning**
DL	**-**	**Deep Learning**
RL	**-**	**Reinforcement Learning**
AI	**-**	**Artificial Intelligent**
RF	**-**	**Random Forest**
SVM	**-**	**Support Vector Machines**
LSTM	**-**	**Long Short-Term Memory**
BERT	**-**	**Bidirectional Encoder Representations from Transformers**
QL	**-**	**Q-Learning**
DQL	**-**	**Deep Q-Learning**
DQN	**-**	**Deep Q-Networks**

CHAPTER 1: INTRODUCTION

1.1   Introduction

Social engineering attacks are among the most persistent and evolving cybersecurity threats, leveraging psychological manipulation to deceive individuals into disclosing confidential information or compromising systems. Techniques such as phishing, baiting, and pretexting exploit trust, urgency, or curiosity, making them difficult to detect using traditional, rule-based security systems. Phishing stands out as a dominant threat due to its scalability, ease of execution, and the increasing sophistication of its tactics.

This study explores the role of Artificial Intelligence (AI) in detecting social engineering attacks, with a primary focus on phishing. The research involves a comprehensive analysis of existing AI approaches—specifically Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL)—to evaluate their effectiveness, challenges, and potential in improving detection accuracy. Rather than developing a prototype, this study critically reviews current methods, datasets, and models to identify strengths, limitations, and areas for improvement in AI-based social engineering detection systems.

1.2   Problem Statement

The problem that has been identified is summarized in Table 1-1 below:

Table 1.2-1 Problem Statement

PS	Problem Statement
PS1	Existing AI models struggle to detect the constantly changing nature of social engineering attacks, resulting in lower detection accuracy.

PS1: Existing AI models struggle to detect the constantly changing nature of social engineering attacks, resulting in lower detection accuracy.

Existing AI models face challenges in accurately detecting social engineering attacks due to the dynamic and evolving tactics used by attackers. Machine learning has gradually introduced a huge basket of "AI capabilities" that can be harnessed for social engineering and phishing attacks (Schmitt & Flechais, 2024). This constant change reduces the effectiveness of traditional detection approaches, leading to lower overall detection accuracy.

1.3   Project Question

Three Project Question (PQ) is constructed based on the problem statement that needs to be answered in this project. The summary of project question is shown in Table 1-2.

Table 1.3-1 Project Question

PS	PQ	Project Question
PS1	PQ1	What are the existing phishing techniques for detecting social engineering attacks in an AI environment?
	PQ2	How accurate are existing phishing techniques in detecting social engineering attacks?
	PQ3	How does the accuracy performance of the proposed approach compare with existing phishing techniques in detecting social engineering attacks?

PQ1: What are the existing phishing techniques for detecting social engineering attacks in an AI environment?

This question explores the range of phishing detection methods, including machine learning, deep learning, and reinforcement learning approaches, which are commonly used to identify social engineering attacks in digital communications.

PQ2: How does the accuracy performance of the proposed approach compare with existing phishing techniques in detecting social engineering attacks?

This question focuses on measuring the performance of existing phishing detection approaches, considering metrics such as accuracy, precision, recall, and overall detection rates.

PQ3: How does the accuracy performance of the proposed approach compare with existing phishing techniques in detecting social engineering attacks?

This question compares the proposed method's performance with existing techniques using the same metrics to see if it offers better detection of social engineering attacks.

1.4   Objective

Based on the project questions formulated in the previous section, appropriate project objectives (PO) are developed. The Project Objective (PO) is summarized in Table 1-3.

Table 1.4-1 Project Objective

PS	PQ	PO	Project Objective
PS1	PQ1	PO1	To investigate existing phishing techniques for detecting social engineering attacks in AI environment.
	PQ2	PO2	To analyze accuracy performance of phishing in detecting social engineering attacks.
	PQ3	PO3	To evaluate the performance of proposed approach in detecting social engineering attacks.

PO1: To investigate existing phishing techniques for detecting social engineering attacks in AI environment.

Understanding current phishing detection techniques is essential for identifying the strengths and limitations of existing approaches. This investigation provides a foundation for developing more effective AI-driven methods to counter social engineering attacks.

PO2: To analyze accuracy performance of phishing in detecting social engineering attacks.

Evaluating the accuracy of phishing detection techniques is crucial for assessing their reliability and effectiveness in real-world scenarios. This analysis will highlight the strengths and weaknesses of existing methods.

PO3: To evaluate the performance of proposed approach in detecting social engineering attacks.

Evaluating the proposed method is essential to determine its effectiveness in identifying social engineering attacks. This evaluation will reveal how well it performs compared to existing techniques, highlighting its potential improvements and limitations.

1.5   Project Scope

The project's scopes are focused on the following aspects:

1. The AI tools considered for phishing attack detection are Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL).

2. The performance of the AI model will be evaluated based on accuracy, precision, recall, and F1-score.

3. The platform used for training and testing the AI models is WEKA and KNIME

1.6 Project Contribution

PS	PQ	PO	PC	Project Contribution
PS1	PQ1	PO1	PC1	Comprehensive review of AI techniques (ML, DL, RL) for phishing detection.
	PQ2	PO2	PC2	A summary of performance metrics (accuracy precision, recall, F1-score) in current AI models.
	PQ3	PO3	PC3	Proposed hybrid AI framework to improve detection accuracy beyond baseline models.

1.7 Thesis Organization

This report is divided into six chapters, namely Chapter 1: Introduction, Chapter 2: Literature Review, Chapter 3: Methodology, Chapter 4: Design, Chapter 5: Implementation, Chapter 6: Discussion and Chapter 7: Conclusion.

Chapter 1: Introduction

This chapter discusses the project's introduction, background, research problem, research questions, objectives, scope, significance of the project, and report organization. It sets the stage for the problem addressed and outlines the goals of the project in the context of AI-based social engineering attack detection.

Chapter 2: Literature Review

This chapter provides a review of relevant literature, discussing previous research on phishing attacks, existing detection methods, and AI techniques such as Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL). It also identifies gaps in the current literature and frames the research questions and objectives.

Chapter 3: Methodology

This chapter outlines the research methodology adopted for the project, including the datasets used, AI techniques implemented (ML, DL, RL), and the performance evaluation metrics. It provides a step-by-step explanation of the model development, data preprocessing, feature extraction, and model training processes.

Chapter 4: Analysis and Design

This chapter explains the project's implementation in detail, including how the project was executed and how the results were generated.

Chapter 5: Implementation

This chapter will cover the installation and configuration of the project's environment, detailing the software used and how it was set up for the current stage of implementation.

Chapter 6: Testing

This section focuses on testing and validating the project using the dataset to obtain results. It also presents a comprehensive discussion of the findings and outcomes.

Chapter 7: Conclusion

This chapter includes the introduction, a summary of the project, its contributions, limitations, suggestions for future work, and the conclusion.

1.8 Summary

The problem description, aim, scope of the project, relevance, and projected output of the project are all clearly outlined in this chapter. The description of the related works to this project is detailed later in Chapter 2.

CHAPTER 2: LITERATURE REVIEW

2.1   Introduction

Social engineering is a psychological manipulation technique where cybercriminals exploit human trust, fear, or urgency to deceive individuals into performing actions that compromise security, such as disclosing confidential information or enabling system access. Instead of relying on traditional hacking tools to exploit software or hardware vulnerabilities, social engineering exploits human behavior, making it one of the most dangerous and successful forms of attack in the modern threat landscape. Techniques include phishing, baiting, pretexting, vishing, and quid pro quo.

Among these techniques, phishing remains the most widespread and damaging form of social engineering attack, and its prevalence has been supported by numerous global studies and industry analyses. Research shows that phishing is consistently ranked as the most common social engineering threat due to its effectiveness, ease of execution, and adaptability across communication channels (Alzahrani et al., 2021; Basit et al., 2022). Attackers exploit digital platforms, particularly email, to impersonate legitimate institutions and deceive users into providing sensitive information.

What sets phishing apart from other social engineering methods is its ability to scale rapidly and target both individuals and organizations with minimal resources. Unlike baiting or pretexting, phishing campaigns can be automated and distributed widely using email bots and phishing kits available on underground forums. These features make phishing a recurring top threat in annual cybersecurity threat reports by multiple academic and professional organizations.

The continued evolution of phishing tactics, from generic spam to personalized spear phishing and smishing (SMS phishing), has contributed to its resilience and growth. This explains why, compared to other forms of social engineering attacks, phishing is the focus of this research. It provides not only a well-documented attack

vector but also a diverse set of datasets and practical applications for testing AI-based detection models.

As the complexity and frequency of phishing attacks grow, the need for intelligent, adaptive defense systems becomes critical. Traditional security measures often fall short, especially against zero-day phishing attacks or personalized spear phishing. This has led to a strong shift toward Artificial Intelligence (AI), specifically Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL), as a robust solution for phishing detection. AI-based models can process large volumes of data, identify hidden patterns, and adapt over time, enabling more accurate and timely threat detection.

Given phishing's prominence and evolving nature, this study focuses exclusively on AI-based phishing detection. Compared to other forms of social engineering, phishing offers both a greater volume of research data and clearer use cases for AI implementation, making it an ideal focal point for academic and practical analysis. The subsequent sections provide a deep dive into AI techniques, available software tools, datasets, and recent research that contribute to building effective, intelligent phishing detection systems.
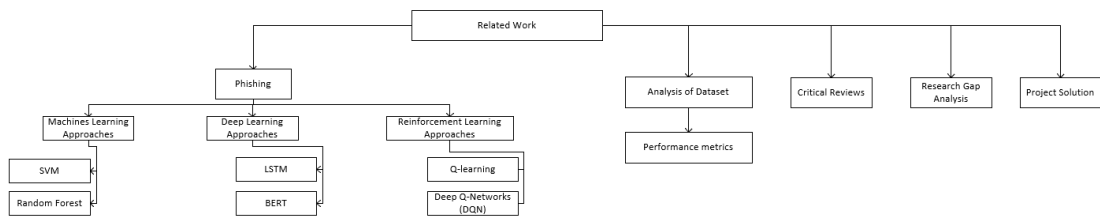
Figure 1: Literature Review Structure Diagram

2.2   Related Work

Recent advancements between 2020 and 2024 have demonstrated the growing impact of AI in combating phishing threats. Researchers have developed increasingly sophisticated models leveraging ML, DL, and RL to classify, detect, and respond to phishing activities in real time.

Rahman et al. (2022) introduced a hybrid model combining Decision Trees and anomaly detection, achieving improved phishing email detection rates in enterprise environments. Their work highlighted the effectiveness of ensemble methods in capturing varied attack patterns. Aljohani and Hossain (2021) employed the BERT transformer model to detect phishing content by analyzing contextual semantics within emails. Their results showed significant accuracy improvements over traditional ML models.

Zhao et al. (2020) used Long Short-Term Memory (LSTM) networks to detect phishing attempts in mobile text messages. LSTM's ability to process sequential information proved critical in identifying suspicious messages that mimic legitimate communications. Nasir et al. (2023) proposed a CNN-based phishing detection system that integrated user behavior logs with content analysis. The combination of behavioral and textual features enhanced detection performance, particularly for spear-phishing cases.

Asker and Essa (2024) emphasized the potential of unsupervised models, such as Autoencoders, to identify phishing emails without labelled data. Their findings suggest these models are well-suited for environments where phishing strategies evolve rapidly, and labelled examples are scarce. Zhang et al. (2023) explored the use of Reinforcement Learning to create adaptive phishing detection systems. Their RL-based framework optimized decision-making policies in real-time, improving resilience against evolving attack vectors while minimizing false positives.

These studies demonstrate a transition from static rule-based systems to adaptive, intelligent models that integrate language understanding, behavior analysis,

and real-time learning. They underscore the importance of hybrid approaches and model interpretability in building effective phishing detection solutions.

2.2.1   Phishing

Phishing is a cyber-attack technique in which malicious actors impersonate trustworthy entities to deceive victims into revealing sensitive information, such as usernames, passwords, credit card details, or login credentials. These attacks often take the form of emails, websites, or messages that closely resemble legitimate communication from known organizations. The purpose of phishing is typically to commit fraud, gain unauthorized access to systems, or install malware.

Phishing continues to be the most prevalent type of social engineering attack globally. Its popularity among attackers stems from its high success rate, scalability, and ability to exploit human behavior rather than technical vulnerabilities. According to the 2023 **Verizon Data Breach Investigations Report**, phishing accounted for more than one-third of all social engineering-related incidents. The Anti-Phishing Working Group (APWG) also reported over 1.2 million phishing attacks in a single quarter, indicating a steady and alarming upward trend. These attacks have evolved from generic bulk emails to more targeted approaches such as spear phishing, business email compromise (BEC), and smishing (SMS phishing), which are increasingly difficult to detect. The figure below shows that Pretexting is now more prevalent than Phishing in Social Engineering incidents. However, when we look at confirmed breaches, Phishing is still on top (Verizon, 2023).
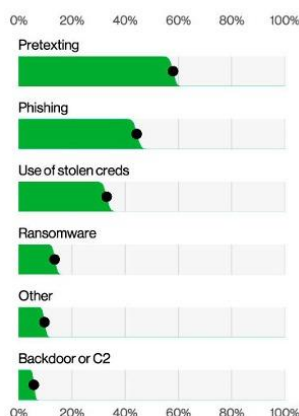
Figure 2: Action varieties in Social Engineering incidents (Verizon, 2023)

Due to the sophistication and frequency of phishing, AI-based techniques have become central to its detection. Machine Learning approaches such as Support Vector Machine (SVM) and Random Forest are commonly used due to their strong classification capabilities on structured datasets. These models analyze features such as email headers, URL characteristics, and content patterns., models like Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) have proven effective in understanding the semantic structure of emails, enabling the identification of sophisticated phishing attempts that evade traditional filters.

Recently, Reinforcement Learning (RL) has emerged as a powerful approach for dynamic and adaptive phishing detection. Unlike ML and DL, which learn from static datasets, RL enables an agent to learn optimal policies through interactions with an environment, receiving feedback in the form of rewards or penalties. Techniques such as Q-learning, Deep Q-Networks (DQN), and Policy Gradient Methods are used to develop adaptive systems that can fine-tune detection thresholds based on evolving user behavior and attack strategies. These RL-based methods are especially useful in real-time systems, where phishing tactics change frequently and require continuous learning to maintain detection accuracy.

2.2.2   Machine Language Approaches

Machine Learning (ML) is a core subset of Artificial Intelligence (AI) that focuses on creating algorithms and models that enable computers to automatically learn and improve from experience without being explicitly programmed for specific tasks. At its essence, ML empowers systems to identify patterns, make predictions, and adapt to new data through iterative learning processes. This adaptability is achieved by training models on vast amounts of labelled or unlabelled data and optimizing performance based on measurable outcomes (GeeksforGeeks, 2023).

Modern ML approaches are categorized into three main types: supervised learning, where models learn from labelled data; unsupervised learning, where models uncover hidden structures in unlabelled data; and reinforcement learning, where agents learn optimal behaviors by interacting with environments and receiving feedback in

the form of rewards or penalties (Kelleher, 2020). ML is widely applied across domains such as cybersecurity, finance, healthcare, and e-commerce due to its ability to process and learn from high-dimensional, complex datasets.

In the context of cybersecurity, ML plays a pivotal role in the detection and mitigation of threats like phishing attacks, malware, and network intrusions. Techniques such as Support Vector Machines (SVM), Random Forest, and Naïve Bayes have demonstrated high effectiveness in classifying malicious content by learning from historical attack patterns and extracting relevant features from structured or textual data (Alzahrani et al., 2021; Kaur & Arora, 2023).

2.2.2.1 Support Vector Machines (SVM)

Support Vector Machine (SVM) model as a hyper plane in which several classes are represented. So that we may reduce the amount of mistakes we make, SVM will create the hyper plane in an iterative fashion. SVM's purpose is to classify datasets such that a maximum marginal hyper plane may be discovered (Jain & Gupta, 2023).

The high accuracy of SVMs in phishing detection has been demonstrated in various studies. For instance, research by Gupta et al. (2023) highlights that SVMs, when properly tuned, can achieve remarkable detection rates and low false positive rates, making them a reliable choice for cybersecurity applications. However, the effectiveness of SVMs comes with certain challenges. The considerable parameter adjustment necessary to maximize the model's performance is one of its primary drawbacks. It can take some time to carefully choose parameters like the kernel parameters and the penalty parameter (C) to balance the choices between variance and bias.

The computational expense associated with Support Vector Machines (SVMs), particularly when using non-linear kernels, can be significant. Training an SVM model involves solving a complex optimization problem, which can be computationally intensive, especially when dealing with large datasets typically encountered in phishing detection systems. This computational burden can limit the practicality of SVMs in real-time phishing detection, where fast processing is crucial to ensure timely

responses and effective protection. Despite these challenges, the high precision and robustness of SVMs in identifying complex patterns in data make them an asset in phishing detection frameworks.

To mitigate these limitations, hybrid approaches that combine SVMs with other machine learning techniques have been explored. These methods aim to capitalize on the strengths of SVMs while addressing their computational drawbacks. For example, integrating SVMs with feature selection techniques can reduce data dimensionality, lessen the computational load and improving real-time applicability. As cyber threats continue to evolve, the role of SVMs in phishing detection remains critical, providing a balance of high accuracy and adaptability necessary for effective cybersecurity defences.
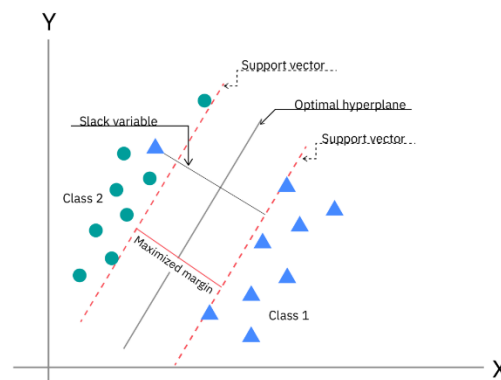
Figure 3: SVM algorithm (Jain & Gupta, 2023)

2.2.2.2 Random Forest

Random Forest is a well-known machine learning method that performs excellently when dealing with regression and classification problems. It creates a lot of decision trees during training, outputs the mode of classes for classification tasks or the mean prediction for regression tasks and increases accuracy and generalization ability by voting or averaging over all trees. Because of the ensemble technique, the model resists overfitting. Because Random Forest can handle big datasets with high dimensionality and is flexible and effective, it is widely employed. It is also resistant to noise and outliers and provides insights into feature importance. Visualization techniques can aid in model evaluation and decision-making (Gunjan & Prasad, 2024).

Random Forest Technique

Random Forest (RF) is a powerful ensemble learning algorithm that has shown strong performance in phishing detection due to its ability to handle complex patterns and noisy data. It operates by building multiple decision trees using different random subsets of the training dataset—a process known as bootstrapping. Each tree is trained independently, which introduces diversity and reduces the risk of overfitting. This is especially beneficial in phishing detection, where malicious patterns can be subtle and varied across different instances. Once the forest is constructed, each decision tree contributes to the prediction process. In classification tasks such as distinguishing between phishing and legitimate activities, each tree casts a "vote" for a class label. The final decision is based on majority voting, where the class with the most votes becomes the output. This collective decision-making mechanism improves both accuracy and stability, making Random Forest well-suited for phishing detection systems that require consistent performance under diverse threat scenarios. Refer to the figure below, which illustrates how multiple decision trees independently classify an instance and then combine their outputs through majority voting to determine the final class.
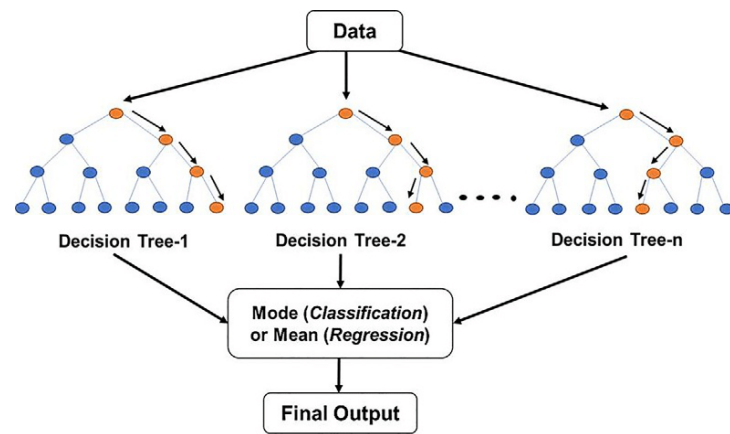
Figure 4: Random Forest Algorithm

2.2.3 Deep Learning Approaches

Deep Learning (DL) is a specialized subfield of Machine Learning (ML) that utilizes artificial neural networks with multiple layers—referred to as **deep neural networks**—to automatically learn complex patterns from large volumes of data. Inspired by the structure and function of the human brain, DL systems can model intricate relationships and hierarchical representations in data, enabling them to outperform traditional ML algorithms in tasks involving high-dimensional, unstructured inputs such as images, audio, and text.

In the context of cybersecurity, and particularly phishing detection, deep learning techniques offer significant advantages. Unlike conventional ML models that rely heavily on manually engineered features, DL models can perform **automatic feature extraction**, allowing them to learn hidden patterns in raw email content, URLs, or user interaction logs. Common DL models applied in phishing detection include:

- **Long Short-Term Memory (LSTM)**: A type of Recurrent Neural Network (RNN) that is effective in capturing sequential dependencies in phishing emails or SMS messages.

- **Transformer models (e.g., BERT)**: These models understand the contextual meaning of words in sentences, making them highly effective in detecting sophisticated and linguistically deceptive phishing emails.

The adaptability and high accuracy of DL models make them increasingly vital for building robust, intelligent, and real-time phishing detection systems.

2.2.3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to effectively capture long-term dependencies in sequential data. In the context of phishing website detection, LSTM plays a crucial role in modeling temporal patterns and contextual features that may not be apparent in static data. As described by Elberri et al. (2024), LSTM was integrated with a Convolutional Neural Network (CNN) in a hybrid deep learning architecture aimed at improving phishing detection accuracy. While CNN excels at extracting spatial features from grayscale images generated from URL and web content data, LSTM contributes by analyzing the sequential behavior and dependencies within the data. This combination allows the model to better understand complex relationships, such as those found in URL structures and embedded script behavior typical of phishing attacks. The study demonstrated that the CNN-LSTM hybrid model outperformed standalone CNN and LSTM models in terms of accuracy, sensitivity, and precision, highlighting LSTM's effectiveness in enhancing the classification of phishing and legitimate websites.

Structure of an LSTM neural network cell

LSTM is a variant of the RNN deep learning architecture designed specifically for tasks like time series analysis and classification. LSTM effectively uses a gating mechanism to deal with vanishing gradient problems in the training process.The LSTM memory cell has four gates named forgetting f, input gate i, control gate c, and output gate o. The fundamental configuration of the LSTM cell is presented in Figure 4 and it consists of the output of the previous memory cell Ct-1. (Elberri et al., 2024)

This neural network uses components such as the input signal at each time step Xt, the current memory cell Ct output, the previously hidden unit Ht − 1, and the currently hidden unit Ht. The forget gate determines the way in which the contribution from the previous time step is incorporated, resulting in a value ranging from zero to one for each datapoint in Ct-1. The input gate regulates the amount of input that is stored in the memory cell from the current time step. Meanwhile, the control gate updates the memory cell contents from Ct-1 to Ct. The output gate dictates the extent to which the internal state influences the external state at the current time step. The

symbol $\otimes$ represents the element-wise multiplication of vector elements, while $\oplus$ signifies the summation of vector along with the application of the σ (sigma) function. To formulate the LSTM artificial neural network. (Elberri et al., 2024)
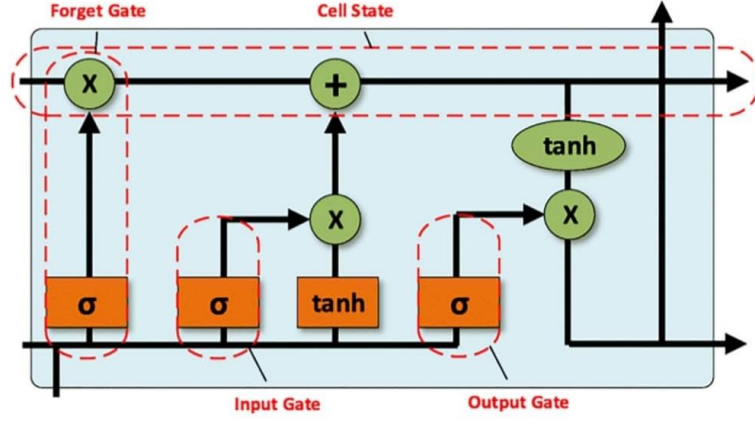
Figure 5: Structure of an LSTM neural network cell (Elberri et al., 2024)

$$f_t = \sigma(W_f.X_t + U_f.h_{t-1} + b_f)$$

$$O_t = \sigma(W_o.X_t + U_o.h_{t-1} + b_0)$$

$$\widetilde{C}_t = \tanh(W_c.X_t + U_c.h_{t-1} + b_c)$$

$$C_t = f_t.C_{t-1} + i_t.\widetilde{C}_t$$

$$h_t = \tanh(C_t) + O_t$$

Figure 6: Equations of LSTM (Elberri et al., 2024)

2.2.3.2 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a powerful deep learning model that has shown great effectiveness in detecting phishing attacks, particularly those involving social engineering through text-based communication. Phishing often relies on carefully crafted language to manipulate users into revealing sensitive information or clicking malicious links. BERT's unique bidirectional architecture allows it to analyze both the left and right context of each word in a sentence, enabling it to detect subtle linguistic patterns and deceptive cues used in phishing attempts. In recent studies, BERT has been applied to phishing detection by transforming message content into 768-dimensional contextual embeddings using the [CLS] token, which captures the overall semantics of the text. These embeddings, when combined with other extracted features such as URLs, email addresses, or phone numbers, are used as inputs for neural network classifiers to distinguish phishing messages from legitimate ones. Compared to traditional text classification techniques, BERT has demonstrated superior performance in identifying context-based phishing strategies, making it a valuable AI tool for building intelligent systems that can proactively detect and prevent social engineering-based cyber threats.
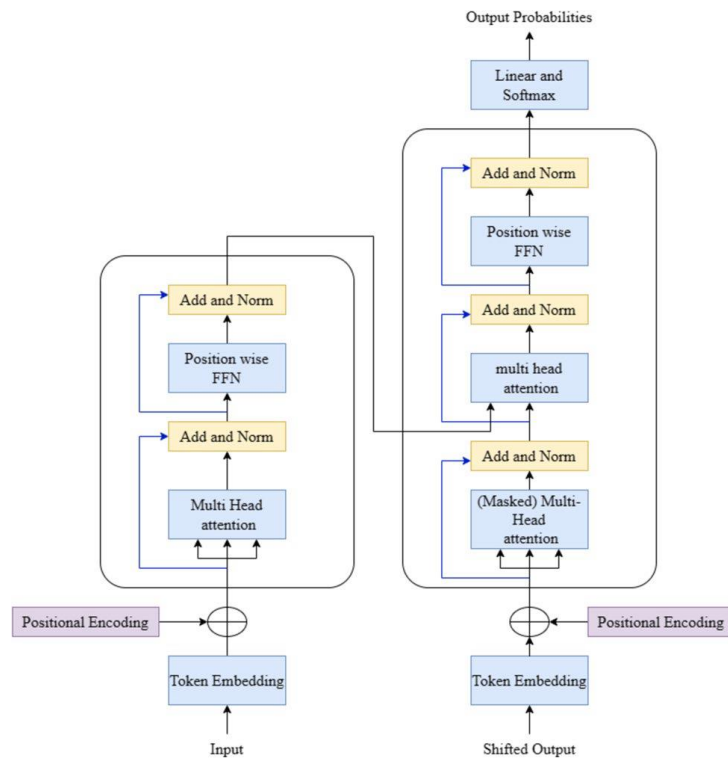
Figure 7: Transformer Model Architecture (Jain et al., 2025)

2.2.4 Reinforcement Learning Approaches

Reinforcement learning (RL) is one of the sub-domains of machine learning. The goal is to let the agent learn how to act based on the environmental state to maximize the expected long-term rewards, where the learning problem can usually be modeled as Markov decision problems (MDPs). Figure 8 shows the interactive feedback loop between the agent and the environment (Wang et al.,2020). The potential of reinforcement learning (RL) in phishing detection remains under explored. RL can enable continuous learning and adaptation, improving detection systems and complementing existing approaches. Research efforts are necessary to develop and validate RL-based methods, driving the creation of next-generation phishing detection systems that are more adaptive, accurate, and efficient (Prasanna B T Nandeesha H D, 2025).
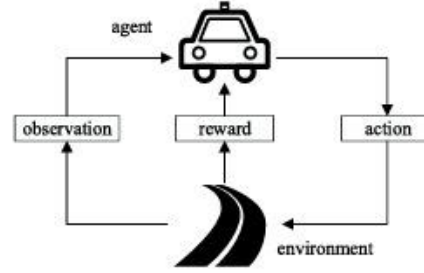
Figure 8: Interaction between the agent and the environment: at each time step, after the agent observes the environment, it chooses an action according to its policy. After the action is executed, the environment gives a reward signal to the agent and transit to a new state.

Reinforcement learning (RL) agents are generally trained in episodes, each consisting of a certain number of steps. Given an episode, the sequence of states, actions, and rewards builds the trajectory or rollout of π. Let k be the index assigned to an episode; the *cumulative discounted reward* is defined as $C_R = \sum_{k=0}^{\infty} \zeta^k R_{t+k+1}$. Then, the objective function to be optimized can be indicated as $Q(s_t, a_t) = \mathbb{E}_\pi[C_R | s_t = s, a_t = a]$ and the maximization problem, which the agent tries to solve, aims at finding $Q * (s_t, a_t) = max_\pi Q(s_t, a_t)$ for all $s \in S$ and $a \in A$.

2.2.4.1 Q- Learning

Q-learning is a value-based reinforcement learning algorithm. The goal of Q-learning is to learn the optimal action-selection policy for an agent interacting with an environment. The agent learns this by updating a table of values called the Q-table where each entry represents the value of taking a particular action in each state (Kovalchuk, 2024).

The Q-learning algorithm uses the following formula to update the Q-value for a state-action pair based on figure below:

$$Q(s,a) = Q(s,a) + \alpha \left[R + \gamma \cdot \max_a Q(s',a') - Q(s,a) \right]$$

Figure 9: Q – Learning equation (Kovalchuk, 2024)

Where:

- $Q(s, a)$ is the Q-value for state s and action a.

- alpha is the learning rate, controlling how much new information overrides old information.

- R is the immediate reward for taking action an in state *s*.

- gamma is the discount factor, representing the importance of future rewards.

- $\frac{max}{a} Q(s', a')$ is the maximum Q-value for the next state *s'*, representing the best possible reward achievable from that state.

2.2.4.2 Deep Q-Learning

Deep Q-Learning or Deep Q Network (DQN) is an extension of the basic Q-Learning algorithm, which uses deep neural networks to approximate the Q-values. Traditional Q-Learning works well for environments with a small and finite number of states, but it struggles with large or continuous state spaces due to the size of the Q-table. Deep Q-Learning overcomes this limit by replacing the Q-table with a neural network that can approximate the Q-values for every state-action pair (Amin, 2024).
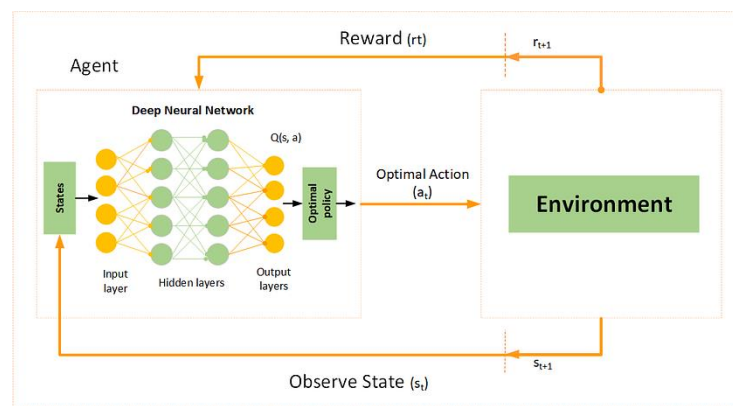
Figure 10: Structure of DQN

Concepts of Deep Q-Learning

1. **Q-Function Approximation**: Instead of using a table to store Q-values for each state-action pair, DQN uses a neural network to approximate the Q-values. The input to the network is the state, and the output is a set of Q-values for all possible actions.

2. **Experience Replay**: To stabilize the training, DQN uses a memory buffer (replay buffer) to store experiences (state, action, reward, next state). The network is trained on random mini batches of experiences from this buffer, breaking the correlation between consecutive experiences and improving sample efficiency.

3. **Target Network**: DQN introduces a second neural network, called the target network, which is used to calculate the target Q-values. This target network is updated less frequently than the main network to prevent rapid oscillations in learning.

4. **Bellman Equation in DQN**: The update rule for DQN is based on the Bellman equation, like Q-Learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)$$

Figure 11: Bellman equation i

Where:

- θ are the weights of the main Q-network,
- θ^- are the weights of the target Q-network,
- s is the current state,
- a is the action taken,
- r is the reward received,
- s' is the next state,
- $\frac{max}{a'} Q(s',a')$ is the maximum Q-value for the next state.

2.3   Analysis

2.3.1   Analysis of Dataset

Feature	Description
URLLENGTH	Length of the full URL (longer URLs can be suspicious).
DOMAINLENGTH	Length of the domain only.
ISDOMAINIP	If domain is an IP address instead of a name (1 = Yes, 0 = No).
TLD	Top-Level Domain (e.g., .com, .org).
TLDLENGTH	Length of the TLD.
TLDLEGITIMATEPROB	Probability that the TLD is used in legitimate websites.

Table 2.3.1.1: Basic URL features

Feature	Description
CharContinuationRate	Measures character consistency; lower values may indicate randomness.
URLSimilarityIndex	How similar the URL is to a known legitimate one (likely from reference data).
URLCharProb, LetterRatioInURL, DegitRatioInURL, SpacialCharRatioInURL	Various measures of character distribution.

Table 2.3.1.2: Statistical & Lexical features

Feature	Description
HasObfuscation	Whether the URL contains confusing or misleading elements.
NoOfObfuscatedChar, ObfuscationRatio	Measures on how heavily obfuscated the URL is.

Table 2.3.1.3: Obfuscation and Manipulation Indicators

Feature	Description
NoOfEqualsInURL, NoOfQMarkInURL, NoOfAmpersandInURL, NoOfOtherSpecialCharsInURL	These characters often appear in phishing query strings.

Table 2.3.1.4: Special characters in URL

Feature	Description
IsHTTPS	Whether it uses HTTPS (1 = Yes, 0 = No).
LineOfCode, LargestLineLength	Measures of the HTML code complexity.
HasTitle, Title, DomainTitleMatchScore, URLTitleMatchScore	Checks alignment between page title and URL/domain.

Table 2.3.1.5: HTTPS and Web Page Content

Feature	Description
HasFavicon, HasSubmitButton, HasPasswordField, HasHiddenFields	Phishing pages often fake visuals or hide elements.
NoOfImage, NoOfCSS, NoOfJS	Counts of resources included— can signal legitimacy.

Table 2.3.1.6: Visual & Form elements

Feature	Description
`NoOfURLRedirect,` `NoOfSelfRedirect`	Phishing pages often redirect to obscure their origin.
`Robots, IsResponsive`	Site behavior and SEO configuration.

Table 2.3.1.7: Redirection & Network Behaviour

Feature	Description
`Bank, Pay,` **`Crypto`**	Binary flags if the site seems related to banks, payment, or cryptocurrency.

Table 2.3.1.8: Suspicious Intent Flags

2.3.2   Performance metrics

These metrics evaluate how well the machine learning models identify phishing websites or emails:

a. Accuracy

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

Figure 12: accuracy formula

- TP: True Positives (phishing sites correctly detected)
- TN: True Negatives (legitimate sites correctly detected)
- FP: False Positives (legitimate sites misclassified as phishing)
- FN: False Negatives (phishing sites missed)

b. Precision

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

Figure 13: Precision formula

c. Recall (Sensitivity)

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

Figure 14: Recall formula

d. F1 score - Harmonic mean of Precision and Recall

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 15: F1 score formula

2.4   Critical Review

Primary studies on Machine Learning techniques.

Ref	Applied Approach	Used Algorithm	Used Data set	Main Findings	Limitation/Challenges
Rashid et al., 2020	Machine Learning	Support Vector Machine	Alexa, Common Crawl archive (5000 URL)	The suggested method categorizes phishing and legal websites with 95.66% of Accuracy.	The study is very shallow and has used only one classifier, i.e., SVM, and five features for detecting phishing websites. A small data set was collected using GNU and Python scripts. Moreover, only one performance metric, i.e., Accuracy, was used for model evaluation.

Basit et al., 2020	Machine Learning	Random Forest K-Nearest Neighbor Decision tree Artificial Neural Network	UCI machine learning repository 11,055 instances 30 features	The combination of K-Nearest Neighbors and Random Forest classifier detects phishing attacks with 97.33% accuracy.	The study has not used multiple data sets to evaluate their ensemble model. Further, the UCI dataset is open source and has normalized features. It does not include the Original URL. The study has also not included any feature selection procedure. The study has picked the open-source data set and existing ML algorithms for their study. It still needs to include the calibration values of each selected ML approach.

Saha et al., 2020	Machine Learning	Random Forest Decision tree	Kaggle 11,504 URL 32 attributes	The highest Accuracy of 97.00% was achieved through the Random Forest classifier.	The study has used only two Machine Learning approaches and only a single dataset. They used the PCA feature selection technique for analyzing data set characteristics. The study intended to use CNN for anticipating phishing attacks. They did not compare the results with existing equivalent techniques. It is a very shallow study.

| Kasim, 2021 | Machine Learning & Heuristic | Support Vector Machine LightGBM Multilayer Perceptron Convolution Neural Network | ISCXURL-2016 2978 instances and 77 different features | The current technique uses the Light Gradient Boosted Machine model to classify the features encoded with SAE-PCA at a rate of 99.60% accuracy. | The study has done the experiment on a limited dataset of 2978 instances, and PCA has reduced the feature selection from 77 to 20; these are also very limited. |

Geyik et al., 2021	Machine Learning	Decision tree	PhishTank	The highest	Performance achieved with the
		Logistic Regression	Alexa	Accuracy produced	dataset is very low compared
		Naive Bayes	Common-crawl	by the Random	to other studies with similar
		Random Forest		Forest classifier is	classifiers and datasets.
				83.0%.	

Table 2.4.1: Primary studies on Machine Learning techniques

Primary studies on Deep Learning techniques

Ref	Applied Approach	Used Algorithm	Results	Main Findings	Limitation/Challenges
Bagui et al., 2019	Deep Learning	LSTM, CNN, and WordEmbedding DL	Word embedding DL Accuracy = 98.89%, F1 = NA DT, NB, SVM, CNN, LSTM Accuracy= < 97.50%, F1 = NA	Showed the context of the email is important in detecting phishing email.	Cannot detect malicious link or attachments inside email body.
Giri et al., 2022	Deep Learning	GloVe + CNN, and BERT + FCN	GloVe + CNN Accuracy = 98%, F1 = 0.9749 BERT+FCN Accuracy = 96%, F1 = 0.9576	Compare the combination of word embedding techniques and DL architectures.	Cannot detect malicious link or attachment inside email body. BERT model has limitation of maximum 512 tokens (words length).
Ramprasath et al., 2023	Deep Learning	RNN with LSTM cells	RNN Accuracy =99.1%, F1 = 0.958 SVM Accuracy = 98.2%, F1 = 0.932	N/A	Cannot detect malicious link or attachment inside email body. No dropout layer to prevent overfitting.

			CkNN Accuracy =98.1%, F1 = 0.928		
Valecha et al., 2021	Deep Learning	Bi-LSTM	Accuracy = 95.97%, F1 = 0.9569	Phishing detection based on gain and loss persuasion cues of text context.	Cannot detect malicious link or attachment inside email body. Manual coding of persuasion cues labels and manual hyperparameter tuning.
Paradkar, 2023	Deep Learning	LSTM, Bi-SLTM, CNN	CNN Accuracy =98.05%, F1 = 0.9826 LSTM Accuracy = 97.32%, F1 = 0.9786 Bi-LSTM Accuracy = 98.04%, F1= 0.9825 NB, LR, SVM, DT Accuracy = <73.23%, F1 = NA	N/A	Cannot detect malicious link or attachment inside email body.
Divakarla et al., 2023	Deep Learning	LSTM, Bi-LSTM, CNN+RNN	LSTM Accuracy = 98.8%, F1 = 0.987	N/A	Cannot detect malicious link or at tachment inside

			Bi-LSTM Accuracy = 95.4%, F1= 0.95 CNN+RNN Accuracy = 97.9%, F1 =0.956		email body. No dropout layer to prevent overfitting.
Gholampour et al., 2023	Deep Learning	ALBERT, ROBERTA, BERT, DE BERTA, DEBERT, SQ, and YOSO	BERT and its variants Accuracy = 98% ~ 99%, F1 = 0.92 ~ 0.97	Developed new adversarial ham/phish dataset. Proposed ensemble method with KNN as shield model to assign correct label before feeding to DL models.	Cannot detect malicious link or attachment inside email body. Require high computational resources. Maximum tokens of BERT an AL BERT is 512. Dataset is small and model may overfitted.
Bountakas et al., 2021	Deep Learning	BERT	Balance dataset: Word2Vec+RF Accuracy = 98.95%, F1 = 0.9897 Other combination Accuracy = <97%, F1 = 0.9744	Compare the combination of NLP techniques and ML models.	Cannot detect malicious link or at attachment inside email body. BERT model has a limitation of maximum 512 tokens.

			Imbalanced dataset: Word2Vec+LR Accuracy = 98.62%, F1= 0.9241 Other combination Accuracy = < 98.42%, F1 = 0.8996		
Qachfar et al., 2022	Deep Learning	BERT	BERT F1=0.991 to 0.998 RF, DT, SVM, SGD, KNN, GNB, LR, LSTM, CNN F1=0.72 to 0.99	Propose method to reduce the impact of imbalanced data by adding synthetic training data.	Cannot detect malicious link or attachment inside email body. BERT model has a limitation of maximum 512 tokens.
Heet al., 2024	Deep Learning	LSTM, Bi-SLTM	LSTM-XGB Accuracy = 98.35%, F1 = 0.9824 L-SVM, L-GNB, L-DTC Accuracy = <97%, F1 = <0.96%	Double-layer detection mechanism for both phishing and insider threats.	Cannot detect image links embedded in phishing emails. Dataset is small and model may be overfitted.
AbdulNabi et al., 2021	Deep Learning	BERT	BERT Accuracy = 97.30%, F1 = 0.9696 BiLSTM Accuracy = 96.43%, F1 = 0.96	N/A	Cannot detect malicious link or attachment inside email body. Maximum

			KNN and NB Accuracy <94%, F1 = <0.94%		input sequence length is 300.
Muralidhar an et al., 2023	Deep Learning	BERT, CNN	Accuracy = 99.2%, F1 = 0.941	Ensemble learning to analyze all email segments including attachment.	Inference time to process and may need high computing resources.
Lee et al., 2021	Deep Learning	CNN-LSTM, BERT	RF-BERT+RF-CNN+LSTM AUPRC=0.9997, F1 =NA RF-Word2Vec+LSTM-CNN+LSTM AUPRC = 0.9851, F1 = NA	Propose modular architecture to analyze all components of email except attachment.	Cannot detect malicious attachment inside email body. BERT model has a limitation of maximum 512 tokens (words length). Require high computational resources.

Table 2.4.2: Primary studies on Deep Learning techniques

Primary studies on Reinforcement Learning

Ref	Applied Approach	Used Algorithm	Result	Main Findings	Limitation/Challenges
H. Kamal et al., 2024	Reinforcement Learning	Q-Learning	Achieved high detection accuracy with faster convergence	Proposed Q-Learning model for phishing detection. Demonstrated that RL outperforms traditional ML models.	Limited to URL-based features. Dataset and real-world application scalability were not fully addressed.
Chatterjee and Namin, 2024	Reinforcement Learning	Q-Learning	High detection rates with optimized policy	Successfully applied Q-Learning for detecting malicious websites, demonstrating high reward optimization.	Focused on specific phishing datasets, potentially limiting generalization.
Nguyen and Reddi, 2023	Reinforcement Learning	Deep Q-Network (DQN)	High adaptability and faster threat response.	Showcased DRL's potential in cybersecurity with real-time learning capabilities	High computational cost and complexity in real-world deployment.

| Chatterjee et al., 2023 | Reinforcement Learning | Deep Q-Learning | High accuracy and faster response time | Demonstrated improved phishing detection accuracy with DRL models. | Potential overfitting and lack of interpretability in complex scenarios. |

Table 2.4.3: Primary studies Reinforcement techniques

Table 2.4.4: Comparison of Methods Used in Previous Studies

Table Comparison of Methods Used in Previous Studies													
Journal/Study	RF	NB	SVM	DT	KNN	AB	LR	CNN	LSTM	BiLSTM	BERT	QL	DQN
Rashid et al. (2020)			/										
Basit et al. (2020)	/			/	/								
Saha et al. (2020)	/			/									
Kasim (2021)			/					/					
Geyik et al. (2021)	/	/		/			/						
Bagui et al. (2019)		/	/					/	/				
Giri et al. (2022)								/			/		
Ramprasath et al. (2023)			/		/				/				
Valecha et al. (2021)										/			
Paradkar (2023)			/	/			/	/	/	/			
Divakarla et al. (2023)									/	/			
Gholampour et al. (2023)					/						/		
Bountakas et al. (2021)	/						/				/		
Qachfar et al. (2022)	/	/	/	/	/	/	/	/	/		/		
He et al. (2024)				/					/	/			

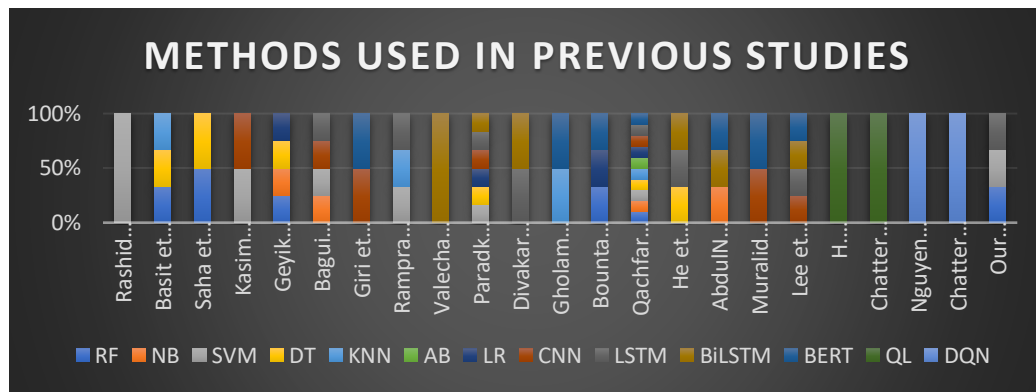|												
Lee et al. (2021)								/	/	/	/	
H. Kamal et al. (2024)											/	
Chatterjee & Namin (2024)											/	
Nguyen & Reddi (2023)												/
Chatterjee et al. (2023)												/
Our proposed model			/						/			

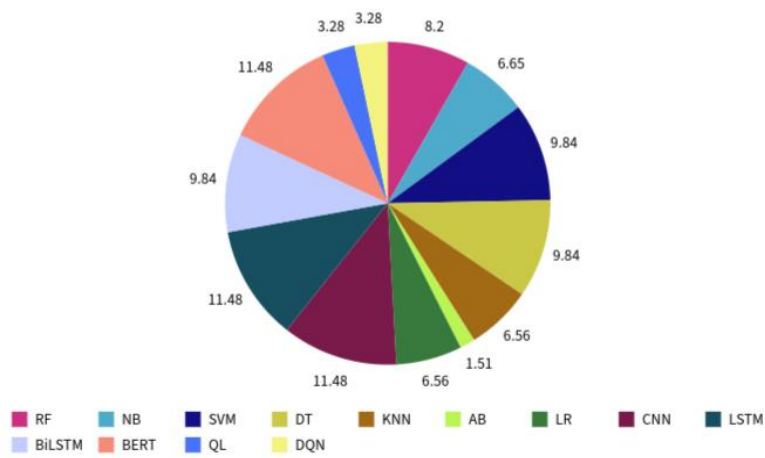**Figure 16: Bar chart of methods used in previous studies**

Figure 17: Pie chart of methods used in previous studies

2.5   **Research Gap Analysis**

Despite significant advancements in AI-driven phishing detection systems, considerable limitations persist in current methodologies. Much of the existing research primarily concentrates on either machine learning (ML) or deep learning (DL) methods in isolation, disregarding the potential benefits of hybrid models that combine both techniques. Additionally, many models are trained and evaluated on outdated or limited datasets, which do not adequately reflect the complexity and evolving nature of modern phishing attacks. While reinforcement learning (RL) offers a promising pathway for enabling systems to adapt to emerging threats, its application in phishing detection has mainly remained underexplored. Furthermore, deep learning models like BERT face practical challenges, including input length constraints and substantial computational costs, which impede their deployment in real-time scenarios.

Additionally, many current detection systems struggle to identify advanced phishing attempts that involve obfuscated links or malicious attachments. The absence of standardized benchmarks and comparative studies further complicates the fair evaluation and refinement of existing solutions. These challenges underscore the need for more flexible, scalable, and adaptive detection systems capable of effectively addressing real-world phishing threats.

Despite these advancements, existing AI-based phishing detection methods still face fundamental challenges due to their reliance on either machine learning (ML) or deep learning (DL) techniques alone. Machine learning models, such as Support Vector Machines (SVM), are highly proficient in structured data analysis but often struggle to process unstructured content, such as text from phishing emails or fraudulent websites. On the other hand, deep learning models like Long Short-Term Memory (LSTM) networks excel at analyzing sequential and unstructured data; however, they are computationally intensive and require substantial amounts of training data, making them less feasible for real-time applications.

This research aims to bridge these limitations by employing a hybrid solution that combines both machine learning (ML) and deep learning (DL) approaches, effectively leveraging the strengths of each. Machine learning models, such as Support Vector Machines (SVM), are well-suited for handling structured data, like URLs, 47

while deep learning models, like Long Short-Term Memory (LSTM), can efficiently process complex unstructured data, like the content of phishing emails or web pages. The hybrid approach not only improves adaptability to dynamic phishing tactics but also enhances detection accuracy and minimizes false positives by utilizing the complementary strengths of both techniques. By integrating the advantages of both machine learning (ML) and deep learning (DL), this hybrid model offers greater robustness and scalability compared to single-model approaches.

In alignment with this objective, the study by Rao et al. (2019) introduced a multi-model ensemble combining LSTM and SVM for phishing detection on mobile devices. The proposed hybrid model demonstrated superior performance compared to traditional methods, achieving an accuracy of 97.30%. This study highlighted that integrating deep learning for feature extraction with machine learning for classification significantly improved detection speed, accuracy, and robustness, particularly in addressing zero-day phishing attacks.

Building on these findings, the hybrid model proposed in this research combines the speed and interpretability of machine learning with the contextual learning capabilities of deep understanding, making it more adaptable to the evolving nature of phishing threats.

In conclusion, while both machine learning and deep learning models have inherent limitations when used independently, the hybrid AI solution offers a more comprehensive and practical approach to phishing detection. This makes it a more viable option for addressing the sophisticated and rapidly evolving landscape of social engineering attacks.

2.6   Project Solution

To address the limitations identified in prior research, particularly the low adaptability of single-model approaches and the inability to detect complex phishing patterns, this project proposes a hybrid Machine Learning (ML) and Deep Learning (DL) solution using the KNIME Analytics Platform. The hybrid approach combines the structured pattern recognition strengths of traditional ML algorithms such as Support Vector Machine (SVM) with the advanced feature extraction capabilities of Deep Learning models like Long Short-Term Memory (LSTM). KNIME is selected as the development environment due to its flexibility, visual workflow design, and native support for both ML and DL through integration with Keras. In this solution, structured features from URLs and website metadata are processed using ML classifiers, while unstructured textual content, such as email bodies or page content, is passed to an LSTM model trained to capture sequential dependencies typical of phishing language. The outputs from both paths are then combined using an ensemble mechanism or decision logic to improve overall detection accuracy. This dual-model strategy leverages the speed and interpretability of ML with the contextual learning strength of DL, enabling a more robust and scalable detection system that adapts to the evolving nature of phishing attacks. By deploying this hybrid model within KNIME, the project ensures a practical, modular framework suitable for academic experimentation and real-world application.

CHAPTER 3: METHODOLOGY

3.1   Introduction

This chapter outlines the methodology used to develop and evaluate a hybrid Machine Learning (ML) and Deep Learning (DL) phishing detection framework using the KNIME Analytics Platform. The process involves data preprocessing, model training, evaluation, and system integration. The hybrid approach leverages structured and unstructured data to enhance detection accuracy. Additionally, the Agile methodology is adopted to enable iterative development, continuous feedback, and adaptive refinement of the system based on performance evaluations.

3.2   Methodology

The project adopts a hybrid methodological strategy combining Agile software development for planning and iterative refinement with experimental workflow development in KNIME, implementing a modular pipeline where structured URL features are classified using ML models like SVM, while unstructured email content is analyzed using LSTM, with all outputs ensembled for the final phishing detection decision.
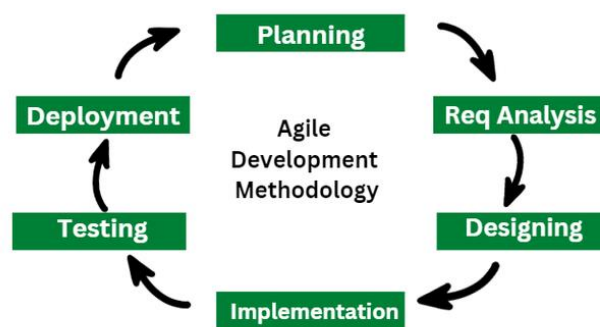
Figure 18: Agile Methodology

3.2.1 Requirement Analysis

3.2.1.1 Hardware Requirement

A laptop is used as a workstation for all tasks, from researching to documenting. The laptop specifications are shown in the table below.

Table 3.2.1.1: Hardware Requirement

Specification	Description
Processor Type	AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz
Operating System	Windows 11 Home Single Language Version: 24H2
Operating System Type	64-bit operating system, x64-based processor
RAM	8.00 GB (7.42 GB usable)
Storage	477 GB
Display Resolution	1920 x 1080, 144 Hz

3.2.1.2 Software Requirement

This project's development includes the usage of some software. The software used in this project is listed in the table below.

Table 3.2.1.2: Software Requirement

Software	Description
Windows 11	An environment of operating system used for project execution.
KNIME Analytics Platform	An open-source platform for data analytics, machine learning, and deep learning, used in this research to implement a Phishing detection using hybrid AI solution.
Microsoft Word 365	Software is used to complete project reporting and documentation.
Microsoft Excel 365	For datasets collections.
Microsoft Visio 2021	Software used to create structure diagram

3.2.2 System Design

The proposed system is designed to accept both structured and unstructured inputs, which are processed through parallel pipelines. The input layer handles structured data such as URL features including length, number of special characters, and domain information, and unstructured data such as email text or website content. In the processing layer, the ML pipeline in KNIME applies models like SVM to classify structured features, while the DL pipeline implemented in KNIME with Keras uses LSTM to analyze the contextual semantics of unstructured text. The ensemble layer then combines the outputs from both pipelines using decision rules such as majority voting or weighted scoring. Finally, the output layer delivers the final classification result, indicating whether the input is phishing or legitimate.

3.2.3   Implementation

This project was implemented utilising the KNIME Analytics Platform, which offers an easy-to-use framework for creating processes that enable deep learning and machine learning. Using conventional machine learning models like Support Vector Machine (SVM), which are accessible in KNIME, the structured data such as URL features was analysed. KNIME's Keras plugin was used to incorporate a Long ShortTerm Memory (LSTM) model for the analysis of unstructured data, including email text. A decision process was used to integrate the predictions made by the ML and DL models after training to provide the final classification outcome. Iterative testing and improvement were possible since the whole process from data input and cleaning to model training and evaluation was controlled inside KNIME's modular workflow interface.

3.2.4   Testing

The system is tested using test data that was not used during training. Accuracy shows how correct the model is. Precision shows how many predicted phishing cases are phishing. Recall shows how many real phishing cases are correctly found. F1-score combines precision and recall. The results are compared between ML models, DL model, and the combined hybrid model.

3.2.5   Deployment

Once all data from the structured and unstructured sources have been processed through their respective pipelines, the results are presented and analysed about other related research studies. A critical evaluation is conducted based on the findings. Graphs are plotted for key performance metrics, including accuracy, precision, recall, and F1-score, to compare the performance of individual ML and DL models, as well as the ensemble model. A comparison table is also provided to summarise the results across all models and metrics. Examples of the graphical and tabular results are illustrated in the figure below.

Machine Learning Models	Precision	Recall	F1-measure
AdaBoost Classifier	94.35	93.93	93.95
Bagging Classifier	96.84	96.77	96.78
Decision Trees	96.79	96.75	96.76
Extra Tree Classifier	95.22	95.10	95.11
Gradient Boosting Classifier	94.40	93.73	93.75
K-Nearest Neighbour	94.01	93.77	93.73
Logistic Regression	96.76	96.71	96.72
Neural Networks	96.66	96.64	96.65
Random Forest	96.79	96.75	96.89
Support Vector Machines	97.68	97.63	97.64

Figure 19: Performances of selected machine learning models

(Sameen et al., 2020)

	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	3081
1	0.99	0.99	0.99	2331
micro avg	0.99	0.99	0.99	5412
macro avg	0.99	0.99	0.99	5412
weighted avg	0.99	0.99	0.99	5412
samples avg	0.99	0.99	0.99	5412

Figure 20: LSTM classification report (Atawneh & Aljehani, 2023)
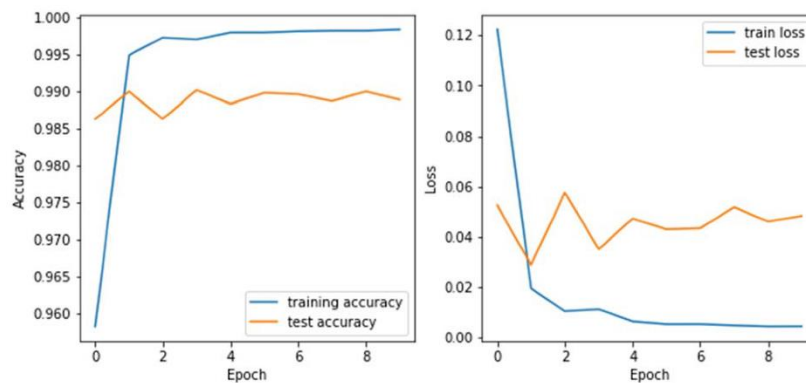
Figure 21: LSTM model training and testing accuracy

(Atawneh & Aljehani, 2023)
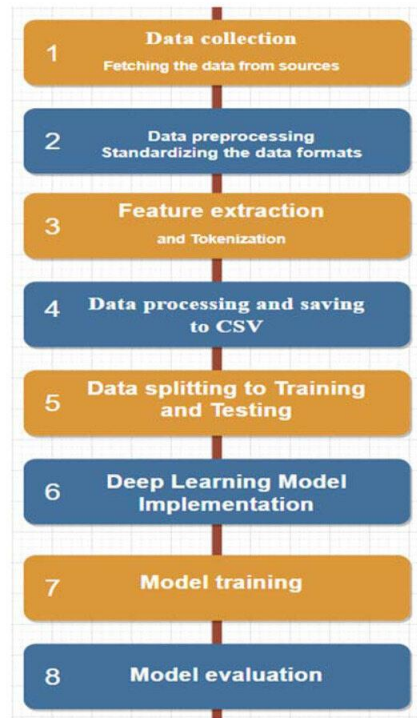
3.2.6 Previously proposed methodology

Figure 22: Methodology for phishing detection

(Atawneh & Aljehani, 2023)

3.3 Project Schedule and Milestones

3.3.1 Project Milestones

A milestone is a project marker that denotes a shift or stage of progress. Thus, project milestones are essential in keeping track of upcoming events or goals across the timeline.

Table 3.3.1: Research Milestone of PSM 1 and PSM 2

Week	Activity
1	• Proposal discussion • Proposal assessment & verification
2	• Chapter 2(Analysis Begins)
3	• Chapter 2 • Chapter 1
4	• Chapter 2
5	• Chapter 2
6	• Chapter 2 • Chapter 1
7	• Chapter 2 (last discussion before mid-semester break)
Mid- Semester Break	
8	• Chapter 2 • Chapter 3
9	• Chapter 3
10	• Chapter 4
11	• Chapter 4/ Project Demo
12	• Chapter 4/ Project Demo
13	• Project Demo • PSM 1 Report
14	• Project Demo • Final report submission and slide presentation for evaluation by Supervisor and Evaluator
15	• Revision week
16	• Final Exam
17	• Final Exam
18	• Installation of the tools might be needed for this project (Knime, Anaconda Navigator, python3 package) • Do corrections for the report that has been commented on by the Evaluator

19	- Do SVM learning process in KNIME - Use 3 sample datasets that contain 3K data by different sources - Get the accuracy of the training and testing
20	- Do report for SVM (chapter 5) - Do LSTM learning process in KNIME
21	- Continue LSTM learning, using python script in KNIME - Get the accuracy of the training and testing - Do report for LSTM (chapter 5)
22	- Do report for Hybrid (SVM + LSTM) chapter 5 - Get the accuracy of the training and testing
23	Generate graph from KNIME to attach on chapter 6 as a result
24	Continuing slide for presentation PSM 2
25	Recheck the report and slide before submitting to the supervisor and evaluator

3.3.2 Project Gantt Chart

Gantt Charts provide a thorough overview of the project from start to finish, as well as all the activities required to complete the project. It aids in demonstrating how far the tasks have progressed.

PHASE	WEEK																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Requirement Analysis	■	■	■	■	■	■																			
System Design							■	■	■	■	■	■	■	■											
Implementation															■	■	■	■							
Testing																			■	■	■	■			
Deployment																							■	■	■

3.4   **Summary**

This chapter discusses the methodology employed in this research, focusing on the application of hybrid solutions, which combine machine learning and deep learning techniques, to detect phishing attacks based on URL and email content. To ensure a structured and timely analysis, this chapter also includes a detailed project milestone and Gantt Chart. This structured figure provides a clear timeline and outlines key activities, helping to manage the analysis's progress effectively. The systematic approach outlined in this methodology ensures that the research is carried out efficiently and thoroughly. The upcoming chapter will delve into the research implementation, providing detailed insight into how the methodologies discussed are practically applied to achieve the analysis's objectives.

CHAPTER 4: DESIGN

4.1   Introduction

This chapter presents the system design of the proposed hybrid Machine Learning and Deep Learning phishing detection framework developed using the KNIME Analytics Platform. The design phase is crucial as it translates the project's conceptual framework into a structured and implementable architecture. It involves both logical and physical designs that define how data flows through the system, how different components interact, and how the system integrates machine learning classifiers with deep learning models to produce accurate phishing detection results. The objective of this chapter is to provide a clear and detailed overview of the system's internal structure, including the data processing pipeline, model training paths, and ensemble decision logic. The design considerations ensure the system is modular, scalable, and adaptable for real-world deployment while remaining aligned with the project objectives and addressing the gaps identified in the literature review.

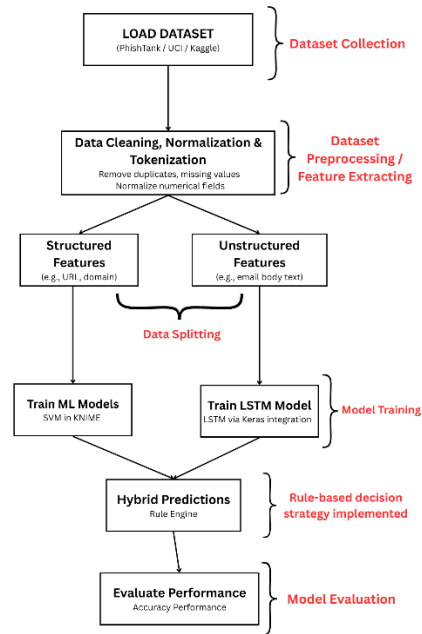**4.2   Design Framework of Hybrid Approach for Phishing Detection in AI**

Figure 23: Design methodology framework for hybrid approach for phishing detection

The process begins with **Dataset Collection**, where phishing and legitimate datasets are obtained from repositories such as PhishTank, UCI, or Kaggle.

Next, the data undergoes **Preprocessing and Feature Extraction**, which includes cleaning (removing duplicates and handling missing values), normalization of numerical fields, and tokenization for textual data. This ensures the dataset is consistent and suitable for model training.

After preprocessing, the dataset is divided in the **Data Splitting** stage into two main feature categories:

- **Structured Features** (e.g., URL-based attributes such as length, presence of IP addresses, or domain details).

- **Unstructured Features** (e.g., raw textual content like email body text).

These features are then processed in **Model Training**:

- Structured features are fed into **Machine Learning model**, specifically Support Vector Machines (SVM) in KNIME.

- Unstructured features are passed into a **Deep Learning model**, Long Short-Term Memory (LSTM), using Keras integration in KNIME.

The outputs of both models are combined in the **Hybrid Predictions** stage, where a **rule-based decision strategy** is implemented via KNIME's Rule Engine. The rule engine ensures consistent decision-making by prioritizing LSTM predictions in cases of disagreement, as LSTM demonstrated higher accuracy during evaluation.

Finally, the system proceeds to **Model Evaluation**, where the performance of the hybrid approach is measured using accuracy as the primary metric. This evaluation validates that the hybrid ensemble, leveraging both structured and unstructured features, enhances phishing detection performance compared to single-model approaches.

4.3   Summary

To summarize, this chapter is essential as it outlines a straightforward approach to developing a phishing detection system that utilizes both machine learning and deep learning techniques. It presents the overall system architecture, detailing how structured and unstructured data are processed through parallel pipelines, and describes the modular design, data preprocessing methods, model selection, and ensemble strategy. The evaluation plan and deployment process are also addressed, including how performance metrics will be measured and visualized. The next phase, implementation, will be guided by the analysis and design established in this chapter.

CHAPTER 5: IMPLEMENTATION

5.1   Introduction

This chapter showcases the real-world execution of the suggested hybrid phishing detection system utilizing the KNIME Analytics Platform. The initiative combines Machine Learning (ML) and Deep Learning (DL) methods to identify phishing attacks from structured URL information and unstructured email/text data, respectively. The ML part utilizes Support Vector Machine (SVM) to analyze structured features like URL length, special character presence, and domain indicators, whereas the DL part uses Long Short-Term Memory (LSTM) networks through Keras integration in KNIME to examine email and text patterns characteristic of phishing attempts.

The execution was done in a modular way, utilizing KNIME's graphical interface for constructing workflows based on nodes. Distinct pipelines were created for structured and unstructured data, with each including preprocessing, model training, and prediction phases. The outputs of both models are merged using an ensemble logic layer to deliver a more precise and dependable phishing detection conclusion.

This chapter details the procedures followed to set up and adjust the working environment, construct and link the KNIME workflows, and run the hybrid detection system. It additionally outlines the process of importing, cleaning, and splitting the dataset, along with the training and evaluation of the ML and DL models. This implementation aims to showcase the practicality and efficiency of a hybrid AI-driven approach for addressing phishing-related social engineering threats.

5.2 Software Development Environment Setup

The development and implementation of the hybrid phishing detection model were carried out using the **KNIME Analytics Platform**, a powerful open-source software designed for data analytics, machine learning, and deep learning workflows. The selection of KNIME was motivated by its modular, node-based interface, ease of integration with Python and Keras, and strong support for both structured and unstructured data processing.

The environment setup involved several essential installations and configurations to ensure a smooth development process for both Machine Learning (ML) and Deep Learning (DL) models:

5.2.1 KNIME Analytics Platform

- Version Used: KNIME Analytics Platform 5.4.4
- KNIME was installed as the core platform for visual workflow design. It allows for seamless creation, execution, and evaluation of ML and DL models without the need for extensive coding.
- The KNIME Deep Learning **-** Keras Integration extension was installed via the KNIME Extension Installer to enable the use of LSTM models.
- The KNIME Python Integration was also installed to support scripting and Python-based nodes where necessary.

5.2.2 Python and Keras Integration

- **Python Version**: Python 3.9.15 (via Anaconda Distribution)
- **Keras Version**: 2.13+ (with TensorFlow as backend)
- Python was installed and configured through **Anaconda Navigator** to manage packages and environments. A dedicated KNIME Python environment was created and linked to KNIME for seamless execution of Python and Keras nodes.

Required libraries installed include:

- tensorflow

- keras

- numpy

- pandas

- scikit-learn

5.3   Software Configure Management

This project follows a simple, yet effective configuration management strategy tailored for academic research, focusing on environment management and the configuration process of AI learning, which will be divided into three types of learning processes.

5.3.1   Environment Management

To maintain consistency in dependencies and avoid compatibility issues, a dedicated Python environment was created using Anaconda. The environment includes all libraries required for KNIME's Deep Learning and Python Integration:

Figure 24: Python 3 conda environment in KNIME Analytics Platform

Based on figure above a .yml file was exported (knime39-env.yml) to ensure the Python environment could be recreated if needed on another system. The environment was explicitly linked in KNIME under Preferences > KNIME > Python (Legacy), pointing to the python.exe within the Anaconda environment.
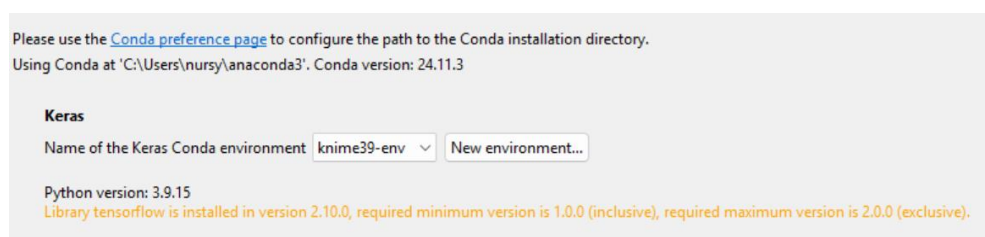

Figure 25: Keras Integration in KNIME Analytics Platform

The figure above shows that Keras has been successfully integrated into the KNIME Analytics Platform using a Conda environment named knime39-env. It also confirms that Python version 3.9.15 and TensorFlow version 2.10.0 are correctly installed, allowing the use of deep learning models like LSTM in KNIME.

5.4   AI Implementation Workflow

5.4.1   SVM

The implementation of the Support Vector Machine (SVM) model for phishing URL detection was developed in KNIME using a step-by-step workflow, as shown in Figure 26. The process starts with the CSV Reader node to load the phishing URL dataset in .csv format shown in Figure 27 and Figure 28.
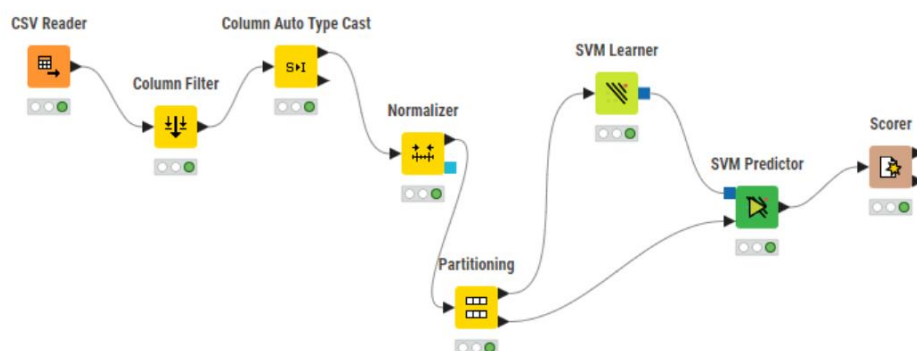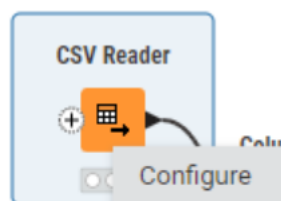

Figure 26: SVM workflow on URL datasets
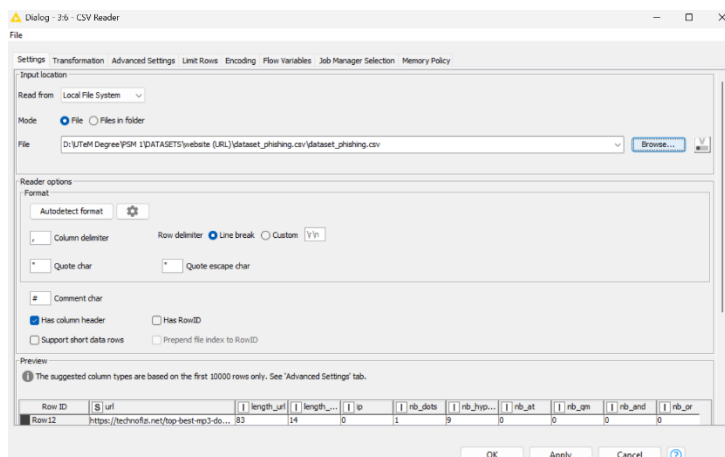

Figure 27: CSV Reader node

Figure 28: Load dataset

A Column Filter node is used to exclude the "*URL*" column, which is not needed for classification. Next, the Column Auto Type Cast node ensures the "*status*" column is correctly recognized as the label. This is shown in Figure 29 and Figure 30.
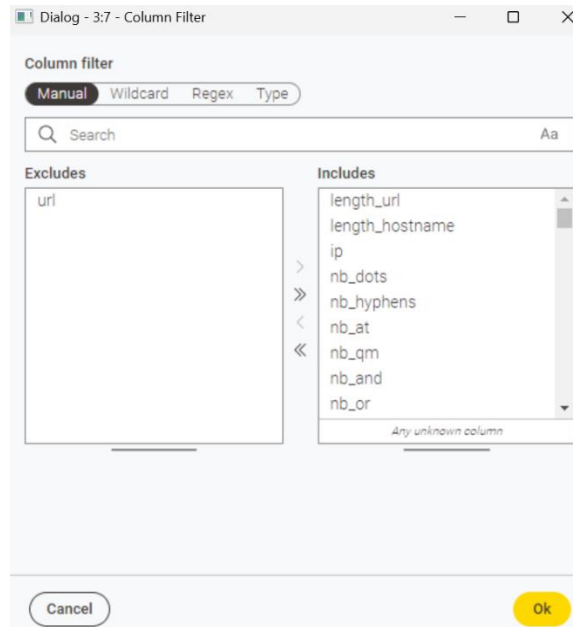

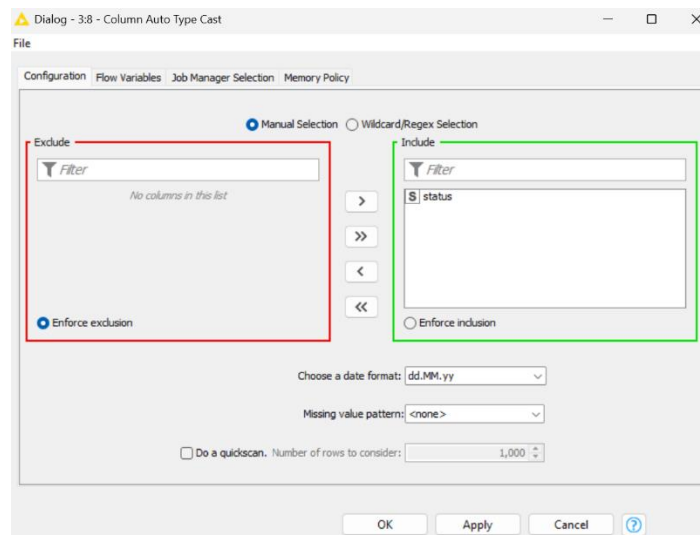**Figure 29: Column Filter**

Figure 30: Column Auto Type Cast

In Figure 31 and Figure 32, it is shown that a Normalizer node is applied to scale the numerical features between 0 and 1 using Min-Max normalization. The dataset is then split using the Partitioning node, with 80% used for training and 20% for testing. Stratified sampling is selected based on the "status" column, and a random seed is set for consistent results.
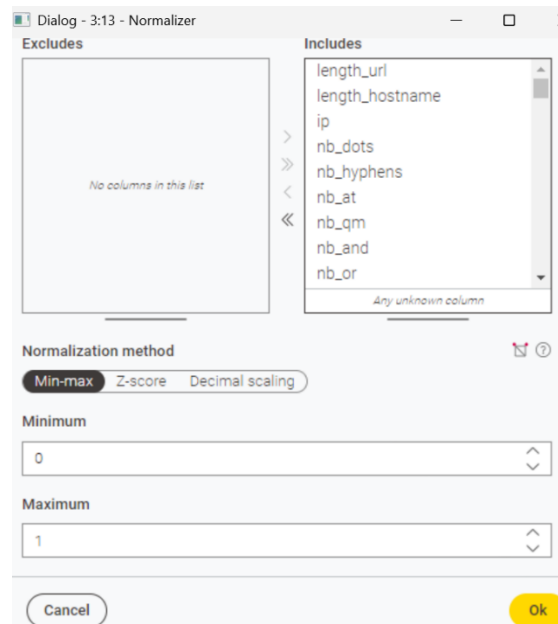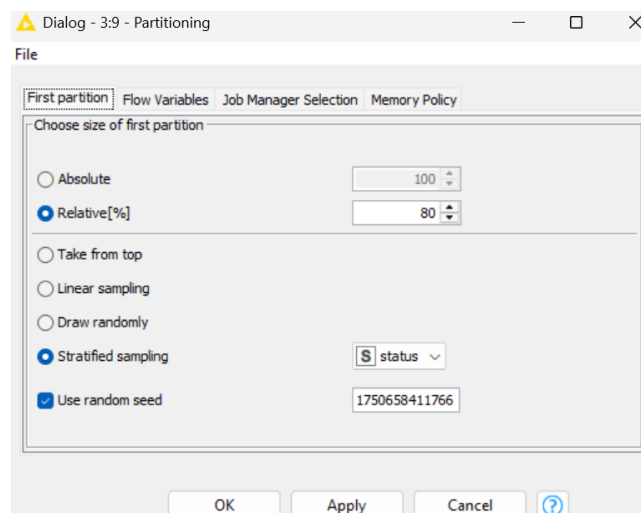

Figure 31: Normalizer configuration

Figure 32: Partitioning configuration

The SVM Learner node is configured with a Radial Basis Function (RBF) kernel and a sigma value of 1.0. This node is trained using the training set. The resulting model is passed to the SVM Predictor, which makes predictions on the test set. Finally, the Scorer node is used to evaluate the performance of the model and display the confusion matrix. As shown in figure below.
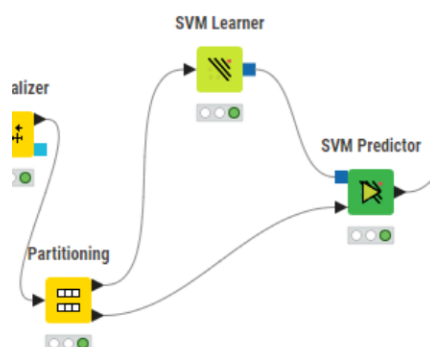

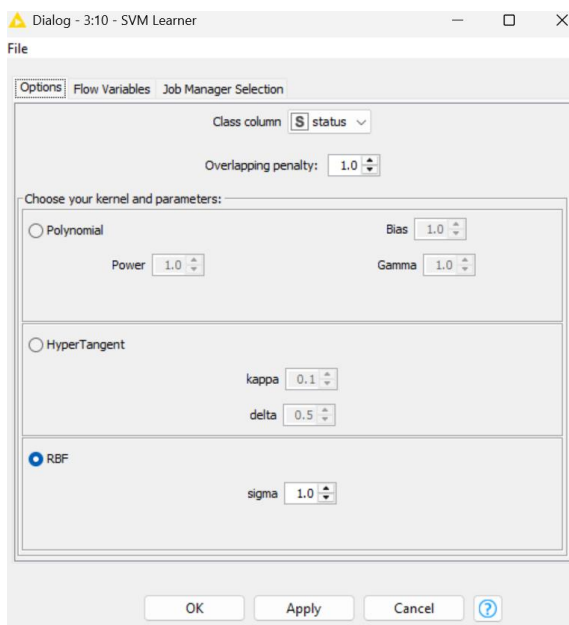**Figure 33: SVM Learner and SVM Predictor**

Figure 34: SVM Learner configurations

This SVM workflow serves as a machine learning baseline for detecting phishing attempts based on structured URL features.
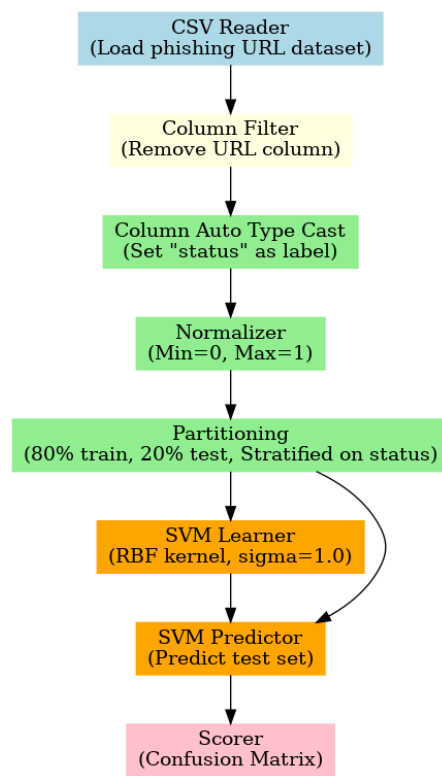
Figure 35: Step by step process of SVM Learning using KNIME

5.4.2 LSTM

The Long Short-Term Memory (LSTM) model for phishing email/text detection was implemented in KNIME using a **Python Script (legacy)** node, which enables direct integration of Python code and Keras deep learning models within the KNIME workflow. The process, illustrated in **Figure 36**, consists of three main nodes: **File Reader**, **Python Script (legacy)**, and **Table View**.
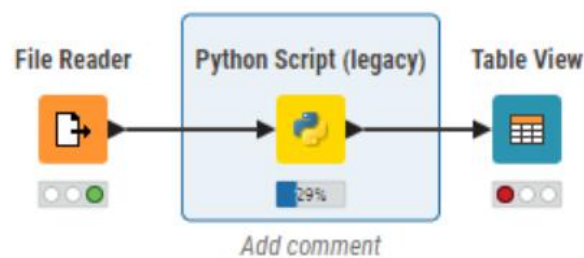
Figure 36: LSTM workflow on emails dataset

The workflow begins with the **File Reader** node, which imports the phishing email dataset in .csv format. This dataset contains the **"Email Text"** column as the input feature and the **"Email Type"** column as the target label, indicating whether an email is phishing or legitimate. The dataset is passed to the **Python Script (legacy)** node, where the LSTM model is defined, trained, and evaluated using TensorFlow and Keras.

```
1   import pandas as pd
2   import numpy as np
3   import re
4   import matplotlib.pyplot as plt
5   from sklearn.model_selection import train_test_split
6   from sklearn.metrics import classification_report
7   from tensorflow.keras.preprocessing.text import Tokenizer
8   from tensorflow.keras.preprocessing.sequence import pad_sequences
9   from tensorflow.keras.models import Sequential
10  from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
11  from tensorflow.keras.callbacks import EarlyStopping
12
13  # --- Clean text function ---
14  def clean_text(text):
15      text = text.lower()
16      text = re.sub(r'[^a-z\s]', '', text)
17      text = re.sub(r'\s+', ' ', text).strip()
18      return text
19
```

Figure 37: Python script code part 1

```
20  # --- Load dataset ---
21  df = input_table_1.copy()
22  df = df[['Email Text', 'Email Type']].dropna()
23  df.columns = ['text', 'label']
24  df['text'] = df['text'].apply(clean_text)
25  df['label'] = df['label'].map({'Safe Email': 0, 'Phishing Email': 1})
26
27  # --- Split data ---
28  X_train, X_test, y_train, y_test = train_test_split(
29      df['text'], df['label'], test_size=0.2, random_state=42
30  )
31
32  # --- Tokenize ---
33  max_words = 10000
34  max_len = 300
35  tokenizer = Tokenizer(num_words=max_words)
36  tokenizer.fit_on_texts(X_train)
37  X_train_pad = pad_sequences(tokenizer.texts_to_sequences(X_train), maxlen=max_len)
38  X_test_pad = pad_sequences(tokenizer.texts_to_sequences(X_test), maxlen=max_len)
39
```

Figure 38: Python script code part 2

```
40  # --- Build LSTM ---
41  model = Sequential()
42  model.add(Embedding(input_dim=max_words, output_dim=128, input_length=max_len))
43  model.add(LSTM(128, return_sequences=True))
44  model.add(Dropout(0.5))
45  model.add(LSTM(64))
46  model.add(Dense(1, activation='sigmoid'))
47
48  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
49
50  # --- Early stopping ---
51  es = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
52
53  # --- Train ---
54  history = model.fit(
55      X_train_pad, y_train,
56      epochs=15, batch_size=32,
57      validation_split=0.1,
58      callbacks=[es],
59      verbose=1
```

Figure 39: Python script code part 3

```python
60  )
61
62  # --- Plot Accuracy & Loss ---
63  plt.figure(figsize=(12, 5))
64
65  # Accuracy plot
66  plt.subplot(1, 2, 1)
67  plt.plot(history.history['accuracy'], label='training accuracy')
68  plt.plot(history.history['val_accuracy'], label='test accuracy')
69  plt.title('Accuracy')
70  plt.xlabel('Epoch')
71  plt.ylabel('Accuracy')
72  plt.legend()
73
74  # Loss plot
75  plt.subplot(1, 2, 2)
76  plt.plot(history.history['loss'], label='train loss')
77  plt.plot(history.history['val_loss'], label='test loss')
78  plt.title('Loss')
79  plt.xlabel('Epoch')
80  plt.ylabel('Loss')
```

Figure 40: Python script code part 4

```python
81   plt.legend()
82
83   plt.show()
84
85   # --- Predict ---
86   y_pred_probs = model.predict(X_test_pad)
87   y_pred = (y_pred_probs > 0.5).astype(int).flatten()
88
89   # --- Classification Report ---
90   report_dict = classification_report(y_test, y_pred, output_dict=True, digits=2)
91
92   # Replace numeric class names with text labels
93   if '0' in report_dict:
94       report_dict['Safe Email'] = report_dict.pop('0')
95   if '1' in report_dict:
96       report_dict['Phishing Email'] = report_dict.pop('1')
97
98   # Convert to DataFrame
99   report_df = pd.DataFrame(report_dict).transpose().reset_index()
100  report_df.rename(columns={'index': 'Class'}, inplace=True)
```

Figure 41: Python script code part 5

```python
102  # Output to KNIME
103  output_table_1 = report_df
```

Figure 42: Python script code part 6

Inside the Python Script node, several key steps were executed:

1. **Dataset Loading** – The dataset from KNIME's input table (input_table_1) is imported into a Pandas DataFrame. The email text is set as the input feature, and the email type is set as the target label. Missing values are removed, and the target labels are mapped into binary form (Safe Email → 0, Phishing Email → 1).

2. **Text Cleaning** – A custom clean_text function is applied to the email text to standardize the input. The function converts text to lowercase, removes non-alphabetical characters, collapses multiple spaces into one, and strips leading/trailing spaces. This reduces noise and ensures consistent formatting.

3. **Data Splitting** – The cleaned dataset is split into training (80%) and testing (20%) subsets using train_test_split, with a fixed random seed to ensure reproducibility.

4. **Text Tokenization and Padding** – The Keras Tokenizer is fitted on the training data with a vocabulary size of 10,000 words. Email text is transformed into sequences of integer word indices. These sequences are then padded or truncated to a uniform length of 300 tokens using pad_sequences, ensuring the LSTM model receives fixed-size inputs.

5. **LSTM Model Construction** – A Sequential Keras model is defined with the following layers:

   o **Embedding layer**: Converts tokenized word indices into dense 128-dimensional vectors.

   o **First LSTM layer**: 128 units, configured to return sequences for further temporal processing.

   o **Dropout layer**: Dropout rate of 0.5 to prevent overfitting.

   o **Second LSTM layer**: 64 units to capture higher-level sequential patterns.

   o **Dense output layer**: Single neuron with sigmoid activation for binary classification.

6. **Model Compilation and Training** – The model is compiled with binary_crossentropy loss and the Adam optimizer, using accuracy as the evaluation metric. Early stopping is applied, monitoring validation loss with a patience of 3 epochs, and restoring the best weights. The model is trained for

up to 15 epochs with a batch size of 32, using 10% of the training data for validation.

7. **Evaluation and Prediction** – The trained model predicts probabilities for the test set. Predictions are converted to binary class labels using a threshold of 0.5. The results are evaluated using a classification report that includes precision, recall, F1-score, and support for each class.

8. **Output to KNIME** – Class labels in the classification report are converted back to their original names (Safe Email and Phishing Email). The report is output as a KNIME data table (output_table_1) for visualization and further analysis.

This LSTM-based workflow enables robust phishing detection by leveraging sequential modelling capabilities, allowing the system to identify patterns in the wording, structure, and flow of phishing messages that may be missed by traditional machine learning approaches.
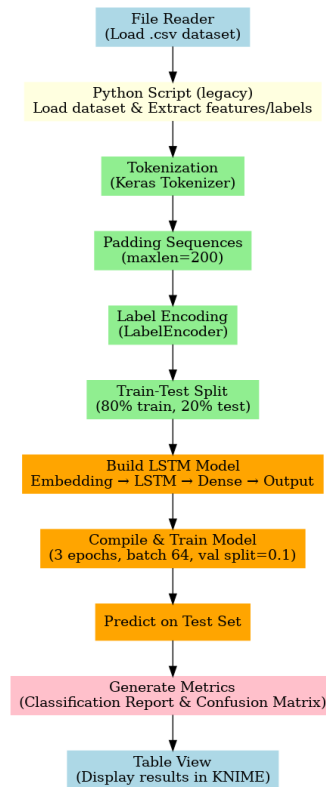

Figure 43: Step by step process of LSTM Learning using KNIME

5.4.3 Hybrid SVM + LSTM

In this study, we extend the hybrid phishing detection framework by employing an SVM model trained on URL features, combined with an LSTM model trained on email text content. This multimodal approach leverages the structural characteristics of URLs alongside the semantic patterns of email text, enhancing the system's ability to detect phishing attacks. The workflow is shown in figure below.
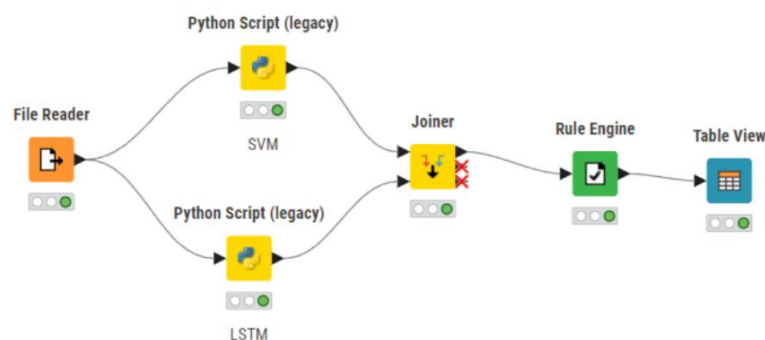
Figure 44: Hybrid phishing detection architecture combining URL-based SVM and email text-based LSTM models

Implementation Overview

The hybrid model is composed of three main components:

1. SVM Model:

   - Utilizes TF-IDF vectorized features extracted from the URL dataset.

   - The model is trained using an 80-20 train-test split.

   - Produces binary predictions for each sample, where 0 indicates a safe URL and 1 indicates a phishing URL.

   - The code implementation for the SVM workflow is provided in the **Appendices** section for reference.

2. LSTM Model:

   - Processes email text that has been cleaned, tokenized, and padded to a fixed length.

- The network architecture includes embedding layers, two stacked LSTM layers, and a dense output layer with sigmoid activation.
- Early stopping is applied during training to mitigate overfitting.
- Outputs binary predictions per email sample.
- The code for this LSTM follows the same implementation approach as the **individual LSTM model** described earlier, ensuring consistency across standalone and hybrid evaluations.

3. Hybrid Prediction:
- Combines the predictions from both SVM and LSTM models based on a simple decision rule:
  - If both models agree on a prediction, that prediction is adopted as the final output.
  - If the predictions differ, the system defaults to the LSTM's prediction, considering its higher accuracy.
- This combination strategy leverages agreement between models while relying on the stronger model when conflicts arise, aiming to improve overall detection accuracy.

Algorithm 1: Hybrid Prediction Combination

```
Input: svm_predictions, lstm_predictions
Output: hybrid_predictions


For each sample i:
  if svm_predictions[i] == lstm_predictions[i]:
    hybrid_predictions[i] = svm_predictions[i]
  else:
    hybrid_predictions[i] = lstm_predictions[i]


Return hybrid_predictions
```
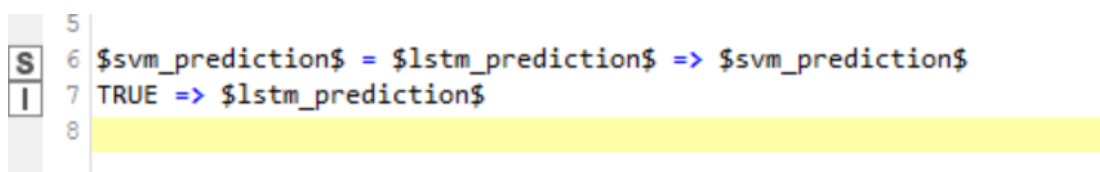
This algorithm combines the predictions from two distinct phishing detection models an SVM trained on URL features and an LSTM trained on email text content to generate a more robust final prediction.

- For each email sample, the algorithm compares the prediction from the SVM model (based on URL analysis) with that from the LSTM model (based on email text).

- If both models agree (both predict either safe or phishing), that consensus prediction is assigned as the final hybrid prediction. This agreement reflects high confidence.

- If the models disagree, the algorithm chooses the LSTM prediction because the LSTM model, leveraging the richer semantic information in email text, generally achieves higher accuracy than the URL-based SVM.

- By combining the strengths of structural URL features and contextual email text, this hybrid approach aims to improve phishing detection beyond what either model can achieve independently.

```
5
6  $svm_prediction$ = $lstm_prediction$ => $svm_prediction$
7  TRUE => $lstm_prediction$
8
```

Figure 45: KNIME Rule Engine Logic for Combining Predictions

The Rule Engine node combines SVM and LSTM predictions with this logic:
- If **both predictions are the same**, use that prediction.
- If they **disagree**, use the LSTM prediction.

This helps the hybrid model rely on agreement when possible and trust the more accurate LSTM when there's a conflict.

5.5   Summary

This section comprehensively outlined the methodology employed to evaluate the individual performances of the Support Vector Machine (SVM) and Long Short-Term Memory (LSTM) models. Each model was independently trained and tested on its respective dataset SVM utilizing URL-derived features, and LSTM processing email text data while maintaining consistent data partitioning and evaluation protocols to ensure fair comparison. The implementation leveraged analogous coding frameworks for both models to standardize the experimental procedure. Subsequently, a hybrid model was developed by integrating the predictions from both classifiers through a deterministic rule-based approach. This facilitated a rigorous comparison of classification accuracies between the standalone and combined models, thereby demonstrating the advantages of model integration for enhanced phishing detection efficacy.

CHAPTER 6: TESTING AND ANALYSIS

6.1   Introduction

This chapter details the testing procedures and analytical evaluation of the proposed hybrid (SVM+LSTM) model for social engineering detection. The primary objective is to assess the system's accuracy, reliability, and robustness in identifying potential social engineering threats. Testing was conducted using pre-processed datasets relevant to phishing and social engineering scenarios, applying defined experimental configurations within the KNIME Analytics Platform. The evaluation focuses on key performance metrics such as accuracy, precision, recall, and F1-score, enabling a comprehensive understanding of the model's effectiveness. Comparative analysis with existing detection techniques is also presented to highlight improvements achieved through the hybrid approach. This chapter not only validates the system's functionality but also identifies its limitations and potential areas for enhancement.

6.2 Result and Analysis

This section presents the evaluation results of the SVM, LSTM, and hybrid models, with a primary focus on accuracy. The results are summarized in tabular form to provide clear, comparative data for each model. Additionally, graphical visualizations such as bar charts and line graphs are used to illustrate performance trends and highlight the improvement achieved by the hybrid approach.

The tables display accuracy values for each model, while the graphs provide an intuitive visual comparison, emphasizing the hybrid model's enhanced detection capability. This combined presentation facilitates a comprehensive understanding of model effectiveness and supports the analysis of their relative strengths and weaknesses.

6.2.1 SVM Result
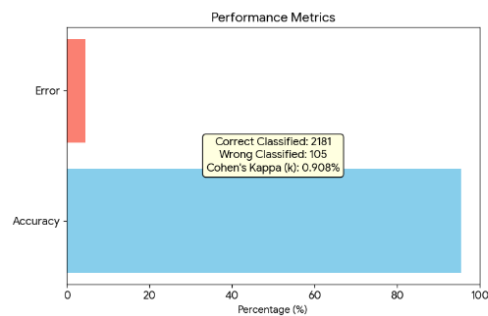
Figure 46: SVM's Accuracy in KNIME

Confusion Matrix

	Predicted: legitimate	Predicted: phishing
Actual: legitimate	1099	44
Actual: phishing	61	1082

Performance Metrics

Metric	Value
Correct classified	2,181
Wrong classified	105
Accuracy	95.407%
Error	4.593%
Cohen's kappa (k)	0.908%

Figure 47: Table SVM's Result

The Support Vector Machine (SVM) model achieved an accuracy of 95.4% in detecting phishing emails based on URL features. Figure 46 illustrates the key performance metrics of the SVM model, visually presenting its accuracy and other relevant measures. Figure 47 provides a detailed summary table that includes both the performance metrics and the confusion matrix, offering insight into the true positive, true negative, false positive, and false negative counts. The high accuracy indicates that the SVM model is effective in distinguishing phishing URLs from safe ones, although some misclassifications are still present as shown in the confusion matrix. This result demonstrates the capability of the SVM classifier in leveraging URL features for phishing detection.

6.2.2   LSTM Result

LSTM Classification Report

Rows: 5 | Columns: 5

Class ↑ String	precision Number (double)	recall Number (double)	f1-score Number (double)	support Number (double)
Phishing Email	0.937	0.968	0.952	1,477
Safe Email	0.979	0.957	0.968	2,250
accuracy	0.962	0.962	0.962	0.962
macro avg	0.958	0.963	0.96	3,727
weighted avg	0.962	0.962	0.962	3,727

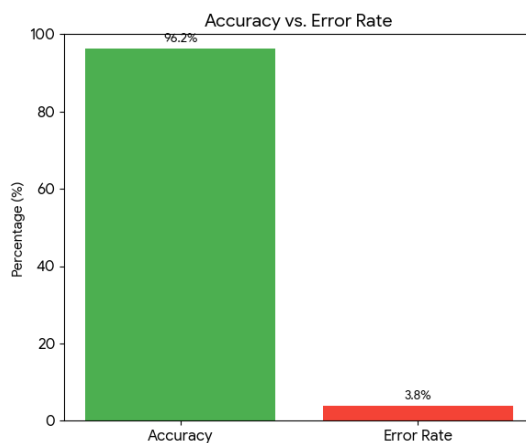**Figure 48: LSTM Classifier Report in KNIME**

Figure 49: LSTM's Accuracy in KNIME

Figure 48 presents the classification report of the LSTM model, summarizing its performance on the phishing detection task using email text data. Figure 49 illustrates the performance of the LSTM model in KNIME, achieving an accuracy of 96.2% with a corresponding error rate of only 3.8%. This result demonstrates the effectiveness of the LSTM in handling unstructured textual data, particularly in capturing sequential patterns within phishing email content, leading to high classification reliability.

6.2.3 Hybrid (SVM + LSTM) Result

Class	Precision	Recall	F1-score
Phishing	95.70%	90.00%	92.80%
Safe Email	97.80%	99.00%	98.40%
Overall			97.2% (Accuracy)

Figure 50: Hybrid performance metrics generate by Google Sheets Writer in KNIME
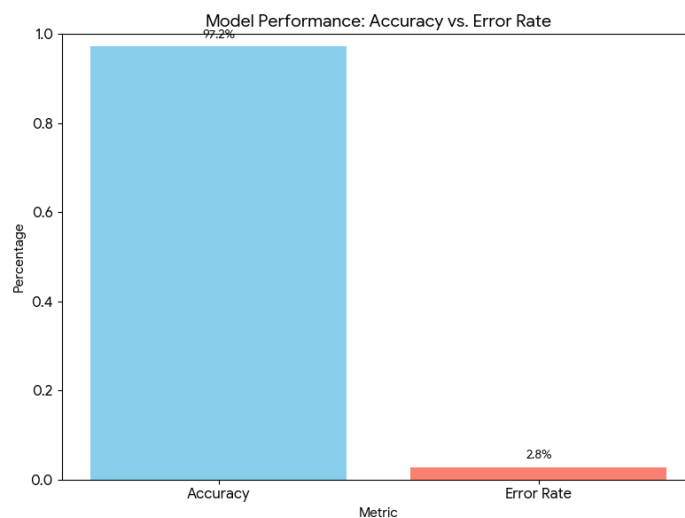

Figure 51: Hybrid performance accuracy vs error rate in KNIME

Figure 50 summarizes the classification performance of the hybrid phishing detection model through precision, recall, and F1-score metrics for each class Phishing and Safe Email along with the overall accuracy. The model achieves a precision of 95.7% and recall of 90.0% for phishing emails, indicating strong capability in correctly identifying phishing instances while maintaining a reasonable detection rate. For safe emails, precision and recall are notably higher, at 97.8% and 99.0% respectively, reflecting the model's effectiveness in accurately recognizing legitimate emails. The F1-scores for both classes reinforce the model's balanced and reliable performance, culminating in an overall accuracy of 97.2%.

Figure 51 provides a visual comparison between the hybrid model's accuracy and its error rate. The graph highlights the model's ability to maintain high accuracy while minimizing misclassification errors. This visual representation supports the conclusion that the hybrid approach, which integrates URL-based SVM and email text-based LSTM predictions, enhances detection accuracy beyond the capabilities of individual classifiers.

These results demonstrate the advantage of combining complementary features and models, validating the hybrid approach as a more effective solution for phishing detection.

6.2.4 Comparison Accuracy

Comparison Accuracy

Rows: 3 | Columns: 2

Model ↑ String	accuracy Number (double)
SVM	0.954
LSTM	0.962
Hybrid	0.972

Figure 52: Comparison Accuracy Table View in KNIME
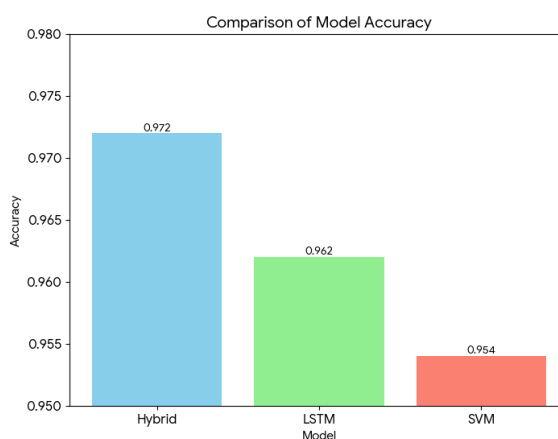
Figure 53: Comparison Accuracy Bar Chart View in KNIME

Figure 52 presents a summary table comparing the accuracy of the SVM, LSTM, and hybrid models. The SVM model achieves an accuracy of 95.4%, while the LSTM model attains a slightly higher accuracy of 96.2%. Notably, the hybrid model, which combines the strengths of both classifiers, demonstrates the highest accuracy at 97.2%.

Figure 53 complements this data with a bar chart visualization that clearly illustrates the performance differences among the three models. The chart highlights

the incremental improvement gained by integrating the SVM and LSTM models into a hybrid system.

This comparison confirms that leveraging a hybrid approach effectively enhances phishing detection accuracy by capitalizing on the complementary strengths of URL-based and text-based analysis, outperforming the individual models.

6.3   Summary

The evaluation results highlighted the performance of the three implemented approaches for phishing detection. The SVM classifier, trained on URL-based features, achieved an accuracy of 95.4% (error rate 4.6%), while the LSTM model, applied to email text data, produced a slightly higher accuracy of 96.2% (error rate 3.8%). Building on these outcomes, a hybrid model was developed by combining the predictions of SVM and LSTM through a rule-based decision strategy. The hybrid approach achieved the highest performance, with an accuracy of 97.2% and a reduced error rate of 2.8%. This reflects an improvement of 1.8% accuracy over the traditional machine learning model (SVM) and 1.0% over the deep learning model (LSTM), alongside a notable error rate reduction of 39% compared to SVM and 26% compared to LSTM. These findings confirm that integrating traditional machine learning with deep learning enhances detection effectiveness, yielding a more resilient phishing detection framework. The results also establish a strong foundation for further discussion of implications and potential improvements in the subsequent chapter.

CHAPTER 7: CONCLUSION

7.1   Introduction

This chapter provides the conclusion of the research project. It summarizes the key objectives, methodologies, and outcomes achieved throughout the study. Furthermore, the chapter highlights the project's contributions, discusses its limitations, and proposes directions for future work to strengthen phishing detection systems.

7.2   Project Summarization

The primary goal of this project was to design and implement a phishing detection system by utilizing both traditional machine learning and deep learning techniques, followed by the development of a hybrid approach that integrates the strengths of both models. The SVM classifier was trained on URL-based features, while the LSTM model was applied to pre-processed email text. Both models were individually tested before combining them in a hybrid framework to compare their performances.

The evaluation showed that the hybrid model achieved superior accuracy compared to the standalone models, thereby validating the hypothesis that integrating machine learning and deep learning improves phishing detection.

To better illustrate the outcome of this research, the key strengths and weaknesses of the project are summarized in **Table 16**.

Table 7.2.1: Strengths and Weaknesses of the Project

Strengths	Weaknesses
- Demonstrated successful integration of ML (SVM) and DL (LSTM) for phishing detection. - Hybrid model outperformed both standalone models in terms of accuracy. - Comparative evaluation provided a clear benchmark between ML, DL, and Hybrid approaches. - Workflow is reproducible and extendable for further research.	- High computational cost for training LSTM, limiting scalability. - Limited feature scope (mainly URL and text data; no image, attachment, or metadata features).

7.3   Project Contribution

This project contributes to the field of cybersecurity by demonstrating the effectiveness of integrating machine learning and deep learning techniques for phishing detection. Specifically, the combination of Support Vector Machine (SVM) and Long Short-Term Memory (LSTM) is shown to enhance detection performance compared to standalone models. A comparative evaluation was conducted on real phishing and legitimate email datasets, where the hybrid approach achieved higher accuracy and reduced error rates. Beyond performance improvements, this study also offers a reproducible workflow that can serve as a foundation for future research, particularly in advancing multi-modal phishing detection strategies.

7.4 Project Limitation

Despite its success, this project is not without limitations. The system primarily focused on text and URL-based features, while other critical phishing indicators such as sender behavior, email attachments, and metadata were not incorporated into the analysis. This restricts the model's ability to capture the full spectrum of phishing techniques commonly used in real-world scenarios. Furthermore, the training of deep learning models such as LSTM required substantial computational resources and longer processing times, which may pose challenges for scalability and real-time deployment in practical environments.

7.5   Future Works

Future research can address these limitations by incorporating additional data sources, such as network traffic patterns, email headers, and visual content like fake logos, to enable a more comprehensive and multi-layered phishing detection system. Furthermore, optimizing the deep learning architecture through advanced techniques such as attention mechanisms, transformer-based models, or federated learning could significantly improve both performance and scalability. Finally, deploying the hybrid model in a real-time environment would provide valuable insights into its practical effectiveness and resilience in detecting phishing attempts under real-world conditions.

7.6 Summary

In conclusion, the project successfully developed and tested a phishing detection framework using SVM, LSTM, and a hybrid integration of both. The hybrid model demonstrated the highest performance, surpassing traditional and deep learning models individually. Although certain limitations remain, the findings confirm that combining different approaches enhances phishing detection capabilities and opens opportunities for more robust and intelligent cybersecurity solutions in the future.

REFERENCES

Ali, S. (2024, December 1). The Role of AI in Social Engineering Attack Prevention: NLP-Based Solutions for Phishing and Scams. https://doi.org/10.13140/RG.2.2.22981.97765

H B, G., & H L, G. (2025). Detection of Phishing Activities Using Deep Learning Approaches. 2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS), 808–810. https://doi.org/10.1109/comsnets63942.2025.10885614

Lavie, O., Shabtai, A., & Katz, G. (2022). A Transferable and Automatic Tuning of Deep Reinforcement Learning for Cost Effective Phishing Detection. ArXiv.org. https://arxiv.org/abs/2209.09033

Patra, C., Giri, D., Maitra, T., & Kundu, B. (2024). A Comparative Study on Detecting Phishing URLs Leveraging Pre-trained BERT Variants. 1–6. https://doi.org/10.1109/cine63708.2024.10881521

Jishnu, K. S., & B Arthi. (2023). Enhanced Phishing URL Detection Using Leveraging BERT with Additional URL Feature Extraction. https://doi.org/10.1109/icirca57980.2023.10220647

Maci, A., Alessandro Santorsola, Coscia, A., & Iannacone, A. (2023). Unbalanced Web Phishing Classification through Deep Reinforcement Learning. Computers, 12(6), 118–118. https://doi.org/10.3390/computers12060118

Mittal, K., Gill, K. S., Chauhan, R., Singh, M., & Banerjee, D. (2023). Detection of Phishing Domain Using Logistic Regression Technique and Feature Extraction Using BERT Classification Model. 1–5. https://doi.org/10.1109/smartgencon60755.2023.10442975

Öznur Şengel. (2024). Analysis of Learning Techniques for Phishing Website Detection. 2022 Innovations in Intelligent Systems and Applications Conference (ASYU), 1–6. https://doi.org/10.1109/asyu62119.2024.10757053

Rawshon Ferdaws, & Majd, N. E. (2024). Phishing URL Detection Using Machine Learning and Deep Learning. https://doi.org/10.1109/aiiot61789.2024.10579005

Sawant, S., Rushabh Savakhande, Om Sankhe, & Tamboli, S. (2024). Phishing Detection by integrating Machine Learning and Deep Learning. https://doi.org/10.23919/indiacom61295.2024.10499100

Schmitt, M., & Flechais, I. (2024). Digital deception: generative artificial intelligence in social engineering and phishing. Artificial Intelligence Review, 57(12). https://doi.org/10.1007/s10462-024-10973-2

Smadi, S., Aslam, N., & Zhang, L. (2018). Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. Decision Support Systems, 107, 88–102. https://doi.org/10.1016/j.dss.2018.01.001

Sospeter, B. K., & Odoyo, W. (2024). AI-Based Phishing Attack Detection and Prevention Using Natural Language Processing (NLP). IC-ITECHS, 5(1), 597–602. https://doi.org/10.32664/ic-itechs.v5i1.1590

Subhash Ariyadasa, Fernando, S., & Fernando, S. (2023). SmartiPhish: a reinforcement learning-based intelligent anti-phishing solution to detect spoofed website attacks. International Journal of Information Security, 23(2), 1055–1076. https://doi.org/10.1007/s10207-023-00778-9

Villanueva, A., Atibagos, C., De Guzman, J., Dela Cruz, J. C., Rosales, M., & Francisco, R. (2022). Application of Natural Language Processing for Phishing Detection Using Machine and Deep Learning Models. 2022 International Conference on ICT for Smart Society (ICISS). https://doi.org/10.1109/iciss55894.2022.9915037

APPENDICES

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix

# Read input table from KNIME
df = input_table_1

# Clean text data - replace missing with empty string
texts = df['URL'].fillna('')
labels = df['Status']

# Split dataset: 70% train, 30% test (less training data)
X_train, X_test, y_train, y_test = train_test_split(
    texts, labels, test_size=0.3, random_state=42)

# TF-IDF vectorization with fewer features (2000 max features)
vectorizer = TfidfVectorizer(max_features=2000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Python Script for SVM 1

```python
# Train SVM with linear kernel and limited max iterations
svm = SVC(kernel='linear', max_iter=1000)
svm.fit(X_train_tfidf, y_train)

# Predict test set
y_pred = svm.predict(X_test_tfidf)

# Calculate accuracy
acc = accuracy_score(y_test, y_pred)
print(f"SVM accuracy: {acc:.4f}")

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Prepare confusion matrix as a DataFrame for KNIME output
cm_df = pd.DataFrame(
    cm,
    columns=['Predicted_Negative', 'Predicted_Positive'],
    index=['Actual_Negative', 'Actual_Positive']
)
```

Python Script for SVM 2

```python
# Prepare output table for KNIME with predictions
output_df = pd.DataFrame({
    'true_label': y_test,
    'svm_prediction': y_pred
})

# Send outputs back to KNIME:
#  - Output 1: predictions table
#  - Output 2: confusion matrix

output_table_1 = output_df
output_table_2 = cm_df.reset_index().rename(columns={'index': 'Actual_Label'})
```

Python Script for SVM 3