Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 10, No. 3, June, 2026

HyDNN: A Hybrid Deep Learning Approach for Phishing URL Detection

Divya J. Dsouza [1], Anisha P. Rodrigues [1*], Roshan Fernandes [2*], Vijaya Padmanabha [3], Mohamed S. Yoosuf [3]

[1] *Department of Computer Science and Engineering, NMAM Institute of Technology (NMAMIT), Nitte (Deemed to be University), Nitte 574 110, Karnataka, India.*

[2] *Department of Cyber Security, NMAM Institute of Technology (NMAMIT), Nitte (Deemed to be University), Nitte 574 110, Karnataka, India.*

[3] *Department of Mathematics and Computer Science, Modern College of Business and Science, Bawshar, Muscat 133, Oman.*

Abstract

Phishing is an online attack in which attackers trick victims into disclosing their sensitive information, such as credentials, financial portal pins, and OTPs, with the intention of identity or financial theft, jeopardizing reputations, and posing a risk to netizens. As the stakes are high, attackers invest considerable effort and time in committing organized crimes to steal valuable user information. The research carried out aims to detect phishing websites using machine learning and deep learning models. In this research, the classification models are applied to three different phishing website datasets, namely the Mendeley phishing dataset and the UCI dataset, which belong to binary classification, and one dataset that falls under multi-class classification. These datasets are publicly available for research. A custom data set is also prepared from recently available websites to reduce the potential bias in the already available data set. The reason for choosing a publicly available dataset is to validate and compare the results obtained from the custom dataset. To optimize the process, various feature selection techniques and dimensional reduction methods are applied, and a comparison of all approaches is summarized. Performance metrics are used for binary and multi-class classification, and then the outcomes obtained are summarized. The Random Forest model performs well with most feature selection techniques by achieving the best accuracy of 98.24% using the embedded feature selection approach for the Mendeley data set, 94.78% for the UCI data set, and 90.57% for the custom data set. Hence, using Random Forest as the base model, deep learning approaches, namely, CNN and LSTMs, are used to check the efficiency. This study shows that the proposed Hybrid Deep Neural Network approach, HyDNN, performs better, providing the best result with an accuracy of 98.87% for the Mendeley dataset, 97.63% for the UCI dataset, and 93.77% for the custom dataset.

Keywords:

Phishing;
Feature Selection;
Dimensional Reduction;
Machine Learning Classifiers;
Deep Learning;
HyDNN.

Article History:

Received:	30	May	2025
Revised:	26	April	2026
Accepted:	23	May	2026
Published:	01	June	2026

1- Introduction

With the advent of the Internet in the late 1990s, there has been an exponential growth in online data. Most users have shifted to storing and accessing their data digitally. Many tasks are now performed paperlessly, and financial transactions are just a click away. The Internet is primarily used to share knowledge, sell products, and create social communities. However, alongside technological advancements come associated challenges. Attackers continually devise new ways to deceive netizens. There has been a significant increase in the number of websites that interact with users, including those belonging to retail, product- and service-based industries, and manufacturing sectors that contribute to economic growth.

* **CONTACT**: anishapr@nitte.edu.in; roshan_nmamit@nitte.edu.in

DOI: https://doi.org/10.28991/ESJ-2026-010-03-033

However, not all websites are genuine. Fake websites are deliberately created by attackers with the sole intention of deceiving users; these are commonly referred to as phishing websites [1].

Phishing is a form of social engineering in which users are tricked into revealing sensitive information such as social security numbers, credit card details, addresses, and other confidential data. There are several types of phishing attacks, with the most prominent being email phishing, vishing, spear phishing, pop-up phishing, smishing, whaling, malware phishing, angler phishing, and image-based phishing. According to the 2023 phishing threat report by Cloudflare, 35.6% of total threats originate from malicious links. It has also been reported that multi-channel phishing attacks may begin with seemingly benign links.

The ten most frequently impersonated organizations include the World Health Organization (WHO), Microsoft, Salesforce, Google, SpaceX, Apple, YouTube, Amazon, T-Mobile, and Mastercard [2]. To mitigate fraudulent activities carried out through phishing, several countermeasures are employed, such as blacklist-based tools, browser filtering mechanisms, anti-phishing add-ons, updated firewalls, security awareness workshops, phishing simulation platforms, and other preventive techniques [3].

1-1- Motivation for the Research

According to recent studies and reports by the Internet Crime Complaint Center (IC3) [4], cybercrime "particularly phishing attacks" is increasing at an alarming rate, with scammers stealing a record $16.6 billion in 2024, a 33% increase compared to the previous year, and phishing among the most frequently reported offenses. Axios Additionally, global cybersecurity firms reported that nearly 900 million phishing attempts were blocked in 2024, marking a 26 % rise over 2023, and newer AI‑ enabled phishing campaigns continue to evolve in sophistication. kaspersky.co.in Industry trend reports also show over 1.13 million phishing attacks in Q2 2025, representing a significant year‑ over‑ year increase and reinforcing the persistent growth of these threats. docs.apwg.org

Most existing literature focuses on addressing phishing detection using traditional machine learning models. However, only a limited number of recent studies have explored the application of deep learning models to handle high‑ dimensional data, and those that do often employ individual architectures in isolation. The primary objective of this research is to leverage the combined advantages of multiple deep learning models on time‑ series datasets and evaluate their efficiency. Experiments were conducted on several thousand phishing URLs, comprising both custom‑ built datasets and publicly available data. The results demonstrate modest improvements compared to those achieved using a single deep learning model alone. Given the rapid growth and increasing sophistication of phishing attacks, even marginal improvements in detection accuracy provide strong motivation to adopt hybrid deep learning models for high‑ dimensional datasets.

This research therefore focuses on optimizing both deep learning and machine learning approaches for phishing website detection, while enhancing performance through the incorporation of feature selection and dimensionality reduction techniques. Feature selection plays a crucial role in model development [5], as performance depends on the relevance and accuracy of the selected features. It is a common misconception that increasing the number of features necessarily improves model accuracy. Instead, models should be evaluated using optimal parameters to balance accuracy with space and time complexity. A comparative analysis is conducted using standard performance metrics relevant to classification tasks.

The research objectives are framed as follows:

- To prepare a custom dataset by collecting benign and malicious URLs from recently updated sources and preprocessing the URLs by extracting the most relevant attributes. The steps involved in feature extraction from URLs are explained using pseudocode in Section 3.4.
- To compare, fine-tune, and evaluate various attribute selection and dimensionality reduction techniques, and to present a summarized report of the results (refer to Section 4.9). This evaluation is conducted using three publicly available phishing datasets—namely, the Mendeley, UCI, and Multi-Classifier datasets—along with the custom dataset, in order to minimize potential evaluation bias.
- To compare machine learning and deep learning models for phishing website detection, supported by appropriate justification.

iv) To propose a hybrid deep learning approach (explained in Section 4.10) for feature extraction and bidirectional data processing "one direction processing the input forward and the other backward" thereby making the model suitable for sequence-based prediction tasks.

2- Literature Reviews

This section briefly reviews related work in the domain of phishing URL detection, focusing on the use of various techniques and models, including machine learning and deep learning approaches. A comprehensive review of the existing literature on phishing detection, feature selection techniques, and machine learning and deep learning

approaches is made. Previous studies have explored various traditional and ensemble-based classifiers, as well as deep learning models, to detect phishing attacks. However, several gaps remain in the literature, including limited comparative evaluation across different datasets and learning modes, reliance on handcrafted features, and the lack of robust hybrid deep learning models that can capture both spatial and sequential dependencies. To address these shortcomings, the present study proposes the HyDNN model, a hybrid architecture combining CNN and BiLSTM layers, designed to enhance feature learning, improve classification accuracy, and provide a scalable solution for phishing detection.

2-1- Related Work on Phishing URL Detection

Patil & Patil [6] describe a malicious web page detection approach based on the static analysis of URLs. In their study, only static URL features were considered, and machine learning classification models were employed for detection. In Alsenani et al. [7], a feature selection–based technique known as Particle Swarm Optimization (PSO) was applied to iteratively optimize model performance. Swarm optimization is inspired by biological behaviors such as animal swarming and herding, and one of the key advantages of PSO is that it requires relatively few parameters.

In Qi et al. [8], two ensemble algorithms—the Fisher Markov–based ensemble phishing detection model (FMPED) and the Fisher Markov–Markov–based ensemble phishing detection model (FMMPED)—were proposed for phishing email detection. Initially, benign emails were eliminated from overlapping regions, followed by undersampling of the remaining benign samples. In the final stage, both benign and phishing emails were trained using the ensemble algorithms. The experimental results demonstrated an accuracy of 99.45%, an F1-score of 99.45%, an AUC of approximately 98.28%, and a G-mean of 98.28%.

Tudosi et al. [9] highlight the importance of distributed firewall architectures for phishing email detection by integrating the detection framework into firewall setups. As discussed by Wang [10], genetic algorithms are preferred over traditional feature selection methods due to their superior accuracy in phishing website detection. An F1-score of 97% was achieved using the Extended Compact Genetic Algorithm (ECGA), an improved variant of genetic algorithms. In Alani & Tawfik [11], a cloud-based machine learning framework for phishing URL detection, termed *PhishNot*, was proposed. Among the evaluated classifiers, the Random Forest algorithm achieved the best performance with an accuracy of 97%. Maennel & Matulevicius [12] emphasize the role of the Selenium WebDriver in heuristic-based phishing detection and introduced a tool called *SeleniumPhishGuard* for identifying phishing login pages. The module was developed independently and can potentially be integrated as a browser plug-in to detect phishing URLs in real time.

Furthermore, Dadkhah et al. [13] proposes a hybrid approach that eliminates irrelevant features and incorporates techniques for searching web pages through search engines. This approach focuses on identifying journal phishing attacks and phishing pages embedded within legitimate websites.

2-2- Machine Learning Approaches

Chandrasegar & Viswanathan [14] employed Decision Tree and Random Forest classifiers with various feature splits to evaluate dataset attributes and compute classification accuracy. The results showed an accuracy of 84.8% when using Random Forest with five features, which increased to 96.3% when six features were considered; however, including all features resulted in a reduced accuracy of 93.20%.

Calzarossa et al. [15], proposed an explainable machine learning–based framework to identify and prioritize the most influential features for phishing website detection. Karim et al. [16] introduced a hybrid LSD model for phishing URL detection, which combines three machine learning techniques—linear regression, support vector machines, and decision trees—using both hard and soft computing approaches. The performance of the proposed model was compared with several conventional machine learning classifiers, including decision trees, gradient boosting, naïve Bayes, linear regression, random forest, k-nearest neighbors, and support vector machines.

Pandey & Mishra [17] presented a novel technique called *Phish-Sight*, which detects phishing websites based on the dominant colors present on web pages. In Basit et al. [18], a new ensemble-based approach for phishing attack detection was proposed using machine learning models such as k-nearest neighbors (KNN), artificial neural networks, decision trees, and random forests. Adewole et al. [19], introduced a rule-based hybrid model utilizing the JRip algorithm and Projective Adaptive Resonance Theory to generate rule sets. When evaluated on two publicly available datasets, the model achieved accuracies of 94.53% and 99.08%, respectively. Jennifer Dsouza et al. [20] compared different execution modes “offline, batch, and incremental” for phishing datasets. This comparison provided insights into how dynamic datasets perform under various learning modes, thereby assisting analysts in informed model selection.

Finally, Soosai Anandaraj et al. [21] proposed a novel Phishing Detection System based on Hybrid Machine Learning (PDS-HML) for classifying malicious URLs. The proposed approach demonstrated the ability to identify phishing patterns even in the presence of attacker obfuscation and evasion techniques.

2-3- Deep Learning Approaches

Catal et al. [22], explored various deep learning approaches for identifying phishing threats. Do et al. [23] discussed current challenges in phishing detection and examined deep learning techniques that can be employed to mitigate such threats, highlighting the advantages of Ensemble Deep Learning (EDL) and Deep Reinforcement Learning (DRL).

Ozcan et al. [24], focused on hybrid techniques by combining Deep Neural Networks (DNNs) with Long Short-Term Memory (LSTM) networks, achieving an improved accuracy of approximately 98.79% on the EBU 2017 dataset. Prabakaran et al. [25] proposed an enhanced deep learning framework using Variational Autoencoders (VAEs) in conjunction with Deep Neural Networks, attaining an accuracy of 97.45% with a response time of 1.9 seconds.

Nguyen et al. [26] introduced a deep learning–based phishing detection approach utilizing hierarchical LSTM architectures along with supervised attention mechanisms to identify phishing content. Ariyadasa et al. [27], proposed the *Smart-Phish* framework, which integrates deep learning with reinforcement learning techniques to counter phishing attacks.

Bountakas & Xenakis [28] presented a novel hybrid ensemble-based method for phishing email detection, employing stacking models with soft voting, and achieved an F1-score of 99.42%. Al-Sarem et al. [29] further investigated the optimization and stacking of ensemble models for phishing website detection. In Feng et al. [30], a novel neural network–based classification approach was introduced, designed using a high-risk minimization principle to achieve high accuracy and strong generalization capabilities. Sahingoz et al. [31] evaluated five deep learning models—artificial neural networks, convolutional neural networks, attention networks, recurrent neural networks, and bidirectional recurrent neural networks—using custom datasets collected from *phishtank.com* and *commoncrawl.org*. The results demonstrated that deep learning approaches outperform traditional machine learning methods.

Kritika [32] presented a comprehensive literature review on phishing URL detection using deep learning techniques, emphasizing defenses against multimodal phishing strategies and zero-day attacks. In Korkmaz [33], a two-stage hybrid phishing detection system combining deep learning and content analysis was proposed to reduce false positives, achieving an accuracy of 98.37% on a high-risk dataset.

Table 1 provides a summarized overview of related work on phishing website detection, including the datasets used, techniques and methods applied, key features considered, and identified limitations.

Table 1. Related works to Phishing Website Detection

Author / Year	Dataset	Technique	Key Points	Limitations
Zhou et al. [34]	Alexa Website, PhishTank	LightGBM	Domain name features perform better than single features	Manual feature selection
Bahaghighat et al. [35]	Phishing Websites Dataset	LR, KNN, NB, RF, SVM, XGBoost	Constant attributes were dropped	Not tested in real-time
Prabakaran et al. [25]	Kaggle, ISCX-URL-2016	VAE, DNN	Automated extraction of features from URLs	Higher false positive rate
Abdul Samad et al. [36]	UCI Repository, Mendeley	ML Classifiers	Data balancing and hyperparameter tuning	Cloud threats not covered
Elsadig et al. [37]	Kaggle Phishing URLs	BERT, NLP-based Deep Learning	Optimized feature selection	Does not support dynamic feature selection
Ramana et al. [38]	UCI Dataset	Ensemble of RF, DT, XGBoost	Effective use of ensemble learning	Limited use of feature selection methods
Ozcan et al. [24]	Ebbu 2017, Custom Dataset	Hybrid DNN–LSTM	NLP features with character embedding	Not tested on noisy instances
Karim et al. [16]	Kaggle	Hybrid LSD-based Models	Ensemble of ML models	Tested on only one dataset
Pandey & Mishra [17]	OpenPhish, Alexa	RF, LR, DT, SVM, NB	Uses website color palette	Depends on screenshot clarity
Basit et al. [18]	Mendeley	Reinforcement Deep Learning	Knowledge Acquisition Process (KAP)	Vulnerable to dynamic attacks
Bountakas & Xenakis [28]	Custom Dataset	Stacking, Soft Voting	Innovative evaluation guidelines	Imbalanced datasets
Mourtaji et al. [39]	Custom Dataset	CART, CNN	Logical rule-based approach	Unreliable in real-time scenarios

2-4- Literature Gap

A comprehensive review of existing literature reveals several research gaps that motivate the present study. Most feature selection–based approaches for phishing detection focus primarily on static datasets, where data are collected in advance and processed offline [6, 8, 34, 40, 41]. In contrast, only a limited number of studies address dynamic or evolving datasets [22, 28, 36]. Consequently, there is a lack of comparative analysis of feature selection techniques that are effective across both static and dynamic phishing datasets. This research addresses this gap by evaluating and comparing feature selection methods using customized static and dynamic datasets, with the aim of identifying the most influential features for phishing URL detection.

Additionally, existing studies often emphasize *what* features are selected, while offering limited justification regarding *why* certain features are deemed important and others are discarded. This work explicitly investigates the feature selection process to provide reasoning behind feature relevance in URL evaluation. The obtained results are systematically compared with state-of-the-art methods to facilitate a clearer understanding of the most suitable feature selection techniques for phishing URL detection.

Furthermore, a significant portion of the literature relies on traditional machine learning approaches for phishing URL detection [14–18]. Although effective, there has been relatively limited exploration of ensemble and hybrid deep learning architectures, such as CNN–LSTM, CNN–LSTM–RNN, LSTM–RNN, bidirectional LSTM, and deep reinforcement learning–based models. Hybrid deep learning models integrate multiple neural network architectures to exploit their complementary strengths.

In this context, the proposed Hybrid Deep Neural Network (HyDNN) combines Convolutional Neural Networks (CNNs) with Bidirectional Long Short-Term Memory (BiLSTM) networks. CNNs are effective in extracting local and structural patterns from URLs, while BiLSTMs capture sequential dependencies in both forward and backward directions. This bidirectional context modeling is particularly advantageous for phishing URL detection, where critical cues may be distributed across different parts of the URL. While HyDNN models are computationally more complex and resource-intensive than simpler CNN+RNN architectures, they offer superior performance in capturing complex URL patterns. In scenarios where computational efficiency or inference speed is a priority, simpler models may be preferred despite marginal performance trade-offs.

Hence, this study adopts an ensemble deep learning approach using the proposed HyDNN model, which leverages the complementary strengths of multiple deep learning architectures to improve the accuracy and robustness of phishing website detection. By integrating convolutional neural networks (CNN) for feature extraction with bidirectional long short-term memory (BiLSTM) layers for sequence modeling, HyDNN is able to capture both spatial and temporal patterns in URL features, providing a more comprehensive representation of the data.

The primary contributions of this work are twofold. First, a comprehensive evaluation of feature selection and dimensionality reduction techniques is conducted to identify the most informative features for phishing detection, as detailed in Section 4.9. This ensures that the models are trained on the most relevant features, improving both efficiency and predictive performance. Second, a hybrid deep learning framework, HyDNN, is implemented, as described in Section 4.10, which optimizes the detection of phishing URLs by combining CNN and BiLSTM layers in a unified architecture. Overall, this approach demonstrates improved performance compared to traditional machine learning classifiers and single deep learning models, highlighting the effectiveness of ensemble hybrid modeling for cybersecurity applications.

3- Methodology

The following section presents the methodology adopted in this study, detailing the model architecture, learning modes, and experimental setup. It describes the procedures employed for data preprocessing, training, and evaluation of the proposed approach.

3-1- Background Study of the Proposed Model

Phishing attacks commonly occur through malicious URL links delivered via emails, websites, pop-up chats, instant messaging tools, phone calls, and other channels. The primary targets are users who are unaware of the risks associated with clicking these malicious URLs. The motivation for this study, therefore, is to mitigate phishing attacks by distinguishing between legitimate and phishing URLs (refer Section 3.2) and by developing methods to quickly and accurately identify fraudulent URLs to protect sensitive information.

This paper introduces a novel approach for phishing detection that emphasizes feature extraction through the selection of relevant URL attributes, which are then input into the proposed hybrid deep learning model. This strategy enables a more precise analysis of phishing attacks, offering insights into effective countermeasures and helping to prevent potential security breaches.

3-2- URL Description

Since Google supports URLs as defined by RFC 3986, the If–Else rules are mapped according to this standard and applied to the extracted features to assess the credibility of each selected attribute. After shortlisting the relevant parameters, the dataset is cleaned to remove missing or null values. The top retrieved features are then used for the study. A word cloud analysis is performed on the URLs to identify the most prominent words appearing within them. Additionally, URLs are verified using popular tools such as VirusTotal and Phishing Link Checker by EasyDmarc to determine whether the websites are legitimate. This curated dataset is then utilized with the optimized model for performance comparison.

The top features selected for the study include: URL length, domain of the URL, counts of letters, digits, and special characters, use of URL shortening services, presence of abnormal URL patterns, HTTPS security, presence of an IP address in the URL, and the target class. All conditions and feature mappings are based on the specifications outlined in RFC 3986.

3-3- RFC 3986

RFC 3986 defines a Uniform Resource Identifier (URI) and provides the generic syntax that specifies the grammar of URIs. The following subsection describes the key components of a URI:

3-3-1- Scheme Name

The description for assigning identifiers inside that scheme is referred to as the scheme name.

3-3-2- Authority component

An authority component is optional. It is the naming authority that is responsible for the governance of the namespace.

3-3-3- Optional Path

It is the hierarchy of components shown in decreasing order and is optional.

3-3-4- Optional Query

It is used to identify a resource. It is also called a fragment identifier and is optional.

The above-listed components are explained with an example: URI starts with a scheme name followed by a colon ":", which is then followed by a double slash: "//" if the authority component exists. Next comes the path, and components inside the path are separated by "/". If there is a query present, a question mark "?" follows the path. If there is a fragment ID, then a pound sign is given. "#".

Hence, the full URI would appear as:

"scheme://authorship/path-component1/path-component2? query#fragmentID"

Example 1:

"http://www.sampleurl.com/q1?operation= searchRetrieve&maxRecords=20"

The above sample URL can be split up as:

Scheme Name - "http";

Authority Component - "www.sampleurl.com";

Path - "q1";

Query - "operation=searchRetrieve&maximumRecords=20".

Example 2:

"mailto:user@example.com"

Scheme: mailto

Authority Component: Not present in this URI.

Path: user@example.com

Query: Not present in this URI

Example 3:

"file:/path/to/file.txt"

Scheme: file

Authority Component: Not present in this URI (if it were present, it would follow the double slashes //).

Path: /path/to/file.txt

Query: Not present in this URI.

The slash ("/"), question mark ("?"), and hash sign ("#") are reserved characters. The extracted features are from different aspects of the URL as shown in Figure 1. Since the URL is used to access web pages, a phisher will also be able to gain control over the subdomain portions of the URL. The phisher can then set any values to them with the sole intention of gaining sensitive information from users. The path and other file components can also be compromised by the phisher. Security defenders struggle to detect phishing domains because of the distinctive elements in the website's domain name. This refers to the challenges faced by cybersecurity professionals in identifying malicious websites designed for phishing attacks. Phishing domains often include:

- Unique or Misleading Domain Names: These domains may look similar to legitimate ones but contain subtle differences, such as typosquatting (e.g., "g00gle.com" instead of "google.com"), or using different top-level domains (TLDs) like ".net" instead of ".com".
- Obscured Intent: Phishing sites often use unique, unfamiliar, or obscure domain names that make it hard to distinguish them from legitimate sites at first glance.
- Short-lived Nature: Phishing domains can be quickly registered and abandoned, making it challenging to track and block them in time.

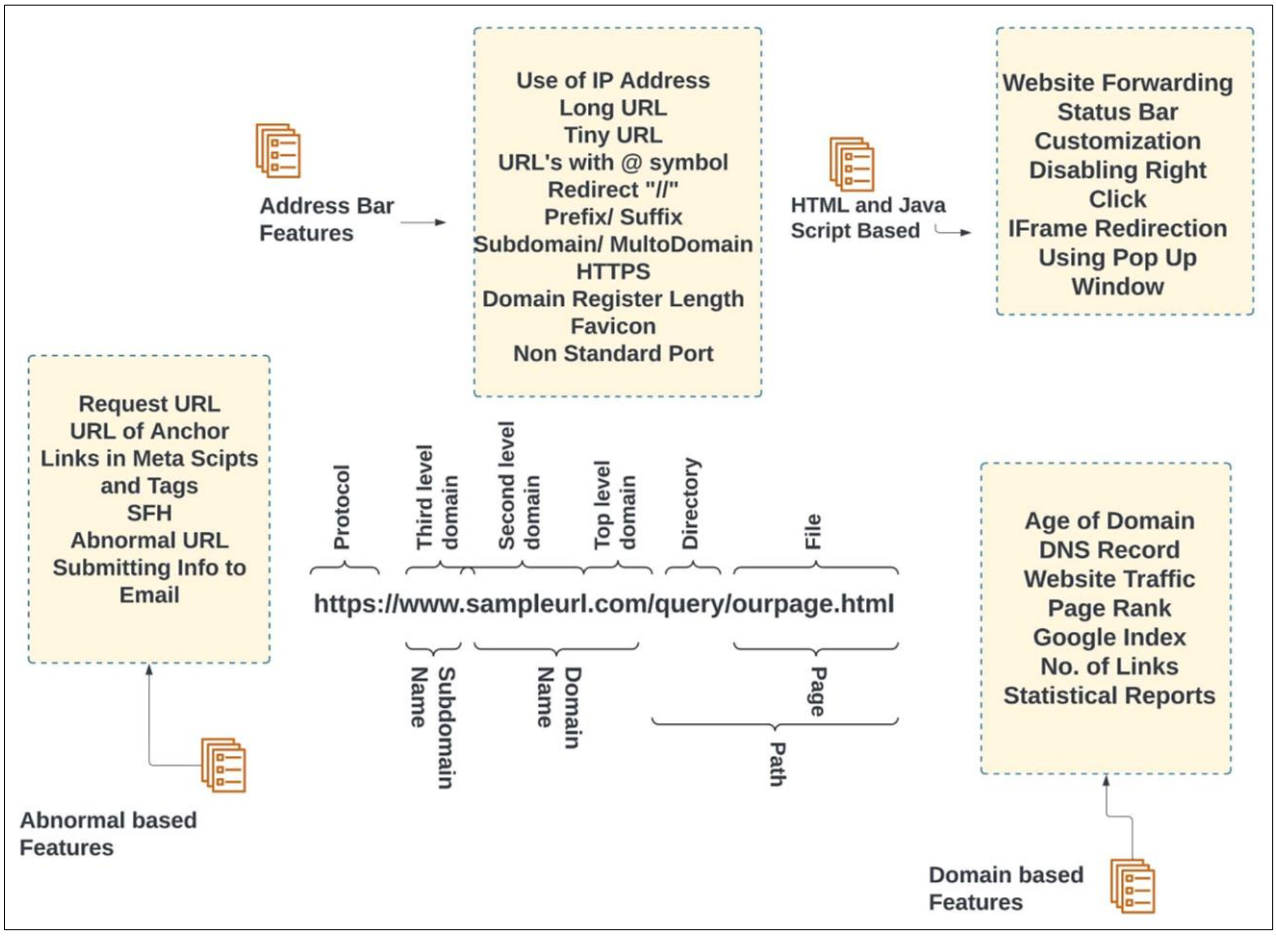


Figure 1. Typical URL Structure

Due to these factors, security systems and defenders may struggle to identify and block phishing domains before they cause harm. Hence, it is crucial to detect if the domain is compromised or fraudulent prior to accessing it.

URLs can be inspected in many ways. For this study, we have divided the URL into four types:

i) Features based on Address Bar

ii) Abnormal-Based Features

iii) Features based on HTML and JavaScript

iv) Features based on Domain

The following sections describe features in detail:

The Address Bar features describe the IP address in detail. For instance, the below feature is interpreted as follows:

a) Using the IP Address

If an IP address is used as an alternative to the domain name in the URL, for instance, "http://121.23.45.11/phish.html," then there is a possibility of stealing the personal information of users through this IP address. Sometimes, there is a possibility of converting these IP addresses to hexadecimal code, like in the below-given link: "http://0x48.0xBC.0xAC.0x52/2/gpal.ac/indexing.html".

Hence, the rule that can be applied is as below:

If The Domain Part contains an IP Address → Phishing

Otherwise→ Legitimate

Similarly, the other features are also evaluated.

Abnormal-based features highlight the details that are indirectly related to the URLs. HTML and JavaScript-based features indicate the details that can be obtained based on web page details. Domain-based features indicate the knowledge the user has regarding the data in detail.

Figure 1 illustrates the categorization of features extracted for phishing URL detection, grouped into four major classes: address-based, abnormal-based, HTML–JavaScript-based, and domain-related features

3-4- Legitimate URL Detection Algorithm

Input

• URL: The URL to be checked for legitimacy.

Output

• True if the URL is legitimate, False otherwise.

Algorithm Steps

1. Check if the URL starts with "https://".
2. Extract the domain from the URL.
3. Check if the domain's IP address is not blacklisted.
4. Check if the WHOIS information of the domain is not suspicious.
5. Check if the domain age is above a certain threshold.
6. Check if the URL path is similar to paths of legitimate URLs.
7. Check if the URL path adheres to the if-else rules applied to Address Bar-based, Abnormal-based, HTML and JavaScript and Domain-based features.
8. If all checks pass, return True indicating the URL is legitimate. Otherwise, return False.

Pseudocode

```
Function isLegitimateURL(URL):
if URL starts with "https://":
   domain = extractDomain(URL)
if not checkDomainIP(domain):
   return False
if not checkDomainWHOIS(domain):
   return False
if not checkDomainAge(domain):
   return False
if not checkPathSimilarity(URL):
   return False
return True
```

```
return False
Function isValidURL(url):
// Initialize result as False
result = False
Try:
Step 1: Basic URL Structure Check
If not isWellFormedURL(url):
Raise Exception("Malformed URL")

Step 2: DNS Resolution
     domain = extractDomain(url)
Try:
ipAddress = resolveDNS(domain)
Catch DNSResolutionError:
Log("DNS resolution failed
for domain: " + domain)
Raise Exception("DNS resolution failed")
Step 3: Check Against Blacklist
If isBlacklisted(domain) or isBlacklisted(ipAddress):
Raise Exception("URL is blacklisted")

Step 4: Check Against Whitelist
If isWhitelisted(domain):
result = True
Return result

Step 5: Analyze URL for Obfuscation Techniques
If containsObfuscation(url):
Raise Exception("URL contains obfuscation")

Step 6: Check for Dynamic DNS
      If isDynamicDNS(domain):
Raise Exception("URL uses dynamic DNS service")

Step 7: Heuristic Analysis
If passesHeuristicChecks(url):
result = True
Else:
Raise Exception("Heuristic analysis failed")

Catch Exception as e:
Log("Error in URL validity check: " + e.message)
Return result

Helper Functions
Function isWellFormedURL(url):
Implement basic structure validation (e.g., scheme, domain, path)
Return True or False
```

```
Function extractDomain(url):
Extract domain from the URL
Return domain

Function resolveDNS(domain):
Resolve the domain to an IP address
Return ipAddress or Raise
DNSResolutionError

Function isBlacklisted(entity):
Check if the domain or IP is in the blacklist
Return True or False

Function isWhitelisted(domain):
Check if the domain is in the whitelist
Return True or False

Function containsObfuscation(url):
Check for common obfuscation techniques(e.g., hexadecimal encoding)
Return True or False

Function isDynamicDNS(domain):
Check if the domain is associated with known dynamic DNS providers
Return True or False

Function passesHeuristicChecks(url):
Perform heuristic analysis (e.g., length of URL, unusual characters)
Return True or False
```

For example, let's consider the URL "https://www.example.com". Applying the algorithm:

Step 1: The URL starts with "https://" hence secure, proceed to step 2.

Step 2: The domain extracted is "www.example.com".

Step 3: checkDomainIP checks if the domain's IP address is not blacklisted and returns True.

Step 4: checkDomainWHOIS checks if the WHOIS information of the domain is not suspicious and returns True.

Step 5: checkDomainAge checks if the domain age, length of URL is above a certain threshold and returns True.

Step 6: checkPathSimilarity checks if the URL path is similar to paths of legitimate URLs and returns True.

Step 7: checkFeatureTypes checks if the features in the URL path maps to features listed in Figure 1 for legitimate URLs and returns True.

Step 8: Since all steps returned True, the algorithm concludes that the URL is legitimate.

3-5- Experimental Setup

The experiments were conducted using Google Colab. The data analysis and machine learning libraries utilized include Keras, NumPy, Pandas, scikit-learn, Matplotlib, Seaborn, and TensorFlow. The datasets were processed and analyzed using both machine learning and deep learning classification algorithms, applied to all extracted features. The experimental setup is structured into the following subphases to provide a detailed overview of the study's methodology:

3-5-1- Simulation Design

Figure 2 illustrates the simulation design of the study, which is divided into three main phases.

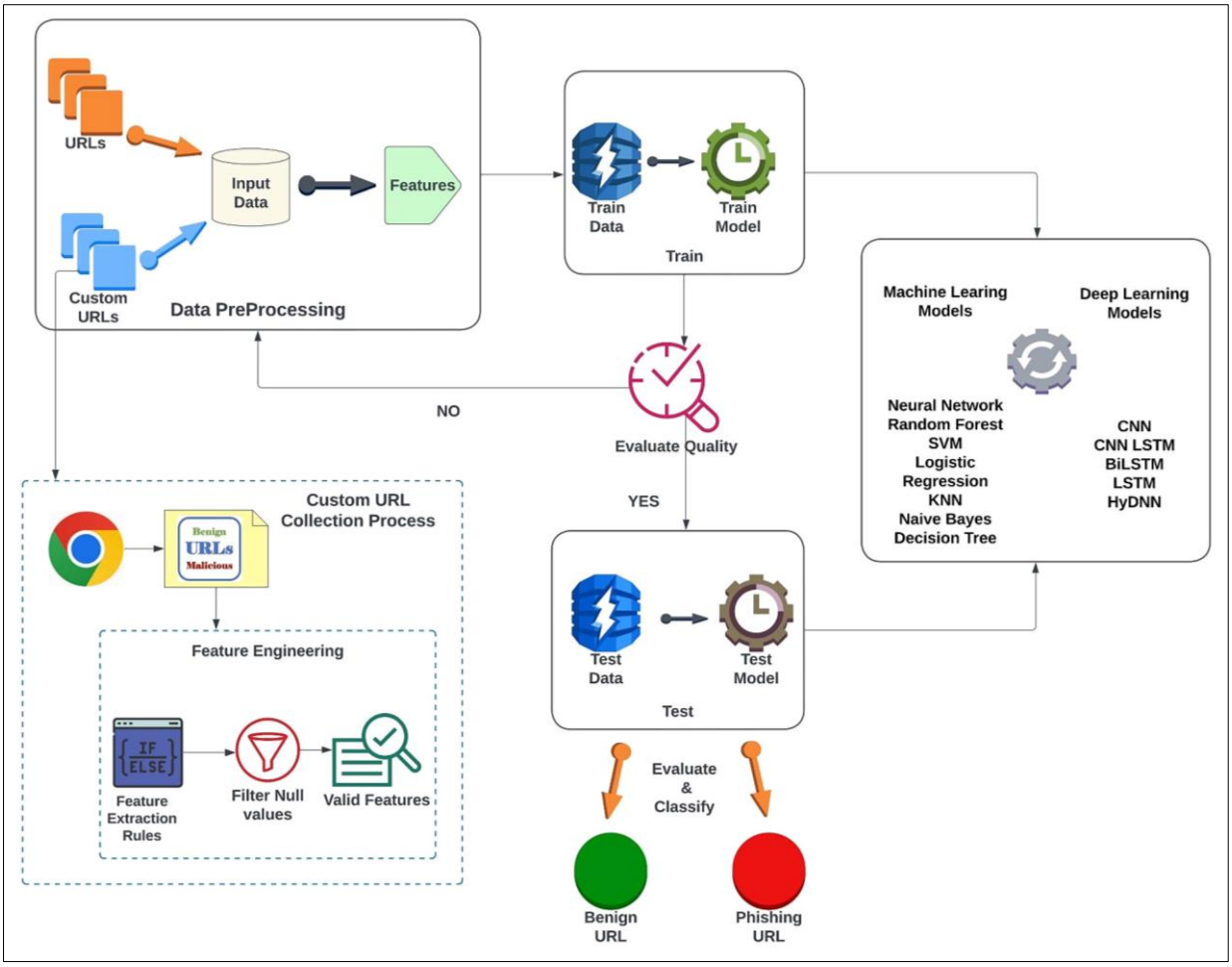


Figure 2. Architectural Diagram of Phishing Websites Detection

Phase 1 – Data Collection: The URLs, comprising both genuine and malicious samples, are collected as input data. The datasets are cleaned to remove missing or null values. The credibility of each URL is verified using the pseudocode, ensuring that only legitimate or confirmed phishing URLs are included.

Phase 2 – Data Preprocessing and Feature Selection: The collected data undergoes preprocessing, which includes handling missing values, balancing the dataset if necessary, and standardizing or normalizing the features. Various feature selection and dimensionality reduction techniques are applied to optimize the dataset, including Recursive Feature Elimination (RFE), Filter-Based Feature Selection (FBFS), embedded feature selection, Principal Component Analysis (PCA), auto-encoders, Random Forest Feature Importance Score (RFFS), and heat-map correlation (HMC) analysis. The dataset is then split into training and testing sets in a 70:30 ratio.

Phase 3 – Classification and Evaluation: Machine learning and deep learning classification algorithms are applied to the optimized datasets. The deep learning models are executed on a T4 GPU in Google Colab for efficient computation.

The overall architectural workflow comprises several components:

- *Data Preprocessing*: Collection of URLs, cleaning missing entries, balancing the dataset, and standardization /normalization.
- *Feature Extraction*: Custom URLs are analyzed to extract address-based, abnormal-based, domain-based, HTML-based, and JavaScript-based features. Each feature is verified against RFC 3986 syntax, and rules are applied to assess their credibility.
- *Model Training and Evaluation*: The processed features are fed into machine learning and hybrid deep learning models for phishing detection, with performance metrics evaluated on the test data.

3-5-2- Data

This study utilizes three publicly available phishing datasets and one custom-built dataset. The first dataset, Dataset 1 (Mendeley Dataset), consists of 5,000 phishing and 5,000 legitimate URLs collected between 2015 and 2017, with a total of 48 features. These URLs were gathered from various sources and real-time websites [41]. The target variable is binary, where legitimate URLs are labeled as 0 and phishing URLs as 1.

Dataset 2 (UCI Dataset) is obtained from the UCI Machine Learning Repository [42] and contains 30 features. The target labels are binary, with -1 representing phishing websites and 1 representing legitimate URLs. This dataset comprises a total of 11,055 URLs, including 6,157 legitimate and 4,898 phishing websites.

The third dataset, Dataset 3 (Multi-Class Dataset), is a multi-class dataset with target labels of 1 for legitimate, -1 for phishing, and 0 for suspicious URLs. It was obtained from Frank [43] and used for comparative analysis in this study. The dataset includes 548 legitimate URLs, 103 suspicious URLs, and 702 phishing URLs, with 11 extracted features.

In addition to the publicly available datasets, this study incorporates a custom-built dataset created by collecting legitimate and phishing websites, including defaced, malware, and suspicious URLs, from reliable sources such as Kaggle and the URL 2016 dataset provided by the University of New Brunswick. These URLs were merged with the custom dataset comprising urls taken after the year 2023 to develop a more robust model and to mitigate potential biases present in individual datasets. The custom dataset contains 5,352 instances with 10 features and a class label, comprising 2,676 legitimate URLs and 2,676 phishing URLs.

All URLs underwent a feature extraction process, as illustrated in Figure 2. The detailed description of URL features, along with the pseudocode for feature extraction and URL validation, is presented in Sections 3.2, 3.3, and 3.4, respectively.

3-5-3- Configuration/Parameters of Models

The datasets were trained and tested using a variety of machine learning and deep learning models. A common base configuration was applied to the following machine learning classifiers: Support Vector Machine (SVM), Decision Tree, Extra Trees, Linear Discriminant Analysis (LDA), Gaussian Naive Bayes, and Gradient Boosting. The tuned hyperparameters for the classifiers are as follows: AdaBoost with a learning rate of 0.1; Multi-Layer Perceptron (MLP) with five hidden units and ReLU activation; k-Nearest Neighbors (k-NN) with a window size of 50; Logistic Regression with a regularization strength of 0.1; and Random Forest with 100 estimators.

The deep learning classification models were configured as follows:

i) CNN: A five-layer one-dimensional convolutional neural network with 16, 32, 64, 128, and 256 filters, respectively, followed by batch normalization. The kernel size was set to 3, dropout to 0.3, and max pooling with a pool size of 2. ReLU activation was used in the hidden layers, and sigmoid activation was applied in the output layer.

ii) CNN–LSTM: This model consists of five one-dimensional convolutional layers with 16, 32, 64, 128, and 256 filters, followed by batch normalization. The kernel size was set to 3, the max pooling size to 2, and dropout to 0.3. ReLU activation was used in the convolutional layers, and sigmoid activation was used in the output layer.

iii) Bidirectional LSTM (BiLSTM): This model employs 64 units representing the output dimensionality, with the tanh activation function. The return sequences parameter was set to true to enable bidirectional processing. A dropout rate of 0.1 was applied, followed by a dense layer with ReLU activation and an output layer with sigmoid activation.

iv) LSTM: The LSTM model incorporates batch normalization and a dropout rate of 0.3. ReLU activation was used in the hidden layers, and sigmoid activation was applied in the output layer.

v) HyDNN: In the proposed hybrid deep neural network, the first input branch comprises three layers of a one-dimensional convolutional network, followed by batch normalization and ReLU activation. The kernel size was set to 3 and the stride to 1. The second input branch consists of three bidirectional LSTM layers with the return sequences parameter set to true. The SELU activation function was employed with a *LeCun normal* kernel initializer. The model was trained using the Adam optimizer for 200 epochs, with the random seed set to 24 and the default learning rate of 0.001. The two branches were merged using sigmoid activation. Among all evaluated models, the proposed HyDNN achieved the highest classification accuracy.

3-5-4- Formulations of the Performance Metrics

The performance metrics used for machine learning and deep learning classification models [44] are:

i) **Accuracy:** It represents correct predictions over a total number of predictions.

$$Accuracy = \frac{True\ Pos + True\ Neg}{True\ Pos + True\ Neg + False\ Pos + False\ Neg} \quad (1)$$

ii) **Precision:** It can be termed as the ratio of true positive entities over the total number of positives (including true positive and false positive entities)

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

iii) **Recall:** It represents the ratio of true positive entities to overall positives in the ground truth.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

iv) **F1 score:** It is a representation of the combination of precision and recall. It can be also called the harmonic mean of the two.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

v) **Specificity:** It represents the model's ability to predict true negative entities over both true and false negative entities.

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Negatives} \quad (5)$$

vi) **Time complexity:** Time complexity evaluation metrics give information about how much time is consumed by the particular machine learning or deep learning model to evaluate the data.

3-5-5- Reproducibility of the Proposed Model

The research conducted in this study is reproducible using publicly available datasets [42, 45]. Additionally, researchers may create their own custom datasets by collecting URLs and supplying them to the proposed model, as described in Algorithms 1 and 2, to enable comparative analysis of results. The custom dataset used in this study is available at *Kaggle* [46] for research purposes. The source code will be made available upon request.

3-6- Feature Optimization Approaches

Feature optimization is a critical preprocessing step in developing an efficient and accurate model. The overall accuracy and performance metrics are highly dependent on the relevance and quality of the selected features. Selecting too many or too few features can negatively affect model performance and lead to suboptimal results. Therefore, identifying the optimal number of features and selecting the most informative ones play a vital role in training effective machine learning models. The threshold values used in feature-based filtering methods were determined empirically through exploratory data analysis and preliminary experiments conducted on the training dataset. The statistical distributions of key features were analyzed, and multiple threshold candidates were evaluated using validation performance to identify values that offered the best trade-off between detection accuracy and false positive rate. This clarification has been incorporated into the manuscript to enhance transparency and reproducibility.

The following section presents the feature selection and dimensionality reduction techniques employed in this study. All three datasets were preprocessed using a range of feature selection and dimensionality reduction methods. The optimal features selected by each approach are discussed in the Results section and summarized in Table 3.

Figure 3 illustrates the overall feature optimization workflow adopted in this study.

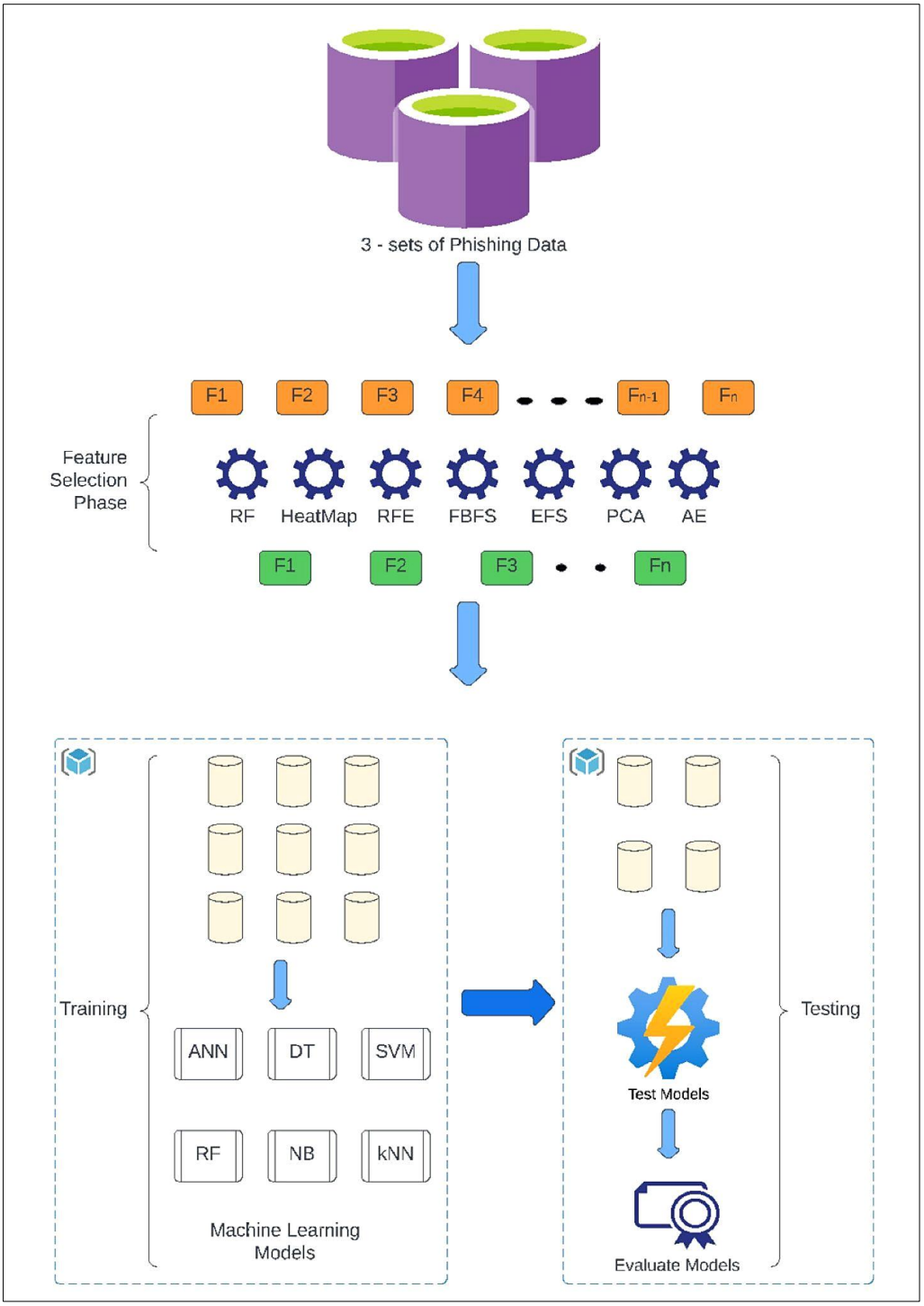


Figure 3. Applying Feature Optimization for Phishing Dataset

In Algorithm 1, the threshold (th) for each feature selection technique is determined empirically based on the corresponding relevance scores. During the feature selection phase, each feature F_i is evaluated using an appropriate evaluation metric, such as information gain, correlation coefficient, or feature importance score. The threshold is defined as a cutoff derived from the distribution of these scores, either as a fixed percentile or as a statistically meaningful value obtained through cross-validation. Features with scores exceeding the threshold are retained for training and testing, while those below the threshold are discarded. This process enables effective dimensionality reduction by preserving only the most informative features, thereby improving classification efficiency and performance.

Subsequently, the reduced feature sets are used to train machine learning and deep learning classification models. The trained models are evaluated on the test data using standard performance metrics. Algorithm 1 outlines the experimental procedure followed for feature selection and model training in this research. The subsequent subsections describe the dimensionality reduction techniques applied in detail.

Algorithm 1. Dimension Reduction and Feature Selection Algorithm

```
Require: Phishing Website Data-sets
Ensure: Optimum Features of the Data-sets
Variables: F - Features, L1, L2, L3 - Class Labels, E - Evaluation, PM -
Performance Metrics, tr - Training Set, ts - Testing Set, th - Threshold
F ⇐ F1, F2, ....Fn
if L1 ⊂ F && L1 = 0 then
Legitimate ⇐ 0
if L2 ⊂ F && L2 = 1 then
Suspicious ⇐ 1
Else Phishing ⇐ -1
end if
end if
▷ Set the Threshold for every Feature Selection Technique
while Fi ⊂ F do
if Fi > th then
Train = (tr ∈ F ) where training is 80% of F
Test = (ts ∈ F ) where testing is 20% of F
else
F = F − Fi
end if
▷ Apply ML models on Optimized Features
          E[P ML1,L2,L3 ] = F => M L(T rain, T est)
end while
```

3-6-1- Autoencoders (AE)

Autoencoders are neural network models that learn compressed representations of input data. They are capable of automatically extracting hidden features, also referred to as latent representations, using artificial neural networks. In this study, the encoder architecture consists of two hidden layers. The first hidden layer contains twice the number of neurons as the input features, while the second hidden layer has the same number of neurons as the input dimension. This configuration is followed by a bottleneck layer, which captures the latent representation of the input data. Leaky ReLU is employed as the activation function in the encoder, and batch normalization is applied to stabilize training. The decoder mirrors the encoder architecture in reverse order to reconstruct the input from the latent space. A linear activation function is used in the output layer to generate continuous numerical values.

3-6-2- Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is a backward feature selection wrapper technique that iteratively removes the least important features at each step and rebuilds the model using the remaining features. In this study, RFE is implemented with a decision tree estimator and evaluated using five-fold stratified cross-validation.

3-6-3- Filter-Based Feature Selection (FBFS)

High-dimensional data can result in increased training time and a higher risk of overfitting. Reducing the dimensionality of the feature space to a subset of relevant features lowers the computational cost of training and can also improve the model's generalization performance. Several filter-based feature selection methods exist, including Pearson's correlation, Linear Discriminant Analysis (LDA), Chi-square, and ANOVA.

In this study, a LASSO-based filter method is employed for feature selection. The LASSO technique automatically identifies informative features by shrinking the coefficients of redundant or irrelevant features to zero. Consequently, only features with non-zero coefficients are retained and used for model training and evaluation.

3-6-4- Feature Importance Score Using Random Forest Technique (RFFS)

This approach computes the importance of each feature by assigning it a relevance score. In the Random Forest technique, the mean decrease in impurity (Gini index) is used to estimate feature importance. A feature is considered more important if it results in a greater reduction in impurity across the decision trees, with lower Gini values indicating higher relevance. The Gini index is defined as:

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2 \quad (6)$$

P_i denotes the probability of the entropy. The scores are calculated based on the impurity criteria achieved based on the results of all the trees' splits on that feature. Figure 4 shows the importance of the feature calculated for data set 2.

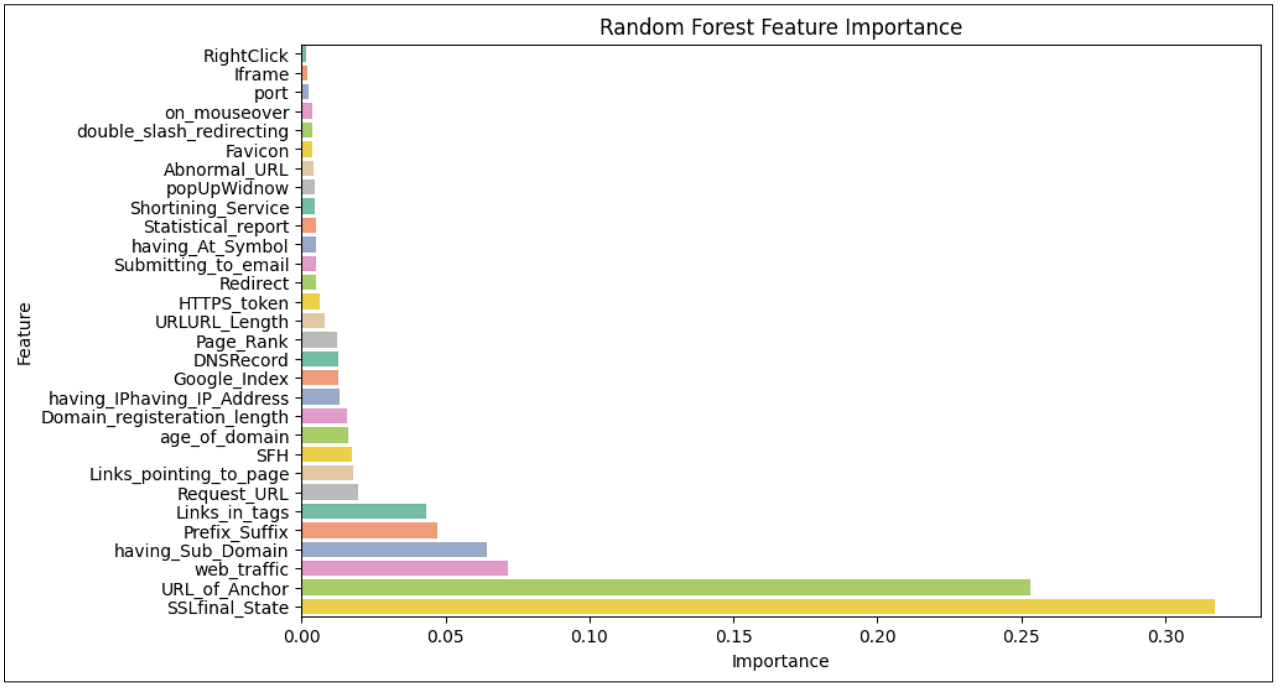


Figure 4. Random Forest Feature Importance Scores for Phishing Dataset 2

3-6-5- Principal Component Analysis (PCA)

The principal objective of Principal Component Analysis (PCA) is to reduce the dimensionality of the data while preserving as much of the inherent variance as possible. This is achieved by transforming the original features into a new set of uncorrelated variables, known as principal components. A covariance matrix is constructed to capture the correlations among the original variables. The eigenvalues and corresponding eigenvectors of the covariance matrix are then computed and sorted in descending order based on the magnitude of the eigenvalues. The top kkk eigenvectors, representing the principal components, are selected as the reduced dimensions. Finally, the standardized original data is projected onto these selected principal components to obtain a lower-dimensional representation of the data.

3-6-6- Heat-Map Correlation (HMC)

A heatmap correlation is a graphical visualization technique commonly used in Python to represent the relationships between variables. Positive values indicate a positive correlation between features, while negative values indicate a negative correlation. Based on the correlation analysis, researchers can make informed decisions regarding feature inclusion or exclusion. Features with correlation values close to zero are potential candidates for removal, as they are likely to contribute minimal information and may introduce noise or redundancy into the model. The threshold employed for selection in the study is ±0.03.

3-6-7- Embedded Feature Selection (EFS)

Embedded methods combine the advantages of both wrapper and filter techniques. While wrapper methods iteratively evaluate features to identify the least important ones, embedded methods perform feature selection concurrently with model training. Embedded methods tend to be more stable and exhibit less variability compared to wrapper approaches.

3-6-8- Theoretical Foundation

The theoretical foundation of this study is based on the hypothesis that combining multiple deep learning architectures can better capture the complex patterns inherent in phishing URLs than single models or traditional machine learning

methods. The proposed HyDNN (Hybrid Deep Neural Network) framework integrates convolutional neural networks (CNNs) for automatic feature extraction with bidirectional long short-term memory (BiLSTM) layers for sequential and contextual modeling. CNN layers are theoretically capable of identifying local spatial patterns in URL features, while BiLSTM layers capture both forward and backward dependencies in sequences, allowing the model to learn the temporal and structural relationships in phishing URLs.

The ensemble nature of HyDNN leverages the complementary strengths of these models, reducing the risk of overfitting and enhancing generalization to unseen data. Additionally, the study incorporates feature selection and dimensionality reduction techniques to ensure that only the most relevant features are used, thereby improving training efficiency and prediction accuracy.

Overall, the theoretical approach is grounded in the principles of hybrid deep learning and feature optimization, where the combination of spatial feature extraction, sequential modeling, and rigorous feature selection leads to a robust framework for phishing URL detection. Sections 4.9 and 4.10 provide detailed descriptions of the feature selection strategies and HyDNN architecture, respectively, to illustrate the implementation of this theoretical approach.

3-6-9- Hybrid Deep Learning Models

Deep learning models have a wide range of applications across various domains due to their ability to automatically extract features and train models without human intervention. Figure 5 illustrates the layer structure of the proposed HyDNN model, and the corresponding pseudocode is provided in Algorithm 2.

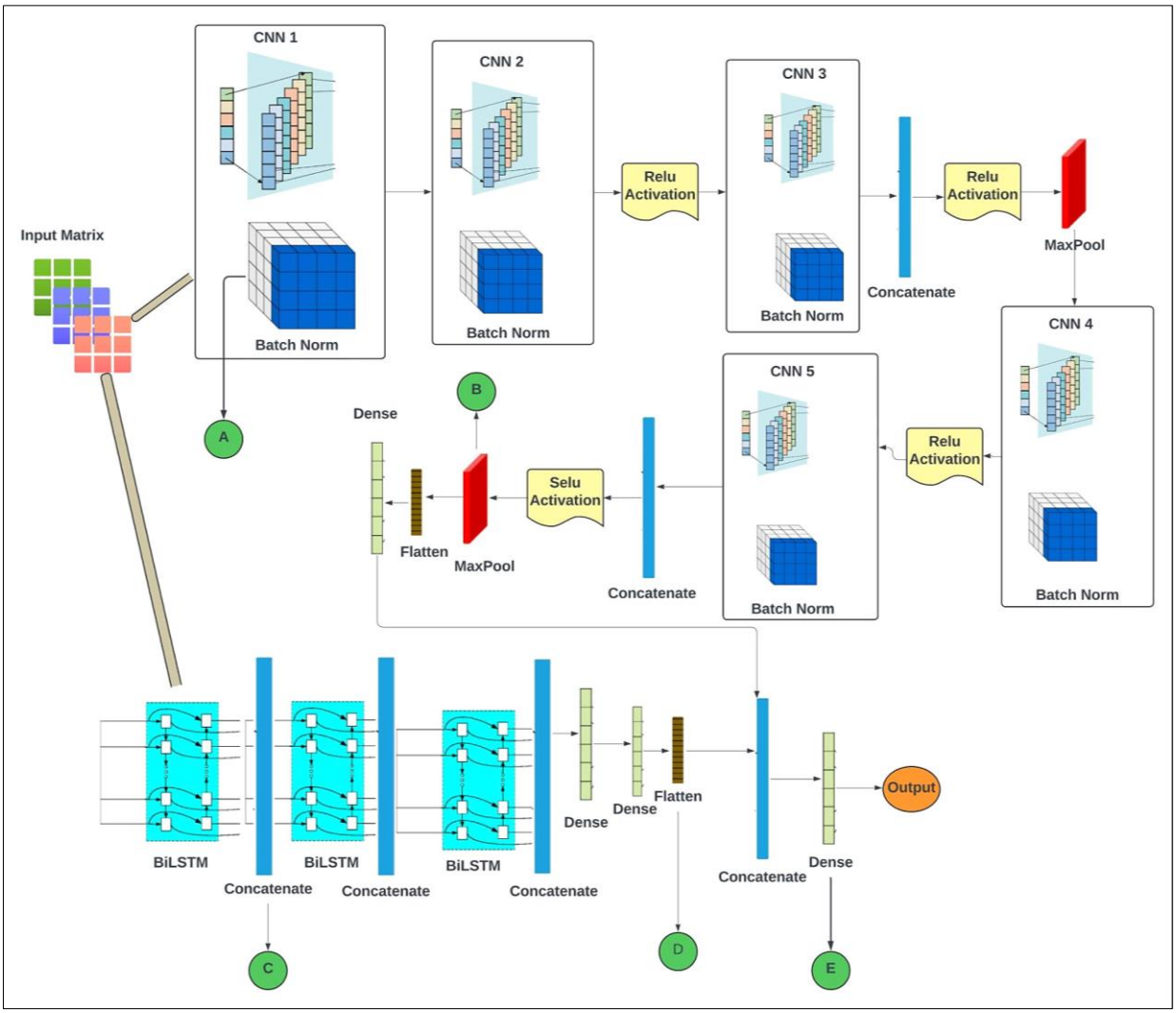


Figure 5. HyDNN Layering used in the Proposed Work

In this study, various deep learning architectures and customized deep neural networks (DNNs) are combined to form a hybrid model aimed at addressing the phishing URL problem. The models employed include CNN, LSTM, and bidirectional LSTM (BiLSTM). The hybrid model, which integrates CNN layers in parallel with BiLSTM layers, achieved superior performance compared to the baseline Random Forest model.

Figures 6 and 7 depict the basic architectures of the CNN and BiLSTM models, while Figure 5 illustrates the modified HyDNN architecture

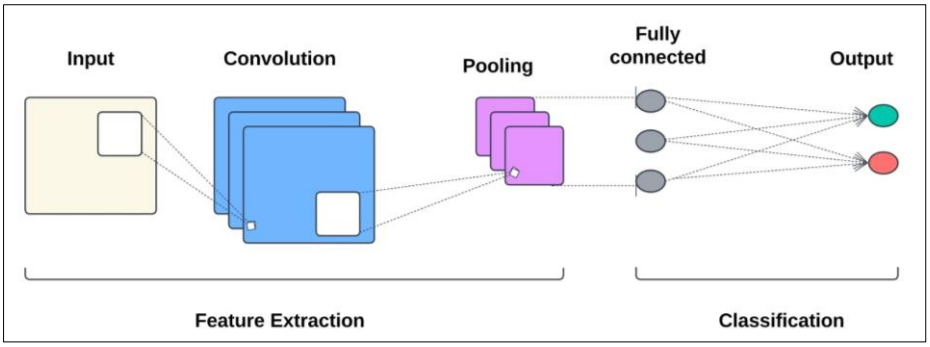


Figure 6. Basic CNN architecture

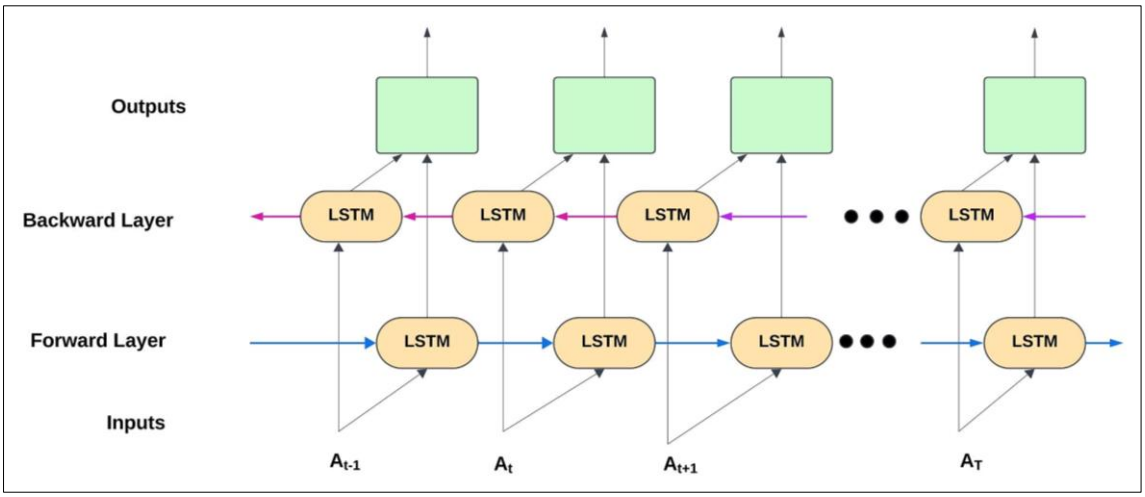


Figure 7. Basic BiLSTM architecture

The CNN–BiLSTM architecture combines convolutional neural network (CNN) layers for feature extraction with bidirectional LSTM (BiLSTM) layers for sequence prediction. BiLSTMs extend standard LSTMs by adding a layer that processes input sequences in the reverse direction, enabling the network to capture both forward and backward temporal dependencies. This allows the model to extract richer spatial and sequential information from the data. Our research demonstrates that the proposed hybrid model, HyDNN, achieves optimal performance in terms of both time and space complexity.

Machine learning models, such as Random Forest, were hyperparameter-tuned to improve accuracy. Random Forest was selected as the baseline model because it outperformed other machine learning classifiers. Its performance was compared with deep learning approaches across the three datasets.

The proposed HyDNN model consists of multiple CNN layers with 16 filters, a kernel size of 3, stride 1, followed by batch normalization with ReLU activation. Max pooling layers are used to downsample the feature maps, retaining the most significant information. The CNN layers are followed by three BiLSTM layers with SELU activation and LeCun normal kernel initialization. Finally, dense and flatten layers with sigmoid activation are used to generate the output. This hybrid architecture achieved the highest performance metrics among the evaluated models.

The annotations in Figure 5 are explained as follows:

A. Batch Normalization: A technique that normalizes the inputs of each layer to have a mean of zero and variance of one within mini-batches, improving network training stability and convergence.

B. MaxPool: A downsampling technique commonly used in CNNs to reduce the spatial dimensions of feature maps while retaining the most critical information.

C. Concatenate: An operation that joins two or more tensors along a specified axis, allowing the model to combine features from different layers or network branches.

D. Flatten: Converts a multi-dimensional tensor into a one-dimensional vector, preparing it for fully connected layers.

E. Dense: Also called a fully connected layer, where each neuron is connected to every neuron in the previous layer, forming a core component of many neural network architectures.

Algorithm 2. Pseudocode for HyDNN Model

```
Input: Feature vector of shape (48, 1)
Output: Binary classification result

Step 1: CNN Branch
inputs ← Input(shape=(48, 1))
out1 ← Conv1D(filters=16, kernel_size=3, strides=1)(inputs)
out1 ← BatchNormalization()(out1)
For _ in range(3):
    out ← Conv1D(filters=32, kernel_size=4, strides=1, padding='same')(out1)
    out ← BatchNormalization()(out)
    out ← Activation("relu")(out)
    out ← Conv1D(filters=16, kernel_size=2, strides=1, padding='same')(out)
    out ← BatchNormalization()(out)
    out ← Concatenate()([out, out1])
    out1 ← Activation("relu")(out)
    out1 ← MaxPooling1D(pool_size=3, strides=2)(out1)
cnn1 ← Flatten()(out1)
cnn1 ← Dense(30)(cnn1)

Step 2: LSTM Branch
x0 ← Bidirectional(LSTM(512, return_sequences=True))(inputs)
x ← Concatenate(axis=2)([inputs, x0])
x1 ← Bidirectional(LSTM(256, return_sequences=True))(x)
x ← Concatenate(axis=2)([x0, x1])
x2 ← Bidirectional(LSTM(128, return_sequences=True))(x)
lf ← Concatenate(axis=2)([x0, x1, x2])
lstm1 ← Dense(128, activation="selu", kernel_initializer='lecun_normal')(lf)
lstm1 ← Dense(1)(lstm1)
lstm1 ← Flatten()(lstm1)

Step 3: Merge CNN and LSTM Features
merge ← Concatenate()([cnn1, lstm1])
output ← Dense(1, activation="sigmoid")(merge)
model6 ← Model(inputs=[inputs], outputs=output)
```

4- Results and Discussion

The following section presents the results obtained from the experimental implementation. The configuration parameters used and fine-tuned for the machine learning models are as follows: Decision Tree (DT) and Random Forest (RF) classifiers were configured with a maximum depth of 10, Gini as the splitting criterion, and 190 estimators for Random Forest. Other classifiers, such as Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN), were used with k set to 5.

For the neural network models, ReLU and tanh activations were applied in the output layer for binary classification tasks, while ReLU and sigmoid activations were used for categorical datasets. As illustrated in Figure 3, the thresholds determined during feature selection (line 8 of Algorithm 1) were applied to filter the features. The subsequent figures present the different feature selection techniques employed, along with the number of features retained for each method and each dataset used in the model training process.

4-1- Without any Filters

The primary objective of this experiment, which applies classification models to the three datasets, is to compare the results obtained before and after applying feature selection filters. This comparison helps evaluate the impact of feature selection on model performance. Figure 8 presents the results of applying machine learning models to the preprocessed datasets without any feature selection filters applied.

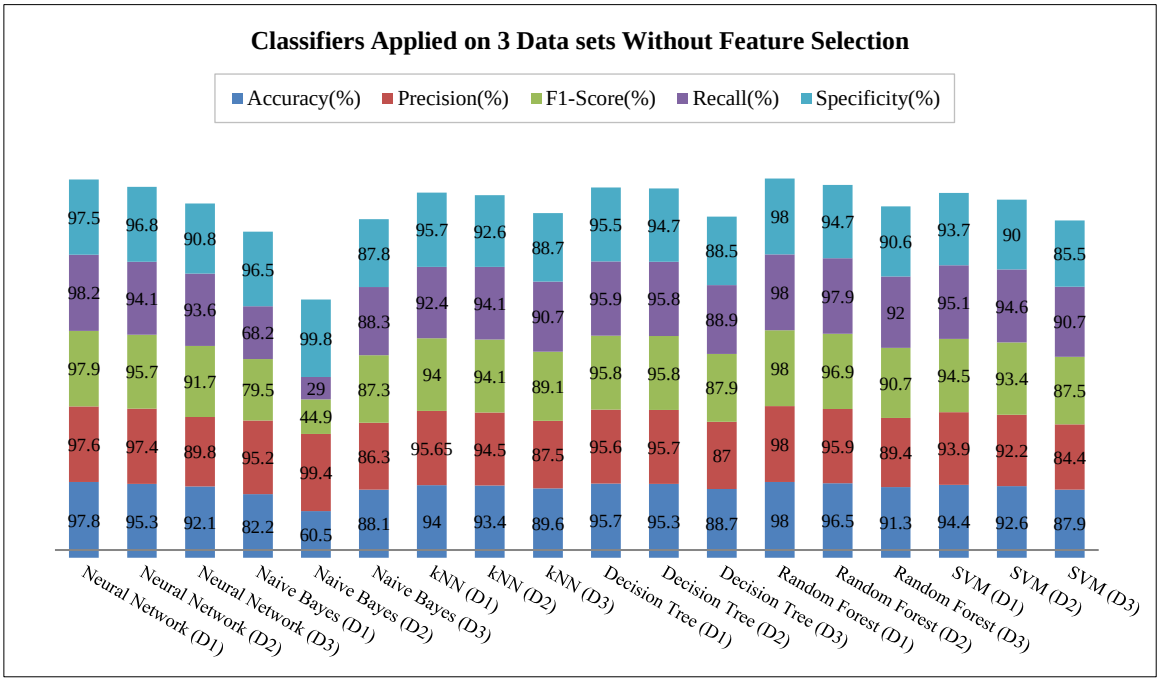


Figure 8. Classifiers Applied on 3 Data sets without Feature Selection

4-2- With Recursive Feature Elimination (RFE)

Figure 9 shows the results obtained from applying machine learning models to data sets with recursive feature elimination (RFE). Features selected by RFE for the different datasets are given below:

For Dataset 1, the number of features selected by RFE is 39 for the dataset 2, the number of features selected by RFE is 17 and for dataset 3, the number of features selected by RFE is 6.

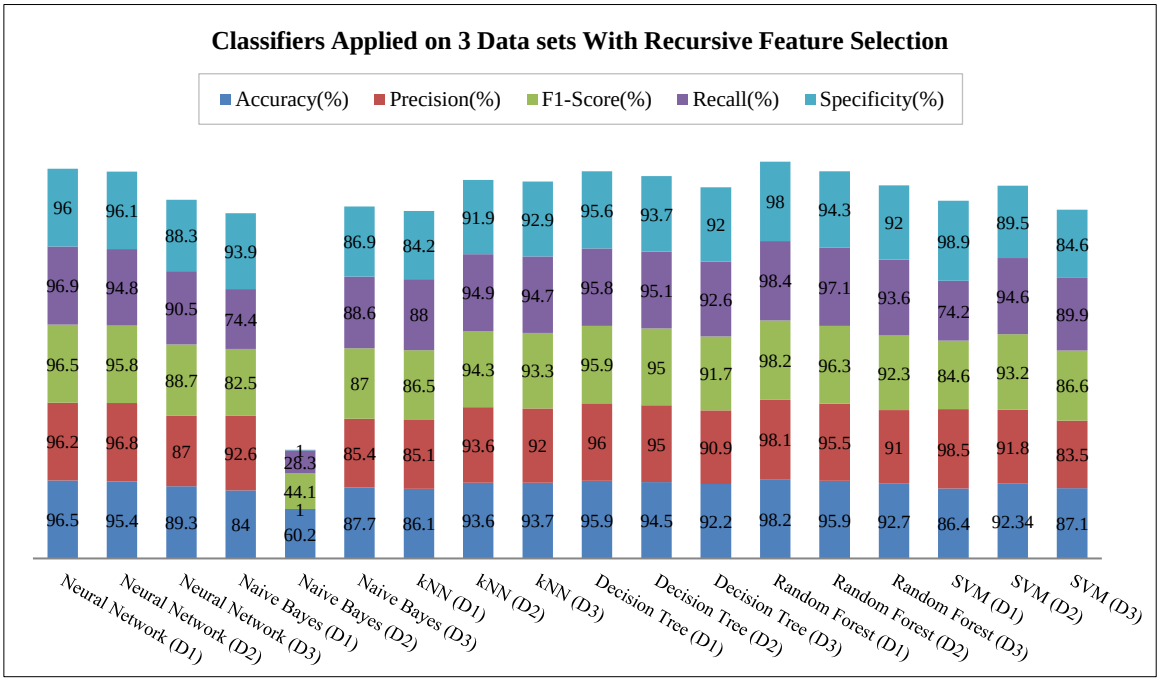


Figure 9. Classifiers Applied on 3 Data sets with Recursive Feature Selection (RFE)

4-3- With Filter based Feature Selection (FBFS)

Figure 10 shows the results obtained from applying machine learning models to data sets with features selected by filter-based feature selection (FBFS).

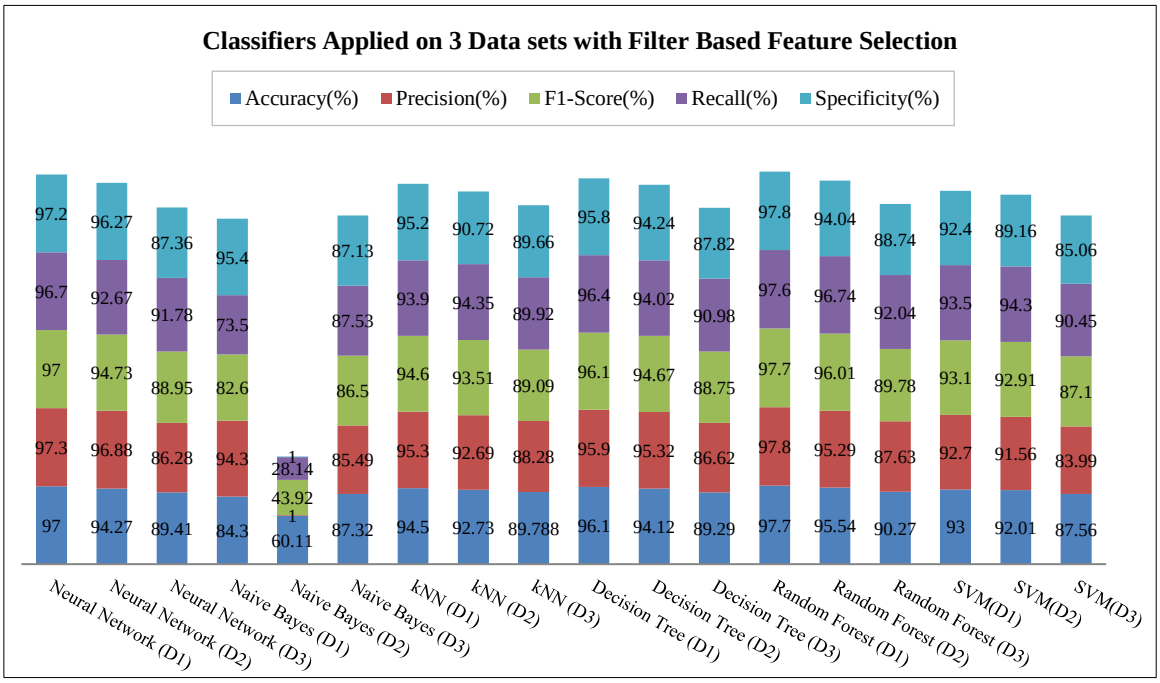


Figure 10. Classifiers Applied on 3 Data sets with Filter Based Feature Selection (FBFS)

For Dataset 1, the filter method used 26 selected features. Dataset 2: Filter Method used 18 selected features. Dataset 3: Filter Method used 6 selected features.

4-4- With Features Selected by Embedded Method

Figure 11 shows the results obtained from applying machine learning models to data sets with features selected by the embedded method. The embedded method focuses on the following aspects:

For dataset 1, the lasso picked 43 variables and eliminated the other 5 variables. For dataset 2, Lasso picked 29 variables and eliminated the other 1 variable, and for dataset 3, Lasso picked all 9 variables.

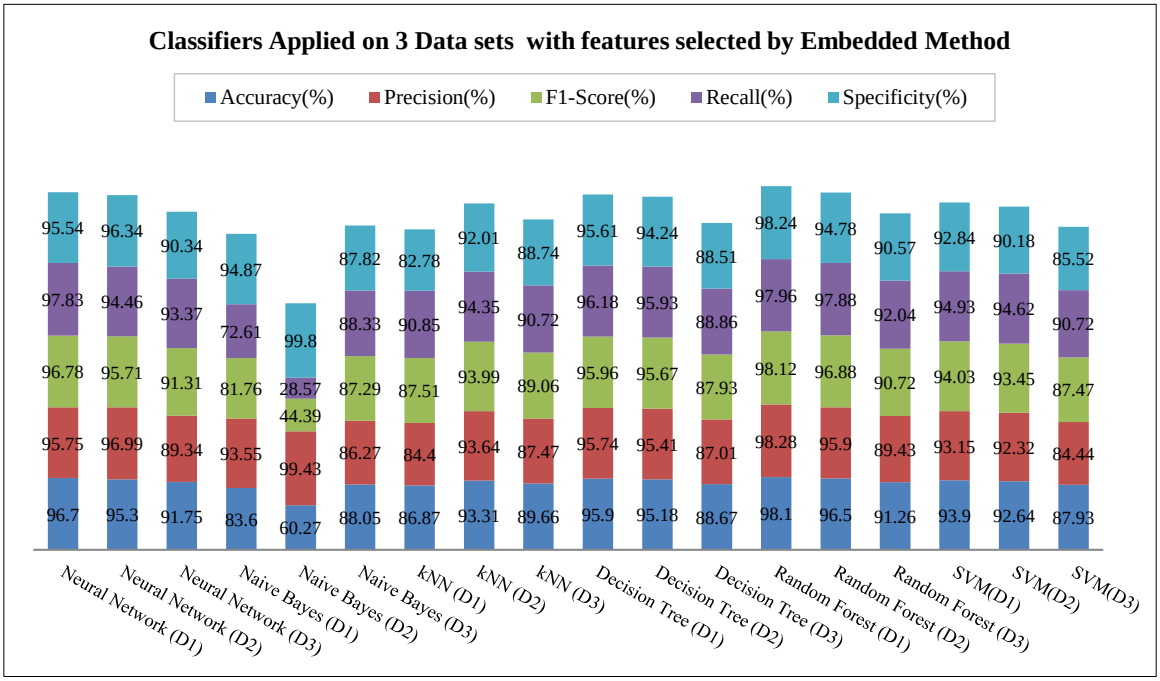


Figure 11. Classifiers Applied on 3 Data sets features selected by Embedded Method

4-5- Autoencoder Method

Figure 12 shows the accuracy of the performance evaluation metric obtained by applying machine learning models to three data sets with AutoEncoder. The input is reconstructed in two parts: the encoder, which takes the input and compresses it into a lower dimensional form, and the decoder, which takes the compressed representation and reconstructs the input.

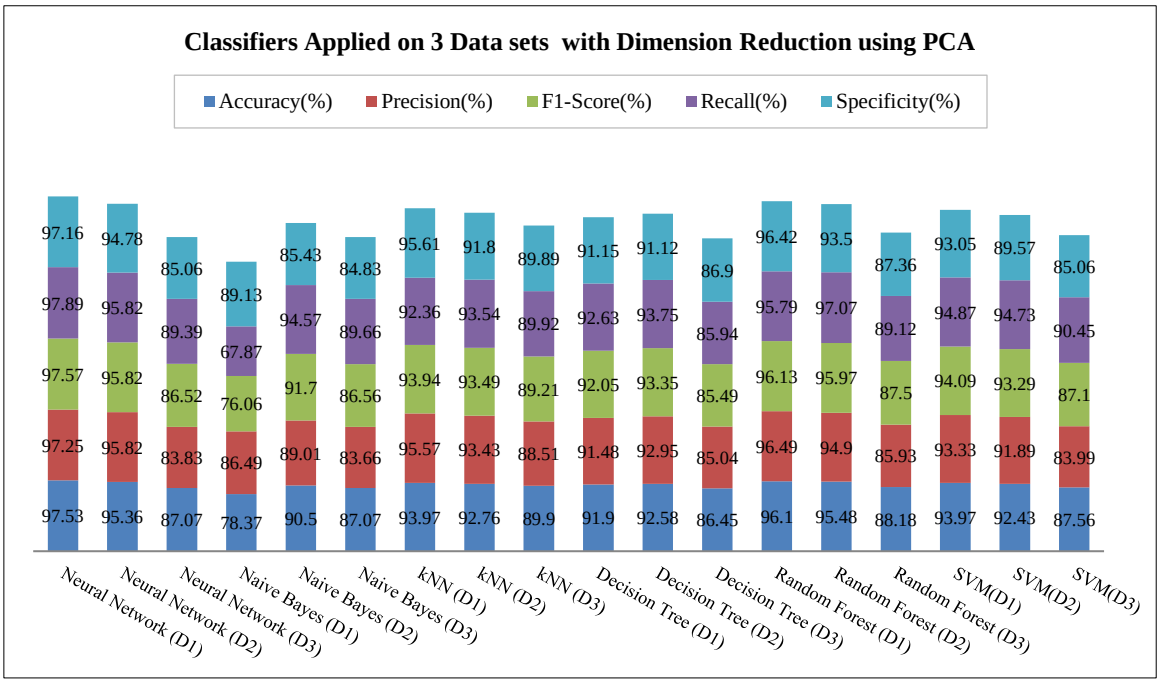


Figure 12. Classifiers Applied on 3 Data sets with features compressed by Autoencoder

4-6- With features selected by the PCA Method

PCA operates on the principle of accepting all input features but focusing only on the relevant ones that exhibit correlation, thereby enabling dimensionality reduction. Principal components are linear combinations of the original features that capture the maximum variability present in the dataset. For most applications, the optimal number of principal components is typically 2–3. In this study, PCA selected 43 principal components for Dataset 1, 18 for Dataset 2, and 6 for Dataset 3. Figure 13 presents the accuracy metrics of various machine learning classifiers applied to the three datasets.

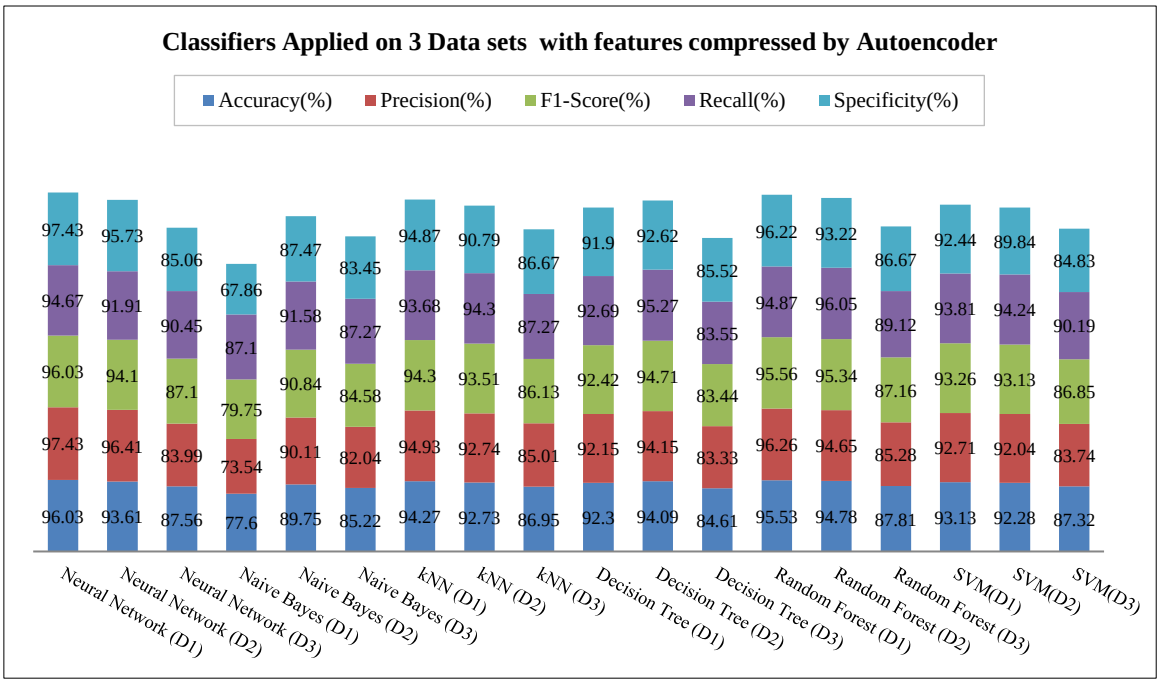
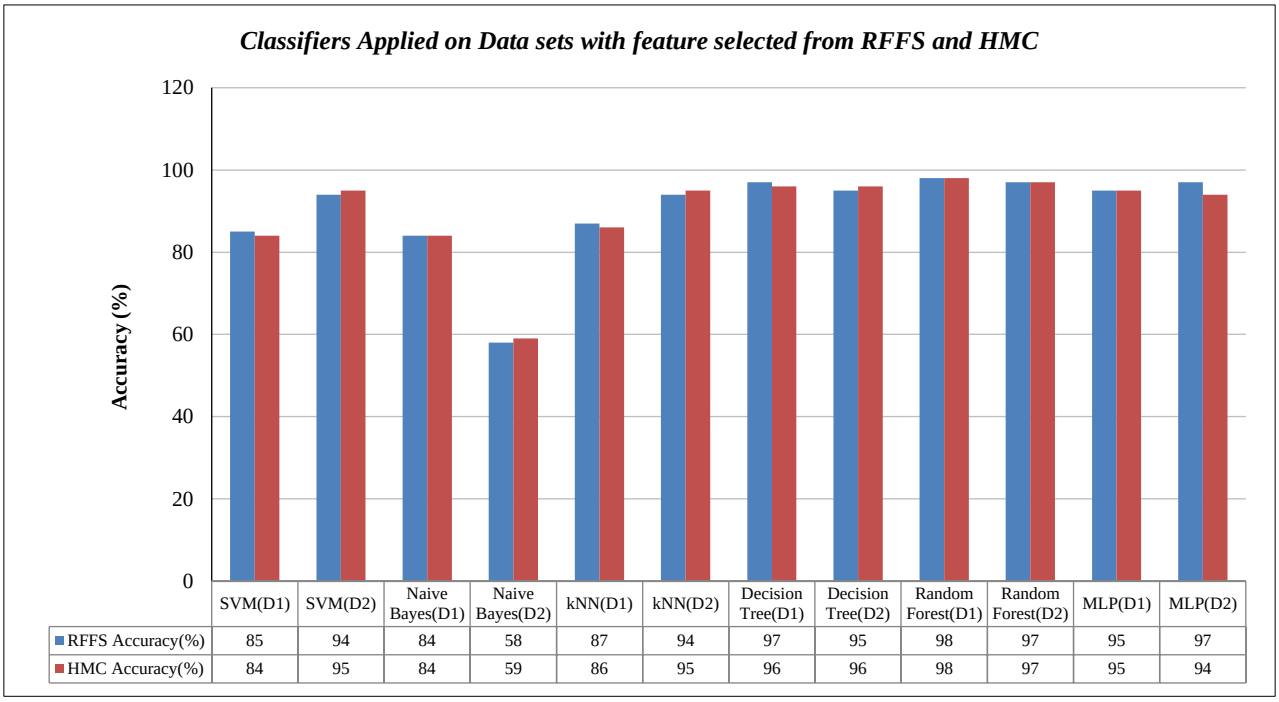


Figure 13. Classifiers Applied on 3 Data sets with Dimension Reduction using PCA

PCA effectively reduces dimensionality without significant loss of information from the original features. It is observed that Naive Bayes does not achieve optimal performance on phishing datasets. This is because Naive Bayes assumes that features are mutually independent, an assumption that does not hold true for phishing data, leading to reduced classification accuracy compared to other models.

4-7- With features selected by Random Forest Feature Importance Score (RFFS)

The Random Forest method for feature selection is widely regarded as one of the most effective approaches. In this study, it was applied to the binary classification datasets, namely Dataset 1 and Dataset 2. Figure 14 presents the results of applying machine learning models to these datasets after feature selection using both the Random Forest Feature Selection (RFFS) method and the Heatmap Correlation approach.



	SVM(D1)	SVM(D2)	Naive Bayes(D1)	Naive Bayes(D2)	kNN(D1)	kNN(D2)	Decision Tree(D1)	Decision Tree(D2)	Random Forest(D1)	Random Forest(D2)	MLP(D1)	MLP(D2)
RFFS Accuracy(%)	85	94	84	58	87	94	97	95	98	97	95	97
HMC Accuracy(%)	84	95	84	59	86	95	96	96	98	97	95	94

Figure 14. Accuracy after Feature Optimization for Phishing Data-sets using Random Forest Feature Selection and Heat Map Correlation Method

Using the RFFS approach on Dataset 1, six features were eliminated: PopUpWindow, ImagesOnlyInForm, DoubleSlashInPath, FakeLinkInStatusBar, HttpsInHostname, and AtSymbol. For Dataset 2, a total of 13 features were removed: RightClick, SubmittingToEmail, PopUpWindow, Port, OnMouseOver, AtSymbol, Favicon, Iframe, DoubleSlashRedirecting, Redirect, HTTPSToken, AbnormalURL, and ShorteningService.

Using the heatmap correlation approach, features with correlation values within the range of ±0.03 were removed to reduce redundancy and improve model performance.

4-8- With features selected by Heat Map Correlation (HMC)

Heat map correlation gives the relationship between all the features of the data set. A negative correlation means that the two attributes are negatively affected. Whereas positive values indicate that there is a positive correlation between the two values. Features with correlations close to zero were removed as they likely add noise or redundancy. The threshold used for selection in the research study is ±0.03. Hence, attributes with values greater than ±0.03 have been added for the experiment. Results of applying heat map correlation on data set 1 and data set 2 are shown in Figure14.

4-9- Summary of Feature Selection Techniques

The best-performing machine learning classifiers corresponding to each feature selection technique across the three datasets are presented in Table 2. It can be observed that the hyperparameter-tuned Random Forest classifier consistently provides optimal results for most feature selection techniques. Consequently, Random Forest is considered the baseline model and is compared with deep learning approaches. Table 3 summarizes the total number of features selected by each filter for processing the various datasets.

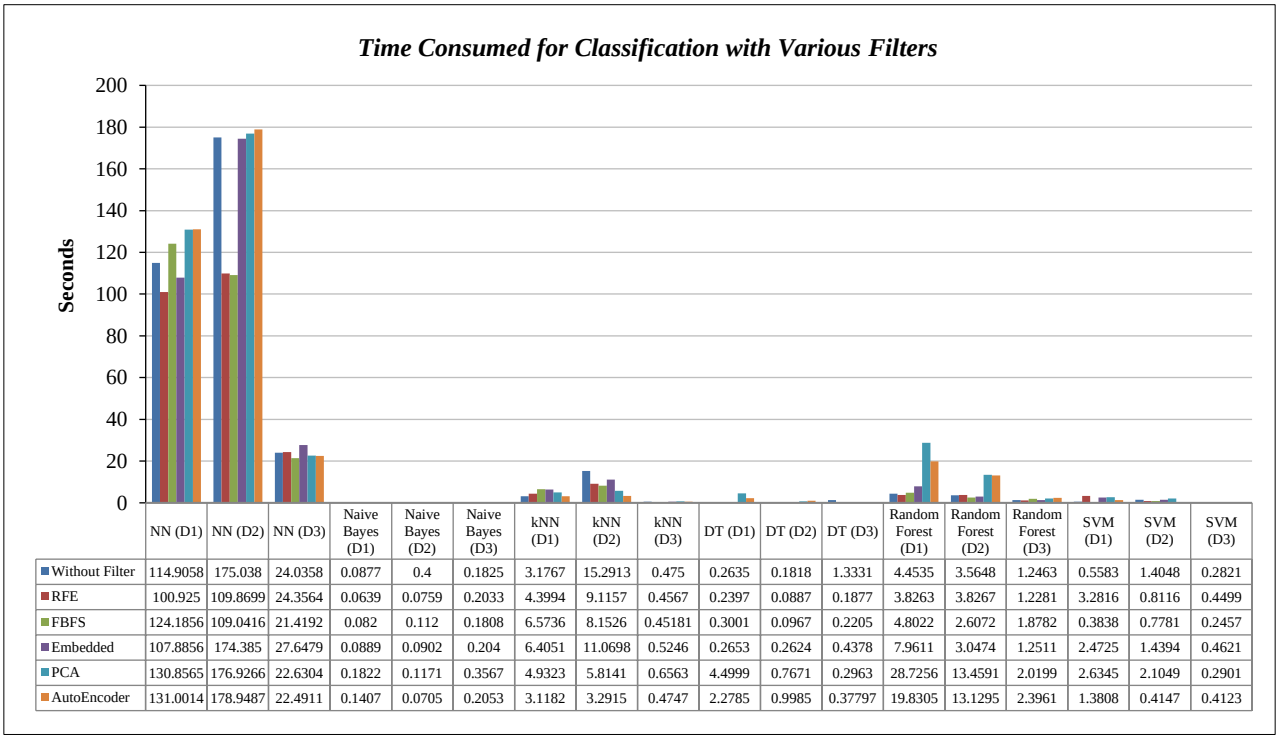
Table 2. Best classifier based on Accuracy metrics for various Feature Selection Techniques.

Feature Selection Technique	Dataset 1 (Accuracy %)	Dataset 2 (Accuracy %)	Dataset 3 (Accuracy %)
Without Filter	Random Forest (98.0%)	Random Forest (96.5%)	Neural Network (92.1%)
PCA	Neural Network (97.53%)	Neural Network (95.36%)	kNN (89.9%)
Auto Encoders	Neural Network (96.03%)	Random Forest (94.78%)	Random Forest (87.81%)
RFE	Random Forest (98.2%)	Random Forest (95.9%)	Random Forest (92.7%)
FBFS	Random Forest (97.7%)	Random Forest (95.54%)	Random Forest (90.27%)
Heat Map Correlation	Random Forest (98.0%)	Random Forest (97.0%)	–
Embedded Method	Random Forest (98.1%)	Random Forest (96.5%)	Neural Network (91.75%)
RF Feature Importance Score	Random Forest (98.0%)	–	Random Forest (97.0%)

Table 3. Total Number of Features selected by various Feature Selection Techniques.

Dataset	No Filter	RFE	FBFS	HM	RFFIS	Embedded	PCA	AE
Dataset – 1	48	39	26	32	28	43	43	43
Dataset – 2	32	17	18	23	18	29	18	18
Dataset – 3	11	6	6	7	6	9	6	6

The Random Forest Feature Importance Score (RFFS) performed particularly well for the phishing datasets, primarily due to its versatility in handling both classification and regression tasks. This method assigns an importance score to each feature based on the Gini importance metric. The Random Forest classifier combined with RFFS achieved the highest accuracy for Datasets 1 and 2, while Recursive Feature Elimination (RFE) applied with the k-Nearest Neighbor classifier yielded better accuracy for Dataset 3. These results demonstrate that the choice of feature selection technique can significantly influence model evaluation metrics. The time required by each filter to select features and process data for machine learning models is illustrated in Figure 15.



	NN (D1)	NN (D2)	NN (D3)	Naive Bayes (D1)	Naive Bayes (D2)	Naive Bayes (D3)	kNN (D1)	kNN (D2)	kNN (D3)	DT (D1)	DT (D2)	DT (D3)	Random Forest (D1)	Random Forest (D2)	Random Forest (D3)	SVM (D1)	SVM (D2)	SVM (D3)
Without Filter	114.9058	175.038	24.0358	0.0877	0.4	0.1825	3.1767	15.2913	0.475	0.2635	0.1818	1.3331	4.4535	3.5648	1.2463	0.5583	1.4048	0.2821
RFE	100.925	109.8699	24.3564	0.0639	0.0759	0.2033	4.3994	9.1157	0.4567	0.2397	0.0887	0.1877	3.8263	3.8267	1.2281	3.2816	0.8116	0.4499
FBFS	124.1856	109.0416	21.4192	0.082	0.112	0.1808	6.5736	8.1526	0.45181	0.3001	0.0967	0.2205	4.8022	2.6072	1.8782	0.3838	0.7781	0.2457
Embedded	107.8856	174.385	27.6479	0.0889	0.0902	0.204	6.4051	11.0698	0.5246	0.2653	0.2624	0.4378	7.9611	3.0474	1.2511	2.4725	1.4394	0.4621
PCA	130.8565	176.9266	22.6304	0.1822	0.1171	0.3567	4.9323	5.8141	0.6563	4.4999	0.7671	0.2963	28.7256	13.4591	2.0199	2.6345	2.1049	0.2901
AutoEncoder	131.0014	178.9487	22.4911	0.1407	0.0705	0.2053	3.1182	3.2915	0.4747	2.2785	0.9985	0.37797	19.8305	13.1295	2.3961	1.3808	0.4147	0.4123

Figure 15. Time Taken by Classifiers Applied on Various Filters

Table 4 summarizes the key features, along with the advantages and limitations of each feature selection and dimensionality reduction technique. Dataset 1 (Mendeley dataset) contains 48 attributes, Dataset 2 (UCI dataset) has 32 features, and Dataset 3 (multi-class dataset) includes 11 features.

Table 4. Pros and Cons of various Feature Selection Techniques

Technique	Key Features	Pros	Computational Trade-offs
Without Filter	Uses all available features	Considers all features without elimination	Computational time increases due to processing unnecessary features
Auto Encoder	Compresses and reconstructs input data by dropping irrelevant features	Non-linear dimensionality reduction using deep neural networks	Requires training; computationally expensive and may not capture complex relationships accurately
Heat Map Correlation	Selects variables highly correlated with the target and weakly correlated among themselves	Retains highly correlated features that improve predictive performance	Pairwise correlation computation is slow and memory-intensive for high-dimensional data
RFFIS (RF Feature Importance Score)	Calculates feature scores using Gini impurity or Information Gain	Handles large, high-dimensional datasets effectively	Aggregation across multiple trees increases computational cost
FBFS (Filter-Based Feature Selection)	Selects features based on statistical test scores	Requires storing only statistical scores per feature	Computing correlations or mutual information in high-dimensional data increases memory usage due to large (n \times n) matrices
Embedded Methods	Combines filter and wrapper approaches	Performs feature selection during training, reducing overhead	Requires tuning parameters such as regularization strength (e.g., LASSO) or tree depth, increasing computation
RFE (Recursive Feature Elimination)	Iteratively trains models on reduced feature subsets	Systematically removes less important features, reducing complexity	Highly computationally expensive due to repeated model training
PCA	Transforms data into a smaller set of principal components	Improves interpretability and reduces dimensionality	Recomputing principal components is costly for dynamic datasets

4-10- Hybrid Deep Learning Models

Deep learning algorithms, including CNN, LSTM, and bidirectional LSTM (BiLSTM), as well as their ensemble variants, were evaluated on two binary-classified datasets and a custom dataset. Tables 5 to 7 present the results of these deep learning models compared to the baseline Random Forest model. Random Forest was chosen as the baseline due to its superior accuracy among the machine learning classifiers.

Table 5. Deep Learning Techniques Applied on Dataset - 1

Classifier / Metrics	Accuracy	Precision	Recall	F1 Score
Random Forest	0.986000	0.985526	0.986825	0.986175
Bidirectional LSTM	0.980000	0.975849	0.984848	0.980328
LSTM Model	0.983000	0.976608	0.990119	0.983317
CNN Model	0.979000	0.981469	0.976943	0.979201
CNN–LSTM Model	0.952667	0.984507	0.920949	0.951668
Proposed HyDNN Model	0.988667	0.986877	0.990777	0.988823

Table 6. Deep Learning Techniques Applied on Dataset- 2

Classifier / Metrics	Accuracy	Precision	Recall	F1 Score
Random Forest	0.970452	0.964336	0.983435	0.973792
Bidirectional LSTM	0.973065	0.968119	0.984156	0.976071
LSTM Model	0.973668	0.966173	0.987396	0.976670
CNN Model	0.976080	0.970609	0.987036	0.978754
CNN–LSTM Model	0.971859	0.968051	0.981995	0.974973
Proposed HyDNN Model	0.976281	0.970287	0.987757	0.978944

Table 7. Deep Learning Techniques Applied on Custom Data

Classifier / Metrics	Accuracy	Precision	Recall	F1 Score
Random Forest	0.934866	0.946894	0.919261	0.932873
Bidirectional LSTM	0.892241	0.874185	0.912451	0.892908
LSTM Model	0.730843	0.795685	0.609922	0.690529
CNN–LSTM Model	0.713602	0.796143	0.562257	0.659065
CNN Model	0.715517	0.780362	0.587549	0.670366
Proposed HyDNN Model	0.937739	0.918843	0.958171	0.938095

The proposed hybrid deep learning model, HyDNN, which combines CNN and BiLSTM layers, outperformed other models across most performance evaluation metrics. HyDNN was constructed with three CNN layers followed by two BiLSTM layers with ReLU activation. The execution time of HyDNN was faster than other deep neural network approaches, owing to the efficient layering of the hybrid architecture. Figures 16 to 20 display the confusion matrices for the deep learning models, while Figures 21 to 25 illustrate the ROC curves for the custom dataset. From these results, it is evident that HyDNN achieves superior performance compared to the other models evaluated.

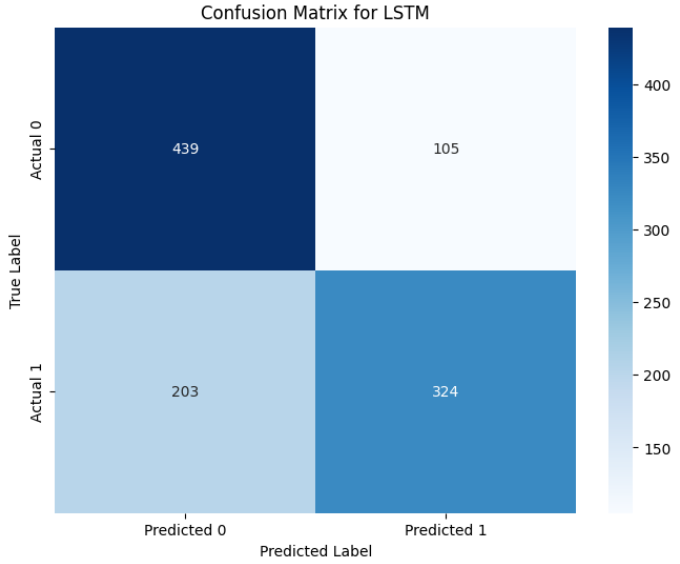


Figure 16. Confusion Matrix-LSTM

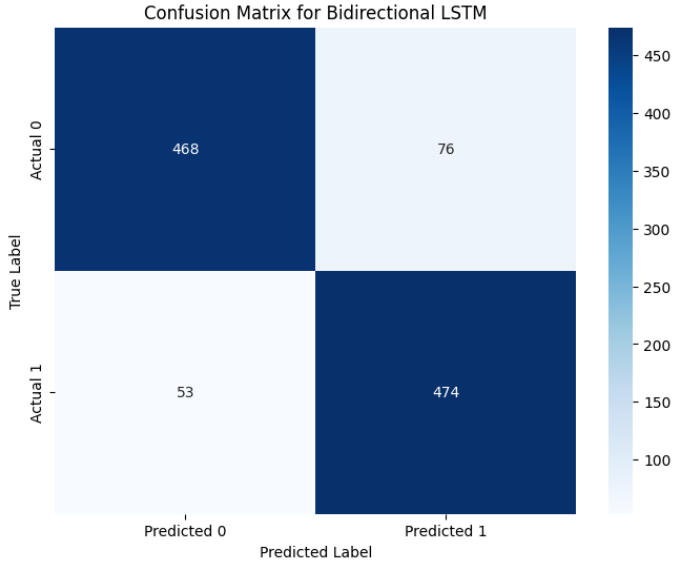


Figure 17. Confusion Matrix-BiLSTM

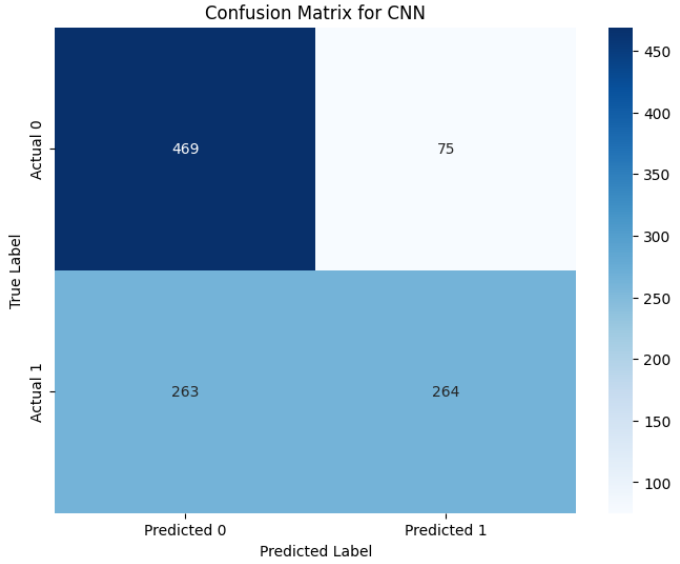


Figure 18. Confusion Matrix-CNN

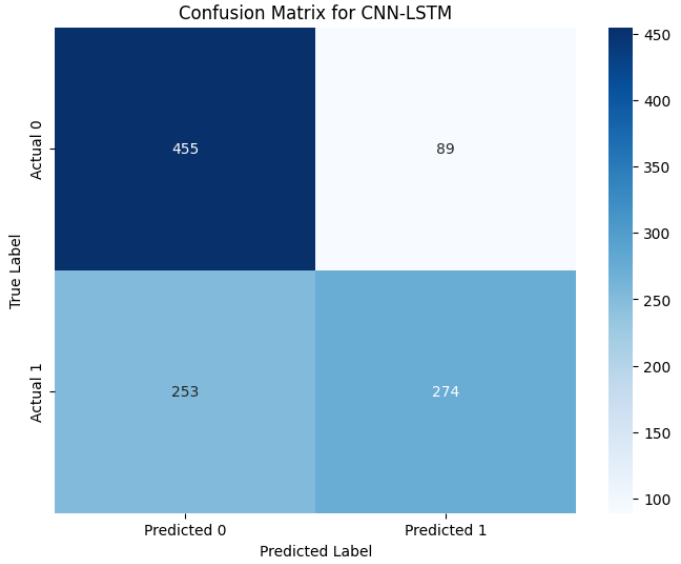


Figure 19. Confusion Matrix-CNN-LSTM

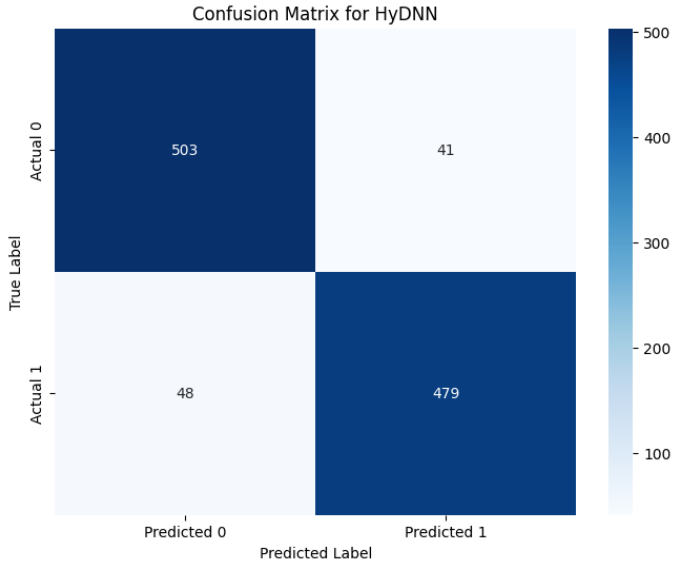


Figure 20. Confusion Matrix-HyDNN

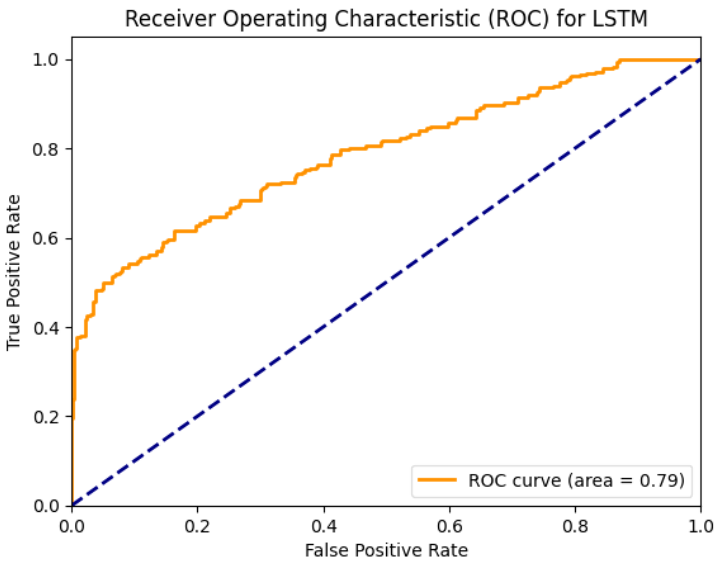


Figure 21. AUC ROC-LSTM

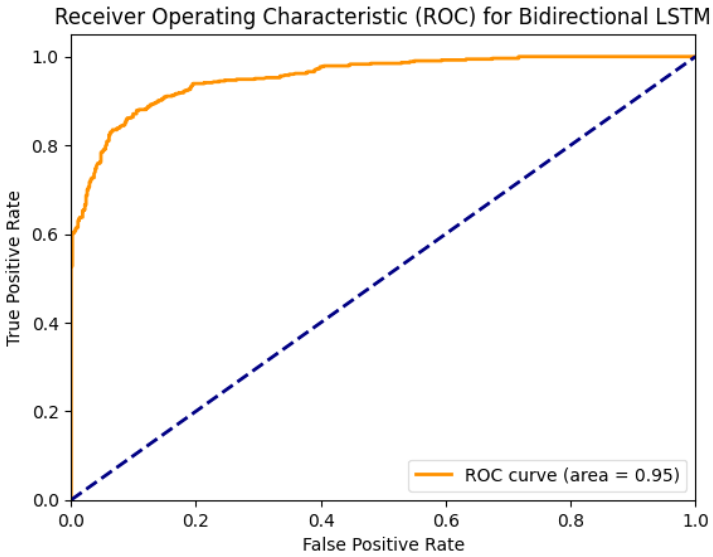


Figure 22. AUC ROC-BiLSTM

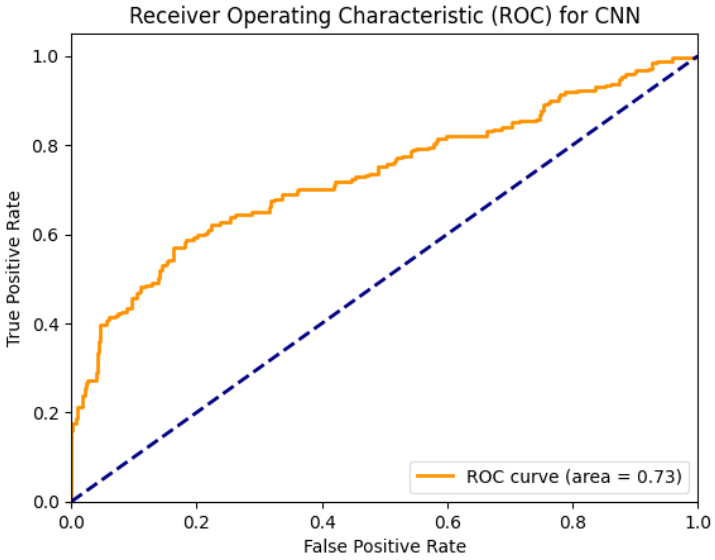


Figure 23. AUC ROC-CNN

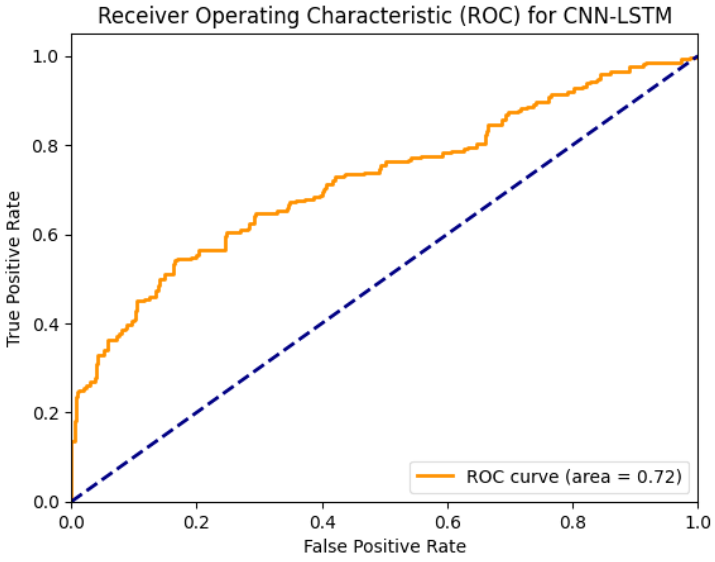


Figure 24. AUC ROC-CNN-LSTM

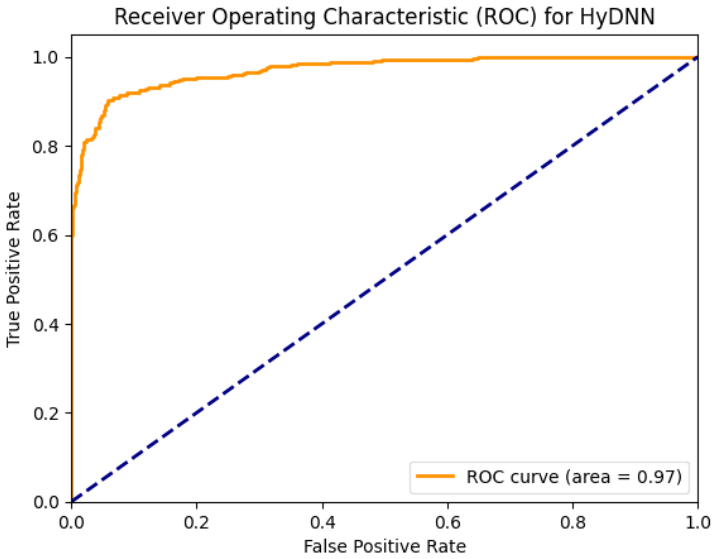


Figure 25. AUC ROC-HyDNN

4-11- Interpretation of Experimental Findings

The experimental results demonstrate that both the choice of feature selection technique and the learning model significantly influence phishing detection performance across different datasets. As shown in Table 2, the hyperparameter-tuned Random Forest classifier consistently outperformed other traditional machine learning models for most feature selection techniques. This consistent behavior indicates Random Forest's strong capability to handle high-dimensional, heterogeneous feature spaces commonly found in phishing URL datasets. Its ensemble nature and robustness to noise make it particularly effective in capturing complex feature interactions, justifying its selection as the baseline model for subsequent comparisons with deep learning approaches.

The feature selection analysis further reveals that Random Forest Feature Importance Score (RFFS) is highly effective for phishing detection tasks, especially for Datasets 1 and 2. By leveraging the Gini importance metric, RFFS prioritizes features that contribute most to impurity reduction, enabling the model to focus on highly discriminative attributes while suppressing redundant ones. This explains the superior accuracy achieved by the Random Forest–RFFS combination on these datasets. In contrast, Dataset 3, which is comparatively smaller and multi-class in nature, benefited more from Recursive Feature Elimination (RFE) combined with the k-Nearest Neighbor classifier. This suggests that wrapper-based methods like RFE, which iteratively evaluate feature subsets, are better suited for datasets with fewer features and more complex class distributions. These findings highlight that feature selection is not universally optimal and must be tailored to dataset characteristics.

The analysis of feature selection time, illustrated in Figure 15, indicates that filter-based methods offer faster processing compared to wrapper-based approaches, making them more suitable for large-scale or real-time phishing detection systems. However, the higher computational cost of wrapper methods is offset by improved accuracy in certain scenarios, emphasizing a trade-off between efficiency and predictive performance. Table 4 further supports this observation by summarizing the strengths and limitations of each feature selection and dimensionality reduction technique.

When extending the evaluation to deep learning models, the results in Tables 5 to 7 clearly show a performance gain over traditional machine learning approaches, including the Random Forest baseline. Convolutional Neural Networks (CNNs) effectively capture local feature patterns, while LSTM and BiLSTM models excel at learning sequential dependencies within URL and content-based features. However, individual deep learning architectures exhibit limitations when used in isolation, either in terms of convergence speed or generalization across datasets.

The proposed hybrid deep learning model, HyDNN, consistently achieved the best performance across most evaluation metrics. By integrating CNN layers for spatial feature extraction with BiLSTM layers for bidirectional temporal learning, HyDNN effectively captures both local and long-range dependencies in phishing data. The use of ReLU activation further improves training stability and convergence. Notably, HyDNN also demonstrated faster execution time compared to other deep learning models, indicating that its architectural design achieves an optimal balance between complexity and computational efficiency.

The confusion matrices presented in Figures 16 to 20 confirm HyDNN's superior classification capability, with a higher number of correctly classified instances and reduced false positives and false negatives. This is particularly

important in phishing detection, where minimizing false negatives is critical for security. Similarly, the ROC curves shown in Figures 21 to 25 indicate a higher area under the curve (AUC) for HyDNN, reflecting its strong discriminative ability across varying decision thresholds.

Overall, these results validate the effectiveness of the proposed feature optimization framework and hybrid deep learning architecture. The study demonstrates that combining optimized feature selection with a carefully designed hybrid deep learning model significantly enhances phishing detection performance, outperforming both traditional machine learning classifiers and standalone deep learning models. This comprehensive evaluation across multiple datasets and learning paradigms establishes the robustness and generalizability of the proposed HyDNN approach.

4-12- State of Art Comparisons

State-of-the-art methods refer to the most advanced, effective, and up-to-date techniques available in a particular research or application domain at the time of study. These methods represent the current benchmark or best-performing approaches reported in the literature and are typically used for comparison to demonstrate improvements or innovations in new research.

In the context of phishing detection, state-of-the-art methods include the latest machine learning, deep learning, and hybrid models that achieve superior performance in terms of accuracy, robustness, and adaptability to evolving attack patterns.

The results are compared with existing articles and depicted in the following table of state-of-the-art methods 8 for dataset 1 and dataset 2. The accuracy performance metrics, training time, system requirements and tools used are shown in the Table 8. Table 8 presents a comparative overview of state-of-the-art methods for phishing detection.

Table 8. State of Art Methods

Article	Classifier	Accuracy (%) Dataset 1	Accuracy (%) Dataset 2	Training Time (sec)	System Requirement / Tools
Alsariera et al. [47]	Boosted NB Tree	98.38	96.94	*NA	Weka Tool
Abdul Samad et al. [36]	Random Forest (RF)	97.74	97.50	*NA	Tuned Hyperparameters
Al-Sarem et al. [51]	Optimal Stacking Ensemble Method	98.58	97.16	*NA	*NA
Abdul Samad et al. [36]	Gradient Boosting (GB)	97.98	97.08	*NA	Tuned Hyperparameters
Abdul Samad et al. [36]	XGBoost (XGB)	98.21 (98.26)	–(97.182)	*NA	Tuned Hyperparameters
Alsariera et al. [47]	Boosted BF Tree	98.37	97.09	*NA	Weka Tool
Alsariera et al. [47]	Support Vector Machine (SVM)	91.49	94.50	*NA	Weka Tool
Khan et al. [48]	RF, ANN	–	97.00	*NA	*NA
Salihovic et al. [49]	RF with Ranker + Principal Component Optimization	–	97.33	*NA	Weka Tool
Salihovic et al. [49]	RF with BestFirst + CfsSubsetEval Optimization	–	94.24	*NA	Weka Tool
Hutchinson et al. [50]	RF – Feature Set E	–	96.50	*NA	*NA
Present study	**Proposed HyDNN**	**98.87**	**97.63**	**190**	**Google Colab**

* Not Available in the article.

4-12-1- Comparative Performance Evaluation

A comprehensive comparison of the proposed approach with existing state-of-the-art methods is presented using the results summarized in Table 8. Previous studies employing traditional machine learning and ensemble techniques—such as Boosted NB Tree, Random Forest, Gradient Boosting, XGBoost, and optimized stacking ensembles—achieve competitive accuracy ranging from 94.50% to 98.38% across Datasets 1 and 2. However, many of these methods rely heavily on handcrafted features, extensive hyperparameter tuning, or external optimization strategies, and most do not report computational cost or training time.

In contrast, the proposed HyDNN model achieves a higher accuracy of 98.87% on Dataset 1, outperforming previously reported methods while maintaining a reasonable training time of 190 seconds on Google Colab. The hybrid architecture of HyDNN, which integrates CNN-based spatial feature extraction with BiLSTM-based sequential dependency modeling, allows for effective automatic feature learning and improved generalization. This comparison demonstrates that the proposed model not only surpasses prior methods in predictive performance but also provides an efficient and scalable solution for phishing detection.

5- Conclusion

Phishing websites are a prevalent form of social engineering attack that target users with the intent of stealing sensitive information such as login credentials, financial data, and personal details. This study provides a comprehensive evaluation of various feature selection and dimensionality reduction techniques specifically tailored for phishing website detection. The focus of these methods is to identify the most influential and informative features, thereby improving the efficiency and accuracy of the detection models.

The research presents extensive experiments across three distinct datasets, employing multiple feature selection methods. Among these, the Random Forest feature selector with feature importance scoring emerged as the most effective, consistently identifying the features most relevant to phishing detection. Experimental results demonstrated that a hyperparameter-tuned Random Forest classifier achieved the highest accuracy among traditional machine learning models. Consequently, it was adopted as the baseline model for comparison with more complex deep learning approaches.

In addition to classical machine learning models, this study investigates hybrid deep learning strategies to enhance phishing detection performance. Using Random Forest as the baseline, deep learning architectures such as CNN, LSTM, and bidirectional LSTM (BiLSTM) were ensembled and applied to the UCI, Mendeley, and custom datasets. The proposed hybrid model, HyDNN, consists of two input branches: one with three CNN layers followed by batch normalization and ReLU activation, and the other with three BiLSTM layers using SELU activation, LeCun normal kernel initialization, and the Adam optimizer. Trained for 200 epochs, HyDNN outperformed the baseline Random Forest classifier, demonstrating superior accuracy and robustness across datasets.

Despite its strong performance, the hybrid deep learning approach requires substantial computational resources, such as GPUs or TPUs, for efficient execution. Future research could explore reinforcement learning or other adaptive strategies to further improve phishing detection. Overall, this study highlights the effectiveness of combining rigorous feature selection with hybrid deep learning models for reliable and high-performance phishing website detection.

6- Declarations

6-1- Author Contributions

Conceptualization, D.J.D. and A.P.R.; methodology, D.J.D. and A.P.R.; software, D.J.D.; validation, R.F., V.P., and M.S.Y.; formal analysis, A.P.R.; investigation, D.J.D.; resources, R.F.; data curation, V.P.; writing—original draft preparation, D.J.D.; writing—review and editing, A.P.R.; visualization, R.F. and V.P.; supervision, A.P.R.; project administration, M.S.Y. All authors have read and agreed to the published version of the manuscript.

6-2- Data Availability Statement

To support reproducibility, the data and the code presented in this study have now been made publicly accessible and are available at the *Kaggle* [45, 46].

6-3- Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6-4- Acknowledgments

We acknowledge the facility provided by NMAM Institute of Technology, NITTE (Deemed to be University) to carry out the experiment.

6-5- Institutional Review Board Statement

Not applicable.

6-6- Informed Consent Statement

Not applicable.

6-7- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

7- References

[1] Ali, M. M., & Mohd Zaharon, N. F. (2024). Phishing—A Cyber Fraud: The Types, Implications and Governance. International Journal of Educational Reform, 33(1), 101–121. doi:10.1177/10567879221082966.

[2] CloudFare. (2023). 2023 Phishing Threats Report. CloudFare, San Francisco, United States. Available online: https://cf-assets.www.cloudflare.com/slt3lc6tev37/17i1am3UMltoOE7UOU1neX/5776ab933524b3cf99ee85eaaedc76f6/BDES-4838_2023_Phishing-Threats-Report-2023.pdf (accessed on May 2026).

[3] Skula, I., & Kvet, M. (2023). Domain Blacklist Efficacy for Phishing Web-page Detection Over an Extended Time Period. 2023 33rd Conference of Open Innovations Association (FRUCT), 257–263. doi:10.23919/FRUCT58615.2023.10142999.

[4] IC3 (2026). Annual Reports, Internet Crime Complaint Center (IC3), Washington, United States. Available online: https://www.ic3.gov/AnnualReport/Reports (accessed on May 2026).

[5] Almseidin, M., Zuraiq, A. A., Al-Kasassbeh, M., & Alnidami, N. (2019). Phishing detection based on machine learning and feature selection methods. International Association of Online Engineering, Vienna, Austria.

[6] Patil, D. R., & Patil, J. B. (2016). Malicious Web Pages Detection Using Static Analysis of URLs. International Journal of Information Security and Cybercrime, 5(2), 57–70. doi:10.19107/ijisc.2016.02.06.

[7] Alsenani, T. R., Ayon, S. I., Yousuf, S. M., Anik, F. B. K., & Chowdhury, M. E. S. (2023). Intelligent feature selection model based on particle swarm optimization to detect phishing websites. Multimedia Tools and Applications, 82(29), 44943–44975. doi:10.1007/s11042-023-15399-6.

[8] Qi, Q., Wang, Z., Xu, Y., Fang, Y., & Wang, C. (2023). Enhancing Phishing Email Detection through Ensemble Learning and Undersampling. Applied Sciences (Switzerland), 13(15), 8756. doi:10.3390/app13158756.

[9] Tudosi, A.-D., Graur, A., Balan, D. G., & Potorac, A. D. (2023). An Email Classification Framework for Phishing Detection in Virtualized Network Environments. 2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet), 1–5. doi:10.1109/RoEduNet60162.2023.10274915.

[10] Wang, J. (2022). An Improved Genetic Algorithm for Web Phishing Detection Feature Selection. 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML), 130–134. doi:10.1109/CACML55074.2022.00029.

[11] Alani, M. M., & Tawfik, H. (2022). PhishNot: A Cloud-Based Machine-Learning Approach to Phishing URL Detection. Computer Networks, 218, 109407. doi:10.1016/j.comnet.2022.109407.

[12] Maennel, O. M., & Matulevicius, R. A. (2026). New Heuristic Based Phishing Detection Approach Utilizing Selenium Web-driver. GitHub, Inc. Available online: https://github.com/nf1s/selenium_phishing_detector (accessed on May 2026).

[13] Dadkhah, M., Shamshirband, S., & Wahab, A. W. A. (2016). A hybrid approach for phishing web site detection. Electronic Library, 34(6), 927–944. doi:10.1108/EL-07-2015-0132.

[14] Chandrasegar, T., & Viswanathan, P. (2019). Dimensionality reduction of a phishing attack using decision tree classifier. 2019 Innovations in Power and Advanced Computing Technologies (I-PACT), 1–4. doi:10.1109/i-PACT44901.2019.8960117.

[15] Calzarossa, M. C., Giudici, P., & Zieni, R. (2024). Explainable machine learning for phishing feature detection. Quality and Reliability Engineering International, 40(1), 362–373. doi:10.1002/qre.3411.

[16] Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing Detection System Through Hybrid Machine Learning Based on URL. IEEE Access, 11, 36805–36822. doi:10.1109/access.2023.3252366.

[17] Pandey, P., & Mishra, N. (2023). Phish-Sight: a new approach for phishing detection using dominant colors on web pages and machine learning. International Journal of Information Security, 22(4), 881–891. doi:10.1007/s10207-023-00672-4.

[18] Basit, A., Zafar, M., Javed, A. R., & Jalil, Z. (2020). A Novel Ensemble Machine Learning Method to Detect Phishing Attack. 2020 IEEE 23rd International Multitopic Conference (INMIC), 1–5. doi:10.1109/INMIC50486.2020.9318210.

[19] Adewole, K. S., Akintola, A. G., Salihu, S. A., Faruk, N., & Jimoh, R. G. (2019). Hybrid Rule-Based Model for Phishing URLs Detection. Emerging Technologies in Computing, iCETiC 2019, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, Cham, Switzerlsnd. doi:10.1007/978-3-030-23943-5_9.

[20] Dsouza, D. J., Rodrigues, A. P., & Fernandes, R. (2024). Multi-modal comparative analysis on execution of phishing detection using artificial intelligence. IEEE Access, 12, 163016-163041. doi:10.1109/ACCESS.2024.3491429.

[21] Soosai Anandaraj, A. P., Ramesh, P. S., Arifulla, S., Madhuri, C. G., & Yeshwanth Malepati, N. (2024). Phishing Detection System Through Hybrid Machine Learning Based on URL. 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), 1–7. doi:10.1109/ICEEICT61591.2024.10718484.

[22] Catal, C., Giray, G., Tekinerdogan, B., Kumar, S., & Shukla, S. (2022). Applications of deep learning for phishing detection: a systematic literature review. Knowledge and Information Systems, 64(6), 1457–1500. doi:10.1007/s10115-022-01672-x.

[23] Do, N. Q., Selamat, A., Krejcar, O., Herrera-Viedma, E., & Fujita, H. (2022). Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions. IEEE Access, 10, 36429–36463. doi:10.1109/ACCESS.2022.3151903.

[24] Ozcan, A., Catal, C., Donmez, E., & Senturk, B. (2023). A hybrid DNN–LSTM model for detecting phishing URLs. Neural Computing and Applications, 35(7), 4957–4973. doi:10.1007/s00521-021-06401-z.

[25] Prabakaran, M. K., Meenakshi Sundaram, P., & Chandrasekar, A. D. (2023). An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders. IET Information Security, 17(3), 423–440. doi:10.1049/ise2.12106.

[26] Nguyen, M., Nguyen, T., & Nguyen, T. H. (2018). A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing. arXiv Preprint, arXiv:1805.01554. doi:10.48550/arXiv.1805.01554.

[27] Ariyadasa, S., Fernando, S., & Fernando, S. (2024). SmartiPhish: a reinforcement learning-based intelligent anti-phishing solution to detect spoofed website attacks. International Journal of Information Security, 23(2), 1055–1076. doi:10.1007/s10207-023-00778-9.

[28] Bountakas, P., & Xenakis, C. (2023). HELPHED: Hybrid Ensemble Learning Phishing Email Detection. Journal of Network and Computer Applications, 210, 103545. doi:10.1016/j.jnca.2022.103545.

[29] Al-Sarem, M., Saeed, F., Al-Mekhlafi, Z. G., Mohammed, B. A., Al-Hadhrami, T., Alshammari, M. T., Alreshidi, A., & Alshammari, T. S. (2021). An optimized stacking ensemble model for phishing websites detection. Electronics (Switzerland), 10(11), 1285. doi:10.3390/electronics10111285.

[30] Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. Q. (2024). The application of a novel neural network in the detection of phishing websites. Journal of Ambient Intelligence and Humanized Computing, 15(3), 1865–1879. doi:10.1007/s12652-018-0786-3.

[31] Sahingoz, O. K., Buber, E., & Kugu, E. (2024). DEPHIDES: Deep Learning Based Phishing Detection System. IEEE Access, 12, 8052–8070. doi:10.1109/ACCESS.2024.3352629.

[32] Kritika, E. (2025). A comprehensive literature review on ransomware detection using deep learning. Cyber Security and Applications, 3(4), 315–343. doi:10.1016/j.csa.2024.100078.

[33] Korkmaz, M., Kocyigit, E., Sahingoz, O. K., & Diri, B. (2022). A Hybrid Phishing Detection System Using Deep Learning-based URL and Content Analysis. Elektronika Ir Elektrotechnika, 28(5), 80–89. doi:10.5755/j02.eie.31197.

[34] Zhou, J., Cui, H., Li, X., Yang, W., & Wu, X. (2023). A Novel Phishing Website Detection Model Based on LightGBM and Domain Name Features. Symmetry, 15(1), 180. doi:10.3390/sym15010180.

[35] Bahaghighat, M., Ghasemi, M., & Ozen, F. (2023). A high-accuracy phishing website detection method based on machine learning. Journal of Information Security and Applications, 77, 103553. doi:10.1016/j.jisa.2023.103553.

[36] Abdul Samad, S. R., Balasubaramanian, S., Al-Kaabi, A. S., Sharma, B., Chowdhury, S., Mehbodniya, A., Webber, J. L., & Bostani, A. (2023). Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection. Electronics (Switzerland), 12(7), 1642. doi:10.3390/electronics12071642.

[37] Elsadig, M., Ibrahim, A. O., Basheer, S., Alohali, M. A., Alshunaifi, S., Alqahtani, H., Alharbi, N., & Nagmeldin, W. (2022). Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction. Electronics (Switzerland), 11(22), 3647. doi:10.3390/electronics11223647.

[38] Ramana, A. V., Rao, K. L., & Rao, R. S. (2021). Stop-Phish: an intelligent phishing detection method using feature selection ensemble. Social Network Analysis and Mining, 11(1), 110. doi:10.1007/s13278-021-00829-w.

[39] Mourtaji, Y., Bouhorma, M., Alghazzawi, D., Aldabbagh, G., & Alghamdi, A. (2021). Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network. Wireless Communications and Mobile Computing, 2021(1), 8241104. doi:10.1155/2021/8241104.

[40] Wurzenberger, M., Höld, G., Landauer, M., & Skopik, F. (2024). Analysis of statistical properties of variables in log data for advanced anomaly detection in cyber security. Computers & Security, 137, 103631. doi:10.1016/j.cose.2023.103631.

[41] Yoosuf, M. S., Vijaya, P., & Mani, J. (2025). DML-IDS: Distributed Multi-Layer Intrusion Detection System for Securing Healthcare Infrastructure. Emerging Science Journal, 9(6), 3157–3173. doi:10.28991/ESJ-2025-09-06-016.

[42] Mohammad, R. M., Thabtah, F., & McCluskey, L. (2016). An improved self-structuring neural network. Trends and Applications in Knowledge Discovery and Data Mining, PAKDD 2016, Lecture Notes in Computer Science, Springer, Cham, Switzerland. doi:10.1007/978-3-319-42996-0_4.

[43] Frank, A. (2010). UCI machine learning repository. University of California, Irvine, United States. Available online: https://archive.ics.uci.edu/ (accessed on May 2026).

[44] Erickson, B. J., & Kitamura, F. (2021). Magician's corner: 9. performance metrics for machine learning models. Radiology: Artificial Intelligence, 3(3), 200126. doi:10.1148/ryai.2021200126.

[45] Kumar, A. (2024). Phishing Website Dataset. Kaggle, San Francisco, United States. Available online: https://www.kaggle.com/datasets/akashkr/phishing-website-dataset (accessed on May 2026).

[46] Dsouza, D. J. (2024). Phising. Kaggle, San Francisco, United States. Available online: https://www.kaggle.com/datasets/divyajenniferdsouza1/phishing (accessed on May 2026).

[47] Alsariera, Y. A., Balogun, A. O., Adeyemo, V. E., Tarawneh, O. H., & Mojeed, H. A. (2022). Intelligent Tree-Based Ensemble Approaches for Phishing Website Detection. Journal of Engineering Science and Technology, 17(1), 563–582.

[48] Khan, S. A., Khan, W., & Hussain, A. (2020). Phishing Attacks and Websites Classification Using Machine Learning and Multiple Datasets (A Comparative Analysis). Intelligent Computing Methodologies, 301–313. doi:10.1007/978-3-030-60796-8_26.

[49] Salihovic, I., Serdarevic, H., & Kevric, J. (2018). The Role of Feature Selection in Machine Learning for Detection of Spam and Phishing Attacks. Advanced Technologies, Systems, and Applications III, 476–483. doi:10.1007/978-3-030-02577-9_47

[50] Hutchinson, S., Zhang, Z., & Liu, Q. (2018). Detecting Phishing Websites with Random Forest. Machine Learning and Intelligent Communications, 470–479. doi:10.1007/978-3-030-00557-3_46.

[51] Al-Sarem, M., Saeed, F., Al-Mekhlafi, Z. G., Mohammed, B. A., Al-Hadhrami, T., Alshammari, M. T., Alreshidi, A., & Alshammari, T. S. (2021). An Optimized Stacking Ensemble Model for Phishing Websites Detection. Electronics, 10(11), 1285. doi:10.3390/electronics10111285.