JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
https://e-journal.uum.edu.my/index.php/jict

How to cite this article:
Yeop, N. S., Zakaria, N. H., & Suendri (2025). Enhancing the effectiveness of machine learning-based phishing email detection via an improved pre-processing technique for data security. *Journal of Information and Communication Technology*, *24*(4), 87-110. https://doi.org/10.32890/jict2025.24.4.4

Enhancing the Effectiveness of Machine Learning-Based Phishing Email Detection via an Improved Pre-Processing Technique for Data Security

[1]Nur Suhaila Yeop, [2]Nur Haryani Zakaria & [3]Suendri
[1&2]School of Computing, Universiti Utara Malaysia, Malaysia
[3]Universitas Islam Negeri Sumatera Utara, Indonesia

[*1]suhailayeop@gmail.com
[2]haryani@uum.edu.my
[3]suendri@uinsu.ac.id
*Corresponding author

Received: 31/1/2025 Revised: 25/6/2025 Accepted: 31/8/2025 Published: 31/10/2025

ABSTRACT

The growing volume and sophistication of phishing emails have become a significant threat to data security, often serving as the initial vector for data breaches. Most past studies have focused on comparing machine learning models to determine the best-performing algorithm. They often neglect the role of pre-processing, which also contributes to the effectiveness of these models. To address this gap, this study investigates the impact of pre-processing techniques on phishing email detection, aiming to strengthen data protection. Three supervised machine learning algorithms, which are Support Vector Machine (SVM), Random Forest and Decision Tree, were selected to undergo two experimental iterations: one with basic pre-processing and the other with an enhanced pre-processing technique including Synthetic Minority Oversampling Technique (SMOTE), Term Frequency-Inverse Document Frequency (TF-IDF), Singular Value Decomposition (SVD) and cross-validation. Using a dataset comprising 28,747 labelled emails, the models were trained, tested, and evaluated based on accuracy, precision, recall, and F1-score, with further insight gained through confusion matrix analysis. Among the models, Random Forest demonstrated the strongest consistent performance across all metrics, while Decision Tree showed the most notable improvement. Although SVM maintains high recall and precision, it is less responsive to the applied pre-processing techniques. This result demonstrates that pre-processing techniques significantly contribute to the performance of the detection models. Overall, these findings highlight the critical role of pre-processing in enhancing phishing email detection, which contributes to stronger organisational resilience.

Keywords: Data protection, machine learning, phishing email detection, pre-processing, supervised learning.

INTRODUCTION

In modern society, email is considered a crucial tool for facilitating interaction, especially when distance and cost become constraints. It has become a centre of collaboration, since it allows users to keep track of records and provides access to various digital services. According to The Radicati Group, Inc. (2023), a total of 347 billion emails are expected to be sent daily, rising to 408 billion by 2027. This growth has undoubtedly brought huge convenience to both personal and professional areas. However, the heavy reliance on email as a communication medium has made it a primary target for cyberattacks, particularly phishing threats that continue to evolve rapidly in both complexity and scale each year. This concern is supported by a report from the Anti-Phishing Working Group (2025), which shows that threat actors have adapted to new technologies and can now bypass traditional defences, as evidenced by the increasing number of phishing attacks in recent years. Currently, this threat is capable of mimicking legitimate interfaces with high accuracy, making it difficult for end users to detect.

The phishing attack is one of the most common techniques used by attackers as an entry point to gain access to a victim's system, posing a significant threat to data privacy. Studies by Gualberto (2020) and Tanti (2024) explain that to deceive victims and obtain their confidential information, cybercriminals impersonate authorised personnel from financial institutions or trusted service providers. Another study from Alkhalil (2021) further elaborates that a phishing attack commonly involves a deceptive message attached with malicious links embedded inside the email. It is claimed to come from a legitimate source with the intention of luring victims into clicking it and redirecting them to a fraudulent website. Once there, the victim may unknowingly enter their private data, such as banking credentials, personal details, and passwords, since they are unaware that this platform is designed to collect sensitive information. As a result, this cunning strategy can lead to a severe financial loss, system compromise and prolonged reputational damage for both individuals and organisations. Pepco Group is a real-life example that illustrates the impact of phishing attacks, having suffered data breaches and operational disruptions due to a single deceptive email, as reported by Kovacs (2024).

Blacklists and heuristic filters are among the traditional methods that face limitations when it comes to identifying phishing attacks, as these techniques rely greatly on human intervention and pre-existing patterns. A study by Hina et al. (2021) noted that these methods, while able to flag threats, are not equipped to recognise new, emerging threats, thus leaving users vulnerable to zero-day attacks. This constraint underscores the urgent need for adaptive, automated, and flexible solutions to accurately detect threats in real-time. Fortunately, machine learning (ML) offers adaptive, automated, and flexible solutions that accurately detect threats in real-time. As emphasised by Chy (2024), even when phishing attempts employ novel tactics, ML techniques can rapidly identify them by learning from historical data, detecting patterns, and recognising anomalies that indicate phishing behaviour. Furthermore, ML analyses subtle cues in language, metadata, or structure that often go unnoticed by static filters, which contributes towards improving early threat detection and reducing the likelihood of false positives. This study aims to contribute to the broader goal of enhancing cybersecurity resilience by strengthening the data protection mechanism against phishing threats. The strength of data protection is achieved through an enhanced pre-processing approach, where aspects such as data balancing and transformation are applied to the raw dataset to prepare the model to detect phishing attempts more effectively.

This study focuses on improving pre-processing techniques for phishing email detection in ML rather than developing new detection models. This decision is grounded in the understanding that the quality of data preparation has a significant impact on model performance. In many real-world scenarios, raw data is noisy, high-dimensional, and imbalanced, which can lead to poor classification results even when

using advanced algorithms. By enhancing the pre-processing pipeline, the study aims to ensure that the data fed into the model is cleaner, more informative and better structured. Moreover, existing ML algorithms such as Support Vector Machine (SVM), Random Forest, and Decision Tree have already demonstrated strong performance in various classification tasks, including phishing detection (Habib et al., 2022). Therefore, rather than reinventing the model architecture, this study seeks to maximise the potential of existing models by optimising the data they learn from. This approach is both practical and impactful, as it addresses a commonly overlooked yet critical stage in the ML workflow.

This paper is organised as follows: the second section will entail all the related work that supports the relevance of this work. A detailed description of the proposed technique is presented in Section Three. A methodology section follows, discussing all related activities necessary for the research to be carried out. The following section presents the results and findings, which are subsequently discussed to reflect the effectiveness of the proposed technique. Finally, this paper concludes by summarising the key findings and highlighting potential future work for consideration.

RELATED WORK

Since ML has surpassed traditional methods by offering adaptive and automated solutions, it has become an essential tool in detecting phishing emails. With the ability to learn from vast datasets, it can identify subtle patterns, such as unusual email headers, domain inconsistencies, and writing styles that may indicate phishing attempts. Nowadays, advanced techniques like Deep Learning (DL) and Natural Language Processing (NLP) are preferable because they are continually improving and have shown promise in the phishing detection field due to their superior semantic analysis capabilities. Despite so, both techniques require extensive computational resources and large sets of labelled datasets. This requirement is often impractical for small to medium-sized organisations, particularly those without a dedicated IT security team, due to the high associated costs. Therefore, the traditional ML model remains relevant in the context of this study and is deemed a suitable approach to be widely adopted across various domains in real-world implementations, whether in cases of phishing detection, intrusion monitoring, or spam filtering (Hussin et al., 2024).

Additionally, Hussin et al. (2024) confirmed that this approach strikes a balance between performance and resource efficiency, making it more practical in resource-constrained environments. Additionally, compared to DL and NLP, ML is more transparent in its decision-making processes, as the results are interpretable and can be easily understood by a human analyst (Rudin, 2019). This transparency is important in cybersecurity applications for operational and regulatory reasons. The implementation of ML in phishing detection, as seen in several studies, has provided meaningful insights related to the methods and tools that enhance classification performance, as illustrated in Table 1.

Table 1

Summary of ML Methods and Tools for Phishing Detection across Selected Studies

Authors	Harikrishnan et al. (2018)	Mohey and Mohsen (2021)	Habib et al. (2022)	Shejole et al. (2023)	Reddy et al. (2023)
Pre-processing Technique	• Numerical representation (TF-IDF) • Dimensional reduction: SVD/Non-negative Matrix Factorisation (NMF)	• Min-Max scaling • Feature selection (stemming, stopword removal, correlation removal)	• Feature scaling	• Tokenisation • Stopword removal • Stemming • Feature extraction (TF-IDF)	• Data cleaning • Feature selection • Applied WEKA filter "Remove Misclassified"
Dataset Size	PhishingCorpus dataset (≈10,000 emails: 4,082 legit + 501 phishing with header; 5,088 legit + 612 phishing no header)	Spam dataset from Kaggle (4601 emails, 58 attributes)	Dataset from Kaggle (11,504 instances, 32 attributes)	Email spam dataset from Kaggle (size not specified, labelled spam/ham)	Phishing dataset (88,489 instances, 126 attributes
Algorithm Used	• Decision Tree • Random Forest • AdaBoost • K-Nearest Neighbours (KNN) • Logistic Regression • Naïve Bayes • SVM	• SVM • Naïve Bayes • Decision Tree • KNN	• KNN • Logistic Regression • Random Forest	• SVM • Naïve Bayes • Decision Tree • Random Forest, KNN	• Random Forest • Naïve Bayes (via WEKA)
Evaluation Metric	• Accuracy • Precision • Recall • F1-score	• Accuracy • Confusion Matrix (10-fold cross-validation)	• Precision • Recall • F1-score • Receiver Operating Characteristic (ROC) Curve	• Accuracy • Precision • Recall • F1-score	• Accuracy (with RF: 99.03%)
Key Findings	High accuracy (up to 99.9% with Random Forest/SVM on TF-IDF + SVD/NMF)	SVM achieved the best accuracy (~91.2%)	Random Forest had the best performance (Precision 97%, AUC=1.0)	SVM achieved the highest accuracy (98.47%), Random Forest ~97%	Random Forest achieved 99.03% accuracy after pre-processing filters

(continued)

Authors	Harikrishnan et al. (2018)	Mohey and Mohsen (2021)	Habib et al. (2022)	Shejole et al. (2023)	Reddy et al. (2023)
Limitation	• Dataset imbalanced • Noted overfitting issues • No external data	• Focused mainly on basic pre-processing • No advanced imbalance handling or dimensionality reduction	• Limited feature engineering	• Relied only on TF-IDF • No handling of class imbalance or dimensionality reduction	• Limited algorithm diversity • Evaluation mainly accuracy • Dataset-specific filters used

Based on Table 1, Harikrishnan et al. (2018) applied feature extraction using the Term Frequency-Inverse Document Frequency (TF-IDF) technique to emphasise significant words in messages along with dimensionality reduction using Singular Value Decomposition (SVD) and Non-negative Matrix Factorisation (NMF). The results showed that algorithms such as Decision Trees and Random Forests excel in both accuracy and stability. In addition, Mohey and Mohsen (2021) also utilise the same techniques to investigate simpler textual features, such as word and character frequency counts. Their study reported that SVM achieved the highest classification accuracy among the models evaluated. Overall, both studies emphasise the importance of selecting effective feature extraction and dimensionality reduction techniques to enable models to focus on the most informative aspects while minimising noise in the dataset.

Meanwhile, a study from Shejole et al. (2023) focused on optimising each model's robustness through the pre-processing phase. TF-IDF was used as a numerical representation of email content to evaluate ML models such as SVM, Naïve Bayes, Decision Tree, K-Nearest Neighbour, and Random Forest. The final result revealed that SVM achieved the best classification performance, closely followed by Random Forest. It demonstrates the reliability of the models used in distinguishing phishing messages from legitimate ones. Moreover, Reddy et al. (2023) utilise the existing filtering feature inside the WEKA tool to eliminate misclassified data instances during training. The outcome yielded an accuracy of 99.03% for Random Forest, confirming the potential of pre-processing strategies in increasing the reliability of the model's effectiveness. A further study by Habib et al. (2022) examined content-based analysis of structural features by utilising unified resource locator (URL)-based feature extraction on a dataset comprising both phishing and legitimate websites. This method was applied to K-Nearest Neighbours, Logistic Regression, and Random Forest, where Random Forest emerged as the top-performing model, achieving 97% precision and an Area Under the Curve (AUC) score of 1.0 due to its strong accuracy.

Despite the impressive outcomes as discussed above, notable gaps remain in the existing literature. The selected researchers in Table 1 have trained and tested different kinds of models to find which one achieves the highest accuracy, thereby proving its effectiveness. Unfortunately, those approaches tend to overlook the role of the pre-processing phase, which plays an important role in determining the reliability of a model (Konda, 2022). Most prior work has applied only basic pre-processing techniques, such as data cleaning, without exploring improvements like resampling for class imbalance or dimensionality reduction (Felix & Lee, 2019). The raw datasets are often inconsistent, as the quality of them can vary, and this input will directly affect the model's performance in learning about the pattern. Felix and Lee (2019) noted that the pre-processing phase is the one that cleans, balances, and modifies the dataset's structure so that the model can understand it easily. A well-preprocessed dataset enhances

the model's ability to classify phishing emails accurately. This method will significantly contribute to model performance and directly support the overall goal, thereby enhancing data protection.

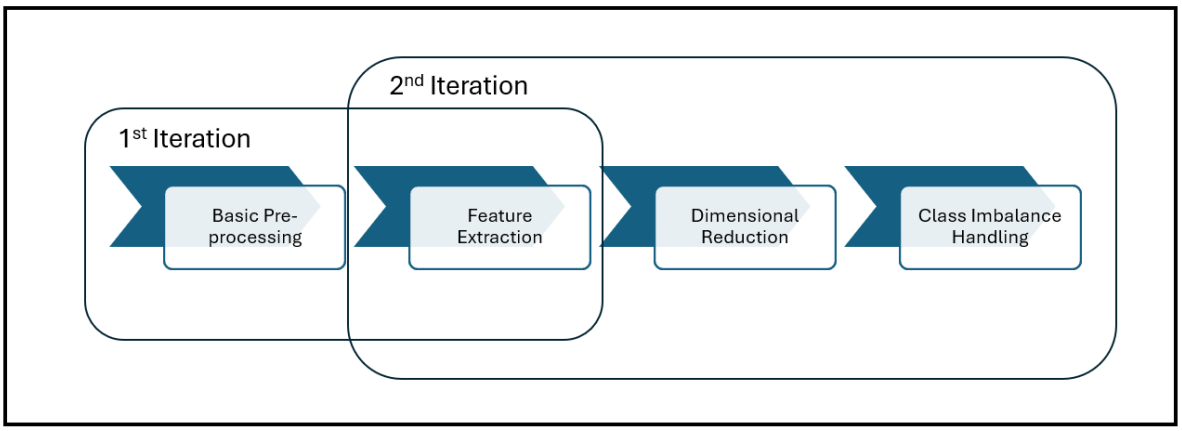
Therefore, this study intends to enhance the effectiveness of ML-based phishing email detection via improved pre-processing techniques. The improvement is implemented through an iterative approach, whereby the first iteration establishes baseline models using standard pre-processing methods. In contrast, the second iteration employs advanced techniques, including the Synthetic Minority Oversampling Technique (SMOTE), TF-IDF, and SVD, to assess their impact on model performance. Furthermore, unlike most prior work that only uses evaluation metrics, this study employed the confusion matrix to provide a transparent view and a detailed analysis of how such models classify between phishing and legitimate emails.

THE PROPOSED PRE-PROCESSING TECHNIQUE

Pre-processing plays a vital role in the success of any ML pipeline, especially when dealing with real-world data that is often messy, imbalanced, and high-dimensional. In this study, the proposed pre-processing technique was designed to systematically prepare the dataset for accurate and robust classification through a two-phase iterative approach. Each phase addresses a specific challenge commonly found in raw datasets, as illustrated in Figure 1.

Figure 1

The Proposed Pre-Processing Technique



First Iteration

Basic Pre-processing

The first iteration involved the basic pre-processing, which plays a crucial role in preparing raw email data for further analysis and classification (Al-Yozbaky & Alanezi, 2023). It begins with normalising the text, which generally involves the process of cleaning and standardising text data so that different forms of the same word or structure are treated equally, thereby reducing noise and improving the quality of features used for detection. Specifically, to conduct text normalisation, the relevant content was extracted from the email, such as the subject, body, and header fields. Since many phishing emails

are formatted using Hypertext Markup Language (HTML), the first step often involves removing HTML tags and scripts to obtain clean, plain text. Then, the text is converted to lowercase to ensure consistency and reduce duplication caused by letter casing. Punctuation marks, special characters, and symbols are removed to minimise noise, except when these characters are explicitly included as potential phishing indicators.

The cleaned text is then tokenised, whereby it is split into individual words, which is then followed by the removal of common stopwords (e.g., “the”, “and”, “is”). The purpose of doing this is to remove text which do not contribute meaningful information to the detection model. Next, stemming or lemmatisation is then applied to reduce words to their root forms, such as converting “running” to “run” or “better” to “good,” for the purpose of standardising word variants. In addition to text normalisation, pre-processing also includes the identification and extraction of URLs, Internet Protocol (IP) addresses, and email addresses found within the email content. These elements may either be removed, replaced with placeholders, or used as features, as phishing emails often contain suspicious links or spoofed sender details.

Feature Extraction

Once the email text has been normalised, the next step is to proceed with the feature extraction process. This involves grouping common features like URL-based attributes. For example, the number of URLs present in the message, whether any URLs contain IP addresses instead of domain names, the presence of shortened URLs (e.g., using bit.ly), or the average length of the URLs. This process is done considering that phishing emails typically rely heavily on misleading or obfuscated links, making these features highly relevant (Ahmed et al., 2024). Besides URL-based attributes, header and sender-related features are also critical in phishing detection. These include the sender’s domain, mismatches between the display name and actual email address, inconsistencies between the "From" and "Reply-To" fields and any anomalies in the "Return-Path" header. These inconsistencies often indicate attempts to impersonate trusted entities. In addition, other text features may include word count, character count, the number of uppercase words, excessive use of exclamation marks and the presence of spammy keywords or phrases like "urgent", "verify your account", or "click here." These features help to identify the aggressive tone or urgency typically found in phishing attempts.

Furthermore, lexical features such as the average word length and the ratio of unique words and common n-grams (i.e., frequent word pairs or triplets) can be extracted to capture linguistic patterns unique to phishing emails. Lexical features are crucial in feature extraction for phishing email detection because they capture surface-level characteristics of the text, such as URL structure, word length, special character usage, and the presence of suspicious keywords that often differ between phishing and legitimate emails. These features are language-independent, easy to compute, and can reveal deceptive patterns, such as obfuscated URLs, urgent language, or excessive punctuation, commonly used in phishing attempts. By analysing these patterns, lexical features provide a fast and effective way for machine learning models to distinguish between malicious and benign messages without requiring deep semantic understanding.

In addition to lexical features, temporal and behavioural features are also crucial in enhancing the accuracy and robustness of phishing email detection systems by capturing suspicious sending patterns, timing anomalies and deviations from normal user or sender behaviour that lexical analysis alone may miss. It includes the time the email was sent, whether it was sent on a weekend, or discrepancies in time zones between sender and receiver, which can also provide meaningful signals, since phishing emails

are often sent at unusual times to evade detection. While lexical features examine the content of the email, temporal and behavioural features analyse the context and timing of the message.

After completing the feature extraction process, the raw email data has been transformed into structured and meaningful representations that machine learning models can effectively use. By considering both lexical and temporal features, they provide a comprehensive view of both the content and context of emails, enabling the detection system to distinguish between legitimate and phishing messages with greater accuracy (Hamadouche et al., 2024). However, given the potentially high dimensionality of the extracted features, especially when combining multiple feature types, a dimension reduction step is applied next.

Second Iteration

In the second iteration, following feature extraction, the process continues with steps focused on refining the feature set and enhancing dataset quality. This involves reducing feature dimensionality to simplify the data and addressing class imbalance to ensure fair representation of all classes. Each of these steps plays a vital role in improving model performance and will be elaborated on in the following sections.

Dimensional Reduction

After extracting a wide range of features from emails, the resulting feature set can be very large and complex. This high dimensionality poses challenges, including increased computational cost, longer training times and a greater risk of overfitting where the model learns noise rather than meaningful patterns. To address these challenges, the dimensionality reduction step is applied, which is generally done by transforming the prominent feature set into a smaller, more manageable set of features while preserving the most essential information (Malik et al., 2023).

Specifically, the process involves either selecting the most relevant features from the existing set or transforming the features into a new, smaller set that retains the essential information. Techniques such as feature selection use statistical or model-based methods to identify and keep only the features that contribute most significantly to distinguishing phishing emails from legitimate ones. Alternatively, feature extraction methods like SVD create new composite features that capture the underlying patterns in the data while reducing dimensionality. By reducing the number of features, dimensionality reduction speeds up computations, helps prevent the model from fitting noise in the training data, and improves its ability to accurately classify new, unseen emails.

The overall impact of dimensionality reduction helps make the detection system faster and more accurate by removing unnecessary information. It makes it easier to understand which features matter most and allows us to see patterns in the data more clearly. This step also enables the system to work quickly enough to catch phishing emails in real-time. By reducing the number of features, it avoids confusion in the model and can reveal useful connections that improve how features are selected. With the feature space simplified, the next challenge to address is class imbalance. This is to ensure the model can effectively learn from both common legitimate emails and the less frequent phishing emails. This balance is necessary before moving forward to the training phase, where the model learns to distinguish phishing messages with greater reliability.

Class Imbalance Handling

With the feature space simplified through dimensionality reduction, the next important challenge to address is class imbalance. In phishing email detection, the number of legitimate emails typically far exceeds the number of phishing emails. This imbalance can cause the model to be biased toward the majority class, making it less effective at identifying phishing attempts (Kumari et al., 2023). To overcome this, several class imbalance handling techniques are available to be applied, ensuring the model receives a balanced and representative dataset during training.

Common approaches include oversampling, undersampling and synthetic data generation methods. In the oversampling method, additional copies of phishing email samples are generated to increase their representation, while the method reduces the number of legitimate email samples to balance the classes. As for synthetic data generation methods, they create new, artificial phishing samples by interpolating between existing ones (Johnson & Khoshgoftaar, 2019). Each method aims to provide the model with enough phishing examples to learn the patterns effectively without overwhelming the training process.

Without addressing the class imbalance issue, the model might achieve high overall accuracy by classifying most emails as legitimate, but it would miss many phishing attempts, resulting in a high false negative rate. By balancing the classes, the model gains a better understanding of phishing characteristics and patterns. It enhances its ability to accurately identify phishing emails, thereby increasing detection rates and reducing false negatives. Additionally, handling class imbalance often leads to more stable and generalisable models that perform well on unseen data, enhancing the overall robustness and trustworthiness of the phishing detection system.

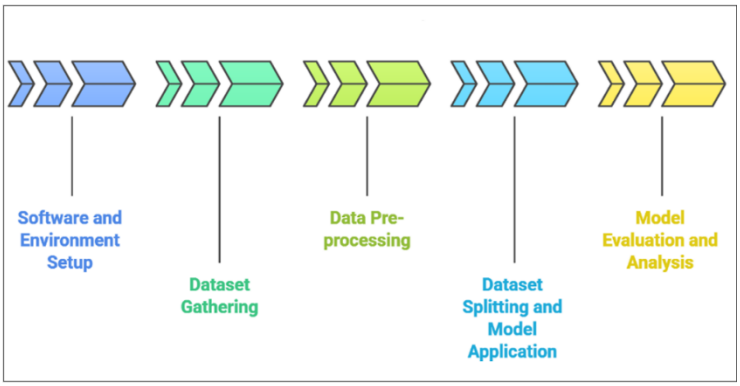
By structuring the pre-processing into these four key steps across two iterations, the study ensures the model is trained on data that is clean, meaningful, and balanced. The first iteration establishes a baseline using fundamental techniques, while the second iteration builds upon this foundation by incorporating more advanced methods such as TF-IDF for feature representation, SVD for dimensionality reduction, and SMOTE for addressing class imbalance. This two-iteration approach not only improves the quality of the input data but also enables a comparative analysis to evaluate how these enhancements affect classification accuracy. The complete pre-processing technique described here was implemented in Phase 3 of the overall methodology, which will be detailed in the following section.

METHODOLOGY

The methodology for this study was divided into several phases, as presented in Figure 2. It begins with setting up the software environment, followed by gathering the dataset, data pre-processing, dataset splitting, and model application, and ends with model evaluation and analysis.

Figure 2

The Methodology for the Experiment



Phase 1: Software and Environment Setup

Before starting the experiment, all the essential software and environment had to be configured to ensure the smoothness of the process. Firstly, the Python language file was downloaded from the official website and installed according to the instructions given. Next, Jupyter Notebook and the necessary libraries, including pandas for data manipulation, sklearn for ML algorithms, matplotlib for data visualisation, and imblearn for handling class imbalance, were installed based on the guidelines provided by the user via the command prompt. To maintain an organised workflow for efficient model management, the project directory was categorised into folders for each different function, such as raw data, processed data, models, and results.

For this experiment, a laptop armed with an 11th Gen Intel Core i5 processor, 16 GB of RAM, and 476 GB of storage was used as the primary hardware platform to provide sufficient computational capacity for training and testing the classification models. This laptop was installed with Jupyter Notebook as the development environment because it supports visualisation and utilises Python as its core programming language, thanks to its extensive libraries. The chosen libraries include Pandas for data pre-processing and manipulation, scikit-learn (sklearn) for implementing ML algorithms and evaluating metrics, Matplotlib for graphical analysis, and imbalanced-learn (imblearn) for addressing class imbalance.

Phase 2: Dataset Gathering

The dataset gathering phase involves searching for the most suitable dataset through online searches and comparing several platforms to ensure it meets the needs of this experiment. Later, the chosen dataset is downloaded, extracted, and stored in the project directory for convenient access throughout the pre-processing and model training. A sufficient data volume is one of the key requirements that must be met to achieve accurate model classification. Thus, a dataset from a known platform called Kaggle was selected due to its medium sample size of 28,747 labelled records, which combine non-phishing and phishing emails, and thorough labelling, all of which are beneficial for the training and testing phases.

Phase 3: Data Pre-processing

As mentioned earlier, the proposed pre-processing technique was done in Phase 3. It is done in a two-iterative approach, which is explained as follows:

The First Iteration

The basic pre-processing step involves handling missing values, removing numeric digits, special characters and other unwanted symbols. This process is essential because raw data often contains inconsistencies, noise or irrelevant information that can mislead ML models or statistical analyses. Handling missing values helps to prevent bias or inaccurate results, while removing unwanted characters ensures that the data is clean, consistent and in a format suitable for further processing. By performing these steps, not only will data quality be improved, but model performance will also be enhanced, ensuring more reliable outcomes from data-driven tasks.

After completing the basic pre-processing step, raw email data is then prepared for ML by transforming both the target labels and the textual content. The transformation begins with converting the email type column into a binary format, where a value of 0 represents a legitimate email and a value of 1 indicates a phishing email. This binary encoding simplifies the classification task and makes it suitable for use with standard ML algorithms that require numerical input. Next, the text data from the email content undergoes normalisation. All characters are converted to lowercase to reduce the size of the vocabulary and eliminate inconsistencies caused by case differences (e.g., “Email” and “email” would be treated as the same word). Following this, the text is tokenised, which means it is split into individual words or tokens. Tokenisation is a crucial step because it breaks down the raw text into structured units that can be quantitatively analysed. These tokens serve as the foundation for the following procedures.

Once the textual data has been tokenised, the next step is to convert it into a numerical format that ML models can understand. It is achieved using the Bag-of-Words (BoW) model. The BoW model transforms the tokenised text into fixed-length numerical vectors by counting the frequency of each word that appears in the text. Essentially, it creates a vocabulary of all unique words in the dataset and represents each document (i.e., email) as a vector, where each element corresponds to the frequency of a specific word in that document. This simple yet effective technique allows textual data to be structured in a way that captures the presence and importance of words, making it suitable for input into ML algorithms.

As previously explained in the proposed technique section, lexical feature extraction further enriches the dataset by capturing specific textual characteristics such as the presence of suspicious keywords (e.g., "urgent," "password," "verify"), the length of the email, and the frequency of special characters or URLs, which are often indicators of phishing attempts. These lexical features offer a deeper insight into the content of the emails, enabling the model to better distinguish between legitimate and phishing messages. In addition to lexical features, temporal features were also extracted to provide contextual information based on time-related patterns. For example, the time an email is sent, the frequency of emails received from the same sender within a short period, or irregular sending times can be strong indicators of phishing behaviour. Incorporating temporal features enables the detection system to consider not only the content of the email but also when and how often it is sent, adding an important dimension to the feature set. Together, lexical and temporal features complement the BoW representation by providing both content-based and behavioural insights, improving the overall effectiveness of the phishing detection model.

The Second Iteration

For the second iteration, the basic pre-processing steps were repeated, with some modifications made to the feature extraction procedure. Instead of the BoW model, the TF-IDF model was utilised. TF-IDF works similarly to the BoW approach, where both techniques convert the text into a numerical value, since computers can only understand data in the form of numbers. However, unlike BoW, Term Frequency (TF) will count how often a word appears in a document, while Inverse Document Frequency (IDF), which stands for, will count how rare that word is across all documents. The output produced an abundance of features, where many of them are redundant, but they tend to exhibit more meaningful information. For example, two emails are below:

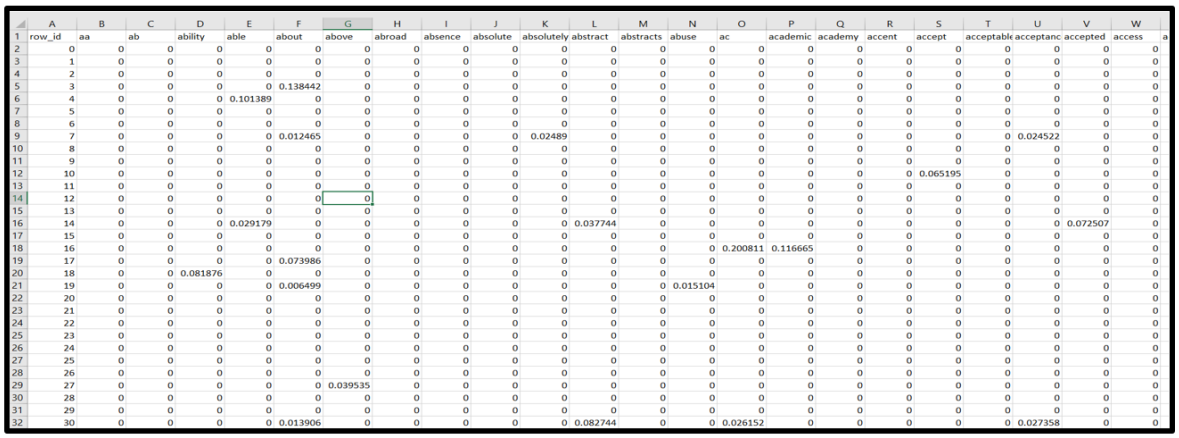
Email A: *Your* account has been *compromised*.
Email B: *Your* package has been delivered.

The word *'your'* appears in both emails, so it will have a low IDF (i.e., it is common across documents). The word *'compromised'* may appear only in phishing emails, making it a high IDF (i.e., rare and more meaningful). Figure 3 illustrates the feature matrix, where each column corresponds to a specific word extracted from the emails, and each row represents an individual email record identified by the row_id column. The values within the cells indicate the importance of each word for that particular email. A value of zero means the word either does not appear in the email or is considered unimportant. Non-zero values signify the presence of the word, with values closer to 1 indicating higher importance. It suggests that the word appears relatively frequently in that specific email but is rare across other emails, highlighting its potential significance for distinguishing phishing content from legitimate messages.

Despite its simplicity and ease of implementation through standard libraries, the TF-IDF approach used to generate these values can create huge feature sets when applied to large collections of emails, due to the large vocabulary size. These large feature sets can slow down processing and increase the model's likelihood of overfitting. It suggests that using dimensionality reduction techniques afterwards is important to simplify the data and improve the model's performance.

Figure 3

Output of Columns After Applying the Feature Extraction Technique

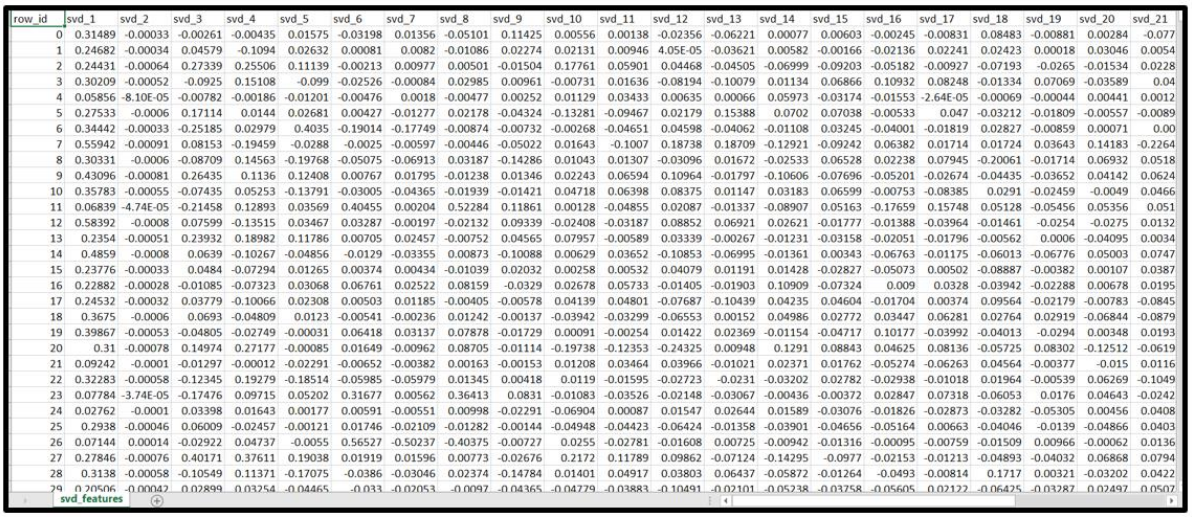


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	row_id	aa	ab	ability	able	about	above	abroad	absence	absolute	absolutely	abstract	abstracts	abuse	ac	academic	academy	accent	accept	acceptable	acceptanc	accepted	access
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	3	0	0	0	0	0.138442	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	4	0	0	0	0.101389	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	7	0	0	0	0	0.012465	0	0	0	0	0.02489	0	0	0	0	0	0	0	0	0	0.024522	0	0
10	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.065195	0	0	0	0
13	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	14	0	0	0	0.029179	0	0	0	0	0	0	0.037744	0	0	0	0	0	0	0	0	0	0.072507	0
17	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0.200811	0.116665	0	0	0	0	0	0	0
19	17	0	0	0	0	0.073986	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	18	0	0	0.081876	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	19	0	0	0	0	0.006499	0	0	0	0	0	0	0	0.015104	0	0	0	0	0	0	0	0	0
22	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	27	0	0	0	0	0	0.039535	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	30	0	0	0	0	0.013906	0	0	0	0	0	0.082744	0	0	0.026152	0	0	0	0	0	0.027358	0	0

TF-IDF looks at each word in an email and gives it a score based on how important that word is in that email compared to others. For example, words like "urgent" or "password" might get higher scores if they appear more in phishing emails. On the other hand, SVD groups related words together into bigger patterns. Instead of checking every single word, SVD finds common themes, such as words related to account security or financial requests, and shows how much that theme appears in the email. If the SVD value is positive, it indicates that the theme is prominent in the email, as evidenced by the co-occurrence of many security-related words. If it is negative, it means the opposite, such as words that suggest a normal, safe email. If the value is close to zero, this indicates that the theme is not really important in that email. Figure 4 illustrates the output of SVD applied to reduce the dimensionality of the space after applying the TF-IDF model.

Figure 4

Output of Columns After Applying Dimensional Reduction Technique



row_id	svd_1	svd_2	svd_3	svd_4	svd_5	svd_6	svd_7	svd_8	svd_9	svd_10	svd_11	svd_12	svd_13	svd_14	svd_15	svd_16	svd_17	svd_18	svd_19	svd_20	svd_21
0	0.31489	-0.00033	-0.00261	-0.00435	0.01575	-0.03198	0.01356	-0.05101	0.11425	0.00556	0.00138	-0.02356	-0.06221	0.00077	0.00603	-0.00245	-0.00831	0.08483	-0.00881	0.00284	-0.077
1	0.24682	-0.00034	0.04579	-0.1094	0.02632	0.00081	0.0082	-0.01086	0.02274	0.02131	0.00946	4.05E-05	-0.03621	0.00582	-0.00166	-0.02136	0.02241	0.02423	0.00018	0.03046	0.0054
2	0.24431	-0.00064	0.27339	0.25506	0.11139	-0.00213	0.00977	0.00501	-0.01504	0.17761	0.05901	0.04468	-0.04505	-0.06999	-0.09203	-0.05182	-0.00927	-0.07193	-0.0265	-0.01534	0.0228
3	0.30209	-0.00052	-0.0925	0.15108	-0.099	-0.02526	-0.00084	0.02985	0.00961	-0.00731	0.01636	-0.08194	-0.10079	0.01134	0.06866	0.10932	0.08248	-0.01334	0.07069	-0.03589	0.04
4	0.05856	-8.10E-05	-0.00782	-0.00186	-0.01201	-0.00476	0.0018	-0.00477	0.00252	0.01129	0.03433	0.00635	0.00066	0.05973	-0.03174	-0.01553	-2.64E-05	-0.00069	-0.00044	0.00441	0.0012
5	0.27533	-0.0006	0.17114	0.0144	0.02681	0.00427	-0.01277	0.02178	-0.04324	-0.13281	-0.09467	0.02179	0.15388	0.0702	0.07038	-0.00533	0.047	-0.03212	-0.01809	-0.00557	-0.0089
6	0.34442	-0.00033	-0.25185	0.02979	0.4035	-0.19014	-0.17749	-0.00874	-0.00732	-0.00268	-0.04651	0.04598	-0.04062	-0.01108	0.03245	-0.04001	-0.01819	0.02827	-0.00859	0.00071	0.00
7	0.55942	-0.00091	0.08153	-0.19459	-0.0288	-0.0025	-0.00597	-0.00446	-0.05022	0.01643	-0.1007	0.18738	0.18709	-0.12921	-0.09242	0.06382	0.01714	0.01724	0.03643	0.14183	-0.2264
8	0.30331	-0.0006	-0.08709	0.14563	-0.19768	-0.05075	-0.06913	0.03187	-0.14286	0.01043	0.01307	-0.03096	0.01672	-0.02533	0.06528	0.02238	0.07945	-0.20061	-0.01714	0.06932	0.0518
9	0.43096	-0.00081	0.26435	0.1136	0.12408	0.00767	0.01795	-0.01238	0.01346	0.02243	0.06594	0.10964	-0.01797	-0.10606	-0.07696	-0.05201	-0.02674	-0.04435	-0.03652	0.04142	0.0624
10	0.35783	-0.00055	-0.07435	0.05253	-0.13791	-0.03005	-0.04365	-0.01939	-0.01421	0.04718	0.06398	0.08375	0.01147	0.03183	0.06599	-0.00753	-0.08385	0.0291	-0.02459	-0.0049	0.0466
11	0.06839	-4.74E-05	-0.21458	0.12893	0.03569	0.40455	0.00204	0.52284	0.11861	0.00128	-0.04855	0.02087	-0.01337	-0.08907	0.05163	-0.17659	0.15748	0.05128	-0.05456	0.05356	0.051
12	0.58392	-0.0008	0.07599	-0.13515	0.03467	0.03287	-0.00197	-0.02132	0.09339	-0.02408	-0.03187	0.08852	0.06921	0.02621	-0.01777	-0.01388	-0.03964	-0.01461	-0.0254	-0.0275	0.0132
13	0.2354	-0.00051	0.23932	0.18982	0.11786	0.00705	0.02457	-0.00752	0.04565	0.07957	-0.00589	0.03339	-0.00267	-0.01231	-0.03158	-0.02051	-0.01796	-0.00562	0.0006	-0.04095	0.0034
14	0.4859	-0.0008	0.0639	-0.10267	-0.04856	-0.0129	-0.03355	0.00873	-0.10088	0.00629	0.03652	-0.10853	-0.06995	-0.01361	0.00343	-0.06763	-0.01175	-0.06013	-0.06776	0.05003	0.0747
15	0.23776	-0.00033	0.0484	-0.07294	0.01265	0.00374	0.00434	-0.01039	0.02032	0.00258	0.00532	0.04079	0.01191	0.01428	-0.02827	-0.05073	0.00502	-0.08887	-0.00382	0.00107	0.0387
16	0.22882	-0.00028	-0.01085	-0.07323	0.03068	0.06761	0.02522	0.08159	-0.0329	0.02678	0.05733	-0.01405	-0.01903	0.10909	-0.07324	0.009	0.0328	-0.03942	-0.02288	0.00678	0.0195
17	0.24532	-0.00032	0.03779	-0.10066	0.02308	0.00503	0.01185	-0.00405	-0.00578	0.04139	0.04801	-0.07687	-0.10439	0.04235	0.04604	-0.01704	0.00374	0.09564	-0.02179	-0.00783	-0.0845
18	0.3675	-0.0006	0.0693	-0.04809	0.0123	-0.00541	-0.00236	0.01242	-0.00137	-0.03942	-0.03299	-0.06553	0.00152	0.04986	0.02772	0.03447	0.06281	0.02764	0.02919	-0.06844	-0.0879
19	0.39867	-0.00053	-0.04805	-0.02749	-0.00031	0.06418	0.03137	0.07878	-0.01729	0.00091	-0.00254	0.01422	0.02369	-0.01154	-0.04717	0.10177	-0.03992	-0.04013	-0.0294	0.00348	0.0193
20	0.31	-0.00078	0.14974	0.27177	-0.00085	0.01649	-0.00962	0.08705	-0.01114	-0.19738	-0.12353	-0.24325	0.00948	0.1291	0.08843	0.04625	0.08136	-0.05725	0.08302	-0.12512	-0.0619
21	0.09242	-0.0001	-0.01297	-0.00012	-0.02291	-0.00652	-0.00382	0.00163	-0.00153	0.01208	0.03464	0.03966	-0.01021	0.02371	0.01762	-0.05274	-0.06263	0.04564	-0.00377	-0.015	0.0116
22	0.32283	-0.00058	-0.12345	0.19279	-0.18514	-0.05985	-0.05979	0.01345	0.00418	0.0119	-0.01595	-0.02723	-0.0231	-0.03202	0.02782	-0.02938	-0.01018	0.01964	-0.00539	0.06269	-0.1049
23	0.07784	-3.74E-05	-0.17476	0.09715	0.05202	0.31677	0.00562	0.36413	0.0831	-0.01083	-0.03526	-0.02148	-0.03067	-0.00436	-0.00372	0.02847	0.07318	-0.06053	0.0176	0.04643	-0.0242
24	0.02762	-0.0001	0.03398	0.01643	0.00177	0.00591	-0.00551	0.00998	-0.02291	-0.06904	0.00087	0.01547	0.02644	0.01589	-0.03076	-0.01826	-0.02873	-0.03282	-0.05305	0.00456	0.0408
25	0.2938	-0.00046	0.06009	-0.02457	-0.00121	0.01746	-0.02109	-0.01282	-0.00144	-0.04948	-0.04423	-0.06424	-0.01358	-0.03901	-0.04656	-0.05164	0.00663	-0.04046	-0.0139	-0.04866	0.0403
26	0.07144	0.00014	-0.02922	0.04737	-0.0055	0.56527	-0.50237	-0.40375	-0.00727	0.0255	-0.02781	-0.01608	0.00725	-0.00942	-0.01316	-0.00095	-0.00759	-0.01509	0.00966	-0.00062	0.0136
27	0.27846	-0.00076	0.40171	0.37611	0.19038	0.01919	0.01596	0.00773	-0.02676	0.2172	0.11789	0.09862	-0.07124	-0.14295	-0.0977	-0.02153	-0.01213	-0.04893	-0.04032	0.06868	0.0794
28	0.3138	-0.00058	-0.10549	0.11371	-0.17075	-0.0386	-0.03046	0.02374	-0.14784	0.01401	0.04917	0.03803	0.06437	-0.05872	-0.01264	-0.0493	-0.00814	0.1717	0.00321	-0.03202	0.0422
29	0.20506	-0.00042	0.02899	0.03254	-0.04465	-0.033	-0.02053	-0.0097	-0.04365	-0.04779	-0.03883	-0.10491	-0.02101	-0.05238	-0.03758	-0.05605	0.02122	-0.06425	-0.03287	0.02497	0.0507

svd_features

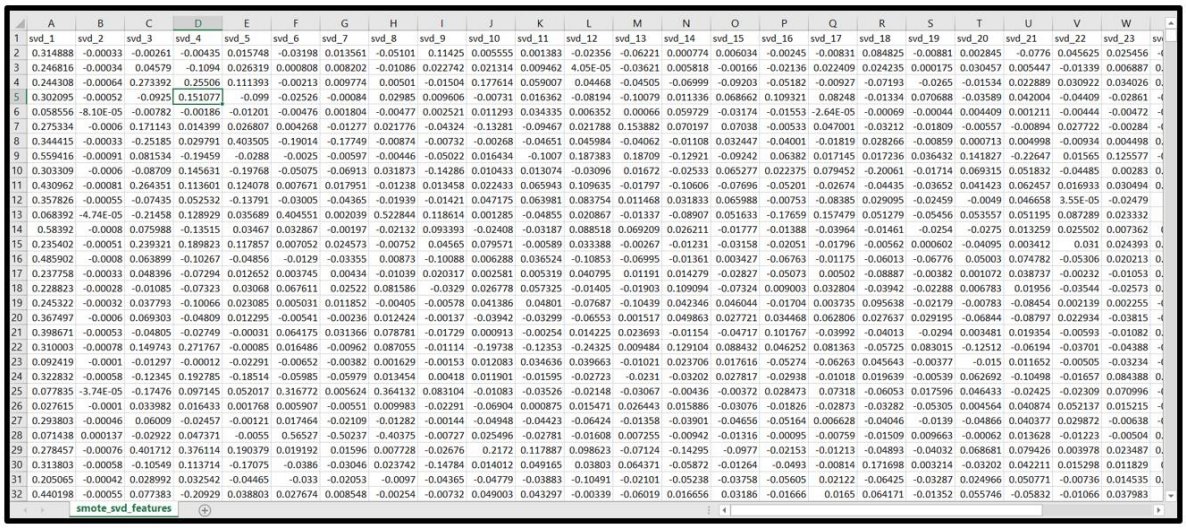
Although SVD addresses the issue of high dimensionality produced by TF-IDF, this dimensionality reduction can make interpretation difficult due to the loss of fine-grained information, such as the importance of specific individual words like "password" or "verify," which may carry critical meaning in identifying phishing emails but get merged into broader, less specific patterns. Thus, the SMOTE technique is utilised on the dataset to overcome this issue. SMOTE is a popular method used to address class imbalance in datasets, such as when phishing emails are significantly fewer in number than legitimate ones. Instead of simply duplicating existing minority class samples, SMOTE creates new synthetic examples to enrich the minority class. It works by selecting a sample from a minority class and finding its nearest neighbours in the feature space. Then, it generates new synthetic data points by interpolating between the original sample and its neighbours. This process creates more diverse and realistic minority class samples, which helps the ML model better learn the characteristics of phishing emails. By balancing the number of phishing and legitimate emails, SMOTE reduces bias toward the majority class, enhances the model's ability to detect phishing attempts, and often results in improved overall classification performance.

When applying SMOTE, the synthetic samples generated contain feature values based on interpolation between existing minority class examples. These values can be either positive or negative, depending on the nature of the original features and how they interact. Unlike SVD, where positive and negative values represent the strength or opposite of specific hidden patterns, SMOTE values simply represent

feature intensities in the synthetic data points. Positive values indicate a stronger presence of certain features, while negative values might arise due to the mathematical interpolation process, but do not carry the same interpretive meaning as in SVD. Instead, SMOTE's goal is to create new, realistic samples that help balance the dataset, improving the model's learning and detection ability. Figure 5 illustrates the output of columns generated after applying the SMOTE technique.

Figure 5

Output of Columns After Applying the Class Imbalance Handling Technique



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	svd_1	svd_2	svd_3	svd_4	svd_5	svd_6	svd_7	svd_8	svd_9	svd_10	svd_11	svd_12	svd_13	svd_14	svd_15	svd_16	svd_17	svd_18	svd_19	svd_20	svd_21	svd_22	svd_23
2	0.314888	-0.00033	-0.00261	-0.00435	0.015748	-0.03198	0.013561	-0.05101	0.11425	0.005555	0.001383	-0.02356	-0.06221	0.000774	0.006034	-0.00245	-0.00831	0.084825	-0.00881	0.002845	-0.0776	0.045625	0.025456
3	0.246816	-0.00034	0.04579	-0.1094	0.026319	0.000808	0.008202	-0.01086	0.022742	0.021314	0.009462	4.05E-05	-0.03621	0.005818	-0.00166	-0.02136	0.022409	0.024235	0.000175	0.030457	0.005447	-0.01339	0.006887
4	0.244308	-0.00064	0.273392	0.25506	0.111393	-0.00213	0.009774	0.00501	-0.01504	0.177614	0.059007	0.04468	-0.04505	-0.06999	-0.09203	-0.05182	-0.00927	-0.07193	-0.0265	-0.01534	0.022889	0.030922	0.034026
5	0.302095	-0.00052	-0.0925	0.151077	-0.099	-0.02526	-0.00084	0.02985	0.009606	-0.00731	0.016362	-0.08194	-0.10079	0.011336	0.068662	0.109321	0.08248	-0.01334	0.070688	-0.03589	0.042004	-0.04409	-0.02861
6	0.058556	-8.10E-05	-0.00782	-0.00186	-0.01201	-0.00476	0.001804	-0.00477	0.002521	0.011293	0.034335	0.006352	0.00066	0.059729	-0.03174	-0.01553	-2.64E-05	-0.00069	-0.00044	0.004409	0.001211	-0.00444	-0.00472
7	0.275334	-0.0006	0.171143	0.014399	0.026807	0.004268	-0.01277	0.021776	-0.04324	-0.13281	-0.09467	0.021788	0.153882	0.070197	0.07038	-0.00533	0.047001	-0.03212	-0.01809	-0.00557	-0.00894	0.027722	-0.00284
8	0.344415	-0.00033	-0.25185	0.029791	0.403505	-0.19014	-0.17749	-0.00874	-0.00732	-0.00268	-0.04651	0.045984	-0.04062	-0.01108	0.032447	-0.04001	-0.01819	0.028266	-0.00859	0.000713	0.004998	-0.00934	0.004498
9	0.559416	-0.00091	0.081534	-0.19459	-0.0288	-0.0025	-0.00597	-0.00446	-0.05022	0.016434	-0.1007	0.187383	0.18709	-0.12921	-0.09242	0.06382	0.017145	0.017236	0.036432	0.141827	-0.22647	0.01565	0.125577
10	0.303309	-0.0006	-0.08709	0.145631	-0.19768	-0.05075	-0.06913	0.031873	-0.14286	0.010433	0.013074	-0.03096	0.01672	-0.02533	0.065277	0.022375	0.079452	-0.20061	-0.01714	0.069315	0.051832	-0.04485	0.00283
11	0.430962	-0.00081	0.264351	0.113601	0.124078	0.007671	0.017951	-0.01238	0.013458	0.022433	0.065943	0.109635	-0.01797	-0.10606	-0.07696	-0.05201	-0.02674	-0.04435	-0.03652	0.041423	0.062457	0.016933	0.030494
12	0.357826	-0.00055	-0.07435	0.052532	-0.13791	-0.03005	-0.04365	-0.01939	-0.01421	0.047175	0.063981	0.083754	0.011468	0.031833	0.065988	-0.00753	-0.08385	0.029095	-0.02459	-0.0049	0.046658	3.55E-05	-0.02479
13	0.068392	-4.74E-05	-0.21458	0.128929	0.035689	0.404551	0.002039	0.522844	0.118614	0.001285	-0.04855	0.020867	-0.01337	-0.08907	0.051633	-0.17659	0.157479	0.051279	-0.05456	0.053557	0.051195	0.087289	0.023332
14	0.58392	-0.0008	0.075988	-0.13515	0.03467	0.032867	-0.00197	-0.02132	0.093393	-0.02408	-0.03187	0.088518	0.069209	0.026211	-0.01777	-0.01388	-0.03964	-0.01461	-0.0254	-0.0275	0.013259	0.025502	0.007362
15	0.235402	-0.00051	0.239321	0.189823	0.117857	0.007052	0.024573	-0.00752	0.04565	0.079571	-0.00589	0.033388	-0.00267	-0.01231	-0.03158	-0.02051	-0.01796	-0.00562	0.000602	-0.04095	0.003412	0.031	0.024393
16	0.485902	-0.0008	0.063899	-0.10267	-0.04856	-0.0129	-0.03355	0.00873	-0.10088	0.006288	0.036524	-0.10853	-0.06995	-0.01361	0.003427	-0.06763	-0.01175	-0.06013	-0.06776	0.05003	0.074782	-0.05306	0.020213
17	0.237758	-0.00033	0.048396	-0.07294	0.012652	0.003745	0.00434	-0.01039	0.020317	0.002581	0.005319	0.040795	0.01191	0.014279	-0.02827	-0.05073	0.00502	-0.08887	-0.00382	0.001072	0.038737	-0.00232	-0.01053
18	0.228823	-0.00028	-0.01085	-0.07323	0.03068	0.067611	0.02522	0.081586	-0.0329	0.026778	0.057325	-0.01405	-0.01903	0.109094	-0.07324	0.009003	0.032804	-0.03942	-0.02288	0.006783	0.01956	-0.03544	-0.02573
19	0.245322	-0.00032	0.037793	-0.10066	0.023085	0.005031	0.011852	-0.00405	-0.00578	0.041386	0.04801	-0.07687	-0.10439	0.042346	0.046044	-0.01704	0.003735	0.095638	-0.02179	-0.00783	-0.08454	0.002139	0.002255
20	0.367497	-0.0006	0.069303	-0.04809	0.012295	-0.00541	-0.00236	0.012424	-0.00137	-0.03942	-0.03299	-0.06553	0.001517	0.049863	0.027721	0.034468	0.062806	0.027637	0.029195	-0.06844	-0.08797	0.022934	-0.03815
21	0.398671	-0.00053	-0.04805	-0.02749	-0.00031	0.064175	0.031366	0.078781	-0.01729	0.000913	-0.00254	0.014225	0.023693	-0.01154	-0.04717	0.101767	-0.03992	-0.04013	-0.0294	0.003481	0.019354	-0.00593	-0.01082
22	0.310003	-0.00078	0.149743	0.271767	-0.00085	0.016486	-0.00962	0.087055	-0.01114	-0.19738	-0.12353	-0.24325	0.009484	0.129104	0.088432	0.046252	0.081363	-0.05725	0.083015	-0.12512	-0.06194	-0.03701	-0.04388
23	0.092419	-0.0001	-0.01297	-0.00012	-0.02291	-0.00652	-0.00382	0.001629	-0.00153	0.012083	0.034636	0.039663	-0.01021	0.023706	0.017616	-0.05274	-0.06263	0.045643	-0.00377	-0.015	0.011652	-0.00505	-0.03234
24	0.322832	-0.00058	-0.12345	0.192785	-0.18514	-0.05985	-0.05979	0.013454	0.00418	0.011901	-0.01595	-0.02723	-0.0231	-0.03202	0.027817	-0.02938	-0.01018	0.019639	-0.00539	0.062692	-0.10498	-0.01657	0.084388
25	0.077835	-3.74E-05	-0.17476	0.097145	0.052017	0.316772	0.005624	0.364132	0.083104	-0.01083	-0.03526	-0.02148	-0.03067	-0.00436	-0.00372	0.028473	0.07318	-0.06053	0.017596	0.046433	-0.02425	-0.02309	0.070996
26	0.027615	-0.0001	0.033982	0.016433	0.001768	0.005907	-0.00551	0.009983	-0.02291	-0.06904	0.000875	0.015471	0.026443	0.015886	-0.03076	-0.01826	-0.02873	-0.03282	-0.05305	0.004564	0.040874	0.052137	0.015215
27	0.293803	-0.00046	0.06009	-0.02457	-0.00121	0.017464	-0.02109	-0.01282	-0.00144	-0.04948	-0.04423	-0.06424	-0.01358	-0.03901	-0.04656	-0.05164	0.006628	-0.04046	-0.0139	-0.04866	0.040377	0.029872	-0.00638
28	0.071438	0.000137	-0.02922	0.047371	-0.0055	0.56527	-0.50237	-0.40375	-0.00727	0.025496	-0.02781	-0.01608	0.007255	-0.00942	-0.01316	-0.00095	-0.00759	-0.01509	0.009663	-0.00062	0.013628	-0.01223	-0.00504
29	0.278457	-0.00076	0.401712	0.376114	0.190379	0.019192	0.01596	0.007728	-0.02676	0.2172	0.117887	0.098623	-0.07124	-0.14295	-0.0977	-0.02153	-0.01213	-0.04893	-0.04032	0.068681	0.079426	0.003978	0.023487
30	0.313803	-0.00058	-0.10549	0.113714	-0.17075	-0.0386	-0.03046	0.023742	-0.14784	0.014012	0.049165	0.03803	0.064371	-0.05872	-0.01264	-0.0493	-0.00814	0.171698	0.003214	-0.03202	0.042211	0.015298	0.011829
31	0.205065	-0.00042	0.028992	0.032542	-0.04465	-0.033	-0.02053	-0.0097	-0.04365	-0.04779	-0.03883	-0.10491	-0.02101	-0.05238	-0.03758	-0.05605	0.02122	-0.06425	-0.03287	0.024966	0.050771	-0.00736	0.014535
32	0.440198	-0.00055	0.077383	-0.20929	0.038803	0.027674	0.008548	-0.00254	-0.00732	0.049003	0.043297	-0.00339	-0.06019	0.016656	0.03186	-0.01666	0.0165	0.064171	-0.01352	0.055746	-0.05832	-0.01066	0.037983

smote_svd_features

Phase 4: Dataset Splitting and Model Development

In this phase, the dataset is split into two categories using the train_test_split function from the sklearn module. Seventy per cent of the dataset is allocated as training data, while the remaining 30% is set aside as testing data. This step ensures that the models are trained using a sufficient amount of data to recognise patterns, while preserving the separate unseen part of the data to prevent biased performance evaluation during the model's assessment. Both iterations applied the same process, except for the second iteration, where the five-fold cross-validation was introduced in the training dataset to improve model generalisation by allowing it to maximise the use of available data.

Phase 5: Model Evaluation and Analysis

In this phase, the performance of each trained model —SVM, Random Forest, and Decision Tree — was evaluated for both iterations of the proposed pre-processing technique. The evaluation employed standard metrics, including accuracy, precision, recall, and F1-score, which together provide a comprehensive view of how well the models distinguish between legitimate and phishing emails. To facilitate better interpretation of the results, confusion matrices were also used to visualise the distribution of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These graphical representations helped identify common misclassification patterns and areas for improvement, ultimately enhancing the overall classification accuracy. By comparing the performance of these models across both iterations with the proposed techniques, the study aimed to determine the most effective approach for phishing email detection.

Measurement Parameters

Measurement parameters refer to the metrics or criteria used to evaluate the performance of a system, process or model, especially in ML and scientific research. In the context of evaluating the proposed technique for email phishing detection, the following measurement parameters were used: accuracy, precision, recall and F1-score. The list of equations presents the measure used for each evaluation in analysing the performance of the models. Equation 1 assesses the entire correctness of the model's predictions, whereas Equation 2 represents the proportion of accurately detected phishing emails among all phishing emails. Furthermore, Equation 3 measures the model's ability to detect all genuine phishing emails, thereby lowering the risk of missed attacks. Meanwhile, Equation 4, calculated as the harmonic mean of precision and recall, creates a balanced assessment by including both false positives and false negatives. In the equation below, True Positive (TP) indicates accurately identified phishing emails, True Negative (TN) represents correctly classified legitimate emails, False Positive (FP) indicates legitimate emails mistakenly labelled as phishing, and False Negative (FN) represents phishing emails misclassified as legitimate ones (Tharwat, 2020).

Accuracy

Calculate the correctly classified emails (both legitimate and phishing) over the total sum of emails in the dataset using the accuracy parameter as in Equation 1.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN}} \quad (1)$$

Where TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative

Precision

Equation 2 represents the precision, a ratio of true positives to the sum of true positives and false positives, which indicates the model's ability to identify phishing emails without misclassifying legitimate emails.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \quad (2)$$

Where TP = True Positive and FP = False Positive

Recall

Recall, as in Equation 3, is also known as sensitivity; it is the ratio of true positives to the sum of true positives and false negatives, which reflects the model's capacity to detect phishing emails while minimising missed detections successfully.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}} \quad (3)$$

Where TP = True Positive and FN = False Negative

F1-score

F1-score, as in Equation 4, is a harmonic mean of precision and recall, offering a balanced evaluation of a model's performance. It is particularly useful in scenarios with imbalanced class distributions, where achieving high precision or recall alone may not reflect true effectiveness.

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

RESULTS AND FINDINGS

The experiment evaluated three ML algorithms, which are SVM, Random Forest, and Decision Tree. The models were assessed using four key performance metrics: accuracy, precision, recall, and F1-score. For a comprehensive evaluation, the results are presented in multiple formats. Tables 2 and 3 display the performance metrics before and after applying improvement techniques in the pre-processing phase, respectively.

Table 2

Result After using Standard Pre-Processing Techniques

	SVM	Random Forest	Decision Tree
Accuracy	96%	96%	91%
Precision	93%	95%	88%
Recall	97%	95%	90%
F1-score	95%	95%	89%

Table 3

Result After Applying Proposed Techniques

	SVM	Random Forest	Decision Tree
Accuracy	94%	96%	93%
Precision	93%	95%	92%
Recall	96%	97%	95%
F1-score	94%	96%	93%

The results in Table 2 show the basic pre-processing result, while Table 3 indicates consistent performance improvements across all models after applying improvement techniques. Random Forest consistently outperformed the other models, achieving the highest scores across all evaluation metrics, making it the most reliable algorithm for detecting phishing emails. It maintained an accuracy of 96% and precision of 95%, while its recall improved from 95% to 97% and F1-score from 95% to 96%. The Decision Tree exhibited the most substantial improvement, with its accuracy increasing from 91% to 93%, precision from 88% to 92%, recall from 90% to 95%, and F1-score from 89% to 93%, indicating

a stronger generalisation capability after the improvement. SVM retained strong performance, experiencing a slight decrease in accuracy from 96% to 94% and F1-score from 95% to 94%, while its recall declined marginally from 97% to 96%.

However, its precision remained stable at 93%, suggesting consistent detection capability. These results demonstrate that the applied improvement techniques significantly enhance model robustness. A confusion matrix is regularly employed to offer additional details on the classification behaviour of the models. This method is a matrix of positive integer values described by Markoulidakis (2024) as the frequency of each possible pairing of predicted and actual class labels. It provides a detailed overview of the classification model's capability, highlighting its strengths and weaknesses in terms of both correct and incorrect predictions, as well as its ability to distinguish between different classes. In this experiment, the Random Forest model, which showed fewer false negatives and false positives, achieved a higher overall classification accuracy, outperforming other models.

However, SVM had a larger number of misclassifications after the improvement technique was applied, particularly both false negatives and false positives. It indicates that its detection accuracy decreased, despite maintaining a balanced classification performance. Meanwhile, the Decision Tree, which was previously prone to greater error rates, now shows significantly improved results, where the number of false positives and false negatives has decreased after applying the improved technique.

Figures 6 and 7 compare the confusion matrices of the SVM model before and after the application of improved techniques. Initially, the model recorded 3,182 true negatives, 169 false positives, 2,165 true positives, and 75 false negatives. Following the improvement, the true positives increased to 3,297, while the false negatives rose slightly to 154. Meanwhile, the number of true negatives decreased slightly to 3,090, while the number of false positives increased to 253.

Figure 6

Confusion Matrix using Standard Pre-Processing Techniques for SVM Model

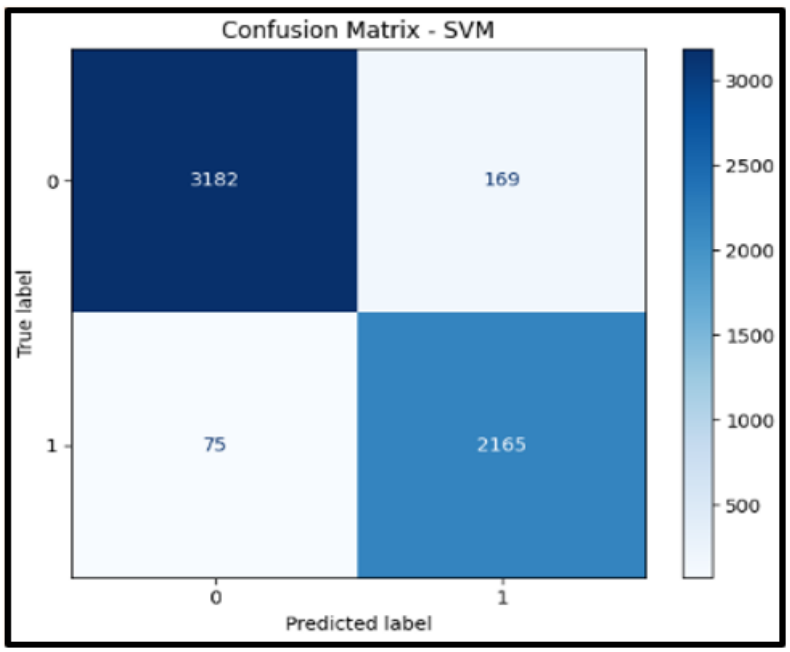
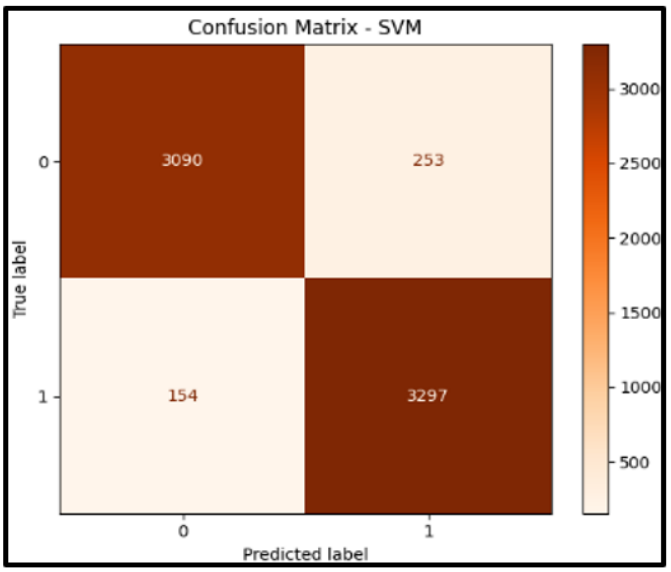


Figure 7

Confusion Matric using Standard Pre-Processing Techniques for SVM Model



Figures 8 and 9 present the confusion matrices of the Random Forest model before and after applying the improved techniques. Initially, the model accurately predicted 3,230 true negatives and 2,120 true positives, with 121 false positives and 120 false negatives. Following the improvement, the model recorded 3,163 true negatives and 3,351 true positives, while the number of false positives increased to 180 and the number of false negatives decreased to 100.

Figure 8

Confusion Matrix using Standard Pre-Processing Techniques for Random Forest Model

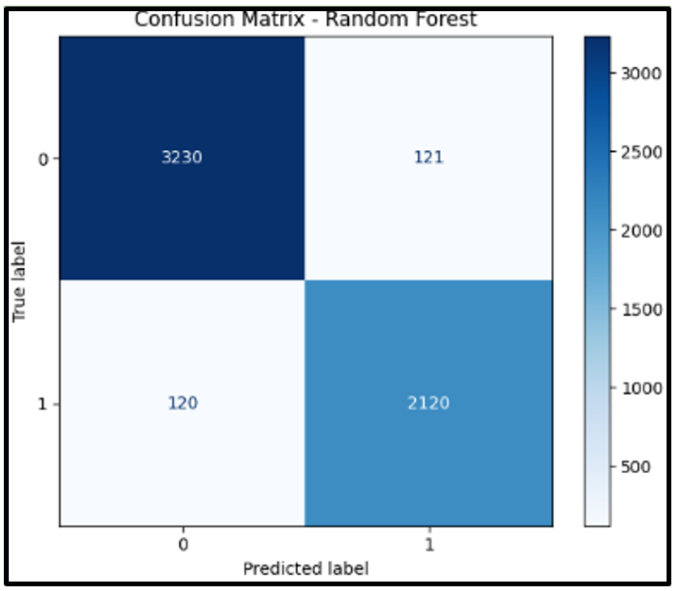
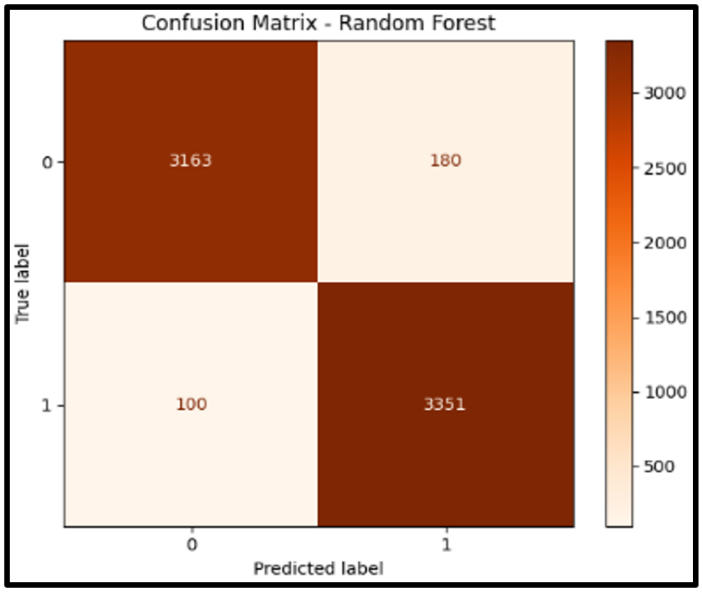


Figure 9

Confusion Matrix After Applying the Improved Technique for Random Forest Model



Figures 10 and 11 present the confusion matrices of the Decision Tree model before and after applying the improved techniques. Before improvement, the model correctly predicted 3,066 true negatives and 2,025 true positives, with 285 false positives and 215 false negatives. After the improvement, the model recorded 3,054 true negatives and 3,277 true positives, while the number of false positives slightly increased to 289, and the number of false negatives decreased to 174.

Figure 10

Confusion Matrix using Standard Pre-Processing Techniques for a Decision Tree Model

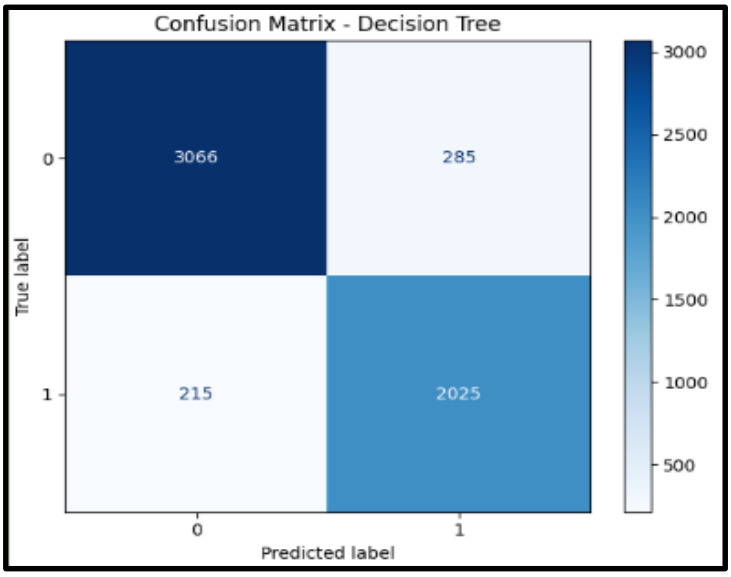
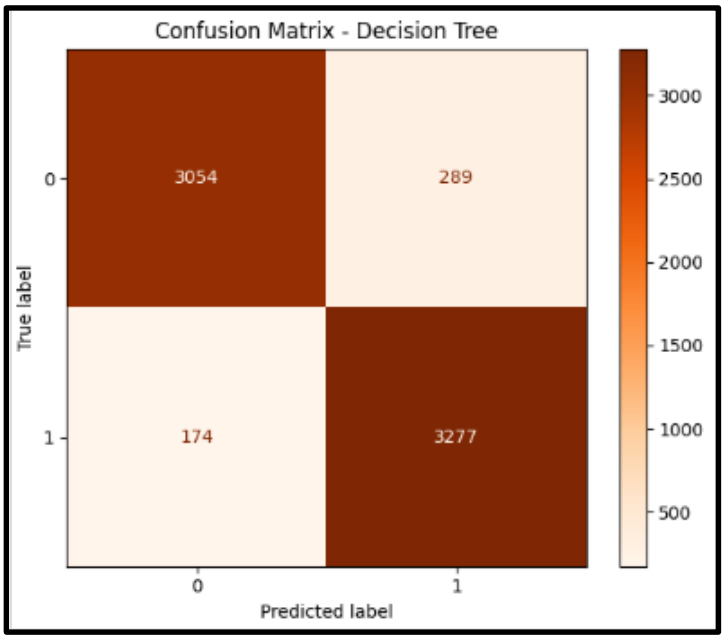


Figure 11

Confusion Matrix using Standard Pre-Processing Techniques for a Decision Tree Model



DISCUSSION

The application of improved pre-processing techniques played a crucial role in enhancing the performance of ML-based models for email phishing detection in this study. When these pre-processing techniques were implemented, they assisted the models in managing the imbalanced datasets, minimising noise, and highlighting significant characteristics, resulting in a more robust learning process. As a result, the algorithms become more capable of detecting and classifying phishing emails, even in complex or incomplete data contexts.

The SVM model's performance demonstrated particularly interesting behaviour once these improvements were applied. The results show that the number of true positives increased, which indicates a better ability to detect phishing emails. In contrast, the decline in true negatives reveals that the model became more cautious, potentially over-flagging legitimate emails as threats. Meanwhile, the number of both false positives and false negatives increased, suggesting that even though the model became more active in flagging phishing attempts, it also made mistakes, misclassifying some legitimate emails and failing to catch others that were actually phishing. Therefore, the model's capability to detect phishing from legitimate emails is slightly lower compared to earlier, as its overall accuracy performance has decreased from 96% to 94%.

Despite the declines in accuracy, SVM maintains a score of 93% for precision and only slightly drops for recall, from 97% to 96%. These two results yield an F1-score of 94%, which implies that the model remains reliable for phishing detection tasks. Even so, the overall outcome suggests that SVM may be less responsive to enhancement approaches due to its fundamental properties, which perform well with sparse and high-dimensional data. Moreover, techniques such as SMOTE, which introduce synthetic samples, and SVD, which reduce dimensionality, could conflict with the margin-based classification strategy of SVM.

Meanwhile, the result of the Random Forest model showed continuous and significant increases. The increment in true positives indicates that the model accurately detects phishing emails as phishing. Next, the drop in the number of false negatives shows that the model grew more successful at reducing undetected risks. The reduction in true negatives was small and had no major impact on the model's performance. Lastly, the number of false positives increased moderately, resulting in false alarms where legitimate emails were incorrectly identified as phishing. Even so, the overall detection ability remained strong, as the accuracy of the model remained consistent at 96%, precision at 95%, and recall increased from 95% to 97%, which contributed to a higher F1-score of 96%. These findings show that the model has an excellent ability to distinguish between phishing and legitimate emails. These results show that Random Forest responds well to the applied improved technique, as the structure of the Random Forest algorithm, which combines the predictions of many decision trees, enables the model to efficiently absorb and benefit from a richer and more balanced dataset.

Aside from Random Forest, the Decision Tree model also benefits greatly from the improved technique due to its adaptable and interpretable structure. The model's ability to detect phishing emails improves as the number of false negatives decreases, resulting in fewer phishing emails being missed. The increase in true positives suggests that most phishing emails are classified accurately. Although a few legitimate emails were misclassified, as indicated by the increase in false positives and the slight decrease in true negatives, the overall performance mitigates these minor disadvantages. The rise in performance across all assessment criteria proves the model's ability to differentiate between phishing and legitimate emails.

In summary, the findings strongly support the study's goal of enhancing data security mechanisms by investigating the role of additional or enhanced pre-processing techniques in improving phishing email detection. The improved scores for recall and F1-score across all three models showcase a greater ability of each model to correctly detect phishing emails while reducing the risk of threats being undetected. Furthermore, a reduction in the number of false negatives significantly contributes to achieving the goal, as undetected phishing emails that bypass the system frequently serve as entry points for unauthorised access, credential theft, and broader data breaches. Moreover, this enhanced pre-processing for phishing email detection contributes to a proactive cybersecurity barrier, preventing attackers from using email systems as a gateway to access critical information. The fewer phishing emails that reach users, the higher the cybersecurity resilience for that system. This builds trust in digital communication systems since it protects both personal and organisational assets. Taken together, the applied improved technique considerably improved the models' detection capabilities and advanced the overall goal of proactive, reliable, and scalable data security.

CONCLUSION

As a strategic plan for a data protection mechanism, this study aims to evaluate the impact of an enhanced pre-processing technique by comparing the performance (before and after using the improved technique) of three ML models, which are SVM, Random Forest, and Decision Tree. The improved techniques, including SMOTE, TF-IDF, and SVD, were utilised to address constraints and enhance each model's ability to detect phishing emails more accurately and consistently. The results revealed that two out of three models benefited greatly from the proposed technique, with Random Forest outperforming the others across all evaluation metrics. Both Random Forest and Decision Tree showed notable improvements in recall and F1-score, as proven by the reduction in false negatives. Although SVM maintained high precision and recall, it was less responsive to improved techniques since SVM is more dependent on data features and pre-processing techniques.

The findings demonstrate the impact of well-chosen pre-processing procedures in improving performance outcomes. These enhanced models can help enterprises protect sensitive information, meet legal requirements, and maintain user confidence. Additionally, the low number of false negatives can reduce undetected threats, and a consistent number of false positives guarantees that sensitive information is better protected against unauthorised access. Importantly, this study reveals that traditional ML techniques are still highly relevant and effective in detecting phishing emails, depending on the quality of pre-processing. The scalability and interpretability they have demonstrated make them ideal for real-world applications where computational resources and transparency are crucial.

Future studies can expand on this result by combining larger, more dynamic, and real-time datasets to better reflect the changing patterns of phishing methods. Larger datasets generally increase model generalizability and robustness by exposing algorithms to a broader range of phishing strategies and writing styles (Divakaran & Oest, 2022). However, this study used a small number of labelled emails to ensure experimental feasibility, maintain computing efficiency, and simplify training and evaluation cycles. Next, new datasets should also be introduced to further evaluate whether the enhanced pre-processing technique allows the models to generalise reliably across different data distributions.

ACKNOWLEDGMENT

This research was supported by the Ministry of Higher Education (MoHE) through the Fundamental Research Grant Scheme (Ref: FRGS/1/2020/ICT03/UUM/02/1). The content of this article is solely the responsibility of the authors and does not necessarily represent the official views of MoHE, Malaysia. The authors would also like to extend their gratitude to the Data Management &Software Solution Research Lab, School of Computing, Universiti Utara Malaysia, for their generous support in sponsoring the publication of this article (S/OCode:14839).

REFERENCES

Ahmed, M. R., Islam, M. M., & Layek, M. A. (2024). *Phishing URL detection using comprehensive feature extraction and machine learning techniques.* IEEE CS BDC Symposium 2024. https://s24.ieeecsbdc.org/papers/156

Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science, 3,* 1-23. https://doi.org/10.3389/fcomp.2021.563060

Al-Yozbaky, R. S., & Alanezi, M. (2023). Phishing emails detection models: A comparative study. *Journal of Modern Computing and Engineering Research, 2023,* 58–67. https://jmcer.org/research/phishing-emails-detection-models-a-comparative-study/

Anti-Phishing Working Group (APWG). (2024). Phishing activity trends report (4th quarter 2024). https://docs.apwg.org/reports/apwg_trends_report_q4_2024.pdf

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

Divakaran, D. M., & Oest, A. (2022). Phishing detection leveraging machine learning and deep learning: A review. *arXiv*. https://doi.org/10.48550/arXiv.2205.07411

Felix, E. A. & Lee, S. P. (2019). Systematic literature review of pre-processing techniques for imbalance data. *IET Software, 13*(6), 479-496. https://doi.10.1049/iet-sen 2018.5193

Gualberto, E. S., De Sousa, R. T., Vieira, T. P. D. B., Da Costa, J. P. C. L., & Duque, C. G. (2020). From feature engineering and topics models to enhance prediction rates in phishing detection. *IEEE Access, 8,* 76368-76385. https://doi.org/10.1109/ACCESS.2020.2989126

Habib, P., Sharma, U., & Sethi, K. S. (2022). Phishing detection with machine learning. *International Journal for Study in Applied Science & Engineering Technology (IJRASET), 10*(12), 1609-1615. https://doi.org/10.22214/ijraset.2022.48276

Hamadouche, S., Boudraa, O., & Gasmi, M. (2024). Combining lexical, host, and content-based features for phishing websites detection using machine learning models. *EAI Endorsed Transactions on Scalable Information Systems, 11*(6), 1-15. https://doi.org/10.4108/eetsis.4421

Harikrishnan, R., Soman, K. P., & Jayaraj, V. (2018). *A machine learning approach towards phishing email detection CEN-Security@IWSPA 2018.* Proceedings of the 1st Anti-Phishing Shared Task Pilot at 4th ACM IWSPA co-located with 8th ACM Confer-ence on Data and Application Security and Privacy (CODASPY 2018) (pp. 21-28). Tempe, Arizona, USA: CEUR-WS.org. https://ceur-ws.org/Vol-2124/paper_7.pdf

Hina, M., Ali, M., Javed, A. R., Srivastava, G., Gadekallu, T. R., & Jalil, Z. (2021). *Email classification and forensics analysis using machine learning.* 2021 IEEE Smart-World, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/ UIC/ATC/IOP/SCI) (pp. 630–635). IEEE. https://doi.org/10.1109/SWC50871.2021.00093

Hussin, I. H., Nazarudin, L. S. I., & Nor Azman, N. A. (2024). Comparative analysis of machine learning and deep learning for email spam detection. *TechRxiv*. https://doi.org/10.36227/ techrxiv.172115119.92836191/v1

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 1-54. https://doi.org/10.1186/s40537-019-0192-5

Konda, B. (2022). The impact of data pre-processing on data mining outcomes. *World Journal of Advanced Research and Reviews, 15*(3), 540–544. https://doi.org/10.30574/wjarr.2022.15.3. 0931

Kovacs, E. (2024). Discount retail giant pepco loses €15 million to cybercriminals. *SecurityWeek.* https://www.securityweek.com/discount-retail-giant-pepco-loses-e15-million-to-cybercrimina ls/

Kumari, A., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2023). Impact of the composition of feature extraction and class sampling in Medicare fraud detection. In M. Tanveer, S. Agarwal, S. Ozawa, A. Ekbal, & A. Jatowt (Eds.), *Neural information processing. ICONIP 2022* (Lecture Notes in Computer Science, Vol. 13625, pp. 708–720). Springer. https://doi.org/10.1007/978-3-031-30111-7_54

Malik, H. K., Al-Anber, N. J., & Al-Mekhlafi, F. A. E. (2023). Comparison of feature selection and feature extraction role in dimensionality reduction of big data. *Journal of Techniques, 5(1), 184–192*. https://doi.org/10.51173/jt.v5i1.1027

Markoulidakis, I. & Markoulidakis, G. (2024). Probabilistic confusion matrix: A novel method for machine learning algorithm generalised performance analysis. *Technologies, 12(7), 113.* https://doi.org/10.3390/technologies12070113

Mohey, H. & Mohsen, S. (2021). Using machine learning techniques for predicting email spam. Special Issue The Fifth Scientific Conference of Obour Institutes "Engineering and Informatics for Sustainable Smart Organizations (pp. 19-23). *Obour Institutes, Egypt: International Journal of Instructional Technology and Educational Studies (IJITES).*

Reddy, G. D., Sreelata, S., & Shreya, D. (Year). Efficient email phishing detection using machine learning. *International Journal of Scientific Study in Computer Science, Engineering and Information Technology, 9(3), 356-360.* https://doi.org/10.32628/IJSRCSEIT

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Shejole, S., Salam, T. A., Gupta, M. K., Sharma, K., & Attar, S. (2023). Email spam detection using machine learning. *International Study Journal of Engineering and Technology (IRJET), 10(5).* https://www.irjet.net/archives/V10/i5/IRJET-V10I5226.pdf

Tanti, R. (2024). Phishing attack: A case study and their prevention techniques. *International Journal for Multidisciplinary Research (IJFMR), 6(5), 1–8.* https://doi.org/10.55041/IJSREM38042

Tawil, A. A., Almazaydeh, L., Qawasmeh, D., Qawasmeh, B., Alshinwan, M. & Elleithy, K. (2024). Comparative analysis of machine learning algorithms for email phishing detection using TF-IDF, Word2Vec, and BERT. *Computers, Materials & Continua, 81(2), 3395–3412.* https://doi.org/10.32604/cmc.2024.057279

The Radicati Group, Inc. (2023). *Email statistics report, 2023-2027*. The Radicati Group, Inc. https://www.radicati.com/wp/wp-content/uploads/2023/04/Email-Statistics-Report-2023-2027-Executive-Summary.pdf