Computers & Security 133 (2023) 103378

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A comprehensive dual-layer architecture for phishing and spam email detection

Jay Doshi*, Kunal Parmar, Raj Sanghavi, Narendra Shekokar

Department of Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India

ARTICLE INFO

Article history:
Received 18 April 2023
Revised 20 June 2023
Accepted 30 June 2023
Available online 5 July 2023

Keywords:
Email phishing
Email spamming
Machine learning
Dual-layer architecture
Deep learning

ABSTRACT

The widespread use of email communication for sharing personal, professional, and financial data has rendered it vulnerable to cyber-attacks. Detecting untrustworthy emails with minimal errors is crucial to counter these threats. This research paper focuses on classifying spam and phishing emails, commonly employed to steal confidential information by impersonating legitimate sources. The scale of such attacks is alarmingly high, causing substantial financial losses across sectors like banking, healthcare, technology, and other businesses. This research aims to classify both spam and phishing emails, addressing the limitations of existing studies that only focus on one type and consider either the email body or content for feature selection. This research incorporates features from both the email body and content during model training. The authors propose a novel approach that ensures highly accurate classification while effectively handling the common issue of data imbalance in email phishing and spam classification. The approach utilizes a dual-layer architecture, with each layer containing a trained or pre-trained model that classifies data instances into their respective classes. Layer 1 classifies the phishing class, while Layer 2 classifies the spam class. Building upon the proven effectiveness of deep learning techniques for text classification and analysis, the proposed architecture employs models like Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN). Experimental evaluation demonstrates the approach's remarkable accuracy, recall, precision, and F1-score, achieving 99.51%, 99.68%, 99.5%, and 99.52%, respectively. This signifies its high efficacy in detecting and classifying malicious emails with minimal errors, thus holding great promise in enhancing system security against cyber-attacks in email communication.

1. Introduction

In recent years, the advancement of technology and widespread access to the Internet have greatly transformed global connectivity and communication. Among the various modes of communication, electronic emails have emerged as a ubiquitous and extensively utilized means, playing a vital role in both personal and professional interactions. With their increasing significance, emails have found applications in diverse industries such as finance, banking, and marketing. However, this growing reliance on email communication has also led to an escalation in cybersecurity threats, particularly in the form of email-based exploitation (Deshpande et al., 2023). Cybercriminals exploit email channels to launch attacks that can cause significant harm to individuals and organizations. In fact, it has been estimated that up to 90% of cyber-attacks originate from emails (cyberattacks, 2020).

Despite efforts to enhance email security, vulnerabilities persist. Attackers employ a range of tactics, including social engineering, email account hacking, and the creation of fraudulent emails, to exploit organizations and compromise their systems. Of these tactics, social engineering strategies are particularly deceptive, as they aim to deceive employees, gain unauthorized access, leak sensitive information, distribute malware, and disrupt critical functions. Addressing these escalating email-based threats and strengthening cybersecurity measures has become imperative. Malicious individuals frequently exploit vulnerabilities in email networks, primarily utilizing spam and phishing emails as their preferred tools of attack.

Spam emails, which are unsolicited and unwanted messages, continue to pose a significant challenge in email communication. They are sent in bulk to numerous recipients and are primarily focused on commercial promotion, advertising products, services, or websites. However, spam emails can also contain scams, chain let-

* Corresponding author.
E-mail addresses: jayutrdoshi14@gmail.com (J. Doshi), parmarkunal003@gmail.com (K. Parmar), rajsanghavi9@gmail.com (R. Sanghavi), narendra.shekokar@djsce.ac.in (N. Shekokar).

https://doi.org/10.1016/j.cose.2023.103378
0167-4048/

ters, and malicious attachments, amplifying the potential risks. The magnitude of the problem becomes evident when considering the statistics listed by Cveticanin (2023). As of 2023, the average daily volume of legitimate email messages is a staggering 347.3 billion, while approximately 56.5% of all emails in 2022 were classified as spam, resulting in a daily volume of around 122.33 billion worldwide. This inundation of unsolicited content comes at a substantial cost, with businesses incurring an annual expense of about $20.5 billion due to email spam. Additionally, phishing scams targeted an alarming 85% of organizations in 2021. Classifying and identifying spam emails are crucial steps in combating these threats and safeguarding email communication.

Phishing, on the other hand, refers to the act of attempting to obtain sensitive information from individuals by disguising them as trustworthy sources. Phishing emails are deceptive messages that aim to deceive recipients into divulging sensitive data or engaging in actions that compromise their security. These emails often mimic legitimate communications from trusted entities such as banks, social media platforms, or online services. The prevalence of phishing attacks is evident from the statistics. According to James (2022), approximately 36% of all data breaches in 2022 were attributed to phishing attacks, resulting in an average cost of $4.91 million per breach. Shockingly, malicious emails account for around 1.2% of all emails sent, equivalent to a staggering daily volume of 3.4 billion phishing emails. Moreover, a concerning 88% of organizations have encountered spear phishing, an advanced and highly targeted form of phishing. Based on the findings by Cerruto et al. (2022), the availability of social network data presents a potential opportunity for phishers to exploit individuals by creating and sending targeted phishing emails. Phishers can create personalized phishing messages by mining publicly available social media data for names, interests, relationships, and locations. They use this amount of information to build confidence and increase the likelihood of their schemes. Phishers can also use social media to make their emails seem more trustworthy.

These risks associated with cybercrimes extend beyond individual users and can have severe repercussions for public and private organizations alike. In our interconnected world, safeguarding data, privacy, and digital infrastructure has become an indispensable necessity. Thus, it is essential to prioritize and enhance cybersecurity measures to protect individuals, organizations, and the overall integrity of our digital ecosystem. Hence, in this paper, the authors focused on identifying spam and phishing emails to mitigate the potential risks associated with cyber-attacks originating from these kinds of malicious emails.

Previous studies in the field of Machine Learning (ML) and Deep Learning (DL) have predominantly focused on either the contents or the body text of emails to extract relevant features. These features typically include attributes such as headers, URLs, syntax, and attachments for the email's structure, or they involve Natural Language Processing and text analysis techniques applied to the email's body text. However, to the best of our knowledge, there is a lack of significant and recent works that combine both the contents and body text of emails to provide a comprehensive representation that can be effectively processed by ML algorithms. Furthermore, there is a dearth of research specifically targeting the simultaneous identification of spam and phishing emails. It is worth noting that data imbalance is a common challenge in detecting phishing emails and spam emails, requiring appropriate handling.

In this research paper, the authors address these gaps by considering both the email's contents and body text as factors to train ML and DL algorithms. Our study focuses on simultaneously identifying spam and phishing emails. To achieve this, a dual-layer architecture pipeline is proposed that can effectively classify phishing and spam emails, aiming for the minimum possible error rate. Additionally, the issue of data imbalance is tackled by ensuring the efficient classification of minority classes. To enhance the performance of the classifiers, various feature engineering techniques have been explored, including feature extraction and feature selection, to obtain an optimal set of features. The accuracy of the classifier is evaluated using metrics such as accuracy, precision, and recall, and compared against state-of-the-art models.

The contributions of this research study can be summarized as follows:

- The research paper presents a novel approach that utilizes a dual-layer architecture for classifying emails into spam and phishing categories, achieving high-performance metrics.
- The proposed dual-layer architecture solves the prevalent problem of data imbalance in email classification, resulting in high recall, precision, and f1-score for minority classes.
- The research demonstrates that considering features from both the email content and email body during model training resulted in high-performance metric values.
- The proposed pipeline in this research enables the simultaneous and efficient classification of both phishing and spam emails.

The paper is divided into 8 sections. Section 2 provides the literature review of the previous works. This builds the foundation for the proposed methodology and the limitations found in the existing works. Section 3 discusses the preliminaries performed on the data set gathered including data pre-processing and feature engineering encompassing feature extraction and feature selection. Section 4 provides an overview of the algorithms implemented for training of classification model. Section 5 describes the design of the proposed architecture of the model. Section 6 states the different variations of the TF_IDF vectorizer implemented and how the proposed architecture has been implemented. Section 7 presents a thorough analysis of the performance of the proposed model in comparison to existing models. This section includes a comparison between the performance of various algorithms based on different evaluation parameters including measures such as accuracy, precision, recall, and F1-score. It also includes a detailed comparison of the proposed architecture with existing research works, highlighting the key differences and advantages of the proposed model. Finally, Section 8 concludes the research and describes the scope of the research, and Section 9 discusses the future scopes of the research work.

2. Related works

Numerous research work has been conducted to explore how to successfully detect and categorize emails. An attempt to detect phishing emails has been made in Abdulraheem et al. (2022) using Principle Component Analysis (PCA) for the selection of attributes, Ranker Search as the search method along with three machine learning algorithms namely Multilayer Perceptron, Decision Tree (J48, C4.5), and Logistic Model Tree. The highest precision observed is 96.9245% by using the Logistic Model Tree algorithm. However, with the increasing dimensionality of data, the computational complexities and time required for the implementation of the Logistic Model Tree increase. Our proposed solution achieves the highest precision of 99.5%.

Deep Learning techniques have been used along with Machine Learning algorithms in Bagui et al. (2019) to achieve the highest accuracy of 98.89% by implementing the Word Embedding technique for detecting phishing emails. Though it has brought improvements in the overall accuracy of the model, this technique is unable to process unknown words or words which are never encountered by the model. This leads to an inaccurate prediction of a new email as every email is different from one another. To eliminate challenges faced during the attribute selection process used in

traditional machine learning algorithms, deep neural networks are used in Ravi et al. (2018) for detecting phishing emails. But with the increasing dimensionality of data, more computing resources are required for training and implementing the model.

A hybrid bagged approach using Naive Bayes Multinomial classifier along with J48 Decision Tree algorithm to achieve the highest precision of 92.78% in Nayak et al. (2021). The model is used only for the detection of spam emails. However, our proposed solution can be used for the detection of both phishing as well as spam emails.

Machine Learning and Deep Learning techniques have been applied in the research study (Bansal and Sidhu, 2021) for detecting spam emails. The precision obtained after applying different algorithms in this hybrid approach is in the range of 88% to 95%. Bidirectional Long Short Term Memory with Convolutional Neural Network has been used in the work proposed in Rahman and Ullah (2020) on two different datasets. The precision obtained varied in both datasets from 86% to 98%.

NLP, or Natural Language Processing, is a powerful technique for analyzing texts. Based on the content of the text, it assigns a set of pre-defined tags or categories and may be used for topic recognition, sentiment analysis, text categorization, etc. The authors of Egozi and Verma (2018) have used a similar NLP-based approach to successfully detect over 95% of ham emails and 80% of phishing emails.

The authors of Kulkarni et al. (2020) implemented the Relief Feature Selection technique for classifying spam emails using header-based features of the email. It resulted in a varying accuracy of 91.06% to 94.78%. A back-propagation neural network (BPNN) has been implemented in the research work done by Yahya et al. (2016) which provided the best accuracy of 99.13% among various other approaches like Naive Bayes, SVM, and Linear Discriminant techniques for classifying phishing emails based on the attributes of the email body.

Attributes from both header and body of the emails are extracted for classifying emails in Ruan and Tan (2009). The results obtained explain the significance of including the attributes from both header and the body of the email in contrast to including the features of either of them while training the model. Based on this observation, in this research work, features from the header and body of the email are extracted for training the classification model.

A new algorithm G-SFO (Grey-Sail Fish Optimization) has been proposed in Samarthrao and Rohokale (2022) to maximize the accuracy and precision of the classification results. It is inspired by Gray Wolf Optimization (GWO) and Sail Fish Optimization (SFO) algorithms. The G-SFO algorithm has been applied to four different datasets and the maximum precision obtained is 98.86%.

An attempt has been made in Gangavarapu et al. (2020b) to detect spam and phishing emails. After feature extraction and feature selection using six methods, an optimal feature subspace has been obtained. Eight machine learning algorithms are then applied and evaluated on the basis of different evaluation metrics of which the Random Forest classifier outperformed with an overall 98.4% accuracy on the ham and spam dataset and 99.4% on the ham and phishing dataset.

A sample handling method has been proposed in Li et al. (2022). LSTM (Long Short-Term Memory Network) neural network model is then used for classifying message body. To avoid overfitting of sample data, Dropout, and Regularization techniques have been used. The optimizer used for adjusting the learning rate is Adam. The proposed approach produced better results as compared to traditional machine learning algorithms with a maximum achieved precision of 0.96.

As discussed by the authors in Dada et al. (2019) and Gangavarapu et al. (2020a), major email service providers such as Google and Yahoo extensively employ email filtering systems to classify emails and combat spam. These systems utilize machine learning algorithms, including Logistic Regression (LR) and Neural Networks (NN), to categorize emails based on various features extracted from different email components such as the body, headers, structure, attachments, and URLs. Notably, Google focuses on IP addresses for feature extraction, while Yahoo prioritizes domains within URLs. These service providers continuously improve their filtering systems by incorporating user feedback. Moreover, Google utilizes Optical Character Recognition (OCR) to analyze attachments. The present research builds upon previous studies and recognizes the effectiveness of Deep Learning Algorithms based on Neural Networks for email classification. This study inspired the researchers to extract features from multiple components of emails, including the body, headers, URLs, and attachments, as these factors significantly contribute to an efficient classification system.

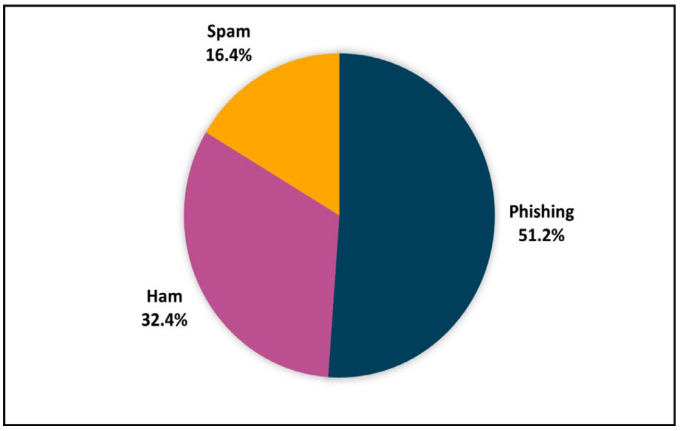

Fig. 1. Dataset distribution.

The ISIC 2019 challenge training dataset is used by the authors of Dedhia et al. (2022) to propose a pipeline to categorize skin lesions into eight categories. The authors' proposed pipeline is leveraged to address the issue of data imbalance in the image dataset. The findings of this research study formed the basis for the development of a dual-layer approach to text classification proposed in this research work.

By analyzing different research studies as illustrated in Table 1, it is observed that Deep Learning techniques are better suited for email classification and text analysis. Thus, a dual-layer architecture using Deep learning techniques has been proposed to classify the type of emails.

3. Preparation of data

3.1. Dataset

In this study, a dataset of real-world phishing emails from Nazario's publicly accessible phishing corpus (Nazario, 2006) and spam and ham emails from the spam assassin project (Foundation, 2006) has been collected. A total of 8218 emails are obtained in raw format.

As shown in Fig. 1, it is evident that the dataset is imbalanced. The Phish class, with 4204 (51.2%) training instances, constitutes the majority of the dataset, while the Spam class, with 1350 (16.4%) instances, and the Ham class, with 2664 (32.4%) instances, belong to the minority class as they constitute relatively small sections of the dataset in comparison to the Phish class.

Table 1
Description of different methods used.

Methods used	Paper referred	Description	Strength	Weakness	Evaluation parameters
Traditional Machine Learning Algorithms	Abdulraheem et al. (2022), Nayak et al. (2021), Bansal and Sidhu (2021)	Different machine learning algorithms like naive Bayes, Random Forest, Decision Tree, Gradient Boosting, Logistic Regression, etc have been applied to the email dataset to classify the type of emails.	Machine learning algorithms are simple to implement and usually do not have a high requirement of computational resources.	The performance of these algorithms is lower than that of Deep Learning techniques and also, it can result in biased outcomes depending upon the values of the sample training dataset.	Highest obtained precision: 96.92%
Deep Learning	Ravi et al. (2018), Bansal and Sidhu (2021), Rahman and Ullah (2020), Li et al. (2022)	Deep Learning techniques like ANN, CNN, RNN, etc are able to understand the complex natures of the problems.	The performance of Deep learning techniques is usually better than traditional machine learning algorithms. It identifies the nature of patterns easily. It is able to handle imbalanced data.	Deep Learning techniques require high computational resources for training.	Highest obtained precision: 98%
Feature Selection	Gangavarapu et al. (2020b)	Feature selection is used to find the best set of features in order to reduce the dimensionality of the sample data.	It helps in narrowing the features to only the most relevant ones. This reduces the time required for producing the outcome.		
There is a high overfitting risk if there is insufficient data. If the dataset is skewed, then majority classes are preferred as compared to minority classes.	Highest obtained accuracy on the ham and spam dataset: 98.4% and Highest obtained accuracy on the ham and phishing dataset: 99.4%				
Word Embedding - Natural Language Processing (NLP)	Bagui et al. (2019), Egozi and Verma (2018)	Word Embedding is a numeric vector input that represents a word in a lower-dimensional space.	It increases the speed of training the model and also stores an approximation of the meaning.	It can exhaustively take up a lot of memory space.	Highest obtained accuracy: 98.89%

3.2. Dataset preprocessing

Fig. 2 illustrates the comprehensive preprocessing techniques employed on the dataset comprising of Ham, Spam, and Phishing emails. Ham and spam emails are obtained as literal strings i.e. raw emails, thus processing is required to carry out computations for email classification. In contrast, phishing emails are received in mbox (mailbox) format (mbox, 2022). Each mbox file contains a significant number of emails that must be processed individually. These mbox files are parsed by the Python Mailbox Module (Foundation, 2022) to retrieve each individual email. Since each retrieved email will be in literal string format (i.e., raw emails), the processing is required before email classification calculations can be carried out. The Mail Parser Module (Mantuano, 2022) is used to process these raw emails in order to retrieve essential fields such as the header, subject, body, and attachment metadata. For each email, on the retrieved subject and body various text transformation operations are performed. Among the various operations is the transformation of all texts to lowercase to preserve consistency throughout the processing. The following process includes sentence and word tokenization to obtain vocabulary. This is the vocabulary that will be considered throughout training. The generated vocabulary is further processed to remove noisy data such as URLs, punctuation, numerals, stopwords, and blacklisted words. These blacklist terms are derived from data observation and analysis, resulting in a list of words that includes font types, HTML tags, domain extensions, and hexadecimal values like 0x0, 0x1, and so on. Furthermore, using the NLTK module (Steven Bird and Loper, 2009) the processed vocabulary is lemmatized. Lemmatization is one of the normalization techniques adopted for text processing. Lemmatization is the process of reducing words to their root form or dictionary form. Lemmatization connects synonymous terms. Finally, Duplicates are discarded from each email category.

3.3. Feature extraction

After dataset preprocessing, various attributes of the emails including the email addresses of the sender and recipients, the subject, header, plain text, HTML text, anchor tags, and attachments are used for feature extraction. The process of feature extraction involves identifying and selecting the most important and relevant features from the dataset that can be used for training the model and making predictions (Kumar Birthriya and Jain, 2022).

It is a crucial step in the overall process of classification, as it helps to improve the accuracy and performance of the model by reducing the dimensionality of the data and eliminating irrelevant or redundant features.

3.3.1. Domain knowledge

The URL present in anchor tags of the email body contains legitimate website links if the email is a genuine one as explained in Abbasi (2022). However, phishers and spammers typically disguise the name of their websites by employing URLs that incorporate IP addresses for the purpose of obscuring their identity. For example, the website URL might look like "https://0xD88AC74F/". Apparently, the infrastructure for such spam and phishing websites is hosted at these hexadecimal IP addresses. Such a method has been sufficient to let such organizations elude detection while spewing massive amounts of email communications, even though web browsers can decipher hexadecimal IP addresses and load the

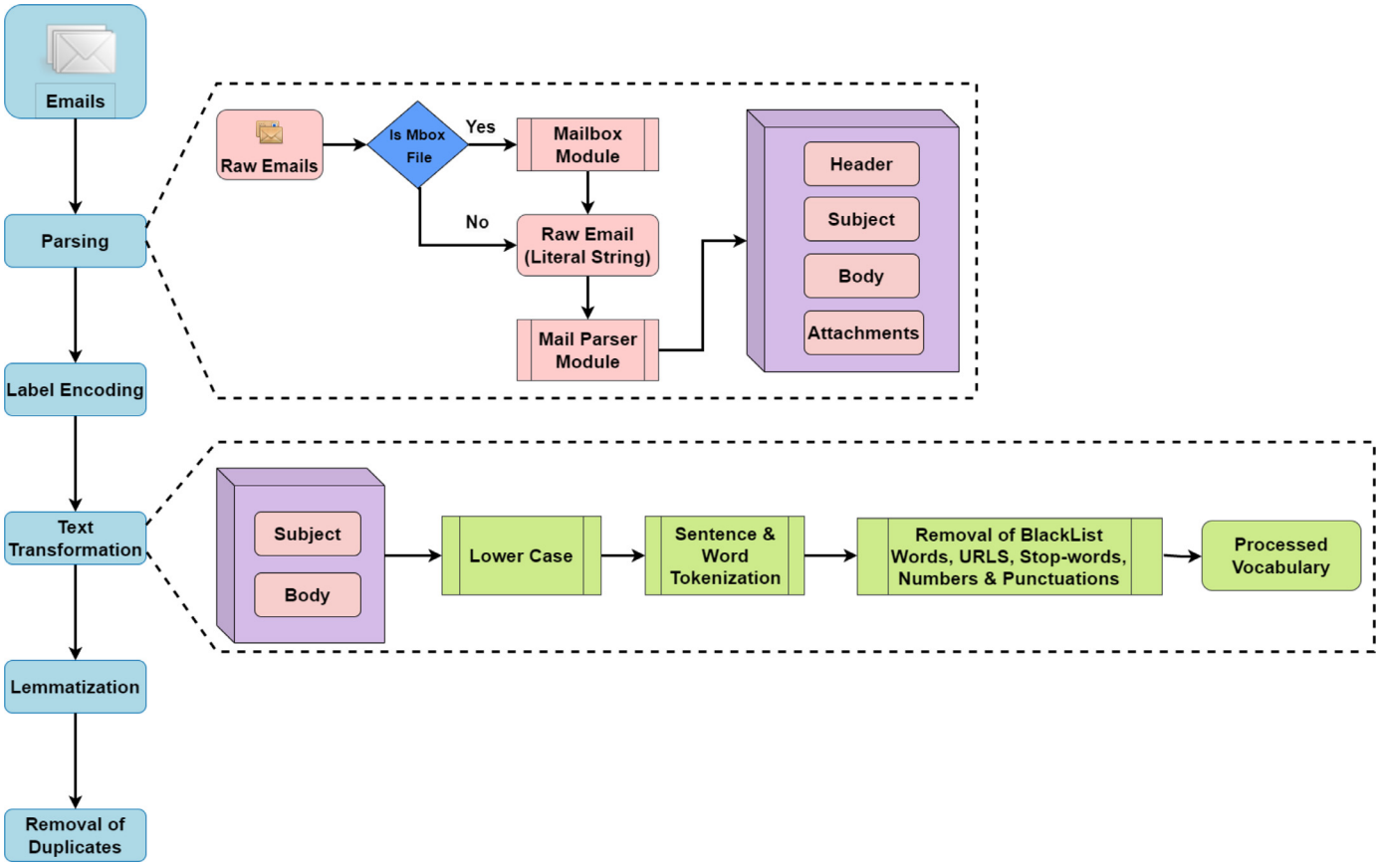

Fig. 2. Dataset preprocessing.

website discovered on the server. It is a Boolean attribute that returns 1 if the IP Address is present else returns 0.

As proposed in Emigh (2007), a domain name for a legitimate company does not have more than three dots. If a domain name in an email has more than three dots, a binary value of 1 is stored. For example, the website domain of "https://www.phishing.google.attack-userId.websearch.73i4654j6gkfdskjfiudsfg-fsdjf.com/" contains more than 3 dots, and thus, the link might potentially lead to a phishing website.

If an email is sent by a legitimate organization, then all the URLs present in the anchor tags will lead to the same domain name of the company. An attacker can trick users into thinking the phishing email is real by using a fraudulent domain name. An email that looks to be from a business address is initially more credible than one from a random domain (Ghosh, 2022). Look-alike domains are one of the most adaptable weapons in a cyber attacker's arsenal because they can be used to execute a wide range of attacks against a corporation. A comparable domain can be used to send emails, host a fake website, or even do both. Look-alike addresses are legal because they are authorized and use legitimate email servers to send emails. Because these malicious emails can get past many of the authentication measures that a company may have in place, there is a high danger of a breach.

The link to which the user has to be redirected is mentioned in the "href" attribute of the anchor tag. For example, ⟨a href="URL Address"⟩Plain Text⟨/a⟩. A legitimate website has the same website location as the plain text which is mentioned in the "href" attribute. As described in Fette et al. (2007), if the link text is a URL and it is a valid link, it should match the website address indicated by the "href" attribute (e.g., ⟨a href="http://www.google.com"⟩google.com⟨/a⟩); if there is a discrepancy between the href attribute and the link text (e.g., ⟨a href=http://www.phishing.com⟩ google.com⟨/a⟩), then the link will potentially lead to a phishing website.

Using the Beautiful Soup Library (Richardson, 2022), features like "containsImageTag", "containsScriptTag", and "containsStyleTag" are extracted from the dataset which represents whether the HTML Structure of the email contains an image tag, script tag, and style tag respectively. The total number of anchor tags present in the HTML Structure is represented by the "noOfAnchorTags" attribute. The "numCharactersBody" and "numCharactersSubject" attributes have the length of the body string and the length of the subject string respectively. The "combinedBodyText" attribute combines the Plain Text, the Context obtained from HTML Structure, and the Subject Context. Natural Language Toolkit (NLTK) (Steven Bird and Loper, 2009) is useful for text categorization, stemming, tokenization, parsing, tagging, and semantic reasoning of the sentence. Using the NLTK libraries, "numWordsBody" and "numWordsSubject" attributes are evaluated by counting the number of words in the string obtained after word tokenizing the body and subject of the email respectively. Similarly, the "numSentencesBody" and "numSentencesSubject" attributes are evaluated by counting the number of sentences in the string obtained after the sentence tokenizing the body and subject of the email respectively.

3.3.2. TF_IDF vectorization

A computer can only interpret numbers and cannot understand natural language. As a result, training a model to interpret natural language texts is challenging. The text corpus must be translated to numbers in such a way that it retains its value as a dataset. TF_IDF is one such text tokenization approach. TF_IDF is a statis-

tical method for valuing a word in accordance with the weighted relevance of its documents over the whole corpus. A document in a corpus is a single piece of text in a larger collection of texts used for analysis. A document in this research refers to as a single email. By counting the number of times a term appears in a certain document, term frequency, or *tf*, is determined. Term frequency signifies the importance of a term pertaining to a specific document. The inverse of a term's cumulative frequency over all documents are used to determine inverse document frequency or *idf*. Inverse document frequency signifies how common a term is in the corpus. TF_IDF is an effective method for removing more frequent terms from documents by giving them a lower value and thus less significance. The contrary is also true, elevating less common terms to greater relevance by giving them a higher value.

3.4. Feature selection

Feature Selection is a process to find an optimal set of features from a pool of features in the training data. It scales down the number of input attributes required to train the classification model. Feature Selection reduces the dimensionality of the dataset and thus, increases the overall performance and brings down the computational cost of modeling. Feature selection is performed on the features obtained by domain knowledge. Different methods are used for feature selection in the proposed study and are discussed below.

3.4.1. Exhaustive feature selection

The exhaustive feature selection algorithm evaluates the best possible combination of feature subsets by brute-force approach. By maximizing a specific performance metric with a random regressor or classifier, the best subset is chosen.

In this research, Exhaustive Feature Selection has been employed with Naive Bayes, Logistic Regression, KNN, Random Forest, Decision Tree, and Adaboost classification algorithms. As the method calculates all feature sets for each algorithm and compares the results, it can be computationally demanding when applied to all possible combinations of feature sets for each model, making it impractical. Due to the computational limitations of this method, other feature selection methods have been explored.

3.4.2. Information Gain

Information Gain estimates the decrease in entropy, which measures the impurity or uncertainty in a set of observations and identifies how a decision tree divides data. By assessing each variable's gain in relation to the target variable, information gain can be used in feature selection.

Fig. 3 shows the Information Gain Scores for different features. 0.1 being the threshold value for feature selection, the final features selected by the Information Gain method are: "htmlStructure", "noOfDots", "containsSameHrefInnerText", "containsImageTag", "noAnchorTags", "numCharactersBody", and "numWordsBody".

Table 2
Chi-square test scores.

Attribute Name	Score
numCharactersBody	63147.48
numWordsBody	56675.91
numSentencesBody	6905.15
htmlStructure	2163.96
noOfDots	2123.71
containsImageTag	1763.24
numCharactersSubject	1333.58
containsSameHrefInnerText	1261.17
containsStyleTag	884.03
containsIpAddress	860.58
attachments	254.48
noAnchorTags	237.02
formStructure	125.73
numWordsSubject	67.29
containsScriptTag	51.08
containsUniqueDomain	43.96
numSentencesSubject	29.77

3.4.3. Chi-Square Test

The chi-square test is used in statistics to determine if two events are independent or not. The expected count E and observed count O are obtained from the data of two variables. Chi-Square Test calculates the deviation between the observed count O and the expected count E. Greater Chi-Square score shows that there is greater relevance between the category and the term.

Table 2 shows the scores obtained after implementing the Chi-Square method for feature selection. 1000 being the threshold value for feature selection, the final features selected by the Chi-Square Test method are: "containsSameHrefInnerText", "numCharactersSubject", "containsImageTag", "noOfDots", "htmlStructure", "numSentencesBody", "numWordsBody", and "numCharactersBody".

3.4.4. F-Classif

F-Classif is a parametric statistical hypothesis method used to determine whether or not the means of more than one sample of data come from the same distribution. It is used when there is one numeric variable and one categorical variable, for instance, input variables as numeric and a categorical classification target variable in a classification problem. The features unrelated to the target variable can be eliminated from the dataset using the test results as a guide for feature selection. Fig. 4 shows the F-Classif Scores for different features.

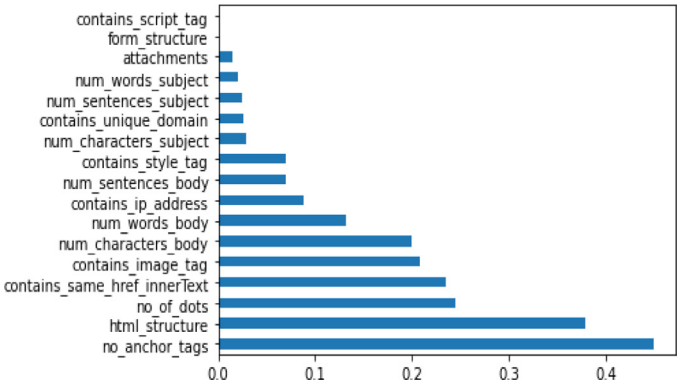

Fig. 3. Information Gain Scores.

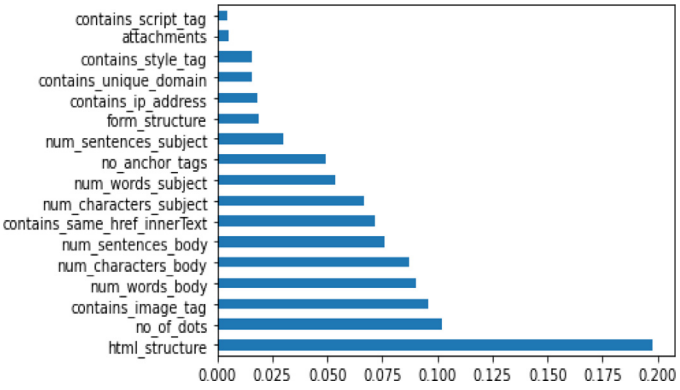

Fig. 4. F-Classif Scores.

For selecting final features by the F-Classif method, 0.075 is selected as the optimal threshold value. The final features selected are: “htmlStructure”, “noOfDots”, “containsImageTag”, “numCharactersBody”, “numWordsBody”, and “numSentencesBody”.

4. Algorithms

4.1. Traditional machine learning algorithms

Detection of an email as spam, phishing, or ham is a classification problem. Data is used by Machine Learning algorithms (Selig, 2022) to train and produce precise results. The goal of Machine Learning is to create a computer program that can access data, identify patterns in it, and use it to educate itself. Machine Learning contains various algorithms that are primarily divided into two distinct categories - Supervised-Learning and Unsupervised-Learning.

4.1.1. Support Vector Classifier (SVC)

The support vector machine generates a single hyperplane or group of hyperplanes in a high-dimensional space that may be used for tasks like outlier detection, regression, or classification. The algorithm is supervised machine learning, and one of its objectives is to identify the hyperplane with the separation between the two classes or maximum margin. SVM (Aswathisasidharan, 2022) has the ability to perform both linear and non-linear classification with the help of different kernels.

4.1.2. K-Nearest Neighbours classifier (KNN)

The K-Nearest Neighbor supervised machine learning classifier Developers assigns new sample data to the class that is nearest in similarity to the existing categories based on the assumption that new sample data and existing sample data are similar. The count of variables for prediction or classification which are closest neighbors of a new unknown variable is denoted by “K”. Different distance methods, such as Euclidean distance, Minkowski Distance, Cosine Distance, and Manhattan Distance, are used to evaluate, compare and validate similarities.

4.1.3. Naive Bayes classifier (NB)

The Naive Bayes Classification method Nagesh Singh Chauhan, KDnuggets, a conditional probabilistic classification methodology, is based on an extension of the Bayes theorem with independence assumptions in-between attributes. Naive Bayes Classifiers may be roughly divided into three groups based on the kind of predictor values and the type of distribution of those values: Bernoulli Naive Bayes Classifier BNBC, Gaussian Naive Bayes Classifier Majumder, and Multinomial Naive Bayes Classifier Kharwal.

4.1.4. Decision Tree classifier (DT)

The decision tree classification algorithm Navlani is a supervised machine learning approach that is used for both classification and regression. Leaf nodes are the outcomes of decisions and have no more branches, whereas Decision nodes are utilized to create decisions and such nodes contain many branches. The conditions that will most effectively split the sample data and predict the category of new sample data are used to partition the sample data recursively.

4.1.5. Logistic Regression classifier (LR)

When a binary result (Yes/No, True/False, 1/0) is anticipated, the supervised classification algorithm known as logistic regression is typically used. A Logistic Regression Classifier (George Lawton Ed Burns, 0000) generates results as a probability between 0 and 1.

4.1.6. Random Forest classifier (RF)

A Random Forest is made up of a lot of decision trees that act together as an ensemble. The random forest takes predictions from each decision tree instead of just one and classifies using the outcomes of the majority of estimates. The random forest technique Donges is more accurate because there are more decision trees in it, which mitigates the effects of overfitting sample data.

4.1.7. Gradient Boosting classifier (GBDT)

Gradient boosting classifier Saini is a collection of machine learning approaches that integrate numerous poor machine learning models to create an effective classification model. The purpose of gradient-boosting classifiers is to decrease the loss, which is the difference between the actual class value of the training sample data and the expected class value.

4.1.8. Extreme Gradient Boosting classifier (XGB)

Extreme Gradient Boosting Classifier Jason Brownlee is a Gradient Boost classifier approach in which decision trees are created in a sequential manner. Numerous trees are built, and the resultant value of all the trees is used to make the final forecast. The values of the trees are scaled by the learning rate.

4.1.9. Bagging classifier (BgC)

By averaging or voting, bagging minimizes overfitting (variance); nevertheless, this increases bias, which is counterbalanced by the decrease in variance. A Bagging Classifier Moamen Elabd is an ensemble estimator that fits base classifiers individually to randomized subsets of the dataset, and then combines the individual outcomes to generate a final classification.

4.1.10. Extra Tree Classifier (ETC)

Extra Tree Classifier (MI—extra tree classifier for feature selection, 0000) generates numerous trees and separates nodes using randomized subsets of characteristics, just like Random Forest Classifier. It does not generate observations since it samples without replacing and splits nodes using randomized splits instead of the optimum splits. Extra Tree Classifier’s randomness stems from the random feature selection and splitting of all observations, not from data bootstrapping.

4.1.11. Adaptive Boosting Classifier (AdaBoost)

AdaBoost or Adaptive Boosting Classifier Sarkar is an Ensemble Learning approach. Decision trees are generally used with AdaBoost with one level or with only one split. This approach generates a model by assigning equal weight to each data piece. As a result, it provides more weight to data pieces that are erroneously classified.

The research study (Salloum et al., 2021) has performed a detailed survey on different machine learning and deep learning techniques for efficient email phishing detection and their survey concluded that deep learning architectures like CNN & RNN prove to be more promising and efficient results.

4.2. Deep learning models

Deep Learning is a subclass of Machine Learning where layers of different networks come in relation. A deep learning model learns by using its own computing algorithm. Even though basic machine learning models achieve better performance at executing their respective tasks as new data is provided, they still need some human assistance. A deep learning model allows an algorithm to evaluate the accuracy of an outcome using its own neural network without the intervention of a person.

Deep learning uses multiple algorithms in a continuing series of actions. Similar to how the human brain makes decisions, it

processes massive amounts of data simultaneously using a variety of techniques. The ability of deep learning algorithms to visualize non-linear and complex relationships in the data is highly essential to their performance.

4.2.1. Artificial neural network

Artificial neural networks, frequently referred to as computer networks, are built on the basis of biological neural networks, which construct the human brain's structure. Neurons in artificial neural networks are linked to one another in the same way as neurons in the human brain are linked to one another at different levels of the networks. Artificial Neural Networks are made up of three layers: the Input Layer, the Hidden Layer, and the Output Layer. The Artificial Neural Network receives an input signal from an external source and quantitatively allocates it using the notation x(n) for each n-th input.

4.2.2. Convolutional neural network

A Convolutional Neural Network Choubey is a Deep Learning technique that can recognize and rank many characteristics and objects in an input image (weights and biases). CNNs learn characteristics with adequate training, whereas in primitive methods, filters are hand-engineered. They are made up of three major layer types: convolutional, fully-connected (FC), and pooling. VGG, ResNet, and Inception are some commonly used CNN models.

4.2.3. Recurrent neural network

Recurrent neural networks are a form of artificial neural network that employs sequential data patterns or time series data. They are widely used in natural language processing (NLP) applications such as NLPDL, translation services, speaker recognition, and picture captioning. Unlike standard deep neural networks, they feature a "memory" that stores all data linked to computations, and the output is sequentially dependent on preceding parts.

In the proposed work, Long Short-Term Memory (LSTM) (Shipra Saxena, 0000), a type of Recurrent Neural Network (RNN) specifically designed to handle sequential data with long-term dependencies, is used for classification tasks. LSTMs can analyze and understand the context in the text by retaining important information from previous inputs, allowing them to make accurate classifications.

4.3. Data resampling: Undersampling and oversampling

As described in the section on dataset collection, it is clear that the dataset acquired suffers from data imbalance. This problem of data imbalance is common in machine learning, indicating a substantial difference in the number of instances between classes. This uneven nature frequently leads to poor machine learning model training and poor performance when classifying new data instances. Traditional data resampling techniques such as undersampling and oversampling are employed to overcome this issue. This method improves the performance of a machine learning model trained to classify unknown data instances. This approach of resampling has been employed and evaluated while training models for all the algorithms mentioned in the above sections.

5. Proposed architecture

While classifying the types of emails, an issue with data imbalance in the dataset is encountered. To cater to this problem, the authors have proposed a novel dual-layer architecture that can effectively address the issue of data imbalance.

The dual-layer architecture consists of two interconnected layers, each equipped with a trained or pre-trained model that can classify data instances into their respective class labels. Specifically, this architecture is designed to handle an imbalanced dataset with N number of class labels, where some classes have significantly more data instances than others. To differentiate between classes with high and low-frequency data instances, the proposed architecture defines majority classes as those that have more data instances than the minority classes. Conversely, minority classes are those that have fewer data instances. By leveraging the trained or pre-trained models in each layer, the dual-layer architecture can effectively handle the imbalanced dataset and classify data instances into their respective class labels.

5.1. Training

According to the mechanism of the Dual Layer architecture proposed in this study, Layer 1 of the two-layer pipeline involves a specific dataset transformation for model training. This transformation involves clubbing all the data instances of minority classes together and relabelling them as 'Hybrid'. Meanwhile, the class labels for the majority of classes remain the same as they are originally. A model (M1) is then trained on this transformed dataset to classify the data instances into either the Hybrid class or their respective majority classes. Moving on to Layer 2, the entire dataset is fed into this layer, but only the data instances belonging to the minority classes are used for model training. Unlike Layer 1, the minority class data instances are not relabelled and remain as they are originally. A model (M2) is trained on this trimmed dataset to classify the data instances into one of the minority classes.

To better illustrate this architecture, Fig. 5 in the study shows the general architecture for training using the Dual Layer approach. The majority class labels are denoted as M_{a_0}, M_{a_1}, Q, M_{a_n}, while the minority class labels are denoted as M_{i_0}, M_{i_1}, Q, M_{i_n}. In Layer 1, the minority classes M_{i_0}, M_{i_1}, Q, M_{i_n} are combined to form a single Hybrid Class and are supplied to Layer 1 for training the model to classify data instances into both the majority classes (M_{a_0}, M_{a_1}, M_{a_2}, Q, M_{a_n}) and the Hybrid Class. In Layer 2, only the data instances belonging to the minority classes(M_{i_0}, M_{i_1}, Q, M_{i_n}) are supplied to train the model to classify data instances into the specific minority classes(M_{i_0}, M_{i_1}, Q, M_{i_n}).

5.2. Testing

The classification process for new data instances in the dual layer approach will be such that if Layer 1 categorizes the data samples as Hybrid Class, then these samples will be supplied to Layer 2, where they will be classified among one of the minority classes. Whereas, at Layer 1, if the data samples are categorized as one of the Majority classes, they will not be supplied to Layer 2 and the calculated label will be considered as the predicted class for that data instance.

Fig. 6 illustrates the general flow of the proposed dual layer approach for data classification, where M_1 and M_2 refer to the trained or pre-trained models at Layer 1 and Layer 2, respectively. To classify a new data instance D_i, it will first be supplied to the model M_1 at Layer 1. M_1 will then categorize D_i into either a majority class(M_{i_0}, M_{i_1}, Q, M_{i_n}) or a hybrid class based on the input features. If D_i is categorized as one of the majority classes let's say M_{i_1}, then the predicted label for D_i will be M_{i_1} calculated from M_1 at Layer 1. However, if D_i is classified as a hybrid class by M_1, then D_i will be passed on to model M_2 at Layer 2, which will classify D_i among one of the minority classes(M_{i_0}, M_{i_1}, Q, M_{i_n}). Suppose D_i is classified as M_{i_1} then it will be considered as the final predicted label for D_i.

In conclusion, the proposed dual-layer architecture approach introduces a versatile and adaptable two-layer framework suitable for any dataset, emphasizing its data independence. Notably, this

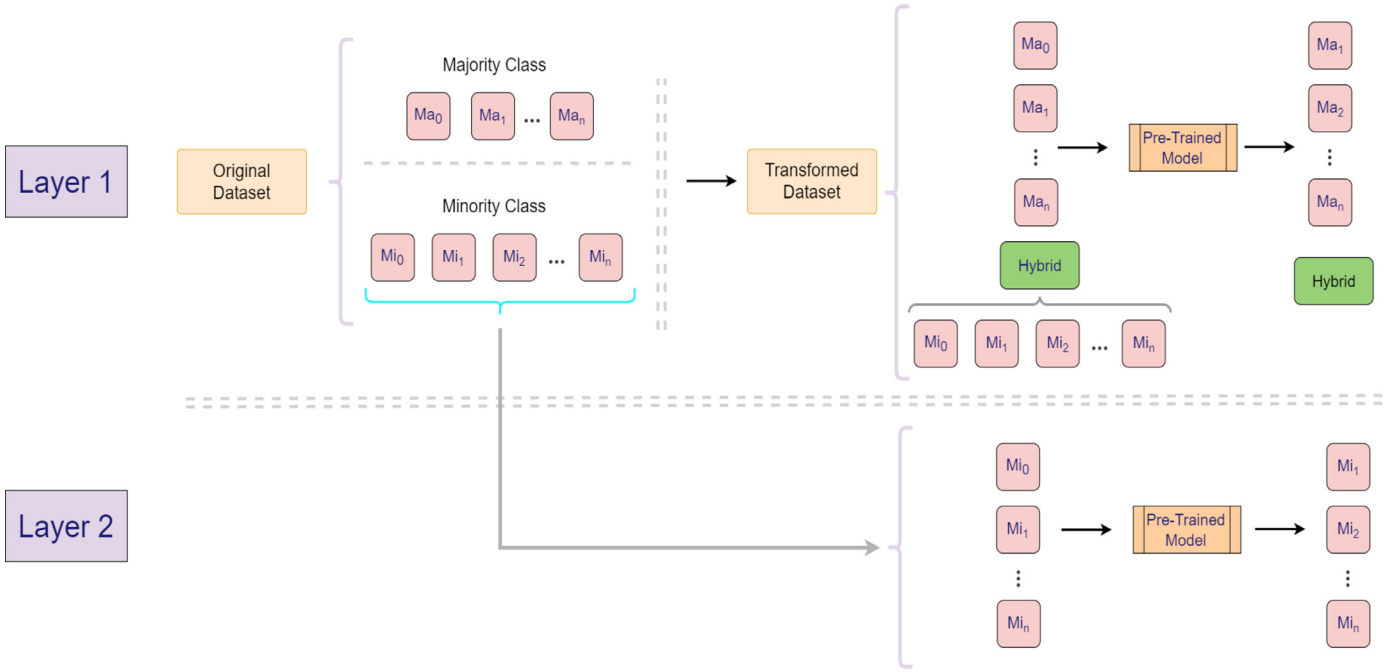

Fig. 5. Dual-Layer Architecture - Training.

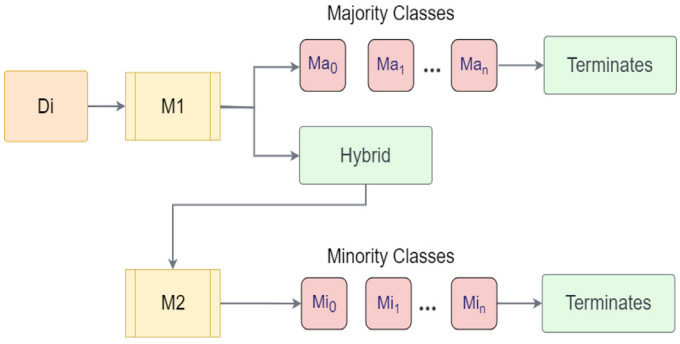

Fig. 6. Dual-Layer Architecture - Testing

approach is flexible in terms of feature engineering and data preprocessing, allowing for various preprocessing techniques to be applied without affecting the model's performance or specifics. Moreover, as highlighted earlier, the dual-layer architecture remains model-independent, enabling the utilization of any ML/DL model for classification at each layer, whether pre-trained or trainable. Importantly, this dual-layer approach not only achieves high classification performance, as discussed in the results section but also effectively addresses the prevalent issue of data imbalance by efficiently classifying minority class instances. This Dual Layer Approach is employed in Custom Deep Learning Model which is illustrated in the Experimentation Section.

6. Experimentation

The experimentation's conducted by the authors in this research aimed to enhance the performance of both traditional and deep learning models by optimizing their parameters within specific ranges. This process involved analyzing the input features used for training, specifically analyzing the vocabulary words and size of the vocabulary for TF-IDF vectorization and selecting domain-specific features. These experiments are conducted on both traditional and deep learning models to identify the optimal set of TF-IDF vectorization features that would improve the overall performance of the models. The results of these experiments are discussed in more detail in the results section.

Figs. 7, 8, and 9 showcases the optimized architectures of custom deep learning models (ANN, CNN, RNN) that achieved maximum performance after hyperparameter tuning. These models are developed from scratch, ensuring that the architecture is specifically tailored to the problem at hand. The experimentation process for building these custom models involved varying the learning rate and batch size hyperparameters. Moreover, architectural parameters such as the number of nodes, hidden layers, dropout layers, optimizers, and activation functions have also been optimized to enhance performance. In the case of CNNs, experimentation has been carried out on the number of nodes in the convolution and max pooling layers. Similarly, for the LSTM deep learning model, tuning has been performed on the number of LSTM units.

6.1. TF_IDF variations

To determine the optimal vocabulary to train the model, three different variations have been implemented and observed. This helped to obtain the optimal version of the TF_IDF Vectorizer for each trained model. All the mentioned variations have been trained on 12 algorithms namely SVC, KN, NB, DT, LR, RF, AdaBoost, BgC, ETC, GBDT, and XGB as well as Neural Networks.

6.1.1. Whole corpus

The entire vocabulary forms the corpus for the TF_IDF vectorizer. After removing duplicate words, a total of 59,557 unique words are there in the vocabulary.

6.1.2. Max features with N-gram range

In this variation, a combination of the N-gram range with max features has been used. The N-Gram ranges considered are (1,1), (1,2), (1,3), (2,2), (2,3), and (3,3). The values of max_features considered are 1000, 2000, 3000, 4000, 5000, 10000, 15000, 20000, 25000, 30000, and 35000. Every viable combination of the N-gram range and max features has been experimented with to obtain the optimal set of hyperparameters for the corresponding algorithm.

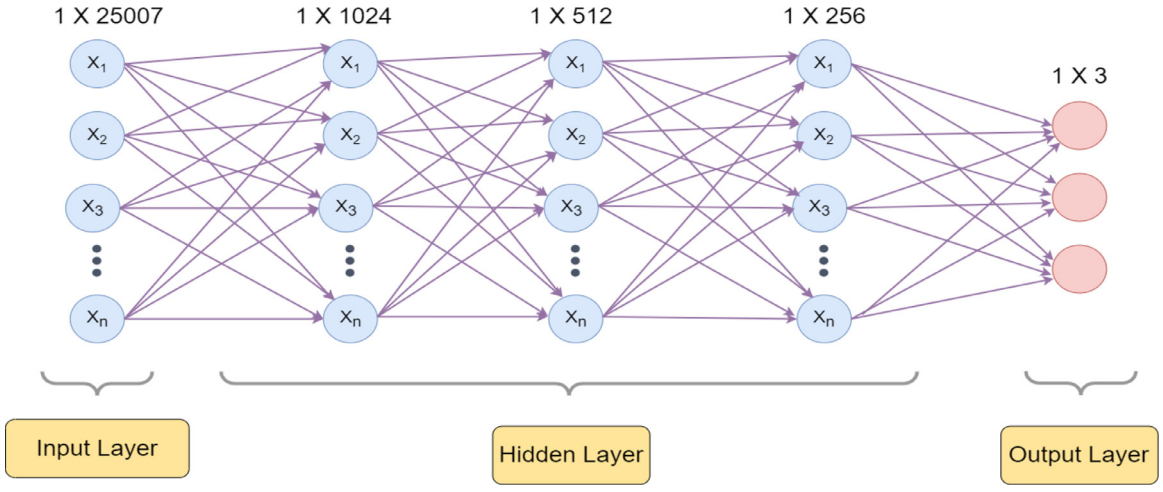

Fig. 7. ANN Model Architecture.

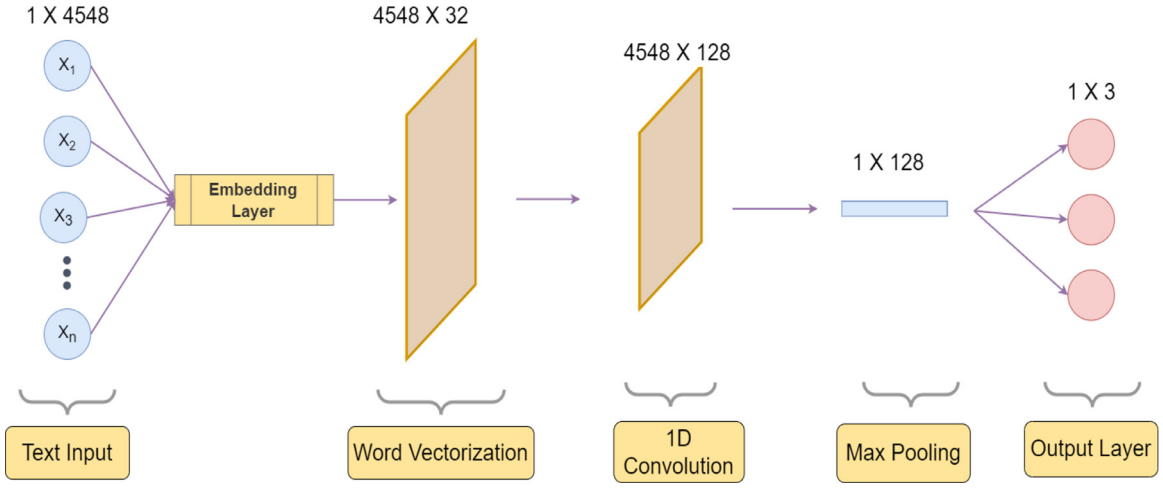

Fig. 8. CNN Model Architecture.

6.1.3. Min_df with N-gram

A combination of min_df (minimum document frequency) and n-gram range is experimented with. The n-gram ranges are the same as in the 'Max Features with Ngram range' variation. The values of minimum document frequency as a lower bound are 2, 3, 5, 10, 15, 20, and 25. In this variation as well, every possible variation of the n-gram range with min-df is considered while training.

6.2. Training model with resampled dataset

Before resampling, the dataset is divided into two parts: the training set and the testing set. The training set accounts for 80% of the total dataset. 80% of all class instances are included in the training set. Thus, the spam, ham, and phish classes each have 1078, 2137, and 3338 training instances. This collection of data instances is then resampled by undersampling or oversampling. The resulting set is used to train the model with machine learning techniques and deep learning models.

In undersampling the majority classes are resampled to be equal to the minority class in terms of class distribution, hence the ham and phish classes are resampled from 2137 and 3338 instances respectively to 1078 instances per class. To resample the classes, instances from each class are randomly deleted to reduce the total class size.

Oversampling is performed by duplicating the ham and spam class instances until the number of instances per class equaled the phish class, i.e., the majority class. This increased the number of ham and spam instances from 2137 and 1078, respectively, to 3338 instances per class.

6.3. Dual layer approach

The class distribution as depicted in Fig. 1 shows that the Phish class is 36% greater in size than the Spam class and 20% greater in size than the Ham class. Thus, there is a huge difference in the occurrences of data instances for the Ham and Spam classes in comparison to the Phish class. Additionally, it can be noted that the differences between the classes exceed the total class size of the minority classes, i.e., the difference between the Spam and Phish class i.e. 3017 is greater than the total number of instances of the Ham class (2664) and the difference of Ham and Phish class i.e. 1703 is greater than the total class size of spam class (1350).

Hence to overcome this high data imbalance, dual-layer architecture is implemented on this dataset. According to the Dual Layer Architecture examined in the Proposed Methodology section, Spam and Ham being the minority classes will be relabelled to form the 'Hybrid' class, whereas Phish being the majority class will not be altered.

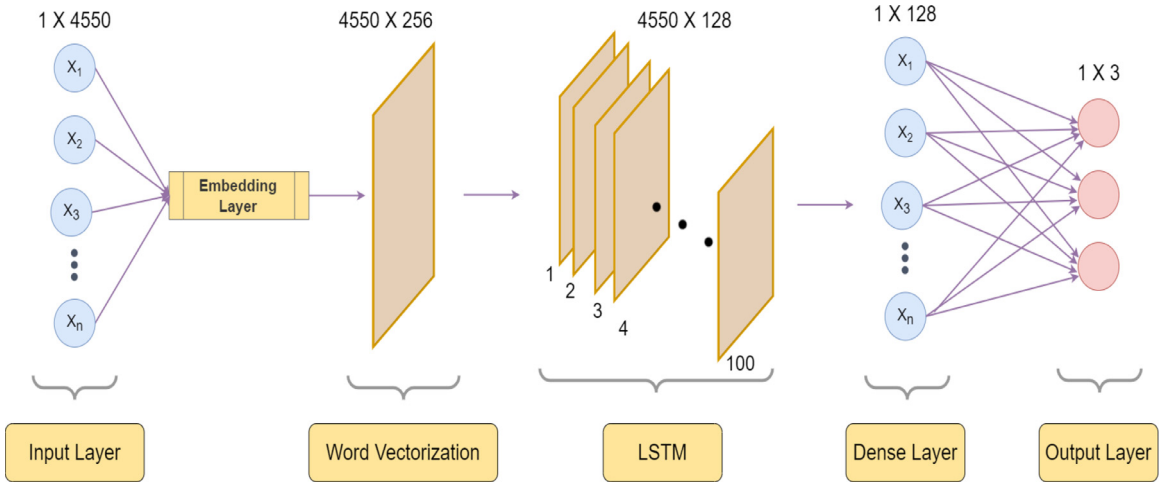

Fig. 9. RNN Model Architecture.

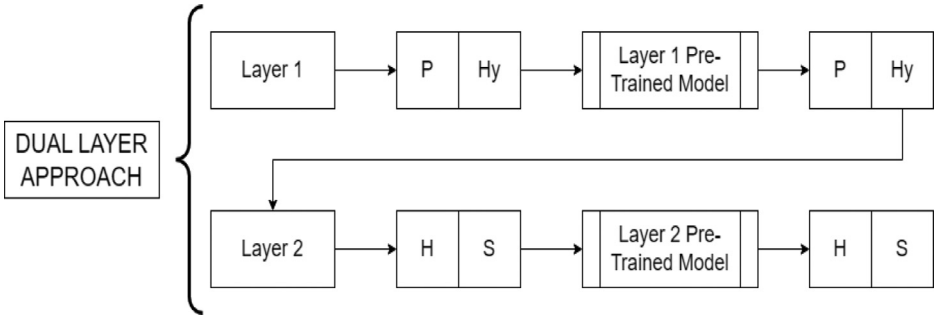

Fig. 10. Dual Layer Approach.

Fig. 10 illustrates the class distribution of the dataset for the dual layer approach where 'P' stands for the Phish class label, 'Hy' stands for the Hybrid class label, 'S' stands for the Spam class label and 'H' stands for Ham class. The model block in the diagram represents the trained or pre-trained model used for the classification of data instances in respective layers. Figs. 7, 8, and 9 represent the architecture of custom ANN, custom RNN, and custom CNN deep learning models, respectively, employed for the role of the trained or pre-trained model in the dual-layer architecture. Deep Learning models are taken into consideration for the implementation of the dual-layer approach as they proved to be effective in comparison to traditional methods which is discussed in the result section.

For this architecture, if a new data instance has to be classified, initially it will be supplied to the first layer of the architecture. If the Layer 1 model categorized that data instance as Phish then that will be considered as its predicted class label. If the Layer 1 model classifies the data instance as Hybrid it will be supplied to the second layer where the Layer 2 model will be employed to classify the data instance among Ham and Spam classes and that calculated class label will be considered as the predicted class by the dual layer architecture. Performance Metrics of the architecture are calculated by comparing the actual labels and the final predicted labels obtained from the dual-layer architecture.

7. Results

This section discusses the results and its analysis, observed for the trained model for the different experimentation methods discussed thus far.

7.1. TF_IDF result analysis

This section analyses the optimal TF_IDF variations for each individual algorithm. Figs. 11, 13, and 12 show the trends in performance metrics for accuracy, average precision, and average recall for each algorithm when compared to all TF IDF variations.

Table 3 summarizes the key findings regarding the variation of TF_IDF which delivers the optimal results for each performance metric when applied to an algorithm. The Accuracy column in Table 3 represents the optimal set of TF_IDF variations when the accuracy metric is employed to evaluate the algorithm. Similarly, columns Recall and Precision represent the optimal iteration of TF_IDF variation when recall and precision are assessed for algorithm evaluation. Column Overall portrays the optimal set of TF_IDF variation when all three performance metrics of Precision, Recall, and Accuracy are simultaneously analyzed for algorithm evaluation. Finally, the Column Hyperparameter states the optimal values of the hyperparameter observed for the corresponding optimal variation for the corresponding algorithm.

It is evident from Table 3 and Figs. 11, 13 and 12 that the Whole Corpus being a Naive Approach is not an optimal variation of TF_IDF for any algorithm. It is imperative to note that barring the Whole Corpus variation, for almost all algorithms, both TF_IDF variants - Max_Features and Min_DF are able to produce comparable optimality for all performance metrics.

It can be drawn from Figs. 11, 13, and 12 that for a majority of the algorithms when evaluated against all performance metrics, although marginally, Max_Features have outperformed Min_Df variation of TF_IDF vectorizer.

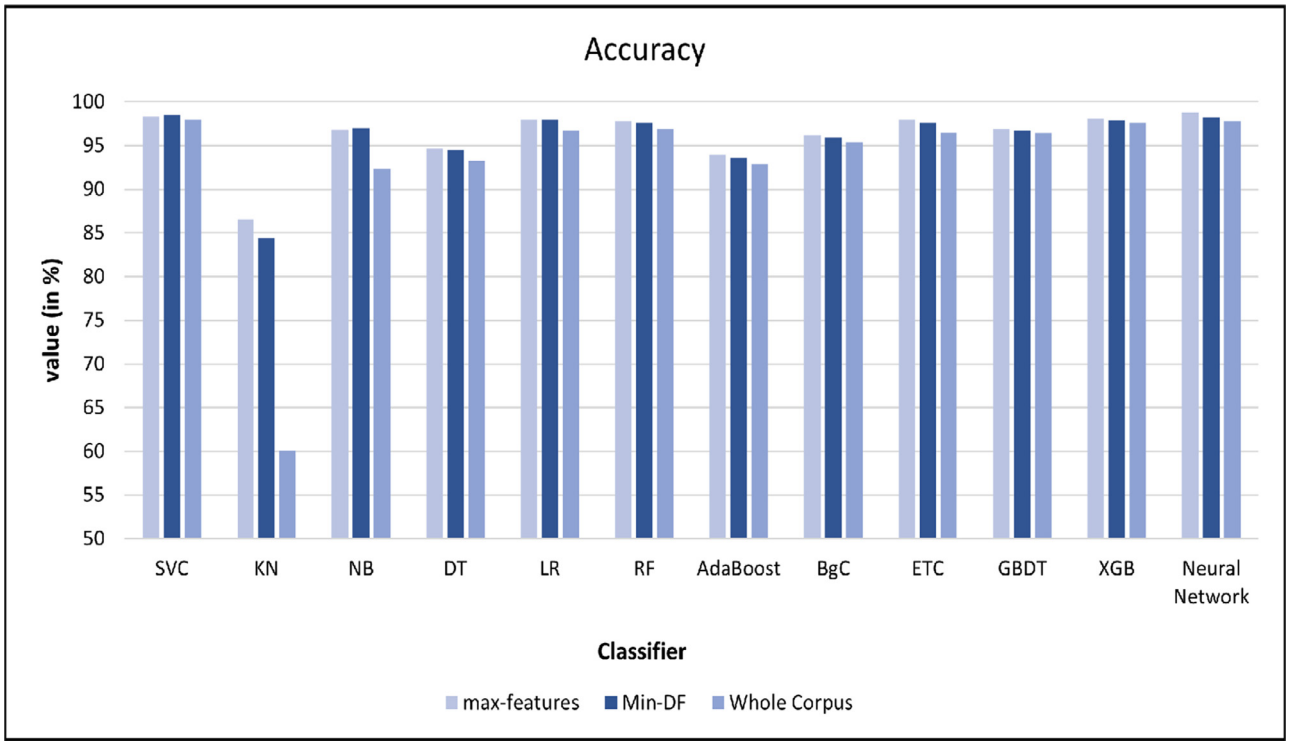

Fig. 11. Comparison of Accuracy values for the TF_IDF variations.

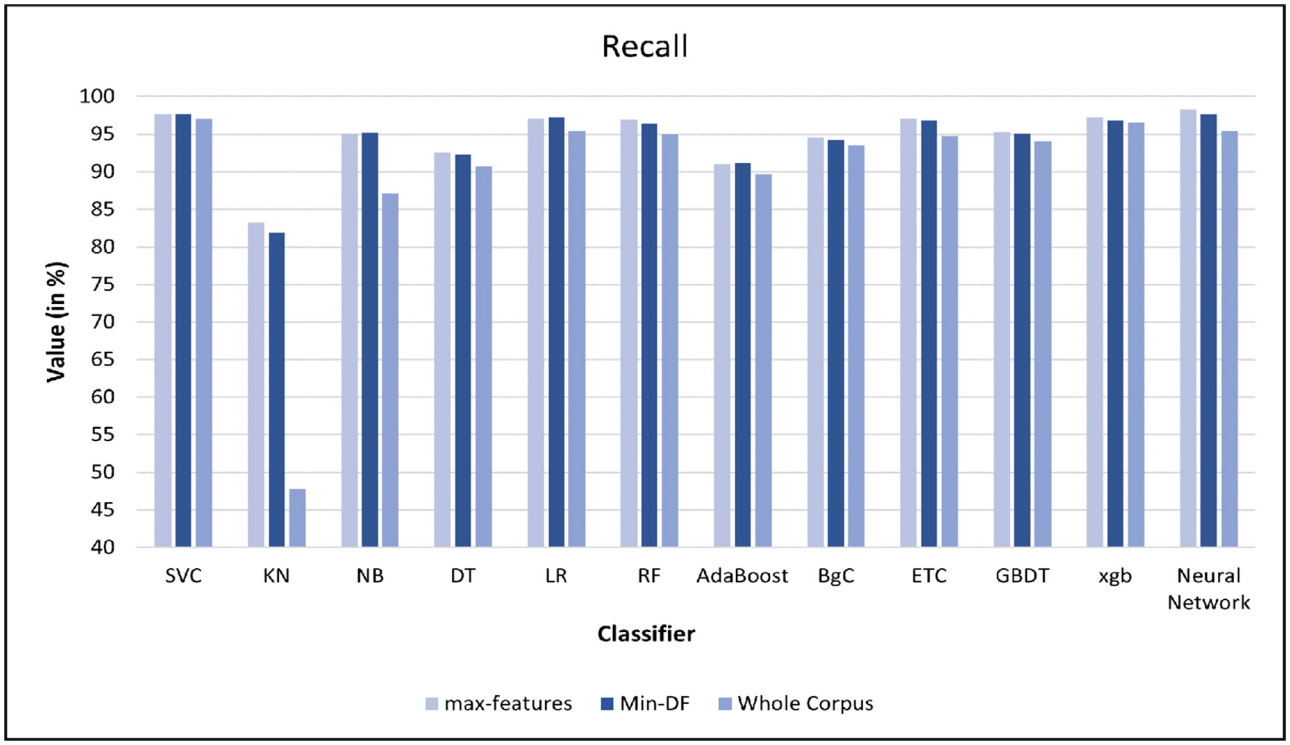

Fig. 12. Comparison of Recall values for the TF_IDF variations.

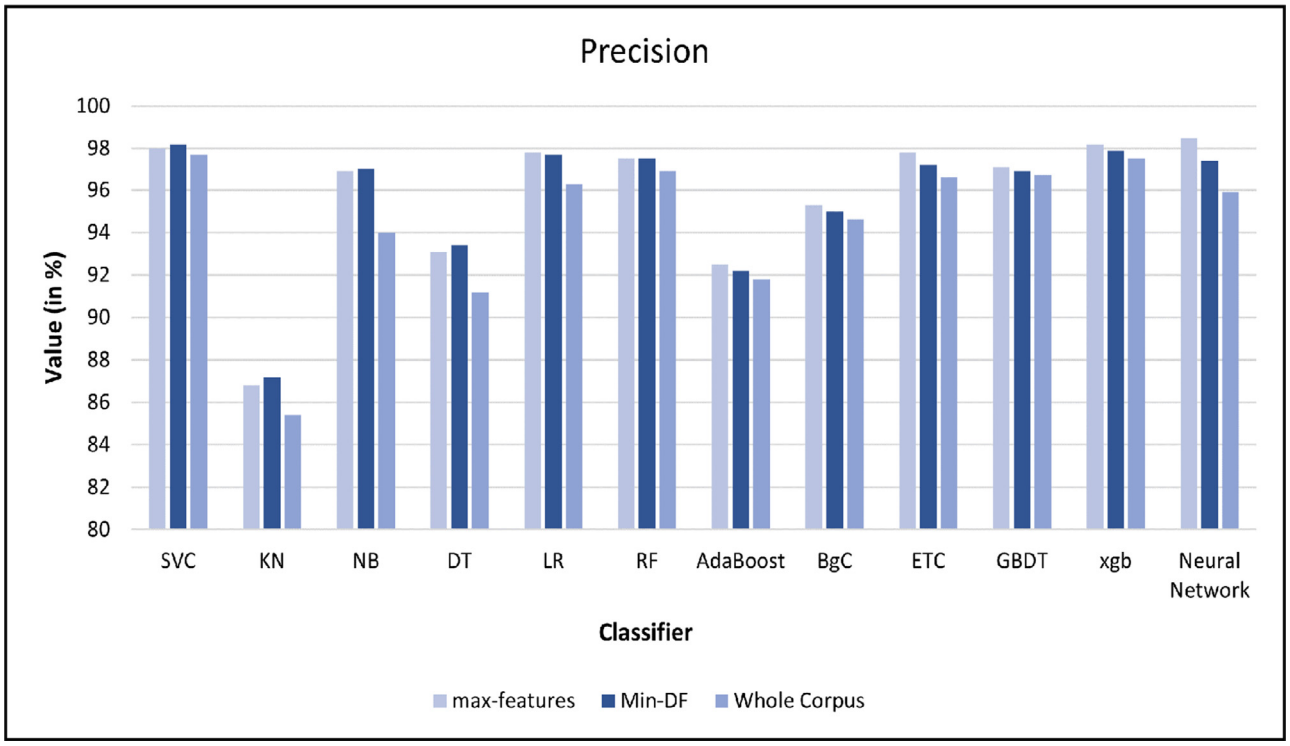

Fig. 13. Comparison of Precision values for the TF_IDF variations.

Table 3
Optimal technique for evaluation parameters.

Classif- iers	Accuracy	Recall	Precis- ion	Overa- ll	Hyperpara- meters
SVC	min df	min df	min df	min df	(1, 2) 3
XGB	max features	max features	max features	max features	(1, 3) 15000
AdaBo- ost	max features	min df	max features	max features	(1, 2) 2000
RF	max features	max features	max features	max features	(1, 3) 2000
NB	min df	min df	min df	min df	(2, 2) 5
LR	min df	min df	max features	min df	(1, 3) 5
KNN	max features	max features	min df	max features	(2, 2) 1000
GBDT	max features	max features	max features	max features	(1, 2) 30000
ETC	max features	max features	max features	max features	(1, 2) 15,000
BgC	max features	max features	max features	max features	(1, 3) 20000
DT	max features	max features	min df	max features	(1, 3) 10000
Neural Networks	max features	max features	max features	max features	(1, 3) 25000

7.2. Data resampling comparisons

This section analyzes the performance of a model trained on a dataset after resampling with undersampling and oversampling techniques, compared to the performance of the model trained on the original dataset. The performance metrics considered for evaluation include accuracy, precision, recall, and F1-score for each individual class.

7.2.1. Accuracy metric

Fig. 14 illustrates the accuracy performance metric noted on traditional and deep learning models trained on the original dataset, after resampling with undersampling and oversampling techniques. As shown in Fig. 14, while resampling techniques can address the issue of data imbalance and enhance the performance of models, they do not always result in a significant improvement in accuracy compared to training on the original dataset. In some cases, such as the XGBT model classifier, oversampling with 97.5% accuracy and undersampling with 97.6% accuracy proved to be effective in comparison to the normal approach with 96.9% accuracy. Similarly, oversampling with 90.8% accuracy outperformed the training on the original dataset having an accuracy of 90.23% in the case of KNN classifier. However, for other classifiers like LRC, SVC, ETC, and ANN, the resampling techniques obtained results at par with the normal execution on the original dataset. For the remaining classifiers, it can be noted that normal execution outperformed the resampling approach.

The use of resampling techniques results in the model failing to generalize in cases of undersampling and oversampling. The subsequent section of the research analyzes the performance metric of individual classes against the resampling and normal approach of model training.

7.2.2. Individual class performance

Accuracy measures the performance of the classifier as a whole system, but for a more comprehensive analysis, it is important to evaluate the performance of individual classes using precision, recall, and F1-score as metrics. These metrics will provide a deeper understanding of the classifier's performance for each individual class.

Figs. 15, 16 and 17 illustrate the performance of various models in terms of average precision, average recall, and average F1-

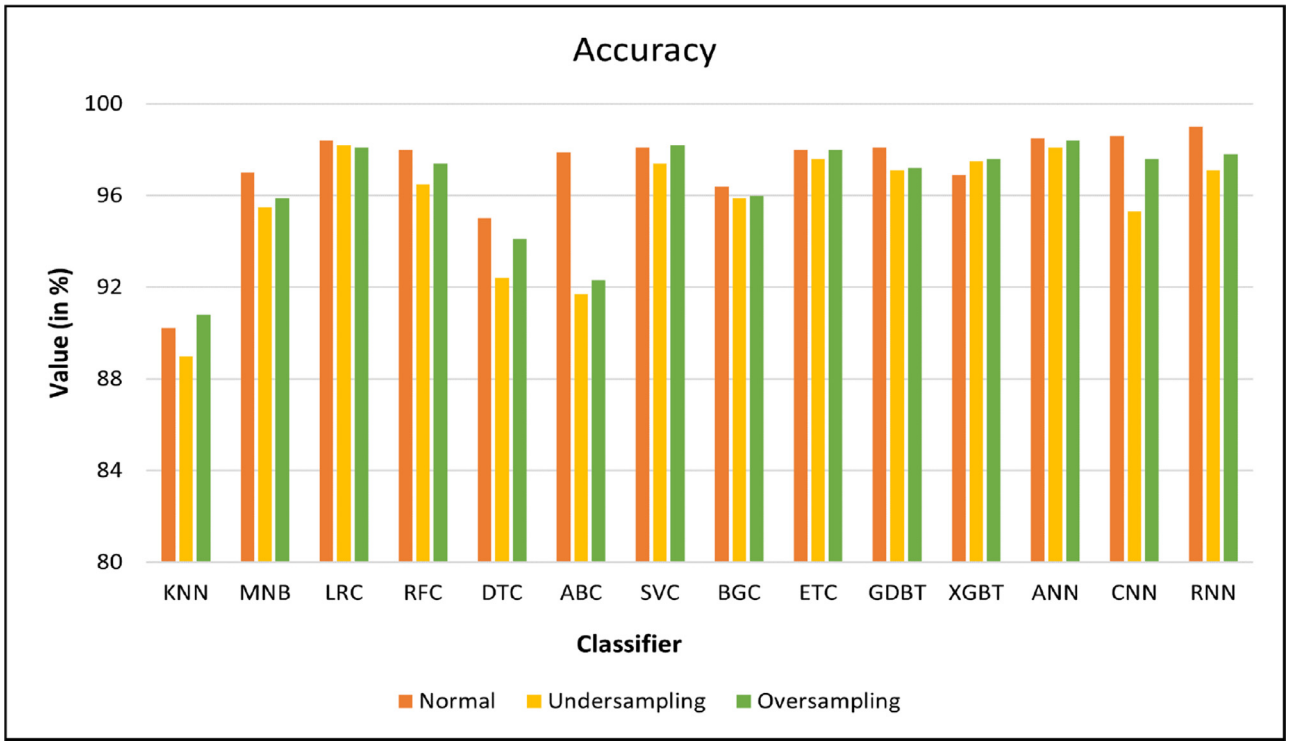

Fig. 14. Comparison of Accuracy for different Resampling Techniques.

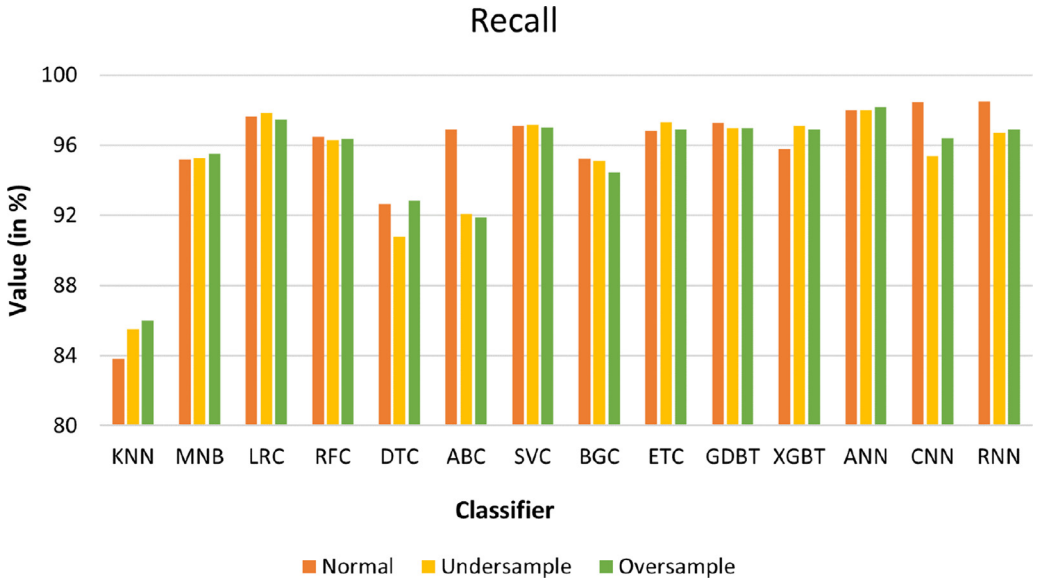

Fig. 15. Comparison of Recall for different Resampling Techniques.

score for the Phish, Spam, and Ham classes, when evaluated under both resampling and normal conditions. The figure demonstrates that the precision performance of most algorithms is better when trained on the original dataset, rather than using resampling techniques like undersampling and oversampling. However, there is an exception with the XGBT algorithm, which performs better when the model is trained on an oversampled dataset.

It is evident from Figs. 15, 16 and 17 and Tables 4, 5 and 6 that for the Recall performance metric, most of the algorithms showed improved performance with the resampling technique, with a maximum improvement of 2%. In particular, the KNN and XGBT classifier models exhibited a marginal enhancement of up to 2%. On the other hand, the ANN, ETC, SVC, DTC, LRC, and MNB recall metrics improved by a maximum of 1%. For the remaining classifier models, it is observed that the normal approach outperformed the resampling techniques.

Lastly, it can be noted from Figs. 15, 16 and 17 for F1-score performance metric for most of the algorithms normal approach outperformed the resampling technique. In the case of KNN, RFC, SVC, ETC, and XGBT the resampling technique outperforms the normal approach by a maximum of 1% margin.

7.3. Internal comparisons

The model is assessed by calculating the values for Accuracy, Recall for each class, Precision for each class, and F1-score for each

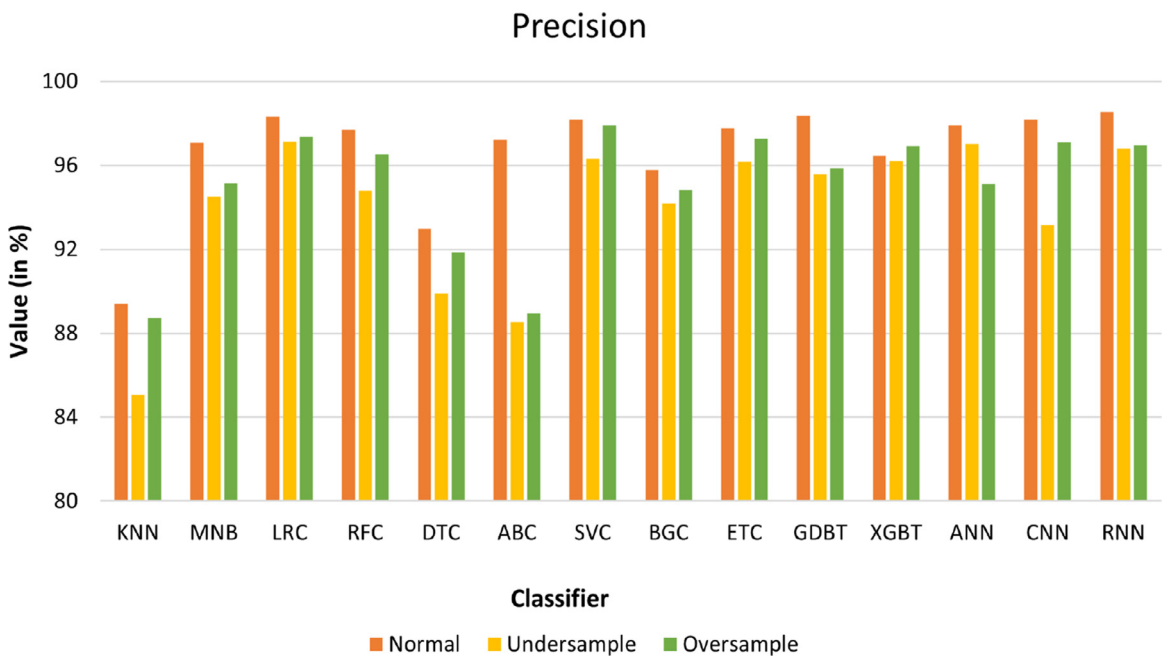

Fig. 16. Comparison of Precision for different Resampling Techniques.

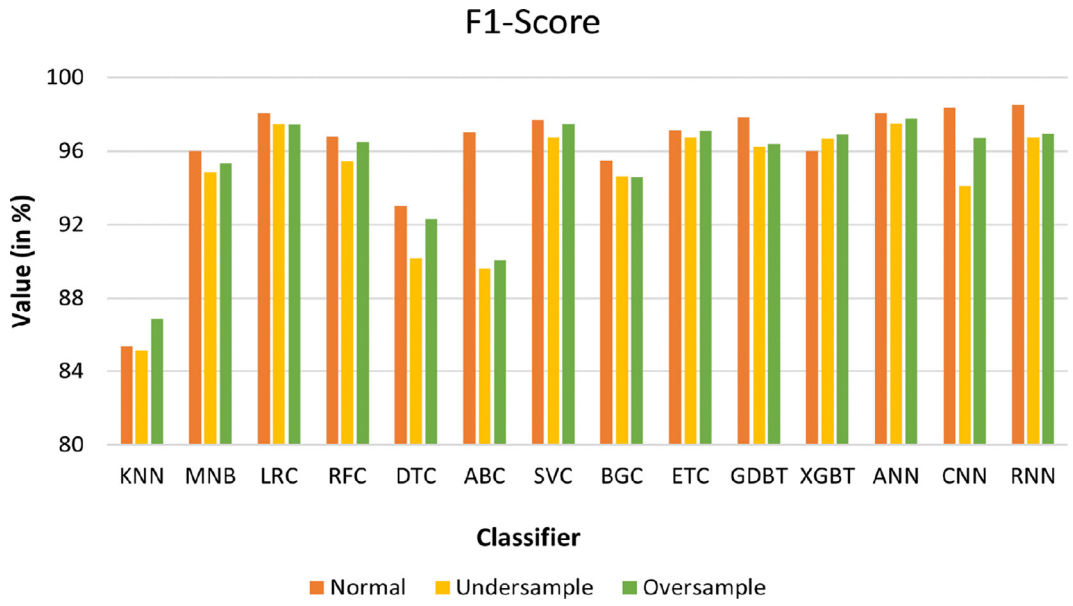

Fig. 17. Comparison of F1-Score for different Resampling Techniques.

Table 4
Performance metrics per class for a model trained on data when No-Resampling technique is applied.

Algorithms	Classifier	Accuracy	Ham Precision	Ham Recall	Ham F1-score	Spam Precision	Spam Recall	Spam F1-score	Phish Precision	Phish Recall	Phish F1-score
Traditional Algorithms	KNN	90.23	84.2	98.3	91	89	57.2	70	95	96	95
	MNB	97	95.2	98.3	97	98	88.2	93	98	99	98
	LRC	98.41	99	99	99	97	94.5	96	99	99.4	99.2
	RFC	98	98.6	97.1	98	96	93	94	98.5	99.3	98.4
	DTC	95	95	94	94	87	86	87	97	98	98
	ABC	97.9	98.3	98.7	98.5	94.7	93	93.8	98.7	99	98.8
	SVC	98.1	99.2	98.1	98.6	97.3	93.7	95.5	98	99.5	99
	BGC	96.4	98.1	96	97	92.2	91.1	91.5	97	98.6	98
	ETC	98	98.3	98.1	98.2	96	93	94	99	99.3	99.2
	GDBT	98.1	99	98.5	98.8	98.1	94	96	98	99.3	98.7
	XGBT	96.9	97	98	97	96.4	90	93	96	99.3	98
Deep Learning Algorithms	ANN	99	99.2	99.2	99.2	97	97	97	99.4	99.3	99.35
	CNN	98.6	99	99.4	99.2	96.3	97	96.8	99.2	99	99.1

Table 5
Performance metrics per class observed when Undersampling applied.

Algorithms	Classifier	Accuracy	Ham Precision	Ham Recall	Ham F1-score	Spam Precision	Spam Recall	Spam F1-score	Phish Precision	Phish Recall	Phish F1-score
Traditional Algorithms	KNN	89	85.5	97.7	91.2	72.8	69	70.8	96.9	89.9	93.3
	MNB	95.5	91	97.7	94.2	93	93.4	93.2	99.5	94.8	97.1
	LRC	98.2	98.7	98.7	98.7	93.2	96.3	94.7	99.5	98.5	99
	RFC	96.5	99	95.4	97.2	86.7	95.9	91.1	98.7	97.5	98.1
	DTC	92.4	95.7	88.2	91.8	78.5	87.5	82.7	95.5	96.7	96.1
	ABC	91.7	97.5	89.9	93.6	69.6	94.5	80.1	98.5	91.9	95.1
	SVC	97.4	99.4	97.3	98.4	91.3	96.3	93.7	98.3	97.9	98.1
	BGC	95.9	98.4	95.8	97.1	86.6	92.6	89.5	97.6	97	97.3
	ETC	97.6	99.2	97.1	98.2	90.3	96.7	93.4	99	98.1	98.6
	GDBT	97.1	98.3	97.1	97.7	89.4	96.7	92.9	99	97.1	98.1
	XGBT	97.5	98.8	97.5	98.2	90.9	95.9	93.4	98.9	98	98.4
Deep Learning Algorithms	ANN	98.1	98.5	98.5	98.5	93	97.4	95.1	99.6	98.1	98.9
	CNN	95.3	97.3	95.4	96.3	82.5	95.7	88.6	99.7	95.1	97.4
	RNN	97.1	97.8	97.5	97.6	94.8	95	94.9	97.8	97.6	97.7

Table 6
Performance metrics per class observed when Oversampling applied.

Algorithms	Classifier	Accuracy	Ham Precision	Ham Recall	Ham F1-score	Spam Precision	Spam Recall	Spam F1-score	Phish Precision	Phish Recall	Phish F1-score
Traditional Algorithms	KNN	90.8	86	97	91.2	84.4	66.1	74.1	95.8	94.9	95.3
	MNB	95.9	90.8	98.1	94.3	95.1	93.7	94.4	99.6	95.1	97.3
	LRC	98.1	98.3	98.9	98.6	94.8	94.8	94.8	99	98.7	98.9
	RFC	97.4	98.5	97.3	97.9	93	93	93	98.1	98.8	98.5
	DTC	94.1	95.1	93	94	83.1	88.9	85.9	97.4	96.6	97
	ABC	92.3	95.6	90.7	93.1	72.9	91.5	81.2	98.4	93.5	95.9
	SVC	98.2	98.7	98.5	98.6	96.9	93	94.9	98.2	99.6	98.9
	BGC	96	97.7	95.2	96.4	90	89.7	89.8	96.8	98.5	97.6
	ETC	98	98.5	98.1	98.3	94.8	93.7	94.2	98.5	99	98.8
	GDBT	97.2	98.1	97.1	97.6	90.6	96.3	93.4	98.9	97.5	98.2
	XGBT	97.6	98.1	98.1	98.1	94.1	94.1	94.1	98.5	98.5	98.5
Deep Learning Algorithms	ANN	98.4	92.2	98.9	99	93.6	97	95.3	99.5	98.6	99
	CNN	97.6	96.9	98.5	97.7	95.8	91.4	93.5	98.6	99.3	98.9
	RNN	97.8	98.3	98.1	98.2	94.8	95	94.9	97.8	97.6	97.7

class. The Recall, Precision, and F1-score values for each class indicate whether the model is able to correctly distinguish emails and categorize them correctly between ham, spam, and phishing.

The important thing to note here is that the comparisons are conducted using data that has not been resampled. The data's integrity has not been compromised because no changes have been made to it. Thus, the numbers in Table 4 are taken into account while analyzing and comparing. These observations are also used for comparison when analyzing the results of the Dual-Layer approach.

Out of all the 11 traditional machine learning algorithms over which the model is trained, the highest value of accuracy is observed by the logistic regression classifier algorithm, which provides an accuracy of 98.41%. On the other hand, when deep learning algorithms are implemented, 98.5% accuracy is the least value observed when training the model over a recurrent neural network. The ANN and CNN algorithms give an accuracy of 98.6% and 99% respectively.

While classifying Ham emails, the logistic regression classifier outperformed the recurrent neural networks. The performance metrics for logistic regression while classifying the Ham class are 99% for precision, 99% for recall, and 99% for F1-score. As for the recurrent neural networks, the values are 98.3%, 99%, and 98.88% for precision, recall, and F1-score respectively.

Similarly, for classifying the Phishing emails, the values of performance metrics for logistic regression are 99%, 99.4%, and 99.2% for precision, recall, and F1-score respectively. As for the RNN, the values observed are 99.2% precision, 99% recall, and 99.1% F1-score.

Whereas, for the spam email classification, it is observed that the recall values for deep learning algorithms are better than the traditional algorithms. The logistic regression classifier, with a recall value of 94.5%, has the highest recall value out of all the 11 traditional algorithms. On the other hand, the recurrent neural networks, with a recall value of 96%, have the least recall value out of the 3 deep learning algorithms. The ANN and CNN both have a recall value of 97%.

7.4. Dual layer comparisons

The values observed for the dual-layer neural network architecture implemented, as can be observed in Table 7, are higher as compared to the single-layer architecture (here, the single-layer architecture refers to a traditional fully connected neural network). The accuracy observed for the single-layer artificial neural networks is 99%, whereas the accuracy observed for the dual-layer artificial neural networks implemented is 99.51%. Similarly, in the case of Convolutional neural networks and Recurrent neural networks, the dual-layer approach performed better than the single-layer approach. The performance metrics for the dual-layer ANN are 99.51% accuracy, 99.5% recall, 99.5% precision, and 99.5% F1-score. The observed metrics for the dual layer CNN are 99.4%, 99.4%, 99%, and 99.2% for accuracy, recall, precision, and F1-score. While for the dual layer RNN, the observed metrics are 99.1% accuracy, 99.1% recall, 98.7% precision, and 98.9% F1-score.

As is shown in Fig. 18, the false negatives when classifying ham emails for the single-layer approach are 3, 4, and 5 for CNN, ANN,

Table 7
Performance of Dual Layer Architecture.

Classifier	Accur- acy	Ham Precision	Ham Recall	Ham F1-score	Spam Precision	Spam Recall	Spam F1-score	Phish Precision	Phish Recall	Phish F1-score
Dual Layer CNN	99.4	99.8	99.8	99.8	97.5	99	98.3	99.6	99.3	99.5
Dual Layer ANN	99.51	100	100	100	99	99	99	99.5	99.5	99.5
Dual Layer RNN	99.1	99.5	100	99.8	97.1	98.2	97.6	99.5	99	99.3

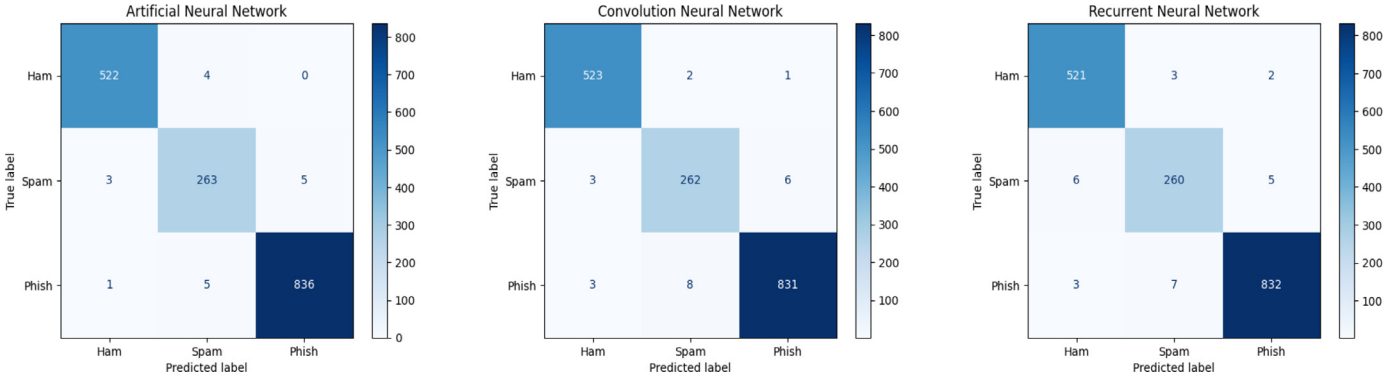

Fig. 18. Confusion Matrices for the Trained Deep Learning Models.

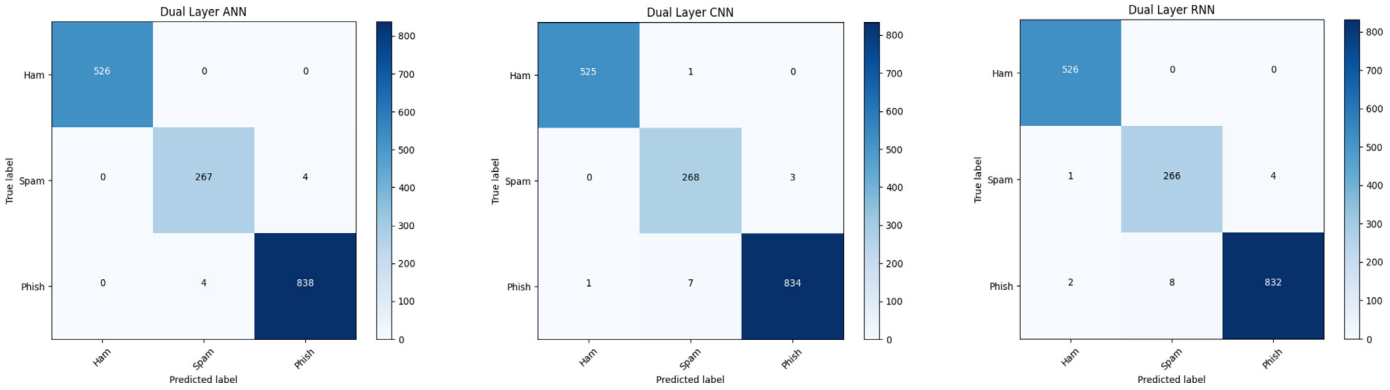

Fig. 19. Confusion Matrices for the Trained Dual-Layer Deep Learning Models.

and RNN respectively. The dual layer approach reduced the false negative values to 1, 0, and 0 for CNN, ANN, and RNN respectively as illustrated in Fig. 19. The false positives for the single-layer approach are 4, 6, and 9 for ANN, CNN, and RNN respectively. These are reduced to 0, 1, and 3 for ANN, CNN, and RNN respectively.

The false negatives for spam email class for the single-layer approach are 8, 9 and 11 for ANN, CNN, and RNN respectively; are reduced to 4, 3, and 5 by the dual-layer approach. The false positives for spam class observed for the single-layer approach are 9, 10, and 10 for ANN, CNN, and RNN respectively. These are reduced to 4, 8, and 8 by the dual-layer approach.

While classifying for the phish email class, the false negative values observed for the single-layer approach are 6, 10, and 11 are reduced in the dual-layer approach to 6, 10, and 8 for ANN, RNN, and CNN respectively. The false positives in the single-layer approach are 5, 7 and 7 are reduced in the dual-layer approach to 4, 4, and 3 for ANN, RNN, and CNN respectively.

For all the 1639 testing samples, the spam and ham classes are the minority classes and phish is the majority class. While the single-layer approach classifies the spam class considerably better than the traditional algorithms. The dual-layer approach is able to achieve even better results than the single-layer approach for the spam class while also not compromising the results for the ham and phish classes.

7.5. Comparison with existing works

The papers for comparison are selected on the basis of what is the classification class (spam or phish), what features are selected after feature engineering, which classifier algorithms are used for training the model, and whether the pre-processing of the data is similar or not. These papers are used as the benchmark for the results and the accuracy of the model acts as the means of comparison. Table 8 presents a detailed comparison of the dual-layer architecture with existing works, highlighting the observed results. Alhogail and Alsabih (2021) suggested a phishing email detection model that leverages deep learning algorithms, incorporating graph convolutional networks (GCN) and natural language processing on the email body text to enhance the accuracy of identifying phishing emails. Their proposed model produced an accuracy of 98.2%. Bountakas and Xenakis (2023) showcases an F1 Score of 99.42% using their novel technique of hybrid ensemble learning for phishing email detection. Bagui et al. (2019) considered the context of the emails as well by considering multiple n-grams and implemented a single layer CNN for classifying phish mails with an accuracy of 97.2%. Ravi et al. (2018) has classified emails with a header into phish and ham emails, and observed an accuracy of 96.5% with CNN and an accuracy of 96.2% with RNN. Rahman and Ullah (2020) incorporates an architecture consisting of Word embeddings, CNN, and Bi-LSTM network to classify spam emails. They ob-

Table 8
Comparison with existing studies.

Classification	Approach	Accuracy
Phish mails only	Bagui et al. (2019)	97.20%
	Ravi et al. (2018) - CNN	96.50%
	Ravi et al. (2018) - RNN	96.20%
	Alhogail and Alsabih (2021)	98.2%
	Bountakas and Xenakis (2023)	99.42% (F1-Score)
	Rahman and Ullah (2020)	98% - 99%
	Bansal and Sidhu (2021)	97.50%
Spam mails only	Kulkarni et al. (2020)	94.78%
	Sheneamer (2021)	96.52%
	Venugopal et al. (2022)	99.42%
Proposed Architecture	Dual layer CNN	99.40%
	Dual layer RNN	99.10%
	Dual layer ANN	99.51%

served an accuracy of between 98–99%, with 98.38% recall, 98.13% precision, and 98.25% F1-score. Bansal and Sidhu (2021) classified spam emails using the TF_IDF approach with the single-layer Artificial neural network model and observed an accuracy of 97.5%. Kulkarni et al. (2020) has featured engineered on the basis of email headers to classify spam emails and are able to obtain an accuracy of 94.78% whereas in the proposed architecture where featured engineers are performed on email body and headers are able to obtain a minimum accuracy of 99.1%. Sheneamer (2021) employed a CNN model with the GloVe model to classify spam emails and achieved an accuracy of 96.52%. Venugopal et al. (2022) proposed their own deep clustering algorithm to classify spam emails based on their severity observing an accuracy of 99.42%. The dual layer approach proposed scores an accuracy of 99.4%, recall of 99.4%, precision of 99%, and an F1-score of 99.2% with CNN. With RNN, the observed values are 99.1% accuracy, 99.1% recall, 98.7% precision, and 98.9% F1-score. The highest overall accuracy for the proposed architecture is observed in the dual-layer ANN model, the value being 99.51%. While most of the existing studies focus on classifying one of the malignant types of emails, viz. Spam and Phishing emails. The proposed architecture is able to classify both classes comparably better than the existing works, while also handling imbalance in the class distribution in the dataset.

8. Conclusion

In contemporary times, protecting oneself and organizations from phishing and spam emails has become a critical concern. This research employed both traditional machine learning algorithms and deep learning techniques to classify emails in an imbalanced dataset. The findings indicate that traditional machine learning algorithms underperformed, achieving a maximum accuracy of only 98.4%, while deep learning models achieved a minimum accuracy of 98.5%. However, deep learning techniques still underperformed for the minority class of spam emails due to high data imbalance. When using undersampling or oversampling techniques, the highest F1-score for the spam class is less than the observed lowest F1-scores for the ham and phish classes. Traditional methods of addressing data imbalance, such as undersampling and oversampling, are explored, but acceptable results have not not been obtained for the minority spam class.

The proposed methodology not only handles high imbalances in data but also improves the performance of the classification model. It appropriately manages data imbalance without leading to oversampling or undersampling of data and appropriately assigns weightage to minority classes. The Dual-Layer architecture achieved a minimum F1-score of 97.6%, which is greater than the maximum F1-scores of 97%, 95.1%, and 95.3% obtained for spam email classification using traditional machine learning and deep learning techniques, with or without resampling. The maximum F1-score obtained for classifying spam emails is 99% achieved by the Dual Layer ANN classifier.

Compared to earlier research studies, the proposed dual-layer architecture yielded improved performance metrics for all individual evaluation factors. This architecture is flexible and may be reproduced and employed for a variety of other classification applications where severe class imbalance is observed. Further research can be conducted in different areas, such as applying the proposed method to other complex text classification tasks and analyzing the effect of different feature extraction, selection, and preprocessing techniques on the proposed method.

9. Future scope

1. Expanding Feature Engineering: In this research, the primary emphasis is given to the analysis of the email's content and email's body. However, the authors have identified a promising avenue for future investigation, aiming to enhance email classification by thoroughly examining the potential role of the content within email attachments, specifically with regards to identifying and mitigating phishing tactics. Considering the significant role that email attachments can play in the classification task, it is crucial to explore their inclusion in the feature engineering process. Future investigations should aim to incorporate the contents of email attachments, thereby enhancing the overall classification performance.
2. Generalizability of the Proposed Architecture: The authors believe that the proposed architecture can be applied to various text classification problems, irrespective of data imbalance issues. To validate this hypothesis, it is necessary to train and test the proposed architecture on complex text datasets. By evaluating its performance on diverse datasets, the generalizability and effectiveness of the architecture can be assessed, ensuring its suitability for real-world applications.
3. Mathematical Models for Multiclass Classification: Employing the proposed architecture for multiclass classification problems, especially those with more than three classes, requires addressing the challenge of class distribution within the architecture's layers. To overcome this, logical and mathematical models need to be developed. These models should be capable of handling the complexities associated with multiclass classification and the unclear distribution of classes. Incorporating such models would ensure accurate classification results in scenarios with numerous classes and intricate class distributions.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Jay Doshi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Supervision, Project administration, Writing – original draft, Writing – review & editing. **Kunal Parmar:** Conceptualization, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Raj Sanghavi:** Conceptualization, Validation, Formal analysis, Writing – original draft, Writing – review & editing. **Narendra Shekokar:** Conceptualization, Writing – review & editing, Supervision.

Data availability

Data will be made available on request.

References

Abbasi, D. F., 2022. Evasive urls in spam. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/evasive-urls-in-spam/.

Abdulraheem, R., Odeh, A., Al Fayoumi, M., Keshta, I., 2022. Efficient email phishing detection using machine learning. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0354–0358. doi:10.1109/CCWC54503.2022.9720818.

Alhogail, A., Alsabih, A., 2021. Applying machine learning and natural language processing to detect phishing email. Computers & Security 110, 102414.

Aswathisasidharan, G., 2022. Support Vector Machine Algorithm. https://www.geeksforgeeks.org/support-vector-machine-algorithm/.

Bagui, S., Nandi, D., Bagui, S., White, R.J., 2019. Classifying phishing email using machine learning and deep learning. In: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–2. doi:10.1109/CyberSecPODS.2019.8885143.

Bansal, C., Sidhu, B., 2021. Machine learning based hybrid approach for email spam detection. In: 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1–4. doi:10.1109/ICRITO51393.2021.9596149.

Bountakas, P., Xenakis, C., 2023. Helphed: hybrid ensemble learning phishing email detection. Journal of Network and Computer Applications 210, 103545.

Cerruto, F., Cirillo, S., Desiato, D., Gambardella, S.M., Polese, G., 2022. Social network data analysis to highlight privacy threats in sharing data. J Big Data 9 (1), 19.

Choubey, V., Text classification using CNN. https://www.medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9.

Coding Ninjas, Code studio. https://www.codingninjas.com/codestudio/library/bernoulli-naive-bayes.

Craw, S., Manhattan distance. https://www.link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_506?error=cookies_not_supported&code=40fa5379-504c-4bb7-804b-c58a3aae7ecb.

Cveticanin, N., 2023. What's on the other side of your inbox - 20 spam statistics for 2023. https://dataprot.net/statistics/spam-statistics/.

cyberattacks, 2020. https://www2.deloitte.com/my/en/pages/risk/articles/91-percent-of-all-cyber-attacks-begin-with-a-phishing-email-to-an-unexpected-victim.html.

Dada, E.G., Bassi, J.S., Chiroma, H., Adetunmbi, A.O., Ajibuwa, O.E., et al., 2019. Machine learning for email spam filtering: review, approaches and open research problems. Heliyon 5 (6), e01802.

Dedhia, S., Trivedi, S., Salvi, S., Jani, J., D'mello, L., 2022. A novel dual model approach for categorization of unbalanced skin lesion image classes. In: Smys, S., Tavares, J.M.R.S., Balas, V.E. (Eds.), Computational Vision and Bio-Inspired Computing. Springer Singapore, Singapore, pp. 635–649.

Deshpande, K., Girkar, J., Mangrulkar, R., 2023. Security enhancement and analysis of images using a novel sudoku-based encryption algorithm. Journal of Information and Telecommunication 0 (0), 1–34. doi:10.1080/24751839.2023.2183802.

Developers, T. R., KNNClassifier - River. https://www.riverml.xyz/dev/api/neighbors/KNNClassifier/.

Donges, N., Random forest classifier: a complete guide to how it works in machine learninghttps://www.builtin.com/data-science/random-forest-algorithm.

Egozi, G., Verma, R., 2018. Phishing email detection using robust NLP techniques. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 7–12. doi:10.1109/ICDMW.2018.00009.

Emigh, A., 2007. Phishing attacks: information flow and chokepoints. Phishing and countermeasures 31–64.

Fette, I., Sadeh, N., Tomasic, A., 2007. Learning to detect phishing emails. In: Proceedings of the 16th International Conference on World Wide Web, pp. 649–656.

Foundation, A. S., 2006. Spam assassin homepage. https://spamassassin.apache.org/old/publiccorpus/.

Foundation, P. S., 2022. Manipulate mailboxes in various formats. https://docs.python.org/3/library/mailbox.html.

Gangavarapu, T., Jaidhar, C., Chanduka, B., 2020. Applicability of machine learning in spam and phishing email filtering: review and approaches. Artif Intell Rev 53, 5019–5081.

Gangavarapu, T., Jaidhar, C.D., Chanduka, B., 2020. Applicability of machine learning in spam and phishing email filtering: review and approaches. Artif Intell Rev 53 (7), 50195081. doi:10.1007/s10462-020-09814-9.

George Lawton Ed Burns, L. R., Logistic regressionhttps://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression.

Ghosh, A., 2022. A deep dive into similar domain name phishing schemes. https://www.redpoints.com/blog/similar-domain-name-phishing/.

James, N., 2022. Phishing attack statistics 2023: The ultimate insight. https://www.getastra.com/blog/security-audit/phishing-attack-statistics/.

Jason Brownlee, Extreme gradient boosting (xgboost) ensemble in pythonExtreme Gradient Boosting (XGBoost) Ensemble in Python.

Karabiber, F., Cosine Similarity. https://www.learndatasci.com/glossary/cosine-similarity/.

Kharwal, A., Multinomial naive bayes in machine learning. https://thecleverprogrammer.com/2021/08/06/multinomial-naive-bayes-in-machine-learning/.

Kulkarni, P., Saini, J.R., Acharya, H., 2020. Effect of header-based features on accuracy of classifiers for spam email classification. International Journal of Advanced Computer Science and Applications 11 (3). doi:10.14569/IJACSA.2020.0110350.

Kumar Birthriya, S., Jain, A.K., 2022. A comprehensive survey of phishing email detection and protection techniques. Information Security Journal: A Global Perspective 31 (4), 411–440.

Li, Q., Cheng, M., Wang, J., Sun, B., 2022. Lstm based phishing detection for big email data. IEEE Trans. Big Data 8 (1), 278–288. doi:10.1109/TBDATA.2020.2978915.

Majumder, P., Gaussian naive bayes. https://www.iq.opengenus.org/gaussian-naive-bayes/.

Mantuano, F., 2022. mail-parser. https://www.pypi.org/project/mail-parser/.

MI—extra tree classifier for feature selection, https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/.

Moamen Elabd, What is bagging classifier?https://www.medium.com/@arch.mo2men/what-is-bagging-classifier-45df6ce9e2a1.

Nagesh Singh Chauhan, KDnuggets, Naïve bayes algorithm: Everything you need to know. https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html.

Navlani, A., Decision tree classification in python tutorial. https://www.datacamp.com/tutorial/decision-tree-classification-python.

Nayak, R., Amirali Jiwani, S., Rajitha, B., 2021. Spam email detection using machine learning algorithm. Mater. Today:. Proc. doi:10.1016/j.matpr.2021.03.147. https://www.sciencedirect.com/science/article/pii/S2214785321021660

Nazario, J., 2006. Phishing corpus homepage. https://monkey.org/~jose/phishing/.

Rahman, S.E., Ullah, S., 2020. Email spam detection using bidirectional long short term memory with convolutional neural network. In: 2020 IEEE Region 10 Symposium (TENSYMP), pp. 1307–1311. doi:10.1109/TENSYMP50017.2020.9230769.

Ravi, V., Hb, B.G., Poornachandran, P., Kumar, M., Kp, S., 2018. Deepanti-phishnet: Applying deep neural networks for phishing email detection cen-aisecurity@iwspa-2018.

Richardson, L., 2022. beautifulsoup. https://pypi.org/project/beautifulsoup4/.

Ruan, G., Tan, Y., 2009. A three-layer back-propagation neural network for spam detection using artificial immune concentration. Soft comput 14 (2), 139–150. doi:10.1007/s00500-009-0440-2.

Saini, A., Gradient boosting algorithm: a complete guide for beginners https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/.

Salloum, S., Gaber, T., Vadera, S., Shaalan, K., 2021. Phishing email detection using natural language processing techniques: a literature survey. Procedia Comput Sci 189, 19–28.

Samarthrao, K.V., Rohokale, V.M., 2022. A hybrid meta-heuristic-based multi-objective feature selection with adaptive capsule network for automated email spam detection. International Journal of Intelligent Robotics and Applications 6 (3), 497–521. doi:10.1007/s41315-021-00217-9.

Sarkar, P., Boosting and adaboost in machine learning. https://www.knowledgehut.com/blog/data-science/boosting-and-adaboost-in-machine-learning.

Saxena, S., Introduction to long short term memory (lstm). https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

Selig, J., 2022. What is machine learning? a definition. https://www.expert.ai/blog/machine-learning-definition/.

Sheneamer, A., 2021. Comparison of deep and traditional learning methods for email spam filtering. International Journal of Advanced Computer Science and Applications 12 (1).

S. of Digital Formats: Planning for Library of Congress Collections, 2022. Mbox email format. https://www.loc.gov/preservation/digital/formats/fdd/fdd000383.shtml.

Steven Bird, E.K., Loper, E., 2009. Natural Language Processing with Python. O'Reilly Media Inc.

Venugopal, I., Bhaskari, L., Seetaramanath, M., 2022. Detection of severity-based email spam messages using adaptive threshold driven clustering. International Journal of Advanced Computer Science and Applications 13. doi:10.14569/IJACSA.2022.0131040.

W. contributors, 2023. Euclidean distance. https://en.wikipedia.org/wiki/Euclidean_distance.

What are neural networks?—IBM. Online available: https://www.ibm.com/topics/neural-networks.

What are recurrent neural networks?—IBM, https://www.ibm.com/topics/recurrent-neural-networks.

What is deep learning? — how it works, techniques & applications, https://www.mathworks.com/discovery/deep-learning.html.

Wikipedia contributors, Minkowski distance. https://en.wikipedia.org/wiki/Minkowski_distance.

Yahya, A., Ahmad, R., Mohd Yacob, Y., Yaakob, N., Ku Azir, K.N.F., 2016. Multi stage phishing email classification 83, 206–214.

Jay Doshi is currently pursuing his bachelors in technology in Computer Engineering from Dwarkadas Jivanlal Sanghvi College of Engineering. His research interests include Machine Learning, Deep Learning, Natural Language Processing and Computer Vision. He has previously worked on research projects in these domains under the guidance of his professors.

Kunal Parmar is currently pursuing his bachelors in technology in Computer Engineering from Dwarkadas Jivanlal Sanghvi College of Engineering. His research interests include Machine Learning, Deep Learning, Natural Language Processing and Game Development. He has previously worked on research projects in these domains under the guidance of his professors.

Raj Sanghavi is currently pursuing his bachelors in technology in Computer Engineering from Dwarkadas Jivanlal Sanghvi College of Engineering. His research interests include Machine Learning, Deep Learning, Natural Language Processing and Data Science. He has previously worked on research projects in these domains under the guidance of his professors.

Narendra Shekokar has received his Ph.D. in Engineering (Network Security) from NMIMS University, Mumbai and he is working as a Professor, Dept. of Computer Engineering at SVKM's Dwarkadas J. Sanghvi College of Engineering, Mumbai (Autonomous college affiliated to Universy of Mumbai). He was a member of Board of Studies at University of Mumbai for more than 5 years, Currently working as BoS Member at R.C.Patel Institute of Technology, Shirpur and also been a member of various committees at University of Mumbai. His total teaching experience is 23 years. Dr. Narendra Shekokar is Ph.D guide for 8 research fellows and more than 26 students at Post Graduation level. He has presented more than 60 papers at International & National conferences and has also published more than 20 research papers in renowned journals. He is Editor of two renewed Book in ML & DL and Cyber Security domain published by Taylor and Francis, CRC Press, USA. He has received the Minor Research Grant twice from University of Mumbai for his research projects. He has delivered expert talk and chaired a session at numerous events and International conferences.