A Comparison of Natural Language Processing and Machine Learning Methods for Phishing Email Detection

Panagiotis Bountakas*
bountakas@unipi.gr
Department of Digital Systems, University of Piraeus
Athens, Piraeus, Greece

Konstantinos Koutroumpouchos
konstantinos.koutroumpouxos@ssl-unipi.gr
Department of Digital Systems, University of Piraeus
Athens, Piraeus, Greece

Christos Xenakis
xenakis@unipi.gr
Department of Digital Systems, University of Piraeus
Athens, Piraeus, Greece

ABSTRACT

Phishing is the most-used malicious attempt in which attackers, commonly via emails, impersonate trusted persons or entities to obtain private information from a victim. Even though phishing email attacks are a known cybercriminal strategy for decades, their usage has been expanded over last couple of years due to the COVID-19 pandemic, where attackers exploit people's consternation to lure victims. Therefore, further research is needed in the phishing email detection field. Recent phishing email detection solutions that extract representational text-based features from the email's body have proved to be an appropriate strategy to tackle these threats. This paper proposes a comparison approach for the combined usage of Natural Language Processing (TF-IDF, Word2Vec, and BERT) and Machine Learning (Random Forest, Decision Tree, Logistic Regression, Gradient Boosting Trees, and Naive Bayes) methods for phishing email detection. The evaluation was performed on two datasets, one balanced and one imbalanced, both of which were comprised of emails from the well-known Enron corpus and the most recent emails from the Nazario phishing corpus. The best combination in the balanced dataset proved to be the Word2Vec with the Random Forest algorithm, while in the imbalanced dataset the Word2Vec with the Logistic Regression algorithm.

KEYWORDS

Natural Language Processing, Machine Learning, Phishing, Email, TF-IDF, Word2Vec, BERT, Random Forest, Logistic Regression

ACM Reference Format:
Panagiotis Bountakas, Konstantinos Koutroumpouchos, and Christos Xenakis. 2021. A Comparison of Natural Language Processing and Machine Learning Methods for Phishing Email Detection. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3465481.3469205

1 INTRODUCTION

Email is the most usual and versatile form of written enterprise communications, especially nowadays, where the COVID-19 pandemic drove to a shift from office to home-based work. Meanwhile, cyber-criminals saw the pandemic crisis as an opportunity to exploit email communications to conduct phishing attacks, where they appear as a reputable entity and attempt to steal private information such as login credentials, install a malware, or lure the victim to visit infected websites. Within a year (mid 2019 to mid 2020), email threats increased by more than 64% [29] [6]. Nowadays, an outburst of phishing email attacks related to COVID-19 has been observed [4], which in a period of one month (February 2020 and March 2020) has seen an increased rate of 667% [2]. Moreover, the development of decentralized data access and cloud services has caused a paradigm shift in communication and file sharing, which in this world of ubiquitous communication cultivates a breeding ground for phishing email attacks. Phishing email attacks are accountable for 90% of data breaches, causing an average financial loss of $3.86 million[1], which means that phishing emails are more responsible for the disclosure of private and confidential information than other attack types. Moreover, 85% of organizations have been victims of phishing[2] at least one time.

As one can understand, the detection of phishing emails is crucial to combat these attacks. Phishing email detection is an active research area for more than a decade. However, with the expansion of phishing emails, the effectiveness of earlier detection approaches, which relied mostly on filtering techniques, like heuristic and blacklisting, is poor [17]. As a next step, researchers attempted to exploit Machine Learning (ML) methods that focus on the emails' contents, such as the email headers, domain, hyperlinks, and word lists to detect phishing emails; nevertheless, the email's contents can be forged leading to false conclusions [9]. Currently, the evolution of Natural Language Processing (NLP) and, more specifically, word embedding techniques have contributed to the development of robust phishing email detection approaches that emphasize the morphology and semantics of the emails' text [10]. Recent works treat the phishing email detection problem as a text classification task, namely, they take into account only the emails' text and apply NLP methods to handle the textual features [7]. The Term Frequency - Inverse Document Frequency (TF-IDF) [35] is a well-known method

[1]https://retruster.com/blog/2019-phishing-and-email-fraud-statistics.html
[2]https://www.keepnetlabs.com/phishing-statistics-you-need-to-know-to-protect-your-organization/

to measure the significance of a word in a document. In the last couple of years it has been the most used NLP technique in the phishing email detection field, where it was deployed as a weighting factor of the words that appear in the email corpus [18] [17] [21] [40]. Word2Vec [28] is a popular method for the creation of word embeddings, namely vector representations of a word, which has seen a few applications in the phishing email detection for the identification of word associations between different emails of an email corpus [33] [13]. Furthermore, recent advances in ML, such as the emergence of a new language model known as Bidirectional Encoder Representations from Transformers (BERT) [11], have revealed promising results in an wide range of classification problems, like malware detection [30] [34] [22] and email user classification [37]. BERT's key innovation is that it can learn the context of a particular word considering both the previous and next words. While previous language models contemplate only one direction to generate unidirectional representations, BERT takes into account both right-to-left and left-to-right directions to produce bidirectional representations. However, to the best of our knowledge, it has not been applied yet to the extraction of textual features from phishing emails.

The motivation of this work stems from the tremendous growth of phishing attacks over the last couple of years, as well as from the inability of the current detection solutions to curb this threat. Recent surveys point out that a major drawback in the phishing email detection field is that previous researches did not consider the advancement of phishing email attacks (i.e., they are using email samples that came from old sources) [10] [7]. A second motivation point is the fact that the efficacy of an ML phishing detection method that focuses on the email's body text using NLP heavily relies on the cooperation of the NLP method with the ML algorithm. However, previous works have not considered deploying several combinations of NLP techniques with ML algorithms (hereafter, the combination of an NLP method with an ML algorithm is referred to as NLP/ML) to identify the most powerful pair (i.e., the NLP/ML pair that achieves the best performance). Overall, we argue that new efficient detection technologies are needed to limit the ongoing threat of phishing email attacks.

In this paper, a comparison approach for the combined usage of NLP and ML methods is introduced that is based on the detection of phishing emails focusing only on the textual information of the emails' body field. More specifically, the proposed comparison approach contains five tasks, the *Email Parsing*, the *Pre-processing*, the *Textual Feature Extraction*, the *Feature Selection*, and the *Classification*. The *Email Parsing* and the *Pre-processing* tasks extract and clean the emails' body texts respectively. The *Textual Feature Extraction* task constructs the text-based feature set using NLP methods, while the *Feature Selection* identifies a feature subset with the most informative features. Finally, the *Classification* task deploys ML algorithms to classify the emails between benign and phishing. A novel aspect of the proposed approach is that the *Textual Feature Extraction* task employs three different NLP methods, named TF-IDF, Word2Vec, and BERT, to extract the textual features from the emails, which are later combined with five ML algorithms (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Trees, and Naive Bayes) to identify the NLP/ML pair that will attain the best results. Two experiments were performed to evaluate the proposed comparison approach. The first experiment was performed on a balanced dataset, where the number of phishing emails is the same as the number of benign emails. The second experiment was performed on an imbalanced dataset, where the ratio between phishing and benign emails is 1:10. Another novelty of this work is that the evolution of phishing emails has been considered by deploying only new phishing emails (2015-2020) in the experiments. The experimental results, which were based on numerous evaluation metrics (e.g., F1-score, accuracy, precision, recall, etc.), showed that the best NLP/ML combination for balanced data is the Word2Vec/Random Forest, and for imbalanced data it is the Word2Vec/Logistic Regression. In summary, our contributions lie in the following aspects:

- We propose a comparison approach for phishing email detection that combines different NLP methods with various ML algorithms.
- We comprehensively evaluate the proposed approach to identify the best NLP/ML combination.
- We take into account the evolution of phishing emails, by employing only new emails (2015-2020).
- We deploy the BERT language model for the extraction of textual features from phishing emails.
- We provide the source code of our approach as open-source to facilitate the security community in detecting phishing email attacks.

The remaining of this paper is structured as follows. Section 2 elaborates on recent previous related works that focus on the emails' text highlighting their advantages and drawbacks. Section 3 analyzes the proposed comparison approach. Section 4 evaluates the numerous NLP/ML combinations and discusses the results of the experiments. Finally, section 5 summarizes the critical points and outlines the conclusions drawn.

2 RELATED WORK

The majority of phishing email detection approaches in the literature process the email's text to identify text-based features and deploy ML and Neural Network (NN) methods to distinguish phishing from benign emails. The application of both NLP and ML/NN for the extraction of informative features from the emails' text and the classification of emails respectively has played an important role in the phishing email detection [24].

Previous works in this area have employed contextual [43], semantic [42], and syntactic [31] features from the emails' text. A recent work that has the same ground as ours is proposed in [40]. The authors focused on the emails' text to distinguish phishing from benign emails. To do so they utilized two techniques, namely TF-IDF and Doc2Vec (a word embedding technique that is based on Word2Vec), to prepare the text-based features and several ML classifiers, such as Decision Tree, Naive Bayes, AdaBoost, Logistic Regression, K-nearest neighbor, Support Vector Machines, and Random Forest, to predict whether an email is phishing. Two imbalanced datasets (4082 benign & 501 phishing emails and 5088 benign & 612 phishing emails) were deployed to measure the effectiveness of the classifiers, using the accuracy as a metric, and the results indicate that the ML classifiers performed better with the

Doc2Vec method. The drawback of this work is that the authors did not use a metric that is suitable for imbalanced data, such as F1-score. Instead, they measured the efficacy of their approach using accuracy, which is biased towards the majority class (namely, the class that contains the most samples, which in their case is the benign emails). The work in this paper improves the approach that presented in [40] (a) by performing feature selection before the classification process to identify the features that contribute to better classification performance, (b) by deploying Word2Vec and BERT techniques, which are new and more well-known in text classification tasks than Doc2Vec, (c) by utilizing F1-score metric that depicts the model's performance when tested in imbalanced data more precisely, and d) by considering the evolution of phishing emails using only new phishing emails.

In [13], the authors presented a phishing detection framework that is based on Recurrent Convolutional NNs, named THEMIS. The Word2Vec method was utilized to obtain the vector sequences from the character-level and word-level of both the emails' header and body fields. THEMIS accomplished 99.848% detection accuracy with a 0.043% FPR. Towards the same direction, the method proposed in [27] combined Convolutional NNs and Keras Word Embedding to detect phishing emails focusing on the text. The authors compared two datasets, one with email headers and one without. The results showed that the model achieves higher detection accuracy (96.8%) when the email headers are not taken into account. A drawback of both [13] and [27] is that the authors did not consider the evolution of phishing emails, which is a significant limitation as the phishing emails have evolved over the years. Furthermore, another limitation of [27] is that the authors deployed an imbalanced dataset (4082 benign and 501 phishing emails) in their experiments, and measure the classifiers' performance only on the classification accuracy without utilizing other metrics (e.g., F1-score, AUC) that depict better the performance on imbalanced data.

The authors of [19] proposed a methodology, named SAFE-PC, for the detection of new phishing campaigns. Their research exploited NLP techniques to extract five features from a real-world dataset of a tier-1 research university: i) commonly known phishing words and their synonyms, ii) words associated with the tier-1 research institution, iii) commonly occurring phishing words from the deployed corpus and their synonyms, iv) proper noun organization names and their types, and v) structural features in the email. A RUSBoost classifier (it combines data sampling and boosting) with three weak learners (i.e., Naive Bayes, Decision Trees, and Perceptron classifiers) was applied to the phishing and benign emails. SAFE-PC detected 71% of the phishing emails, which had been erroneously classified by the Sophos antivirus whilst having an FPR of 15%. Furthermore, SAFE-PC outperformed SpamAssassin, which was non-competitive as it had a detection rate of less than 10%. In [12], the authors considered also features like word counts, punctuation, and stopwords. They extracted in total 26 features, which were utilized to compare different ML models. The best results were accomplished by linear kernel SVM (83% TPR and 96% TNR). The limitations of the aforementioned works are mainly on the deployed features, which mostly relied on variations of words, stop words, punctuation counts, and ratios between them that can be easily evaded by adversaries.

Gualberto et al. [17] aimed at extracting distinctive features from the text of phishing emails. The authors dealt with several problems, such as the text context portion that was contained in the extracted feature set, the sparsity, and "the curse of dimensionality". With 10 features their approach reached 99.95% accuracy with the XGBoost algorithm. An extension of their previous work [17] was presented in [18], where the authors developed a multi-stage approach to discover the purpose of the email (phishing or benign). The text-based features were extracted via the TF-IDF technique and two methods were employed to process the features further. In the first method, the Chi-Square method as well as the Mutual Information were exploited to enhance the dimensionality reduction, whilst in the next method the Principal Component Analysis along with Latent Semantic Analysis techniques were deployed. The second method accomplished the best performance with the XGBoost algorithm (100% accuracy and 100% F1-score) using the SpamAssassin[3] and Nazario datasets [5]. Both [17] and [18] attain comparable results on the same dataset that contains 4150 benign and 2279 phishing emails; however, in [17] the evaluation was performed using cross-validation, while in [18] the dataset was separated into 70% for training and 30% for testing (namely, 1261 benign and 678 phishing emails). Moreover, the evolution of phishing emails has not been considered in any of these researches as the deployed phishing emails are outdated.

In [21], the authors employed TF-IDF and dimensionality reduction techniques for the classification of emails between phishing and benign. The experiments were performed on the IWSPA dataset [45], where promising results were achieved (99.9% accuracy); however, the authors mentioned that their ML classifiers overfit the testing data due to the imbalanced dataset. A combination of the TF-IDF technique with domain-level features was proposed in [39]. The authors utilized both TF-IDF textual features and 40 domain-level features as input to the ML classifiers. For the experiments, the IWSPA dataset was used, and the best performance, in terms of F1-score (0.98), was attained by the Logistic Regression classifier. The evaluation data of both [21] and [39] are outdated as IWSPA dataset mostly contains old phishing emails.

Another recent work is [20], where the authors proposed a detection system that utilizes Recurrent Neural Networks (RNNs) for the detection of phishing emails taking into account only the textual structure of the email. To perform the classification a word tokenization took place. For the evaluation, the authors compared the RNN classifier on two datasets (in which only the benign emails have been changed) with the "textAnalysis" classifier proposed in [44] and the Dynamic Markov Chain (DMC) model proposed in [8]. The RNN classifier outperforms the "textAnalysis" classifier; however, it did not manage to achieve better results than the DMC model. The drawback of this work is mainly on the deployed phishing emails, where the authors did not mention their creation date, hence it is uncertain whether in this work the evolution of phishing emails was considered. Also, the performance of the model that was proposed in [20] was worse than the performance of DMC.

In general the major limitations that were identified in the literature are: limited evaluation metrics or metrics that are inappropriate to measure the classifiers performance (e.g., in case of imbalanced

[3]https://spamassassin.apache.org/

data) [40] [27] [39] [21], the phishing emails that have been deployed for evaluation purposes are old [27] [13] [20] [17] [18], and in some works the considered textual features are not robust [19] [12]. Moreover, although NLP and ML have been utilized in phishing email detection for several years, the literature misses proofs regarding which NLP method works better for phishing email detection. The previous researches in the field did not elaborate on various NLP methods to extract text-based features and support their arguments.

In this paper, we focused on the emails' body text, which has been proved that it can achieve state-of-the-art results in the detection of phishing attacks. However, we considered that the classifiers' efficiency greatly depends on the way that the text is processed to extract the text-based features that will be fed in the ML algorithms. Therefore, the main hypothesis of this paper is to identify the most appropriate NLP/ML combination for phishing email detection by comparing three state-of-the-art methods (i.e., TF-IDF, Word2Vec, and BERT) and five well-known ML algorithms (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Trees, and Naive Bayes). To the best of our knowledge, we are the first that deployed BERT for the extraction of textual features from phishing emails. The comparison was performed in the same environmental setup, using the same datasets. Furthermore, the limitations of the literature, which were discussed in the previous paragraph, have been addressed in this research. First, the evolution of phishing emails was considered by employing in the experiments the newest phishing emails (2015-2020) of the Nazario dataset (see Section 4.1). Second, two experiments were performed, one with a balanced dataset and one with an imbalanced dataset to evaluate the proposed approach on different benign/phishing email ratios, as well as to highlight the fact that the accuracy is an inappropriate metric for imbalanced data. The experimental results on the imbalanced dataset were based on the outcome of the F1-score metric that measures the algorithm's performance without being biased (like accuracy).

3 PROPOSED APPROACH

In this section, the details of the proposed comparison approach are presented. In subsection 3.1, an overview of the approach is depicted. Subsections 3.2 and 3.3 describe the parsing and the preprocessing of the emails respectively. In subsection 3.4, the textual feature extraction process is described, where three different NLP methods have been employed. Finally, subsection 3.6 presents the final task, which is the classification process.

3.1 Overview

The main objective of this work is to find the best NLP/ML pair for the textual feature extraction and the classification of phishing emails by comparing state-of-the-art techniques. In Figure 1, the architecture of the proposed approach is presented. The *email parsing* task associates the extracted body text of emails with their respective class and places them in a matrix. The *pre-processing* task is responsible for cleaning the emails' text and converting them to a uniform format. To this end, the texts are converted into lowercase, and the special characters, stopwords, and punctuation marks are removed. Then, the lemmatization process takes place, where the various inflected forms of the words are grouped to be considered as a single item. In the *textual feature extraction* task the TF-IDF, Word2Vec, and BERT NLP methods are applied to the clean text to extract the textual features, which will be in a form that is understandable by the ML classifiers. The *feature selection* task, using the Chi-square method, discriminates the features to maintain only the most informative ones to reduce the training time and increase the classification accuracy. The selected features are then submitted to the *classification* task, where several well-known ML algorithms have been utilized to classify the emails between benign and phishing.

3.2 Email Parsing

The *email parsing* is the process through which the emails' text bodies are separated from the other email fields and stored together. Before analyzing how the parsing of emails is performed, it is important to mention the grounds for focusing only on the emails' bodies, without considering other information such as the email's headers. We argue that the email's body text includes a wealth of information that can be utilized for the detection of phishing emails, as it is usually random since it is controlled by people. However, the body of phishing emails reveals similar traits, in deep semantics, which vary from benign emails. Moreover, by dealing only with the email's body, our approach avoids the processing of personal information, such as the sender and receiver headers of the email.

Both benign and phishing emails undergo a parsing phase, where their corresponding body texts are extracted. During the parsing phase, the emails' texts are organized in a matrix along with their respective class (phishing or benign) to facilitate the pre-processing task in the processing of the emails. This is achieved by putting each email's text to a matrix with n rows, where n is the number of email samples and two columns, where the first column refers to the emails' body text (e) and the second to the email's class (l).

$$M = \begin{bmatrix} e_{1\times 2} & l_{1\times 2} \\ e_{2\times 1} & l_{2\times 2} \\ e_{3x1} & l_{3\times 2} \\ \vdots & \vdots \\ e_{n\times 1} & l_{n\times 2} \end{bmatrix} \tag{1}$$

3.3 Pre-processing

The *pre-processing* task is a fundamental task of the proposed approach, as it removes irrelevant and redundant information from the emails' body texts, as well as it converts the texts into a uniform format. The purpose of this task is to facilitate the *textual feature extraction* task identifying the most informative words of the emails' corpus. This is achieved by employing the first column of the matrix M (e) as input, where the emails' body texts are included. More specifically, all the words are transformed into lowercase and all the special characters, punctuation marks, stopwords, and HTML elements (in case they appear in an email) are removed. The hyperlinks that may exist in the email's body text are replaced with a fixed string (if it was replaced with a word and this word appears in the text then the results will be forged). Afterwards, a tokenization process takes place, in which the words of the text are separated by a delimiter and converted from a string to a list of words. In our case,

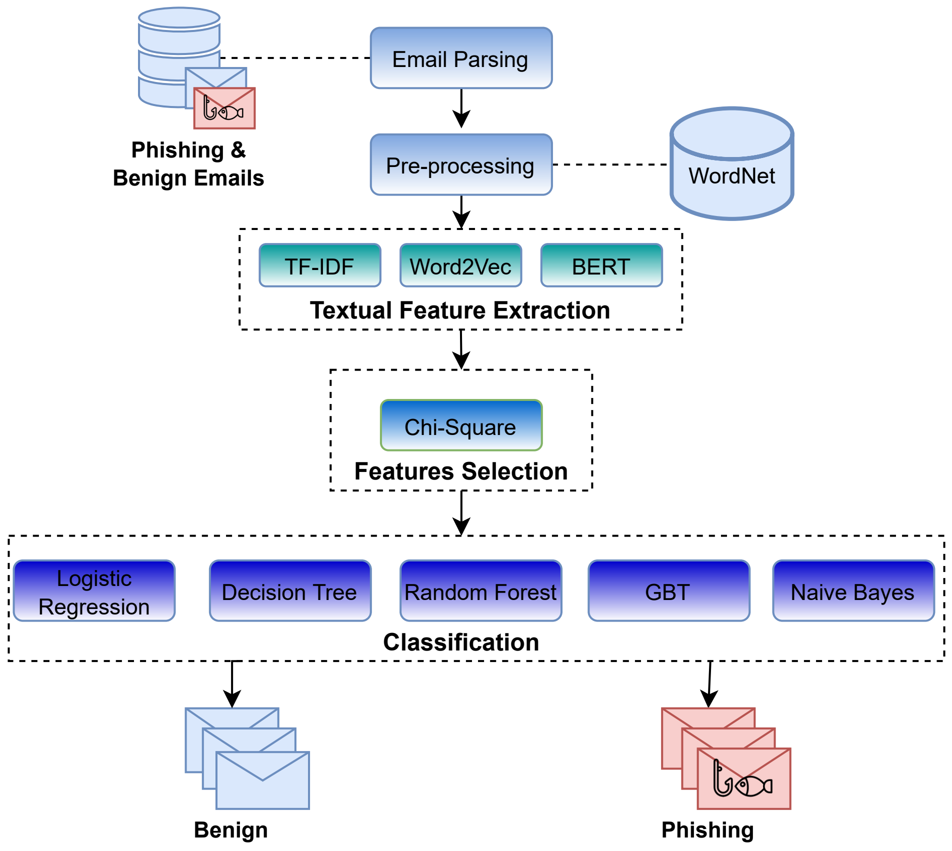

Figure 1: Proposed Approach Architecture

we considered each token/word as a sequence of characters (a-z), so this was our delimiter. Later, the pre-processing phase continues with the lemmatization phase that is the core of the *pre-processing* task.

Lemmatization is the process that reduces the inflectional forms of a word to keep its root form; thus, the resulting word set, which is going to be processed by the NLP algorithms, is smaller because all the inflections of a word are converted to one. An alternative method to convert a word to its original form is stemming. Even though both methods have the same goal, we chose the lemmatization, as it is based on a morphological analysis of the words, hence it provides a meaningful form of the words, while stemming cuts the end or/and the beginning of a word based on prefixes and suffixes of inflected words. The lemmatization process needs a vocabulary that contributes to the identification of the words' original form (known as lemma). For this purpose, the WordNet database [14] was employed as the vocabulary. WordNet is a popular lexical repository that includes semantic relations between words in many (more than 200) languages. In WordNet, the words are linked into semantic relations and grouped into conceptual synonym sets (synsets) that express the meaning of a notion. The connection among these sets offers a grid of substantially related works and notions that enhance the outcome of NLP. This approach deploys the WordNet to reduce the semantics of the emails' texts. Namely, via the lemmatization process, the diversity of the text has been reduced by replacing a group of words that are synonyms and have the same lemma. For instance, the lemma of the word *better* is the word *good*. Moreover, the Part-of-Speech (POS) tagging process has been used. The POS indicates the grammatical category of each word (i.e, noun, verb, adjective, and adverb) and facilitates the lemmatization process to further reduce the word instances.

3.4 Textual Feature Extraction

We treated the phishing email detection problem as a text classification task, which is an essential part of NLP. Therefore, the *textual feature extraction* task applies NLP methods for converting the output of the pre-processing task of the email's body text into text-based features to be processed by the ML algorithms. In this way, our approach could be also applied to data that do not contain private and sensitive information, i.e., email addresses, domains, etc. To accomplish this task, three well-known NLP methods, named TF-IDF [35], Word2Vec [28], and BERT [11], have been employed. The rationale for selecting three methods was to test their performance in the same environment and conclude which method is better when combined with ML algorithms. The result of this task is three diverse feature sets generated by each method. In the following paragraphs, we elaborate more on TF-IDF, Word2Vec, and BERT to facilitate the reader to comprehend the presented notions.

3.4.1 Method 1: TF-IDF. The TF-IDF [35] is a method often deployed in information retrieval, text mining, and in recent years it has been applied also in phishing email detection [18] [17]. We applied TF-IDF in the emails' clean text to extract the text-based features. TF-IDF outputs a weight that indicates the importance of a word in a collection of email texts. The weight is computed using two terms:

(1) **Term frequency (TF)**: It depicts the number of times a given word (term) appears in the text. Since each text is of a different size, TF is normalized to avoid a bias toward bigger texts. The TF of a term t in an email e is computed as:

$$TF(t,e) = \frac{f_{t,e}}{\sum\limits_{t' \epsilon e} f_{t',e}}, \tag{2}$$

where $f_{t,e}$ is the frequency of the term t in the email e.

(2) **Inverse document frequency (IDF)**: It depicts the general importance of a term in a collection of emails. It is calculated as a logarithm of the ratio of the total number of emails to the number of emails in which the term appears. If t is a term and C is a collection of emails, the IDF is computed as:

$$IDF(t,C) = \log \frac{n}{|e \epsilon C : t \epsilon e|}, \tag{3}$$

where n is the total number of emails in C.

The TF-IDF weight is the product of TF and IDF, and is computed as:

$$TF\text{–}IDF(t,e,C) = TF(t,e) * IDF(t,C). \tag{4}$$

Based on the TF-IDF weight (equation 4) the importance of a word in the email corpus is calculated, which contributes in the construction of the text-based feature set.

3.4.2 Method 2: Word2Vec. The Word2vec [28] has gained a lot of popularity in the text mining field; however, it is not so popular in the phishing email detection field [13]. It is a method for NLP, which is capable of capturing the context of a word in a text, its relation with other words, semantic and syntactic similarity, etc. Word2Vec deploys a NN model to learn word connections from a big collection of texts, which in this work is a collection of email texts. In particular, Word2Vec creates a vectorized representation of words, named embeddings, in which similar words are close distance-wise in the embedded space. Word2Vec is implemented in two ways, (a) Continuous Bag of Words (CBOW), and (b) skip-gram. The CBOW method predicts the target word based on the context; namely, the order of words does not influence the prediction. The skip-gram deploys the neighboring words to predict the target word. In this work, the skip-gram method has been deployed along with the hierarchical softmax method to train the model. The train complexity of skip-gram is calculated by:

$$Q = C * (D + D * log_2(V)) \tag{5}$$

Where (C) denotes the maximum distance of the words, (D) depicts the word representation, and (V) the dimensionality. In skip-gram, every word w is connected with two vectors that represent the word (u_w) and the context (v_w). The probability of correctly predicting a word w_i by giving a word w_j using the softmax method is:

$$p(w_i|w_j) = \frac{e^{u_{w_i} v_{w_j}}}{\sum_{l=1}^{V} e^{u_l v_{w_j}}} \tag{6}$$

The Word2Vec text-based feature set is created based on the equations 5 and 6.

3.4.3 Method 3: BERT. The emergence of BERT [11] has brought a significant advance in NLP tasks, as its core innovation is that it applies bidirectional transformer[4] encoders to get the contextual meaning of a text. Namely, as opposed to directional models (e.g., Recurrent NN, Convolutional NN, etc), which read the text from one direction (right-to-left or left-to-right), BERT reads the entire text from both directions. This feature allows the model to learn the context of a word based on the previous and next words. For instance, for the models that do not consider the context (like TF-IDF and Word2Vec), the word "account" would have the same representation in both "account for my actions" and in "bank account", while BERT will generate a representation of each word based on the other words in the sentence.

BERT performs two main steps:

(1) Input processing: Prior to embedding, a [CLS] token is added at the beginning of the input text (in our case the email's body), which shows the starting point. In case that a pair of sentences is used as input a [SEP] token is applied to separate the two sentences. During the embedding process, the text is parsed into three embedding layers: (1) The token embedding layer, which finishes the word segmentation and encodes the words via a vocabulary, (2) the segment embedding layer, which facilitates the model to differentiate the pair of sentences, and (3) the position embedding layer, which helps the model to acquire the sequence information of words. The input representation of the text is constructed by summing the embeddings of each layer and is used by the model to extract the feature vectors.
(2) Pre-training strategies:
    - Mask Language Model (MLM): A technique to randomly mask 15% of the words of a text, which are given to the model during the training and it tries to predict them. In this way, BERT learns the context of the text.
    - Next Sentence Prediction (NSP): In cases that two sentences are used as input to the BERT model, NSP facilitates the model to learn the relationship between the sentences (MLM helps the model to learn the relationship between the words), namely whether there is a semantic connection between the two sentences.

For each word in the body text of emails, both the previous and the next words are considered to produce the BERT text-based feature set.

3.5 Feature Selection

The feature selection task aims to identify a subset of the most informative features from the original text-based feature set to improve the performance of the ML classifiers in terms of accuracy and training time [16]. In this research, the Chi-Square feature selection [47] was used, as in previous works it has achieved promising results and it is easy to implement [46] [18]. The input data in this task is the feature sets of the three NLP methods (i.e., TF-IDF, Word2Vec, and BERT), and the output is a reduced subset for each method.

[4]Transformer [41] is a NN model that is based solely on attention mechanism and it performs well on language understanding tasks.

Chi-Square is a hypothesis test relating to statistics, which measures the relationship between two variables and identifies the level of correlation. This paper measures the dependency of a text-based feature with the email's class (i.e., phishing or benign). The score of Chi-Square is calculated by the equation (7):

$$x^2 = \sum_{l=1}^{n} \frac{(O_l - E_l)^2}{E_l}, \tag{7}$$

where O_l is the observed frequency, namely the number of observations of a class, and E_l is the expected frequency, namely the number of expected observations of a class when the feature and the class are independent. When the value of the equation 7 is closer to 0 the feature depends less on the class, otherwise, when the value is closer to 1, it depends more on the class.

3.6 Classification

The *classification* task is the crux of the proposed approach. The detection of phishing emails is by nature a binary classification problem. That is, the emails are grouped into two classes, benign emails, and phishing emails. We employ a binary variable to depict the class of an email, namely *x = 0* when the email is benign and *x = 1* when the email is phishing. The binary classification is performed via supervised ML. The objective of supervised learning is to find a discriminating function that learns to predict the output (email's class) based on the input (email's features).

The input of the *classification* is the three feature subsets created in the *feature selection* task, while the output is the prediction of the ML algorithm. To carry out the classification task and the comparative analysis, five well-known ML algorithms were deployed. These are: Logistic Regression (LR) [23], Decision Tree (DT) [32], Random Forest (RF) [38], Gradient Boosting Trees (GBT) [15], and Naive Bayes (NB) [36]. The grounds for choosing these algorithms were based on their performance that has been proved before in several binary classification tasks, as well as on previous works in the phishing email detection field. The *classification* task is divided into two phases; (1) the training phase, where a training dataset is used that contains the samples along with their label (benign or phishing) and the algorithm learns the discriminating function, and (2) the testing phase, where a testing dataset, which contains the samples without their associated labels, is provided to the trained algorithm and it predicts the label of each sample. The training phase concludes with one trained model for each algorithm that are later deployed by the testing phase to classify the emails between benign and phishing.

An essential part of the classification task is the hyperparameter tuning. Hyperparameters are the parameters of the algorithm that control the learning process and as hyperparameter tuning is defined the process of optimizing their value. In this approach, the hyperparameters' values have been estimated via manual tuning, which might require more effort, but it contributes to a deeper understanding of each algorithm's behavior and to a better knowledge of how the modifications of the hyperparameters affect the training of the algorithm. In particular, for the LR algorithm the hyperparameters that accomplished the best results are: the *maxIter=10* (maximum number of iterations for the classifier's training), the *threshold=0.5* (the threshold in the case of binary classification), the *regParam=0.0* (regularization parameter), and the *elasticNetParam=0.0* (ElasticNet mixing parameter, linearly combines the L_1 and L_2 penalties. In our case, where the value is 0.0, there is only the L_2 penalty). For the DT algorithm the hyperparameters that deployed are: *maxDepth=5* (the maximum depth of the trained tree), *maxBins=32* (the maximum number of bins when categorizing continuous features), and *minInfoGain=0.0* (the minimum information gain, after which a split at a tree node is considered). For the RF algorithm the hyperparameters that contributed to better performance are: the *numTrees=20* (the number of trees that are to be trained as part of the RF classifier), *maxBins=32*, *minInfoGain=0.0*, and *maxDepth=5*. For the GBT the deployed hyperparameters are: the *maxDepth=3*, the *maxIter=5* (the maximum number of gradient boosting iterations), the *maxBins=16*, the *stepSize=0.1* (the learning rate / step size towards reducing each estimator's contribution), and the *lossType='logistic'* (the Loss function which is aimed to be minimized with the classifier's training process - we used the Logistic Loss). Finally, for the NB the *smoothing=1.0* hyperparameter utilized (the smoothing parameter assists against the issue of zero probability in the NB algorithm).

4 EVALUATION

This section presents a comprehensive evaluation of the proposed approach through its classification results. The experiments conducted in this paper were performed in a virtual machine stored in a VMWare ESXi server with Intel Xeon 4114, 2.20 GHz x 8 CPU, and 16 GB RAM. The OS of the virtual machine was Ubuntu 20.04 64-bit. The comparison approach was implemented in Apache Spark [1]. At this point, it is important to mention that the source-code is released as open-source[5] to help the security community to curb the threat of phishing emails.

4.1 Datasets

The experimental data were obtained from two publicly available email collections; the Enron email corpus [3] that was used as the benign dataset and the Jose Nazario's phishing corpus [5] that was used as the phishing dataset. According to the survey in [7], these datasets have been widely used in previous works. The Enron corpus is a large dataset that contains about 500,000 emails from 158 employees of the Enron Corporation. It is one of the few publicly available mass collections of real emails in which the emails were made public during the legal investigation of the corporation. Benign emails correspond to interactions among different employees on day-to-day work issues. Over the years, the Enron corpus gained popularity as it was used in various research fields, such as phishing detection, NLP, and data mining [13] [26] [25]. The phishing corpus, at the time of writing, is the most popular phishing email dataset as it has been deployed in various relevant research works over the years [43] [42] [31] [18] [17]. The phishing emails of the corpus came from the personal inbox of its creator, which contains several phishing emails from 2004 until 2020. In this work, we focused on the quality of the phishing emails and not the quantity, thus, we have deployed only the newer phishing emails that take into account the progress of the phishing email attack (i.e., the phishing

[5]https://github.com/KostasKoutrou/Text_Phishing_Email_ML_Classification

Table 1: Balanced dataset.

Dataset	Benign	Phishing	Total
Training set	1,125	1,125	2,248
Testing set	282	282	566
Total	1,407	1,407	2,814

Table 2: Imbalanced dataset.

Dataset	Benign	Phishing	Total
Training set	11,200	1,125	12,325
Testing set	2,800	282	3,082
Total	14,000	1,407	15,407

email structure is not the same as it was ten years ago). For this reason, all the phishing emails from the years 2015-2020 were used.

Two diverse datasets were created for the evaluation of the proposed approach; the balanced dataset and the imbalanced dataset. In both datasets, we have utilized all the phishing emails from the years 2015-2020 of the phishing corpus, which in total are 1,407. In the balanced dataset the same number of benign emails has been randomly chosen from the Enron corpus. Regarding the imbalanced dataset, a more realistic scenario[6] has been considered in which the ratio between phishing and benign emails are 1:10. Therefore, 14,000 emails have been randomly chosen from the Enron corpus. The ratio between training and testing in both experiments is 80:20. The structure of the balanced and imbalanced datasets is presented in Table 1 and Table 2 respectively.

4.2 Metrics

Considering that True Positive (TP) is the number of correctly classified phishing emails, False Positive (FP) is the number of mistakenly classified benign emails, True Negative (TN) is the number of correctly classified benign emails, and False Negative (FN) is the number of mistakenly classified phishing emails, the deployed metrics upon which we rely our evaluation are the following:

- *Recall*: Depicts the correctly classified phishing emails compared to the total number of phishing emails:
$$TPR = \frac{TP}{TP + FN} \tag{8}$$
- *Precision*: Depicts the correctly classified phishing emails compared to the total number of emails that were classified as phishing:
$$Precision = \frac{TP}{TP + FP} \tag{9}$$
- *Accuracy*: Percentage of correctly classified emails compared to the total number of emails:
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$
- *F1-Score*: A metric to measure the accuracy, which considers both the *precision* and *recall*, namely it calculates the balance between the *precision* and *recall*. It presents a better measure in case of class imbalance (e.g. when the phishing emails are fewer than benign).
$$F1\text{–}Score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{11}$$
- *False Positive Rate (FPR)*: Represents the mistakenly classified benign emails compared to the total number of benign emails:
$$FPR = \frac{FP}{FP + TN} \tag{12}$$
- *False Negative Rate (FNR)*: Represents the mistakenly classified phishing emails compared to the total number of phishing emails:
$$FNR = \frac{FN}{TP + FN} \tag{13}$$
- Receiver Operating Characteristic (ROC) Curve: Is a graph that plots the TPR and the FPR. Particularly, it depicts the performance of a classification algorithm at various classification thresholds. As an easy to understand guideline, the closer the ROC curve is to the shape of an upside down L, the better.
- Area Under the ROC Curve (AUC): Measures the area under the ROC curve.

[6]In daily basis the benign emails that an organization receives are much more than the phishing emails

4.3 Results & Discussion

Fifteen different combinations of NLP and ML methods have been tested in this paper. In particular, the TF-IDF, Word2Vec, and BERT methods were joined with five state-of-the-art ML algorithms, which are the LR, DT, RF, GBT, and NB. The performance of each combination was evaluated on different ratios between the emails (phishing and benign) using a *balanced* dataset with the same ratio of emails and an *imbalanced* dataset with ten times more benign emails than phishing.

4.3.1 *Experiment 1: Balanced Dataset.*

In the balanced dataset, the accuracy is used as a metric for the identification of the best detection performance, since the distribution of the samples across the classes is equal and it is a metric easier for the reader to comprehend.

For the TF-IDF method, all the ML algorithms accomplished more than 90% accuracy. In particular, RF, BN, LR, GBT, and DT, achieved 93.41%, 92.77%, 92.48%, 91.47%, and 90.17% classification accuracy respectively. However, the best results were divided between the RF and LR classifiers. RF achieved the best results on accuracy, precision, FPR, and AUC, while LR achieved the best results in recall, F1-score, and FNR. Thus, on the balanced dataset the TF-IDF method had a greater performance with the RF and the LR classifiers. The results of TF-IDF on all the evaluation metrics in the balanced dataset are shown in Table 3. The ROC curves for TF-IDF are presented in Figure 2, where one can notice that the plots of RF, NB, and GBT are almost similar, nevertheless the AUC of RF is slightly greater.

With the Word2Vec method, overall, all the ML algorithms achieved better classification results than TF-IDF, since the accuracy of all ML algorithms was above 95%. More specifically, RF, GBT, LR, DT, and NB accomplished 98.95%, 97.48%, 96.77%, 96.25%, and 95.64% classification accuracy respectively. The results indicate that the RF performs better than the other algorithms with the Word2Vec features attaining highest accuracy with very little FPR (1.4%) and FNR (0.68%). The results of Word2Vec on all the evaluation metrics in the balanced dataset are depicted in Table 4. The ROC curves for Word2Vec are shown in Figure 3, where in general the results were promising for all the ML algorithms, and especially for the RF, LR, and GBT that achieved more than 99% AUC. On average, the ROC curves in Word2Vec are the best among the three NLP methods.

The ML algorithms with BERT accomplished the worst results in comparison with TF-IDF and Word2Vec. The worst accuracy (66.54%) was achieved by the NB, while the best (84.49%) was achieved by LR. The accuracy of DT, RF, and GBT ranged from 80.68% (DT) to 83.27% (RF). Furthermore, LR attained the lowest FNR (15.9%), while the lowest FPR was accomplished by RF (6.4%). The results of BERT on all the evaluation metrics in the balanced dataset are depicted in Table 4. The ROC curves for BERT are presented in Figure 4. As it can be noticed, the BERT ROC curves are the worst of the three NLP methods.

4.3.2 Experiment 2: Imbalanced Dataset. When the accuracy is deployed to measure the performance of an ML algorithm on imbalanced data it leads to biased results, as its estimation is biased towards the class that contains the most samples (in our case its the benign). To overcome this challenge, we deployed the F1-score metric that is less susceptible to be affected by class imbalance.

For the TF-IDF method, as one can notice from Table 6, the LR and GBT algorithms made the most accurate predictions with 89.96% and 81.83% F1-scores respectively. The DT algorithm accomplished 70.3% F1-score, while the RF and NB did not perform well. Moreover, the inappropriateness of accuracy as a measurement in experiments with imbalanced data can be seen in the results of NB, where even though all the phishing samples were misclassified, its classification accuracy was 92.05%. The ROC curve of all the ML algorithms with TF-IDF features in the imbalanced dataset can be seen in Figure 5, where LR overcome all the other algorithms.

In this experiment, again the ML algorithms with Word2Vec, overall, accomplished better results in comparison with TF-IDF. In particular, the LR attained a 92.41% F1-score. Overall, the performance of all the ML algorithms, except from NB (it failed to classify all the phishing samples), was promising as they achieved more than 84% F1-score. The detailed results of all the ML algorithms with the Word2Vec features, as well as the ROC curves can be found in Table 7 and Figure 6 respectively. Similarly to the experiments performed with the balanced dataset, in this case as well, the use of Word2Vec resulted in the best ROC curves among the three NLP methods.

As in experiment 1, the ML algorithms did not manage to accomplish good results with BERT features, since the F1-score of all the algorithms was worse than the F1-scores with TF-IDF and Word2Vec features. The greatest F1-score was achieved by LR (63.92%) and it was significantly lower than the score of LR with TF-IDF (89.96%) or with Word2Vec (92.41%). Table 8 presents the results of all the ML algorithms with BERT in the imbalanced dataset, while the ROC curves are shown in Figure 7.

4.3.3 Discussion. Regarding the first experiment, in the balanced dataset, almost all the algorithms with both TF-IDF and Word2Vec attained promising results. However, the results indicate that the Word2Vec features are more informative for the ML algorithms,

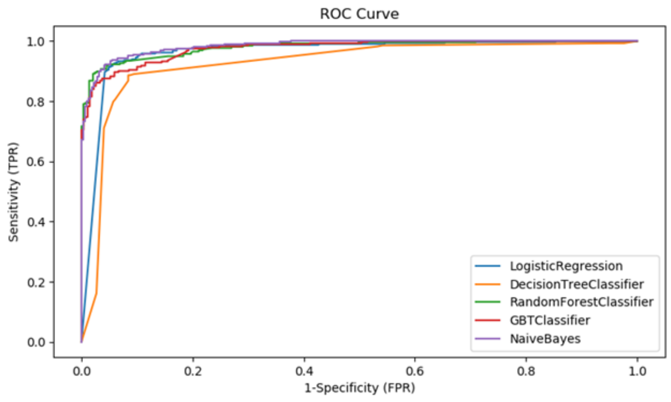

Figure 2: ROC curve TF-IDF balanced dataset.

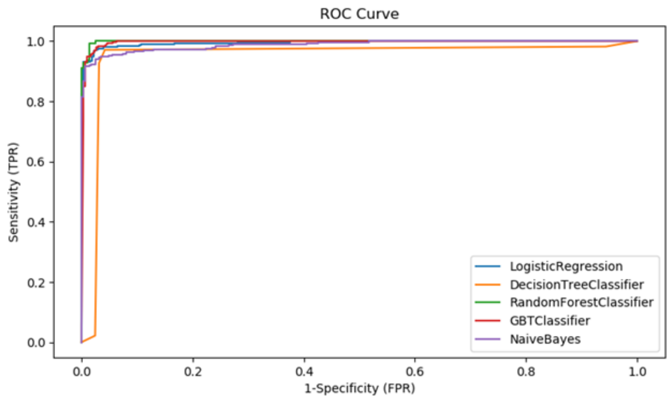

Figure 3: ROC curve Word2Vec balanced dataset.

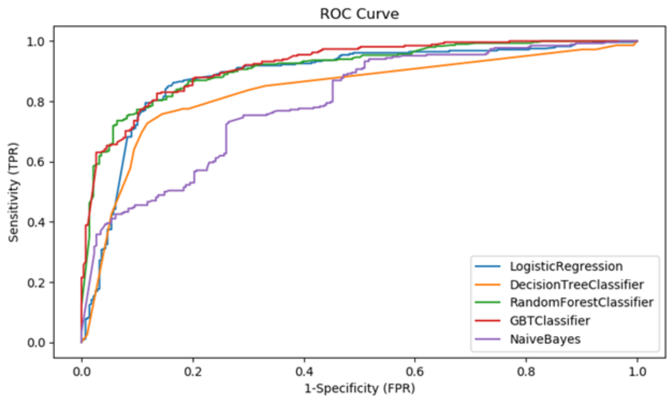

Figure 4: ROC curve BERT balanced dataset.

as in all the evaluation metrics, their results were better. More specifically, the Word2Vec/RF pair attained the best classification performance with 98.97% accuracy and it proved to be the best combination when trained with balanced data. In the second experiment, we relied only on the f1-score to make decisions about the NLP/ML combinations, as it is the most accurate metric for imbalanced data. Again, the ML algorithms with the Word2Vec features performed

Table 3: TF-IDF results balanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	0.9248	0.9110	**0.9488**	**0.9295**	0.1013	**0.0511**	0.9766
DT	0.9017	0.9063	0.8864	0.8962	0.0841	0.1135	0.8385
RF	**0.9341**	**0.9743**	0.8837	0.9268	**0.0207**	0.1162	**0.9795**
GBT	0.9147	0.9301	0.8837	0.9268	**0.0207**	0.1162	0.9761
NB	0.9277	0.9652	0.8853	0.9235	0.0309	0.1146	0.5479

Table 4: Word2Vec results balanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	0.9677	0.9755	0.9554	0.9653	0.0213	0.0445	0.9948
DT	0.9625	0.9535	0.9709	0.9621	0.0456	0.0290	0.9532
RF	**0.9895**	**0.9863**	**0.9931**	**0.9897**	**0.0140**	**0.0068**	**0.9988**
GBT	0.9748	0.9727	0.9761	0.9744	0.0264	0.0238	0.9961
NB	0.9564	0.9646	0.9479	0.9562	0.0349	0.0520	0.8200

Table 5: BERT results balanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	**0.8449**	0.85	**0.8409**	**0.8454**	0.1510	**0.1590**	0.8834
DT	0.8068	0.8358	0.7577	0.7949	0.1452	0.2422	0.6945
RF	0.8327	**0.9243**	0.7357	0.8193	**0.0640**	0.2642	0.9098
GBT	0.8245	0.8185	0.8339	0.8261	0.1849	0.1660	**0.9184**
NB	0.6654	0.7407	0.5223	0.6126	0.1877	0.4776	0.5169

Table 6: TF-IDF results imbalanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	**0.9841**	0.9031	**0.8961**	**0.8996**	0.0082	**0.1038**	**0.9861**
DT	0.9592	0.8655	0.5919	0.7030	0.0081	0.4080	0.5539
RF	0.9443	**1**	0.3193	0.4841	**0**	0.6806	0.9666
GBT	0.9728	0.9032	0.7480	0.8183	0.0071	0.2519	0.9792
NB	0.9205	0	0	0	**0**	1	0.5317

Table 7: Word2Vec results imbalanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	**0.9862**	0.9319	**0.9163**	**0.9241**	0.0067	0.0836	**0.9966**
DT	0.9768	0.9082	0.7908	0.8455	0.0069	0.2091	0.7753
RF	0.9769	**0.9441**	0.7602	0.8423	0.0039	0.2397	0.9830
GBT	0.9901	0.9736	0.8566	0.8477	0.8521	**0.0139**	0.1522
NB	0.9195	0	0	**0**	**0**	1	0.8658

Table 8: BERT results imbalanced dataset.

Classifier	Accuracy	Precision	Recall	F1-score	FPR	FNR	AUC
LR	**0.9407**	0.6286	**0.6501**	**0.6392**	0.0337	**0.3498**	0.9087
DT	0.9347	0.6527	0.4982	0.5651	0.0246	0.5017	0.2472
RF	0.9388	0.7142	0.5105	0.5954	0.0197	0.4894	0.8947
GBT	0.9399	0.6861	0.5774	0.6271	0.0253	0.4225	**0.9263**
NB	0.9199	**0.8125**	0.050	0.0945	**0.0010**	0.9498	0.5446

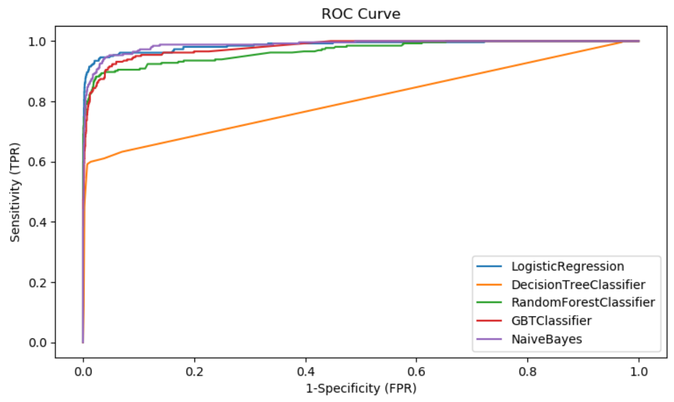

Figure 5: ROC curve TF-IDF imbalanced dataset.

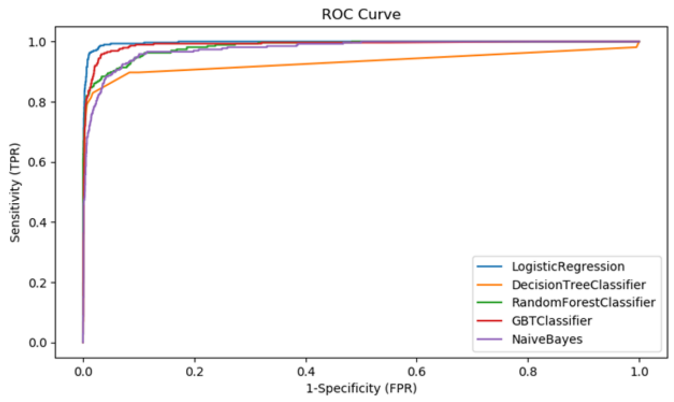

Figure 6: ROC curve Word2Vec imbalanced dataset.

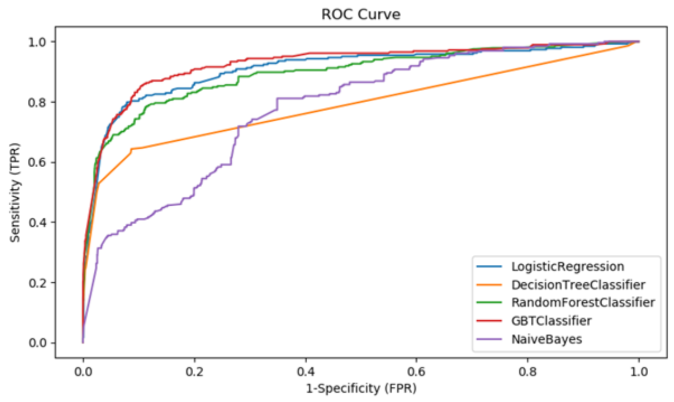

Figure 7: ROC curve BERT imbalanced dataset.

better, achieving state-of-the-art results. Based on the results of the second experiment, we concluded that the Word2Vec/LR pair is the best combination when trained with imbalanced data, since it had the highest F1-score (92.41%). BERT, even though it is a powerful NLP method, did not manage to accomplish good results when deployed for textual feature extraction and combined with ML algorithms for phishing email detection. Finally, based on the results of the second experiment, one can notice that the accuracy is not an appropriate metric to measure the performance of classifiers in imbalanced datasets, as its results are biased towards the majority class.

5 CONCLUSION

The modus operandi of modern phishing attacks leverage the unprecedented events due to the COVID-19 pandemic, to deceive users and threaten their privacy by gaining access to their private information, such as credentials, and banking accounts. This research focused on the comparison of different combinations of NLP and ML methods for the detection of phishing emails to identify which combination performs better on phishing email detection. The TF-IDF, Word2Vec, and BERT NLP methods were deployed to extract textual features from the emails' body text and build three distinct feature sets. Each feature set was then processed by the chi-squared feature selection method to identify the most informative features that will be fed on the LR, DT, RF, GBT, and NB to classify the emails between phishing and benign. To evaluate the NLP/ML combinations two experiments were performed, one with a balanced ratio between the phishing and benign emails and one with an imbalanced ratio, where the benign emails are ten times more than phishing. The evaluation results showed that the Word2Vec method is the most appropriate for an ML phishing email detection approach that focuses on the emails' body text. In particular, the Word2Vec/RF combination accomplished the best results in the balanced dataset and the Word2Vec/LR in the imbalanced.

The future work will focus on an approach that will combine two types of features, namely features retrieved via the NLP methods from the emails' body (which were the main focus of this paper) and features retrieved from the email's contents, such as URLs, email header fields, and attachments, to test whether this approach will lead to better results. Furthermore, the classification task will be reinforced using deep learning algorithms to tune the results of the phishing email detection approach.

ACKNOWLEDGMENTS

This research was partially funded by the European Commission (Horizon 2020 Programme), and particularly by the projects CyberSec4Europe (Grant Agreement no. 830929) and SECONDO (Grant Agreement no. 823997), as well as by the Greek state (Operational Programme Competitiveness, Entrepreneurship and Innovation 2014-2020 (EPAnEK)), under the project NetPHISH (Grant Agreement no. T1EΔK-05112).

REFERENCES

[1] Accessed: April 2021. Aparche Spark - Unified Analytics Engine for Big Data. https://spark.apache.org/.
[2] Accessed: April 2021. Enisa Threat Landscape 2020 - Phishing. https://www.enisa.europa.eu/publications/phishing.
[3] Accessed: April 2021. Enron Email Dataset. http://www.cs.cmu.edu/~./enron/.
[4] Accessed: April 2021. Interpol COVID-19 Cybercrime Analysis Report. https://www.interpol.int/News-and-Events/News/2020/INTERPOL-report-shows-alarming-rate-of-cyberattacks-during-COVID-19.
[5] Accessed: April 2021. Jose Nazario Phishing Email Corpus. https://monkey.org/~jose/phishing/.
[6] APWG. 2021. Phishing Activity Trends Report 4th Quarter 2020. https://docs.apwg.org/reports/apwg_trends_report_q4_2020.pdf.

[7] Abdul Basit, Maham Zafar, Xuan Liu, Abdul Rehman Javed, Zunera Jalil, and Kashif Kifayat. 2020. A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems* (2020), 1–16.
[8] André Bergholz, Gerhard Paaß, Frank Reichartz, Siehyun Strobel, and Schloß Birlinghoven. 2008. Improved phishing detection using model-based features. In *In Fifth Conference on Email and Anti-Spam, CEAS*.
[9] Esteban Castillo, Sreekar Dhaduvai, Peng Liu, Kartik-Singh Thakur, Adam Dalton, and Tomek Strzalkowski. 2020. Email Threat Detection Using Distinct Neural Network Approaches. In *Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management*. 48–55.
[10] A. Das, S. Baki, A. El Aassal, R. Verma, and A. Dunbar. 2020. SoK: A Comprehensive Reexamination of Phishing Research From the Security Perspective. *IEEE Communications Surveys Tutorials* 22, 1 (2020), 671–708. https://doi.org/10.1109/COMST.2019.2957750
[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[12] G. Egozi and R. Verma. 2018. Phishing Email Detection Using Robust NLP Techniques. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. 7–12.
[13] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang. 2019. Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism. *IEEE Access* 7 (2019), 56329–56340.
[14] Ingo Feinerer and Kurt Hornik. 2020. *wordnet: WordNet Interface*. https://CRAN.R-project.org/package=wordnet R package version 0.1-15.
[15] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.
[16] D Asir Antony Gnana, S Appavu Alias Balamurugan, and E Jebamalar Leavline. 2016. Literature review on feature selection methods for high-dimensional data. *International Journal of Computer Applications* 975 (2016), 8887.
[17] Eder S Gualberto, Rafael T De Sousa, P De B Thiago, João Paulo CL Da Costa, and Cláudio G Duque. 2020. From feature engineering and topics models to enhanced prediction rates in phishing detection. *Ieee Access* 8 (2020), 76368–76385.
[18] Eder Souza Gualberto, Rafael Timoteo De Sousa, Thiago Pereira De Brito Vieira, João Paulo Carvalho Lustosa Da Costa, and Cláudio Gottschalg Duque. 2020. The Answer is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering. *IEEE Access* 8 (2020), 223529–223547.
[19] C. N. Gutierrez, T. Kim, R. D. Corte, J. Avery, D. Goldwasser, M. Cinque, and S. Bagchi. 2018. Learning from the Ones that Got Away: Detecting New Forms of Phishing Attacks. *IEEE Transactions on Dependable and Secure Computing* 15, 6 (2018), 988–1001.
[20] Lukáš Halgaš, Ioannis Agrafiotis, and Jason Nurse. 2020. *Catching the Phish: Detecting Phishing Attacks Using Recurrent Neural Networks (RNNs)*. 219–233.
[21] NB Harikrishnan, R Vinayakumar, and KP Soman. 2018. A machine learning approach towards phishing email detection. In *Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP)*, Vol. 2013. 455–468.
[22] Maryam Heidari and James H Jones. 2020. Using bert to extract topic-independent sentiment features for social media bot detection. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 0542–0547.
[23] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. Vol. 398. John Wiley & Sons.
[24] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab. 2019. A Comprehensive Survey for Intelligent Spam Email Detection. *IEEE Access* 7 (2019), 168261–168295. https://doi.org/10.1109/ACCESS.2019.2954791
[25] Greg Kessler. 2010. Virtual business: An Enron email corpus study. *Journal of Pragmatics* 42, 1 (2010), 262–270.
[26] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, 217–226.
[27] Hiransha M, Nidhin Unnithan, Vinayakumar R, and Soman Kp. 2018. Deep Learning Based Phishing E-mail Detection CEN-Deepspam.
[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]
[29] Mimecast. 2021. Securing the Enterprise in the COVID world, The State of Email Security.
[30] Rajvardhan Oak, Min Du, David Yan, Harshvardhan Takawale, and Idan Amit. 2019. Malware detection on highly imbalanced data through sequence modeling. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 37–48.
[31] Gilchan Park and Julia M Taylor. 2015. Using syntactic features for phishing detection. *arXiv preprint arXiv:1506.00037* (2015).
[32] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.
[33] Vinayakumar Ra, Barathi Ganesh HBa, Anand Kumar Ma, Soman KPa, Prabaharan Poornachandran, and A Verma. 2018. DeepAnti-PhishNet: Applying deep neural networks for phishing email detection. In *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*. Tempe, AZ, USA, 1–11.
[34] Abir Rahali and Moulay A Akhloufi. 2021. MalBERT: Using Transformers for Cybersecurity and Malicious Software Detection. *arXiv preprint arXiv:2103.03806* (2021).
[35] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. New Jersey, USA, 133–142.
[36] Irina Rish et al. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Vol. 3. 41–46.
[37] Krupal Shah, Nirav Shah, Shaival Shah, and Dip Patel. 2020. Email User Classification and Topic Modeling. In *Proceedings of the Future Technologies Conference*. Springer, 359–377.
[38] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1. 278–282 vol.1.
[39] Nidhin A Unnithan, NB Harikrishnan, S Akarsh, R Vinayakumar, and KP Soman. 2018. Machine learning based phishing e-mail detection. *Security-CEN@ Amrita* (2018), 65–69.
[40] Nidhin A Unnithan, NB Harikrishnan, R Vinayakumar, KP Soman, and Sai Sundarakrishna. 2018. Detecting phishing E-mail using machine learning techniques. In *Proc. 1st Anti-Phishing Shared Task Pilot 4th ACM IWSPA Co-Located 8th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*. 51–54.
[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[42] Rakesh Verma and Nabil Hossain. 2013. Semantic feature selection for text with application to phishing email detection. In *International Conference on Information Security and Cryptology*. Springer, 455–468.
[43] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. 2012. Detecting phishing emails the natural language way. In *European Symposium on Research in Computer Security*. Springer, 824–841.
[44] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. 2012. Detecting Phishing Emails the Natural Language Way. In *Computer Security – ESORICS 2012*, Sara Foresti, Moti Yung, and Fabio Martinelli (Eds.). 824–841.
[45] Rakesh M. Verma, Victor Zeng, and Houtan Faridi. 2019. Data Quality for Security Challenges: Case Studies of Phishing, Malware and Intrusion Detection Datasets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2605–2607. https://doi.org/10.1145/3319535.3363267
[46] Masoumeh Zareapoor and KR Seeja. 2015. Feature extraction or feature selection for text classification: A case study on phishing email detection. *International Journal of Information Engineering and Electronic Business* 7, 2 (2015), 60.
[47] Yujia Zhai, Wei Song, Xianjun Liu, Lizhen Liu, and Xinlei Zhao. 2018. A chi-square statistics based feature selection method in text classification. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 160–163.