Identification of Pharming in Communication Networks using Ensemble Learning

N. A. Azeez[1*], S. S. Oladele[1], O. Ologe[2]

[1]Department of Computer Sciences, Faculty of Science, University of Lagos, Akoka, Nigeria
[2]Department of Applied Geophysics, Faculty of Science, Federal University Birnin Kebbi, Nigeria

ABSTRACT: Pharming scams are carried out by exploiting the DNS as the main weapon while phishing attacks employ spoofed websites that appear to be legitimate to internet users. Phishing makes use of baits such as fake links but pharming leverages and negotiates on the DNS server to move and redirect internet users to a fake and simulated website.Having seen several challenges through pharming resulting into vulnerable websites, personal emails and accounts on social media, the usage and reliability on internet calls for caution. Against this backdrop, this work aims at enhancing pharming detection strategies by adopting machine learning classification algorithms. To further obtain the best classification results, an ensemble learning approach was adopted. The algorithms used include K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Support Vector Machine, Adaptive Boosting, Gradient Boosting, and Extra Trees Classifier. During the testing process, the classifiers were tested against four popular metrics: accuracy, recall, precision, F1 score, and Log loss. The results demonstrate the performance of all algorithms used, as well as their relationships. The ensemble model that included Logistic Regression, K-Nearest Neighbors, Decision Tree, Support Vector Machine, Gradient Boosting Classifier, AdaBoost Classifier, Extra Trees Classifier, and Random Forest produced the best results after evaluating them on the two datasets. Random Forest Classifiers showed a better performance of the classifiers, with mean accuracies of 0.932 and 0.939, respectively for each of the datasets when compared to 0.476 and 0.519 obtained for Naive Bayes.

KEYWORDS: Pharming, Ensemble learning, Communication networks, Cybersecurity, Performance, Classifiers

[Received Mar. 29, 2022; Revised May 19, 2022; Accepted May 20, 2022] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

The e-mail has developed into a critical networking method for both corporate and personal purposes. There is an increasingly rising threat known as "pharming" among the many security concerns that impact computer users (Azeez et. al., 2019). Hackers use pharming attacks to persuade people to access a fake website, call a fake phone number, or download malicious software to steal personal information such as account numbers, login credentials, credit card numbers, passwords, or PINs. To put it simply, pharming is the practice of sending unsolicited e-mails from fraudulent individuals through a simulated website from a DNS server.

It may even be a gateway to a malicious website designed to trick individual into uploading malware or divulging personal details (Hewage, *et al*., 2021). Pharming scams are carried out by exploiting the DNS as the main weapon while phishing attacks employ spoofed websites that appear to be legitimate to internet users. Phishing makes use of baits such as fake links but pharming leverages and negotiates on the DNS server to move and redirect internet users to a fake and simulated website.

Another form of pharming is spear-pharming, which is a targeted pharming attack in which attackers redefine their targets and strategies with craft messages that are both personal and significant. To this effect, spear pharming is difficult to spot and much harder to defend against (Rashid, 2020).

It is reasonable to believe that certain similar targets and attackers exist. Phishers are motivated by two things: money or facts (which usually leads to money). But, when it comes to cyber espionage at the corporate or nation-state level, the consequences of a well-placed pharming e-mail are astounding. Over the years, this has/or can lead to the loss of trade secrets, collapse of global economies, and even endangering national security (Jang-Jaccard & Nepal, 2014).

In May 2017, a massive pharming attack targeting millions of Gmail users hit google, in which the hacker gained access to the e-mail histories of users. The phishers were able to pose e-mails as coming from a known source and ask target to search the attached file using this information. Users were asked to allow a bogus app to handle their e-mail accounts after clicking a link to an attack file (Hadnagy & Fincher, 2015).

As cyber defense technology evolves at a rapid pace, pharming attacks remain a major cybersecurity threat. (Human

*Corresponding author: nazeez@unilag.edu.ng
doi: http://dx.doi.org/10.4314/njtd.v19i2.10

Factor Report, 2019). One of the key reasons for this puzzle is that humans remain a significant weakness, and study into human identification decisions of pharming e-mails has not progressed as quickly as it should have (Gonzalez, *et al*., 2014).

While phishers are adaptive in their actions and actively change their behaviors to prevent detection, the conventional methods used for the majority of e-mail filters for detecting these e-mails are static and therefore ineffective in dealing with the most recent pharming trends (Azeez et. al., 2021).

This work aims to identify pharming e-mails using ensemble learning. In the course of this attempt, nine machine learning algorithms were considered which include K-Nearest Neighbors, Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Support Vector Machine, Adaptive Boosting, Gradient Boosting, and Extra Trees Classifier (Protti, 2003). A final evaluation was carried out to compare the overall performance of results against studies of similar nature.

The methods employed in this approach are more suited to this work because while ensemble training models do not always accurately produce a completely accurate prediction, they are useful for determining and covering up shortcomings in classification models when performing as an individual (Dong, *et al*., 2019).

The rest of the paper is organized as follows: related works about pharming are discussed in Section II while Section III focuses on the methodology for detecting pharming attacks based on URLs and the performance evaluation metrics. Section IV presents the results obtained from the numerous experiments carried out in the study, while Section V concludes the paper with some recommendations on the future direction on how the work may be extended.

II. RELATED WORKS

Pharming emails are the main entry point for pharming websites, according to a series of articles detailing the different pharming email tools used. Machine learning algorithms that use supervised or unsupervised learning methods to detect and classify pharming e-mails can be used to detect and classify pharming e-mails (Dada *et al*., 2019).

Chandrasekaran, *et al*., in 2006 proposed a method for classifying phishing emails based on structural properties. They used a total of 25 elements, with both style markers (such as the word’s suspended, account, and security) as well as structural elements such as the subject line layout and the structure of the body greeting. They put 200 emails to the test (100 phishing and 100 legitimate). They used simulated annealing as a feature selection algorithm. Information Gain (IG) was used to list these features based on their usefulness after selecting a feature collection. To classify phishing emails based on the chosen features, Support Vector Machine (SVM) was adopted. 95% detection rate was achieved with a low false-positive rate (Azeez et. al., 2019).

Jameel & George, in 2013, proposed a phishing identification model that focused on 18 extracted features from an e-header mail's and HTML body. To determine whether the checked email is phishing or not, their built model uses statistically dependent criteria called features existence weight. After conducting tests, the model used only 7 of the 18 extracted e-mail features. The result yielded a strong identification rate of 97.79 percent and a fast required processing time of 0.0004 msec.

Another phishing identification model based on derived email properties was developed to identify phishing emails. These characteristics were found in the message's HTML and header content, and it used a feed-forward neural network to identify the checked email as harm or phish email (Jameel & Loay, 2013). The evaluation revealed that a high recognition rate of 98.72 percent and a brief response period of 0.00067 milliseconds.

Ona *et al*., in 2019 defined how to create a Scrum-based implementation for a real-time learning, feature selection, and neural network algorithms. The solution provides a tool that can detect and counteract a pharming attack that has been registered on the e-mail system. Using the blacklist of PhishTank for validation of test results, the proof of concept demonstrated that the applied feature selection method discards obsolete electronic mail properties while the neural network algorithm accepts these characteristics, resulting in optimal learning with no redundancies. The findings of the three data sets were evaluated, and the average accuracy was 93.9 percent.

From data extraction, filtering, integration, aggregation, and knowledge extraction, the concepts of knowledge discovery were applied by Paliath *et al*., in 2020. They compared six machine-learning methods for detecting pharming using a limited number of carefully selected functions. They measured false positives, false negatives, mean absolute error, memory, accuracy, and F-measure. The false positive and negative rates were very poor. They found that Naive Bayes has the lowest true positive score and that Neural Networks, with an overall accuracy of 99.4%, is the most reliable for effective pharming identification. However, neural networks had a significant MAE rate of 1.5 percent and caused virtually no deterioration in classification efficiency.

Singh *et al*., in 2020 conducted a research into the characteristics of phishing emails that made them impossible to spot by humans (Sampson, 2015). They took an existing data collection of phishing and ham emails and expanded it by gathering annotations of the features that made phishing emails. They then used the new, annotated data set to conduct cluster analyses to classify the types of emails and their attributes. After evaluating the accuracy of detection in each segment the results show that the resemblance of phishing email features to innocuous email features plays a critical role in detection accuracy. It was also pointed out that phishing emails that are the most analogous to ham emails have the lowest accuracy, while phishing emails which are the most distinct to ham emails have the highest accuracy.

Rawal *et al*., in 2017 conducted an experiment in which the identification of a phished email address was treated as a classification challenge. They conducted tests by classifying emails as phish or ham through the use of machine learning algorithms. It was discovered that classification using SVM and Random Forest classifier yielded a maximum accuracy of 99.87%.

With the use of NLP to extract useful keywords and vector embedded techniques, as well as various machine learning algorithms on a corpus of e-mails, identified and classified phished e-mails. Oladimeji (2019), found that Naïve Bayes had the highest classification accuracy of 99.0% compared to SVM with an accuracy of 98.6% and KNN with an accuracy of 96.9% respectively.

III. METHODOLOGY

The method adopted entails four stages: data collection and labelling, pre-processing, classification, performance evaluation.

A. Data Collection and Labelling

This stage encompasses the acquisition and labelling of various email datasets from the public corpora; Kaggle and monkey.org. The acquired datasets were in either raw message format or compiled in an mbox file and already labelled as either Ham or Phish by the sources they were acquired from. A total of 5,040 emails were obtained during collection, with 2,280 labelled as Phish and 2760 labelled as Ham.

The first dataset was acquired from: (https://www.kaggle.com/beatoa/spamassassin-public-corpus).

The dataset found on Kaggle was a subset of the SpamAssassin public corpus prelabelled by a researcher. This dataset consists of files containing the raw email messages and required preprocessing to acquire useful features.

The second dataset which was collected over time from a private mailbox, was acquired from the repository: (http://monkey.org/~jose/wiki/doku.php?id=PharmingCorpus). Dataset 2 was of the mbox format (a file consisting of email files and metadata) and also required further preprocessing to extract the needed features.

B. Email Preprocessing and Parsing

This phase describes the process utilized in the parsing, extraction and engineering of features from the collected datasets. After collecting the needed data, all files and email messages were parsed and trimmed for the email content which consists of the header and body. Parsing the collected dataset required careful consideration due to the presence of Unicode characters which make frequent occurrences in email bodies.

1) Feature extraction

After parsing and obtaining each email body, the dataset is then pre-processed via several steps. Python was used to parse each email's data. The parsed data is then binary encoded with a value of 1 if the feature exists and a value of 0 if it does not exist. The desired output of email data is also encoded, with a value of 0 for ham email and a value of 1 for a pharming email. The list of categorization characteristics is based on Kim Soon *et al*., 2020. The characteristics utilized for categorization in this study were mentioned in Table 1. All of these encoded characteristics are recorded in a CSV file to be utilized for categorization.

2) Feature Storing

Following the feature extraction phase, two files (phish and ham emails) were generated and stored in .xlsx format for use in the training phase.

C. Email Classification

In this phase, the primary aim is to observe how well the stacking ensemble method works when variations of classification models are used as base learners when compared to each model individually. The base learners used are KNearest Neighbors, Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Support Vector Machine, Adaptive Boosting, Gradient Boosting, and Extra Trees Classifier (Gansterer. & Pölz, 2009).

Naïve Bayes (NB): This is a classification method as well as a machine learning algorithm. The fundamental mechanics of the algorithm was motivated by the Bayes Theorem when the output variable is discrete. The Bayes Theorem, which states the following equation: (Al-Saaidah, 2017).

$$P(A|B) = \frac{P(A|B)*P(A)}{P(B)} \quad (1)$$

To make it simpler to grasp, equation 1 can be rewritten with input variables X and output variables y. This equation calculates the likelihood of y given the input features X.

$$P(y|X) = \frac{P(X|y)*P(y)}{P(X)} \quad (2)$$

Alternatively, P(X|y) can be re-written as:

$$P(X|y) = P(X_1|y) * P(X_2|y) * ... * P(X_n|y) \quad (3)$$

Since P(X) is a constant, we can extract from the equation and add a proportionality.

$$P(y|X) \propto P(y) * \prod_{i=1}^{n} P(X_i|y) \quad (4)$$

The Naive Bayes aims to choose the class y with the highest likelihood. Using the Argmax equation (an operation that finds the argument with the max value from a target function). The maximum y value in this case is:

$$y = argmax_y [P(y) * \prod_{i=1}^{n} P(X_i|y)] \quad (5)$$

A derivative of the Naïve Bayes called **Gaussian Naïve Bayes** is used as an ensemble member in the course of this study. This algorithm is built on Bayes' theorem with the assumption of independence between a pair of characteristics, making it a very efficient classifier as opposed to more complex approaches. It is represented by the mathematical equation:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (6)$$

1) Support Vector Machine (SVM)

This is a memory-efficient classifier that works well in spaces that are characterized with high-dimension and employs only a subset of training points in the decision function. It is a depiction of the training data as points in space separated into groups by a wide distance. New data is plotted into the same space and can be estimated to belong to that group depending on which side of the distance they fall on. Since the SVM does not compute likelihood estimates directly, a costly five-fold cross-validation is expected (Ray, 2017).

Table 1: Features used in classification of emails.

Feature	Description
F0	This is a binary feature. If there is HTML code contained in the email,
F1	If the number of images used as links exceeds two,
F2	Availability of more than three domains in an e-mail
F3	More than three embedded links in an e-mail.
F4	This feature has a value of 1 if the message contains HTML code in the form > tag.
F5	If the "From" domain is unequal to the "ReplyTo" domain.
F6	Supposing the size of the message is less than 25 KB, this feature has a value of 1.
F7	If the message contains java script code, this feature has a value of 1.
F8	If there is a mismatch between the target and displayed text of URLs in the email
F9	If the message contains the words "click here," "click here," "here," or "login" in the text of links.
F10	If there are more than three dots in the domain
F11	If the message has the @ symbol in the URL.
F12	If the message's URL has a port value apart from ports 80 and 443
F13	If the domain of an embedded links in the body of HTML is not equal to the senders's domain
F14	If https:// is used instead of http:// to trick the user into thinking it is a real URL supported by Secure Socket Layer (SSL).
F15	The presence of hexadecimal numeric representation in the URL of an e-mail.
F16	Categorization of an e-mail as spam by SpamAssassin3.2.3.5 Win32

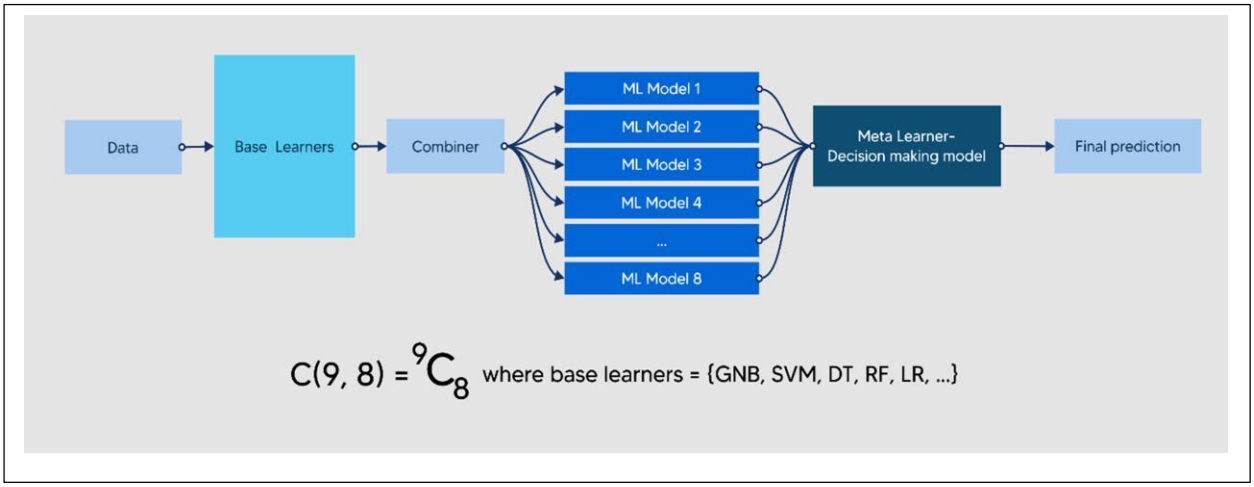

Figure 1. Architecture of the proposed ensemble.

2) Decision tree (DT)

This classifier requires little data structuring, simple to understand and visualize, and can handle both categorical and numerical data. A decision tree produces a collection of rules that can be used to identify data when given a set of attributes and a class. Despite their benefits, Decision Trees are inherently inconsistent since minor changes in data will result in an entirely different tree being created.

The entropy of a given information source x, H(x) is defined as follows:

$$H(x) = \sum_{xeX} p(x) \log p\ (x) \qquad (7)$$

where p(x) is the probability of occurrence of x (Azeez, *et al*., 2021).

NB: A decision tree can contain categorical (YES/NO) as well as numerical data.

3) Random forest (RF)

This is a meta-estimator that uses an average to maximize the model's predictive accuracy while avoiding over-fitting by fitting several decision trees on various sub-samples of datasets. In most cases, reducing over-fitting and using a random forest classifier is more effective than making a decision. This algorithm falls short by being difficult to implement, slow in real-time prediction and being overall a complex algorithm (Azeez *et al*, 2021b).

The mathematics behind the Random Forest:

Regression problems

The mean square error (MSE) is used to decide how the data splits from each node by using the Random Forest Algorithm to solve regression problems.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2 \qquad (8)$$

where N = number of data points

fi = model's return value and yi = data point's actual value

The distance between each node and the expected real value is calculated using this formula, assisting you in determining which branch is the best choice for your tree. In

this case, fi is the value returned by the decision tree, and yi is the value of the data point being evaluated at a given node.

$$Gini = 1 - \sum_{i=1}^{c}(P_i)^2 \quad (9)$$

where *pi* = relative frequency of observed classes
c = number of classes

This algorithm uses class and likelihood to calculate the Gini of each branch on a node, deciding which branch is more likely to occur. Here, pi denotes the relative frequency of the class observed in the dataset, and c denotes the number of classes. Entropy can also be used to decide how nodes in a decision tree branch.

$$Entropy = \sum_{i=1}^{c} -P_i * log_2\,(p_i) \quad (10)$$

where *pi* = relative frequency of observed classes
c = number of classes

4) Logistic regression

This classifier is useful for determining how many independent variables interact with a single outcome variable. Using a logistic equation, logistic regression predicts the odds representing the potential outcomes of a single trial.

$$log\left[\frac{p}{1-p}\right] = \beta_o + \beta(Age) \quad (11)$$

The odd ratio is described by (p/1-p). Whenever the log of odd ratio is positive, the likelihood of success is still greater than 50% (Plonus, 2020).

5) K-Nearest Neighbours (KNN)

This is a robust algorithm for noisy training data that is also useful when the data is massive. When calculating the distance between each instance of all samples in the training dataset, this classifier has a high computing cost. The probability in which points share the greatest probabilities are used to classify them. The distance function can be Minkowski, Euclidean, or the Hamming distance.

The Minkowski distance defines the distance between two points in normed vector space and is a generalization of the Euclidean distance (N-dimensional real space).
Consider the following two points: P1 and P2, the first and second positions, respectively.

P1: (X1, X2... XN)
P2: (Y1, Y2... YN)
Where
P1 = set of first positions
P2 = set of second positions

The Minkowski interval between positions P1 and P2 is then calculated as follows:
The distance equals Euclidean distance when p=2.
In order to visualize this formula:

$$\sqrt[p]{(x1-y1)^p + (x2-y2)^p + \cdots + (xN-yN)^p} \quad (12)$$

6) AdaBoost classifier

AdaBoost, also known as Adaptive Boosting, is an ensemble classifier that comprises several classifier algorithms, with its outcome being the combined output of the other classifier algorithms. The classifier combines weak classifier algorithms (in this example, decision trees) to generate a strong classifier.

A boosted classifier is a classifier in the form:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \quad (13)$$

f_t is considered a very weak learner that obtains x as input. It further generates an output hypothesis, $f(x_i)$ for the sample in the training set. A weak learner is picked and allotted a coefficient α_t for an iteration t such that the total training error E_t of the cumulative t-stage boost classifier is reduced.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_i\, h(x_i)] \quad (14)$$

Here $F_{t-1}(x)$ is considered as the boosted, $E(F)$ is known as error function and $f_t(x) = \alpha_t h(x)$ is the weak learner that is considered for addition to the final classifier.

7) Gradient Boosting Classifier

This is a collection of machine learning algorithms that combine several weaker models into a powerful huge one with highly predictive output. Models of this type are popular due to their ability to accurately classify datasets. Gradient boosting classifiers uses decision trees in model building. Predictions are carried out in steps, which are: Gathering and Analysis of data, Odds and probability Calculation, Residual Calculation, Building of Decision Trees, Calculating the Output value, Calculating probabilities based on new values (Azeez et. al., 2021a).

8) Extra Trees Classifier

A classifier similar to the Random Forest classifier, it is also known as the Extremely Randomized Trees. Essentially, it's built on decision trees. Extra Trees Classifier, and like the Random Forest, randomly selects some decisions and subsets of data to prevent data overlearning and overfitting. Extra Trees builds many trees and divides nodes using random subsets of attributes, but with two significant differences: it performs no bootstrap observations and nodes are divided into randomized splits rather than optimum splits.

<u>*Combination of variation*</u>: In the course of reducing the number of computations to be done with resource constraints in place, a total of 8 combinations out of 9 selected base learners are used in the stacking ensemble.

$$^{n}C_r \; where\; n\; is\; the\; total\; list\; of\; base\; learners$$

$$n = [GNB, KNN, DT, RF, SVM, LR, SVM, ABC, GBC, ETC]$$

where *n* is the collection of base learners and *r* is a combination selected from *n*. After selection, these models are passed to an ensemble for stacking and fitting.

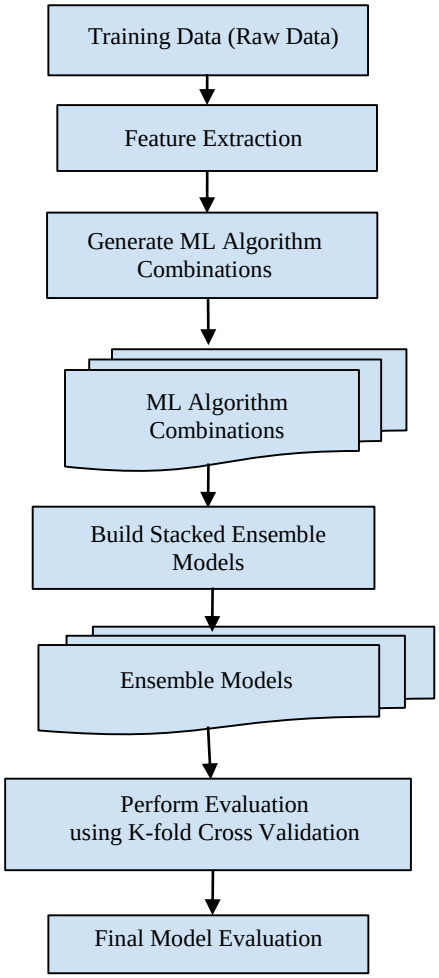

Figure 2: Flowchart of the proposed ensemble model.

Table 3: Keys and the actual names of the base learners represented.

Key	Base Learner Name
LR	Logistic Regression
KNN	K Nearest Neighbour
DT	Decision Tree
SVM	Support Vector Machine
BAYES	Naïve Bayes
RF	Random Forest Classifier
GBC	Gradient Boosting Classifier
ABC	AdaBoost Classifier
ETC	Extra Trees Classifier

D. Model Development

This phase constitutes the development of the stacking ensemble, which has 2 levels (level 0 & 1), with level 0 holding a stack of all the base learners and level one, the final estimator also known as the meta learner. Fundamentally, stacking may be thought of as assuming that a basic "wisdom of crowds" (e.g. averaging) is acceptable but not optimum and that better results can be obtained if we can identify and assign greater weight to experts in the crowd. The experts and less-experienced experts are determined based on their ability in novel settings, such as out-of-sample data.

Therefore, a weighted approach is used in the evaluation of the base learners, a weighted average employs a wide range of model types as contributing members. As a result, a weighted average implies that some contributing individuals are better than others and weights model contributions appropriately. (Brownlee, 2021)

On level 0, the base learners consist of selections from the members Naïve Bayes, K Nearest Neighbour, Decision Tree, Random Forest, Logistic Regression, Support Vector Machine, AdaBoost Classifier, Gradient Boosting Classifier, and Extra Trees Classifier (Azeez et. al., 2020).

Table 2: Performance metrics.

S/N	Technique	Description	Formula
1.	Accuracy	The proportion of correctly detected pharming e-mails in the dataset to the overall number of e-mails in the dataset.	$Acc = \frac{tp + tn}{tp + fp + tn + fn}$
2.	TP Rate	measures the proportion of pharming e-mails that are correctly identified	$TPR = \frac{tp}{tp + fn}$
3.	TN Rate	measures the proportion of ham e-mails that are correctly identified.	$TNR = \frac{tn}{fp + tn}$
4.	Precision	Proportion of correctly identified pharming e-mails over all identified pharming	$p = \frac{tp}{tp + fp}$
5.	Recall	Proportion of positive examples that were classified correctly	$r = \frac{tp}{tp + tn}$
6.	Confusion Matrix	The summaries of the Positively and Negatively expected effects are set out in a table to visualize the ensembles' success metric. A heat map is used to facilitate the visual analysis of the data.	

On level 1 The Logistic Regression model is used as the meta-model due to its optimality in performing classification tasks (predicting a class label). A 2 phase split and evaluation of each dataset are performed, where the first phase is an 8:2 split into training and test datasets, with the training set fed into the base learners and their resulting predictions into the meta estimator. Then, the preceding phase performs a 10-fold cross-validation with 3 iterations on the training set and passed to the final ensemble model for fitting and evaluation. Each individual base learner and their resulting ensembles were fitted and tested against two datasets to measure their performances with the stated metrics.

E. Performance Evaluation

For performance comparison, the results from all evaluated combinations were compared with the result from each base learner model. The metrics of importance were measured and computed, which include Accuracy (Initial Split Evaluation), Mean Accuracy (Cross Validation Evaluation), Precision, Recall, Log loss False Positive, True Positive, True Negative, False Negative, and F-score.

IV. RESULTS AND DISCUSSION

Machine learning techniques were used to identify pharming emails based on well-researched characteristics, while the algorithms' effectiveness was evaluated and empirically proven. The Random Forest Classifier was found to be the best base classifier, with an average of 0.935 across datasets 1 and 2, but the Stacking Ensemble of all base learners outperformed it, with an average of 0.938. The proposed Ensemble combination models performed well on the available datasets, with the best combination: (LR|KNN|DT|SVM|GBC|ABC|ETC|RF) having an average accuracy of 0.943 when the Naive Bayes classifier was not used (the worst-performing base classifier).

Table 4 shows the resulting analysis of the performance metric for each base learner and an ensemble featuring all of them. The evaluation metrics include but are not limited to accuracy, mean accuracy, log loss, precision, recall, and F1 score.

The base learners are KNearest Neighbors, Decision Tree, Random Forest, Gaussian Naive Bayes, Logistic Regression, Support Vector Machine, Adaptive Boosting, Gradient Boosting, and Extra Trees Classifier.

From the results shown in Table 4, after performing a 10-fold cross validation and split test validation on the resulting model, the resulting accuracies shows the Random Forest classifier as the most accurate, closely followed by the Adaptive Boost classifier. The Bayes algorithm was found to be the least accurate and least sensitive, showcasing the lowest recall score.

The Ensemble model consisting of all base learners as members outperformed its constituent learners with better F1 score and overall accuracy than each individual base models. Table 5 shows resulting analysis of the performance metric for each base learner and an ensemble featuring all of them with regards to dataset 2. From the results shown in Table 5, the resulting accuracies shows again the Random Forest classifier as the most accurate, closely followed by the Gradient Boosting classifier on this dataset. The Bayes algorithm was again found to be the least accurate and least sensitive, showcasing the lowest recall score. The Ensemble model outperformed its constituent learners in terms of accuracy.

Table 6 highlights results across 9 combinations of the base learners in ensembles, with evenly distributed scores in Precision, Log loss, Recall, and F1 score for most combinations. The notable exception is the ensemble with the Bayes base learner as a member, this ensemble combination performed the overall best across all metrics in dataset 1.

Table 7 provides a slightly different representation of metrics evaluated from dataset 2 when compared to results from dataset 1. Performance across all metrics are the same with the exception of the mean accuracy after the 10-fold cross validation is performed on each ensemble combination. A notable observation is the ensemble without the inclusion of the Naïve Bayes base learner compared favorably with other combinations in mean accuracy of 0.941275 out of all other combinations. This further buttress the conclusion that the Bayes model is a bad estimator and should not be considered when building an ensemble using the stacking method. In an attempt to give a professional medium for viewing code repository and specifically, to give room for reproducibility, the link is provided under '*Note*'.

Table 4: Results obtained by individual base classifier [dataset 1].

Models	Accuracy	Mean_Accuracy	Log loss	Precision	Recall	F1
LR	0.923228	0.929961	2.651601	0.931578	0.923228	0.922218
KNN	0.899606	0.925528	3.467477	0.913727	0.899606	0.897602
DT	0.925197	0.932420	2.583609	0.934136	0.925197	0.924173
SVM	0.923228	0.931105	2.651601	0.931578	0.923228	0.922218
BAYES	0.474409	0.476215	18.15367	0.721093	0.474409	0.334928
GBC	0.925197	0.934059	2.583609	0.934136	0.925197	0.924173
ABC	0.923228	0.933896	2.651601	0.931578	0.923228	0.922218
ETC	0.925197	0.931763	2.583609	0.934136	0.925197	0.924173
RF	0.925197	0.932747	2.583609	0.934136	0.925197	0.924173
ensemble(stacking)	0.925197	0.933566	2.583611	0.933125	0.925197	0.924250

Table 5: Results obtained by individual base classifier [dataset 2].

Models	Accuracy	Mean_Accuracy	Log loss	Precision	Recall	F1
LR	0.946850	0.936683	1.835724	0.950788	0.946850	0.946434
KNN	0.946850	0.934060	1.835724	0.950788	0.946850	0.946434
DT	0.940945	0.937339	2.039695	0.945104	0.940945	0.940457
SVM	0.940945	0.934221	2.039698	0.943816	0.940945	0.940554
BAYES	0.462598	0.518910	18.56162	0.617398	0.462598	0.321883
GBC	0.948819	0.933565	1.767733	0.953153	0.948819	0.948396
ABC	0.948819	0.933237	1.767733	0.953153	0.948819	0.948396
ETC	0.942913	0.937504	1.971705	0.946768	0.942913	0.942466
RF	0.942913	0.939472	1.971705	0.946768	0.942913	0.942466
ensemble(stacking)	0.944882	0.940945	1.903715	0.948444	0.944882	0.944472

Table 6: Results obtained by ensemble combinations [dataset 1].

Models	Accuracy	Mean_Accuracy	Log loss	Precision	Recall	F1
LR|KNN|DT|SVM|BAYES|GBC|ABC|ETC	0.925197	0.933402	2.583611	0.933125	0.925197	0.924250
LR|KNN|DT|SVM|BAYES|GBC|ABC|RF	0.925197	0.933566	2.583611	0.933125	0.925197	0.924250
LR|KNN|DT|SVM|BAYES|GBC|ETC|RF	0.925197	0.933402	2.583611	0.933125	0.925197	0.924250
LR|KNN|DT|SVM|BAYES|ABC|ETC|RF	0.925197	0.933074	2.583611	0.933125	0.925197	0.924250
LR|KNN|DT|SVM|GBC|ABC|ETC|RF	0.927165	0.933074	2.515620	0.935667	0.927165	0.926207
LR|KNN|DT|BAYES|GBC|ABC|ETC|RF	0.925197	0.933241	2.583611	0.933125	0.925197	0.924250
LR|KNN|SVM|BAYES|GBC|ABC|ETC|RF	0.925197	0.933238	2.583611	0.933125	0.925197	0.924250
LR|DT|SVM|BAYES|GBC|ABC|ETC|RF	0.927165	0.933731	2.515620	0.935667	0.927165	0.926207
KNN|DT|SVM|BAYES|GBC|ABC|ETC|RF	0.927165	0.933238	2.515620	0.935667	0.927165	0.926207

Table 7: Results obtained by ensemble combinations [dataset 2].

Models	Accuracy	Mean_Accuracy	Log loss	Precision	Recall	F1
LR|KNN|DT|SVM|BAYES|GBC|ABC|ETC	0.944882	0.940781	1.903715	0.948444	0.944882	0.944472
LR|KNN|DT|SVM|BAYES|GBC|ABC|RF	0.944882	0.940617	1.903715	0.948444	0.944882	0.944472
LR|KNN|DT|SVM|BAYES|GBC|ETC|RF	0.944882	0.940945	1.903715	0.948444	0.944882	0.944472
LR|KNN|DT|SVM|BAYES|ABC|ETC|RF	0.944882	0.940781	1.903715	0.948444	0.944882	0.944472
LR|KNN|DT|SVM|GBC|ABC|ETC|RF	0.944882	0.941275	1.903715	0.948444	0.944882	0.944472
LR|KNN|DT|BAYES|GBC|ABC|ETC|RF	0.944882	0.940945	1.903715	0.948444	0.944882	0.944472
LR|KNN|SVM|BAYES|GBC|ABC|ETC|RF	0.944882	0.941110	1.903715	0.948444	0.944882	0.944472
LR|DT|SVM|BAYES|GBC|ABC|ETC|RF	0.944882	0.940945	1.903715	0.948444	0.944882	0.944472
KNN|DT|SVM|BAYES|GBC|ABC|ETC|RF	0.944882	0.940781	1.903715	0.948444	0.944882	0.944472

Note

https://github.com/soldierlytomcat/Ensemble-Learning-Project

V. CONCLUSION

Since the advent of email as a means of communication, cybercriminals have heavily exploited the act of pharming. It has cost users a lot of money and resources. When using such means of communications, individuals and even organizations are no longer safe. As usage of emails in communication progresses alongside advances in technology, identification of legitimate emails without the help of sufficiently advanced machines gets harder.

Hence, diverse approaches to using machine learning techniques for pharming e-mail detection and classification is crucial. Given that supervised learning was the methodology used, ensemble models were tested with Ham and Phish, and their performance metrics recorded. In future works, this model can be integrated with a server-based email pharming detector, where harmful emails can be easily identified and prevented in real-time.

With a variety of public corpora hosts email datasets a verified collection of a larger sample of pharming emails are quite difficult to obtain without selective and meticulous archiving over a period of time. A better approach for gathering pharmed emails would come in handy to properly identify features that will contribute to better identification of pharming.

REFERENCES

Alkhalil, Z.; C. Hewage; L. Nawaf and I. Khan. (2021). Phishing Attacks: A Recent Comprehensive Study and a New Anatomy. Frontiers in Computer Science 3(6): 563060.

All Answers Ltd. (2018). Spam Filtering Software Using JAVA. Available online at: https://ukdiss.com/examples/spam-filtering-software.php?vref=1. Accessed on March 13, 2022.

Al-Saaidah, S. A. (2017). Detecting Phishing Emails Using Machine Learning Techniques. Available online at: https://www.meu.edu.jo/libraryTheses/590422b4d5dd8_1.pdf . Accessed on March 17, 2022.

Azeez, N.A., O.E. Odufuwa; S. Misra; J. Oluranti; and R. Damaševičius. (2021). Windows PE Malware Detection Using Ensemble Learning. Informatics. 2021; 8(1):10. https://doi.org/10.3390/informatics8010010

Azeez, N.A.; A.M. Ihotu; and S. Misra. (2021b). Adopting Automated White-List Approach for detecting Phishing Attacks, Journal of Computers & Security 108 (2021) 102328: 1-18.

Azeez, N.A.; B.B. Salaudeen; S. Misra; R. Damasevicius and R. Maskeliunas. (2019). Identifying Phishing Attacks in Communication Networks using URL Consistency Features, International Journal of Electronic Security and Digital Forensics (InderScience).
https://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijesdf

Azeez, N.A.; O.E. Adio; A.W. Yekinni and C.J. Onyema. (2020). Evaluation of Machine Learning Algorithms for Filtering and Isolating Spammed Messages. FUTA Journal of Research in Sciences, 16(1): 26-38.

Azeez, N.A.; O.E. Odufuwa; S. Misra; J. Oluranti; R. Damaševičius. (2021). Windows PE Malware Detection Using Ensemble Learning. Informatics 2021, 8, 10. https://doi.org/10.3390/informatics8010010.

Azeez, N.A; S.O. Idiakose; C.J. Onyema; and C.V. Vyver (2021a). Cyberbullying Detection in Social Networks: Artificial Intelligence Approach. Journal of Cyber Security and Mobility, 10 (4): 1–30. doi: 10.13052/jcsm2245-1439.1046.

Azeez, N.A; T.J. Ayemobola; S. Misra; R. Maskeliūnas R. Damaševičius. (2019). Network Intrusion Detection with a Hashing Based Apriori Algorithm Using Hadoop MapReduce. Computers. 2019; 8(4):86.

Brownlee, J. (2021). Essence of Stacking Ensemble for Machine Learning. Available online at: https://machinelearningmastery.com/essence-of-stacking-ensembles-for-machine-learning/. Accessed on January 24, 2022.

Chandrasekaran, M.; K. Narayanan, and S. Upadhyaya. (2006). Phishing E-mail Detection Based on Structural Properties. New York, s.n., 2-8.

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

Dong, X; Z. Wu; W. Cao and Q. Ma. (2019). A survey on ensemble learning. Frontiers of Computer Science (print) 14(5): 241–258.

Gansterer, W. and Pölz, D. (2009). E-Mail Classification for Phishing Defense. Toulouse, s.n., 449-460.

Gonzalez, C.; N. Ben-Asher; A. Oltramari; and C. Lebiere, (2014). Understanding Cyber Situational Awareness in a Cyber Security Game involving Recommendation. Cognition and Technology, 93-117.

Hadnagy, C. and Fincher, M. (2015). Phishing Dark Waters; The Offensive and Defensive Sides of Malicious E-mails. s.l.:John Wiley & Sons, Inc.

Human Factor Report. (2019). The Human Factor Report Proofpoint. Available online at: https://www.proofpoint.com/us/resources/threat-reports/human-factor.html. Accessed on February 15, 2022.

Jameel, G.Z. and George, K. (2013). Detection of Phishing Emails using Feed Forward Neural Network. International Journal of Computer Applications, 7(77): 10-15.

Jameel, G.Z. and George, K. (2013). Detection Phishing Emails Using Features Decisive Values. International Journal of Advanced Research in Computer Science and Software Engineering, 3(7): 257-262.

Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity. Journal of Computer and System Sciences, 80(5): 973-993.

Oladimeji, O. O. (2019). Text Analysis and Machine Learning Approach to Phished Email Detection. International Journal of Computer Applications, 182(36).

Oña, D. Zapata, L.Fuertes, W. Rodríguez, G. Benavides, E. and Toulkeridis, T. (2019). Phishing Attacks: Detecting and Preventing Infected E-mails Using Machine Learning Methods, 2019 3rd Cyber Security in Networking Conference (CSNet), 2019, 161-163, doi: 10.1109/CSNet47905.2019.9108961.

Paliath, S.; Qbeitah, M. A. and Aldwairi, M. (2020). PhishOut: Effective Phishing Detection Using Selected Features, 2020 27th International Conference on Telecommunications (ICT), 2020; 1-5, doi: 10.1109/ICT49546.2020.9239589.

Plonus, M. (2020). 9 - Digital systems. In: M. Plonus, ed. Electronics and Communications for Scientists and Engineers (Second Edition). 2nd ed. s.l.:Butterworth-Heinemann, 355-480.

Protti, D. J. (2003). Public Health. Encyclopedia of Information Systems.

Rashid, F. Y. (2020). 8 types of phishing attacks and how to identify them. Available online at: https://www.csoonline.com/article/3234716/8-types-of-phishing-attacks-and-how-to-identify-them.html. Accessed on September, 2021.

Rawal, S.; B. Rawal; A. Shaheen; and S. Malik. (2017). Phishing Detection in E-mails using Machine Learning. International Journal of Applied Information Systems, 12(7): 21-24.

Ray, S. (2017). Commonly used Machine Learning Algorithms | Data Science.

Sampson, M. (2015). Electronic Mail. International Encyclopedia of the Social & Behavioral Sciences (Second Edition).
Available online at:
https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/. Accessed on November, 2021.

Singh, K.; Aggarwal, P.; Rajivan, P. and Gonzalez, C. (2020). What makes phishing emails hard for humans to detect? in the Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 64(1): 431-435.

Soon, G.K.; On, C.K. Rusli, N.M. Fun, T.S. Alfred, R. and Guan, T.T. (2019). Comparison of simple feedforward neural network, recurrent neural network and ensemble neural networks in phishing detection. Journal of Physics: Conference Series, 1502, International Conference on Telecommunication, Electronic and Computer Engineering 2019 22-24 October 2019, Melaka, Malaysia.