PhisherCop: Developing an NLP-Based Automated Tool for Phishing Detection

Naheem Noah, Abebe Tayachew, Stuart Ryan, and Sanchari Das

{Naheem.Noah, Abebe.Tayachew, Stuart.Ryan, and Sanchari.Das}@du.edu

Phishing poses a major security risk to organizations and individuals leading to loss of billions of dollars yearly. While risk communication serves as a tool to mitigate phishing attempts, it is imperative to create automated phishing detection tools. Numerous Natural Language Processing (NLP) approaches have been deployed to tackle phishing. However, phishing attempts continue to increase exponentially, which reiterates the need for more effective approach. To address this, we have developed an anti-phishing tool called; PhisherCop. PhisherCop is built upon Stochastic Gradient Descent classifier (SGD) and Support Vector Classifier (SVC) which showed an average accuracy of 96%, performing better than six other popular classifiers including: Decision Tree, Logistics Regression, Random Forest, Gradient Boosting Classifier, K-Nearest Neighbors and Multinomial Naive Bayes. Our tool is significant in distinguishing between phishing and legitimate content both over emails and text messages.

INTRODUCTION

Phishing is an act propagated by cybercriminals whereby they send malicious contents to individuals with the aim of tricking them to fall for a scam via several sources including emails and text messages (Das et al., 2019; Dhamija et al., 2006). Phishing attackers forward socially-engineered messages to users' accounts in order to persuade them to reveal personal information which may be used to gain unauthorized access (Akinyelu & Adewumi, 2014). Phishing through SMS or text messages, otherwise called 'Smishing' (Mishra & Soni, 2019) is furthermore concerning because the scale and capacity of mobile phone interactions (Mishra & Soni, 2021). However, most approaches used by various email or SMS filters today seem to be less effective against modern phishing attacks (Verma et al., 2012), the rate of adoption is low (Abbasi et al., 2012), and the proliferation of phishing attacks is steadily increasing (Milletary & Center, 2005).

To this aid, we implemented NLP (Natural Language Processing) techniques to detect phishing in email and text messages by developing PhisherCop. We collected 501 spam and 2551 ham email corpus from Kaggle [1], and 5,574 SMS messages which include legitimate or spam from Grumbletext website [2] and performed feature extraction and vectorization. Thereafter, we implemented classification employing eight machine learning classifiers namely; Stochastic Gradient Descent classifier (SGD), Support Vector Classifier (SVC), Decision Tree Classifier (DT), Logistics Regression Classifier (LR), Random Forest Classifier (RF), Gradient Boosting Classifier (GBC), K Nearest Neighbors Classifier (KNN) and Multinomial Naive Bayes Classifier (MNB). In addition to providing a repository of phishing messages, we show that SGD and SVC showed a higher accuracy in phishing detection at an average of 96%.

[1]https://www.kaggle.com/veleon/ham-and-spam-dataset

[2]http://www.grumbletext.co.uk/

RELATED WORK

Anti-phishing tools are designed to detect phishing contents (Das et al., 2022; Shankar et al., 2019). These tools make use of different approaches which includes whitelisting and blacklisting of known legitimate and phishing URLs respectively or through the heuristics in URL and message content (Das et al., 2020; Zeydan et al., 2014). These tools are implemented at different application levels; client-side or sever-side and integrated into browsers or deployed on web portal (Khonji et al., 2013). The introduction of NLP in the design of these tool has been productive in the analysis of text and detection of statements indicative of phishing (Naheem et al., 2022; Peng et al., 2018).

The tool designed by May et al., allows users to input their email content which is parsed into the SVM classifier to predict if the email is legitimate or illegitimate. They combined URL-based, content-based and behavior-based features for their classification.They analyzed features like domain sender, blacklist words in subject and content, IP address in URL, dot in URL, symbol in URL, unique sender, unique domain, inconsistent hyperlink and return path (Form et al., 2015). MobiFish, a mobile-based anti-phishing tool utilized Optical Character Recognition (OCR) for detecting phishing web pages. The tool scans the URL before a browser accesses it to detect if the domain name is an IP address. Eventually the tool notifies the user of the possibility of a phishing attack, highlighting the suspected web page (Wu et al., 2014). However, this tool is limited to mobile devices and no classification comparison was conducted where our work adds to the literature.

PHISHERCOP: IMPLEMENTATION DETAILS

Our proposed system PhisherCop is designed to identify emails and SMS as legitimate or phished. The user provides the tool with the email or SMS content and the system parses the

content into a number of features for classification.

Data Collection and Processing

To start developing the tool, we collected the Email Spam and Ham Corpus by Spam Assassin [3] available at Kaggle [4]. The corpus contains 2551 ham and 501 spam emails. For the SMS, we collected the SMS Spam Collection public corpus available in Kaggle [5]. The corpus contains 5,574 SMS messages in English tagged as legitimate or spam. These messages are collected from Grumbletext website [6], NUS SMS Corpus [7], Caroline's Tag PhD Thesis [8] and, SMS Spam Corpus v.0.1 Big [9] Thereafter, we removed stop words using the set of 127 English stop-words available in the NLTK library such as "the", "a", "an", "in" [10], emojis, emoticons, punctuation marks, and HTML tags. Tokenization involves the splitting of text into series of words in order to aid word processing in NLP. Each of this smaller words are called Tokens. We implemented stemming using Porter Stemmer which helps to map related words to the same stem. For instance 'talking' is stemmed to its root form, 'talk'. To transform tokinized words into features, we introduced Term Frequency Inverse Document Frequency (TFIDF) which down weight words that frequently occur across the document. Term Frequency-Inverse Document Frequency(TF-IDF) is measured by the division of the number of times a word appears against the total number of words in the document. We utilized the scikit-learn library to implement TfidfVectorizer which combines both CountVectorizer and TfidfTransformer. TF-IDF is computed for a term t of a document d in a document set as follows (*FeatureExtraction*, n.d.)

$$Tfidf(t,d) = tf(t,d) * idf(t)$$

and idf(t) is computed as:

$$idf(t) = log[n/df(t)] + 1$$

Where;
n = total number of documents in the document set
df(t) = document frequency of t i.e. the number of documents in the document set that contain the term t.

Comparison of Machine Learning Classifiers

SGD Classifier: According to scikit-learn [11], Stochastic Gradient Descent Classifier is most efficient in the fitting of linear classifiers. It is used to find coefficient that ensures the minimizing of a cost function. Even though SGD is efficient, it is highly sensitive to feature scaling and also requires regularization parameter and number of iterations (Kabir et al., 2015).

Decision Tree Classifier: Unlike SGD Classifiers, Decision Trees are non-parametric. The model understands and learns certain decision rules that are deducted from the data. While the decision nodes introduces the partitioning, the leaf nodes handle the predictions (Swain & Hauska, 1977).

Logistics Regression Classifier: Logistics regression is a regression model that predicts that a condition lies between 0 and 1. Logistics regression performs well on numerical data. Generally, the group share the same variance across, and the data is modeled using a sigmoid function (Afrianto & Wasesa, 2020).

SVC Classifier: Support Vector Classification is one of the approach to supervised learning, mostly effective when the dimension is greater than the number of the sample. The classifier is very memory efficient. However, in cases where we have a higher number of features compared to the sample, over regularization is important so as to avoid over-fitting (Dogan et al., 2016).

Random Forest Classifier: Random forest is an ensemble of decision trees that models the bagging method where the increase in the number of learning models results in a better result. The prediction is taken by calculating the mean of the output accrued from the various decision trees (Pal, 2005).

Gradient Boosting Classifier: Gradient Boosting Classifiers supports both binary classification and multi-class classification problems, as well as regression. It creates a strong predictive model by combining several weak learning models. The goal of the model is to minimize the loss function. Gradient boosting has three components; the loss function which helps to estimate the performance of the model, the weak learner that classifies the data and the additive model which ensures the sequential addition of the weak learners (Iqbal & Barua, 2019).

KNN Classifier: The k-nearest neighbors (KNN) is a lazy and non-parametric supervised machine learning algorithm which leverages voting mechanism to determine the class in which an unseen observation belongs to. The class that has the majority vote becomes the class for that particular data point. KNN is utilized for classification and regression, however, the run-time computing cost is high for large datasets (Feng et al., 2017).

Multinomial Naive Bayes: Multinomial Naive Bayes classifier is most suitable for classifications involving discrete features. The algorithm based on the Bayes Theorem is mostly used in NLP to calculate and predict the tag for a given sample. It provides the highest probability as output (Jiang et al., 2016).

[3]https://spamassassin.apache.org/old/publiccorpus/
[4]https://www.kaggle.com/veleon/ham-and-spam-dataset
[5]https://www.kaggle.com/uciml/SMS-spam-collection-dataset
[6]http://www.grumbletext.co.uk/
[7]http://www.comp.nus.edu.sg/ rpnlpir/downloads/corpora/SMSCorpus/
[8]http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf
[9]http://www.esp.uem.es/jmgomez/SMSspamcorpus/
[10]https://www.nltk.org/
[11]https://scikit-learn.org/stable/modules/sgd.html

The Unique Approach

We have implemented a unique approach in our model classification. Our approach includes: *1. Reviewed existing phishing tools and selected eight most prominent classifiers that have been previously implemented. 2. Implemented the classifiers on our training and test data and compared the Accuracy score of these classifiers separately on SMS content, email address, email body and subject line. 3. Designed a user-friendly web interface which collects any or all of the content, subject line and email address of a message. With the classifiers with the highest level of accuracy integrated to the portal, the legitimacy of the content is determined.*

SMS Classification Comparison

SMS only contains text. We collected our SMS ham and spam data, cleaned and pre-processed the data, implemeted tokenization and feature vectorization using TFIDF and performed model and hyperparameter tuning with GridSearch [12]. GridSearch builds model for combinations of hyperparameters specified and evaluates the model based on each combination. The result is a model with the best accuracy score as well as the best hyperparameters (Syarif et al., 2016). This procedure was repeated for the eight classifiers. While all the classifiers performed above 85% of accuracy, SGD shows the highest level of accuracy at 98.4%, closely followed by SVC at 98.1%. The lowest accuracy was found in Random Forest with an accuracy of 88.5%.

Email Classification Comparison

Unlike SMS, emails have address, subject line, and body. We performed analysis on each of this section and applied the classifiers to them. Our system allows a user to submit any of the address, subject line or body of the email. It then generates the accuracy score of any or all of the information passed. In the case where the user submits all the three inputs, the analysis is conducted and an average score is calculated. As done in the SMS classification, the email address, subject line and body are cleaned and pre-processed. Afterwords vectorization and hyperparamter was performed with TFIDF and GridSearch respectively.

Email Address Classification: In the training dataset, SGD classifier performed best with an accuracy of 93.5% and it is followed by KNN with 92.9%. Random Forest showed the lowest level of accuracy in predicting spam email addresses with 84.1%. Logistic Regression and Decision Tree are more closer with 85.6% and 85.4% respectively.

Email Subject Classification: SVC showed an accuracy of 93.7% while SGD showed an accuracy of 93.4% in the subject classification. This is followed by Gradient Boosting and KNN at 89.6% and 89.4% respectively. Random Forest exhibits the lowest accuracy level at 85.3%

Email Body Classification: The highest level of accuracy was seen in the email body classification. SGD classifier and SVC ranked higher in accuracy with 98.7% and 98.3% respectively. This was followed by KNN and Gradient Boosting with 98.0% and 96.5% respectively. While Random Forest (88.8%) increased in accuracy compared to email address and email body, it remained the lowest.

PHISHERCOP OVERVIEW AND PERFORMANCE

PhisherCop was implemented by using Python Flask Framework [13]. Our tool's interface is user-friendly and provides fields for users to input the email address of the sender, email body and email subject or the SMS message as shown in figure 2. Thereafter, the legitimacy score is calculated. If the score is greater than 50%, the content is considered to be spam, otherwise it is declared "Not Phishing".

Web Interface

The web portal is designed with Hypertext Markup Language (HTML) [14] and Cascading Style Sheets (CSS) and styled with CSS [15]. The interface is designed with three fields and a submit button. The content field is mandatory and collects either the email body or the SMS message, however, the subject and the email address are optional, only to be inputted for email messages.

Input Analysis

We introduced the pickle module [16] to embed the classifiers with the highest accuracy in our web application. The module uses the dump function to embed or pickle a fitted machine learning classifier into a file or object which can then be loaded, unpickled or deserialized through the load function. The inputted content, email address and subject is parsed to the unpickled classifier for the classification in order to determine its legitimacy. For email addresses, the address is converted into lowercase and tokenized to form multiple tokens which is passed as a string into the classifier. For the optional subject line, the text is converted into lowercase, bad symbols compiled via regex is removed as well as stop words from the nltk corpus [17]. Also, we removed emojis and emoticons before the resulting words are tokenized and passed to the unpickled classifier for the subject line classification.

[12] https://scikit-learn.org/stable/modules/grid$_search.html$
[13] https://flask.palletsprojects.com/en/2.0.x/
[14] https://www.w3schools.com/html/
[15] https://www.w3schools.com/css/
[16] https://docs.python.org/3/library/pickle.html
[17] https://www.nltk.org/api/nltk.corpus.html

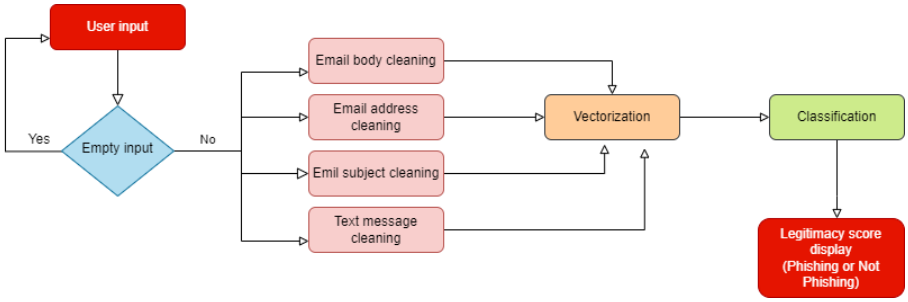

Figure 1: PhisherCop Architecture

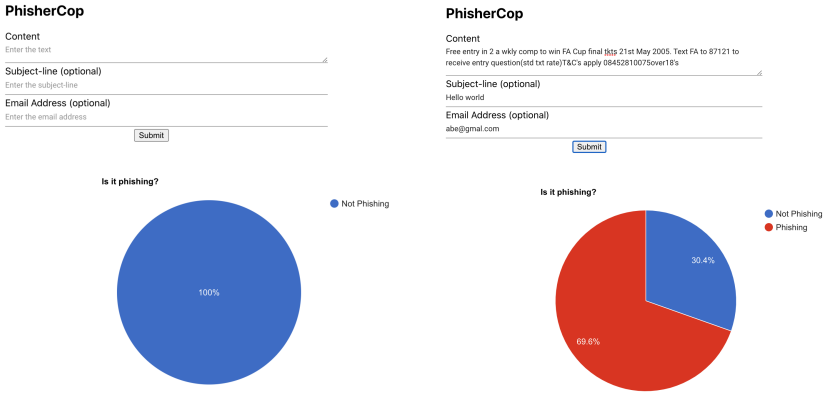

Figure 2: PhisherCop in Action

Tool Performance

We collected the Email Spam and Ham Corpus by Spam Assassin [18] available at Kaggle [19], and the SMS Spam Collection public corpus available at Kaggle [20]. Each dataset was divided into two parts, the training dataset and the test dataset. We split the datasets into 70% training and 30% testing. To evaluate the performance of the classifiers, we measured the accuracy. Accuracy helps to determine the fraction of correct prediction our model satisfies based on the inputted training data. Generally, the more our model generalizes to unseen data, the better the predictions and insights produced (Yin et al., 2019). We calculated the accuracy for binary classification in terms of positives and negatives highlighted below (Aggarwal et al., 2014):

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

Where;
TP = True Positives; this is the number of spam emails that have been correctly predicted by our classifier
TN = True Negatives; this is the number of ham emails that have been correctly predicted by our classifier
FP = False Positives; this is the number of spam emails that have been wrongly predicted by our classifier
FN = False Negatives; this is the number of ham emails that have been wrongly predicted by our classifier
The SMS content, email address, subject line and body were individually classified and the average accuracy score was displayed. Our experiment indicated that SGD better predicted SMS content with an accuracy score of 98.4% when fitted to the training data. This highlights that SGD performs well when used to predict the legitimacy of a SMS. Although (Shahi & Shakya, 2018), indicated that Neural Network better predicted SMS, they only compared with SVM and Decision Tree. While SGD was more accurate for us compared with SVM and Decision Tree, we did not compare with Neural Network.

For email classification, SGD showed a higher level of accuracy for email address and email body at an average accuracy of 93.5% and 98.7% respectively. This is synonymous to the work of (Elshoush & Dinar, 2019), where SGD accrued an accuracy of 98.1% when applied to a dataset containing 5728 emails, where 1368 are spam and 4360 are ham emails. SVC gave the highest accuracy for email subject lines at an average accuracy of 92%.

PhisherCop; uses SGD classifier to determine the legitimacy of SMS content or email body inputted into the content field of the interface. The resulting output for the user is a pie chart that shows the score of the information in terms of phishing and legitimate as shown in 2. If the score has less than 50% of "Not Phishing", the information passed is assumed to be phished or spam, however, if "Not Phishing" is greater than 51%, the information passed is legitimate. An average

[18]https://spamassassin.apache.org/old/publiccorpus/
[19]https://www.kaggle.com/veleon/ham-and-spam-dataset
[20]https://www.kaggle.com/uciml/SMS-spam-collection-dataset

score i.e 50% denotes that the classifier is unable to identify if the content is spam or legitimate. The model's performance in detecting spam information increases as it is exposed to more unseen data.

CONCLUSION

In this paper, we report on the development of an anti-phishing detection tool which we created, called PhisherCop. We implemented NLP and ML techniques to differentiate between legitimate and spam emails, and also legitimate SMS and spam SMS. We compared eight ML classifiers namely; SGD, SVC, DT, LR, RF, GB, KNN, and MNB Classifier. In our analysis, we realized SGD classifier performed very well on detecting spam SMS and spam email address, and spam email body. Our analysis found out that SVC gave higher accuracy for the detection of spam subject lines. Upon the completion of our comparison, we implemented PhisherCop; a web based tool that predicts the validity of an email (subject line, email address and body) and SMS through the Accuracy score. For PhisherCop, we leveraged the best performing classifiers we identified from our classification comparison.

LIMITATIONS AND FUTURE WORK

We implement NLP and ML techniques into the development of an anti-phishing tool called PhisherCop. To achieve this, we compared the accuracy of eight ML classifiers and integrated a combination of two of them; SGD and SVC which performed the best into our web based tool. In the future, we will include BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019) as a baseline for our ground truth evaluation. Additionally, we plan to extend this work by performing a usability study to understand the efficiency and effectiveness of the tool. Finally, other evaluation metrics such as Precision, Recall, Specificity and F1 score could be explored to determine the robustness of the model.

REFERENCES

Abbasi, A., Zahedi, F., & Chen, Y. (2012). Impact of anti-phishing tool performance on attack success rates. In *2012 ieee international conference on intelligence and security informatics* (pp. 12–17).

Afrianto, M. A., & Wasesa, M. (2020). Booking prediction models for peer-to-peer accommodation listings using logistics regression, decision tree, k-nearest neighbor, and random forest classifiers. *Journal of Information Systems Engineering and Business Intelligence*, *6*(2), 123–132.

Aggarwal, S., Kumar, V., & Sudarsan, S. (2014). Identification and detection of phishing emails using natural language processing techniques. In *Proceedings of the 7th international conference on security of information and networks* (pp. 217–222).

Akinyelu, A. A., & Adewumi, A. O. (2014). Classification of phishing email using random forest machine learning technique. *Journal of Applied Mathematics*, *2014*.

Das, S., Abbott, J., Gopavaram, S., Blythe, J., & Camp, L. J. (2020). User-centered risk communication for safer browsing. In *International conference on financial cryptography and data security* (pp. 18–35).

Das, S., Kim, A., Tingle, Z., & Nippert-Eng, C. (2019). All about phishing: Exploring user research through a systematic literature review. *arXiv preprint arXiv:1908.05897*.

Das, S., Nippert-Eng, C., & Camp, L. J. (2022). Evaluating user susceptibility to phishing attacks. *Information & Computer Security*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 581–590).

Dogan, Ü., Glasmachers, T., & Igel, C. (2016). A unified view on multi-class support vector classification. *J. Mach. Learn. Res.*, *17*(45), 1–32.

Elshoush, H. T., & Dinar, E. A. (2019). Using adaboost and stochastic gradient descent (sgd) algorithms with r and orange software for filtering e-mail spam. In *2019 11th computer science and electronic engineering (ceec)* (pp. 41–46).

Featureextraction. (n.d.). Retrieved from `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`

Feng, L., Liu, G., Luo, S., & Liu, S. (2017). A transferable framework: Classification and visualization of mooc discussion threads. In *International conference on neural information processing* (pp. 377–384).

Form, L. M., Chiew, K. L., Tiong, W. K., et al. (2015). Phishing email detection technique by using hybrid features. In *2015 9th international conference on it in asia (cita)* (pp. 1–5).

Iqbal, A., & Barua, K. (2019). A real-time emotion recognition from speech using gradient boosting. In *2019 international conference on electrical, computer and communication engineering (ecce)* (pp. 1–5).

Jiang, L., Wang, S., Li, C., & Zhang, L. (2016). Structure extended multinomial naive bayes. *Information Sciences*, *329*, 346–356.

Kabir, F., Siddique, S., Kotwal, M. R. A., & Huda, M. N. (2015). Bangla text document categorization using stochastic gradient descent (sgd) classifier. In *2015 international conference on cognitive computing and information processing (ccip)* (pp. 1–4).

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, *15*(4), 2091–2121.

Milletary, J., & Center, C. C. (2005). Technical trends in phishing attacks. *Retrieved December*, *1*(2007), 3–3.

Mishra, S., & Soni, D. (2019). Sms phishing and mitigation approaches. In *2019 twelfth international conference on contemporary computing (ic3)* (pp. 1–5).

Mishra, S., & Soni, D. (2021). Dsmishsms-a system to detect smishing sms. *Neural Computing and Applications*, 1–18.

Naheem, N., Abebe, T., Stuart, R., & Sanchari, D. (2022). Phishercop - an automated tool using ml classifiers for phishing detection. In *2022 ieee symposium on security and privacy.*

Pal, M. (2005). Random forest classifier for remote sensing classification. *International journal of remote sensing*, *26*(1), 217–222.

Peng, T., Harris, I., & Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *2018 ieee 12th international conference on semantic computing (icsc)* (pp. 300–301).

Shahi, T. B., & Shakya, S. (2018). Nepali sms filtering using decision trees, neural network and support vector machine. In *2018 international conference on advances in computing, communication control and networking (icacccn)* (pp. 1038–1042).

Shankar, A., Shetty, R., & Nath, B. (2019). A review on phishing attacks. *International Journal of Applied Engineering Research*, *14*(9), 2171–2175.

Swain, P. H., & Hauska, H. (1977). The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, *15*(3), 142–147.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2016). Svm parameter optimization using grid search and genetic algorithm to improve classification performance. *Telkomnika*, *14*(4), 1502.

Verma, R., Shashidhar, N., & Hossain, N. (2012). Detecting phishing emails the natural language way. In *European symposium on research in computer security* (pp. 824–841).

Wu, L., Du, X., & Wu, J. (2014). Mobifish: A lightweight anti-phishing scheme for mobile phones. In *2014 23rd international conference on computer communication and networks (icccn)* (pp. 1–8).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Yin, M., Wortman Vaughan, J., & Wallach, H. (2019). Understanding the effect of accuracy on trust in machine learning models. In *Proceedings of the 2019 chi conference on human factors in computing systems* (pp. 1–12).

Zeydan, H. Z., Selamat, A., & Salleh, M. (2014). Survey of anti-phishing tools with detection capabilities. In *2014 international symposium on biometrics and security technologies (isbast)* (pp. 214–219).