


Detecting Turkish Phishing Attack with Machine Learning Algorithm

Melih Turhanlar¹ ^a and Cengiz Acartürk^{1,2} ^b

¹Cybersecurity Department, Middle East Technical University, Ankara, Turkey

²Cognitive Science Department, Middle East Technical University, Ankara, Turkey

Keywords: Turkish Phishing Attacks, Machine Learning, Bag of Words, TF-IDF, Imbalanced Dataset.


Abstract: Phishing attacks are social engineering attacks that aim at stealing a victim's personal information. The primary motivation is the exploitation of human emotion. The body of a phishing message usually includes a webpage link, aiming at convincing the victim to click and submit credentials. The victim typically connects to a mock webpage. There exist solutions for mitigating phishing attacks, such as phishing detection by Natural Language Processing (NLP). We present a framework for detecting phishing text in Turkish by running machine learning classifiers on an imbalanced phishing data set. The training dataset includes emails, SMS, and tweets. Our findings reveal that the Logistic Regression Synthetic Minority Over-Sampling Technique achieves high performance compared to a set of machine learning models tested in the study.


1 INTRODUCTION

Cyber-attacks consist of social engineering aspects besides technically empowered methodologies. A *phishing attack* is an illegal activity in which the attacker mimics a genuine website with a fake one. Using social engineering techniques, attackers aim at deceiving computer users in order to obtain their personal information, such as credit card numbers, usernames and passwords (Goodrich and Tamassia, 2018). The increase in the mobile market and the use of social media increases the size of the attack surface. Accordingly, cyber-attacks increase their impact through social engineering, besides employing attacks with technical complexity. The European Union Agency for Network and Information Security (ENISA, 2020) reports that social media attacks, conducted through SMS and messaging applications, have increased by 85% for the past decade.

Human users are the main target of phishing attacks. In particular, attackers aim at designing attacks exploiting human vulnerabilities, such as fear and anxiety (Wu et al., 2006). For this, an attacker usually sends bulk emails, SMS messages or uses social media platforms to spread the phishing content (Goodrich and Tamassia, 2018; Toolan and Carthy, 2010). Recently, phishing attacks are ranked as top threats in cyber threat ranking lists (ENISA, 2020).

Following the available work in other languages, the goal of the present study is to analyze Turkish texts in phishing attack vectors, detect Turkish phishing attacks with machine learning classifiers, compare BoW (TF-IDF) and BoW (Frequency) methods, besides Under Sampling, Over Sampling and SMOTE Sampling methods, and introduce a Turkish Phishing Attack dataset. Since the past decade, social media has resulted in enriching the style of phishing attacks. Consequently, the analysis of natural language text gained importance for detecting phishing attacks, besides the classical methods, such as analyzing email headers, blocked list URLs, and IP address (Gualberto et al., 2020; Peng et al., 2018; Azeez et al., 2021). Nevertheless, social media phishing usually includes limited body text in size, unlike emails. The amount of phishing text is also scarce for underrepresented languages. This situation requires the application of specific NLP techniques processing the available data. We present a phishing analysis methodology in Turkish and introduce a data set of phishing text in Turkish (<https://www.turkceoltalama.org>) consisting of 119 Turkish phishing texts. We used the Bag of Words (Frequency) and Bag of Words (TF-IDF) for feature selection. We created the dataset by collecting Turkish phishing emails sent to an official email address of a large state university computer center in Turkey and short messages (SMS) and social media posts and tweets.

^a  <https://orcid.org/0000-0001-5452-596X>

^b  <https://orcid.org/0000-0002-5443-6868>

2 BACKGROUND AND LITERATURE REVIEW

Human is the weakest link in cybersecurity. Therefore, cybercriminals target humans to bypass computer security systems and applications. In social engineering, attackers aim to steal personal information and credentials by using specific methods to manipulate users. Most popular tactics include *Pretexting* (imitating acquaintance and pretending to be someone), *Baiting* (promising to give some gift to the victim), *Quid pro Quo* (pretending to help to the victim in order to obtain confidential information)(Jain and Gupta, 2016). More generally, phishing attacks are social engineering attacks in which attackers use a baiting tactic and aim to steal victim's personal information by utilizing the victim's curiosity and fear emotions(Goodrich and Tamassia, 2018). The attacker usually sends a link with a text which triggers the victim's emotions, such as curiosity and fear. If the victim clicks the link in the text, a connection is made to a mock web server or a fake site that looks like the actual website while being under the control of the attacker. By filling the HTML forms on the fake webpage, the victim may send their credentials unwittingly not to the actual website but actually to the attacker.

A phishing attack is usually executed in three phases (Aleroud and Zhou, 2017). It starts with a **preparation phase**. In that phase, the attacker prepares the attack vector. For this, the attacker may use penetration test tools or write scripts for preparing the attack vector. The attacker then clones the original webpage before starting a phishing campaign. After that, the attacker picks a URL that looks compatible with the text in the attack vector. In addition, the fake website is served on a web server under the control of the attacker. The second phase of the attack is the **execution phase**. In this phase, the attacker chooses a media type. The media may be an email, an SMS, or a social media platform. The text content is adapted according to the type of social media. The attackers mostly use the news media content, email content, a connection to a cloud system for changing a password, or a webpage that the victim submits credit card information for an urgent online payment. The goal of the critical content is to trigger victims' fear and curiosity emotions. Later, the attacker sends the attack vector to the bulk of users, also on social media platforms or through advertisements for a more effective attack vector. The final phase is the **exploitation phase**. In this phase, the attackers use credentials, credit card numbers, personal information of the victims obtained during the campaign.

In the present study, we used Bag of Words (Term Frequency-Inverse Document Frequency), and Bag of Words (Frequency). Term Frequency-Inverse Document Frequency (TF-IDF) is a method for extracting the weight of a word in a text. In a text, a word TF-IDF is a product of $f_{w,t}$, where word frequency in a text, and $\log\left(\frac{|T|}{f_{w,T}}\right)$ where $|T|$ is a count of text in a text corpus, $f_{w,T}$ is the frequency of the word in text corpus.

$$(w_d = f_{w,t} * \log\left(\frac{|T|}{f_{w,T}}\right)) \quad (1)$$

To give an example, assume that an email corpus consists of two emails: Mail-1 = {*This is a phishing attack*} and Mail-2 = {*This is not a phishing attack*}. When we calculate the TF-IDF values of the corpus, all words are presented with 0 but only {*not*} will be presented with 1. Accordingly, this example shows the importance {*not*} as a single instance that identifies the difference between the two sentences. Bag of Words (Frequency) method, which has been used in text classification for transforming sentences into vectors. This method is simpler than the TF-IDF. All of the text content in our corpus was transformed into sentences with frequency values. All the words that we used have their specific locations in the Bag of Words (Frequency) representations.

A review of the previous work reveals numerous studies for mitigating phishing attacks, including technical solutions, such as white and black listings of phishing URLs (Jain and Gupta, 2016; Prakash et al., 2010; Azeez et al., 2021), sending fake entries (Li and Schmitz, 2009), and employing a one-time password (Khan, 2013). Machine Learning have been used widely for detecting phishing attacks. For instance, Garera et al. used logistic regression to detect malicious phishing URLs. Their dataset consisted of 1,245 malicious and 1,263 non-malicious URLs (Garera et al., 2007). Zareapoor et al. classified phishing attacks into two categories, namely deceptive phishing and malware phishing. They proposed a methodology with multiple steps detect deceptive phishing attacks (Zareapoor and K.R., 2015). Fette et al. used not only features, such as whether emails contained javascript or not, the number of dots, and the number of reference links besides the text content (Fette et al., 2007). Abu-Nimeh et al. used 71 features, 60 of them being BoW (TF-IDF) values. They used 5,152 raw English emails, 1,409 of them being phishing emails (Abu-Nimeh et al., 2009). Some of those datasets did not include social media posts. Miyamoto et al. employed nine machine learning techniques on 1,500 phishing URLs and 1,500 legitimate URLs, by using heuristic models employed in CANTINA, also including BoW (TF-IDF) (Miyamoto et al., 2009).

They obtained the best results by using AdaBoost. The dataset was in English. Basnet et al. used the contents of 5,152 ham and 1,409 phishing English emails without using heuristic features (Basnet R. B., 2010). By using the English phishing email corpus (Nazario, 2019), they deployed Confidence-Weighted linear classifiers. Toolan et al. used 4,116 phishing emails in (Nazario, 2019) with 4,202 ham emails (Toolan and Carthy, 2010). They used five features, including the IP address, HTML codes in email content, script in email content, number of links in email body, and period numbers of link. The SVM and C5.0 returned the best accuracy rates.

3 METHODOLOGY

3.1 Dataset

Our dataset consisted of two categories, namely the ham emails and tweets category, and the phishing data. The ham emails were obtained from the IT department call center of the university that the study was conducted. The ham emails were obtained in the mbox format. All non-Turkish emails were left out of the analyses. The ham tweets were obtained from Twitter. Since most of the phishing attacks were in the domain of finance and banking, the ham tweets were selected from Turkish Bank Tweets. A total of 1,200 ham tweets, which were tweeted by Turkish Banks on Twitter, were added to the dataset.

The phishing SMS messages and social media posts were obtained from Twitter. The Tweetdeck application was employed to collect phishing text from Twitter. The Tweetdeck keywords included Named Entities, as well as Turkish common words that were related to phishing. The keywords were {Ten Turkish Banks, Banking Supervision and Assessment Agency, Revenue Administration, Turkish Police Agency, Phishing, Swindler}. All the phishing attacks were identified manually and the content was checked by inspecting the website links inside the attack vectors. We included the same attack vectors to the dataset only once. Accordingly, the texts were assumed to be different in case they involved at least one different word (except for the Named Entities).

The final dataset included 119 different phishing examples. Due to the scarcity of the available, different phishing samples in Turkish, the dataset was established as an imbalanced dataset. The counts of non-phishing and phishing samples are presented in Table 1.

3.2 Preprocessing

The preprocessing pipeline included the following steps: cleansing the syntax, tokenization, removing stop words, and lemmatization. Cleansing is the first step in the preprocessing section. For cleansing HTML tags, the BeautifulSoup 5 Python library is used. Next was tokenization, to parse sentences into words. The tokenization function takes a sentence, uses whitespaces as delimiters and divide sentences into words (He et al., 2011; Zareapoor and K.R., 2015; Zhai and Massung, 2016). Next, stop words are removed since they have no effects in building classifiers (Basnet R. B., 2010; He et al., 2011). We used NLTK 6 library in Python for this. The next step is lemmatization. Lemmatization means finding the root of a given word by eliminating its suffixes and derivative form, in order to represent the word as a single item. The lemmatization process decreases the feature count in the dataset, thus reducing the possibility of overfitting. Turkish is an agglutinative language, i.e. words can be generated with root and suffixes. We use Zemberek 7 for lemmatization. Zemberek is a tool created for Turkish NLP (Akin and Akin, 2019).

3.3 Sampling Methods

In real-world problems, researchers mostly counter with imbalanced data in which one group contains much more data than the other group (Johnson and Khoshgoftaar, 2019). Our dataset is a typical imbalanced dataset, where the number of phishing samples (119) is much smaller than the number of non-phishing samples (3,526). When dealing with an imbalanced dataset, the available methods are Data-Level methods, Algorithm-Level methods, and Hybrid methods (Chawla, 2005). In the present study, the Data-Level methods were employed. These were respectively Under Sampling, Over Sampling, and Synthetic Minority Over-Sampling Method (SMOTE). In order to see the effects of data-level methods, we included Non-Sampling into our methodology. The term Non-Sampling is used to refer to the use of a dataset without sampling. After taking preprocessed text vectors in training data, the sampling methods were not implemented, and they were directly given to machine learning algorithms. Therefore, the training size in Table 1 did not change in this case.

Another method that we employed is Under Sampling, referring to randomly eliminating data from majority class so that minority class size and majority class size become equal. In the present study, Non-Phishing training and Phishing dataset contain 79 examples. While this method decreases the computing

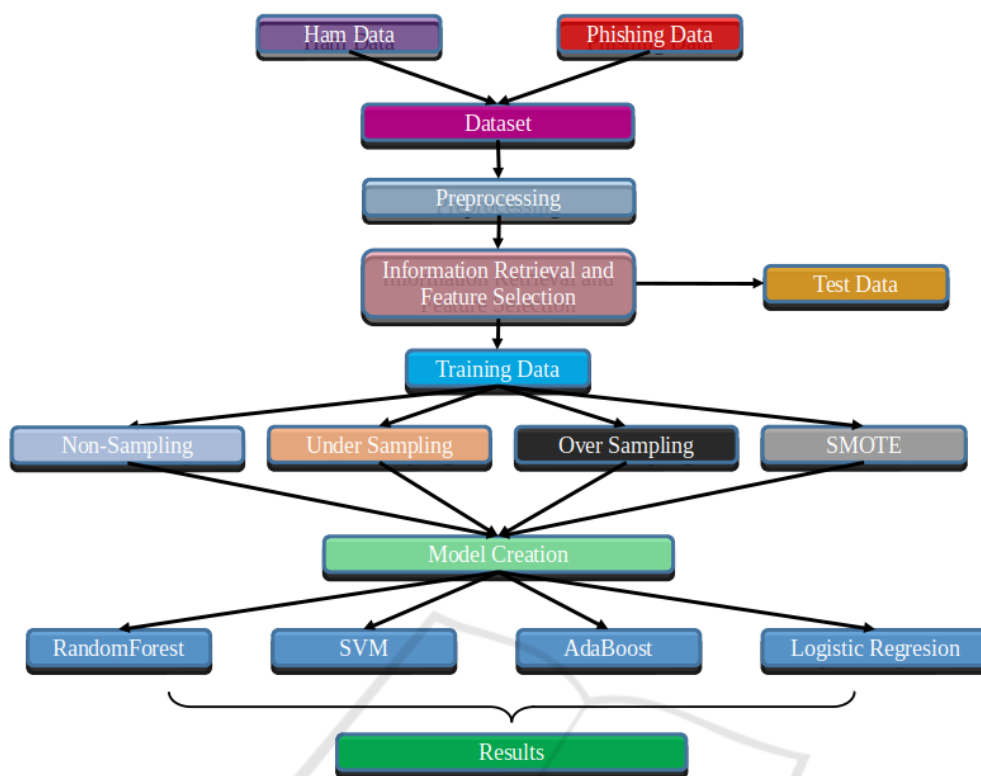


Figure 1: Methodology.

Table 1: Dataset.

	Non-Phishing Samples	Phishing Samples	Ratio
Total	3,526	119	0,033
Training Data	2,363	79	0,033
Test Data	1,163	40	0,034

load, which may be an advantage in intense datasets, it also causes losing information from the majority class. Implementation of Under Sampling was done via the Scikit 8 learn library.

Over Sampling is another type of re-sampling method that we employed. In this method, instances of majority class stay the same, but the minority class instances are increased randomly. Here no new instances are created, only the instances of the minority class were multiplied randomly. While this method avoids losing information on the majority class, repeated instances of minority class may result in overfitting (Chawla, 2005; Chawla et al., 2004). Implementation of Over Sampling was done via the Scikit learn library. An important challenge with the application of Over Sampling method is the overfitting. One alternative to avoid the issues caused by the application of Over Sampling is to create new training data during the training of the model without adding the same instances of the minority class. Synthetic Minority Over Sampling Technique (SMOTE) sam-

pling method adds new instance to minority class for compensating with majority class. It takes k neighbor minority class instances and creates line segments between these instances. Lastly, it creates new minority class instances on these line segments (Chawla, 2005; Chawla et al., 2002). The implementation of SMOTE was done by the Scikit learn library.

3.4 Information Retrieval and Feature Selection

After the preprocessing of data, information retrieval methods were applied to the data. For the implementation, two methods were employed: BoW (TF-IDF) and BoW (Frequency). The Scikit learn library was used. Before choosing the parameters of the information retrieval methods, the texts were processed step by step. First, the frequency of all words was calculated. After calculating the frequency of all words in the dataset, 10 most used words were removed from

the dataset. Also, the words which had less frequency (the bottom 10) were removed from the dataset.

Later, for feature selection and finding optimum `max_feature` parameter of BoW (TF-IDF) and BoW (Frequency), binary search was implemented manually on the dataset. Binary search is an algorithm which used for searching. It is applied by splitting the set into two equal subsets and comparing it with others for finding appropriate value (Rosen, 2002). For all samples, firstly, a feature count was taken and given as a parameter into BoW (TF-IDF) and BoW (Frequency). Secondly, with the feature counts the optimum parameters of all sampling methods and classifiers were found with the help of the Grid Search method in Scikit learn. Later, optimum parameters of ML classifiers were given as parameters and again all models with new parameters were created. Lastly, the means of F1 scores in the test dataset were taken and examined. As a result, 913 has been found and the optimum number of feature count given into `max_feature` parameter and vectors created.

3.5 Model Creation

In the present study, we used four machine learning classifiers. In this section, we present how we found their optimal parameters during the feature section and model creation phases.

Firstly, we implemented Random Forest, which is an ensemble learning method used for classification. During training, it creates decision trees and by using them it classifies the data (Marchal, 2015). Secondly, we employed Logistic regression, which is a popular binary classification method. Thirdly, we used AdaBoost, which is a boosting method that fits weak classifiers and training them in order to increase success with random guessing. Lastly, we implemented SVMs (Support Vector Machines). SVM aims at finding the lowest true error and it works well on text classification due to its advantage for avoiding overfitting.

By following the common practice in ML model development, we divided the dataset into a training set (66%) and a test set (33%). The methodology pipeline was run on a desktop computer with Intel Core i7-8550U CPU 1.80 GHz, 20 GB RAM, and Linux Ubuntu 16.04 operating system. The following section presents the results of the study.

4 RESULTS

Since the dataset was an imbalanced dataset, evaluating the results solely on accuracy scores may be

misleading. Therefore, performance metrics of classifiers have been evaluated on other metrics. We employed the the F1-Score and the Area Under Curve (AUC) for evaluating the models. The findings are presented in Table 2. The results show that the Under Sampling training underperformed compared to Non-Sampling, Over Sampling, and SMOTE Sampling training on the selected classifiers. The highest score was obtained for LR BoW (TF-IDF) with the SMOTE method. In addition, the BoW(Frequency) model did not show high performance, especially on the SMOTE Sampling method.

More specifically, the results show that the Logistic Regression BoW (TF-IDF) model with the SMOTE Sampling method is the one that showed the highest performance, returning the highest F1 score (0.923). It is followed by the SVM BoW (TF-IDF) SMOTE Sampling (F1 score of 0.909), the SVM BoW (TF-IDF) model with Over Sampling (0.909), and finally the Random BoW (TF-IDF) with Over Sampling (0.907). On the other hand, the BoW (Frequency) method gave the highest F1 score for AdaBoost BoW (Frequency) Over Sampling (0,886) and the AdaBoost BoW (Frequency) Non-Sampling (0,883), and then Logistic Regression BoW (Frequency) Non-Sampling (0.883).

Table 3 shows precision and recall values of all classifiers. Logistic Regression BoW (TF-IDF) with SMOTE Sampling method precision is 0.947 and recall value is 0.90. The classifiers also returned false alarms to non-phishing instances while detecting most of the phishing attacks. On the other hand, among 40 phishing instances in the test dataset, 30 of 32 classifier models detected over 30 phishing instances (i.e., %75 of all the phishing instances were successfully detected).

5 DISCUSSION AND CONCLUSION

In the present study, we investigated four machine learning classifiers, namely Random Forest, Logistic Regression, AdaBoost, and Support Vector Machine with two methods BoW (TF-IDF) and BoW (Frequency). Due to the imbalanced characteristics of the dataset, we employed various sampling methods, namely Under Sampling, Over Sampling, SMOTE Sampling, and Non-Sampling. Consequently, we developed 32 machine learning models.

In the literature, there exist three main methods to process imbalanced datasets; data-level methods, algorithm-level methods, and hybrid methods. We used the data-level methods, thus the sampling meth-

Table 2: F1 and AUC Scores of Classifiers.

	Non-Sampling		Under Sampling		Over Sampling		SMOTE	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC
RF B.Tf.	0.806	0.837	0.698	0.889	0.907	0.900	0.892	0.862
RF B.Freq.	0.823	0.850	0.673	0.889	0.873	0.900	0.596	0.922
LR B.Tf.	0.892	0.912	0.755	0.903	0.889	0.947	0.923	0.949
LR B.Freq.	0.883	0.923	0.758	0.891	0.872	0.923	0.584	0.895
AdaBoost B.Tf.	0.880	0.911	0.550	0.875	0.883	0.911	0.886	0.922
AdaBoost B.Freq.	0.883	0.923	0.522	0.862	0.886	0.935	0.496	0.897
SVM B.Tf.	0.895	0.924	0.847	0.896	0.909	0.936	0.909	0.936
SVM B.Freq.	0.842	0.910	0.643	0.884	0.857	0.910	0.525	0.878
Dummy B.Tf.	0.067	0.461	0.049	0.547	0.062	0.520	0.060	0.564
Dummy B.Freq.	0.050	0.462	0.046	0.470	0.071	0.504	0.064	0.531

RF: Random Forest
 LR: Logistic Regression
 B.Tf.: Bag of Word TF-IDF
 B.Freq.: Bag of Word Frequency

Table 3: Precision and Recall Values of Classifiers.

	Non-Sampling		Under Sampling		Over Sampling		SMOTE		Total (Mean)	
	P	R	P	R	P	R	P	R	P	R
RF B.Tf.	1.00	0.675	0.560	0.925	0.971	0.850	0.970	0.825	0.875	0.818
RF B.Freq.	1.00	0.700	0.537	0.900	1.00	0.775	0.459	0.850	0.749	0.806
LR B.Tf.	0.970	0.825	0.637	0.925	0.878	0.900	0.947	0.900	0.858	0.887
LR B.Freq.	0.918	0.850	0.654	0.900	0.894	0.850	0.452	0.825	0.729	0.856
AdaBoost B.Tf.	0.942	0.825	0.395	0.900	0.918	0.850	0.897	0.875	0.788	0.862
AdaBoost B.Freq.	0.918	0.850	0.372	0.875	0.897	0.875	0.350	0.850	0.634	0.862
SVM B.Tf.	0.944	0.850	0.800	0.900	0.945	0.875	0.945	0.875	0.908	0.875
SVM B.Freq.	0.888	0.800	0.500	0.900	0.891	0.825	0.390	0.800	0.667	0.831
Dummy B.Tf.	0.035	0.55	0.026	0.400	0.033	0.525	0.032	0.475	0.031	0.487
Dummy B.Freq.	0.026	0.400	0.024	0.375	0.038	0.600	0.034	0.500	0.030	0.468
Mean	0.947	0.796	0.556	0.903	0.924	0.85	0.676	0.85		

P=Precision
 R=Recall

ods for the training. The algorithm-level methods do not add new samples to a dataset. They rather have an impact during the learning process. The algorithm-level methods are mainly employed through cost-sensitive approaches, by setting values in a cost matrix based on experiences(Johnson and Khoshgoftaar, 2019). Due to practical challenges in the implementation of the algorithm-level methods, we preferred the data-level methods, thus leaving the application of algorithm-level methods to a further study.

Table 4 shows the results of the other studies in the literature. Zareapoor et al. and Basnet et al. (Basnet R. B., 2010; Zareapoor and K.R., 2015) investigated email content and they used a balanced dataset (see (Nazario, 2019)for a recent version of the dataset). Zareapoor et al.(Zareapoor and K.R., 2015) study fea-

ture selection methods, respectively Chi-Square, Gain Ratio, Information Gain and without feature selection for comparing results. They have calculated the F1 scores for the SVM as 0.994, for AdaBoost 0.995, and for Random Forest 0.995. As seen in Table 2, our models have almost reached their value with Random Forest BoW (TF-IDF) Over Sampling (0.906), AdaBoost BoW (TF-IDF) SMOTE Sampling (0.8861) and SVM BoW (TF-IDF) SMOTE (0.909).

Basnet et al.(Basnet R. B., 2010) compared two machine learning algorithms on the balanced dataset, Confidence-Weighted Linear Classifiers and SVM with a linear kernel. They report 0.998 and 0.997 respectively. On the other hand, Miyamoto et al.(Miyamoto et al., 2009) created a dataset that contains only the URLs, reporting scores lower than the

Table 4: Results of Other Studies Authors.

Features		Results		Dataset	
		Classifiers	F1 Scores	Phishing	Non-Phishing
(Zareapoor Seeja, 2015)	Email Content	SVM	0.994	500	1,400
		Ada Boost	0.995		
		RF	0.995		
(Miyamoto, Hazeyama, Kadobayashi, 2009)	URLs	AdaBoost	0.858	1,500	1,500
		LR	0.855		
		SVM	0.856		
		RF	0.855		
(Basnet Sung, 2010)	Email Content	CWLC	0.998	1,409	5,152
		SVM	0.996		

ones found in the present study. This finding suggests that including the URLs as the sole feature may lead to suboptimal scores.

The present study reported an initial analysis of a small-scale, imbalanced phishing dataset in Turkish. In future work, we plan to grow the dataset by presenting it as a public, crowdsourcing service. A larger dataset will allow using more advanced learning methods than the classical machine learning models, such as deep learning models. An alternative is to implement algorithm-level methods or hybrid methods in addition to the data-level methods on the imbalanced datasets. Another method of enlarging the dataset is to translate English phishing datasets, such as (Nazario, 2019) into Turkish, and use it as a training dataset on machine learning classifiers. Nevertheless, it is important to take into account the characteristics of phishing emails across cultures, in particular the role of cultural traits in the society. For instance, the content of the phishing emails may include more financial statements in a country than the other. Those culture-specific characteristics may have a significant impact in the dataset, thus the models.

Another aspect of the study that has been left to future work is the discrimination between spam emails and phishing emails. Recently, obtaining a Turkish spam email sample is easier than obtaining a Turkish phishing attack sample. Although spam and phishing attacks differ from each other, they could have some same features, too. By employing transfer learning methods, inherent features of Turkish spams may even help the models learn Turkish phishing attacks.

Phishing attacks mostly contain malicious URLs inside them and aims to access our personal information, credit card numbers, credentials, etc. On the other hand, Spam is mostly used for advertising purposes without containing any malicious URLs. But both can be spread via the same media types like

emails, SMS messages, etc. In literature, there is also some Turkish spam detection researches (Özgür et al., 2004). As future work, Turkish spam data can be used as training data, and models can be tested with a phishing dataset. Also, the same features can be mapped between spam emails and phishing attacks.

REFERENCES

- Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2009). Distributed phishing detection by applying variable selection using bayesian additive regression trees. In *2009 IEEE International Conference on Communications*, pages 1–5.
- Akın, A. A. and Akın, M. D. (2019). Zemberek, an open source nlp framework for turkic languages.
- Aleroud, A. and Zhou, L. (2017). Phishing environments, techniques, and countermeasures: A survey. *Computers & Security*, 68:160–196.
- Azeez, N. A., Misra, S., Margaret, I. A., Fernandez-Sanz, L., and Abdulhamid, S. M. (2021). Adopting automated whitelist approach for detecting phishing attacks. *Computers & Security*, 108:102328.
- Basnet R. B., . S. A. H. (2010). Classifying phishing emails using confidence-weighted linear classifiers. In *International Conference on Information Security and Artificial Intelligence (Isai)*. Isai.
- Chawla, N. V. (2005). *Data Mining for Imbalanced Datasets: An Overview*, pages 853–867. Springer US, Boston, MA.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357.
- Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6.
- ENISA (2020). Enisa threat landscape report 2020, 15 top cyberthreats and trends. In *ENISA threat landscape report 2020*, page 138.

- Fette, I., Sadeh, N., and Tomasic, A. (2007). Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, page 649–656, New York, NY, USA. Association for Computing Machinery.
- Garera, S., Provos, N., Chew, M., and Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode, WORM '07*, page 1–8, New York, NY, USA. Association for Computing Machinery.
- Goodrich, M. and Tamassia, R. (2018). *Introduction to Computer Security*. Pearson, 2nd edition.
- Gualberto, E. S., De Sousa, R. T., De B. Vieira, T. P., Da Costa, J. P. C. L., and Duque, C. G. (2020). From feature engineering and topics models to enhanced prediction rates in phishing detection. *IEEE Access*, 8:76368–76385.
- He, M., Horng, S.-J., Fan, P., Khan, M. K., Run, R.-S., Lai, J.-L., Chen, R.-J., and Sutanto, A. (2011). An efficient phishing webpage detector. *Expert Systems with Applications*, 38(10):12018–12027.
- Jain, A. K. and Gupta, B. B. (2016). A novel approach to protect against phishing attacks at client side using auto-updated white-list. *EURASIP Journal on Information Security*, 2016(1):9.
- Johnson, J. M. and Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27.
- Khan, A. (2013). Preventing phishing attacks using one time password and user machine identification. *International Journal of Computer Applications*, 68.
- Li, S. and Schmitz, R. (2009). A novel anti-phishing framework based on honeypots. In *2009 eCrime Researchers Summit*, pages 1–13.
- Marchal, S. (2015). *DNS and Semantic Analysis for Phishing Detection*. PhD thesis, Aalto University.
- Miyamoto, D., Hazeyama, H., and Kadobayashi, Y. (2009). An evaluation of machine learning-based methods for detection of phishing sites. In Köppen, M., Kasabov, N., and Coghill, G., editors, *Advances in Neuro-Information Processing*, pages 539–546, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Nazario, J. (2019). Phishing emails [accessed 3 July 2019].
- Peng, T., Harris, I., and Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 300–301.
- Prakash, P., Kumar, M., Kompella, R. R., and Gupta, M. (2010). Phishnet: Predictive blacklisting to detect phishing attacks. In *2010 Proceedings IEEE INFOCOM*, pages 1–5.
- Rosen, K. H. (2002). *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 5th edition.
- Toolan, F. and Carthy, J. (2010). Feature selection for spam and phishing detection. In *2010 eCrime Researchers Summit*, pages 1–12.
- Wu, M., Miller, R. C., and Little, G. (2006). Web wallet: Preventing phishing attacks by revealing user intentions. In *Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS '06*, page 102–113, New York, NY, USA. Association for Computing Machinery.
- Zareapoor, M. and K.R., S. (2015). Feature extraction or feature selection for text classification: A case study on phishing email detection. *International Journal of Information Engineering and Electronic Business*, 7:60–65.
- Özgür, L., Güngör, T., and Gürgen, F. (2004). Adaptive anti-spam filtering for agglutinative languages: a special case for turkish. *Pattern Recognition Letters*, 25(16):1819–1831.
- Zhai, C. and Massung, S. (2016). *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool.