Malicious URL Classification Using Extracted Features, Feature Selection Algorithm, and Machine Learning Techniques

Jo Simon Ambata, Jose Gaurana, Dan Jacinto, and Joel De Goma
School of Information Technology Department, Computer Science
Mapúa University
Makati, Philippines
ambata.josimon@gmail.com, gauranajose@gmail.com,
jacinto.dnj@gmail.com, jcdegoma@mapua.edu.ph

Abstract

Websites have different purposes. Some of which intend legitimate functions in the economy while some of which intend harmful cases towards users. Although various research has been made to address this problem, these detection systems still leave plenty of room for improvement, specifically on its performances. This study was based on the recommended approach for future work by a related work wherein it contains 10 base features of a URL for its classification. The recommended approach states that an extended number of features from the base features increases the detection accuracy. In this paper, it proposes a comparison between the performance of three cases: the base 10 features, an extended feature set, and a set where a feature selection algorithm is applied. The researchers utilized machine learning algorithms to build models in classifying legitimate and malicious URLs. The study showed that there is a directly proportional relationship with a model's number of features and a model's performance. Extending the number of features of the data set leads to an increase with the performance of each model.

Keywords
URL Classification, machine learning, feature selection algorithm

1. Introduction

The advent of the internet has transformed the world into a global village, thus revolutionizing the communication technologies that have impacted the lives of people as well as the growth of businesses (Shivangi et al. 2018). Malicious URLs are an important security issue to the Internet, which has a significant economic impact. By now, it is still a challenging problem (Chunlin et al. 2018). Machine learning techniques have been increasingly applied to solve the problems relating to information security and cybersecurity. Malicious URL (Uniform Resource Locator) detection is one of these (Vanhoenshoven et al. 2016).

This paper aims to compare three approaches on URL classification. These approaches will differ on the features used in model creation. The first approach contains a set of 10 features adapted from Cuzzocrea et al. (2018). The second approach extends the feature set of the first approach. The last approach implements a feature selection algorithm on the features of the second approach to contain the top number of features. J48, k-Nearest Neighbors, Sequential Optimal Optimization, and Random Forest are the classifiers to be used on the models.

The following classifiers were selected because they yielded acceptable performance based on previous studies. Random Forest was used in the work of Cuzzocrea et al. (2018), Chunlin et al.(2018), Vanhoenshoven et al. (2016), Al-Janabi et al. (2017), and Yerima and Alzaylee (2020). For K-Nearest Neighbor, the related works Vanhoenshoven et al. (2016) and Al-Janabi et al. (2017). For J48, the works of Cuzzocrea et al. (2018), Chunlin et al. (2018), Aydin et al. (2020), and Yerima and Alzaylee (2020). And for Support Vector Machines the works of Chunlin et al. (2018), Aung and Yamana (2019), and Vanhoenshoven et al. (2016) wherein the researchers used C-Support Vector Classification. The different machine learning algorithms were used to determine the algorithm that has the best performance. The previous studies had different results for each algorithm.

1.1 Objectives

This research aims to provide the recommended approach from Cuzzocrea et al. (2018) for malicious URL detection using machine learning techniques. Adding features from the related works of Aung and Yamana (2019), Al-Janabi et al. (2017), and Chapla et al. (2019). Afterwards, applying a feature selection algorithm in determining the relevant properties in obtaining the performance of the model.

1.) To determine how extending the features influenced the performance of the model.
2.) To compare the results of the models' performance with and without the feature selection method.
3.) To determine which Machine Learning Algorithm produced the best model in malicious URL detection.

2. Literature Review

Alfredo Cuzzocrea et al. (2018) focused on applying machine learning techniques in order to detect whether a website is legitimate or phishy. It used the dataset PhishingData.csv from the UCI Machine Learning Repository. The features stated in their study contains 10 attributes, namely: SFH, prefix/suffix, IP, Request_URL, URL_of_Anchor, web_traffic, URL_Length, age_of_domain, sub_domain, having_IP_Address. This paper used a Website Phishing dataset in the form of .csv file format. The classification analysis consists of building deep learning classifiers that have a goal to assess the feature vector accuracy to determine between phishing and normal web pages. In the learning phase, the researchers considered k-fold cross-validation as it is randomly divided into k subsets. One subset is set as a validation dataset for the model's testing. The k – 1 subset is used as training data. This process is repeated for k = 10 times. They then obtained an estimate by finding the average of the k results. In evaluating the effectiveness of a classification method, they followed a procedure to do so: build a training set, build a testing set, run the training phase and then applied the learned classifier to each element. Machine Learning techniques are used to classify websites either as legitimate or phish.

Mustafa Aydin et al. (2020) aim to present their findings on the methods of detecting phishing websites. For the dataset, they collected URLs in PhishTank for the fraudulent URLs and Google Search Engine for the legitimate URLs. Through their feature extraction, they obtained 133 separate features related to the websites' URL. As for the feature selection algorithm, the Gain Ratio attribute and ReliefF attribute are selected. These techniques are used to minimize the feature matrix dimension by eliminating irrelevant and unnecessary properties. Data mining algorithms with classifier algorithms are used to achieve the result. The classifiers they used are: Naïve Bayes, Sequential Minimal Optimization, and J48 algorithms. SMO and J48 had satisfactory results in its performance in detecting phishing websites. While Naive Bayes had a large difference from the two classifiers and would not be recommended.

3. Methods

3.1 Conceptual Framework

The researchers conducted three cases for model creation. These three cases differ based on the feature set used. The first case used the 10 base features adapted from Cuzzocrea et al. (2018). The second case extends the feature set by adding 46 features taken from the research of Aung and Yamana (2019), Al-Janabi et al. (2017), and Chapla et al. (2019). The third case applied a feature selection algorithm, Gain Ratio, on the second approach's features to create a different feature set containing the top number of ranked features. Each case then created models using the following classifiers: J48, K-Nearest Neighbors, C-Support Vector Classification, and Random Forest. An 8:2 ratio is used to split the dataset for training and testing. Stratified 10-Fold Cross Validation is the validation method that was used. Each model per approach was compared to determine the best classifier. Each approach is compared to determine if extending the features and applying a feature selection algorithm influences the models' accuracy. The following are the criteria to be used for comparison: accuracy, precision, and recall.

3.2. Data Gathering

The URLs are obtained in a dataset by Kumar from Kaggle where he based his dataset in PhishTank and other sites. It contains approximately 450,000 URLs where 77% of it is benign and 23% is malicious.

3.3. Data Pre-Processing

Once the dataset has been obtained, it undergoes a data cleaning process. This data cleaning involved removing duplicated domains and unsuccessful status resulting in 60,317 URLs remaining. It contains 44,467 legitimate URLs and 15,850 malicious URLs, giving an imbalanced dataset. In the event of missing values, instead of dropping the data, the researchers imputed these missing values. For the numerical data, the mode of the feature is the filler of the missing values.

.

3.4. Feature Extraction

The researchers used the programming language Python in the software Jupyter Notebook to extract the necessary features. This includes JSON to Python in the requests library, HTTPS Connections, and Regular Expressions search. For features that require premium access, such as web traffic, domain rank, and number of referring links, the researchers subscribed to SerpStat, a search engine optimization platform. This provided the researchers a premium API code to access the said features. The research consisted of a total of 56 features. 10 of these are the base features taken from Cuzzocrea et al. (2018) and 33 features taken from Aung and Yamana (2019), 5 features from Al-Janabi et al. (2017), 8 features from Chapla et al. (2019). The first case contains the 10 base features listed on Table 1. The second case contains the 10 base features and 50 features listed on Table 2. The third case contains the top 75% features after applying Gain Ratio feature selection algorithm to the extended set, the ones listed in Table 2. All values of the features extracted were then normalized by applying the min-max normalization method.

3.5 List of Features

Table 1. 10 Base Features

Feature	Description
url_length	length of the URL in characters
prefix_suffix	having '-' in prefix/suffix of URL
having_ip_address	has IP address in URL
domain_age	age of domain of the URL
sub_domain	if URL has subdomain
web_traffic	web traffic of a website of a URL
SFH	Server Form Handler
url_anchor	containing anchor text
request_url	request of a URL
ip	is IP blacklist

Table 2. Extended Features

Feature	Description
10 Base Features	Table 4's Features
double_forward_slash_redirect	if path of URL contains a "//" to redirect users to a phishing page
exe	if URL contains an exe file
sensitive_words	if URL contains sensitive words such as confirm, account, bank, secure, login, sign in, webscr, submit, update, logon, wp, cmd, admin
port_number	which port number is used
free_hosting	if URL uses a free hosting domain
percentage_hash	percentage of '#' in URL
percentage_at	percentage of '@' in URL
percentage_dash	percentage of '-' in URL
percentage_dot	percentage of '.' in URL
percentage_dollar	percentage of '$' in URL
percentage_ast	percentage of '*' in URL
percentage_open_bracket	percentage of '[' in URL
percentage_open_parenthesis	percentage of '(' in URL
percentage_open_curly_bracket	percentage of '{' in URL
percentage_close_bracket	percentage of ']' in URL
percentage_close_curly_bracket	percentage of '}' in URL
percentage_close_parenthesis	percentage of ')' in URL
percentage_plus	percentage of '+' in URL
percentage_semicolon	percentage of ';' in URL
percentage_tidle	percentage of '~' in URL
percentage_colon	percentage of ':' in URL
percentage_apostrophe	percentage of ''' in URL
percentage_forward_slash	percentage of '/' in URL
percentage_percent	percentage of '%' in URL
percentage_question_mark	percentage of '?' in URL
percentage_comma	percentage of ',' in URL
percentage_equal	percentage of '=' in URL
percentage_ampersand	percentage of '&' in URL
percentage_exclamation_mark	percentage of '!' in URL
percentage_underscore	percentage of '_' in URL
count_NAN	total count of non-alphanumeric characters
input_form	number of input forms in

	URL
number_of_dots	count of '.' in URL
number_of_letters	count of letters in URL
number_of_digits	count of digits in URL
number_of_referringlinks	number of links from domains to URL's domain
domain_rank	strength of a website's total backlink profile (SerpStat Domain Rank)
number_of_percentage	count of '%' in URL

number_of_equals	count of '=' in URL
number_of_ampersand	count of '&' in URL
number_of_questionmark	count of '?' in URL
number_of_hyphen	count of '-' in URL
number_of_at	count of '@' in URL
having_atsign	URL contains an '@'
is_redirect	if URL redirects to a different
dns_record	if URL has DNS record

3.6. Feature Selection

The researchers applied a feature selection algorithm for the third case. The algorithm to be used is Gain Ratio adopted from the method of Aydin et al. (2020). It runs individually on the feature dataset and is analyzed by Orange data mining and classification software tool. To find the threshold for the optimal model, the researchers created models for the top 25%, top 50%, top 75%, and the mean of scores and then picked the model that generated the highest performance. This concept was based on Jin et al. (2007) in terms of testing the performance of the number of features by increments. From this, the researchers concluded that in this dataset, more features would result in a better performance of the model. The best approach in applying Gain Ratio algorithm in selecting features is using the top 75% features (top 42 out of 56 features) since this yielded the best performance among the other approaches in using Gain Ratio algorithm.

3.6.1. Gain Ratio

Gain Ratio is a modification of the information gain that reduces its bias on high-branch attributes. It takes the number and size of branches into account when choosing an attribute as described in the CCSU website.

3.7. Classifiers

The classifiers to be used are k-Nearest Neighbor, Implementation of Sequential Minimal Optimization on C-Support Vector Classification, Random Forest and J48.

3.7.1. K-Nearest Neighbors

K-Nearest Neighbors is a classification algorithm that uses historical data rather than performing a learning algorithm. It stores the available class data points and classifies the new data points based on the distance similarity (Vanhoenshoven et al. 2016).

3.7.2. C-Support Vector Classification

Sequential Minimal Optimization is an agile responding method proposed by John C. Platt to support SVM. It trains a set of volume that is greater and better than chunking (Aydin et al. 2020).C-Support Vector Classification will be used. C-SVC is an implementation based on LIBSVM which implements an SMO-type algorithm.

3.7.3. Random Forest

Random Forest is a machine learning technique that builds an ensemble of random decision trees that classifies data based on the majority vote. It helps to avoid overfitting of the training set (Cuzzocrea et al. 2018).

3.7.4. J48

J48 is a decision tree algorithm that uses the concept of information entropy. It computes the highest information gain and uses it as the splitting criteria (Cuzzocrea et al. 2018).

3.8. Training and Validation

For training, the researchers split the dataset into an 8:2 ratio for training set and test set, respectively. From this, a stratified 10-fold cross-validation was used. This is performed to assess the effectiveness of the model wherein it would reduce the chances of overfitting. It works well with an imbalanced dataset because the dataset is partitioned into k groups such that the validation data has an equal number of instances of the target class.

3.9. Performance Criteria

The criteria for comparison of the performance of the models focused on the accuracy, precision, and recall. The researchers utilized a confusion matrix. This describes the classification results in detail Chunlin et al. (2018). A confusion matrix contains an instance of a predicted class and actual class. This is used to detect whether that instance is a true positive (TP) where both the predicted class and actual class is positive, false positive (FP) where the predicted class is positive, but the actual class is negative, true negative (TN) where both the predicted class and actual class is negative, and false negative (FN) where the predicted class is negative, but the actual class is positive. This can be used to compute for accuracy, precision, and recall.

3.9.1. Accuracy

The ratio between the correctly predicted outcomes, and the sum of all predictions (Yerima and Alzaylee 2020). The formula for accuracy is: (TP+TN)/(TP+TN+FP+FN)

3.9.2. Precision

The ratio between all true positives, and all positive predictions. This determines if the model was correct in predicting the positives (Yerima and Alzaylee 2020). The formula for precision is: TP/(TP+FP)

3.9.3. Recall

The ratio between all true positives, and all actual positives. Determines how many positives did the model identify out of all the possible positives (Yerima and Alzaylee 2020). The formula for recall is: TP/(TP+FN)

4. Data Collection

The features from the dataset were extracted using python. In some features, the researchers subscribed to Serpstat, a search engine optimization tool, to acquire data that required license. For the top ranked features, Orange application tool was used. As for the performance results, classifiers in the sklearn library in Python were used.

5. Results and Discussion

5.1 Numerical Results

5.1.1. Base Features Set

Table 3. Results of Classifiers to the Base Features Set

Dataset	Method	Accuracy	Precision	Recall
Base	Random Forest	94.222%	94.254%	94.222%
	J48	93.593%	93.969%	93.593%
	C-SVC	93.112%	93.631%	93.112%
	kNN	94.231%	94.406%	94.231%

For the results of Table 3, using the 10 base features set for the classification of the URLs, Random Forest had an accuracy, precision, and recall of 94.222%, 94.254%, and 94.222%, respectively. J48 had 93.593%, 93.969%, and 93.593%. C-SVC had 93.112%, 93.631%, and 93.112%. kNN had 94.231%, 94.406%, and 94.231%. In terms of using the base feature set, the K Nearest Neighbors classifier generated the best performance. The classifier that had the worst performance, on the other hand, was C-SVC.

5.1.2. Extended Features Set

Table 4. Results of Classifiers to the Extended Features Set

Dataset	Method	Accuracy	Precision	Recall
Extended	Random Forest	99.552%	99.553%	99.552%
	J48	99.403%	99.405%	99.403%
	C-SVC	99.461%	99.462%	99.461%

	kNN	99.337%	99.337%	99.337%

For the results of Table 4, using the extended features set for the classification of the URLs, Random Forest gained an accuracy, precision, and recall of 99.552%, 99.553%, and 99.552% respectively. J48, on the other hand, gained 99.403%, 99.405%, and 99.403%. C-SVC gained 99.461%, 99,462%, and 99.461%. Lastly, kNN gained 99.337%, 99.337%, and 99.337%. Using the extended feature set, the Random Forest classifier had the greatest result in terms of the model's accuracy, precision, and recall. The J48 algorithm had the least result in performance.

5.1.3 Feature Selection Algorithm Feature Set

Table 5. Results of Classifiers to the Features Selected Set using Gain Ratio

Dataset	Method	Accuracy	Precision	Recall
Gain Ratio	Random Forest	99.528%	99.528%	99.528%
	J48	99.130%	99.129%	99.130%
	C-SVC	99.469%	99.470%	99.469%
	kNN	99.387%	99.388%	99.387%

For Table 5, using the top 75% features selected using Gain Ratio algorithm for the classification of the URLs, Random Forest had an accuracy, precision, and recall of 99.528%, 99.528%, and 99.528%, respectively. J48 had 99.130%, 99.129% and 99.130%. C-SVC had 99.469%, 99.470%, and 99.469%. kNN had 99.387%, 99.388%, and 99.387%. The Random Forest algorithm generated the best performance in the feature selection approach. The worst performance by a classifier was the J48 algorithm.

5.2 Graphical Results

5.2.1. Performance of Base Feature Set versus Extended Feature Set

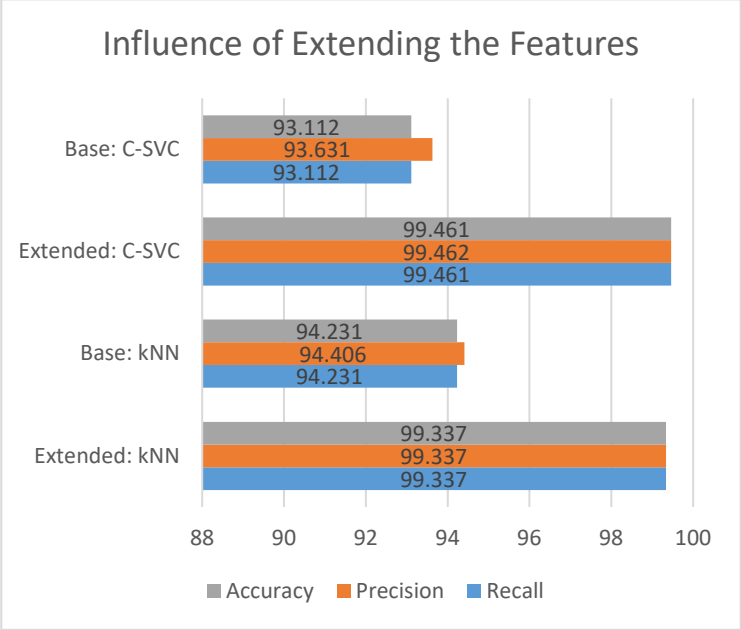

Figure 1. Influence of Extending the Features

For Figure 1, extending the features had a drastic increase in the performance of the model. It had influenced the performance by at least 5.106% and at most 6.349% in accuracy, precision, and recall. kNN had the lowest increase while C-SVC had the highest increase among the classifiers.

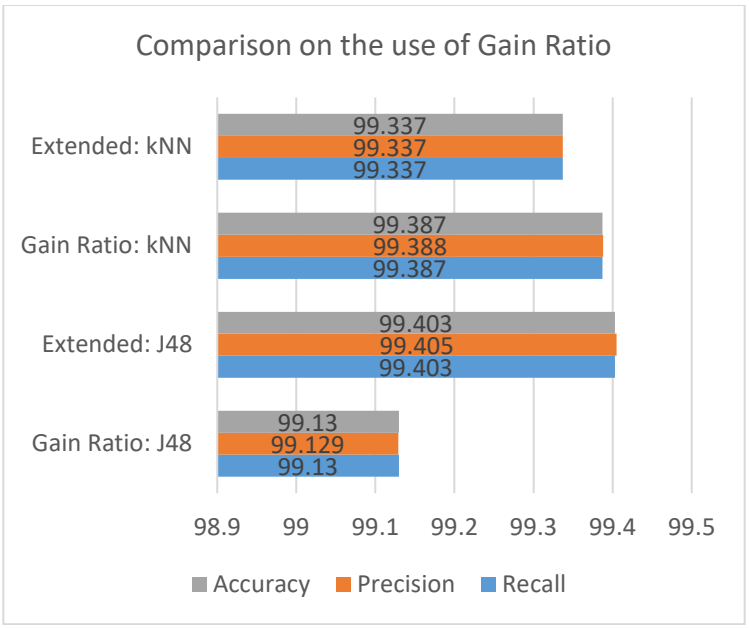

Figure 2. Comparison on the use of Gain Ratio

5.2.2. Performance of Extended Feature Set versus Feature Selection Set

For figure 2, applying Gain Ratio Feature Selection algorithm to the Extended Features in getting the top 75% of the features had a slight difference. The difference from the preceding model ranges from –0.273 to +0.050. The biggest decrease was for the J48 having a difference of –0.273 while the biggest increase was for the kNN having a difference of +0.050.

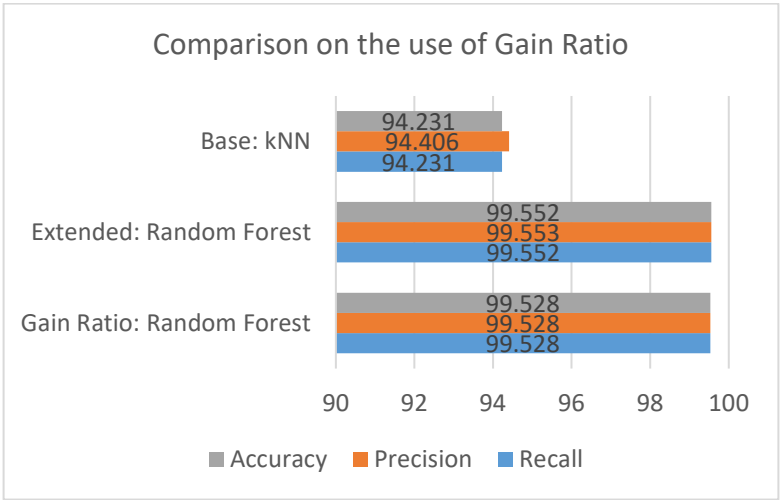

Figure 3. Comparison of the Best Model in each Case

5.2.3. Best Model in all the Cases of URL Detection

For figure 3, applying Gain Ratio Feature Selection algorithm to the Extended Features in getting the top 75% of the features had a slight difference. The difference from the preceding model ranges from –0.273 to +0.050. The biggest decrease was for the J48 having a difference of –0.273 while the biggest increase was for the kNN having a difference of +0.050.

5.3 Proposed Improvements

For future work, the researchers propose to include the number of features used and machine processing cost in acquiring the performance of a model. From this, the researchers recommend finding a proper calculation on how to

get the most cost-efficient model. For this reason, it would assist in a possible future development of URL detection in a more efficient way wherein it would also work on weak and slow computers/mobile devices. Moreover, adding more features from the features this study has would be suggested. The researchers also recommend exploring different feature selection algorithms as it may yield different importance values to the respective features.

5.4 Validation

A stratified 10-fold cross-validation was used. This is performed to assess the effectiveness of the model reducing the chances of overfitting. It works well with an imbalanced dataset because the dataset is partitioned into k groups such that the validation data has an equal number of instances of the target class.

6. Conclusion

By the end of the study, the researchers were able to apply the recommended approach from Cuzzocrea et al. (2018) for malicious URL detection using machine learning techniques. Features were added from the related works of Aung and Yamana (2019), Al-Janabi et al. (2017), and Chapla et al. (2019). Feature selection algorithm method was applied to determine which are the relevant features in obtaining the performance of the model.

As recommended by Cuzzocrea et al. (2018), extending the features influenced the performance of the model. In the study, each of the models had an increase in performance. This is because with several features included, it considers numerous combinations of those features to yield the optimal classifier for the model (Chatterjee and Namin 2019). Feature selection method was used on the extended features using Gain Ratio Algorithm. In the study, each of the models had a decrease in performance but also a significant increase in performance as compared to the base features. The average best performance in the variations of selecting the top number of features in the Gain Ratio algorithm is the top 75% of the features. However, the Extended Feature Set still had a better performance than the Feature Selection Set. Overall, the machine learning algorithm that produced the best model in malicious URL detection is Random Forest in the Extended Features Set which achieved an accuracy, precision, and recall of 99.552%, 99.553%, and 99.552%, respectively. Compared to the result of the work of Cuzzocrea et al. (2018), their model using J48 yielded 92.3% in precision and 91.6% in recall. As for the related work with regards to Gain Ratio feature selection algorithm in the study of Aydin et al. (2020), with the best accuracy of 97.18% using J48. From these, this study resulted in better performance in accuracy, precision, and recall.

References

Al-Janabi M., Quincey E., and Andras P., Using supervised machine learning algorithms to detect suspicious URLs in online social networks. *In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017 (ASONAM '17)*, New York, NY, USA, 1104–1111, 2017.

Aung E., and Yamana H., URL-based Phishing Detection using the Entropy of Non-Alphanumeric Characters. *In Proceedings of the 21st International Conference on Information Integration and Web-based Applications Services*(iiWAS2019), New York, NY, USA, 385–392, 2019.

Aydin M., Butun I., Bicakci K., and Baykal N., Using Attribute-based Feature Selection Approaches and Machine Learning Algorithms for Detecting Fraudulent Website URLs. *In Proceedings of 2020 10th Annual Computing and Communication Workshop and Conference (CCWC),* Las Vegas, NV, USA, pp. 0774-0779, 2020.

CCSU, Available: https://cs.ccsu.edu/~markov/ccsu_courses/DataMining-7.html

Chapla H., Kotak R., and Joiser M., A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier. *2019 International Conference on Communication and Electronics Systems (ICCES),* pp. 383-388, 2019.

Chatterjee M., and Namin A., Detecting Phishing Websites through Deep Reinforcement Learning. *In Proceedings of 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC 2019),* WI, USA, pp. 227-232, 2019.

Cuzzocrea A., Martinelli F., and Mercaldo F.., Applying Machine Learning Techniques to Detect and Analyze Web Phishing Attacks. *In Proceedings of the 20th International Conference on Information Integration and Web-based Applications Services (iiWAS2018*), New York, NY, USA, 355–359, 2018.

Jin X., Li R., Shen X., and Bie R., Automatic web pages categorization with ReliefF and Hidden Naive Bayes, *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07 (2007).* 2007.

Kaggle, Available: https://www.kaggle.com/siddharthkumar25/malicious-and-benign-url

Liu C., Wang L., Lang B., and Zhou Y., Finding effective classifier for malicious URL detection. *In Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS 2018)* , New York, NY, USA, 240–244, 2018.

Shivangi, S., Debnath, P., Sajeevan, K., and Annapurna, D, Chrome Extension For Malicious URLs detection in Social Media Applications Using Artificial Neural Networks And Long Short Term Memory Networks. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI),* pp. 1993-1997, 2018.

Sirigineedi S., Soni J.,, and Upadhya H., Learning-based models to detect runtime phishing activities using URLs. *In Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis (ICCDA 2020),* New York, NY, USA, 102–106, 2020.

Vanhoenshoven F., Nápoles G., Falcon R., Köppen V., and Köppen M., Detecting malicious URLs using machine learning techniques. *In Proceedings of 2016 IEEE Symposium Series on Computational Intelligence (SSCI 2016),* Athens, pp. 1-8, 2016.

Yerima S., and Alzaylaee M., High Accuracy Phishing Detection Based on Convolutional Neural Networks. *In Proceedings of 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS 2020),* Riyadh, Saudi Arabia, pp. 1-6, 2020.

Acknowledgements

First of all, we would like to thank Joel De Goma for his guidance and advice throughout the entire project. He has helped us with the approvals, revisions, and insights regarding this subject matter. With this, he helped us see the clearer objective and the right direction towards our goal.

We are thankful to the people in Serpstat who assisted us in achieving the datasets for our experiments. Especially to Eugene Romanuk for assisting in providing us details with regards to subscribing to their Search Engine Optimization Tool. Moreover, we would like to thank our Customer Success Manager in the time-being of our subscription, Tanya Voronina, who helped us with the methods, API, clarifications, definitions, and other queries which assisted in the accomplishment of the features in our datasets.

We are truly grateful to Doc. Madhavi Devaraj for clarifying with the definition and purpose of several processes in our methodology. In addition, we would like to mention her reference of Ian James Recto, an expert in Machine Learning, wherein she connected us with him. From this, he assisted, clarified, and approved our processes in achieving the final results of the algorithms we used.

Biographies

Jo Simon Ambata is a graduating college student in Computer Science in the School of Information Technology Department of Mapúa University. He specializes in Artificial Intelligence in his respective program. He has been an academic scholar throughout his academic years in college. Currently, he is an intern as a Software Engineer focusing on Front-End Development in You_Source Inc. This is his first time collaborating in submitting a conference paper. His interests include software development, front-end development, project management, machine learning, artificial intelligence, and data analytics.

Jose Lean Gaurana studies at Mapúa University taking a Bachelor of Science in Computer Science and is currently at his final year waiting for graduation. He specializes in the field of Application Development and is a part of the University's Center for Student Advising. He is a recipient of the Gokongwei Brothers Foundation NextGEN Scholarship program and is currently working as a Web Development Intern at Ten Elleven Manila Inc. He plans to pursue a career in Back-End Development after graduation. His interests are Software Development, Cloud Computing, and Internet of Things.

Dan Nicole Jacinto is a graduating college student in Computer Science in the School of Information Technology Department of Mapúa University. He specializes in Artificial Intelligence in his respective program. He is also a consistent academic scholar of the university. Currently, he is taking his internship at You_Source Inc. as a Software Engineer focusing on back-end development. This is his first time collaborating in submitting a conference paper. His interests include artificial intelligence, machine learning, research, back-end development, web development, and data analytics.

Joel De Goma is a PhD student of Mapúa University in Mapúa University. He earned B.S. in Electronics Engineering, Master of Science in Electronics Engineering, and Master of Science in Computer Science from Mapúa University. He had completed research projects in the fields of neural networks, sensors, image processing, health-related, and voice recognition.