Information Sciences 484 (2019) 153–166

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A new hybrid ensemble feature selection framework for machine learning-based phishing detection system

Kang Leng Chiew[a], Choon Lin Tan[a,*], KokSheik Wong[b], Kelvin S.C. Yong[c], Wei King Tiong[a]

[a] Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Kota Samarahan, Sarawak 94300, Malaysia
[b] School of Information Technology, Monash University Malaysia, Bandar Sunway, Selangor 47500, Malaysia
[c] Department of Electrical and Computer Engineering, Faculty of Engineering and Science, Curtin University, CDT 250, Miri, Sarawak 98009, Malaysia

ARTICLE INFO

Article history:
Received 25 March 2018
Revised 22 January 2019
Accepted 24 January 2019
Available online 24 January 2019

Keywords:
Phishing detection
Feature selection
Machine learning
Ensemble-based
Classification
Phishing dataset

ABSTRACT

This paper proposes a new feature selection framework for machine learning-based phishing detection system, called the Hybrid Ensemble Feature Selection (HEFS). In the first phase of HEFS, a novel Cumulative Distribution Function gradient (CDF-g) algorithm is exploited to produce primary feature subsets, which are then fed into a data perturbation ensemble to yield secondary feature subsets. The second phase derives a set of baseline features from the secondary feature subsets by using a function perturbation ensemble. The overall experimental results suggest that HEFS performs best when it is integrated with Random Forest classifier, where the baseline features correctly distinguish 94.6% of phishing and legitimate websites using only 20.8% of the original features. In another experiment, the baseline features (10 in total) utilised on Random Forest outperforms the set of all features (48 in total) used on SVM, Naive Bayes, C4.5, JRip, and PART classifiers. HEFS also shows promising results when benchmarked using another well-known phishing dataset from the University of California Irvine (UCI) repository. Hence, the HEFS is a highly desirable and practical feature selection technique for machine learning-based phishing detection systems.

1. Introduction

Phishing is a form of cyber-attack that utilises counterfeit websites to steal sensitive user information such as account login credentials, credit card numbers, etc. Throughout the world, phishing attacks continue to evolve and gain momentum. In June 2018, the Anti-Phishing Working Group (APWG) reported as many as 51,401 unique phishing websites [3]. Another report by RSA estimated that global organisations suffered losses amounting to $9 billion due to phishing incidents in 2016 [5]. These statistics have proven that the existing anti-phishing solutions and efforts are not truly effective.

The most widely deployed anti-phishing solution is the blacklist warning system, found in conventional web browsers such as Google Chrome and Mozilla Firefox. The blacklist system queries a central database of already-known phishing URLs, thus it is unable to detect newly launched phishing websites [20,29]. A more promising and intelligent anti-phishing

* Corresponding author.
E-mail addresses: klchiew@unimas.my (K.L. Chiew), 17010025@siswa.unimas.my (C.L. Tan), wong.koksheik@monash.edu (K. Wong), kelvin.yong@curtin.edu.my (K.S.C. Yong), wktiong@unimas.my (W.K. Tiong).

https://doi.org/10.1016/j.ins.2019.01.064
0020-0255/

solution is the machine learning-based detection system. This technique capitalises on large amount of indicators, or more commonly known as *features*, giving it the flexibility to recognise new phishing websites.

The accuracy of a machine learning-based phishing detection system depends primarily on the selected features. Most anti-phishing researchers focus on proposing novel features or optimising classification algorithms [13,16], in which developing a proper feature selection and analysis technique are not their main agenda. As such, irrelevant features may still exist, increasing the cost (i.e., storage, power, training time, etc.) of a technique but not contributing to the overall accuracy. Thus, a systematic feature selection framework is needed to identify a compact set of features that are truly effective.

In general, there are two main techniques for selecting features, namely, the *wrapper* and *filter measure*. The wrapper method runs iteratively, where each run involves generating a feature subset and evaluating it via classification (i.e., training and testing). When evaluating a larger set of features, the number of iterations in wrapper method will scale up exponentially and become computationally impractical for real time applications. On the other hand, filter measures are metrics computed from statistical and information theories, which can reflect the goodness of each individual feature without invoking any particular classifier [1,28]. Thus, in this study, we focus on the filter measures, which are far less computationally intensive and hence, more practical for feature selection.

Previous phishing detection studies that utilise filter measures for feature selection are incomplete. Filter measures, on their own, can only rank the features according to importance. Obtaining a reduced feature subset requires choosing a *cut-off rank*, where only the features positioned above this cut-off rank are selected. To the best of our knowledge, optimisation of the feature cut-off rank has never been explored before in the literature. There is no systematic framework to direct filter measures-based feature selection in identifying the optimal cut-off rank.

In this paper, a novel algorithm called Cumulative Distribution Function gradient (CDF-g) is proposed as an automatic feature cut-off rank identifier. The CDF-g algorithm exploits patterns in the distribution of filter measure values to determine the optimal cut-off rank, leading to a highly effective and compact set of baseline features. To further improve the baseline features' stability and reduce overfitting, we extended the CDF-g algorithm into a hybrid ensemble structure consisting of *data perturbation* and *function perturbation* techniques. Data perturbation entails subsampling the dataset into multiple partitions, while function perturbation involves applying different filter measures on the same dataset [19]. We termed the complete feature selection framework as the Hybrid Ensemble Feature Selection (HEFS).

This paper has made the following contributions: (a) Exploiting a novel algorithm, i.e., CDF-g, using a hybrid ensemble structure to produce a stable and robust subset of features; (b) Achieving competitive detection accuracy while using reduced machine processing resources; (c) Eliminating the need for human interventions by using a fully computable technique to automatically select the most effective phishing features; (d) Providing a flexible and robust feature selection framework that can be utilised universally on different datasets; (e) Accelerating the development of accurate and efficient machine learning-based phishing detection systems by applying the proposed feature selection framework and the recommended best performing classifier; (f) Prescribing the baseline features (i.e., best feature subset) as the *de facto* standard for anti-phishing researchers to benchmark newly proposed machine learning techniques, and; (g) Constructing a large publicly accessible machine learning dataset that is highly relevant to most anti-phishing researches.

The remainder of the paper is organised as follows: Section 2 highlights the motivations of this paper. Section 3 discusses related works and their limitations on feature selection with regards to phishing detection application. Section 4 puts forward the proposed feature selection framework. Section 5 describes the experimental setup, presents the results, and discusses the major findings. Finally, Section 6 concludes this paper.

2. Motivation

Our work in this paper is driven by the following motivations:

(a) There is no existing framework to reliably determine the cut-off ranks (i.e., cut-off point) for the features ranked by filter measures, thus researchers resort to the most convenient way, which is by arbitrarily selecting a cut-off rank (e.g., top-10, top-20, etc.) [1,21,28]. Obviously, such an arbitrary selection is not an optimised technique because it is prone to overstating or understating the cut-off rank. Hence, we are motivated to explore on the optimisations of feature dimensionality.

(b) Different machine learning datasets vary in characteristics such as feature type, number of features, etc. Through our preliminary experiments, we found that the effectiveness of the existing feature selection technique in [22] and [27] are dataset dependent (i.e., not robust), and they have failed to identify the appropriate feature cut-off rank when applied on our dataset. Thus, we are inspired to study further on the filter measure values, which led to the proposal of a flexible and intelligent algorithm called CDF-g.

(c) In phishing detection, there is no common agreement on which classifier is more preferable when the objective is to maximise detection accuracy. For example, a number of studies have suggested that the Random Forest classifier outperformed other classifiers [14,18,24,25]. However, some recent papers still continue to employ other classifiers (i.e., not Random Forest) in their machine learning-based phishing detection system [7,8,15,23]. Hence, the question on "which classifier is the most predictive one" remains as a huge subject of debate. For that, we are motivated to provide more insights into this issue by performing a large scale benchmarking, where six classifiers are tested with different feature subset combinations derived from various filter measures rankings.

3. Related works

In this section, we review the recent phishing detection techniques that are based on feature selection, and evaluate different classification models.

Toolan and Carthy [28] investigated into the selection of features among 40 features, which are gathered from prior techniques designed for the detection of spam and phishing. Their primary analysis involves ranking features across 3 different datasets using Information Gain (IG), with the aim of identifying a representative set of features. By performing an intersection (i.e., AND) function over the top-10 IG rankings of the three datasets, they have identified 9 consistent features. However, no justification is provided as to why they have considered only the top-10 IG ranked features. In addition, the authors ran a C5.0 classifier over 3 feature sets with varying IG values, and showed that the feature sets with higher IG values perform better than those with lower IG values. Nevertheless, the study in [28] is incomplete, as it only assessed the performance of a single filter measure, namely IG, from a general perspective.

In Khonji et al.'s work [14], they benchmarked the performance of feature selection methods for phishing email classification. Several conventional feature evaluators are compared, which include filter measures (i.e., IG and Relief-F), Correlation-Based Feature Selection (CFS), and the wrapper method. Results showed that the wrapper method coupled with the best-first forward searching method outperforms feature subsets derived by IG and Relief-F, while CFS performs the worst among them. In addition, the experimental results also suggest that the Random Forest classifier is most predictive, and it is consistently outperforming the C4.5 and SVM classifiers. However, as highlighted in Section 1, the wrapper method is computationally expensive, which prohibits its widespread usage in machine learning-based phishing detection. Therefore, current research direction on feature selection should focus more on filter measures, which are less intensive to compute.

A similar study is conducted by Basnet et al. [4], which evaluated only two common feature selection techniques, namely, the CFS and wrapper method. The authors tested two feature space searching techniques (i.e., genetic algorithm and greedy forward selection) on features derived from the webpage itself as well as third party sources such as search engines. Performance evaluation of the feature subsets are conducted using Naive Bayes, Logistic Regression and Random Forest classifiers. Experimental results revealed that the wrapper method achieves higher detection accuracies when compared to CFS, which is consistent with the results reported in [14]. However, the authors acknowledged that the wrapper method is indeed computationally more intensive, thus preventing it from being a feasible approach in feature selection applications.

Qabajeh and Thabtah [21] assessed a total of 47 features for phishing email detection using IG, Chi-Square and CFS. For both IG and Chi-Square ranked features, the authors noticed a larger reduction of filter measure values which appeared between the 20th and 21st feature, and utilised this gap in the filter measure values as the cut-off rank to select the top 20 features as the reduced feature set. Results indicate that the detection accuracy remains stable when using the reduced feature set. However, it is unclear on how to identify the cut-off rank from a computational point of view. Further tests are conducted using 12 common features derived by intersecting the feature sets of IG, Chi-Square and CFS, achieving a decrease of only 0.28% in average accuracy when compared to the full feature set. Thus, their study has demonstrated the merits of filter measures in reducing feature dimensionality without sacrificing the classification accuracy.

Recently, Thabtah and Abdelhamid [27] exploited Chi-Square and IG to benchmark the features for phishing website detection. The authors suggested a more systematic way of identifying the cut-off ranks for features ranked by IG and Chi-Square. A threshold-based rule set is proposed, which defines the cut-off ranks as two successive features having at least 50% difference in values of IG and Chi-Square. If a cut-off rank occurs below the recommended minimum value of filter measure (i.e., 10.83 for Chi-Square and 0.01 for IG), it will be discarded. In another phishing detection work, Rajab [22] utilised the same rule set to determine cut-off ranks for their feature set evaluation. Nevertheless, the proposed rule set in [22] and [27] are not robust and may fail to identify the appropriate cut-off ranks for certain datasets when the reduction in filter measure values are more uniform. As such, the current filter measures-based feature selection studies in the phishing detection field are lacking a systematic approach to identify the optimal cut-off rank.

4. The proposed feature selection framework

Fig. 1 shows an overview of the proposed feature selection framework. For better understanding, we employ a top-down presentation approach, where the major components and processes in Fig. 1 are firstly introduced without going in depth on details of the related algorithm. At this stage, it is sufficient to treat the CDF-g component as a black box that functions as a feature cut-off rank identifier (detailed explanation on CDF-g component is presented in Section 4.2).

4.1. The hybrid ensemble strategy

In the field of feature selection, there are two major types of ensemble techniques, namely data perturbation and function perturbation. Data perturbation applies the same feature selection technique on multiple subsets of a dataset, while function perturbation applies multiple feature selection techniques on the same set of data. Combining both data perturbation and function perturbation results in a hybrid perturbation ensemble technique. A number of studies have suggested that the ensemble strategy could be a promising approach to improve classification performance and to obtain a more stable subset of features [6,19,21,28]. Recently, Hadi et al. [12] also explored the benefit of dataset partitioning concept and agreed on its potential in reducing the complexity of the classifier. Thus, these studies have motivated us to propose the Hybrid Ensemble

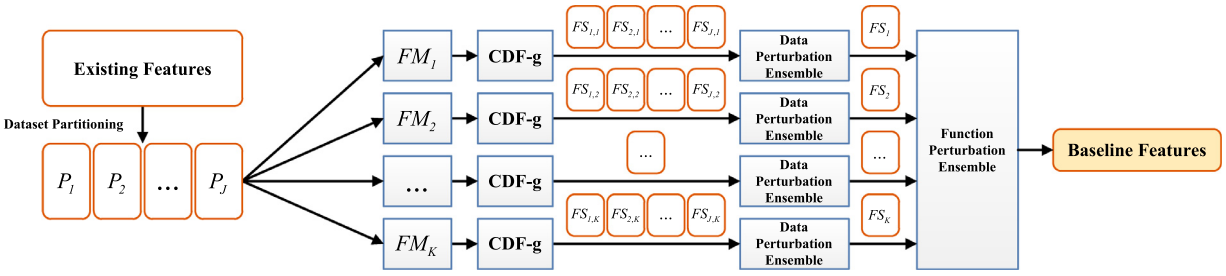

Fig. 1. Overview of the proposed feature selection framework.

Feature Selection (HEFS) framework, which consists of two cycles, namely, data perturbation cycle and function perturbation cycle.

Formally, the proposed HEFS framework is described as follows. Let the j-th partition and k-th filter measure be denoted as P_j and FM_k, respectively, as shown in Fig. 1. We begin by describing the process flow of the data perturbation cycle. The full dataset (i.e., complete samples and complete features) are randomised before being divided by stratified subsampling into J partitions. Randomisation refers to the shuffling of the sample rows, and is utilised to ensure that the samples in each partition are more balanced.

For a dataset partition P_j, the filter measure FM_k computes the filter measure values based on the raw feature values in that partition. The filter measure values are sorted and passed to the next stage, where the CDF-g algorithm is then applied to produce the feature cut-off rank, $\tau_{j,k}$. After repeating the aforementioned procedure for $j = 1, 2,..., J$, a list of feature cut-off ranks $\{\tau_{1,k}, \tau_{2,k},..., \tau_{J,k}\}$ is created respectively for each dataset partition by using filter measure FM_k. The mean of the feature cut-off ranks, i.e., $\bar{\tau}_k$, is obtained using Eq. (1), and referenced on $\{P_1, P_2,..., P_J\}$ to generate an array of primary feature subsets $\{FS_{1,k}, FS_{2,k},..., FS_{J,k}\}$. Specifically, each primary feature subset $FS_{j,k}$ consists of the top-$\bar{\tau}_k$ features in P_j that are sorted according to filter measure values. The data perturbation ensemble component will apply an intersection operation, as shown in Eq. (2), on the array of primary feature subsets to produce a secondary feature subset FS_k. In this intersection operation, only features that are common among the primary feature subsets are retained. By considering the consensus of all partitions, it helps to discard any irregular features that may exist in certain partitions. A feature that is truly predictive is most guaranteed to exist in all dataset partitions, thus enabling it to be selected (through the intersection operation) as part of the secondary feature subset. This concludes a single data perturbation cycle.

$$\bar{\tau}_k = \frac{1}{J}\sum_{j=1}^{J}\tau_{j,k}\,. \tag{1}$$

$$FS_k = \bigcap_{j=1}^{J} FS_{j,k}\,. \tag{2}$$

On the other hand, the function perturbation cycle can be defined as running the data perturbation cycle over K different filter measures, which produces an array of secondary feature subsets $\{FS_1, FS_2,..., FS_K\}$ that is ready to be fed into the function perturbation ensemble component. The function perturbation ensemble component aggregates its inputs via a union operation, as shown in Eq. (3), to obtain the best feature subset, i.e., baseline feature set. Through the function perturbation cycle, the intelligence of different filter measures can be leveraged, thus leading to the baseline feature set that is less susceptible to overfitting.

$$\text{Baseline feature set} := \bigcup_{k=1}^{K} FS_k\,. \tag{3}$$

4.2. CDF-g algorithm — An automatic feature cut-off rank identifier

The goal of selecting subset of features in machine learning-based phishing detection is to reduce the feature space without compromising on the detection accuracy. In feature subset selection approaches that utilise filter measures, determining an optimal cut-off rank is crucial in achieving the aforementioned goal. The cut-off rank is defined as the specific threshold rank in an ordered list of filter measure values, where any features located beyond the cut-off rank is considered irrelevant and therefore discarded. Fig. 2 illustrates the cut-off rank τ, where the selected feature subset is surrounded by a discontinuous rectangle. As highlighted in Sections 2 and 3, the existing cut-off rank identification techniques [21,22,27] is found to be unstable and not robust. Hence, it is highly desirable to devise a new approach that is more flexible to determine the optimal cut-off rank. In this section, we present a novel algorithm named CDF-g, which aims at overcoming the limitations of existing feature cut-off rank identification techniques.

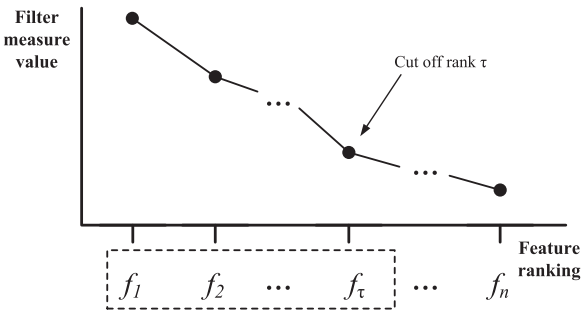

Fig. 2. Feature subset selection using cut-off rank.

4.2.1. CDF-g concepts and definitions

In this section, we provide a theoretical background on Cumulative Distribution Function (CDF) and describe how it can be adapted in our proposed feature selection algorithm. Let X be a discrete random variable and x represents a possible value of the random variable. The probability that the discrete random variable X takes on a particular value x is typically called the probability density function or the frequency function, as expressed below:

$$P(X = x)\,. \tag{4}$$

The CDF of the random variable X is defined as Eq. (5).

$$F_X(t) = P(X \leq t)\,. \tag{5}$$

The notation $F_X(t)$ is a function of t that represents the CDF for the random variable X.

Adapting the CDF in our feature selection context, the discrete random variable is assumed to represent the value of filter measure computed for each feature. Thus, in this paper, we leveraged the CDF curve to visualise the cumulative distribution of filter measure values. By analysing the gradient changes in the CDF curve, we can identify *plateau* regions along the range of filter measure values. Plateau regions are the sections of a curve with zero gradients, which indicates that the successive values have relatively larger gaps. In the context of our feature selection work, a plateau region within a CDF curve of filter measure values represents separation between two feature subsets that differ significantly in terms of predictive power. As such, our approach exploits this concept to pinpoint the cut-off rank of the top ranked features. We termed the aforementioned CDF gradient approach as CDF-g.

To compute the gradient at any point R_i on the CDF curve, we used the central differences for the interior points, and one-side (forward or backwards) differences for the boundaries, as shown in Eq. (6):

$$\text{Gradient, } G(R_i) = \begin{cases} \dfrac{F_X(t_{i+1}) - F_X(t_i)}{h}, & \text{if } i = 1\,; \\ \dfrac{F_X(t_{i+1}) - F_X(t_{i-1})}{2h}, & \text{if } 1 < i < n\,; \\ \dfrac{F_X(t_i) - F_X(t_{i-1})}{h}, & \text{if } i = n\,; \end{cases} \tag{6}$$

where the default distance $h = 1.0$, and n denotes the number of points on the CDF curve.

The pseudo-code of applying CDF-g to identify the feature cut-off rank is shown in Algorithm 1, and we further explained it through an example. For instance, using IG as filter measure, we compute the values of IG for the full set of features. The computed values are then normalised and sorted in increasing order. Next, a CDF curve is calculated and plotted using the sorted values. Based on the gradient of the CDF curve, the first plateau located above 50% of the cumulative value is identified and its corresponding normalised IG value is mapped to the features ranking to obtain the cut-off rank, as shown by the red vertical dotted line in Fig. 3. The scatter plot in Fig. 3 provides a clearer understanding on the relationship between the cut-off rank and the features distribution along the range of normalised IG values.

In plotting the complete CDF curve, it is necessary to set the number of uniform intervals called *bins*. For example, if $bins = 15$ and $1.2 \leq x_i \leq 12.3$, the plot points on the CDF curve are computed by running Eq. (5) over $t = \{1.20, 1.94, 2.68, \ldots, 12.30\}$. In our study, we empirically set the number of *bins* to be twice the number of features and 50% as the threshold for locating the first plateau, where these parameters has provided optimal trade-offs between feature space reduction and classification accuracy.

4.3. Features preparation

Features utilised in existing phishing website detection studies can be categorised into two major types, namely: (a) internal features, and; (b) external features. Internal features are derived from webpage URL and HTML source codes, all of which can be directly acquired from the webpage itself. On the other hand, external features came from querying third

Algorithm 1 Feature cut-off rank identification using CDF-g.

Input: V, b ▷ V = Feature vector file, b = Bin size
Output: τ ▷ τ = Feature cut-off rank

```
Begin
FM_values = ComputeFilterMeasure(V)
FM_values = Normalise(FM_values)
FM_values = Sort(FM_values)
CDF_values = ComputeCDF(FM_values, b)
for all c ∈ CDF_values do
    if c > 0.5 then
        CDF_gradient = ComputeGradient(c)
        if CDF_gradient = 0.0 then
            τ = MapToFeatureRanking(c)
            break
        end if
    end if
end for
End
```

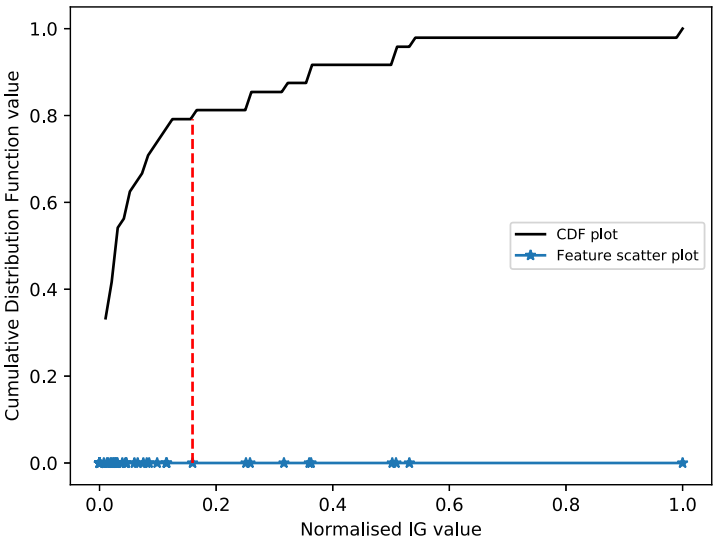

Fig. 3. Example of locating the plateau on the CDF curve of IG values.

party services such as domain registry, search engine, WHOIS records, etc. In this work, our benchmarking focuses solely on internal features. The reasons for excluding external features are threefold:

(a) The webpage URL and HTML source code is the most basic data available in phishing datasets. Using commonly accessible features ensures that our study is relevant to most anti-phishing researchers.
(b) External data are highly volatile. For example, the results of search engines change frequently.
(c) Access to external resources is unstable and unpredictable. For example, the public API for querying the Google PageRank metric was officially deprecated in 2016, thus affecting many phishing detection techniques [10,23,30] that depend on the PageRank feature.

To automate our feature extraction, we leverage on the Selenium WebDriver[1], a script-based automation framework for browser. A Python script containing predefined parsing rules is executed, which in turn will direct the browser to load the webpage. Next, the parsing rules will search the rendered webpage content, extract the feature value, and save it to a text file. The advantage of using an actual browser in feature extraction is to add robustness against variations in HTML content compared to parsing approach based on regular expressions. In addition, it closely emulates the environment where users normally interact with websites.

[1] http://www.seleniumhq.org/projects/webdriver/

Overall, a total of 48 features are extracted from webpage URLs and HTML sources codes. These features were selected after a comprehensive review of the related works on machine learning-based phishing website detection. It is based on 11 research papers published between year 2007 and 2016. A detailed listing of the features is provided in the Appendix section.

5. Results analysis and discussions

5.1. Dataset preparation

In order to gather the required features, we collected websites on two separate sessions, i.e., between January to May 2015 and between May to June 2017. Specifically, we selected 5000 phishing webpages based on URLs from PhishTank[2] and OpenPhish[3], and another 5000 legitimate webpages based on URLs from Alexa[4] and the Common Crawl[5] archive.

The collection of webpages is automated by using the GNU Wget[6] tool and Python script. Apart from the complete HTML documents, we also downloaded the related resources (e.g., images, CSS, JavaScript) to ensure all downloaded webpages can be rendered properly in the browser. In addition, the screenshot of every webpage is stored for further inspection and filtering.

The downloaded dataset were further processed to remove defective webpages which failed to load or "Error 404" pages in both the phishing and legitimate dataset. Duplicate instances of webpages are also removed. After filtering the samples, feature extraction is performed based on the descriptions in Section 4.3. The complete dataset is available for download as the Waikato Environment for Knowledge Analysis (Weka[7]) machine learning software's Attribute-Relation File Format (ARFF) file at [26].

5.2. Experimental setup

Experiment processes such as calculating the filter measure values and running classifications (i.e., training and testing) are performed using Weka. It should be noted that all the classifiers deployed are using the default parameters as specified by Weka. Other studies have explored the benefits of fine tuning or optimising these parameters. However, our aim is to show the impact of feature selection and assess its effect over different types of classifiers, without going in depth of fine tuning the classifiers parameters. The proposed CDF-g algorithm is implemented in Python programming language using the third party package called NumPy[8].

Since we have constructed a large size dataset, we set the total number of dataset partitions $J = 10$. This is because each obtained partitions can still have reasonable amount of phishing and legitimate webpage samples to properly train a classifier. As for the number of filter measures to be utilised for function perturbation, we set $K = 3$, to prevent the final baseline feature set from being inflated by conflicting features due to differences in filter measure rankings.

In running the experiments, we followed a classification approach similar to the train-test split. For each partition, 70% of the data is utilised for training while 30% is reserved for testing purpose. The classification accuracy for each partition is computed as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7}$$

where TP, TN, FP and FN denote true positive, true negative, false positive and false negative, respectively. Lastly, the accuracies for all partitions are averaged to produce a single score. Experiments are conducted on a desktop computer equipped with an Intel Core i5 2.5GHz CPU, 6GB RAM and Windows 7 Home Premium 64-bit operating system.

5.3. Evaluation I — Performance of various classifiers on different feature subsets derived from filter measures rankings

In this section, we evaluate the detection performance of various classifiers on incremental feature subsets. Classifiers that are commonly found in Weka are selected, namely, SVM, Naive Bayes, C4.5, Random Forest, JRip, and PART. The filter measures used to rank the features and generate the incremental feature subsets are Information Gain, Relief-F, Symmetrical Uncertainty, Chi-Square, Gain Ratio, and Pearson Correlation Coefficient. Results are presented subsequently in Figs. 4, 5, 6, 7, 8, and 9.

Based on the results, the performances of different classifiers can be seen to converge and stabilise at different stable states of accuracy. Here, stable state of accuracy is defined as the maximum accuracy level that remains constant even as more features are being added. Therefore, it is imperative to identify the optimal top-n feature subset, which is the bare

[2] https://www.phishtank.com/
[3] https://www.openphish.com/
[4] https://www.alexa.com/
[5] http://commoncrawl.org/
[6] https://www.gnu.org/software/wget/
[7] https://www.cs.waikato.ac.nz/ml/weka/
[8] https://pypi.python.org/pypi/numpy/

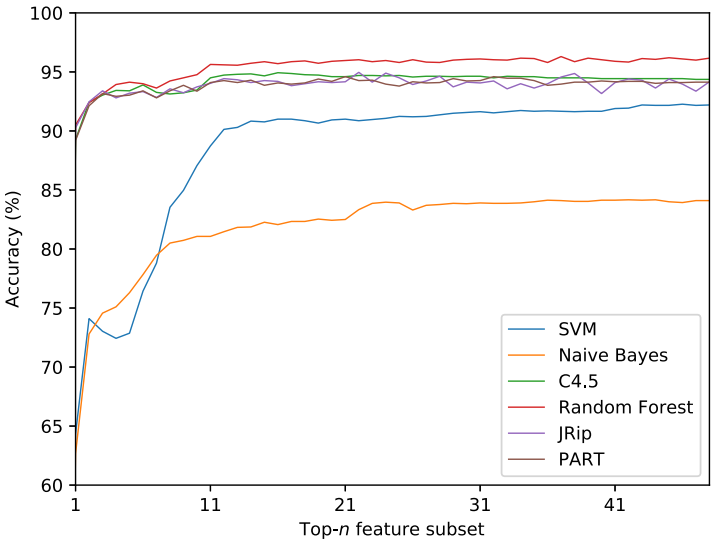

Fig. 4. Performance of Chi-Square ranked feature subsets.

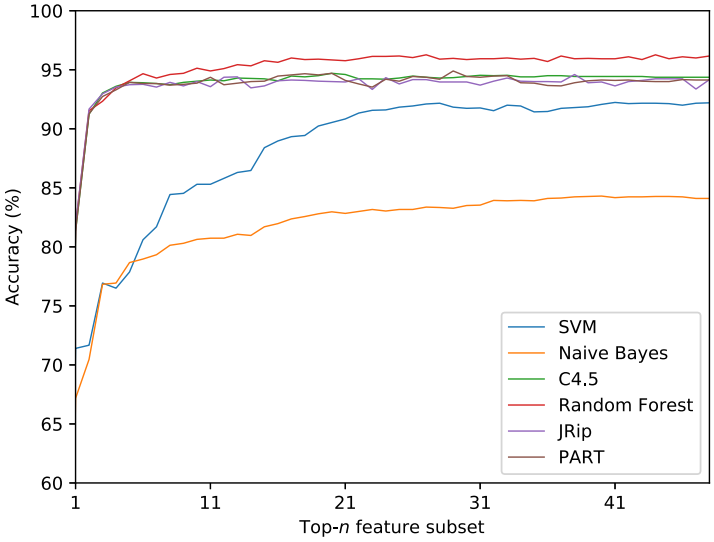

Fig. 5. Performance of Gain Ratio ranked feature subsets.

minimum number of features needed to achieve the stable state of accuracy that a specific classifier can offer. As such, the novel CDF-g algorithm is proposed to meet this demand by automatically determining the optimal top-*n* feature subset without any prior classifications, where the results are presented later in Section 5.5.

Comparing the stable states of accuracy among the classifiers, Random Forest appears to consistently outperform all remaining classifiers by scoring above 95% accuracy. Therefore, our finding here is consistent with the earlier studies in [14,18,24,25] which also claim that the best classifier in terms of accuracy is Random Forest.

5.4. Evaluation II — Fitness of filter measure with selected classifier

Capitalising on the best classifier recommendation in Evaluation I, another experiment is conducted to analyse the effects of using different filter measures with Random Forest. From this experiment, we provide insights on which filter measure is the fittest one to be utilised with Random Forest. Here, the fitness of a filter measure is gauged by its initial accuracy (i.e., accuracy of the top-1st feature ranked by the filter measure) and how fast it reaches the stable state of accuracy when the features are incrementally added according to the ranking of that filter measure. The results are shown in Fig. 10.

Based on Fig. 10, Information Gain and Chi-Square performed the best by attaining the same initial accuracy at 90.5%, followed by Symmetrical Uncertainty at 88.9%. The favourable results of Information Gain could be attributed to the fact

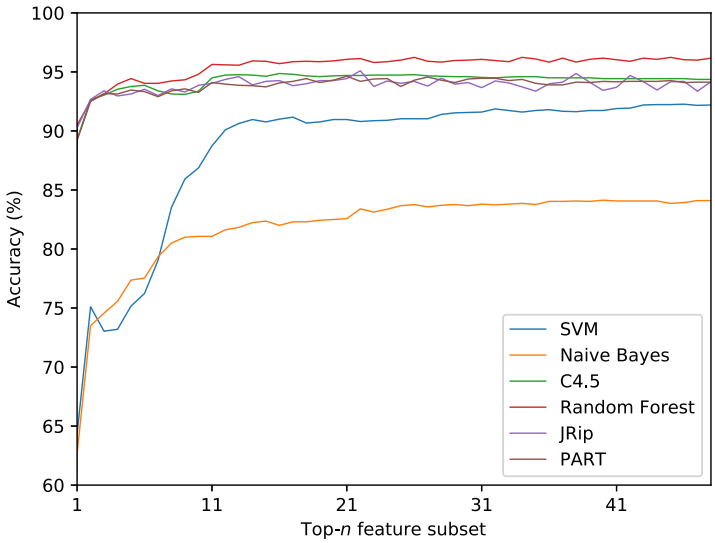

Fig. 6. Performance of Information Gain ranked feature subsets.

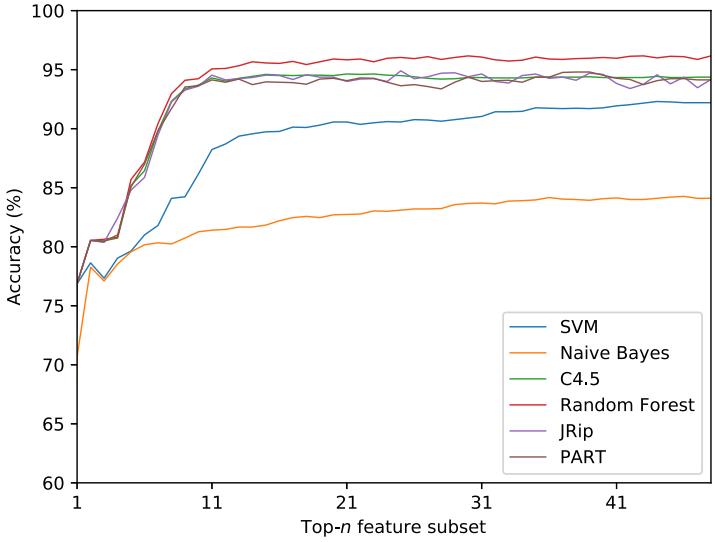

Fig. 7. Performance of Pearson Correlation Coefficient ranked feature subsets.

that the Random Forest classifier and Information Gain are both founded on similar information metrics, i.e., the entropy. Our results also correlate with [21], where the performance of Information Gain and Chi-Square are found to be comparable.

On the other hand, Gain Ratio achieved relatively low initial accuracy at 81.8%. The least fit filter measures with Random Forest seems to be Relief-F and Pearson Correlation Coefficient, which only achieved 76.9% of initial accuracy. The performance of Relief-F is consistent with the results reported in [14], where feature subsets based on Relief-F exhibited higher error rates as compared to Information Gain when the Random Forest classifier is used.

Therefore, based on the analysis results, it is reasonable to conclude that Information Gain, Chi-Square, and Symmetrical Uncertainty are the top-3 fittest filter measures to be used in combination with Random Forest classifier.

5.5. Evaluation III — Performance of baseline features

Through the proposed CDF-g algorithm and HEFS framework, we managed to identify a set of baseline features (i.e., best feature subset) that capitalises on up-to-date phishing schemes and contributes significantly towards the phishing detection rate. This evaluation aims to benchmark the performance of the baseline feature set against the full feature sets. Based on the finding from Evaluation II, we have selected Chi-Square, Information Gain, and Symmetrical Uncertainty as the filter measures in the function perturbation cycle. The final baseline feature set contains 10 features, which are listed in Table 1.

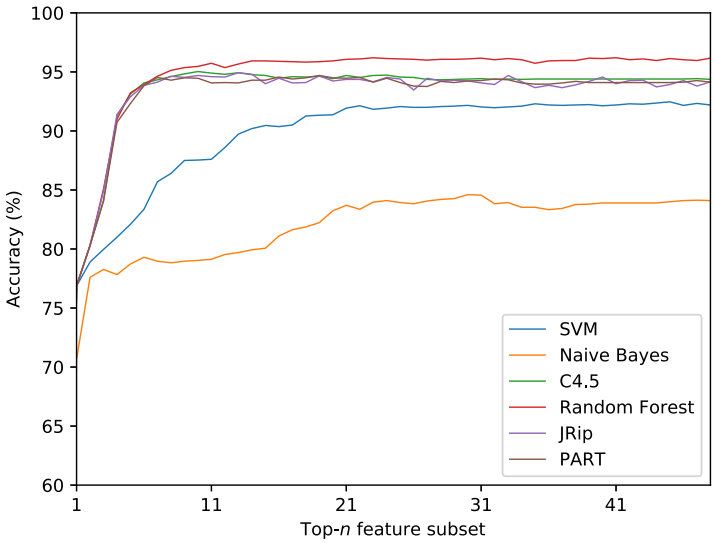

Fig. 8. Performance of Relief-F ranked feature subsets.

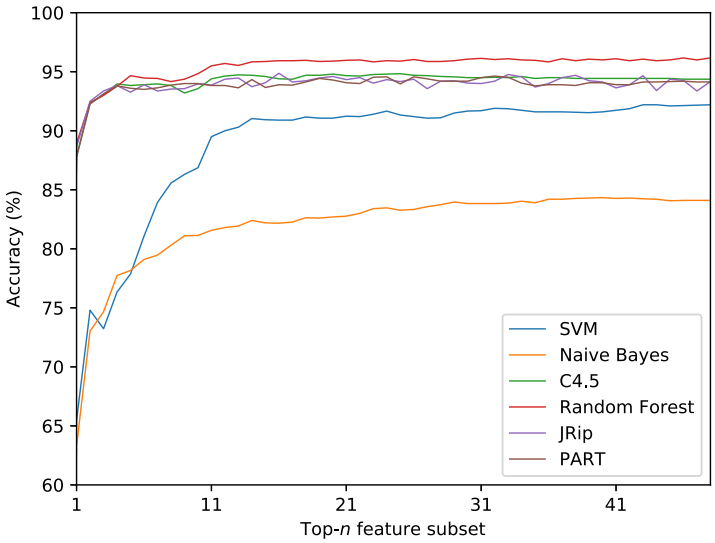

Fig. 9. Performance of Symmetrical Uncertainty ranked feature subsets.

Evaluation results shown in Table 2 suggest that the baseline features using Random Forest outperform all full feature sets that use C4.5, JRip, PART, SVM, and Naive Bayes. This proves that the derived baseline features, when coupled with Random Forest classifier, are highly effective in distinguishing between phishing and legitimate websites. When comparing baseline and full features on the same classifier (Random Forest), the baseline features only experienced a minimal accuracy deterioration of 1.57% while achieving a massive 79.2% reduction in feature space. In short, the evaluation results have validated the effectiveness of the proposed HEFS framework, which yielded a highly effective set of baseline features. The baseline features can be prescribed as the *de facto* standard for the anti-phishing researchers to evaluate newly proposed machine learning techniques.

A brief analysis on the baseline features has revealed interesting findings. It is surprising that some frequently promoted features in existing phishing detection studies are not chosen as the baseline features by our proposed feature selection approach. Currently, features such as *NumDots, UrlLength, AtSymbol, NoHttps*, and *IpAddress* are still regarded as essential features for phishing detection. However, this study has shown otherwise, and indeed the use of these features only contribute little to the accuracy of a phishing detection technique. This is potentially due to the evolution of phishing trends in recent years, where phishers are employing new schemes to evade detection. Hence, these features have eventually become obsolete. As such, our findings in this evaluation should serve as an urgent notice to anti-phishing researchers to discontinue the use of these outdated features.

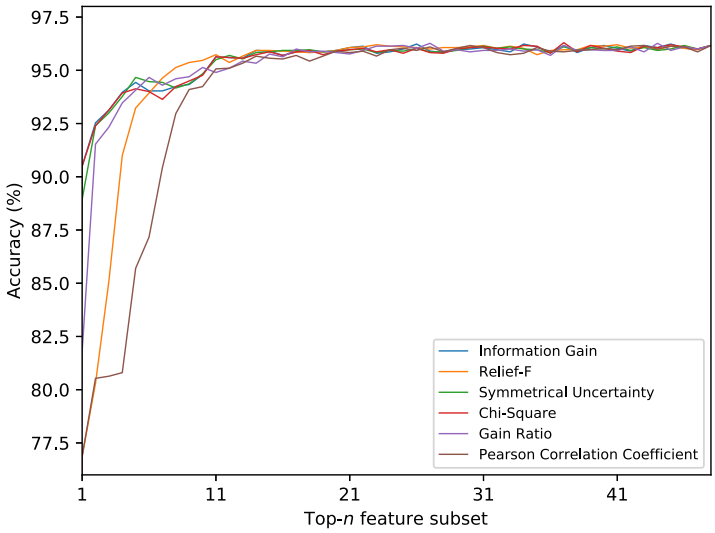

Fig. 10. Performance of Random Forest on different filter measure ranked features.

Table 1
Baseline feature set.

FrequentDomainNameMismatch	*PctNullSelfRedirectHyperlinks*
PctExtNullSelfRedirectHyperlinks	*PctExtResourceUrlsRT*
NumNumericChars	*ExtMetaScriptLinkRT*
PctExtHyperlinks	*SubmitInfoToEmail*
NumDash	*NumSensitiveWords*

Table 2
Performance comparison between baseline and full features.

Classifier	Feature set	Number of features	Accuracy (%)
Random Forest	Full	48	96.17
Random Forest	Baseline	10	94.60
C4.5	Full	48	94.37
JRip	Full	48	94.17
PART	Full	48	94.13
SVM	Full	48	92.20
Naive Bayes	Full	48	84.10

Table 3
Average runtime per sample for classification phase.

Feature selection technique	Number of features	Classification time (ms)
Chi-Square	37	0.0670
Information Gain	34	0.0673
Symmetrical Uncertainty	46	0.0783
HEFS	10	0.0523

5.6. Evaluation IV — Runtime analysis comparison

Evaluation is conducted to benchmark the runtime performance of HEFS against existing feature selection techniques, namely, the filter measures. Note that for each respective filter measure, we only report the runtime performance of its best feature subset that achieved the highest accuracy.

Table 3 shows the average classification time (i.e., testing time) per sample, where 1500 phishing and 1500 legitimate data samples are evaluated. Results indicate that the baseline feature set produced using HEFS has the lowest testing time, which is reasonable since reducing the number of features will also reduce the computation complexity of the classifier. Thus, the proposed HEFS framework is empirically shown to be advantageous when dealing with large datasets and real time applications.

Table 4
Performance benchmarking between the proposed HEFS framework and FACA [11].

Classifier	Feature set	Number of features	Accuracy (%)
FACA	Full	30	92.40
Random Forest	Full	30	94.27
Random Forest	Baseline (derived from HEFS)	5	93.22

5.7. Evaluation V — Benchmarking on other machine learning dataset

To further validate the effectiveness of the proposed technique, the HEFS framework is benchmarked with a recent phishing detection technique by Hadi et al. [11]. The authors proposed a new classification algorithm called Fast Associative Classification Algorithm (FACA) and evaluated it on a large phishing dataset from the UCI machine learning repository [17]. The dataset, which consists of 11,055 instances and 30 different features, has been widely utilised for benchmarking purposes in phishing detection research. Further details on the features and dataset descriptions are given in [17]. The motivation to select [11] for benchmarking is two-fold: (a) Associative Classification (AC) algorithm employs built-in feature selection process in the form of rule pruning, which is highly relevant to our proposed feature selection framework, and; (b) AC is a promising and competitive classification approach that can be considered as one of the state-of-the-art classification techniques [1,2,12].

The benchmarking results shown in Table 4 suggest that our proposed HEFS framework outperforms the feature selection and classification model in [11] while effectively reducing up to 83.33% in feature dimensionality. In other words, assuming that FACA has indeed performed feature selection (in the form of rule pruning), its accuracy when tested on the full features (30 in total) is still inferior compared to the baseline features (5 in total) derived by using our proposed HEFS framework. The suboptimal performance of FACA may imply that its classification algorithm is less powerful, or it may also suggest that the feature selection (rule pruning) in FACA is not truly effective. In addition, the promising results in this evaluation also suggest that the proposed HEFS is robust and thus, can flexibly adapt to different datasets.

6. Conclusion and future work

In this work, a new feature selection framework called HEFS is proposed, where existing filter measures are leveraged to find an effective subset of features for utilisation in machine learning-based phishing detection. As part of the HEFS, we propose a novel algorithm called CDF-g to automatically determine the optimal number of features. The CDF-g algorithm works by exploiting patterns in the distribution of filter measure values, thus enabling it to flexibly adapt to different datasets. The CDF-g is extended and integrated in a hybrid ensemble structure to yield a highly effective set of baseline features. Results suggest that the baseline features perform best when integrated with Random Forest classifier, achieving a competitive accuracy of 94.6% using only 20.8% of the original number of features. In addition, the baseline features utilised on Random Forest outperforms full set features used on SVM, Naive Bayes, C4.5, JRip, and PART classifiers. Thus, the proposed feature selection framework is proven to be more advantageous than the current approaches, which tend to be computationally expensive. The practical implications of this research include: (a) enhancing the detection accuracy and computational efficiency of machine learning-based phishing detection systems; (b) providing a fully automatic, flexible and robust feature selection framework that can be utilised universally on different datasets to produce highly effective optimal feature subset; (c) prescribing a set of baseline features for future benchmarking of machine learning-based anti-phishing techniques, and; (d) accelerating the development of powerful and compact phishing detection system by applying the proposed feature selection framework and the recommended best performing classifier.

In future, it could be worthwhile to investigate the impact of feature selection using other classification algorithms such as associative classification. Researchers may also explore on feature transformation as another viable feature reduction approach. In addition, future anti-phishing works should consider to discontinue the use of some obsolete features as reported in Section 5.5, which are found to be no longer effective.

Acknowledgements

The funding for this project is made possible through the research grant obtained from UNIMAS under the Dana Pelajar PhD [Grant No: F08/DPP/1649/2018]. This work is also supported by the Sarawak Foundation Tun Taib Scholarship Scheme. A sincere thank you to Gladys Yee-Kim Tan for her diligent proofreading and language assistance on this paper.

Appendix A

Table A1
Phishing website features.

No.	Identifier	Value type	Description
1	*NumDots*	Discrete	Counts the number of dots in webpage URL [1,31].
2	*SubdomainLevel*	Discrete	Counts the level of subdomain in webpage URL [13,30]
3	*PathLevel*	Discrete	Counts the depth of the path in webpage URL [9].
4	*UrlLength*	Discrete	Counts the total characters in the webpage URL [16,17].
5	*NumDash*	Discrete	Counts the number of "-" in webpage URL [9,16,17,30].
6	*NumDashInHostname*	Discrete	Counts the number of "-" in hostname part of webpage URL [9,16,17,30].
7	*AtSymbol*	Binary	Checks if "@" symbol exist in webpage URL [9,13,16,17,30].
8	*TildeSymbol*	Binary	Checks if "∼" symbol exist in webpage URL [8].
9	*NumUnderscore*	Discrete	Counts the number of "_" in webpage URL [9].
10	*NumPercent*	Discrete	Counts the number of "%" in webpage URL [8].
11	*NumQueryComponents*	Discrete	Counts the number of query parts in webpage URL [8].
12	*NumAmpersand*	Discrete	Counts the number of "&" in webpage URL [8].
13	*NumHash*	Discrete	Counts the number of "#" in webpage URL [8].
14	*NumNumericChars*	Discrete	Counts the number of numeric characters in the webpage URL [9].
15	*NoHttps*	Binary	Checks if HTTPS exist in webpage URL [9,13,15–17].
16	*RandomString*	Binary	Checks if random strings exist in webpage URL [8].
17	*IpAddress*	Binary	Checks if IP address is used in hostname part of webpage URL [10,13,16,17,30].
18	*DomainInSubdomains*	Binary	Checks if TLD or ccTLD is used as part of subdomain in webpage URL [30].
19	*DomainInPaths*	Binary	Checks if TLD or ccTLD is used in the path of webpage URL [13,17,30].
20	*HttpsInHostname*	Binary	Checks if HTTPS in obfuscated in hostname part of webpage URL.
21	*HostnameLength*	Discrete	Counts the total characters in hostname part of webpage URL [15].
22	*PathLength*	Discrete	Counts the total characters in path of webpage URL [15].
23	*QueryLength*	Discrete	Counts the total characters in query part of webpage URL [15].
24	*DoubleSlashInPath*	Binary	Checks if "//" exist in the path of webpage URL [23].
25	*NumSensitiveWords*	Discrete	Counts the number of sensitive words (i.e., "secure", "account", "webscr", "login", "ebayisapi", "signin", "banking", "confirm") in webpage URL [10,30].
26	*EmbeddedBrandName*	Binary	Checks if brand name appears in subdomains and path of webpage URL [30]. Brand name here is assumed as the most frequent domain name in the webpage HTML content.
27	*PctExtHyperlinks*	Continuous	Counts the percentage of external hyperlinks in webpage HTML source code [13,16,17].
28	*PctExtResourceUrls*	Continuous	Counts the percentage of external resource URLs in webpage HTML source code [13,16,17].
29	*ExtFavicon*	Binary	Checks if the favicon is loaded from a domain name that is different from the webpage URL domain name [17].
30	*InsecureForms*	Binary	Checks if the form action attribute contains a URL without HTTPS protocol [30].
31	*RelativeFormAction*	Binary	Checks if the form action attribute contains a relative URL [30].
32	*ExtFormAction*	Binary	Checks if the form action attribute contains a URL from an external domain [16,17].
33	*AbnormalFormAction*	Categorical	Check if the form action attribute contains a "#", "about:blank", an empty string, or "javascript:true" [17].
34	*PctNullSelfRedirectHyperlinks*	Continuous	Counts the percentage of hyperlinks fields containing empty value, self-redirect value such as "#", the URL of current webpage, or some abnormal value such as "file://E:/" [13,17,30].
35	*FrequentDomainNameMismatch*	Binary	Checks if the most frequent domain name in HTML source code does not match the webpage URL domain name.
36	*FakeLinkInStatusBar*	Binary	Checks if HTML source code contains JavaScript command onMouseOver to display a fake URL in the status bar [16,17].
37	*RightClickDisabled*	Binary	Checks if HTML source code contains JavaScript command to disable right click function [16,17].
38	*PopUpWindow*	Binary	Checks if HTML source code contains JavaScript command to launch pop-ups [8,16,17].
39	*SubmitInfoToEmail*	Binary	Check if HTML source code contains the HTML "mailto" function [17].
40	*IframeOrFrame*	Binary	Checks if iframe or frame is used in HTML source code [17].
41	*MissingTitle*	Binary	Checks if the title tag is empty in HTML source code [8].
42	*ImagesOnlyInForm*	Binary	Checks if the form scope in HTML source code contains no text at all but images only.
43	*SubdomainLevelRT*	Categorical	Counts the number of dots in hostname part of webpage URL. Apply rules and thresholds to generate value [16].
44	*UrlLengthRT*	Categorical	Counts the total characters in the webpage URL. Apply rules and thresholds to generate value [16,17].
45	*PctExtResourceUrlsRT*	Categorical	Counts the percentage of external resource URLs in webpage HTML source code. Apply rules and thresholds to generate value [17].
46	*AbnormalExtFormActionR*	Categorical	Check if the form action attribute contains a foreign domain, "about:blank" or an empty string. Apply rules to generate value [17].
47	*ExtMetaScriptLinkRT*	Categorical	Counts percentage of meta, script and link tags containing external URL in the attributes. Apply rules and thresholds to generate value [17].
48	*PctExtNullSelfRedirectHyperlinksRT*	Categorical	Counts the percentage of hyperlinks in HTML source code that uses different domain names, starts with "#", or using "JavaScript ::void(0)". Apply rules and thresholds to generate value [17].

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.01.064

References

[1] N. Abdelhamid, A. Ayesh, F. Thabtah, Phishing detection based associative classification data mining, Expert Syst. Appl. 41 (13) (2014) 5948–5959.
[2] M. Aburrous, M.A. Hossain, K. Dahal, F. Thabtah, Associative Classification Techniques for predicting e-Banking Phishing Websites, in: Proceedings of the International Conference on Multimedia Computing and Information Technology (MCIT), Sharjah, United Arab Emirates, 2010, pp. 9–12.
[3] Anti-Phishing Working Group, Phishing Activity Trends Report, 2nd Quarter 2018, 2018, Retrieved from http://docs.apwg.org/reports/apwg_trends_report_q2_2018.pdf, accessed 16.11.18.
[4] R.B. Basnet, A.H. Sung, Q. Liu, Feature selection for improved phishing detection, in: Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), Dalian, China, 2012, pp. 252–261.
[5] H. Bleau, 2017 Global Fraud and Cybercrime Forecast, 2016, Retrieved from https://www.rsa.com/en-us/blog/2016-12/2017-global-fraud-cybercrime-forecast, accessed 02.11.17.
[6] V. Bolón-Canedo, K. Sechidis, N. Sánchez-Maroño, A. Alonso-Betanzos, G. Brown, Insights into distributed feature ranking, Inf. Sci. (Ny) (2018).
[7] K.L. Chiew, E.H. Chang, S.N. Sze, W.K. Tiong, Utilisation of website logo for phishing detection, Comp. Secur. 54 (2015) 16–26.
[8] X.M. Choo, K.L. Chiew, D.H.A. Ibrahim, N. Musa, S.N. Sze, W.K. Tiong, Feature-based phishing detection technique, J. Theor. Appl. Inf. Technol. 91 (1) (2016) 101–106.
[9] H.M.A. Fahmy, S. Ghoneim, PhishBlock: A hybrid anti-phishing tool, in: Proceedings of the International Conference on Communications, Computing and Control Applications (CCCA), Hammamet, Tunisia, 2011, pp. 1–5.
[10] S. Garera, N. Provos, M. Chew, A.D. Rubin, A Framework for Detection and Measurement of Phishing Attacks, in: Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM), Alexandria, Virginia, USA, 2007, pp. 1–8.
[11] W. Hadi, F. Aburub, S. Alhawari, A new fast associative classification algorithm for detecting phishing websites, Appl. Soft Comput. 48 (2016) 729–734.
[12] W. Hadi, G. Issa, A. Ishtaiwi, ACPRISM: Associative classification based on PRISM algorithm, Inf. Sci. (Ny) 417 (2017) 287–300.
[13] M. He, S.-J. Horng, P. Fan, M.K. Khan, R.-S. Run, J.-L. Lai, R.-J. Chen, A. Sutanto, An efficient phishing webpage detector, Expert Syst. Appl. 38 (10) (2011) 12018–12027.
[14] M. Khonji, A. Jones, Y. Iraqi, An empirical evaluation for feature selection methods in phishing email classification, Comp. Syst. Sci. Eng. 28 (1) (2013) 37–51.
[15] M. Moghimi, A.Y. Varjani, New rule-based phishing detection method, Expert Syst. Appl. 53 (2016) 231–242.
[16] R.M. Mohammad, F. Thabtah, L. McCluskey, An assessment of features related to phishing websites using an automated technique, in: Proceedings of the International Conference for Internet Technology and Secured Transactions, London, UK, 2012, pp. 492–497.
[17] R.M. Mohammad, F. Thabtah, L. McCluskey, Phishing Websites Data Set, 2015, Retrieved from http://archive.ics.uci.edu/ml/datasets/Phishing+Websites, accessed 02.11.17.
[18] H.H. Nguyen, D.T. Nguyen, Machine Learning Based Phishing Web Sites Detection, in: Proceedings of the International Conference on Advanced Engineering Theory and Applications (AETA), Ton Duc, Ho Chi Minh City, Vietnam, 2016, pp. 123–131.
[19] B. Pes, N. Dessì, M. Angioni, Exploiting the ensemble paradigm for stable feature selection: a case study on high-dimensional genomic data, Inform. Fusion 35 (2017) 132–147.
[20] S. Purkait, Examining the effectiveness of phishing filters against DNS based phishing attacks, Inform. Comp. Secur. 23 (3) (2015) 333–346.
[21] I. Qabajeh, F. Thabtah, An experimental study for assessing email classification attributes using feature selection methods, in: Proceedings of the 3rd International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Amman, Jordan, 2014, pp. 125–132.
[22] K.D. Rajab, New hybrid features selection method: A case study on websites phishing, Secur. Comm. Netw. 2017 (2017).
[23] G. Ramesh, I. Krishnamurthi, A comprehensive and efficacious architecture for detecting phishing webpages, Comp. Secur. 40 (2014) 23–37.
[24] O.K. Sahingoz, E. Buber, O. Demir, B. Diri, Machine learning based phishing detection from URLs, Expert Syst. Appl. 117 (2019) 345–357.
[25] A. Subasi, E. Molah, F. Almkallawi, T.J. Chaudhery, Intelligent phishing website detection using random forest classifier, in: Proceedings of the International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2017, pp. 1–5.
[26] C.L. Tan, Phishing Dataset for Machine Learning: Feature Evaluation, Mendeley Data, v1, 2018, Retrieved from https://doi.org/10.17632/h3cgnj8hft.1, accessed 20.05.18.
[27] F. Thabtah, N. Abdelhamid, Deriving correlated sets of website features for phishing detection: A Computational intelligence approach, J. Inform. Knowl. Manag. 15 (04) (2016) 1650042.
[28] F. Toolan, J. Carthy, Feature selection for Spam and Phishing detection, in: Proceedings of the eCrime Researchers Summit (eCrime), Dallas, Texas, USA, 2010, pp. 1–12.
[29] G. Varshney, M. Misra, P.K. Atrey, A survey and classification of web phishing detection schemes, Secur. Comm. Netw. 9 (18) (2016) 6266–6284.
[30] G. Xiang, J. Hong, C.P. Rose, L. Cranor, CANTINA+: A Feature-Rich machine learning framework for detecting phishing web sites, ACM Trans. Inf. Syst. Secur. 14 (2) (2011) 21.
[31] H. Zuhair, A. Selamat, M. Salleh, The effect of feature selection on phish website detection, Int. J. Adv. Comp. Sci. Appl. 6 (10) (2015) 221–232.