Iterative Construction of Hierarchical Classifiers for Phishing Website Detection

Jemal Abawajy, Gleb Beliakov, Andrei Kelarev, Morshed Chowdhury
School of Information Technology, Deakin University, 221 Burwood Hwy, Burwood 3125, Australia
Email: {jemal.abawajy,gleb.andrei.k,morshed.chowdhury}@deakin.edu.au

Abstract—This article is devoted to a new iterative construction of hierarchical classifiers in SimpleCLI for the detection of phishing websites. Our new construction of hierarchical systems creates ensembles of ensembles in SimpleCLI by iteratively linking a top-level ensemble to another middle-level ensemble instead of a base classifier so that the top-level ensemble can generate a large multi-level system. This new construction makes it easy to set up and run such large systems in SimpleCLI. The present article concentrates on the investigation of performance of the iterative construction of such classifiers for the example of detection of phishing websites. We carried out systematic experiments evaluating several essential ensemble techniques as well as more recent approaches and studying their performance as parts of the iterative construction of hierarchical classifiers. The results presented here demonstrate that the iterative construction of hierarchical classifiers performed better than the base classifiers and standard ensembles. This example of application to the classification of phishing websites shows that the new iterative construction combining diverse ensemble techniques into the iterative construction of hierarchical classifiers can be applied to increase the performance in situations where data can be processed on a large computer.

Keywords—phishing websites, ensemble classifiers, hierarchical multi-level classifiers, Random Forest

I. INTRODUCTION

Experiments evaluating classifiers applied to particular areas are important, since their outcomes can be used in order to improve the performance of future applications and can contribute to choosing directions of future research. For any given algorithm that produces very good outcomes in certain applications, there always exist examples of data sets in other domains where different algorithms are more effective. This is also confirmed by the so-called “no-free-lunch” theorems, which imply that a single algorithm can never be best for all types of problems [1]. The performance of every category of algorithms depends on the dimension of a data set and the number of instances, types of attributes, the nature of functional relations and dependencies among the attributes and other parameters.

We introduce a new iterative construction of hierarchical classifiers combining diverse ensembles into one unified system in SimpleCLI. This construction iteratively connects a top-level ensemble to another middle-level ensemble, so that the top-level ensemble can generate a large multi-level system in SimpleCLI. Our new iterative construction is explained in Section III. Here let us only mention that this construction is illustrated in Figure 1 in the case of the option that has achieved the best F-measure in experiments presented in this paper. Figure 2 shows how to generate the whole construction in SimpleCLI, see Section III for more details.

Every ensemble classifier at the middle level of this construction is an integral part of the ensemble classifier at the top level, and in turn every base classifier at the bottom level is included as a part of the ensemble classifier of the middle level, see Section III for more details. Using one ensemble as an integral part of another ensemble makes it easy to set up and run such hierarchical classifiers in SimpleCLI, even though their levels can be very large.

Fig. 1. The iterative construction of hierarchical classifier for phishing website detection in the case of the option that achieved the best F-measure in experiments presented in this article.

The present article is devoted to experiments comparing the performance of the new iterative construction of hierarchical classifiers, their base classifiers and standard ensemble classifiers in the special case of application to the detection of phishing websites. While phishing is an important direction that has been actively investigated recently, the aim of our paper is to develop a general technique that may be useful for various applications in information security. Let us refer to the Anti-Phishing Working Group [2] and recent papers [3]–[9] for preliminaries on phishing. The authors hope that the outcomes of this example of application prove helpful for the future development of classifiers in other branches of information security too.

Our new results show that the novel iterative construction of hierarchical classifiers achieved substantially better
performance in comparison with the base classifiers or standard ensemble classifiers. This demonstrates that the new method of combining diverse ensemble techniques in SimpleCLI into one unified iterative hierarchical classifier incorporating diverse ensembles as parts of other ensembles can be applied to improve classifications.

The paper is organized as follows. Previous work is presented in the next section. Section III describes new iterative construction of hierarchical classifiers investigated in this paper. Section IV is devoted to preprocessing of data. Sections V and VI deal with the base classifiers and ensemble classifiers. Section VII contains the outcomes of experiments comparing the effectiveness of base classifiers, ensemble classifiers and the iterative construction of hierarchical classifiers. These results are discussed in Section VIII. Main conclusions are presented in Section IX.

II. OVERVIEW OF PREVIOUS WORK

A. Recent Work on Multi-level Classifiers

Several different techniques for the design of hierarchical multi-tier classifiers are well known in artificial intelligence and data mining. Since a complete bibliography on this very general area would be more appropriate in a book, let us only include a few recent examples of papers devoted to multi-level systems and then discuss two most relevant articles.

Efficient multi-tier classifiers and multi-classifier systems have been explored recently, for example, in [7], [10]–[13]. All previous systems were different from our new construction. In particular, they used a smaller number of classifiers each time and combined the classifiers manually often via selecting the features poorly classified by one tier to be processed at the next tier.

A MATLAB program was written and presented in [14] to create large hierarchical ensembles of multi-level classifiers and apply them for the diagnosis of Alzheimer’s disease. First, the program partitions the whole brain image into a number of local 3D patches. Then it trains two low-level classifiers for each patch. After that it builds a new set of high-level classifiers corresponding to different regions of the brain. Using a forward greedy search strategy it chooses a subset of the high-level classifiers with larger discriminating capacity. Finally, it combines the output of the selected high-level classifiers using a weighted voting. As we see, this hierarchical system uses different data mining ideas to create a hierarchical structure. It has produced promising results, but it cannot be applied to other areas, since it relies on the local spatial contiguity of the brain regions.

Another relevant approach is the multi-tier classification model for phishing email filtering proposed in [15]. It incorporates an innovative method for extracting the features of phishing emails based on weighting of message content and message header and selection of the features according to a priority ranking. The impact of rescheduling the classifier algorithms in a multi-tier classification process is examined to find out the optimum scheduling. A detailed empirical performance and analysis of the proposed algorithm demonstrated that the proposed algorithm reduces the false positive problems substantially with lower complexity. The multi-tier construction considered in [15] included only four classifiers simultaneously and used data mining ideas different from those employed in the present article.

B. Recent Work on the Detection of Phishing

This section contains a few recent examples of work devoted to the detection of phishing. The readers are referred to [15] for a more comprehensive overview of earlier literature on the prevention of phishing attacks. In particular, it is explained in [15] that the growing scale and sophistication of phishing attacks motivates further research and the development of new advanced approaches.

The papers [5] and [16] carried out a systematic investigation of a hybrid feature selection approach based on a combination of content-based and behaviour-based features. The main objective of the paper is to identify behaviour-based features that cannot be disguised by an attacker. The proposed approach mines the attacker behaviour based on the email headers.

The paper [17] is devoted to unsupervised authorship analysis of phishing webpages that aims to enable advanced analysis of phishing attacks beyond basic exploratory studies. The authors use salient features from webpages to derive properties concerning the author and group webpages so that all webpages produced by one author are grouped together.

The paper [18] analyses the results of a phishing exercise conducted to train users cultivating users resistance against phishing attacks. A different game design framework for training users against phishing was investigated in [19].

A qualitative study conducted in [20] investigated the human factors and psychological mechanisms associated with phishing attacks. Investigated phishing victimization using a dual-process theory of information processing called the Heuristic-Systematic Model. They included an introduction to the Heuristic-Systematic Model and an explanation of how and why it can be applied to study victimization by phishing. A research model was then developed for an explorative study.

The article [21] investigated both phishing emails and webpages. The best discovery rate achieved for phishing websites was 81.6%. The proposed methodology not only detected phishing attacks, but also discovered the entity or organization that the attackers impersonate during phishing attacks. The methodology employed natural language processing and machine learning. It combined the use of Conditional Random Field, CRF, and Latent Dirichlet Allocation, LDA, operating on both phishing and non-phishing data. Utilizing the discovered topics and named entities as features, the next stage classified each message as phishing or non-phishing. For messages...
classified as phishing, the final stage discovered the impersonated entity using CRF. At this stage, not all emails were included in the data for experiments. The approach discovered the impersonated entity from messages that are classified as phishing, with a discovery rate of 88.1%. The automatic discovery of impersonated entity from phishing aims to help the legitimate organization to take down the offending phishing site. This protects their users from falling for phishing attacks, which in turn leads to satisfied customers. Automatic discovery of an impersonated entity also aims to help email service providers to collaborate with each other to exchange attack information and protect their customers.

A robust technique counteracting phishing was proposed in [22]. It is based on new inputs not considered previously in a single protection platform. These include Legitimate site rules, User-behaviour profile, PhishTank, User-specific sites, and Pop-Ups from emails. A neuro-fuzzy scheme with five inputs is utilized to detect phishing sites. In this study, two-Fold cross-validation was applied for training and testing of the proposed method. A total of 288 features with 5 inputs were used.

The paper [23] proposes a new phishing webpage detection approach based on a called transductive support vector machine, T SVM, which is a semi-supervised learning method. Firstly, the features of web image are extracted including gray histogram, colour histogram, and spatial relationship between subgraphs. These features are intended to complement other features and eliminate the disadvantages of phishing detection based only on document object model, DOM. Then the features of sensitive information are examined by using page analysis based on DOM objects. In contrast to the weaknesses of support vector machine, SVM, trained on little and poorly labelled samples, this method introduces the T SVM to train a classifier that takes into account the distribution information implicitly embodied in the large quantity of the unlabeled samples, and has better performance than SVM.

A behaviour-based model for trustworthiness testing was proposed in [24]. Instead of testing a website against a set of known inputs and comparing the expected outputs with the actual ones, the trustworthiness testing determines whether the response behaviour of a website matches our knowledge of phishing or legitimate website behaviours to decide whether a website is phishing or legitimate. A suspected website was considered as a web-based program and was tested based on a behaviour model. The model was described using the notion of a Finite State Machine, FSM, that captured the submission of forms with random inputs and the corresponding responses.

III. NEW ITERATIVE CONSTRUCTION OF HIERARCHICAL CLASSIFIERS

This section is devoted to the theoretical model of the new iterative construction of hierarchical classifiers in SimpleCLI.

Standard ensemble classifiers combine a collection of base classifiers into a common classification system. Here we introduce and explain our new iterative construction of hierarchical classifiers.

This paper introduces a new iterative hierarchical construction in SimpleCLI, which makes it easy to combine diverse ensemble methods into one scheme. Our experiments are devoted to performance evaluation of the new iterative hierarchical construction of classifiers for phishing websites.

This paper deals with a new construction of hierarchical classifiers, illustrated in Figure 1 for the particular case of the very best option evaluated in experiments presented in this paper. This diagram corresponds to the generation of the hierarchical classifier in SimpleCLI, and shows which ensemble classifier generates each of the base classifiers. All base classifiers pass their output on to Level 2 ensemble classifiers. The Level 2 ensemble classifiers combine the output of base classifiers. Their output in turn is analysed by the Level 3 ensemble classifier that makes the final decision for the whole hierarchical multi-level classification system. Arcs not connected to classifiers indicate possible additional classifiers. The whole system may involve thousands of base classifiers, but it is easy to set it up, since in most cases the Level 2 classifiers generate the whole collection of their base classifiers automatically given just one instance of a base classifier. Likewise, all Level 2 ensemble classifiers are generated by the Level 3 ensemble classifier automatically given only one instance of a Level 2 ensemble classifier. This means that the Level 3 ensemble classifier generates its Level 2 classifiers and executes them in exactly the same way as it usually handles base classifiers. Similarly, each Level 2 ensemble applies its method to combine its base classifiers as usual. The whole system is generated automatically in SimpleCLI, as illustrated in Figure 2.

---

© 2014 ACADEMY PUBLISHER
phishing website detection. These new results show, in particular, that Random Forest performed best in this setting for our data set considered in this article, and that novel iterative construction of hierarchical classifiers in SimpleCLI can be used to achieve further improvement of the classification outcomes. The hierarchical classifiers with large levels based on Random Forest achieved better performance compared with the base classifiers or simpler ensemble classifiers.

The unified iterative construction of hierarchical classifiers in SimpleCLI can produce multi-level classifiers with large levels that require a lot of computer memory to train, especially for very large data sets, where they can be used to improve performance. If a data set is small and a hierarchical classifier is larger, then it will revert to using just one base classifier and produce the same outcomes as the base classifier. As we will see in Section VII below, our experiments show that the iterative construction of hierarchical classifiers is effective if diverse ensembles are combined at different levels. The authors believe that this approach to designing ensembles of classifiers deserves further investigation for other large data sets and application directions too.

IV. FEATURE EXTRACTION AND REDUCTION

We use the same data set and the same set of features of phishing websites considered by the authors in [3], since it is suitable for this study. Our new experiments used a collection of simple features extracted during work on the paper [3]. Similar data sets are available from the downloadable databases at the PhishTank [25]. The present article investigates a novel method for improving performance of the classifiers, and we did not attempt to extract more sophisticated collections of features. The extraction of features is very important for applications, for example, see [26]–[36], but it is not the main focus of the present article.

Since this paper concentrates on the contribution of hierarchical multi-level classifiers with large levels, for the purposes of this work, we applied the bag-of-words model and extracted only a simple collection of the features reflecting the content of the websites. As in [3], we used term frequency–inverse document frequency word weights, or TF-IDF weights, to select words as features. Features were extracted using a flexible preprocessing and feature extraction system implemented in Python by the third author.

These weights are well known in feature extraction for text categorization [37], see also [38]. They are defined using the following concepts and notation. Suppose that we are extracting features from a data set \( E \), which consists of \( |E| \) websites. For a word \( w \) and a website \( m \), let \( N(w, m) \) be the number of times \( w \) occurs in \( m \). Suppose that a collection \( T = \{ t_1, \ldots, t_k \} \) of terms \( t_1, \ldots, t_k \) is being looked at. The term frequency of a word \( w \in T \) in a website \( m \) is denoted by \( \text{TF}(w, m) \) and is defined as the number of times \( w \) occurs in \( m \), normalized over the number of occurrences of all terms in \( m \):

\[
\text{TF}(w, m) = \frac{N(w, m)}{\sum_{i=1}^{k} N(t_i, m)}
\]

The document frequency of the word \( w \) is denoted by \( \text{DF}(w) \) and is defined as the number of websites in the given data set where the word \( w \) occurs at least once. The inverse document frequency is used to measure the significance of each term. It is denoted by \( \text{IDF}(w) \) and is defined by the following formula

\[
\text{IDF}(w) = \log \left( \frac{|E|}{\text{DF}(w)} \right).
\]

The term frequency–inverse document frequency of a word \( w \) in website \( m \), or TF-IDF weight of \( w \) in \( m \) is defined by

\[
\text{TF-IDF}(w, m) = \text{TF}(w, m) \times \text{IDF}(w, m).
\]

We collected a set of words with highest TF-IDF scores in all websites of the data set. For each website, the TF-IDF scores of these words in the website were determined. These weights and additional features were assembled in a vector. In order to determine the TF-IDF scores we used Gensim, a Python and NumPy package for vector space modelling of text documents. These features were collected in a vector space model representing the data set. Further reduction of the set of features was accomplished using the Information Gain and Goodman–Kruskal Correlation Coefficient. By the Information Gain, \( IG \), here we mean the expected value of the information gain, that is the mutual information \( I(X, Y) \) of \( X \) and \( Y \). It is equal to the reduction in the entropy of \( X \) achieved by clarifying the value of the variable \( Y \), see [38]. Initially, we explored and compared the following four correlation coefficients.

The Pearson Linear Correlation Coefficient, PLCC, is also called the Pearson’s Product-Moment Correlation Coefficient, [38]. It is often helpful in various situations and has low complexity. To calculate the PLCC the features and class labels have to be assigned numerical values. The PLCC is calculated to assess the correlation between the values of the feature and the numerical class labels of instances. We refer the readers to [38] and [39] for complete explanations and include a brief formula here. Let \( f \) be a particular feature of the data set. The mean value of \( f \) in all instances is denoted by \( f' \), so that

\[
f' = \frac{\sum_{i=1}^{n} (d_i)}{n},
\]

where \( d_i \) stands for the value of the feature \( f \) in the corresponding instance. The standard deviation \( \sigma(f) \) can be calculated as

\[
\sigma(f) = \sqrt{\frac{\sum_{i=1}^{n} (f' - d_i)^2}{n}}.
\]

The numerical class label of the instance \( i \) (with the value of feature \( f \) equal to \( d_i \)) is denoted by \( I_i \). The mean of numerical labels of class values of the instances is denoted

© 2014 ACADEMY PUBLISHER
by \( I' \). The standard deviation of the labels of instances is denoted by \( \sigma(I) \). The covariance \( \text{cov}(f, I) \) between \( f \) and \( I \) is defined by

\[
\text{cov}(f, I) = \frac{1}{n} \sum_{i=1}^{n} (f_i - \bar{f})(I_i - \bar{I}).
\]

Then the PLCC is denoted by \( \rho(f, I) \) and is defined by the following formula:

\[
\rho(f, I) = \frac{\text{cov}(f, I)}{\sigma(f)\sigma(I)}. \tag{6}
\]

see [38] and [39] for more explanations.

Second, we used the Spearman Rank Correlation Coefficient, SRCC, also known as Spearman’s Rho [38], [40], [41]. It assesses how well the relationship can be described using a monotonic function, which does not have to be linear. The SRCC \( \rho \) is a measure of association based on the ranks of the data values. It is given by the formula

\[
\rho = \frac{\sum(R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum(R_i - \bar{R})^2}(\sum(S_i - \bar{S})^2)}. \tag{7}
\]

where \( R_i \) is the rank of the \( i \)-th \( x \)-value, \( S_i \) is the rank of the \( i \)-th \( y \)-value, \( \bar{R} \) is the mean of the ranks of \( x \)-values, and \( \bar{S} \) is the mean of the ranks of \( y \)-values. The values of \( \rho \) belong to the segment \([-1; 1]\). Values close to 1 indicate that there is a good correlation (described by a monotonically increasing function). Having found the SRCC for each feature, we ordered the original features by the values of their Spearman Rank Correlation Coefficients. The features with higher values were selected for the next stage of our procedure.

The Kendall Rank Correlation Coefficient, KRCC, is also called the Kendall’s Tau, [38], [40], [41]. Our experiments have shown again that it produces outcomes very similar to the SRCC.

The Goodman–Kruskal Correlation Coefficient, GKCC, is also called the Goodman–Kruskal’s Gamma, [38], [40], [41]. It is defined as the difference between the number of concordant pairs \( C \) and the number of discordant pairs \( D \) of the two rankings, as a proportion of all pairs, ignoring ties:

\[
G = (C - D)/(C + D). \tag{9}
\]

GKCC tests for a weak monotonicity between the two rankings. The value of GKCC ranges between +1 to -1, and it is equal to 0 for independent variables.

We ranked all the preliminary variables according to the values of their rank correlation coefficients. The higher the ranking of the feature, the more relevant it is to the classification result. The least important features can then be removed.

All of these correlation coefficients can be used to delete less relevant features with almost zero correlation to the classes of phishing websites and legitimate websites, respectively. Our experiments have shown that the Goodman–Kruskal Correlation Coefficient produced better results among all correlation coefficients. This is why the final tables of this paper include only the outcomes obtained using the Goodman–Kruskal Correlation Coefficient, where they are compared with the results obtained using Information Gain.

V. Base Classifiers

The following classifiers available in WEKA [42] were used as base classifiers in our experiments with outcomes presented in Section VII: FURIA [43], J48 [44], LibLINEAR [45], LibSVM [46]–[48], Random Forest [49], SMO [50]–[52]. These robust classifiers were chosen since they represent most essential types of classifiers available in WEKA [42] and performed well for our data set.

FURIA is a fuzzy unordered rule induction algorithm due to [43]. It extends the RIPPER algorithm by learning fuzzy rules instead of conventional rules and learning unordered rule sets instead of rule lists. At the same time it uses the same simple and comprehensive rule sets, and applies a novel rule stretching method. Experimental results show that FURIA outperforms RIPPER and J48, see [43].

J48 creates a C4.5 decision tree by adding attributes to the tree as explained in [44]. At every step the feature with the highest information gain is added. This means that every next attribute is chosen so that it is best in discriminating the instances in the training set. The classifier can generate pruned or unpruned C4.5 trees.

LibLINEAR is an open source library for large scale linear classification [45]. Experiments demonstrate that it is fast and is very efficient on large sparse data sets.

LibSVM is a library for Support Vector Machines originally implemented as described in [47], see also [48]. New official implementation document is [46]. At the time of our study LibSVM included in WEKA could handle only binary classes.

Random Forest constructs a forest of random trees as explained in [49]. To control the variation in generating the set of random trees, Random Forest uses the process of random selection of features proposed in [53], [54] and independently in [55].

SMO is a fast implementation of Support Vector Machines using Sequential Minimal Optimization. It generates a collection of hyperplanes in the \( n \)-dimensional space that separate classes of the data best and have large margins, i.e., distances to the nearest data points in the space, as explained in [50]–[52].

VI. Ensemble Classifiers

We used SimpleCLI command line in WEKA [42] to investigate the performance of the following ensemble techniques: AdaBoost [56], Bagging [57], Daggging [58], Decorate [59], Grading [60], MultiBoost [61] and Stacking [62]. Consensus functions can also be used as a replacement for voting to combine the outputs of several classifiers, as it was done in [4], [9], [63]. These functions did not result in substantial improvement and have not
been included in the tables with final outcomes in the present paper though.

AdaBoost uses several classifiers in succession. Each classifier is trained on the instances that have turned out more difficult for the preceding classifier. To this end all instances are assigned weights, and if an instance turns out difficult to classify, then its weight increases. We used the highly successful AdaBoost classifier described in [56].

Bagging (bootstrap aggregating), generates a collection of new sets by resampling the given training set at random and with replacement. These sets are called bootstrap samples. New classifiers are then trained, one for each of these new training sets. They are amalgamated via a majority vote, [57], see also [64] and [65].

Dagging is useful in situations where the base classifiers are slow. It divides the training set into a collection of disjoint (and therefore smaller) stratified samples, trains copies of the same base classifier and averages their outputs using vote, [58].

Decorate is an acronym for Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples. Decorate constructs special artificial training examples to created varied hypotheses for building diverse ensembles of classifiers. As all other general meta-learners, Decorate can use any base classifier as a template. It builds a diverse ensemble of classifiers based on the template and trained on varied collections of samples including artificial training examples. A comprehensive set of tests have established that Decorate consistently generates ensembles more accurate than the base classifier, Bagging, Random Forests, which are also more accurate than Boosting on small training sets, and are comparable to Boosting on larger training sets, [59].

Grading trains base classifiers and grades their output as correct or wrong; these graded outcomes are then combined, [60].

MultiBoost extends the approach of AdaBoost with the wagging technique, [61]. Wagging is a variant of bagging where the weights of training instances generated during boosting are utilized in selection of the bootstrap samples, [66]. It is explained in [61] that experiments on a large and diverse collection of UCI data sets have demonstrated that MultiBoost achieves higher accuracy significantly more often than wagging or AdaBoost.

Stacking can be regarded as a generalization of voting, where meta-learner aggregates the outputs of several base classifiers, [62].

VII. EXPERIMENTS EVALUATING PERFORMANCE

Experimental setup and the outline of all options tested in our empirical study are depicted in Figure 3, which shows that we compared base classifiers, ensemble classifiers and hierarchical multi-level classifiers with large level for the sets of features selected using TF-IDF scores, the Goodman-Kruskal correlation coefficient and Information Gain.

We used 10-fold cross validation to evaluate the effectiveness of classifiers in all experiments and avoid overfitting. The following measures of performance of classifiers are often used in this research direction: precision, recall, F-measure, accuracy, sensitivity, specificity and Area Under Curve also known as the Receiver Operating Characteristic or ROC area.

Notice that weighted average values of the performance metrics are usually used. This means that they are calculated for each class separately, and a weighted average is found then. In particular, our results included in this paper deal with the weighted average values of precision. In contrast, the accuracy is defined for the whole classifier as the percentage of all websites classified correctly, which means that this definition does not involve weighted averages in the calculation. Precision of a classifier, for a given class, is the ratio of true positives to combined true and false positives.

Sensitivity is the proportion of positives (phishing websites) that are identified correctly. Specificity is the proportion of negatives (legitimate websites) which are identified correctly. Sensitivity and specificity are measures evaluating binary classifications. For multi-class classifications they can be also used with respect to one class and its complement. Sensitivity is also called True Positive Rate. False Positive Rate is equal to 1 - specificity. These measures are related to recall and precision. Recall is the ratio of true positives to the number of all positive samples (i.e., to the combined true positives and false negatives). The recall calculated for the class of phishing websites is equal to sensitivity of the whole classifier.

All tables of outcomes in this paper include the F-measure, since it combines precision and recall into a single number evaluating performance of the whole system, [67]. The F-measure is equal to the harmonic mean of precision and recall

$$F\text{-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

(10)

The weighted average F-measure is contained in the standard WEKA output for all classifiers.

It is enough to indicate only the F-measure, because in all our experiments each precision turned out approximately equal to the corresponding recall, and so they are
both quite close to their respective F-measure, as it is usually the case for well balanced clusterings.

First, we include the results of experiments comparing the performance of several base classifiers for phishing websites. The performance of the SMO, LibSVM and LibLINEAR depends on the SVM type, the kernel and several numerical parameters. We have considered all types of SVMs and kernels in SMO, LibSVM and LibLINEAR that could handle the format of our data without additional preprocessing. The F-measure of outcomes obtained using all of these kernels are presented in Table I. For each of these cases, one can use the optimization procedure explained in [48]. More advanced optimization techniques presented in [68] can also be applied here. The results obtained for base classifiers are included in Table II and illustrated in Figure 4. Only the best kernels of SMO, LibSVM and LibLINEAR found in Table I have been included in Table II. For ease of comparison, summary diagrams of the F-measure of all ensemble algorithms are given in Figure 5. In these tests all ensembles were used with one and the same base classifier, RandomForest, in all tests. (We also tested several other ensembles with different base classifiers, but they turned out worse, so that their outcomes are not included.)

Finally, we include the results of experiments evaluating the iterative construction of hierarchical classifiers. This is the main focus of the paper. These experiments included all combinations of Bagging, Decorate and MultiBoost, since these ensemble methods produced better F-measure in Table III. Summary diagrams of the F-measure of all hierarchical three-level algorithms are presented in Figure 6. In our experiments, each system contained 21 ensembles and 400 base classifiers. We have not included repetitions of the same ensemble technique in both levels, since tests have shown that they do not produce further improvement. The outcomes of the hierarchical three-level classifiers with large levels are based on RandomForest. Preliminary tests demonstrated that ensemble classifiers based on RandomForest were also more effective than the ensembles based on other classifiers.

F-measure of the resulting ensemble classifiers are presented in Table III, which shows improvement as compared to the base classifiers. For ease of comparison, summary diagrams of the F-measure of all ensemble algorithms are given in Figure 5. In these tests all ensembles were used with one and the same base classifier, RandomForest, in all tests. (We also tested several other ensembles with different base classifiers, but they turned out worse, so that their outcomes are not included.)

Finally, we include the results of experiments evaluating the iterative construction of hierarchical classifiers. This is the main focus of the paper. These experiments included all combinations of Bagging, Decorate and MultiBoost, since these ensemble methods produced better F-measure in Table III. Summary diagrams of the F-measure of all hierarchical three-level algorithms are presented in Figure 6. In our experiments, each system contained 21 ensembles and 400 base classifiers. We have not included repetitions of the same ensemble technique in both levels, since tests have shown that they do not produce further improvement. The outcomes of the hierarchical three-level classifiers with large levels are based on RandomForest. Preliminary tests demonstrated that ensemble classifiers based on RandomForest were also more effective than the ensembles based on other classifiers.

F-measure of the resulting ensemble classifiers are presented in Table III, which shows improvement as compared to the base classifiers. For ease of comparison, summary diagrams of the F-measure of all ensemble algorithms are given in Figure 5. In these tests all ensembles were used with one and the same base classifier, RandomForest, in all tests. (We also tested several other ensembles with different base classifiers, but they turned out worse, so that their outcomes are not included.)

Finally, we include the results of experiments evaluating the iterative construction of hierarchical classifiers. This is the main focus of the paper. These experiments included all combinations of Bagging, Decorate and MultiBoost, since these ensemble methods produced better F-measure in Table III. Summary diagrams of the F-measure of all hierarchical three-level algorithms are presented in Figure 6. In our experiments, each system contained 21 ensembles and 400 base classifiers. We have not included repetitions of the same ensemble technique in both levels, since tests have shown that they do not produce further improvement. The outcomes of the hierarchical three-level classifiers with large levels are based on RandomForest. Preliminary tests demonstrated that ensemble classifiers based on RandomForest were also more effective than the ensembles based on other classifiers.
Table III

<table>
<thead>
<tr>
<th>Ensemble Classifiers</th>
<th>F-measure</th>
<th>GKCC</th>
<th>IG</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost</td>
<td>0.936</td>
<td>0.950</td>
<td></td>
</tr>
<tr>
<td>Bagging</td>
<td>0.945</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>Dagging</td>
<td>0.914</td>
<td>0.921</td>
<td></td>
</tr>
<tr>
<td>Decorate</td>
<td>0.947</td>
<td>0.962</td>
<td></td>
</tr>
<tr>
<td>Grading</td>
<td>0.923</td>
<td>0.933</td>
<td></td>
</tr>
<tr>
<td>MultiBoost</td>
<td>0.942</td>
<td>0.952</td>
<td></td>
</tr>
<tr>
<td>Stacking</td>
<td>0.932</td>
<td>0.945</td>
<td></td>
</tr>
</tbody>
</table>

Table IV

<table>
<thead>
<tr>
<th>Hierarchical Classifiers</th>
<th>F-measure</th>
<th>GKCC</th>
<th>IG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 3</td>
<td>Level 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bagging</td>
<td>Decorate</td>
<td>0.973</td>
<td>0.977</td>
</tr>
<tr>
<td>Bagging</td>
<td>MultiBoost</td>
<td>0.969</td>
<td>0.975</td>
</tr>
<tr>
<td>Decorate</td>
<td>Bagging</td>
<td>0.972</td>
<td>0.977</td>
</tr>
<tr>
<td>Decorate</td>
<td>MultiBoost</td>
<td>0.970</td>
<td>0.975</td>
</tr>
<tr>
<td>MultiBoost</td>
<td>Bagging</td>
<td>0.968</td>
<td>0.972</td>
</tr>
<tr>
<td>MultiBoost</td>
<td>Decorate</td>
<td>0.971</td>
<td>0.975</td>
</tr>
</tbody>
</table>

Figure 5. F-measure of ensemble classifiers for phishing website detection using the Goodman–Kruskal Correlation Coefficient and Information Gain.

Figure 6. F-measure of the iterative constructions of hierarchical classifiers in SimpleCLI for phishing website detection using the Goodman–Kruskal Correlation Coefficient and Information Gain.

VIII. DISCUSSION

Our work shows that the iterative construction of hierarchical classifiers is quite easy to use and can be applied to improve classifications, if diverse ensembles are combined at different levels of the construction. It is an interesting question for future work to investigate the effectiveness of hierarchical classifiers generated in SimpleCLI for various other large data sets.

Random Forest outperformed other base classifiers for the phishing websites data set, and Decorate improved its outcomes better than other meta classifiers did. The best outcomes were obtained by the new iterative construction of hierarchical classifiers where Bagging is used in Level 3 and Decorate in Level 2.

The performance of classifiers considered in this paper depends on several numerical input parameters. In order to have a uniform comparison of outcomes across all types of constructions of classifiers, we used them with the same default values of these parameters in all experiments. It would be also interesting to investigate how the outcomes change when the input parameters are optimized using the optimization techniques presented in [68]. Optimization algorithms involved are computationally intensive, and so this question can be best addressed separately in subsequent publications.

IX. CONCLUSION

We carried out a systematic investigation of a new iterative construction of hierarchical multi-level classifiers with large levels, where diverse ensembles are combined into a unified system by integrating different ensembles at a lower level as a part of another ensemble at the top level. Our experiments evaluated the performance of this iterative construction in SimpleCLI for a data set of phishing websites and have demonstrated the feasibility and performance of the approach. The experimental outcomes show that the iterative construction can be used to improve classifications. Such classifiers produced better results compared to the base classifiers or standard ensemble classifiers.

ACKNOWLEDGMENT

The authors are grateful to three reviewers for comments that have helped to improve the text of this
article, and for suggesting several interesting questions for future research. The first author was supported by Discovery grant DP0880501 from Australian Research Council. The third author was supported by ARC Discovery grant DP0449469. All authors were supported by several Deakin-Ballarat collaboration grants.

REFERENCES


Jemal Abawajy is a Professor and the Director of the Parallel and Distributing Computing Lab at Deakin University, Australia. He is a Senior Member of IEEE and was a member of the organizing committees for over 100 international conferences serving in various capacities including chair, general co-chair, vice-chair, best paper award chair, education chair, session chair and program committee member. He has published more than 200 refereed articles, supervised numerous PhD students to completion and is on the editorial boards of many journals.

Gleb Beliaev is a Professor at the School of Information Technology at Deakin University, where he has been working since 1999 after several postdoc positions in Australia and overseas. He is a Senior Member of IEEE and is an Associate Editor of two flagship journals in the area of fuzzy systems. His research focuses on computational mathematics, numerical data analysis, fuzzy systems and aggregation operators. He is an author of a monograph on aggregation functions, two edited volumes and over 150 refereed papers.

Andrei Kelarev is an author of two books and 194 journal articles, and an editor of 5 international journals. He worked as an Associate Professor in the University of Wisconsin and University of Nebraska in USA and as a Senior Lecturer in the University of Tasmania in Australia. He was a Chief Investigator of a large Discovery grant from Australian Research Council and was a member of the program committees of several conferences.

Morshed Chowdhury is a faculty of School of Information Technology, and the founder member of the Parallel and Distributing Computing Lab at Deakin University, Australia. He has organised IEEE sponsored ACIS2007 and SNPD2012 conferences as a Co-Chair. Dr. Chowdhury was a member of the organizing committees for over 50 international conferences serving in various capacities. He was awarded a best research paper in Annual International conference on “Information Technology Security (ITS2010)” in 2010. He has published more than 70 refereed articles and is a member of the editorial board of International Journal of Software Innovation (IJSI), IGI Global, US. He has been supervising seven PhD students.