Bio-inspired Machine Learning Mechanism for Detecting Malicious URL Through Passive DNS in Big Data Platform

Saad M. Darwish, Ali E. Anber, and Saleh Mesbah

Abstract Malicious links are used as a source by the distribution channels to broadcast malware all over the Web. These links become instrumental in giving partial or full system control to the attackers. To overcome these issues, researchers have applied machine learning techniques for malicious URL detection. However, these techniques fall to identify distinguishable generic features that are able to define the maliciousness of a given domain. Generally, well-crafted URL's features contribute considerably to the success of machine learning approaches, and on the contrary, poor features may ruin even good detection algorithms. In addition, the complex relationships between features are not easy to spot. The work presented in this paper explores how to detect malicious Web sites from passive DNS based features. This problem lends itself naturally to modern algorithms for selecting discriminative features in the continuously evolving distribution of malicious URLs. So, the suggested model adapts a bio-inspired feature selection technique to choose an optimal feature set in order to reduce the cost and running time of a given system, as well as achieving an acceptably high recognition rate. Moreover, a two-step artificial bee colony (ABC) algorithm is utilized for efficient data clustering. The two approaches are incorporated within a unified framework that operates on the top of Hadoop infrastructure to deal with large samples of URLs. Both the experimental and statistical analyses show that improvements in the hybrid model have an advantage over some conventional algorithms for detecting malicious URL attacks. The results demonstrated that the suggested model capable to scale 10 million query answer pairs with more than 96.6% accuracy.

S. M. Darwish
Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, 163 Horreya Avenue, El-Shatby, P.O. Box 832, Alexandria 21526, Egypt

A. E. Anber (✉)
Faculty of Computers and Information, Damanhour University, Damanhour, Egypt
e-mail: ali.anber@damanhour.edu.eg

S. Mesbah
Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt

A. E. Hassanien et al. (eds.), *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*, Studies in Big Data 77,
https://doi.org/10.1007/978-3-030-59338-4_9

Keywords Machine learning · Malicious URLs · Passive DNS · Big data · Hadoop · Genetic algorithm · Artificial bee colony

1 Introduction

The web has become a platform for supporting a wide range of criminal enterprises such as spam-advertised commerce, financial fraud, and malware propagation. The security group has responded to this by creating blacklisting tools. The distribution channels use harmful URLs as a medium to spread malware throughout the Internet. These relationships help to give the attackers, who use systems for various cyber-crimes, partial or comprehensive control over the system. Systems with the ability to detect malicious content should be quick and precise to detect such crimes [1]. DNS data analysis has several advantages compared to other methods like the blacklist of compromised domains as the DNS traffic has a considerable number of useful features to classify domain names affiliated with fraudulent activities [2].

Detection of malicious domains through the analysis of DNS data has a number of benefits compared to other approaches such as blacklists [2, 3]. First, DNS data constitutes only a small fraction of the overall network traffic, which makes it suitable for analysis even in large-scale networks that cover large areas. Moreover, caching, being an integral part of the protocol, naturally facilitates further decrease in the amount of data to be analyzed, allowing researchers to analyze even the DNS traffic coming to top level domains. Second, the DNS traffic contains a significant amount of meaningful features to identify domain names associated to malicious activities. Third, many of these features can further be enriched with associated information, such as autonomous system number, domain owner, and so on, providing an even richer space exploitable for detection. The large amount of features and the vast quantity of traffic data available have made DNS traffic a prime candidate for experimentation with various machine-learning (ML) techniques applied to the context of security. Fourth, although the solutions to encrypt DNS data exist, still a large fraction of DNS traffic remains unencrypted, making it available for the inspection in various Internet vantage points. Fifth, sometimes researchers are able to reveal attacks at their early stages or even before they happen due to some traces left in the DNS data.

Detecting malicious URLs is an essential task in network security. Despite many exciting advances over last decade for malicious URL detection using machine learning techniques, there are still many open problems and challenges which are critical and imperative, including but not limited to the following [4]: (1) **High volume and high velocity**: The real-world URL data is obviously a form of big data with high volume and high velocity. It is almost impossible to train a malicious URL detection model on all worlds' URL data using machine learning. (2) **Difficulty in collecting features**: collecting features for representing a URL is crucial for applying machine learning techniques. In particular, some features could be costly (in terms of time) to collect, e.g., host-based features. (3) **Feature representation**: In addition to high volume and high velocity of URL data, another key challenge is the very high-dimensional features. Some commonly used learning techniques, such as feature selection, dimension reduction and sparse learning, have been explored, but they are far from solving the challenge effectively. Besides the high dimensionality issue, another more severe challenge is the evolving high dimensional feature space, where the feature space often grows over time when new URLs and new features are added into the training data. This again poses a great challenge for a clever design of new machine learning algorithms which can adapt to the dynamically changing feature spaces. (4) **Concept drifting and new emerging challenges**: Another challenge is the concept drifting where the distribution of malicious URLs may change over time due to the evolving behaviors of new threats and attacks. This requires machine learning techniques to be able to deal with concept drifting whenever it appears. Besides, another recent challenge is due to the popularity of URL shortening services, which take a long URL as input and produce a short URL as an output.

A variety of approaches have been attempted to tackle the problem of Malicious URL Detection. According to the fundamental principles, these approaches can be broadly grouped into three major categories: (i) Blacklisting, (ii) Heuristics, and (iii) Machine Learning approaches. The key principles of each category are briefly described in [4]. Despite many exciting advances over last decade for malicious URL detection using machine learning techniques, there are still many open problems and challenges which are critical and imperative, including but not limited to the following [3]: high volume and high velocity, difficulty in acquiring labels, difficulty in collecting features, feature representation, and concept drifting [1]. In recent times, improvements in technology and infrastructure have led to creating the problem of big data. Accordingly, it is necessary for cyber security professionals to design and implement novel methods to efficiently and effectively mitigate cyber security threats in a big data world. Hadoop, an open-source distributed storage platform that can run on commodity hardware, has been utilized to better accommodate the big data storage requirements of massive volume and fast-speed processing criteria of potentially very complex, heterogeneous data structures [5].

1.1 Problem Statement and Research Motivation

Compromised websites that are used for cyber-attacks are named as malicious URLs. In fact, nearly one third of all websites were identified as potentially malicious in nature, showing that malicious URLs are used widely in order to commit cyber-crimes [3]. To overcome the problem to maintain and update a blacklist, some systems inspect the web page content and analyze the behavior after visiting the Web page. Unfortunately, this increases the run time overhead and affects the user experience. With the development of machine learning, several classification-based methods, using the features of Web page content and URL text, are also used to detect malicious

URLs. However, the attackers adjust their strategies accordingly and invent new kinds of attacks [6]. This work is motivated by the observation that attackers tend to abuse certain domain features during this process. By basing a detection system on such features, we can effectively detect and blacklist the malicious web pages.

1.2 Contribution and Methodology

This paper addresses the problem of detecting malicious Web sites using machine learning over URL-based features. The challenge is to construct a system that is accurate, scalable, and adaptive. To this end, the contributions of this paper are the following: (1) Show that better classification is possible by extracting more meaningful URL data characteristics that outweigh the advantages that a skilled learner makes. For this reason, a bio-inspired reduction process is applied that adopts GA to refine the lists of features (optimal features), with the goal of building a robust and efficient learning model. (2) Adapt a two-step artificial bee colony (ABC) algorithm for efficient Malicious URL clustering.

The rest of this paper is organized as follows: Sect. 2 presents a review of the recent related works. Section 3 introduces the proposed malicious URL detection model. Section 4 exhibits the experimental results to evaluate the performance study of the proposed model. Finally, we conclude the paper and give future directions in Sect. 5.

2 Related Work

Because the suggested technique is based on passive domain name-based features of the URL, more emphasis in surveying related work that incorporates those features are placed. Ma et al. [6] presented an approach for classifying URLs automatically as either malicious or benign based on supervised learning across both lexical and host-based features. Another approach was suggested by Zhang et al. in 2007 [7] that presented the design and evaluation of Carnegie Mellon Anti-phishing and Network Analysis Tool (CANTINA), a novel content-based approach for detecting phishing web sites. CANTINA uses the well-known Term Frequency/Invest Document Frequency (TF-IDF) algorithm and applies it to anti-phishing, to robust hyperlinks, the idea of overcoming problems with page.

Kan and Thi [8] introduced the notion of bag-of-words representation for classifying URLs. Concurrently, in 2009, Guan et al. [9] had examined the aspect of instant messaging (IM) for classifying URLs. Although they used several URL-based features, they also take advantage of a number of IM-specific features such as message timing and content. Yet, this algorithm needs more URL message samples to make their experiments more accurate and convincible. Watkins et al. [3] a strategy focused on an integral feature of Big Data was launched in 2017: the overwhelming majority of network processing in a historically guarded business (i.e., using defense-in-depth) is non-malicious. The core objective of Bilge et al. [10] work is to build an exposure system that is designed to detect such domains in real time by applying 15 unique features grouped into four categories. Although URLs cyber security models have been studied for nearly many decades, there is still room to make it more efficient and practical in the real application.

According to the aforementioned review, it can be found that past studies were primarily devoted to (1) Blacklisting, which cannot predict the status of previously URLs or systems based on site content or behavior assessments that require visits to potentially risky sites. (2) Not addressing the issues related to the selection of optimal feature set from the pool of extracted features. In general, with appropriate classifiers, it is feasible to automatically shift through comprehensive feature sets (i.e., without requiring domain expertise) and identify the most predictive features for classification. However, to best of my knowledge, little attention has been paid to devising a new bio-inspired feature selection technique for a malicious URL detections system that relies on a big number for training samples (big data environment). Most of the current bio-inspired optimization techniques for malicious URL detection depend on combining two or more algorithms to enhance the exploration and exploitation fitness of the basic algorithm. The next Section discusses in detail the suggested model that integrates the Hadoop framework to handle big data with a bio-inspired artificial Bee colony algorithm for URL classification.

3 The Proposed Model

The work presented in this paper explores how to detect malicious Web sites from passive DNS based features. This issue naturally leads to new algorithms in which biased features may be chosen while Malicious URLs are constantly being spread. So, the suggested model adapts a bio-inspired feature selection technique to choose an optimal feature set in order to reduce the cost and running time of a given system, as well as achieving an acceptably high recognition rate. Moreover, a two-step artificial bee colony (ABC) algorithm is utilized for efficient data clustering. The two approaches are incorporated within a unified framework that operates on the top of the Hadoop infrastructure to deal with large samples of URLs. Herein, a modified representation learning model based on genetic algorithm is proposed to select the most representative features that keep the classification accuracy as high of the state of the art models that use hundreds or thousands of features, allowing possible embedded programs to run fast looking for the characteristics that match malicious URL behavior. Moreover, this model requires large datasets to train and to tune the learning prediction algorithm. To address this problem, Apache Hadoop is employed as a distributed computing platform. Figure 1 shows the main components of the suggested prediction model, and how these components are linked together and the following subsections discuss its steps in detail.

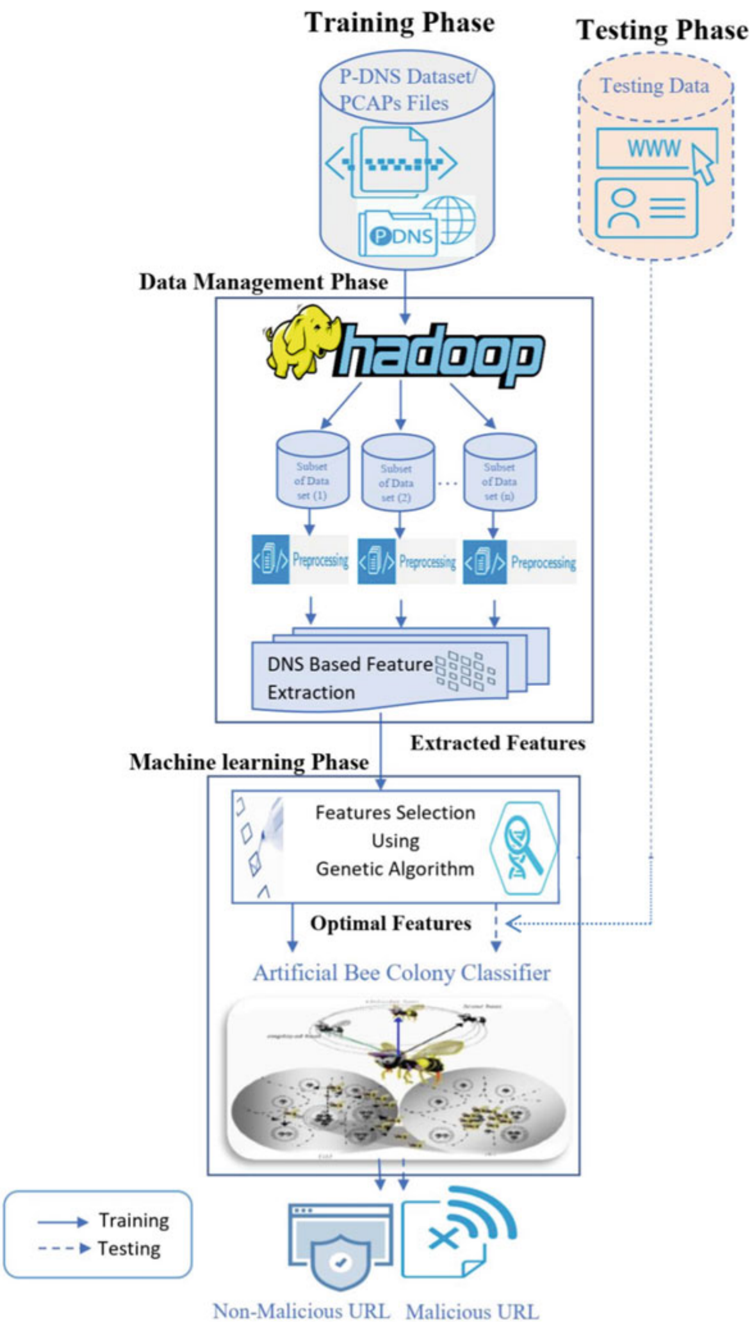

Fig. 1 The proposed malicious URL detection model

3.1 Training Phase

Step 1: Passive DNS Dataset. The challenges of access to DNS data faced by the research community lie in two aspects [2]: (1) first is the data collection phase; the peculiarity of many existing DNS-based malicious domain detection techniques is that they work best in big data scenarios. Thus, they may not be able to produce meaningful results on datasets collected in small networks. Meanwhile, integrating data from DNS servers belonging to different organizations would often face significant bureaucratic/legal obstacles, due to the sensitive nature of DNS logs. (2) Even a bigger challenge lies in data sharing. Unfortunately, security related data are notoriously sensitive and hard to share. In general, Passive DNS data collection happens through the installation of sensors to DNS servers or the connection to DNS server logs for the purpose of obtaining real DNS queries. Furthermore, passive DNS data are linked to the behavior of individual users, so passive DNS data could be used to detect malicious domains with techniques that rely on user-level features (e.g., temporal statistics of user queries).

Step 2: Data Management Phase. Traditional computing storage platforms like relational databases do not scale effectively against the onslaught of big data challenges posed by malicious URL detection. There should be only two types of headings. The headings of the lower level stay unnumbered and formatted as run-in headings. To address this problem, some authors suggested using distributed computing platforms such as Apache Hadoop. Hadoop, an open-source distributed storage platform that can run on commodity hardware, has been utilized to better accommodate Big data processing requires of massive volume and high speed along with heterogeneous data structures theoretically very complex. Hadoop provides a software framework for distributed storage and distributed computing. It divides a file into the number of blocks and stores it across a cluster of machines. It does distributed processing by dividing a job into a number of independent tasks. These tasks run in parallel over the computer cluster.

Hadoop MapReduce includes several stages [11]: In the first step, the program locates and reads the «input file» containing the raw data. As the file format is arbitrary, there is a need to convert data into something the program can process. The «InputFormat» and «Record Reader» does this job. InputFormat uses the InputSplit function to split the file into smaller pieces. Then the Record Reader transforms the raw data for processing by the map. It outputs a list of key-value pairs. Once the Mapper processes these key-value pairs, the result goes to «OutputCollector». There is another function called «Reporter» which intimates the user when the mapping task finishes. In the next step, the Reduce function performs its task on each key-value pair from the Mapper. Finally, Output Format organizes the key-value pairs from Reducer for writing it on HDFS.

Step 3: Data Preprocessing. Data pre-processing is an important phase in machine learning, since the quality of the data and its useful information affects the capacity of the proposed model to learn directly; therefore, it is extremely important that the data are preprocessed before feeding it into the model [12]. Data preprocessing is the process of simply transforming raw data into an understandable format. Preprocessing involves various steps that help to convert raw data into a processed and sensible format such as data cleaning, data integration, data transformation, and data reduction. The main use of the cleaning step is based on detecting incomplete, inaccurate, inconsistent and irrelevant data and applying techniques to modify or delete this useless data.

Step 4: Feature Extraction. In machine learning, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set [6]. Content-features usually require downloading the web-page, which would affect the feature collection time. In general, the success of a machine learning model critically depends on the quality of the training data, which hinges on the quality of feature representation. Given a URL $u \in \mathbb{U}$ where $\mathbb{U}$ denotes a domain of any valid URL strings, the goal of feature representation is to find a mapping $g : \mathbb{U} \rightarrow \mathbb{R}^d$ such that $g(\mathbb{U}) \rightarrow X$ where $X \in \mathbb{R}^d$ is a d-dimensional feature vector, that can be fed into machine learning models. The process of feature representation can be further broken down into two steps:

- **Feature Collection**: This phase is engineering oriented, which aims to collect most if not all relevant information about the URL.
- **Feature Preprocessing**: In this phase, the unstructured information about the URL is appropriately formatted and converted to a numerical vector so that it can be fed into machine learning algorithms.

The suggested model relies on DNS Answer-based, TTL Value-based, and Domain-Name-based features. See [3] for more details.

Step 5: Feature Selection Using Genetic Algorithm. It is essential to select a subset of those features which are most relevant to the prediction problem and are not redundant. Heuristic search is an intelligent search process through an extremely wide range of solutions to detect a satisfactory solution. No exhaustive sequential selection process can generally be guaranteed for the optimal subset; any ordering of the error probabilities of each of the 2^n feature subsets is possible. In this case, an instance of a GA-feature selection optimization problem can be described in a formal way as a four-tuple (R, Q, T, f) defined as [13, 14]:

- ***R*** is the solution space (initial population—a combination of 16 feature vector per URL—a matrix n × 16) where n represents the number of URL samples. Each bit is signified as a gene that represents the absence or existence of the feature within the vector. Every feature vector is represented as a chromosome.
- ***Q*** is the feasibility of predicate (different operators—selection, crossover, and mutation). The crossover is the process of exchanging the parent's genes to

produce one or two offspring. The purpose of mutation is to prevent falling into a locally optimal solution of the solved problem [14]. A uniform mutation is employed for its simple implementation. The selection operator retains the best fitting chromosome of one generation and selects the fixed numbers of parent chromosomes. Tournament selection is probably the most popular selection method in genetic algorithm due to its efficiency and simple implementation.

- ζ is the set of feasible solutions (new generation populations). With these new generations, the fittest chromosome will represent the URL feature vector with a set of salient elements. This vector will specify the optimal feature combination explicitly according to the identification accuracy.
- f is the objective function (fitness function). The individual that has higher fitness will win to be added to the predicate operators mate. Herein, the fitness function is computed based on accuracy Acc value that shows the difference between the real URL's classification, and it's computed one.

$$Acc = (True\ Positive + True\ Negative)/(no.\ Positive + no.\ Negative) \tag{1}$$

Accuracy (Acc) is the ratio of the correctly identified domains to the whole size of the test set. The higher the value is, the better ($Acc \in [0, 1]$). True Positive (TP) is the correctly identified malicious domains, True Negative (TN) is the correctly identified benign domains, P is the total number of malicious domains, and N is the total number of benign domains.

Step 6: Artificial Bee Colony (ABC) Classifier. The final step includes employed an artificial bee colony to classify the URLs malicious or benign based on the training dataset that contains the best feature vector of each URL. In this case, to accelerate the convergence rate and maintaining the balance between exploration and exploitation, in this research work two-step ABC algorithm is utilized to improve the ABC algorithm for clustering problems by using K-means algorithm [15]. The combination of ABC and K-means (named ABCk) uses the merits of the k-means and ABC algorithms for solving the problem of malicious URL classification. In this case, the suggested model uses sensitivity as a fitness function. Sensitivity is defined in Eq. 2 as the ratio of the True Positives to the sum of the True Positives and the False Negatives. The True Positives are the correctly identified malicious domains, and the False Negatives are the domains that are malicious but were incorrectly identified as non-malicious.

$$\text{Sensitivity} = \frac{True\ Positive}{True\ positive + False\ negative} \tag{2}$$

3.2 Testing Phase

In this step, given the unknown URL, the model starts with extracting the features vector for this URL that follows the indices of the best features vector learned from the training stage. This extracted feature vector is then classified according to its similarity to the final cluster centers generated from applying the artificial bee colony classifier in the training phase.

4 Experimental Analysis and Results

In this section, many experiments are conducted to validate the performance of the suggested malicious URL detection model and compare it with some common detection techniques. The performance is validated in terms of precision, false positive rates, Accuracy, True Negative rate, Recall, and F-Measure based. The experiment was carried out in Intel Xeon E5-2620v3 @ 2.4 GHz (12 CPUs) processor with 32.00 GB RAM implemented in Java. The experiments are conducted using a benchmark dataset that is comprised of captured passive DNS data, which are answers (e.g., IP address, time to live (TTL), record counts) from authoritative DNS servers given domain name queries from the browsers of users. A big dataset totaling 184 million rows is extracted.

- **Experiment 1: (The significance of features selection)**

  Aim: To validate the benefits of employing a feature selection module within the suggested model; this experiment implements the suggested model using both full features vector and optimal features vector to investigate the difference between the two runs in terms of detection accuracy and time.
  Observations: The results in Figs. 2 and 3 reveals that the use of optimal features achieves an increase in accuracy in terms of True Positive, True Negative, False Positive and False Negative of approximately 1% compared to using full feature vector. Although this increase is relatively small, the benefit is to reduce the testing time for each URL from approximately one second to 500 ms.
  Discussions: As the proposed model tries to select of the most prominent features that contain the URLs characteristics which is able to distinguish Web sites either malicious or benign, so this features vector as expected yields increasing in detection accuracy. One possible explanation of these results is that the feature selection module able to remove redundant features (high correlated features) and discards features leading to mislabelled based on fitness function.

- **Experiment 2: (Classifier evaluation)**

  Aim: Since there is a wide range of supervised classification algorithms, this set of experiments is conducted to assess a sample of the collected dataset

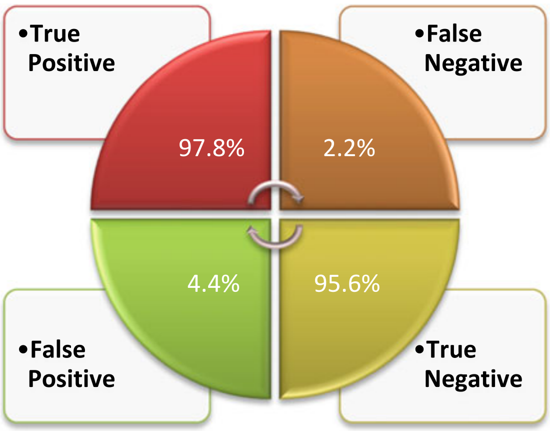

Fig. 2 Confusion matrix using optimal features

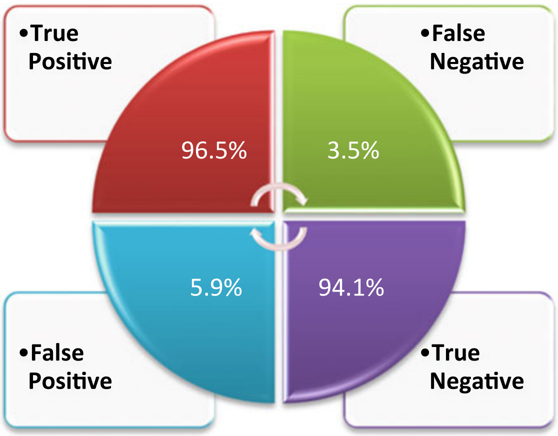

Fig. 3 Confusion matrix using all features

according to three classifiers using Weka [16] as well as ABC and two-step ABC classifiers. The three classifiers were tested covering tree-based (Random Forest, C4.5) and function-based (SVM). The classification was made without parameters tuning through a ten-fold cross-validation as a first step to select the most promising approach.

Observations: Results for accuracy, true positives and true negatives are given in Fig. 4 for each classifier. Among the tested classifiers, SVM yields the worst accuracy (86.31%) while being efficient in identifying legitimate URLs (93.1%). Tree-based classifiers have approximately the same performance (around 90%) with disproportionate true positives and true negatives. The ABC classifier has approximately the same performance of random forest classifier. The best performer is two-step ABC classifier, correctly classifying 96.6% of URLs, being the best.

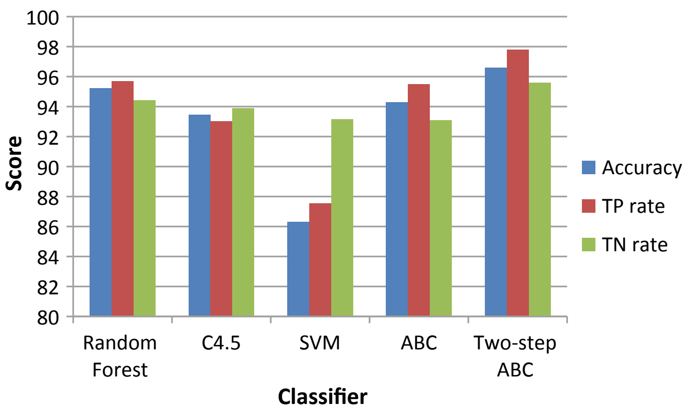

Fig. 4 Classification results for 6 classifiers

Discussions: ABC classifier takes a long time because of its stochastic nature. It is also observed that the position of bees and food sources is identified randomly initially which takes more time for optimizing, especially in clustering problems. The solution search equation of ABC is good at exploration, but poor at exploitation, which results in the poor convergence. It is also noted that the convergence speed of ABC algorithm is decreased as dimensions of the problem are increased. In two-step ABC algorithm, the initial positions of food sources are identified using the K-means algorithm instead of random initialization. So, it yields more accurate classification.

- **Experiment 3: (Concept drift)**

Aim: Phishing tactics and URL structures keep changing continuously over time as attackers come up with novel ways to circumvent the existing filters. As phishing URLs evolve over time, so must the classifier's trained model to improve its performance. In general, retraining algorithms continuously with new features is crucial for adapting successfully to the ever-evolving malicious URLs and their features. An interesting future direction would be to find the effect of variable number of features using online algorithms in detecting phishing URLs. Herein, the whole passive DNS data set is divided into 15 patches; each patch contains approximately 1,200,000 samples with different numbers of benign and malicious URLs. The suggested model was trained with each patch separately to validate its classification error rates using each patch as a training set.

Observations: Fig. 5 shows the classification error rates for the suggested classifier after training using different patches. The x-axis shows the patch number in the experiment. The y-axis shows the error rates on testing the suggested classifier. Figure 5 reveals that the error rate fluctuates between 2.2

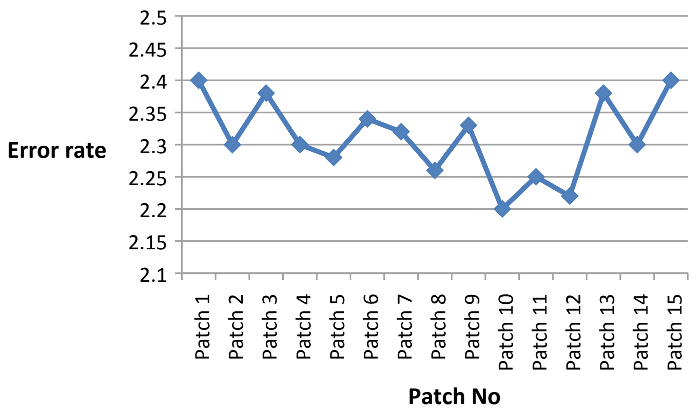

Fig. 5 Error rates of the suggested classifier after training them on different patches

and 2.4% for all patches. So, the suggested model demonstrates the stability regarding error rate despite the changing the dataset used.

Discussions: one possible explanation for these results is that the suggested model is built based on optimal features vector that has discriminative ability to distinguish between benign and malicious URLs. This feature vector minimizes the inter-class similarity while maximizes intra-class similarity. The inter-class cluster show the distance between data point with cluster centre, meanwhile intra-class cluster show the distance between the data point of one cluster with the other data point in another cluster.

5 Conclusion

DNS data carry rich traces of the Internet activities, and are a powerful resource to fight against malicious domains that are a key platform to a variety of attacks. To design a malicious domain detection scheme, one has to consider the following major questions that hinder the advances of the field: (1) data sources: what types of DNS data, ground truth and auxiliary information are available; (2) features and data analysis techniques: how to derive features to match intuitions of malicious behaviors, and what types of detection techniques the malicious domain discovery problem can be mapped to; (3) evaluation strategies and metrics: how to evaluate the robustness of a technique given the adaptive nature of attackers, and what metrics to use for these purposes.

The work presented in this paper proposes a new model to discover malicious domains by analyzing passive DNS data. This model takes advantage of the dynamic nature of malicious domains to discover strong associations among them, which are further used to infer malicious domains from a set of existing known malicious ones. The central finding of the work in this paper is that machine learning techniques can alleviate the disadvantages of blacklists, heavyweight (use more features and so have a higher accuracy) and lightweight (useless features and consumes the features from the browser) classifiers for detecting malicious URLs. This paper demonstrated that the approach of using an optimal number of features (middleweight) extracted from the original features vector had clear advantages over larger feature sets. One of the major contributions of the work in this paper was to explore a hybrid machine learning technique (K-mean and ABC) that used selected discriminative features and the use of the Hadoop framework to handle the big size of URLs data. The results demonstrated that the suggested model capable of scaling 10 million query answer pairs with more than 96.6% accuracy. The major limitation of the suggested model is that it cannot take an arbitrary given domain and decide whether it is potentially malicious or not. Similarly, if a domain never shares IPs with other domains, it will not appear in the domain graph, and the suggested model is not applicable to such domain either. Future endeavors in this regard include improving the level of filtering of non-malicious data to provide the analysts with an even smaller set of candidate data to investigate.

References

1. Sayamber, A., Dixit, A.: Malicious URL detection and identification. Int. J. Comput. Appl. **99**(17), 17–23 (2014)
2. Zhauniarovich, Y., Khalil, I., Yu, T., Dacier, M.: A survey on malicious domains detection through DNS data analysis. ACM Comput. Surv. **51**(4), 1–36 (2018)
3. Watkins, L., Beck, S., Zook, J., Buczak, A., Chavis, J., Mishra, S.: Using semi-supervised machine learning to address the big data problem in DNS networks. In: Proceedings of the IEEE 7th Annual Computing and Communication Conference (CCWC), pp. 1–6, USA (2017)
4. Sahoo, D., Liu, C., Hoi, S.: Malicious URL Detection Using Machine Learning: A Survey. arXiv preprint arXiv:1701.07179, pp. 1–21 (2017)
5. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N., Dagon, D.: Detecting malware domains at the upper DNS hierarchy. In: Proceedings of the 20th USENIX Conference on Security (SEC'11), pp. 1–16, USA (2011)
6. Ma, J., Saul, L., Savage, S., Voelker, G: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceeding of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254, France (2009)
7. Zhang, Y., Hong, J., Cranor, L.: CANTINA: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web, pp. 639–648, Canada (2007)
8. Kan, M.-Y., Thi, H.: Fast webpage classification using URL features. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 325–326, Germany (2005)
9. Guan, D., Chen, C., Lin, J.: Anomaly based malicious URL detection in instant messaging. In: Proceedings of the Joint Workshop on Information Security, Taiwan (2009)
10. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: EXPOSURE: a passive DNS analysis service to detect and report malicious domains. ACM Trans. Inf. Syst. Secur. **16**(4), 1–28 (2014)
11. Manikandan, S., Ravi, S.: Big data analysis using Apache Hadoop. In: Proceedings of the International Conference on IT Convergence and Security (ICITCS), pp. 1–4, China (2014)
12. Figo, D., Diniz, P., Ferreira, D., Cardoso, J.: Preprocessing techniques for context recognition from accelerometer data. Pers. Ubiquit. Comput. **14**(7), 645–662 (2010)

13. El-Sawy, A., Hussein, M., Zaki, E., Mousa, A.: An introduction to genetic algorithms: a survey, a practical issues. Int. J. Sci. Eng. Res. **5**(1), 252–262 (2014)
14. Sivanandam, S., Deepa, S.: Introduction to Genetic Algorithms. Springer, USA (2007)
15. Kumar, Y., Sahoo, G.: A two-step artificial bee colony algorithm for clustering. Neural Comput. Appl. **28**(3), 537–551 (2015)
16. Veček, N., Liu, S., Črepinšek, M., Mernik, M.: On the importance of the artificial bee colony control parameter 'Limit'. Inf. Technol. Control **46**(4), 566–604 (2017)