IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON EMERGING APPROACHES TO CYBER SECURITY

Received December 17, 2019, accepted January 5, 2020, date of publication January 27, 2020, date of current version February 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969780

An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs

AYMAN EL AASSAL, SHAHRYAR BAKI, AVISHA DAS, AND RAKESH M. VERMA, (Member, IEEE)
Department of Computer Science, University of Houston, Houston, TX 77204, USA

Corresponding author: Rakesh M. Verma (rverma@uh.edu)

This work was supported in part by the NSF under Grant CNS 1319212, Grant DGE 1433817, and Grant DUE 1356705, in part by the U.S. Army Research Laboratory, and in part by the U.S. Army Research Office under Contract W911NF-16-1-0422.

ABSTRACT We perform an in-depth, systematic benchmarking study and evaluation of phishing features on diverse and extensive datasets. We propose a new taxonomy of features based on the interpretation and purpose of each feature. Next, we propose a benchmarking framework called 'PhishBench,' which enables us to evaluate and compare the existing features for phishing detection systematically and thoroughly under identical experimental conditions, i.e., unified system specification, datasets, classifiers, and evaluation metrics. PhishBench is a first in the field of benchmarking phishing related research and incorporates thorough and systematic evaluation and feature comparison. We use PhishBench to test methods published in the phishing literature on new and diverse datasets to check their robustness and scalability. We study how dataset characteristics, e.g., varying legitimate to phishing ratios and increasing the size of imbalanced datasets, affect classification performance. Our results show that the imbalanced nature of phishing attacks affects the detection systems' performance and researchers should take this into account when proposing a new method. We also found that retraining alone is not enough to defeat new attacks. New features and techniques are required to stop attackers from fooling detection systems.

INDEX TERMS Feature engineering, feature taxonomy, framework, phishing email, phishing URL, phishing website.

I. INTRODUCTION

Phishing is a popular form of social engineering attack wherein the attacker deceives a victim through impersonation. Emails and messages with malicious attachments or poisoned URLs (Uniform Resource Locator) redirecting to malicious websites are a few of the common attack vectors used in phishing. Technological advancement has provided phishers with better tools to launch dangerous and sophisticated attacks. The 2018 Phishlabs report on phishing trends [60] mentions that the targets of phishing attacks shifted from individuals to enterprises. To make matters worse, phishers now have access to free SSL certificates. Nearly half of all phishing websites currently use HTTPS, which was one of the major indicators of the legitimacy of websites [45]. Another report published by APWG in the first quarter of 2019 states that the number of phishing attacks increased by 30% from the previous quarter and that the primary targets were the Software-as-a-Service (SaaS) and webmail services [3].

Phishing has become such a pernicious threat that researchers have devoted increasing attention to combating it since 2004. The query 'phishing' on DBLP[1] shows approximately 55 research papers on average are published every year that address phishing attacks and their detection for the email, website and URL vectors. The literature on phishing also includes several surveys, e.g., [12], [19], that try to compare these techniques at least at a theoretical level. However, the research on phishing detection raises several questions that have not been adequately addressed in previous literature. How to choose the most appropriate phishing detection technique in a given specific scenario? What does a good solution mean in this context? More fundamentally, are the claims of accuracy, and other metrics, made by research papers true and realistic?

As a security challenge, phishing has several parameters which need careful attention for a good solution [52], [58].

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

[1] https://dblp.uni-trier.de/

These include: (i) *The active attacker* who is constantly learning the defensive methods and trying to break them, (ii) *diverse and representative datasets*, (iii) *imbalanced scenarios and use of proper metrics for evaluation*, and (iv) *real-time or near real-time detection*. Researchers often fail to consider how their features can be defeated by active attackers. They sometimes demonstrate their techniques on non-representative datasets like URLs from Alexa.com, which only contain domain names. They generally use balanced datasets even though phishing is an imbalanced classification problem since the number of legitimate samples is higher than phishing ones. The lack of new public datasets results in training on old data, which could affect the detection rate for new and improved phishing attacks.

Owing to this huge parameter space it is difficult to identify a technique that is optimal across all the parameters. Different methods have been published using a variety of features and classifiers and tested on different datasets. However, no framework was published that can systematically and efficiently compare different methods, features, and classifiers on the same datasets. Consequently, a benchmarking study is the need of the day. This benchmarking exercise, however, is non-trivial and poses an array of unique challenges.

- Given the huge diversity of features and classifiers employed by proposed techniques, a universal framework for phishing is quite difficult to be summarized and abstracted. Nonetheless, building a systematic framework is critical to compare, analyze and diagnose the techniques from a common viewpoint (under identical experimental conditions).
- In this study, we need to either collect the code from the researchers or re-implement their methods. Given the large body of work, this itself is a significant challenge. Furthermore, authors often do not give clear enough description of the features and implementation methods they used, which makes it a challenging task to reproduce the experiments in the literature.
- When proposing new features or approaches, authors often test their work across limited metrics, datasets, and/or parameters. To evaluate the techniques as comprehensively as possible, we need to identify a suite of metrics and a diverse set of parameters that consider the security challenges and characterize all aspects of the phishing detection problem.

We present in this paper a benchmarking framework, PhishBench, which can help researchers in the field of phishing detection by providing a template to develop new methods and features as well as a platform to compare their proposed techniques with previous works. We also review the features used in the literature, propose a taxonomy of features and test previous methods on new datasets[2] using PhishBench. To summarize, we make the following contributions:

- We study the different classes of attributes that have been used in previous literature and propose systematic taxonomies of features for URLs, websites, and emails (Section IV) based on how these features are interpreted on the victim's end. We further identify possible sources for new features. We also determine the features that lead to the best detection performance and have the fastest extraction time in the context of phishing detection (Section VII-D).
- We implement a flexible and customizable benchmarking framework, PhishBench (Section V), and collect diverse datasets (Section VI) to facilitate research on phishing detection. The framework includes a total of 226 (83 URL and Website and 143 email) features along with the top machine learning algorithms for supervised learning, deep neural learning, online learning, and imbalanced learning. We plan to make PhishBench available for future research.
- Leveraging PhishBench, we conduct a systematic benchmarking study and an in-depth evaluation of phishing features and techniques including automated machine learning suites like AutoML and TPOT (Section VII). The experiments include comparing classifiers' performance on balanced as well as imbalanced datasets, running scaling experiments, studying feature importance and finally, comparing with prior research work.

Paper Organization: The next section compares PhishBench with other tools and gives an overview of previous related works. Section III includes some necessary background. Proposed feature taxonomies are motivated and presented in Section IV. Section V contains a description of PhishBench and its utility. Datasets used in the literature, as well as our datasets, are discussed in Section VI. Experiments are reported in Section VII along with analyses to answer the questions posed above. Section VIII concludes.

II. RELATED WORKS

We divide the related works into two sections: (1) works that provided a benchmarking framework, and (2) studies on feature importance.

A. BENCHMARKING FRAMEWORKS

In the machine learning era, there are several tools that researchers can use instead of implementing everything from scratch. Scikit-learn for Python developers, Weka for Java developers, Caret for R developers, and several stand-alone tools such as RapidMiner and SPSS are available so that people can build their frameworks. These tools provide various kinds of machine learning and data mining methods but require some effort to build a benchmarking system. Someone needs to write the code for feature extraction (if the features are not already collected) and feed the features to the system.

New automated machine learning (AutoML) frameworks have been developed to help users, who do not have any expertise in this field, to deploy and run machine learning

[2]Information on how to obtain a copy of these datasets can be found at http://www2.cs.uh.edu/~rmverma/datasets.html.

TABLE 1. Comparison of existing frameworks with PhishBench. *New*: Whether adding new features is possible.

Source	Feature Extraction		Learning Methods	Automation	Imbalanced Methods	Feature Importance		Ease of Use	GUI
	New	Existing				Selection	Ranking		
RapidMiner	✗	✗	32[a]	✓	✗	✓	✓	High	✓
Scikit-learn	✗	✗	42[b]	✓[d]	✗	✓	✓	Low	✗
Weka	✗	✗	45[c]	✗	✗	✓	✓	High	✓
PhishBench	✓	✓	31	✓[e]	✓	✓	✓	Moderate	✗

[a] https://rapidminer.com/products/studio/feature-list/
[b] https://scikit-learn.org/stable/user_guide.html
[c] http://weka.sourceforge.net/doc.dev/weka/classifiers/Classifier.html
[d] Not part of Scikit-learn - https://automl.github.io/auto-sklearn/master/
[e] Uses AutoSklearn and TPOT

algorithms. Several automated systems can be found on the AutoML website[3] including TPOT [46] and H2O [54]. Some AutoML frameworks use multiple preprocessing steps (scaling, feature selection, etc.) on the given data and train several machine learning algorithms along with hyperparameter tuning to find the best model. The models can also be ensembled or stacked to find the best combination. The downside to this solution is the long time it takes to find the best pipeline - taking somewhere between hours to even days. These frameworks generally include options to stop the algorithm from running and get the results at a specific time, however, there is no guarantee that the provided solution is the best model for the problem. Also, these systems evaluate their models on a dataset of preprocessed and extracted features and do not offer feature extraction modules, contrary to PhishBench.

Despite considerable research on detecting phishing URLs, websites, and emails, there is no existing framework that collates commonly used attributes and algorithms in phishing detection. Most researchers do not make their code publicly available. Moreover, there is a lack of common datasets that researchers can use to compare newer techniques with previous work. However, the authors of URLNet system [36] have made their system available online.[4] Their system leverages deep learning architecture (using character and word embedding) to analyze URLs. Although URLNet just covers one learning method (Neural Networks) and a specific feature, other researchers can still add and use it as part of their systems. We also found three other available sources for phishing detection [5], [39], [51], but none of them have any feature extraction nor different types of detection methods. They just used a fixed learning method (Random Forest, Naive Bayes, and Covariance Matrix) on an already prepared dataset with a set of fixed features.

Comparison with existing frameworks:

Table 1 summarizes similar automated frameworks that have been used for machine learning-based tasks. We did not find any comprehensive framework designed specifically for phishing to include in this table. We look at whether these automated systems allow the addition of new or existing feature extraction methods. We only add the number of supervised methods under the "learning methods" column to enable a direct comparison with PhishBench.

Other criteria are whether the frameworks implement automated learning algorithms (AutoSKLearn, TPOT, etc.), include methods to handle imbalanced datasets, implement feature importance (selection and/or ranking), ease of use (the level of coding required) and the type of user interface. Weka and RapidMiner have a Graphical User Interface (GUI) implemented, which makes them easier for people with lower programming skills. Building a machine learning workflow in Scikit-learn is not straightforward and requires knowledge of Python. Although PhishBench does not offer a GUI, being able to change its workflow by turning different options on and off in the configuration file makes it more user friendly and easier to use than Scikit-learn. In the Ease of Use column of Table 1, we use *high* when no programming is required and *low* when significant prior coding knowledge is essential to add a function or algorithm.[5] We define ease of use as *moderate* when users need some effort to extend the functionalities, as for PhishBench (Section V-B). The top three rows of Table 1 show the need for a convenient framework that enables researchers to implement and evaluate new phishing detection features and models without too much effort.

B. FEATURE IMPORTANCE

Several studies analyzed the importance and impact of the features used for learning [2], [14], [15], [17], [18], [20], [22], [26], [28], [29], [37], [40], [44], [56], [64]. Table 2 summarizes them from four aspects: 1) feature ranking methods, 2) top five features, and 3) dataset ratio and 4) dataset sources. For URL classification, we observe the prevalence of lexical features under syntactic and semantic categories and a few pragmatic network features. We will describe these categories in more detail in Section IV: Features and Taxonomies. URL features tend to dominate also in the website classification papers. However, this could be due to the infrequent use of website features. Note that there are some conflicts between these results. Potential reasons for these conflicts are the use of different datasets or different criteria for ranking features. In this paper, we conducted a thorough study using diverse datasets and multiple runs to address these issues.

[3] https://www.ml4aad.org/automl/
[4] https://github.com/Antimalweb/URLNet
[5] To add algorithms to Scikit-learn, please look at https://scikit-learn.org/stable/faq.html#what-are-the-inclusion-criteria-for-new-algorithms

TABLE 2. Comparison of previous works on feature importance.

Category	Paper	Dataset	Ratio	Metric	Top Features
URL	[15]	Alexa, DMOZ PhishTank Openphish	6:1	IG	HostName 1-4Gram URL QuadGram
	[26]	Twitter API	4:1	IG	Type of referring domains Link/domain creation difference Domain age, Link creation hour Type of encoders
	[22]	DMOZ Phishtank	N/A	IG, Chi-2	caramispa_com, paypal_com cgi_bin, uk_webapp, com_uk
	[37]	Twitter APIs	7:1	F-score	Account creation date Number of sources Number of initial URLs Tweet text similarity Follower-friend ratio
	[40]	Alexa (crawled) Openphish	4:1	IG, CFSS[a]	Domain token length TLD, URL & Domain length File name length
Website	[18]	Facebook's API	100:1	IG	Facebook.com URL URL param. length Post type, Link field length No. of parameters in URL
	[24]	Private	N/A	Odds Ratio	Host based obfuscation Organization in URL's path word: 'confirm', 'banking', 'signin'
	[48]	Millersmiles Phishtank Yahoo directory	N/A	IG, Chi-2	SSLfinal state, URL of anchor Web traffic, Having sub-domain prefix/suffix
	[30]	UCI Machine Learning Repo.	1.2:1	Omitting redundant features	having-Sub-Domain, SSLfinal-State Domain-registration-length Favicon, URL-of-Anchor
	[44]	Alexa PhishTank	4:1	mean impurity decrease	Avg. cyclomatic complexity # of landing page variants avg. number of external blocks Set-Cookie, lines of code (LOC)
Email	[33]	Private	2:1	IG, Chi-2	Alexa rank, word "n't" Number of replies Number of attachments word 'something'
	[29]	SpamAssassin Nazario	2:1	IG	SpamAssassin, Urlnumlink Bodyhtml, Urlnumperiods Sendnumwords
	[2]	SpamAssassin Nazario	1:1	IG	SpamAssassin, Bodyhtml Urlnumlink, Urlnumperiods URL target/text mismatch
	[56]	SpamAssasin Nazario	1:1	IG	No. of function words Body richness, Subject richness No. of characters, No. of URLs
	[64]	SpamAssasin Nazario	1:1	IG	Bodyhtml, Hex characters Number of domains URL different from anchor Urlnumperiods

[a] *Correlation-based Feature Subset Selection*

Another problem with the above rankings is that most of them used IG, which is known to prefer features that take a large number of values. Other techniques, e.g., Gain Ratio and Chi-squared, can be used to avoid this problem. Authors in [28] used Gain Ratio and Symmetrical Uncertainty besides IG but they only tested a small number (7) of features. Based on their results, the ranking for all three methods was almost similar (''Email's subject blacklist word'' and ''IP-Based URL'' were the best features). Researchers in [14] also used gain ratio, Fisher score, and Chi-square on a dataset of legitimate and malicious emails (emails with a malicious attachment). They only used the features that can be extracted from the email itself without using external services. Since their goal was detecting emails with malicious attachments, their set of best features are quite different from the above-mentioned works, *type of attachment*, *content type*, *content disposition* and *email span time*[6] were the top features for their dataset.

Authors in [22] used classification on a different subset of features for URL detection instead of using feature selection methods. They showed that host-based features have the best performance (94%) on their dataset and lexical features have the lowest performance (84%). The same method has been used by [37] for twitter suspicious URL detection. Their results showed that the similarity of the account creation dates, the relative number of source applications, and the relative number of initial URLs (beginning URLs that redirect users to the same URL) are important features.

Previous works rarely report time information. They usually consider a narrow set of classifiers and metrics and use balanced datasets without checking for diversity. In this paper, we do a thorough study of feature importance using diverse datasets.

[6] difference between the email's sent and received time.

III. PRELIMINARIES

Term Frequency - Inverse Document Frequency (TFIDF) is a popular statistical feature. It is a term weighting scheme that uses term frequency in a document and log of the inverse popularity of the term in the collection [50]. It is defined by Equation (1), where $n_{t,d}$ is the number of times term t appears in a document d, N_d is the total number of terms in d, D is the total number of documents, and d_t is the number of documents containing the term t. In our feature extraction, the TFIDF vector extracted from an email or website content is considered as a single feature.

$$TFIDF(t,d) = \frac{n_{t,d}}{N_d} * log_e(\frac{D}{1+d_t}) \tag{1}$$

We use the below metrics to report our results.

- Confusion Matrix: The total number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) identified by the classifier. Reported as raw values or percentages. We also define P = TP + FN and N = TN + FP.
- Accuracy: The ratio of correctly classified instances versus the total number of instances (Equation (2)).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

- Precision: The ratio of instances correctly identified by the classifier as relevant versus the total number of instances classified as relevant (Equation (3)).

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

- Recall: The ratio of instances correctly identified by the classifier as relevant versus the total number of instances that are relevant (Equation (4)).

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

- F_1-score: The harmonic mean of precision and recall (Equation (5)).

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

- Geometric Mean: The geometric mean of true negative rate (specificity) and recall (Equation (6)).

$$G - Mean = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} \tag{6}$$

- Balanced Detection Rate [1]: A metric to measure the number of minority class instances that were correctly classified and to penalize appropriately using the incorrectly classified instances of the majority class (Equation (7)).

$$BDR = \frac{TP}{1 + FP} \tag{7}$$

- Area Under Curve: When using normalized units, the area under the curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [21].
- Matthew's Correlation Coefficient [6]: It takes into account true and false positives and negatives and is generally regarded as an imbalanced measure that can be used even if the classes are of very different sizes (Equation 8)).

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TN + FN)(P)(N)}} \tag{8}$$

IV. FEATURES AND TAXONOMIES

Over the past decade, researchers have identified and categorized features extracted from phishing attack vectors in various ways. Some papers classify features from the viewpoint of the attack type, some classify them based on how or where they appear in an attack vector [2], [7], [8], [65]. However, to our knowledge, no one has given systematic taxonomies that are independent of detection approaches and cover all the possible features that can be extracted from the attack vectors. Phishing vectors, i.e., emails, websites, and URLs, are all specialized strings. Hence, language/logic categories: syntax, semantics, and pragmatics, can be effectively used to categorize their attributes. We now present two taxonomies: one which groups URL and website features, and the other for emails.

A. TAXONOMIES

Building a taxonomy requires a comprehensive view of the literature, an awareness of potential gaps/opportunities, and a systematic and comprehensive approach. In a previous paper [16], we surveyed more than 250 papers on phishing detection research and user studies. The final taxonomy presented here was achieved after several iterations in building taxonomies and testing them by populating each class with examples. Care was also taken to ensure that the classes on each level were truly ''orthogonal/independent'' in scope and closely matched conceptually or physically so that one could not be nested inside the other. The two taxonomies, shown in Figures 1 and 2, have the main components of a webpage and email respectively as the second levels (the root in each figure is at level one). Each component is subsequently broken down to the following groups: syntactic, semantics and pragmatics, constituting the third level, which we explain below.

- ***Syntactic features*** are based directly on the format and syntactic correctness of the vector whether an email, URL, or website. For example, in a properly constructed URL, the top-level domain or TLD ('.com') should only appear once before the path of the URL. However, that is not always the case in a malicious URL. This issue is relevant to the syntax of URLs, thus we consider the TLD position as a *syntactic* feature. While studying website content, we may simply count all the tags, another syntactic feature. An example of a syntactic email feature would be the format of the body content, e.g., text/plain and text/HTML which are common email formats.
- ***Semantic features*** focus on the meaning and interpretation of the textual content in emails, URLs, and websites. One example of a semantic feature for websites can be the meaning of HTML elements, e.g., the number of hidden objects. Another example of a semantic feature is the presence of a special character such as '@' in a link. In the context of a phishing email, some words can have special meaning and induce urgency e.g., 'act now' or 'click here'. Such words can be blacklisted. Therefore, an attribute that uses blacklisted words for detection will be categorized as a semantic feature.

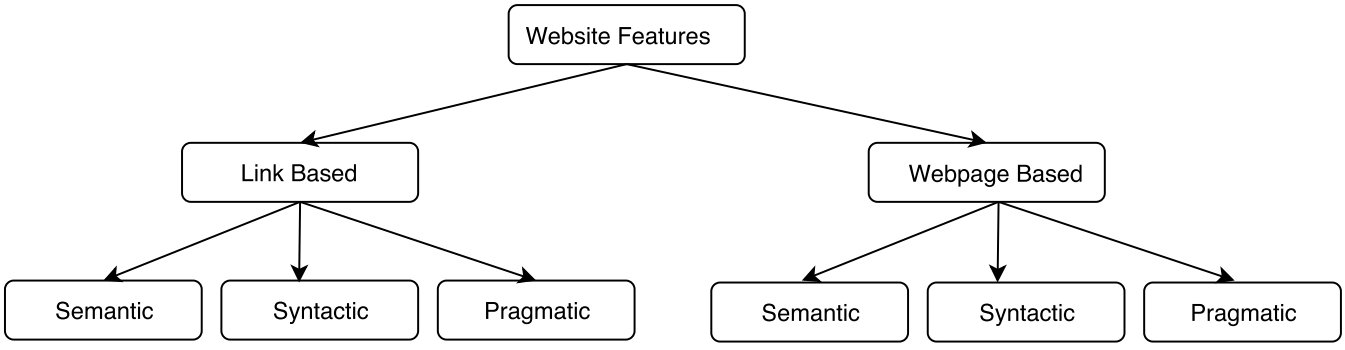


FIGURE 1. URL and Website feature taxonomy. Each category at the bottom level is a subtree which consists of the finer feature categories - ***Lexical, Network and Script*** further described in Section IV-B.

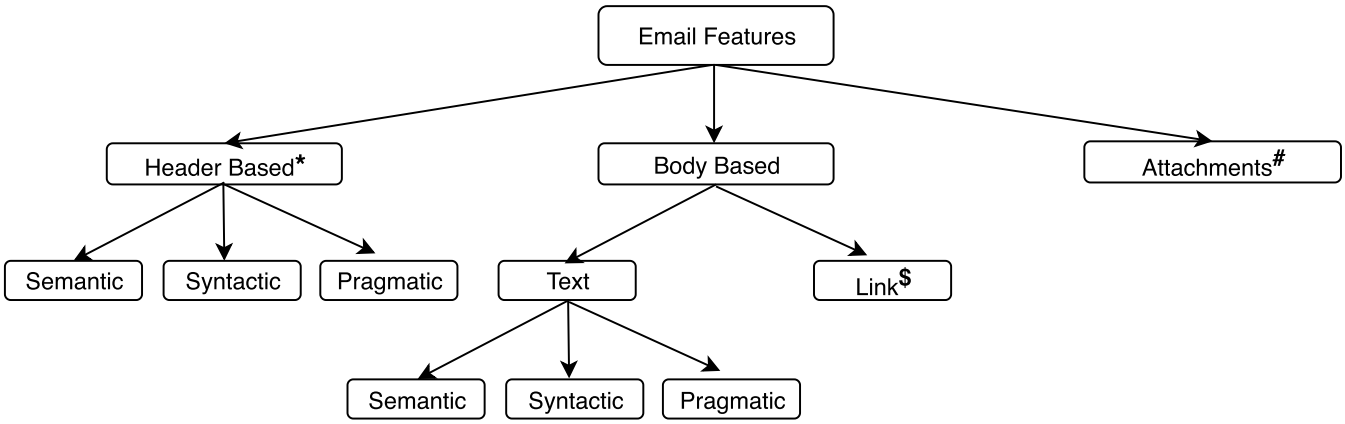


FIGURE 2. Email feature taxonomy. Each category at the bottom level is a subtree which consists of the finer feature categories - ***Lexical, Network and Script*** further described in Section IV-C. * Script-based features cannot be extracted from email headers. $ Please see the subtree on Link Based features under the Website taxonomy. # Attachment features have been described in Section IV-C.

- ***Pragmatic features*** are not directly concerned with the syntax or semantics of the email, URL or website. This class of features has been the least explored in phishing detection literature. For example, disabling right clicks in websites is a technique used by attackers to prevent users from viewing and saving the source code. It does not relate to the syntax of the HTML content nor its semantics. Thus we categorize it as a pragmatic feature. Other examples include the details of a website's registration, the age of the website, etc. In emails, the readability scores of the email body act as a pragmatic feature.

The Bottom Level. The bottom level in the taxonomies, which is not shown in the taxonomy figures, consists of two-three classes: Lexical, Network, and Script. These classes apply to all the leaves in the figures, except for the three leaves in the Header subtree, since scripts are not allowed in the header of an email. We describe them with examples below.

B. URL AND WEBSITE FEATURES

The two important ''actors'' in website phishing attacks are the web browser and the victim. While building our taxonomy, we consider how a web browser and the victim interpret a URL or a website. Our taxonomy builds and improves upon the categorizations in [19], [49].

We implement 51 URL and 32 website features in PhishBench, which encompass all types of major URL and website based attribute classes: Lexical, Network, and Script. For the selection of the implemented features, we study previous literature that uses URLs and websites for phishing detection and reports high performance. We also consider features that are ranked high by feature ranking methods like information gain [47]. We provide examples of URL and website features to make each of these categories easier to understand. For a more comprehensive list of features, we refer the reader to [16].

URL feature classes:

- Lexical: Longest token in the domain (syntactic), presence of the target brand (semantic), and *presence of blacklisted words, whitelisted words* (pragmatic).
- Network: Syntactic category of features include the presence of port number, whether the domain name is an IP address, etc. Under the semantic category, are features such as whether there is a match between the port and the protocol. Pragmatic category of network features include attributes like AS number, WHOIS information, etc.
- Script: Features such as number of functions like escape, eval, etc., in the script, are syntactic. Presence of different file extensions like '.php', '.js', etc. in the URL path (semantic), and time needed to load a script resource (pragmatic).

Website features classes:

- Lexical: Term frequency-inverse document frequency (TF-IDF) features from webpages (syntactic), presence of brand names on the webpage (semantic), and attributes like processing suspicious or blacklisted content (pragmatic).
- Network: An example of a syntactic attribute is the content length of the webpage (all features extracted from HTTP protocol are considered network features). Semantic network features include download time based on the type of content (image, audio, video), while examples of pragmatic features could be the Autonomous System (AS) number.
- Script: Number of external scripts (syntactic), whether a script is malicious or has specific behavior such as creating popups (semantic), whether a script loads asynchronously or is deferred, (pragmatic).

C. EMAIL FEATURES

After a thorough study of the literature, we identified the recurring features used for phishing email detection.

We implement the features that have been used in at least four papers in the literature that report high performance or were highly ranked by feature ranking methods. These criteria resulted in a total of 143 email features in PhishBench. The breakdown of these features is as follows, the total number of features in each section has been added in the parentheses: Body (76), Header (68), URL (24). At the next level, the totals are: Semantic (30), Syntactic (79), and Pragmatic (34). One exemplar feature from each class is given below.

Header feature class:

The header of an email, while often hidden from the user, contains a lot of information including the route taken by the email, sender and recipient servers, timestamps, content format, etc. Previous researchers have extensively used header features for phishing detection. We demonstrate the importance of header features in Section VII, where we present classification results with and without these features.

- Lexical: Presence of message ID field (syntactic), features that detect lexical attributes such as detection of blacklisted words in the subject (semantic), and whether the email has plain-text content (pragmatic).
- Network: Features that extract network-related information, e.g. the comparison between the sender and return-path domains (syntactic), mismatch of the IP address between consecutive Received fields (semantic) and DKIM (DomainKeys Identified Mail) authentication results (pragmatic).

Body features class:

Email content can be in different formats (text, HTML, etc.) and can also include URLs. Hence, we separate the Body class into a Text subclass (includes text content and scripts from HTML content) and a URL subclass.

Text feature classes

- Lexical: Such features can be extracted using text analysis and Natural Language Processing, as well as from the HTML content of the email. These include the presence or count of tags such as ''href'' and ''onmouseover'' (syntactic), presence of difficult words in the text (semantic), presence of blacklisted words and action words, etc. (pragmatic).
- Network: These features can be extracted when the email body is an HTML document. We have given examples of such features under the webpage taxonomy.
- Script: Old email clients would allow scripts in email. However, with new clients, this class of features is relevant only when the email body is an HTML document. We have given examples of such features under the website category.

Email URL feature class: URL features need to be *aggregated* since emails do not have a fixed number of URLs, otherwise, the feature vectors will be of incompatible lengths. The ''aggregated'' feature extracts the targeted information from all the links in the email and returns a value which could be continuous or boolean. Examples of these features are the number of URLs that use a secure connection, number of URLs with different domains than the sender's address, number of URLs with an IP address, etc.

Email Attachments class: Emails have optional attachments. Attachment analysis can be a huge topic in itself and is beyond the scope of our taxonomy. From the email analysis point of view, features can be the number and type(s) of attachments, content disposition, and whether any of them are malicious (e.g., contain malware).

V. PhishBench

PhishBench is a benchmarking framework that can be used by researchers to systematically test and compare their features and classification results on common datasets. PhishBench provides feature extraction code for over 200 features and over 30 classification algorithms. Next, we present the architecture of PhishBench and describe its modules.

A. ARCHITECTURE OF PhishBench

PhishBench's architecture stems from machine learning based detection. It has five modules along with comprehensive logging of events and time-tracking, which are distributed throughout the modules, for debugging and comparison purposes.

Input Module: It handles dataset specifications, the dataset type (email, URL, etc.), location in the system, and evaluation mode (training or testing).

Feature Extraction and Ranking Module: It determines, extracts, ranks, and post-processes the features, e.g., using Min-Max scaler.[7]

Classification Module: It provides several options: batch, online, deep learning, and methods for imbalanced datasets.

Evaluation Module: It provides a set of 12 metrics (9 for classification, and 3 for clustering) for performance evaluation on balanced and imbalanced datasets.

PhishBench is modular and customizable. Based on the user's discretion, each of the existing modules can be kept or dropped, and it allows easy addition of new methods to the existing modules. The users only need to add the code for their new methods in corresponding modules (features in the Feature module, classifier in the Classification module, etc.). The users are offered an extensive list of parameters to select from and customize, including features to be extracted, feature selection and ranking methods, classifiers, and evaluation metrics.

PhishBench can be run at different stages of phishing detection experiments. A user/researcher can use it for feature extraction and ranking. If features are already extracted, then it is possible to run the classification and evaluation modules with desired classifiers and evaluation metrics respectively. PhishBench modules are discussed in depth below.

[7]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

B. PhishBench MODULES

The modules implemented in PhishBench are independent and, provided the correct input, can run separately based on the user's needs. Figure 3 illustrates the different modules and flow of the data between them.

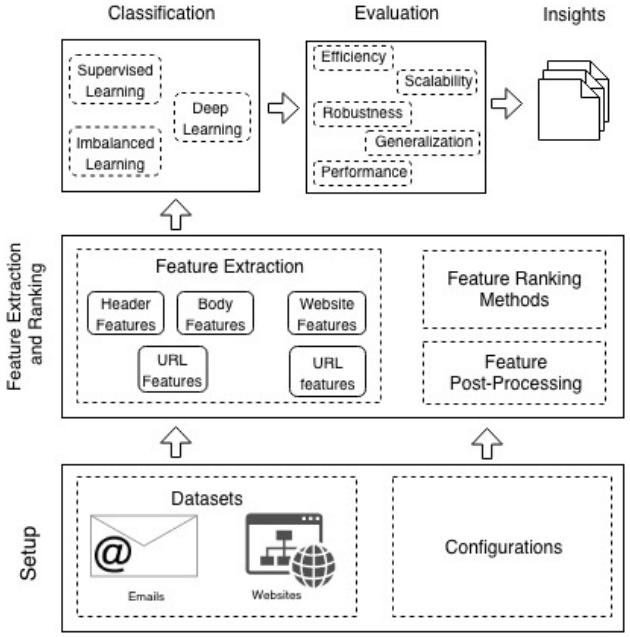


FIGURE 3. Schema of proposed benchmarking framework and system evaluation for phishing.

1) INPUT MODULE

This module handles loading the dataset into memory and has the necessary preprocessing functions implemented to extract features from different types of input. It accepts two types of inputs: a list of URLs for phishing websites/URLs detection, and a folder of email files for email based detection. For URL datasets, *if specified by the user*, the module parses the HTML content of the website linked to each URL and downloads all the network meta-data available such as WHOIS information and HTTP response time. For email datasets, the module parses the emails to extracts all the header fields and body contents. It handles content extraction and decoding for different body types including text, HTML and base64 encoded content. The parsed and decoded content from websites and emails is then used to extract features in the Feature Extraction module.

2) FEATURE EXTRACTION

This module contains the necessary functions to extract the features. The features implemented by default come from an extensive study of phishing literature as mentioned and described in Section IV. This module is customizable as it allows the selection of the features to be extracted, and extensible as new features can be added to the framework by implementing a function with required inputs and outputs. The framework will be published with an attached ReadMe file that explains the necessary steps to do so. The flexibility provided by this module helps researchers compare the results of their new features with already existing ones in the literature. It also reports the time needed for feature extraction.

3) FEATURE PROCESSING & RANKING

The main functions of this module are ranking and normalizing the features to use them as input for classification. We used the Scikit-Learn library to implement the ranking metrics/algorithms listed below.

- Information Gain (IG): selection based on the decrease in entropy after a dataset is split on an attribute [47].
- Gini Index (Gini): splits a feature into different classes to indicate its effectiveness for classification [55].
- Chi-Square Metric (Chi-2): measures the independence between the occurrence of a specific feature and a specific class [42].
- Recursive Feature Elimination (RFE): The model is first trained on all the features, and the importance score for each feature is computed. Then the lowest ranked features are removed after each iteration [27].

Normalization is done with the following methods: Max-Absolute scaler (scale each feature by its maximum absolute value), Min-Max scaler (subtracts the minimum value in the feature and then divides by the range), Mean scaling (subtracts the average value of the feature and then divides by the range), and L2 normalization [32].

This module outputs a file containing a sorted list of features based on the results of the ranking algorithm used , and a sparse matrix of the best features returned by the algorithm. The number of the best features is specified by the user.

4) CLASSIFIERS

This module implements the most used classifiers in the phishing detection literature including both supervised and unsupervised, weighted, balanced and imbalanced methods. We also integrate two Automated Machine Learning (AutoML) libraries: *(i)* AutoSklearn [23] and *(ii)* Tree-Based Pipeline Optimization Tool (TPOT) [46] into PhishBench. AutoML frameworks provide methods and tools for non-expert users [4]. Given a dataset of extracted features, systems like AutoSKLearn and TPOT automate the entire pipeline of selecting and evaluating a wide variety of machine learning algorithms and subsequently outputting the decision. While auto-sklearn uses meta-learning along with Bayesian optimization to search the best algorithms from Python's Scikit-learn library; TPOT uses genetic programming to select the best Scikit-learn pipeline [25].

For the supervised classification module, we implement the following learning algorithms:

- Support Vector Machines (SVM)
- Random Forest (RF)
- Decision Tree (DT)
- Gaussian & Multinomial Naive Bayes (GNB & MNB)

- Logistic Regression (LR)
- K Nearest Neighbors (kNN)
- Boosting (AdaBoost)
- Bagging
- Online Learning (e.g., AROW, OGD, ECCW, AdaFOBOS, etc.) [63]
- Deep Neural Networks (DL) [36]
- Imbalanced learning (e.g., Balanced Bagging Classifier, RUSBoost Classifier) [38]
- Hellinger Distance Decision Tree (HDDT) [13]

To handle imbalanced datasets, we also implement different under and over sampling methods including Repeated Edited Nearest Neighbor, ADASYN and SMOTE [38]. The user has the option to choose which classifiers to run, weighted or not, and with or without imbalanced methods. This module also reports the running time for each classifier. It is also extensible, as it is possible to add new classifiers by either implementing new methods or importing existing libraries. The user can choose which classifiers to run before each iteration.

5) EVALUATION METRICS

The selection of proper evaluation metrics is of utmost importance. Class imbalance in the evaluation dataset may result in base-rate fallacy wherein a metric like Accuracy may not be the best choice to evaluate classification performance. So we made sure to implement multiple metrics for accurate result evaluation. PhishBench reports all the metrics whose definitions are given in the Preliminaries Section.

VI. DATASETS

Diverse and extensive datasets are essential for a complete and thorough study of the features, otherwise, the feature importance may be biased or inaccurate. Since email is a popular attack vector for delivering phishing URLs or malware, our goal is to not only study URLs and webpages but also emails. Thus phishing emails and URLs are orthogonal dimensions of the problem. So, we use two separate dataset types: URLs and emails.

A. URL DATASET COLLECTION

Legitimate Dataset: We use Alexa top websites as a basis for the legitimate website. Instead of using Alexa directly, *since it just reports the top domains and removes the subdomains and URL path (if they exist)*, we use the top domains list as a seed for our crawler *to generate a more realistic dataset*.[8] We also limit the crawler to crawl up to three levels (it follows links within another website only to the depth of three) and only collect up to 10 URLs from each domain. Alexa categorizes domains based on 17 categories.[9] To increase the diversity of legitimate websites, we retrieved the top 40 websites from each category (on Sep 5, 2018). We excluded *Adult* category from the list. We also removed *Regional* and *World* since all the URLs in these two lists appeared in other categories.

Phishing Dataset: For the phishing websites, we use three different sources: PhishTank (Sep 5, 2018), APWG (Oct 30, 2018) and OpenPhish (Sep 5, 2018).

The feature extractor removes an instance if any of the URL, network and website features are not available (if the website is offline, it cannot retrieve the WHOIS info, etc.). After using all of these datasets and ranking the features, we realized that most of the top features are words from TFIDF features for "*Name of Languages*" (e.g., "latvieu", "kiswahili", "slenska", etc.). To discover the reason for this, we searched the phishing dataset for those words. We found that many phishing websites present in our dataset try to mimic the login page of different email providers, and these login pages have a drop-down list for language options. Since these words do not exist in the legitimate instances in our dataset, the classifiers can easily pick up on these features and consider all login pages as phishing websites. To solve this bias in our dataset, we run our crawler once more with the same seed. But instead of collecting all the websites, we just collect the pages with a login form. We use loginform 1.2.0[10] which is a Python library to detect the pages with login forms. This resulted in 4,370 legitimate websites with login forms. Table 3 shows some basic statistics of the different datasets that we collected. The third column (Extracted) shows the number of URLs whose features were successfully extracted. The next two columns are the number of unique domains and TLDs, and the last column shows the total number of login pages.

TABLE 3. Statistics of the URL/website dataset. Extracted - number of URLs whose features are extracted, Domains - number of unique domains, TLDs - number of unique TLDs, Logins - number of login pages.

Source	URLs	Extracted	Domains	TLDs	Logins
Alexa	31,163	29,173	9,554	285	2,056
Alexa Login	4,370	3,992	1,960	117	3,992
Legitimate	**35,533**	**33,165**	**10,405**	**294**	**6,048**
PhishTank	26,346	20,803	10,813	406	4,999
APWG	66,929	45,382	7,760	319	2,812
OpenPhish	2,249	1,336	710	94	326
Phishing	**95,524**	**67,521**	**18,126**	**481**	**3,637**

B. EMAIL DATASET COLLECTION

In collecting our email dataset, we made sure to increase the diversity by including phishing and legitimate emails from different sources.

Legitimate Dataset: We gathered 10,500 emails in total. We downloaded 6,779 emails from archives published by Wikileaks [62] (Hacking Team: 718, DNC: 3,098, GI files: 1,066, Sony: 1,120, National Socialist Movements: 678, Citizens Commission On Human Rights: 88, Plum emails: 11). We also downloaded 2,046 emails from randomly selected inbox folders of the public Enron dataset [35], and 1,675 ham emails from SpamAssassin [53]. We note that

[8]Many previous works that use Alexa seem to have missed this issue.
[9]https://www.alexa.com/topsites/category
[10]https://pypi.python.org/pypi/loginform

Enron's email headers tend to be shorter and have been sanitized to protect name/email information, and SpamAssassin email headers have been "lightly sanitized."

Phishing dataset: We collected 10,500 emails in total. We downloaded 8,433 emails from the Nazario phishing email dataset [43]. We also downloaded 1,048 emails from its recently published 2015 to 2017 emails. We also added the 1,019 emails from SpamAssassin. Note that the SpamAssassin dataset is made of old phishing emails, which are close to spam. It was used several times in literature, so we increase the diversity of our dataset by adding samples from it. We also draw attention to the fact that it is harder to find phishing email datasets. However, we conduct a diversity analysis to make sure that the dataset is diversified.

C. DATASET DIVERSITY

Using a diverse dataset is crucial for having a generalizable trained model. We now evaluate the diversity of our dataset from different aspects, e.g., URL domains, Top Level Domain (TLD), email content, etc.

URL: Although there is no widely-accepted method for analyzing the diversity of a dataset, the following questions can help in this regard: How many different domains exist in the dataset? How many different TLDs are there? If many URLs in the dataset are from the same domain, that means there is a bias towards some specific websites. For the TLDs, the situation is a little different since in the real world a uniform distribution among different TLDs is unlikely. Some TLDs are used more often, e.g., ".com" and ".org" and some are rarely used,[11] e.g., ".gw" and ".ax." So, we do not expect our dataset to have a uniform distribution of TLDs.

To illustrate the distribution of the domains and TLDs, we calculated their frequency in our phishing and legitimate datasets. Then, we counted the number of URLs whose domain (TLD) is among the 50 most frequent domains (TLDs). Figure 4a shows the percentage of URLs which are from the top 50 domains. For the legitimate datasets, the percentage is almost zero since we limited the number of URLs per domain to 10. Among the phishing ones, Openphish and APWG are almost similar but the Phishtank dataset is much more diverse. Figure 4b compares the percentage of URLs which are from the top 50 TLDs. It shows a huge gap between the phishing and legitimate datasets which shows phishers mainly use certain TLDs.

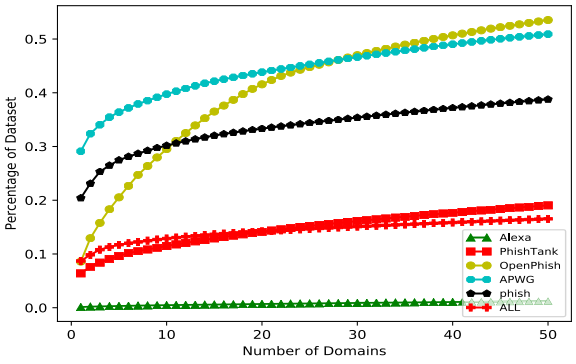
Emails: To analyze the diversity of our email dataset, we compare their contents to see how much similarity exists between them. We extract the text content of each email (including the header) and remove all the HTML tags and CSS. We also filter out the stop words. Then we extract the TFIDF vector for each email.

We use the cosine similarity measure, which is the cosine of the angle between two vectors (Equation (9)), to measure the pairwise similarity between the TFIDF vectors extracted from the dataset.

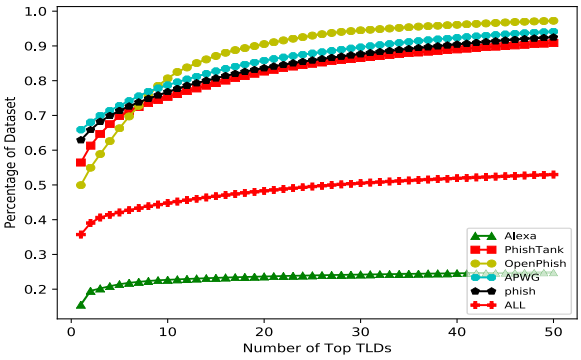
$$similarity(A, B) = cos\,\theta = \frac{A.B}{||A||\,||B||} \tag{9}$$

We show in Table 4 the ranges of pairwise similarities in both datasets (with/out header). Each column of the table shows the percentage of the email pairs that fall within that range of similarity. The table shows that about 85% of pairs have 0-10% similarity which is a sign of diversity among the emails.

[11] http://www.seobythesea.com/2006/01/googles-most-popular-and-least-popular-top-level-domains/



(a) CDF of top 50 Domains in each dataset separately and all datasets combined



(b) CDF of top 50 TLDs in each dataset separately and all datasets combined

FIGURE 4. Distributions of domains and TLDs in different dataset sources. Phish: combined PhishTank, OpenPhish and APWG.

TABLE 4. Percentage of email pairs with different ranges of similarities in the dataset with/out header.

Dataset	Ranges of Similarities					
	[0-10]	(10-20]	(20-30]	(30-40]	(40-50]	>50
WH[a]	85.44%	10.47%	2.60%	0.85%	0.29%	0.33%
NH[b]	84.29%	10.74%	3.92%	0.55%	0.18%	0.29%

[a] With Header
[b] No Header

VII. EXPERIMENTS AND RESULTS

We systematically evaluate the phishing URL and email detection models with the implemented features. Classifiers are trained and then tested on balanced datasets (equal number of phishing and legitimate instances). The experiment

is repeated on imbalanced datasets with different ratios of phishing to legitimate instances. We also run additional experiments on the features we extracted, such as feature ranking, tracking the feature extraction times, and testing robustness of features on a more recent dataset. These experiments will help us determine the best subset and categories of features in terms of performance and time efficiency for our problem.

Before running the classifiers on different datasets, we perform hyperparameter tuning using Randomized Parameter Optimization [9] to optimize the classifiers. We randomly split the dataset into training and testing sets (90:10) and ran the randomized cross-validation (10 fold) search on the training set with the maximum number of 100 iterations. For the SVM, we only used a linear kernel since other kernels did not converge after two days of running. We also use two existing AutoML softwares, TPOT and AutoSklearn. Tables 5 and 6 show the best parameters for every classifier used for phishing website and email detection respectively.

TABLE 5. Best parameters obtained by running randomized parameter optimization on website features.

Classifiers	Best Parameters
RF	n_estimators: 80, max_depth: 90, min_samples_split: 10, min_samples_leaf: 1 max_features: auto, bootstrap: False,
DT	max_depth: 70, min_samples_split: 2, min_samples_leaf: 1
GNB	var_smoothing: 1e-06
MNB	Alpha: 0.1
LR	solver: sag, C: 4
KNN	K: 5
Bagg.	n_estimators: 90, max_features: 1.0, bootstrap: False, bootstrap_features: True, base: DT
Boost.	n_estimators: 100, learning_rate: 1.5, algorithm: SAMME, base: DT
SVM	penalty: L1, loss: hinge, dual: true, C: 100, multi_class: crammer_singer
Ada-Fobos	eta: 0.25, delta: 0.5
DL	dev_pct: 0.2, delimit_mode: 0, min_word_freq: 1, emb_mode: 5, emb_dim: 128, filter_sizes: 3,4,5,6, nb_epochs: 5, batch_size: 50
TPOT	classifier: LogisticRegression, C: 15.0, penalty: l2, dual:False

TABLE 6. Best parameters obtained by running randomized parameter optimization on email features.

Classifiers	Best Parameters
RF	n_estimators: 90, max_depth: None, min_samples_split: 2, min_samples_leaf: 1 max_features: auto, bootstrap: False,
DT	max_depth: 110, min_samples_split: 3, min_samples_leaf: 1, max_features: None
GNB	var_smoothing: 1e-09
MNB	Alpha: 0.1
LR	solver: newton-cg, C: 1.0
KNN	K: 3
Bagg.	n_estimators: 90, max_features: 1.0, bootstrap: True, bootstrap_features: False, base: DT
Boost.	n_estimators: 80, learning_rate: 1.0, algorithm: SAMME.R, base: DT
SVM	penalty: L2, loss: squared_hinge, dual: true, C: 1.0, multi_class: ovr
AROW	r: 32.0
DL	dev_pct: 0.2, delimit_mode: 0, min_word_freq: 1, emb_mode: 3, emb_dim: 25, filter_sizes: 3,4,5,6, nb_epochs: 5, batch_size: 50
TPOT	classifier: Logistic Regression, C: 0.5, penalty: l2, dual: True

We added to PhishBench the Library for Scalable Online Learning (LIBSOL),[12] which includes 17 different online learning algorithms. Instead of reporting all these voluminous results, we only report the best performing method. It is chosen by computing the average ranking of each method in all experiments (separately for emails and websites). AutoSklearn automatically creates a weighted ensemble of several supervised learners, so we only report the results of the model. We specify the specifications of the machines used in the following section.

[12]https://github.com/LIBOL/SOL

A. SYSTEM CONFIGURATION

We used two different machines for running the experiments in parallel. One for all URL/website experiments and a second one for email experiments. For the website and URL, we used a machine with 128GB RAM and Intel Xeon(R) W-2123 (3.60GHz) processor running Ubuntu 18.04 (with Nvidia Quadro P1000 for deep learning experiment). We ran the email experiments on a machine with 512GB RAM and Intel(R) Xeon(R) E5-2667 v4 (3.20GHz) processor running Linux Red Hat Enterprise Server 7.6 (wit Nvidia Tesla M10 for deep learning experiment).

B. BALANCED VS IMBALANCED EXPERIMENTS

We use all 226 features (83 URL/website and 143 emails) implemented in PhishBench in our experiments to evaluate the performance of the models and features in different scenarios.

1) PHISHING URL DETECTION

We now evaluate classification algorithms on various combinations of the URL datasets mentioned in Table 3.

Balanced Datasets: As mentioned earlier, the PhishTank dataset is more diverse compared to Openphish and APWG, so here we compare the performance of classifiers on each of these sources separately to analyze its effect on classifier performance. We tested on three different combinations that were combined in a way to keep the legitimate and phishing ratio equal: 1) Openphish and Alexa Login 2) PhishTank and Alexa 3) Alexa, PhishTank, OpenPhish and a random subset of 10k URLs from APWG to ensure that the dataset is balanced. In the rest of the paper, we refer to dataset three as "website dataset." Tables 7, 8, and 9 present the classification results on each of these datasets.

TABLE 7. Performance of classifiers on Openphish and Alexa Login. DL: Deep learning.

Classifier	Time(s)	F_1-score	Acc.	G-mean	MCC	BDR
RF	0.72	**99.23**	**99.62**	**99.49**	**98.98**	**200.5**
DT	0.24	95.78	97.93	97.32	94.41	66.0
GNB	0.04	50.78	52.63	61.08	35.63	150.0
MNB	**0.01**	92.36	96.05	96.60	89.95	96.0
LR	1.9	97.33	98.68	98.60	96.47	132.3
5NN	0.13	93.63	96.80	96.58	91.55	65.0
Bagg.	6.4	98.44	99.24	98.71	97.95	100.2
Boost.	0.34	95.09	97.55	97.34	93.49	78.6
SVM	0.4	98.46	99.24	98.98	97.96	133.3
DL	2077	94.48	97.38	95.59	92.80	36.1
Ada-Fobos	0.01	94.16	96.99	97.73	92.36	193.5
HDDT	1.37	23.30	61.65	41.97	-2.20	2.97
AutoSk	3595.8	98.07	99.06	98.07	97.45	42.33
TPOT	1818.6	98.83	99.44	98.84	98.47	127.0

TABLE 8. Performance of classifiers on PhishTank and Alexa. DL: Deep learning.

Classifier	Time(s)	F_1-score	Acc.	G-mean	MCC	BDR
RF	30	96.22	96.87	96.95	93.58	52.8
DT	17.8	94.38	95.37	95.32	90.46	27.6
GNB	1.56	64.16	78.01	68.92	57.01	2.7
MNB	**0.32**	90.45	91.59	92.36	83.69	47.9
LR	60.56	95.98	96.67	96.73	93.16	45.9
5NN	13.89	94.64	95.53	95.67	90.86	37.9
Bagg.	216	97.07	97.59	97.59	95.04	57.6
Boost.	1508	**97.12**	**97.61**	**97.72**	**95.14**	**82.0**
SVM	1.7	96.07	96.75	96.79	93.32	46.0
DL	18364	95.45	96.29	96.23	92.33	34.1
Ada-Fobos	0.09	95.8	96.51	96.62	92.85	48.9
HDDT	86.16	41.31	51.62	49.36	0.16	1.44
AutoSk	5351.26	95.76	96.46	95.78	92.78	14.92
TPOT	1204.71	94.94	95.79	94.95	91.38	14.37

TABLE 9. Performance of classifiers on Alexa whole, PhishTank, Openphish and APWG10K. DL: Deep learning.

Classifiers	Time(s)	F1	Acc.	G-mean	MCC	BDR
RF	44.0	97.04	97.22	97.30	94.46	68.6
DT	22.0	95.27	95.59	95.63	91.15	28.7
GNB	2.1	59.88	72.99	65.45	51.21	2.0
MNB	**0.4**	92.26	92.41	92.65	85.41	47.4
LR	74.0	96.28	96.53	96.56	93.04	36.6
KNN	24.9	95.22	95.47	95.60	91.03	44.2
Bagg.	298.9	**97.61**	**97.77**	**97.81**	**95.53**	**70.8**
Boost.	2290.0	97.59	97.75	97.80	95.50	70.8
SVM	69.9	96.64	96.87	96.90	93.72	40.9
DL	25080.0	95.78	96.05	96.10	92.00	35.1
Ada-Fobos	0.1	96.15	96.39	96.47	92.79	45.6
HDDT	136.17	46.37	50.28	49.89	0.03	1.16
AutoSk	5441.8	97.47	97.63	99.16	95.26	25.64
TPOT	483.4	86.28	85.81	75.30	85.32	27.56

There is a gap between the performance of classifiers on the Openphish and Alexa versus Phishtank and Alexa. On the Openphish and Alexa (Table 7), three classifiers reached more than 98% F_1-score but on the Phishtank and Alexa (Table 8), the best F_1-score is 97.12%. The differences in BDR metric values provided give an even better perspective of classifiers' performance in Tables 7 and 8. While seven classifiers show a BDR greater than 100 in Table 7, the highest BDR of 82 was recorded by Boosting in Table 8. This agrees with our discussion on the diversity of PhishTank and Openphish.

For all datasets combined (Table 9), Bagging, Boosting and the model chosen by AutoSklearn are the top performing classifiers. Comparing their training times, Bagging is the fastest and AutoSklearn is slowest. We describe in more detail our observations regarding model training time in Section VII-C.

Imbalanced Dataset: In the real world, the chance of a URL being phishing is much lower than being legitimate and this affects the performance of classifiers. We evaluate our models on imbalanced datasets by varying the legitimate to phishing ratio to have a more realistic evaluation of the models' performance. We fix the total number of legitimate and phishing URLs to 36,457 by randomly sub-sampling from the balanced dataset (the allowable maximum size for the 1:10 experiment) and change the legitimate to phishing ratio from 1:1 to 1:10. Figure 5 demonstrates classifiers' performance using the F_1-score on the different dataset ratios. We report the F_1-score instead of accuracy since it is a metric appropriate for classification tasks on imbalanced datasets [6]. Other proper metrics for imbalanced datasets (G-mean, MCC, BDR) revealed a similar trend, hence they were omitted. We were not able to run HDDT on the feature set including TFIDF features since our machines ran out of memory.

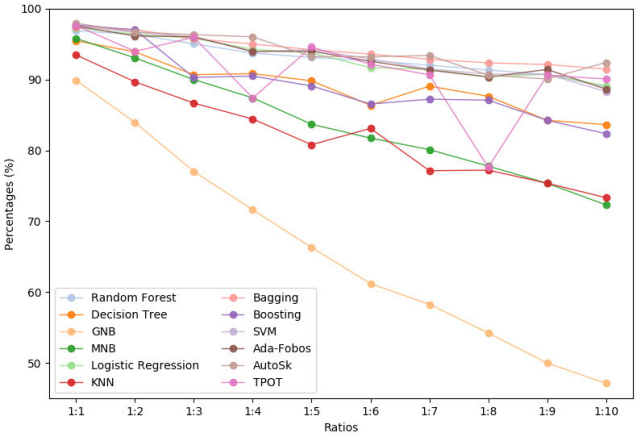


FIGURE 5. Changes in F_1-score Performance with varying ratios between legitimate and phishing URLs.

A common observation across all classifiers is the downtrend in the F_1-score with the class imbalance ratio. GNB has the most decline in performance (from 89% to 47%) while AutoSklearn and Bagging have the least decline, 5.5%, and 5.9% respectively. Boosting, which had a similar performance to Bagging in 1:1 ratio, has a three times bigger decline than Bagging (15.3%), which makes it a bad choice for the real-world scenario. We also tested various re-sampling techniques (e.g., ADASYN, AllKNN, SMOTE, NearMiss, etc.) but, unfortunately, none of them notably improved the results.

2) PHISHING EMAIL DETECTION

Next, we perform the above experiments on email datasets.

Balanced Datasets: Tables 10 and 11 show the performance of classifiers on different sets of features. Depending on whether header features were extracted or not, we refer to these tables as With Header and No Header respectively. The model created by Auto-Sklearn performed the best in both With Header and No Header experiments. The highest F_1-score for With header emails was achieved by LR, SVM, and Auto-Sklearn (99.95%). For the case of No header features, Auto-Sklearn and RF achieved the highest F_1-score: 99.09%. The difference in performance for the two cases can be explained by the fact that email headers contain important information, which can be used to discriminate between legitimate and phishing samples. The fastest classifier in both cases was KNN, and the slowest were Bagging and Auto-Sklearn.

TABLE 10. Email dataset results: With header.

Classifiers	Time (s)	F_1-score	Acc.	G-mean	MCC	BDR
RF	8.64	99.90	99.90	99.90	99.80	349.0
DT	7.21	99.76	99.76	99.76	99.52	521.5
GNB	12.49	97.91	97.95	97.94	95.97	33.64
MNB	0.05	98.36	98.38	98.37	96.79	24.88
LR	4.63	**99.95**	**99.95**	**99.95**	**99.90**	**1046.0**
KNN	**0.01**	97.40	97.33	97.29	94.76	199.0
Bagg.	421.00	99.85	99.85	99.85	99.71	348.6
Boost.	7.48	99.85	99.85	99.85	99.71	522.5
SVM	0.28	**99.95**	**99.95**	**99.95**	**99.90**	**1046.0**
DL	128.16	99.85	99.85	98.56	**99.90**	1044.0
AROW	0.43	98.18	98.19	98.19	96.00	43.0
HDDT	8.16	49.11	49.38	49.38	-1.2	0.9
AutoSk	3595.96	**99.95**	**99.95**	**99.95**	**99.90**	**1046.0**
TPOT	3156.04	**99.95**	**99.95**	**99.95**	**99.90**	**1046.0**

TABLE 11. Email Dataset Results: No header.

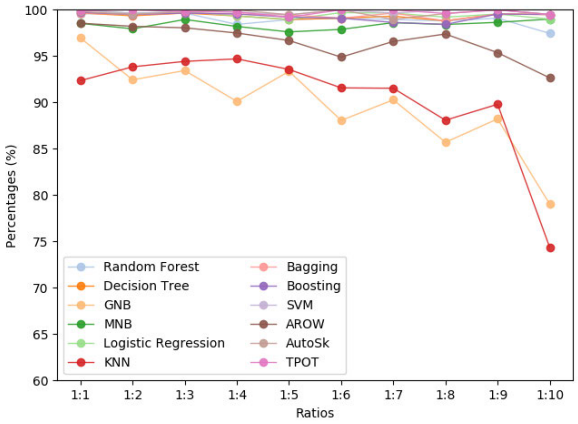
Classifiers	Time (s)	F_1-score	Acc.	G-mean	MCC	BDR
RF	13.828	98.61	98.61	98.61	97.25	43.37
DT	15.650	97.24	97.23	97.23	94.47	35.06
GNB	9.690	94.36	94.52	94.48	89.22	17.35
MNB	0.022	96.08	96.14	96.13	92.33	11.23
LR	3.422	98.52	98.52	98.52	97.04	57.38
KNN	**0.008**	89.14	88.14	87.66	77.51	25.90
Bagg.	833.797	98.05	98.04	98.04	96.09	48.85
Boost.	325.433	98.80	98.80	98.80	97.62	57.72
SVM	0.583	98.72	98.71	98.71	97.42	85.91
DL	126.207	97.89	97.86	97.84	95.74	63.43
AROW	0.254	96.61	96.66	96.65	93.38	19.07
HDDT	7.38	49.11	49.09	49.09	-1.88	0.95
AutoSk	3595.69	**99.09**	**99.09**	**99.09**	**98.19**	**115.11**
TPOT	629.66	94.57	94.52	94.51	89.05	18.90

Imbalanced Dataset: We repeat the classification experiment but with different phishing to legitimate ratios to simulate real-world scenarios. The setup was similar to the URL experiment, we used random sub-sampling to reduce the size of the dataset to 11,550 emails (which is the maximum size that we could use to obtain the 1:10 ratio) and we test on a range of different ratios from 1:1 to 1:10. Figure 6 shows the results of this experiment. Same as the URL experiment, we see a decreasing trend in F_1-score as we increase the ratio of legitimate to phishing emails. The classifiers with the steepest decline are KNN (18% in 6a and 23.8% in 6b). We observe a higher drop in performance for most of the classifiers when tested on emails without header information in comparison to when header information is available. This suggests the importance of the header in phishing email detection. The classifiers with the highest F_1-scores at 1:10 ratio are Boosting and DT (99.507%) in 6a with AutoSk and TPOT as close seconds (99.502%), and RF (91.66%) in 6b.

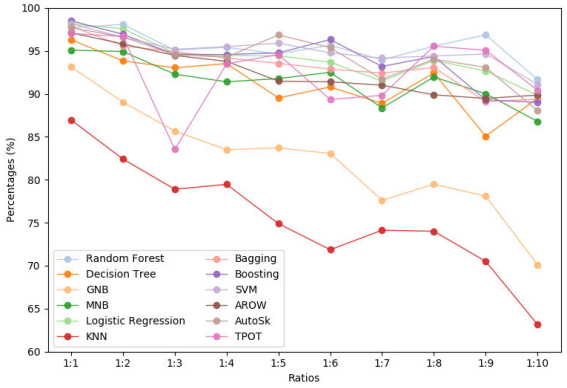
3) KEY TAKEAWAYS

We learned from these experiments on Websites and emails the following:

- For the URL experiments, we observed a relationship between the diversity of the dataset and classifier performance. Classifiers performed worse on the PhishTank dataset compared to OpenPhish which was less diverse in terms of the number of domains.



(a) With Header Features



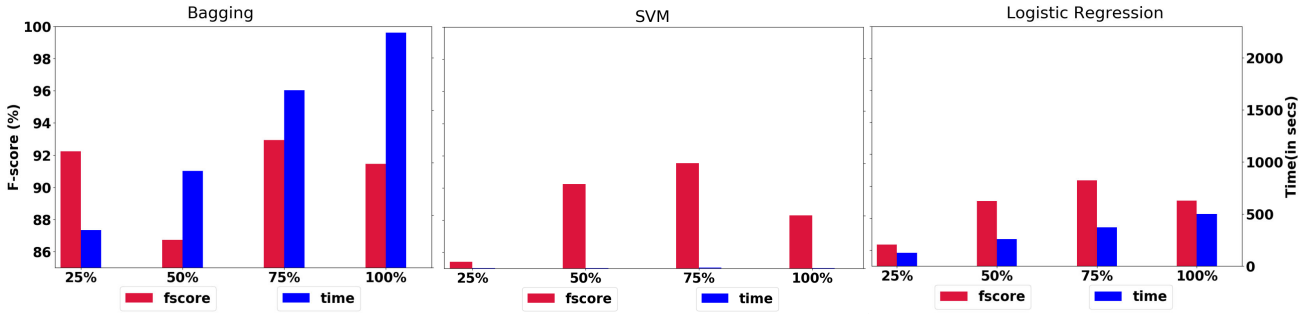
(b) No Header Features

FIGURE 6. Changes in F_1-score with varying ratios and varying ratios between legitimate and phishing emails.

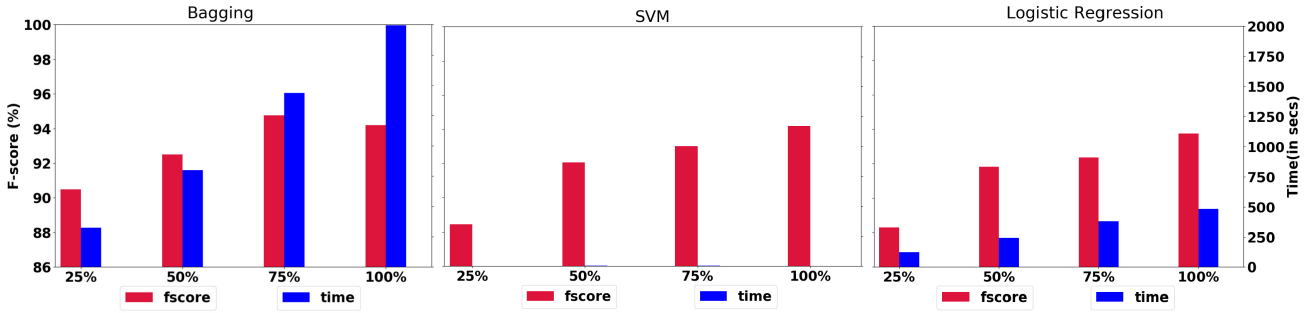
- Evaluating classifiers on a balanced dataset for an imbalanced problem can result in base-rate fallacy. The least decline in F1_Score observed – RF for emails (7%) and AutoSklearn and Bagging for websites (6%).
- Classifier performance decreases as the dataset becomes more imbalanced.
- Not a single classifier could be singled out as the best performing model *even for the same problem* in all different scenarios. Security professionals and researchers need to test and select the best suited classifier for their dataset and features. Thus we can say that there is no one size fits all solution.
- The existing methods to tackle imbalanced datasets such as over-sampling and under-sampling did not create any difference in classifier performance on both website and email experiments.

C. SCALING EXPERIMENTS

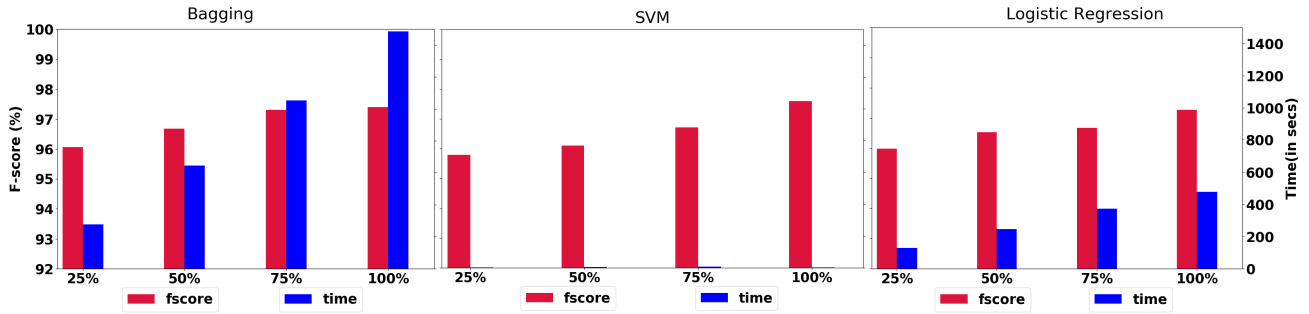
The size of the training dataset also affects the performance of classifiers. On one hand, having more phishing and legitimate samples can help classifiers build a more general model. On the other hand, it can make the training more time-consuming and complex by mixing the phishing and legitimate instances in the feature space, hence leading to a weaker



(a) F_1-scores and Time comparison for ratio 1 to 10



(b) F_1-scores and Time comparison for ratio 1 to 5



(c) F_1-scores and Time comparison for ratio 1 to 1

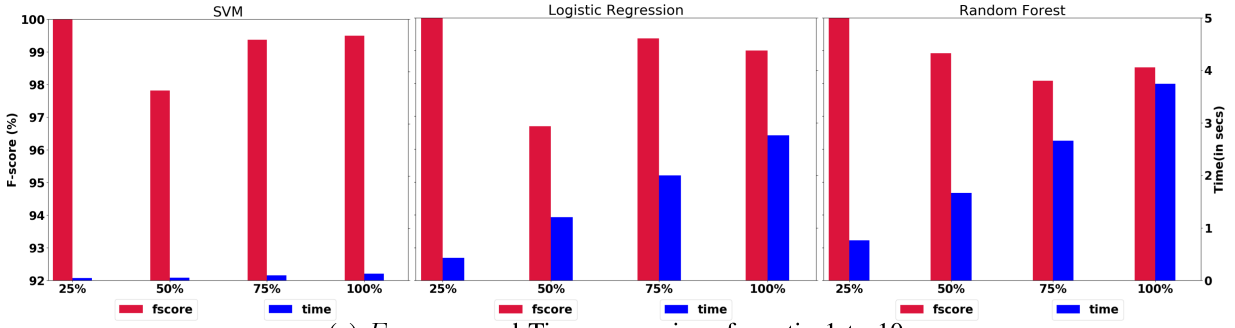
FIGURE 7. Scaling performance of top classifiers on websites on three ratios 1:1, 1:5, and 1:10.

model. In this experiment, we train and test the classifiers on different sizes of the same dataset. We fix the ratio of legitimate to phishing samples, and gradually decrease the number of samples from 100% to 25%, with a step of 25%. The smaller datasets are created by randomly removing some instances from the larger ones. So, the larger dataset always contains all the samples from the smaller dataset.

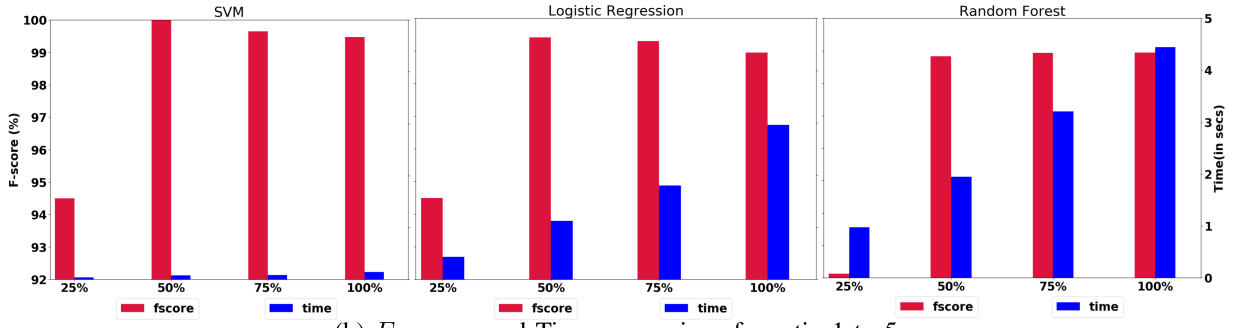
Through this experiment, we analyze how classifiers' performance and training time change depending on the size of the dataset. We reduce the size of both website and email dataset to 75%, 50%, and 25% of the original size. We repeat this process for three different ratios, 1:1, 1:5 and 1:10, to test how the effect of having imbalanced samples manifests with different dataset sizes. Then we train/test all our classifiers on the reduced datasets. Figures 7, 8, and 9 show the performance of the top three classifiers on the reduced datasets as well as on the original size dataset. To make the comparison easier, we chose the top three classifiers from the balanced dataset and kept them constant for the ratios 1:5 and 1:10.

In the website experiment, we see an upward trend in the F_1-score as we increase the size of the dataset for the 1:1 and 1:5 ratios. However, the classifiers are less stable for the 1:10 ratio. In the email experiment, we notice a general increase in performance relative to the size of the dataset for 1:1 ratio. That increase, however, is less stable in the 1:5 ratio case. For 1:10 ratio, no specific pattern can be discerned for the performance of the models. We can conclude from this experiment that with highly imbalanced datasets, the performance is hard to predict based on the size of the dataset. This can be caused by the variation of the difference between the number of legitimate and phishing samples. Even with a fixed ratio, an increase in size leads to a bigger difference. The training time, as expected, increases in all the cases with the size of the dataset.

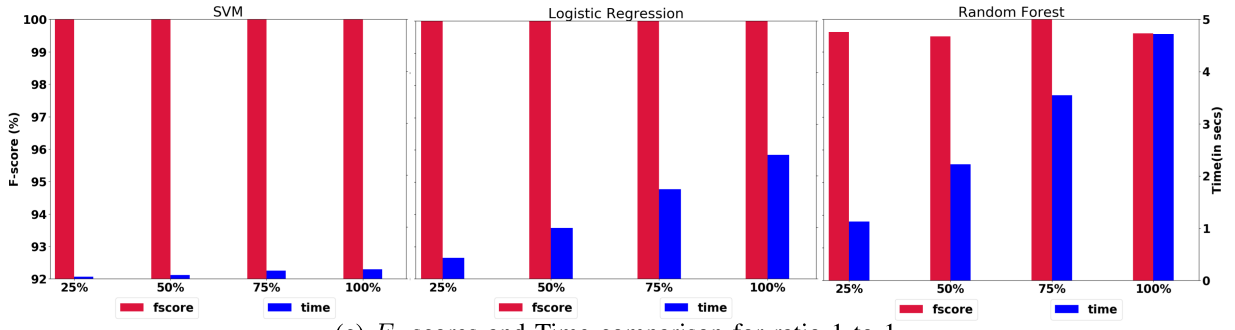
We did a regression analysis on the top models to test the effect of dataset size and ratio on the classifier performance (Table 12). The regression results showed that the dataset ratio has an inverse relationship with the F_1-score



(a) F_1-scores and Time comparison for ratio 1 to 10



(b) F_1-scores and Time comparison for ratio 1 to 5



(c) F_1-scores and Time comparison for ratio 1 to 1

FIGURE 8. Scaling performance of top classifiers on emails with header on three ratios 1:1, 1:5, and 1:10.

TABLE 12. Regression results for classifiers' F_1-score (standard deviations).

Variable	Website	Email WH	Email NH
(Intercept)	95.24†(1.54)	98.41†(1.32)	93.87†(3.24)
Ratio	-0.66†(0.14)	-0.08 (0.12)	-0.67*(0.30)
Size	0.02 (0.01)	0.01 (0.01)	0.04 (0.04)
Adjusted R^2	0.65	-0.02	0.27

*, † indicates significance at the 90% and 99% level respectively.

(performance drops as the dataset gets more imbalanced) and it is significant for websites and emails the without header. Interestingly, the ratio can explain by almost two-thirds (Adjusted R^2 of 0.65) the variance in performance for website detection. The dataset size is not a significant predictor of classifier performance. On emails with headers, the classifier performance is immune to changes in the aforementioned attributes which requires a deeper study.

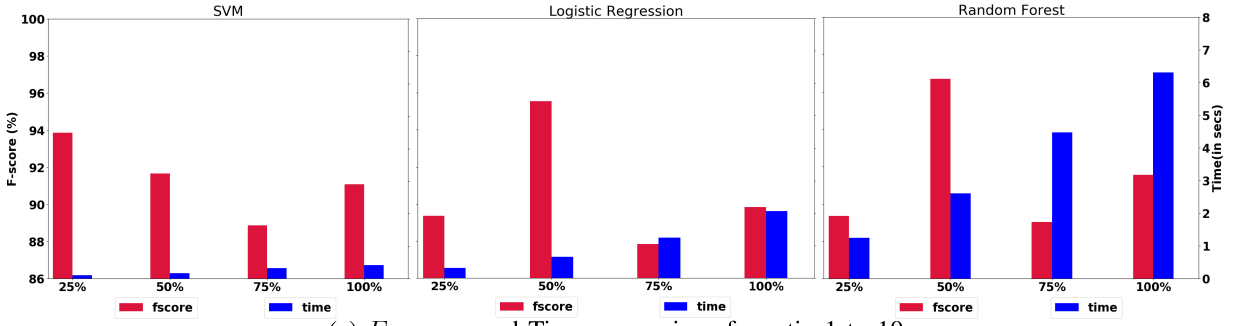
1) KEY TAKEAWAYS

We learned the following from our scaling experiment:

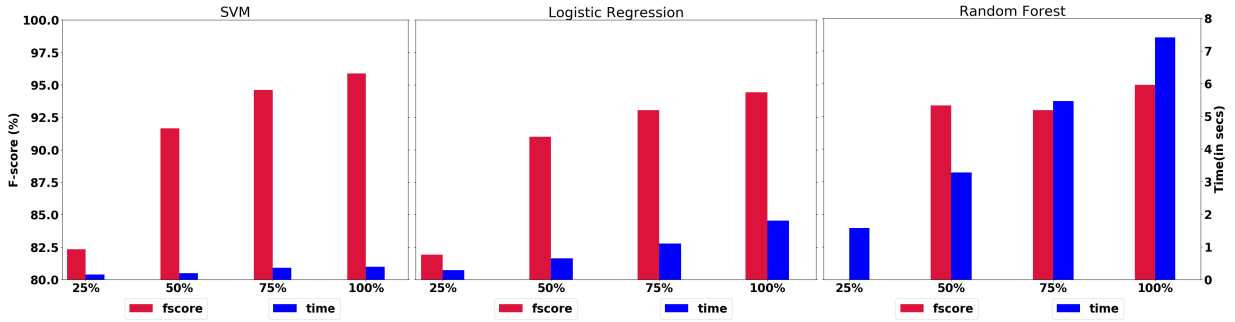
- As shown with regression analysis, dataset imbalanceness has an opposite effect on classifier performance (Table 12) and it can be a significant factor in classifier performance on some datasets. F_1-score declines as the dataset gets more imbalanced.
- When the dataset is balanced, there is a general increase in performance with larger datasets for both URLs and emails. But the relationship between performance and larger dataset size becomes unclear with the effect of class imbalance. The regression analysis did not find any significant relationship between dataset size and classifiers' F_1-score.

D. FEATURE IMPORTANCE

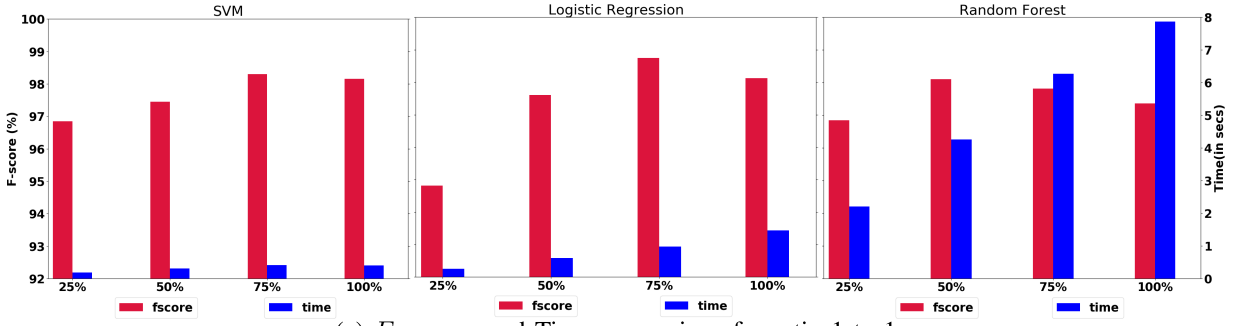
We now analyze the importance of each feature via four feature ranking methods. Then, we compare the performance



(a) F_1-scores and Time comparison for ratio 1 to 10



(b) F_1-scores and Time comparison for ratio 1 to 5



(c) F_1-scores and Time comparison for ratio 1 to 1

FIGURE 9. **Scaling performance of top classifiers on emails with no header on three ratios 1:1, 1:5, and 1:10.**

of the model trained only on the top features with the model trained on all the features. In the end, we report the extraction time of the features that affect models' response time regardless of their speed in making the inference.

1) FEATURE RANKING

It is important to ascertain the main features that contribute to the detection process. We use multiple feature ranking techniques in the feature-ranking module of PhishBench: Information Gain (IG), Gini Index (Gini), Chi-Square Metric (Chi-2), and Recursive Feature Elimination (RFE). Here, we report the top 20 features ranked by these four methods.

Website/URL: Table 13 shows the top 20 features extracted from the phishing and legitimate websites. In the list of top features, we see features from different categories described in Section IV (except RFE, which mostly has TFIDF features), which shows each of those aspects contributes to better performance. Features like the number of anchor tags, number of suspicious script functions, fall under the HTML-based semantic category for website content. The semantic URL feature which appears among the top 20 is the character distance which belongs to the lexical-based subcategory. The common syntactic URL features (TLDs like edu, org, etc. and *has https*) ranked among the top 20 belong to the lexical sub-category. The top pragmatic features belong to the sub-category of network features like *AS Number*, *DNS total time to live*, expiration date. Among the top features, attributes from the syntactic category appear the most when compared to the semantic and pragmatic categories.

Email: For emails, we conducted two sets of experiments, with and without header features. Tables 14 and 15 report the top 20 features, including TFIDF, with and without header features respectively. We notice that in Table 14, the top 20 features are mostly TFIDF features. This makes sense since email headers belonging to one source (same company or institution) tend to have more similarities. The feature ''zzzzason'' was ranked as the fourth important feature by

TABLE 13. Top 20 extracted features from legitimate and phishing URLs and websites. Features between the quotations are TFIDF features. url_char_distance_char - Difference between the distribution of the character 'char' in English text and their distribution in URL.

Rank	IG	Gini	Chi-2	RFE
1	url_char_distance_w	inbound_href_count	as_number=16509	"posta"
2	inbound_href_count	url_char_distance_w	more than 3 dots	"phisher"
3	dns_ttl	dns_ttl	url_char_distance_w	"organisms"
4	creation_date	creation_date	as_number=15169	"nomination"
5	# of anchor	has_https	URL is redirect	"grassroots"
6	# of dashes	Website_content_type	as_number=46606	"divisions"
7	has_https	expiration_date	# of "search"	"curse"
8	"provider"	outbound_href_count	# of suspicious JS func.	"criticism"
9	Website_content_type	# of slashes	# of iframes in script	"bylaws"
10	# of slashes	# of dashes	Brand in URL	"attendee"
11	expiration_date	# of name server	url_char_distance_c	"ambitious"
12	TLD=org	url_length	# of exec	"accelerated"
13	outbound_count	# of tags	url_char_distance_k	"abstract"
14	# of tags	domain_length	# of iframes	# of titles
15	outbound_href_count	outbound_count	"hotmail"	# of slashes
16	Average domain token length	TLD=edu	"gmail"	# of scripts
17	url_char_distance_r	TLD=org	url_char_distance_n	# of iframes
18	url_length	URL_Is_Redirect	"aol"	# of digits
19	url_char_distance_s	digit_letter_ratio	url_char_distance_d	# of punctuation
20	# of digits	content_length	# of name server	inbound_href _count

TABLE 14. Top 20 extracted features from Emails with header features. Features between the quotations are TFIDF features.

Rank	IG	Gini	Chi-2	RFE
1	"keywords"	"keywords"	return_path	"keywords"
2	"fetchmail"	"fetchmail"	binary_re	"iamunique"
3	Received_count	Received_count	Added_Suffix_ Prefix_Binary	"autolearn"
4	"delivered"	"zzzzason"	"login"	"2014"
5	return_path	"netnoteinc"	"keywords"	"webnote"
6	"authas"	"slashnull"	X_virus_scanned	"login"
7	"oct"	"oct"	Binary_table_tag	"qqqqqqqqqq"
8	"iamunique"	"eire"	"postfix"	return_path
9	"alert"	"delivered"	X_mailer	"eire"
10	"spamtraps"	"pop3"	Plain_Text	"wrote"
11	"account"_ count_in_body	"alert"	"uid"	"000"
12	"date"	"cdo"	Proportion_Words _No_Vowels_Body	"group"
13	"values"	"internal"	Binary_Img_Links	"3d"
14	"grp"	"0102"	"esmtp"	"javamail"
15	"linux"	"3d"	"delivered"	"hackingteam"
16	"afree"	"attained"	X-Priority	"libpst"
17	number_of_unique _words_body	"organization"	IP_Address_binary	"organization"
18	"path"	return_path	From_To_ Strings_in_Body	"sony"
19	"postfix"	"efi"	double_slashes_ average_count	Received_count
20	"abe"	"assures"	"monkey"	"your account" _count_in_body

the Gini Index, primarily because it was used as a normalization method to hide usernames in the SpamAssassin dataset. However, this could lead to a lack of generalization if the classifiers are trained on one dataset and tested on another. This issue was not pointed out in the literature since researchers often use the same dataset for training and testing [28], [29], [65]. We also notice the presence of non-English words like "que" or "eire" in Tables 14 and 15, due to communications between foreign individuals in some of the datasets that we used (e.g., Hacking Team and Sony datasets). Unlike website features, we observe that most of the higher ranked email features belong to one category (lexical). This means that lexical features from email body and header are more important and the other subcategories from our email taxonomy are largely under-utilized.

Description of features: Here we give a brief description of recurrent features in the tables that need clarification. Features placed under quotes are tokens extracted with TFIDF.

TABLE 15. Top 20 extracted features from Emails with no header features. Features between the quotations are TFIDF features.

Rank	IG	Gini	Chi-2	RFE
1	"account"	"account"	Added_Suffix_ Prefix_Binary	"jpg"
2	DNS_Info_ Exists_Binary	DNS_Info_ Exists_Binary	Binary_table_tag	"wrote"
3	number_of_html_ comment_tags	Binary_table_tag	Binary_Img_Links	"itinerary"
4	"pm"	"click"	Proportion_Words_ No_Vowels_Body	"webmail"
5	Body_Size	number_of_ words_body	IP_Address_binary	"wir"
6	"click"	outbound_href_ count_average	From_To_ Strings_in_Body	gunning_fog
7	"enron"	number_of_color	"account"	"lose"
8	"receive"	"que"	double_slashes_ average_count	"enron"
9	"wrote"	"enron"	recipient_name_body	"dear"
10	"que"	"pm"	Binary_HTML_Links	"nsm"
11	"ebay"	"dear"	Binary_Differ_ Displayed_Link	"mailbox"
12	automated_ readability_index	"remove"	Anchor_differ_ Displayed_Link	"para"
13	"contribution"	"contribution"	Binary_3_Dots	end_tag_count
14	"mailbox"	"dnc"	Binary_URL_ BagofWords	"dnc"
15	count_href_tag	number_of_special _characters_body	"account"_count_ in_body	"paypal"_count _in_body
16	"bank"_ count_in_body	number_of_html_ comment_tags	"your account"_ count_in_body	Number_URL
17	"dnc"	"receive"	hidden_text	"ebay"
18	"outbound"_ count_average	"bitte"	"paypal"	dale_chale _readability_index
19	table_tag_count	Body_Size	"ebay"	"account"
20	"unsubscribe"	"wrote"	"protect"	"privately"

- *Received_count*: Number of Received header field.
- *Plain_Text*: A binary feature that is equal to 1 if the email is in plain text format.
- *Proportion_Words_No_Vowels*: Ratio of words with no vowels to the total number of words in the email body.
- *Binary_Img_Links*: A binary feature that is equal to 1 if there is an <img> tag in the email body.
- *Binary_Differ_Display_Link*: A binary feature that equals 1 if a link displayed in the email body is different than the redirected website.
- *Binary_URL_Bag_of_Words*: A binary features that equals 1 if one or more of the following words *'click', 'here', 'login', or 'update'* are in a URL in the email body [2].
- *Protocol_Port_Match_Binary*: A binary feature that equals 1 if a URL in the email body does not have a matching protocol and port e.g., HTTP with a different port than 8080.

In both website and email experiments, rankings obtained from IG and Gini are close to each other but has lower correlation from Chi-2. RFE's ranking is different from the other three methods. Instead of using rank aggregation methods for combining the different rankings, we first study the correlation between these rankings for *all* the features as suggested in [59]. Tables 16, 17, and 18 show the pairwise Spearman rho's correlation of ranks for all website and email features. We observe a strong correlation between IG and Gini in both website and email features. Chi-2 has a low correlation with other ranking methods. RFE shows different correlations for website and email features. There is a weak correlation for website features, but moderate-high correlation for email features. Because of these disagreements between different

TABLE 16. Spearman rho correlations between different ranking methods on website dataset.

Ranking	Gini	Chi-2	RFE
IG	0.845	0.299	0.078
Gini		0.296	0.197
Chi-2			0.158

TABLE 17. Spearman rho correlation between different ranking methods on all Email datasets combined (With header).

Ranking	Gini	Chi-2	RFE
IG	0.56	0.14	0.37
Gini		0.27	0.47
Chi-2			0.47

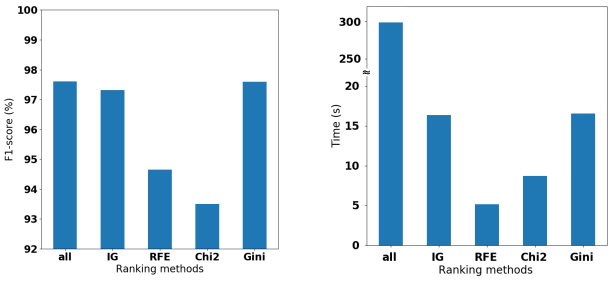
TABLE 18. Spearman rho correlation between different ranking methods on all Email datasets combined (No header).

Ranking	Gini	Chi-2	RFE
IG	0.82	0.14	0.54
Gini		0.18	0.63
Chi-2			0.39

ranking methods, instead of using rank aggregation methods, we report the performance of models trained on each of the top 20 features.

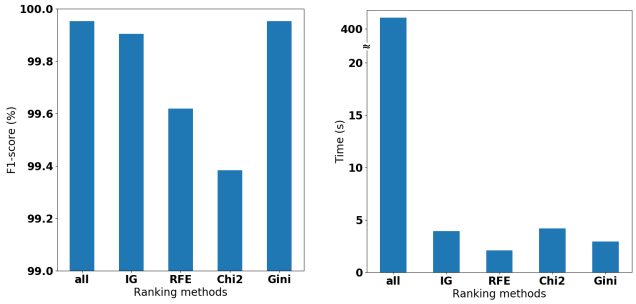
2) FEATURE EVALUATION

We conduct a set of evaluation experiments to further investigate the effect of using the top 20 features on system performance and time. We compare the performance of the models by training them on the set of all features versus using the top 20 features obtained from the four ranking methods mentioned above. Such an evaluation also requires observing whether there are any improvements in training time. For features extracted from the URL datasets, Figure 10 shows the performance of the top classifier on the top 20 features from IG, Gini, Chi-2, and RFE respectively, as well as on all the features. We notice a slight drop in performance with a smaller set of features in all the experiments, but a huge decrease in running time (from 300 seconds to less than 20 seconds).



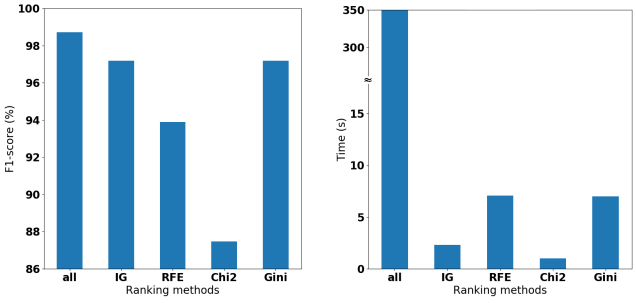
(a) F_1-score (b) Training time

FIGURE 10. Comparison of performance of top classifier with all features vs top 20 features for websites.



(a) F_1-score (b) Training time

FIGURE 11. Comparison of performance of top classifier with all features vs top 20 features for Emails - with header features.



(a) F_1-score Comparison (b) Training time

FIGURE 12. Comparison of performance of top classifier with all features vs top 20 features for Emails - without header features.

Figures 11 and 12 show the feature evaluation for features extracted from emails. We report in these figures the classifier that achieved the highest F1_score using all features and its training time. Same as the URL dataset, we observe a noticeable decrease in training time with a small drop in performance when using the top 20 features.

Models trained on top features selected by Chi-2 performed worse compared to Information Gain in both website and email detection. We converted all the features into binary using one-hot encoding after extracting the features. This could be the reason for Chi-2's poor performance since it is sensitive to small frequencies in cells of the contingency table (which happens as a result of one-hot encoding and having sparse feature space). On the other hand, it ameliorates the limitation of Information Gain, viz., choosing features that take a large range of values.

The conclusion is that the trade-off between time and performance should be considered by the users and researchers when deploying their detection methods.

3) FEATURE EXTRACTION TIME

When a user wants to open a website or click on a link in an email, a detection system is expected to detect malicious contents before they can cause any harm to the user. So, real-time detection of phishing websites/emails is a critical part of detection systems. The inference speed of the model used for detection is one aspect of having a real-time detection

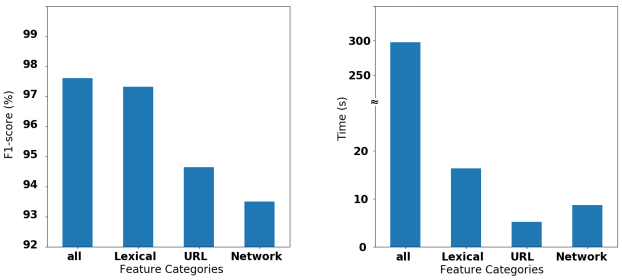
system. On the other hand, extracting the features from emails and URLs is something that the system needs to do for every instance. Table 19 shows the extraction time for some of the website features. It only contains a subset of the features that take the most amount of time. We do not report extraction time for email features since they are based on text analysis, which is very fast (on the order of milliseconds and microseconds). As seen in Table 19, the extraction of URL features is much faster than that of lexical (all types syntactic, semantic and pragmatic) features. Network features take the most time for extraction. Adding network and lexical features increases the performance of the model, but it adds a time overhead for feature extraction. There is a trade-off between adding expensive features and model performance, which is generally ignored.

TABLE 19. Avgerage extraction time for a subset of website features (ms). Gray cells are subset of the features on top of them in white cells (e.g., *dns ttl* is subset of *DNS lookup*.

URL		Lexical		Network	
Feature	Time	Feature	Time	Feature	Time
url length	0.0003	▽**HTML GET**	2.393s	▽**whois**	2.49s
domain length	0.0115	# of head	0.178	creation date	0.0073
Kolmogorov Shmirnov	0.2639	# of tags	0.256	expiration data	0.0019
url character distance	0.16	# of html	0.2	updated date	0.0019
digit-letter ratio	0.0115	# of body	0.2	AS number	0.0005
top level domain	0.013	# suspicious content	48	▽**DNS lookup**	618.0
average path token length	0.0146	# of iframes	0.2	# of name server	0.0006
longest domain token	0.0097	# of input	0.2	dns_ttl	262.27
# of punctuation	0.0077	# of img	0.2	DNS info Exists	18.7
has more than 3 dots	0.0010	inbound count	9.9661	**ipwhois**	270.01
Kullback Leibler divergence	0.0756	outbound count	9.9237		
has at symbol	0.0003	rightclick disabled	0.4		
Total	0.721s	Total	2.464s	Total	3.66s

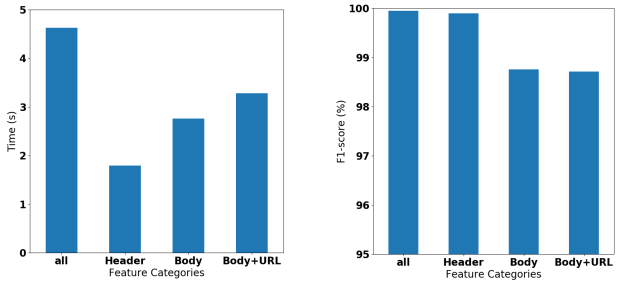
To get a better understanding of the trade-off between the features and the performance of the model, we extract features from different classes: lexical (all types syntactic, semantic and pragmatic), URL, and network, and evaluate the classifier performance on each of these categories separately. This experiment also helps us gain insights regarding influential feature classes that contribute to the classifier performance and which features lead to a faster classification. For email features, the extraction time is very low (on the order of milliseconds and microseconds) so they do not affect the classification time.

Figures 13 and 14 show the F_1-score and training time of top classifier trained on each category of website and email features. For website features, considering both F_1-score and training time, lexical (HTML) features give the best performance. It reduces the training time by a factor of 18 and only loses 0.3% of F_1-score. The problem with lexical features, as reported in Table 19, is their extraction time, which slows down the overall detection time. For email features, using only the header features gives almost the same performance compared to using all features. It also reduces the training time and unlike websites' lexical features, it does not require huge extraction time.



(a) F_1-score (b) Time

FIGURE 13. Comparison of the performance of top classifier with all features vs features from each category of websites features.



(a) F_1-score (b) Time

FIGURE 14. Comparison of the performance of top classifier with all features vs features from each category of email features.

4) KEY TAKEWAYS

We learned from the feature ranking and evaluation experiment:

- We found a strong correlation between feature rankings obtained by the IG and Gini methods. On the other hand, correlations of Chi-2's rankings with rankings of other methods were very low.
- For websites, features from all top three categories, syntactic, semantic and pragmatic, described in our proposed taxonomy contribute to the classifier performance. On the other hand, for email, TFIDF features contributed the most to the performance of classifiers.
- Training models on a subset of top features is 20 times faster compared to using all features with almost the same classification performance (0.5% reduction).
- Feature extraction time is an important aspect of having a real-time detection system for websites. Extracting network and HTML features increases extraction time by six seconds on average and delays the final classification.

E. COMPARING WITH PREVIOUS WORK

In the following experiments, we replicate the methods in selected research from the literature on phishing detection and compare them on the same datasets using PhishBench. The goal of this experiment is two-fold: (i) Do a proper comparison with previous works i.e., same classifier parameters, same dataset, and one set of evaluation metrics.

(ii) Identify how methods developed in the past perform on a different and imbalanced dataset.

The papers were selected based on how well the security challenges were addressed, reproducibility of the system, and the rank of the conference or the journal that published the paper (venues with h5-index higher than 20).

We selected a total of nine systems for our comparison experiment: three each on URL detection [10], [36], [57], website detection [41], [61], [66], and email detection [31], [34], [65]. The datasets used for testing the detection systems have different legitimate to phishing ratios (1:1, 1:5, and 1:10), and were used previously in Sections VII-B.1 and VII-B.2 in the ratio experiment for URLs/Websites and emails respectively. For emails, we use the With Header dataset. Before reporting results, we explain the selected research and features that were extracted to reproduce these systems.

1) PHISHING URL/WEBSITE DETECTION

The model proposed in [61] used Google's PageRank and Gmail domain reputation which is not public, so we trained and tested their model without those features. Researchers in [41] just studied several features on phishing and legitimate websites without introducing any model. We train all the classifiers in PhishBench with those features and report the best result. Authors in [66] used the Extreme Learning Machine (ELM) classifier that we did not have in PhishBench, so we added ELM to PhishBench to evaluate their method. We used an existing Python library for the ELM classifier.[13]

Researchers in [36] use character and word based n-grams as features to train a deep convolutional neural network. Authors in [10] use bag-of-words based lexical features extracted from URLs, which are used as input to a confidence-weighted online learning algorithm for classification. In [57], authors use a different set of attributes, viz., distance-based features (K-L Divergence, Euclidean distance, character frequencies), as well as URL length ratio, presence of suspicious words, frequency of target words, punctuation symbols, etc.

2) PHISHING EMAIL DETECTION

Islam and Abawajy [34] developed a multi-tier classifier to detect phishing emails. Their system relies on a majority vote between three classifiers. They extracted 21 features in total from each email: six from the subject, two from the priority and content-type headers, and 12 from the message body.

Zareapoor and Seeja [65] used different feature ranking and feature selection methods to shorten the feature vector and increase the accuracy of their Bagging classifier. The authors extracted 20 features previously used in [2]. The length of the extracted feature vector also suggests the use of a Term Frequency method, which builds a matrix of all the words in the corpus and respective frequency in each document.

[13]https://pypi.org/project/elm/

Ganesh *et al.* [31] used FastText [11] which is a library for text classification and representation learning. It transforms the text into continuous vectors that can be used in language-related tasks. The authors were able to achieve high results in the IWSPA-AP 2018 phishing pilot [1].

3) RESULTS

Using PhishBench, we replicated the features used in these papers and adjusted the classification methods according to the methods they described. We implemented and ran these methods with hyper-parameter tuning, even if some of the works did not report using it. The main issue that we encountered during this experiment is the reproduction of the methods implemented by previous research. We found that a majority of the detection literature lacks sufficient details or clear explanations, thus hurting their reproducibility. However, we take care to be as faithful as possible, erring on the side of generosity when details are missing, and do our best to replicate the work described.

Table 20 shows the performance reported by the previous works and their PhishBench implementations on each detection category (URL, website, and email). We tested them on three datasets with different ratios (1:1, 1:5, and 1:10), which we used in the ratio experiment in Section VII-B. For emails, we only used the ''with header'' dataset to make the comparison fair.

TABLE 20. F_1**-score of previous works and our model on URLs, website and email detection.**

Type	Paper	Reported Performance	PhishBench Results		
			Balanced	1to5	1to10
URLs	[57]	92.7%	89.3%	81.4%	75.6%
	[36]	93.6%	94.7%	89.5%	76.1%
	[10]	88.7%	83.4%	78.5%	71.8%
Websites	[61]	96.2%	95.9%	87.7%	71.9%
	[66]	99.2%	69.3%	26.2%	4.7%
	[41]	NA	95.2%	88.1%	76.3%
Emails	[34]	97.0%	90.2%	92.46%	96.53%
	[65]	97.7%	99.0%	98.0%	98.0%
	[31]	99.0%	99.0%	98.0%	89.0%

As we can see, in most of the cases the reported performance by the previous works is only slightly lower than their performance on our balanced dataset except for [34], [66]. We suspect the main cause for lower comparative performance of [34] is the use of term frequency features and word embedding (FastText) by [65] and [31] respectively. The difference is much bigger for [66] and we believe the inferior performance of the ELM classifier is the main reason for this observation (it gave the same performance even when trained using all the features). We also notice a decline in performance as the datasets are more imbalanced, except for [34]. This shows that their implementation of majority voting between three classifiers works well with imbalanced datasets. We see from this experiment that text analysis methods like TFIDF and word embedding capture the inherent differences between legitimate and phishing samples, and as

long as the model is retrained, it should scale well with new and bigger datasets.

4) KEY TAKEWAYS

We learned from this experiment the following:

- Phishing attacks are evolving every day and training the existing defense mechanism on newer attack samples is not enough to detect new attacks.
- We observed a decrease in previous works' reported performance when we tested their methods on a recent dataset even with retraining. This shows the importance of having a robust set of features that go beyond current attacks and selecting a diverse and high-quality dataset for model building.
- Seven out of nine of the previous works do not perform well in an imbalanced scenario.

F. FUTURE ATTACKS

Phishing attacks are constantly evolving with time. it is then necessary to test the robustness of detection systems. One way to do that is to train classification methods on one dataset and then test on a different one. We do this experiment on URLs and emails using PhishBench.

URLs: We collected 2,000 new URLs from PhishTank and Alexa (1,000 each) in July 2019 which resulted in 718 and 887 valid URLs respectively (valid URLs are those for which all the features are successfully extracted). We used the top three classification models trained on the balanced experiment in Section VII-B.1 without any retraining (Bagging, Boosting and AutoSklearn). These models were able to achieve 57.11%, 74.67% and 79.95% F_1-score respectively using all the features. We see a huge drop comparing to the results that we got in Section VII-B (Bagging: 97.61%, Boosting 97.59, AutoSklearn: 97.47%). To examine the effect of retraining on the performance, we split the 1,605 newly collected URLs into train and test sets and retrain the top three classifiers with the old dataset plus the new training set. It improved the performance of two classifiers (Bagging: 70.41%, AutoSklearn: 81.53%) and decreased Boosting's performance to 62.9%. Despite the improvement, it is still far from a perfect detection system. It shows that phishing attacks are evolving so rapidly that simple retraining is not enough to keep up with them.

Emails: We collected 197 phishing emails that were *not* detected by our university mailing system from 2013-2018 and built a test set. We used the classifiers as trained in the balanced experiment in Section VII-E.2. We ran the best models for the ''with header'' dataset, which were LR, SVM, AutoSK, and TPOT (Table 10). In terms of accuracy, the models achieved 97.96%, 96.44%, 97.46%, and 98.47% respectively. We notice a slight decline in performance compared to the previous experiment, however, we can argue that these are good results considering these emails were not used in the training set.

We also evaluated our top models on the publicly available Employment Scam Aegean Dataset,[14] which contains 866 fraudulent job advertisements and 17,014 legitimate ones published between 2012 and 2014. These emails do not have headers, thus we used the top 3 models for no header emails, which are AutoSk, Boosting, SVM in that order (Table 11). We conducted this evaluation in two ways. i) we use the whole Jobscam dataset as the testing set and the models trained on the balanced dataset in Section VII-B without retraining. The model performance was quite low. The SVM classifier achieved the highest F_1-score of 19.01%. Boosting and AutoSklearn come in second and third with F_1-scores of 9.61% and 4.36% respectively. ii) In the second experiment, to examine the retraining effect, we included 90% of the emails from the Aegean Dataset in the training set and the remaining 10% was used for testing. The testing set did not include any emails from the other datasets. Similar to the URL experiment, we observed that retraining improved the results but the classifier performance is still low. The SVM classifier achieved the best result with 86.15% F_1-score. AutoSklearn did better than Boosting with 59.87% F_1-score. The Boosting classifier achieved the lowest results with 18.94% F_1-score.

This experiment emphasizes the importance of verifying the robustness of classification models and the need for regular retraining with newer types of attacks as well as revising the feature sets to cope with ever evolving attacks.

Key takeways: Attackers always change their attack techniques to bypass defense mechanism. Retraining using a more recent dataset slightly helps existing models to detect newer attacks but, as we mentioned in the previous section, retraining alone will not be enough to deal with new attacks.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel taxonomy of features for phishing emails, websites and URL detection based on their structure and how the attributes are processed by the web and email servers. Then, we proposed PhishBench – a novel phishing detection framework intended to act as a ready-to-use platform for security researchers to compare their work with proposed state-of-the-art phishing detection methods using proper evaluation metrics and different classification algorithms.

Through a variety of experiments, we evaluated several dimensions of the phishing problem including scaling and ratio of phishing to legitimate class. Using PhishBench, we conducted a benchmarking study to evaluate phishing detection features used in previous literature and their performance using different classification methods on balanced and imbalanced datasets. The experiments showed that the classification performance dropped when the ratio between phishing and legitimate decreased towards 1 to 10. The decline in performance ranged from 5.9% to 42% in F_1-score. Additionally, PhishBench was also used to test previous methods

[14] http://emscad.samos.aegean.gr/, Accessed: July 18, 2019

on our new and diverse datasets – these experiments proved that term frequency features like TFIDF and word embedding methods are robust as long as they are retrained on newer datasets.

There are many directions for future work. For example, more features could be extracted from the phishing datasets. Another possibility for future improvement is including real-time retraining and evaluation capability.

ACKNOWLEDGMENT

(Ayman El Aassal and Shahryar Baki contributed equally to this work.)

REFERENCES

[1] A. E. Aassal, L. Moraes, S. Baki, A. Das, and R. Verma, "Anti-phishing pilot at ACM IWSPA 2018: Evaluating performance with new metrics for unbalanced datasets," in *Proc. Anti-Phishing Shared Task Pilot at 4th ACM IWSPA*, 2018, pp. 2–10. [Online]. Available: http://ceur-ws.org/Vol-2124/#anti-phishing-pilot

[2] D. Almomani, T.-C. Wan, A. Manasrah, A. Taha, M. Baklizi, and S. Ramadass, "An enhanced online phishing e-mail detection framework based on evolving connectionist system," *Int. J. Innov. Comput. Inf. Control*, vol. 9, no. 3, pp. 169–175, 2012.

[3] A.-P. W. Group. (2019). *Phishing Activity Trends Report–1st Quarter 2019*. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2019.pdf

[4] A. Balaji and A. Allen, "Benchmarking automatic machine learning frameworks," Aug. 2018, *arXiv:1808.06492*. [Online]. Available: https://arxiv.org/abs/1808.06492

[5] S. Baskaran. (2018) *Phishing URL Classification*. [Online]. Available: https://github.com/srirambaskaran/phishing-url-classification

[6] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for models assessment over imbalanced datasets," *J. Inf. Eng. Appl.*, vol. 3, no. 3, p. 10, 2013.

[7] A. Bergholz, J. Chang, G. Paaß, F. Reichartz, and S. Strobel, "Improved phishing detection using model-based features," in *Proc. CEAS*, Aug. 2008, pp. 1–10.

[8] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, and S. Strobel, "New filtering approaches for phishing email," *J. Comput. Secur.*, vol. 18, no. 1, pp. 7–35, Jan. 2010.

[9] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[10] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in *Proc. 3rd ACM workshop Artif. Intell. Secur.*, 2010, pp. 54–60.

[11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Jul. 2016, *arXiv:1607.04606*. [Online]. Available: https://arxiv.org/abs/1607.04606

[12] K. L. Chiew, K. S. C. Yong, and C. L. Tan, "A survey of phishing attacks: Their types, vectors and technical approaches," *Expert Syst. Appl.*, vol. 106, pp. 1–20, Sep. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418302070

[13] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," *Data Min Knowl. Discovery*, vol. 24, no. 1, pp. 136–158, Jan. 2012, doi: 10.1007/s10618-011-0222-1.

[14] A. Cohen, N. Nissim, and Y. Elovici, "Novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods," *Expert Syst. Appl.*, vol. 110, pp. 143–169, Nov. 2018.

[15] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran, "A lexical approach for classifying malicious URLs," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2015, pp. 195–202.

[16] A. Das, S. Baki, A. El Aassal, R. Verma, and A. Dunbar, "SOK: A comprehensive reexamination of phishing research from the security perspective," *IEEE Commun. Surveys Tuts.*, to be published.

[17] P. Dewan and P. Kumaraguru, "Towards automatic real time identification of malicious posts on Facebook," in *Proc. 13th Annu. Conf. Privacy, Secur. Trust (PST)*, Jul. 2015, pp. 85–92.

[18] P. Dewan and P. Kumaraguru, "Facebook inspector (FbI): Towards automatic real-time detection of malicious content on Facebook," *Social Netw. Anal. Mining*, vol. 7, no. 1, p. 15, 2017.

[19] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani, "Systematization of knowledge (SOK): A systematic review of software-based Web phishing detection," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2797–2819, Sep. 2017.

[20] B. Eshete and V. N. Venkatakrishnan, "WebWinnow: Leveraging exploit kit workflows to detect malicious URLS," in *Proc. 4th ACM Conf. Data Appl. Secur. Privacy*, 2014, pp. 305–312.

[21] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[22] M. N. Feroz and S. Mengel, "Examination of data, rule generation and detection of phishing URLs using online logistic regression," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2014, pp. 241–250.

[23] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Nice, France: Curran Associates, 2015, pp. 2962–2970.

[24] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proc. ACM workshop Recurring Malcode (WORM)*, 2007, pp. 1–8.

[25] P. J. A. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source AutoML benchmark," Jul. 2019, *arXiv:1907.00909*. [Online]. Available: https://arxiv.org/abs/1907.00909

[26] N. Gupta, A. Aggarwal, and P. Kumaraguru, "Bit.ly/malicious: Deep dive into short URL based e-crime detection," in *Proc. APWG Symp. Electron. Crime Res.(eCrime)*, Sep. 2014, pp. 14–24.

[27] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.

[28] I. R. A. Hamid and J. Abawajy, "Phishing email feature selection approach," in *Proc. IEEE 10th Int. Conf. Trust, Security Privacy Comput. Commun.*, Nov. 2011, pp. 916–921.

[29] I. R. A. Hamid and J. H. Abawajy, "Profiling phishing email based on clustering approach," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jul. 2013, pp. 628–635.

[30] D. Hassan, "On determining the most effective subset of features for detecting phishing Websites," *Int. J. Comput. Appl.* , vol. 122, no. 20, pp. 1–7, Jul. 2015.

[31] H. B. B. Ganesh, R. Vinayakumar, M. A. Kumar, and K. P. Soman, "Distributed representation using target classes: Bag of tricks for security and privacy analytics," in *Proc. 4th ACM Int. Workshop Secur. Privacy Anal. (IWSPA)*, Mar. 2018, pp. 1–6.

[32] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[33] C. Huang, S. Hao, L. Invernizzi, J. Liu, Y. Fang, C. Kruegel, and G. Vigna, "Gossip: Automatically identifying malicious domains from mailing list discussions," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, 2017, pp. 494–505.

[34] R. Islam and J. Abawajy, "A multi-tier phishing detection and filtering approach," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 324–335, Jan. 2013, doi: 10.1016/j.jnca.2012.05.009.

[35] B. Klimt and Y. Yang, "The Enron corpus: A new dataset for email classification research," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 2004, pp. 217–226.

[36] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," Feb. 2018, *arXiv:1802.03162*. [Online]. Available: https://arxiv.org/abs/1802.03162

[37] S. Lee and J. Kim, "Warning bird: A near real-time detection system for suspicious URLs in Twitter stream," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 3, pp. 183–195, May 2013.

[38] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, vol. 18, no. 17, pp. 1–5, 2017.[Online]. Available: http://jmlr.org/papers/v18/16-365

[39] J. Ma. (2009). *Detecting Malicious*. [Online]. Available: http://www.sysnet.ucsd.edu/projects/url/

[40] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious URLs using lexical analysis," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2016, pp. 467–482.

[41] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi," in *Proc. LEET*, Apr. 2008, p. 4.
[42] P. Meesad, P. Boonrawd, and V. Nuipian, "A chi-square-test for word importance differentiation in text classification," in *Proc. Comput. Sci. Inf. Technol.*, vol. 6, pp. 110–114, Jan. 2011.
[43] J. Nazario. (2004). *The Online Phishing Corpus*. [Online]. Available: https://monkey.org/~jose/phishing/
[44] A. Niakanlahiji, B.-T. Chu, and E. Al-Shaer, "PhishMon: A machine learning framework for detecting phishing Web pages," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2018, pp. 220–225.
[45] P. Nohe. (2018). *HTTPS Phishing: 49% of Phishing Websites Now Sport The Green Padlock*. [Online]. Available: https://www.thesslstore.com/blog/https-phishing-green-padlock/
[46] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, Jul. 2016, pp. 485–492.
[47] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[48] K. D. Rajab, "New hybrid features selection method: A case study on Websites phishing," *Secur. Commun. Netw.*, vol. 2017, Mar. 2017. Art. no. 9838169.
[49] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious URL detection using machine learning: A survey," Jan. 2017, *arXiv:1701.07179*. [Online]. Available: https://arxiv.org/abs/1701.07179
[50] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1986.
[51] R. Shukla. (2015) *Identifies Phishing Websites Using a Treebag Model*. [Online]. Available: https://github.com/rishy/phishing-websites
[52] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, Berleley, CA, USA, May 2010, pp. 305–316.
[53] (2018). *Spam Assassin*. [Online]. Available: http://www.csmining.org/index.php/spam-assassin-datasets.html
[54] T H Team. (2015). *H_2O: Python Interface for H2O, Python Package Version 3.1.0.99999*. [Online]. Available: http://www.h2o.ai
[55] E. Test, L. Zigic, and V. Kecman, "Feature ranking using Gini index, scatter ratios, and nonlinear SVM RFE," in *Proc. IEEE Southeastcon*, Apr. 2013, pp. 1–5.
[56] F. Toolan and J. Carthy, "Feature selection for Spam and Phishing detection," in *Proc. eCrime Res. Summit*, 2010, pp. 1–12.
[57] R. Verma and K. Dyer, "On the character of phishing URLs: Accurate and robust statistical learning classifiers," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, 2015, pp. 111–122.
[58] R. Verma, M. Kantarcioglu, D. Marchette, E. Leiss, and T. Solorio, "Security analytics: Essential data analytics knowledge for cybersecurity professionals and students," *IEEE Secur. Privacy*, vol. 13, no. 6, pp. 60–65, Nov. 2015.
[59] R. M. Verma and M. Fastovets, "Meta-searching: Should search engine rankings be aggregated," Univ. Houston, Houston, TX, USA, Tech. Rep. UH-CS-10-09, 2010.
[60] E. Volkman. (2018). *The 2018 Phishing Trends & Intelligence Report Now*. [Online]. Available: https://info.phishlabs.com/blog/2018-phishing-trends-intelligence-report-released
[61] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," *Proc. 17th NDSS*, Jan. 2010.
[62] (2018). *WikiLeaks*. [Online]. Available: https://www.wikileaks.org/
[63] Y. Wu, S. C. Hoi, C. Liu, J. Lu, D. Sahoo, and N. Yu, "SOL: A library for scalable online learning algorithms," *Neurocomputing*, vol. 260, pp. 9–12, Oct. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217306239
[64] A. Yasin and A. Abuhasan, "An intelligent classification model for phishing email detection," Aug. 2016, *arXiv:1608.02196*. [Online]. Available: https://arxiv.org/abs/1608.02196
[65] M. Zareapoor, , and S. K. R, "Feature extraction or feature selection for text classification: A case study on phishing email detection," *Eng. Electron. Bus.*, vol. 7, no. 2, pp. 60–65, Mar. 2015.
[66] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage ELM for phishing Web pages detection using hybrid features," *World Wide Web*, vol. 20, no. 4, pp. 797–813, Jul. 2017.

AYMAN EL AASSAL received the master's degree in computer science from the University of Computer Science (ENSIAS), Rabat, Morocco. He is currently pursuing the Ph.D. degree with the University of Houston. His current field of research is in cyber security focusing on intrusion detection.

SHAHRYAR BAKI received the bachelor's degree in computer engineering from the University of Tehran, and the master's degree in computer engineering from the Sharif University of Technology, Tehran, Iran. He is currently pursuing the Ph.D. degree with the University of Houston. His current fields of research are cyber security, usable security, and natural language processing.

AVISHA DAS received the B.Tech. degree in electronics and communication engineering from the West Bengal University of Technology, West Bengal, India. She is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Houston. Her research interests include natural language understanding and generation, cybersecurity, and information retrieval.

RAKESH M. VERMA (Member, IEEE) is currently a Professor of computer science with the University of Houston (UH), where he is leading a research group that applies reasoning and data science to cybersecurity challenges. He is the coauthor of *Cybersecurity Analytics* (CRC Press, 2019), which discusses key data analysis techniques for cybersecurity challenges. He has co-organized the 1st Anti-phishing Shared Task, in 2018 with proceedings in the CEUR workshop series; the training dataset is publicly available for academic research upon request. Since 2015, he has been co-organizing and editing the proceedings of the ACM International Workshop on Security and Privacy Analytics. He was an Editor of Frontiers of Big Data in the Cybersecurity Area, an ACM Distinguished Speaker, from 2011 to 2018, and the winner of two Best Paper Awards. He received the Lifetime Mentoring Award from UH. He is a Fulbright Senior Specialist in computer science.