Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
COVID-19 malicious domain names classification

Paul K. Mvula *, Paula Branco, Guy-Vincent Jourdan, Herna L. Viktor

School of Electrical Engineering and Computer Science (EECS), University of Ottawa, Ottawa, ON K1N 6N5, Canada

A R T I C L E I N F O

Keywords:
Machine learning
Cybersecurity
Phishing attacks
Supervised learning
Hoeffding trees
Online learning

A B S T R A C T

Due to the rapid technological advances that have been made over the years, more people are changing their way of living from traditional ways of doing business to those featuring greater use of electronic resources. This transition has attracted (and continues to attract) the attention of cybercriminals, referred to in this article as “attackers”, who make use of the structure of the Internet to commit cybercrimes, such as phishing, in order to trick users into revealing sensitive data, including personal information, banking and credit card details, IDs, passwords, and more important information via replicas of legitimate websites of trusted organizations. In our digital society, the COVID-19 pandemic represents an unprecedented situation. As a result, many individuals were left vulnerable to cyberattacks while attempting to gather credible information about this alarming situation. Unfortunately, by taking advantage of this situation, specific attacks associated with the pandemic dramatically increased. Regrettably, cyberattacks do not appear to be abating. For this reason, cybersecurity corporations and researchers must constantly develop effective and innovative solutions to tackle this growing issue. Although several anti-phishing approaches are already in use, such as the use of blacklists, visuals, heuristics, and other protective solutions, they cannot efficiently prevent imminent phishing attacks. In this paper, we propose machine learning models that use a limited number of features to classify COVID-19-related domain names as either malicious or legitimate. Our primary results show that a small set of carefully extracted lexical features, from domain names, can allow models to yield high scores; additionally, the number of subdomain levels as a feature can have a large influence on the predictions.

1. Introduction

The novel coronavirus disease, or COVID-19, is a highly contagious respiratory and vascular disease caused by infection with severe acute respiratory syndrome coronavirus 2 (SARS-Cov-2). On December 8, 2019, WHO officials reported the earliest onset of symptoms, and they declared the outbreak a pandemic on March 11, 2020. The disease went on to infect over 460 million people and caused over 6 million deaths in 225 countries and territories, as of March 15, 2022. As a result of the COVID-19 pandemic, public health officials and governments have been working together to lower the transmission and death rates and control the pandemic. For example, they mandated various measures in place, including social distancing, mandatory wearing of face masks when going into public and crowded places such as supermarkets, and frequent hand washing. Most importantly, officials in many countries ordered people to stay at home and avoid unnecessary trips. Moreover, some governments imposed stricter measures, such as border closures, lockdowns, and curfews. Despite these measures, the transmission rate and death toll have not abated. The active measures put in place have forced companies worldwide to switch from offline working to remotely working from home in response to some countries’ authorities banning gatherings of more than five people. In addition, workers who had traveled overseas for holidays were stuck in the countries they were visiting. Schools were also forced to change over to distance learning. The IBM X-Force Incident Response and Intelligence Services (IRIS), 1 together with Quad9, 2 a public recursive DNS resolver offering end users security, high performance, and privacy, have been tracking cybercrime that has capitalized on the coronavirus pandemic since its beginnings. These services have uncovered a variety of COVID-19 phishing attacks against individuals and organizations with a potential interest in the technologies associated with the safe delivery of the

1. The code (and data) in this article has been certified as Reproducible by CodeOcean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.

E-mail addresses: pmvul089@uottawa.ca (P.K. Mvula), pbranco@uottawa.ca (P. Branco), gjourdan@uottawa.ca (G.-V. Jourdan), hviktor@uottawa.ca (H.L. Viktor).

1. https://www.ibm.com/security/services/ibm-x-force-incident-response-and-intelligence.

2. https://quad9.net.

https://doi.org/10.1016/j.eswa.2022.117553

Received 22 April 2021; Received in revised form 4 May 2022; Accepted 7 May 2022
Available online 20 May 2022
0957-4174/© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-
nc-nd/4.0/).
vaccine against the coronavirus. IRIS and Quad9 observed that over 1000 coronavirus-related malicious domain names were created between February and March 2020. Correspondingly, in a trend report released in 2020, the Anti-Phishing Working Group (APWG), an international consortium dedicated to promoting research, education, and law enforcement to eliminate online fraud and cybercrime, has revealed an increase in the number of unique malicious websites. Since that remarkable spike in March, the same month the outbreak was declared a pandemic, the number of COVID-19-related threat reports has been increasing on a week-to-week basis, according to data collected from IBM X-Force. Meanwhile, Quad9’s attempts to detect and block malicious COVID-19-related domain names have led to reports of a substantial increase in the number of domain names blocked as being associated with malicious COVID-19 activity. According to IBM X-Force IRIS research, attackers are using email, SMS text messages, and social media to deliver malicious websites of spoofed government portals requiring users to input financial and personal data. After obtaining this information from victims, attackers may open bank accounts and even apply for loans in their names. These findings highlight that anti-phishing developments are being made by industry and academia to prevent attacks, paying special attention to those linked to COVID-19. (Basit, 2021). This emphasizes the significance of developing effective solutions for COVID-19-related attacks, which are on the rise. According to Le Page (2019), anti-phishing development efforts can be grouped into 3 main areas, as depicted in Fig. 1:

- Detection techniques.
- Understanding the phishing ecosystem (phishing prevention).
- User Education.

Although user education and understanding the phishing ecosystem are crucial for phishing prevention, end users, as the weakest link in the security chain (Schneier, 2000), remain vulnerable to attacks because attackers can use new techniques and exploit those vulnerabilities to trick them into clicking on links and inputting their personal information. This phenomenon was especially evident during the onset of the COVID-19 pandemic, where the extraordinary situation and subsequent lockdowns left end users more prone to respond atypically to phishing attacks. Specifically, cybercriminals leveraged people’s heightened online activity spurred by COVID-19 to launch coronavirus-themed phishing attacks. In this paper, we focus on detection techniques in the context of the pandemic. This paper introduces a ML model that classifies COVID-19-related domain names as malicious or legitimate by using different ML algorithms and comparing them to yield the best result. First, we collected legitimate, confirmed malicious domain names containing keywords related to COVID-19 from publicly available sources to construct our own data set. Next, we extracted useful features from the generated data set and then moved on to perform model construction and algorithm tuning. In summary, our contributions are as follows:

1. We addressed the problem of malicious domain name detection, focusing on those related to COVID-19.
2. We proposed a small feature set that may be extracted from the data in a timely manner.
3. We developed both online and batch learning models for malicious domain name detection.
4. We conducted several experiments with varied data distributions and feature sets to evaluate the performance of both the batch and online learning methods.

The remainder of the paper is organized as follows: the next section examines related works and literature concerning malicious URLs and domain name detection. Then, Section 3 discusses the difficulties involved in the detection of a malicious domain name. The steps taken in the acquisition of the data set and the feature extraction process are explained in Section 4, while Section 5 describes the experimental setup and proposed model. Finally, Section 6 presents and compares the results of the different algorithms, and Section 7 offers conclusions and suggests future works on the topic.

2. Related works

Software-based detection techniques are generally divided into three classes: visual similarity-based (VSB) detection systems (Jain & Gupta, 2017), list-based (LB) detection systems (Cao et al., 2008; Han et al., 2012), and machine learning based (MLB) detection systems.

2.1. VSB detection systems

VSB approaches can be grouped into HTML DOM (HyperText Markup Language Document Object Model), Cascading Style Sheet (CSS) similarity, visual features, visual perception, and hybrid approaches (Almomani, 2013). Rosiello et al. (2007) presented an approach based on the reuse of the same data across different sites. In the proposed approach, a warning would be generated if a client reused the same data (i.e., same username, password, and so forth) on numerous websites. The system analyzed the DOM tree of the first web page where the data was initially entered and the subsequent web page where the information was reused. If the DOM tree between these two web pages was discovered to be similar, then the system considered it as an attack, or else a legal reuse of data. Although this system presented a high true positive rate of almost 100%, it failed when the malicious sites contained only images. Abdelnabi et al. (2020) introduced Visual-PhishNet, a triplet-network framework that learned a similarity metric between any two same-website pages for visual similarity phishing detection. The training of VisualPhishNet was divided into two stages: training on all screenshots with random sampling and fine-tuning by retraining on those samples that were difficult to learn in the first stage. The authors reported a Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) of 98.79% on their data set, VisualPhish, which contained 155 websites with 9363 screenshots. Although this approach yielded a high ROC and could detect zero-day attacks, it failed when the malicious pages had pop-up windows covering the site’s logo.

Fig. 1. Anti-phishing Development Areas. Varshney et al. (2016).
Dunlop et al. (2010) developed goldphish, a browser plug-in, which was designed to recognize malicious sites. It extracted the suspicious site’s logo and converted it to text using optical character recognition (OCR) software; the extracted text was then used as a query for the Google search engine. The suspicious site’s domain was compared with the top search results, and if a match was detected, the site was declared legitimate; otherwise, it was considered malicious. Although goldphish could detect new attacks (known as zero-hour attacks) and identify well-known companies’ logos, it was unable to convert text from logos with dark backgrounds. Chiew et al. (2015) presented an extension of goldphish that used the logo for phishing detection. This approach was divided into two stages: logo extraction and site identity confirmation. The hybrid method used ML to extract the right site’s logo from all pictures present. Next, the right logo was queried in “Google Images” to obtain the corresponding domain names in the results. Then, like goldphish, the suspicious site’s domain was compared with the top search results. Like goldphish, this extension could detect zero-hour attacks, but it had a false positive rate of 13%.

2.2. LB detection systems

These systems feature two types of lists: one list containing malicious URLs or domain names (the blacklist) and another one containing legitimate URLs or domain names (the whitelist). Blacklists are constructed from previous domain names that have been flagged as malicious. These URLs and domain names can be found in spam messages, by anti-virus software, and from other sources. Blacklisting a domain name makes it impossible for the attacker to reuse the same domain name to create other URLs from the flagged domain name. Tanaka and Kashima (2019) proposed SeedsMiner, a method that collected malicious candidate domain names or URLs from open-source intelligence (OSINT), then used popular anti-virus software to create a blacklist. They first collected domain names and URLs from publicly available feeds, then crawled those URLs/domain names using a client honeypot, and finally assessed the files downloaded by the crawler using four popular anti-virus software packages. The URL/domain name was added to the blacklist if at least one anti-virus flagged a downloaded file originating from it as malicious. The authors compared the blacklist to Google Safe Browsing and found that their method reported 75% more malicious URLs/domain names than Google Safe Browsing.

Cao et al. (2008) developed the automated individual white-list (AIWL), a system that builds a white-list by registering the IP address of each site with a login interface visited by the user. A user who visits a site is notified of any inconsistency with the registered information of the visited website. One of the AIWL’s pitfalls is that it warns the user each time they visit a site with a login interface for the first time.

2.3. MLB detection systems

ML has proven itself a very useful tool in cybersecurity as well as other fields of computer science and has extensively featured in the literature for malicious activity detection. The case of malicious domain name/URL detection is simply a classification task classifying phishing domain names/URLs from legitimate to malicious with the generated weights from the features and data set; therefore, generating a good model requires that the data set must contain good data and a set of relevant features should be extracted from the data. Jain and Gupta (2018) presented an anti-phishing approach that used machine learning to extract 19 client-side features to classify websites. The authors used publicly available phishing pages from Phishtank⁴ and OpenPhish⁵ along with legitimate pages from Alexa’s well-known sites, some top banking websites, online payment gateways, etc. With the use of ML, their proposed approach yielded a 99.39% true positive rate. Zhang et al. (2007) developed CANTINA, a text-based phishing detection technique that extracted a feature set from various fields of a web page using the term frequency–inverse document frequency (TF–IDF) algorithm (Zhang et al., 2007). Next, the top five terms with the highest TF–IDF values were queried using the Google Search engine. If the website was included in the top “n” results, it was classified as legitimate. However, CANTINA was only sensitive to English, which meant its performance was affected by the language used on the website. The enhanced model, CANTINA++, Xiang et al. (2011), included 15 features taken from different fields of the page, such as the DOM tree and URL. Although the system achieved a 92% accuracy rate, it produced many false positives. Sahingoz et al. (2019) used a data set consisting of 73575 URLs, 37175 phishing URLs from Phishtank (2018), and 36400 legitimate URLs collected with Yandex Search API⁶. From these URLs, 40 different NLP-based features were extracted with different modules, such as the Word Decomposer Module (to separate sub-words of words longer than 7 characters), the Maliciousness Analysis Module (to detect typo-squatting by calculating the edit distance between malicious and legitimate URLs) and others, and 1701 word features were extracted, which were then reduced to 102 with a feature-reduction mechanism using Weka’s built-in functions. The authors obtained an accuracy rating of 97.98% using the random forest algorithm. Zhu et al. (2020) proposed DTOSF-ANN (Decision Tree and Optimal Features based Artificial Neural Network), which adopted ANN to construct a classifier. They first evaluated the importance of each of the 30 extracted features from the URL, then proceeded to optimal features selection, resulting in the construction of an optimal feature vector for the classifier. Next, the authors trained the proposed approach on a data set collected by Mohammad et al. (2014), containing 4898 legitimate websites and 6157 phishing websites, then tested the model on 14582 samples collected from Phishtank and Alexa from December 2018 to June 2018, reporting 97.80% accuracy. Verma and Das (2017) proposed a model that separated URLs into n-grams and demonstrated the effectiveness of this method using Shannon’s entropy. The authors made a positive contribution by proving that those character distributions in the URLs were skewed due to confusion techniques used by attackers. Thus, we borrowed the idea of using Shannon’s entropy as a feature from this paper. Tajaddodianfar et al. (2020) proposed TException, a character- and word-level embedding model for detecting phishing websites. They used fastText, Bojanowski et al. (2017), to generate embeddings from words found in URLs and build an alphabet of all characters seen at training in order to generate low-dimensional vectors (embeddings) for the characters. Each of these embeddings was passed to multiple convolutional layers in parallel, each with different filter sizes, whose outputs were then concatenated for classification. They trained and tested their approach on data collected from Microsoft’s anonymized browsing telemetry data. The training set consisted of 1.7M samples, with 20% sampled for validation, and the testing set comprised 20M samples. By randomly initializing word embeddings for the fastText model, the authors reported a 0.28% error rate and 0.9943 Area Under the Receiver Operator Characteristic Curve. In a real-time environment, the detection of an attack should be effective and instantaneous. LB approaches are fast; however, they are limited because the update of those lists in a timely manner constitutes a difficult task and often requires more system resources. Therefore, they are not able to detect zero-hour attacks. In contrast, while VSB approaches may achieve high accuracy, their weaknesses include the following: they are very complex in nature, they have high processing costs because of the necessity of storing prior knowledge about the websites or large databases of images, and they fail to detect zero-hour attacks, (Jain & Gupta, 2016).

Lastly, MLB techniques tend to require computational resources and time for training, as well as frequent updates to the feature set for

⁴ https://www.phishtank.com/.
⁵ https://www.openphish.com/.
⁶ https://yandex.com/dev/xml/doc/.
the proposed approach because it can be efficiently deployed in a DNS.

However, developing an efficient approach that focuses solely on domain names is a strength of domain names. The amount of available data is the key advantage.

Thus, an important challenge associated with using solely domain names, or at least featuring low false positives and false negatives. Therefore, not only was a good data set

3. Challenges in domain name detection

During the COVID-19 pandemic, attackers have employed various approaches to steal information from users and avoid being detected by the security systems in place. One technique includes registering malicious domain names related to COVID-19. The literature offers many approaches designed to detect malicious activities, some of which were described in the previous section. Most of them aim at detecting attacks from entire URLs. Although those methods can be effective, they are limited in the sense that knowledge of the URL is only obtained from the attack itself (usually by examining the phishing message). Detection of the malicious site prior to the attack is not possible if the URL is required. Moreover, since several malicious URLs can be generated from a single domain name, when a URL is flagged as malicious, an attacker can simply alter the combination and generate another URL with the same domain name. To avoid these limitations, we tailored our approach by only looking at the domain name. Domain names are a combination of:

- The Top-Level Domain (TLD) name, under which domain names can be registered by end users,
- The Second-Level Domain (SLD) name, which is the “name” registered by end users,
- The Sub-Domain(s), which can be added at will by SLD owners.

We limited our focus to domain names because once a domain name has been flagged as malicious, all the URL combinations from that domain name will be considered malicious since a TLD can only be set once, upon its creation. In seeking to increase the effectiveness of an attack and obtain more victims, attackers mainly use a combination of techniques, such as random characters, typo-squatting, etc., in the domain name to trick users into believing they are browsing a legitimate website. Because we only used publicly available information to detect malicious domain names, data protection regulations prevented us from accessing useful registration information that could be used for our purpose.

Therefore, it was difficult for us to find a high-quality, worldwide-accepted data set consisting entirely of verified malicious and legitimate domain names, or at least featuring low false positives and false negatives. Thus, an important challenge associated with using solely domain names is the amount of available data. However, developing an efficient approach that focuses solely on domain names is a strength of the proposed approach because it can be efficiently deployed in a DNS resolver, such as IBM Quad9, and thus protect users there without re-training when attackers start bypassing the current features. Nevertheless, their great advantage is that using new sets of classification algorithms and features can improve accuracy, making the scheme adaptable, (Varshney et al., 2016).

4. Data set and data preprocessing

In computer science, the quality of the output is determined by the quality of the input, as stated by George Fuechsle in the concept “Garbage in, Garbage out”. Therefore, not only was a good data set needed for the implementation and comparison of the models, but a set of useful features also had to be extracted from the data set.

\[
H(x) = - \sum_{i=0}^{n} p(x_i) \log_b p(x_i)
\]

4.1. Data set

We constructed our own labeled data set containing two classes of domain names: legitimate and malicious domain names. The malicious domain names came mainly from DomainTools and PhishLabs. DomainTools provides a publicly available list of COVID-19 related domain names, which is updated daily. On their list, each domain name is associated with a risk score of 70 or higher by DomainTools’ risk-scoring mechanisms. For this research, we only kept domain names with a risk score greater than or equal to 90 because these domain names were not manually checked; the higher the risk score, the more likely the domain names were to be confirmed as malicious. PhishLabs, on the other hand, provided intelligence on COVID-19 related attacks from March to September 2020, and the information collected on malicious domain names is available on their website. Though we aimed to utilize a publicly available list of confirmed legitimate domain names accepted worldwide, we were not able to find any. Therefore, we compiled our own list with the use of several search engines’ APIs. We used Google Custom Search, Yandex Search, and MagicBaidu Search API. First, a specific keyword list was built with the COVID-19-related keywords shown in Table 1. Then, the keywords were sent to the search APIs to obtain the first hundred results, which had a very low chance of being malicious pages. This phenomenon originates from the fact that web crawlers do not assign a high rank to malicious domain names because of their long lifetime (Le Page, 2019). Therefore, this collection process causes limitations in terms of legitimate domain names size, making the number of legitimate domains retrieved much smaller than the number of malicious domains. After these efforts, our data set consisted of 59,464 domain names: 5971 malicious domain names from the search APIs and 53,493 malicious domain names collected from DomainTools and PhishLabs. As we were given access to VirusTotal’s Academic API, we passed our data set through the API and kept only those domain names that were found in the VirusTotal database and were either flagged as malicious or not. We discarded those domain names that were not found in VirusTotal’s database. After cross-checking with VirusTotal, our data set consisted of 37,610 malicious domain names and 3904 legitimate domain names, totaling 41,514 domain names.

4.2. Feature extraction

After building the data set, extracting a set of relevant features was the next critical step. Korkmaz et al. (2020) and Buber et al. (2017) list and identify several features commonly used for phishing website detection. After a long discussion with our domain expert, we extracted a set of lexical features from the domain names in the data set. Initially, 12 features, divided into 9 groups, of which 7 can be found in the articles listed above, were extracted as shown in Table 2:

| Feature Name | Description |
|--------------|-------------|
| length       | Length of the string. |
| entropy      | Shannon's entropy produces an estimate of the average minimum number of bits needed to encode a string of symbols based on the alphabet size and the frequency of the symbols, as shown in (1), where \( p(x_i) \) divides the number of appearances of a character in the string by the length of the string. |
|...

7 https://publicsuffix.org/.
8 https://www.quad9.net/.
9 https://www.domaintools.com/resources/blog/.
10 https://www.phishlabs.com/covid-19-threat-intelligence/.
11 https://developers.google.com/custom-search.
12 https://www.microsoft.com/en-us/bing/apis/bing-web-search-api.
13 https://github.com/1049451037/MagicBaidu.
14 https://www.virustotal.com.
5. Experimental setup

For a huge data set, the process might take days; the domain lookups were not used in this project because fetching them would have required excessive time. We used the Tranco list generated on 06 February 2021, which contains 1 million list of domain names where domain names are most likely to be legitimate. We also implemented a description of malicious-domain-names-classification/). In addition, a description of the features is provided in the repository. In the κ architecture, we applied one algorithm, the Very Fast Decision Tree (VFDT), or Hoeffding Tree classifier (HT), which, as stated by the authors Domingos and Hulten (2000), finds the best attribute to test at a given node by only considering a small subset of the training examples that pass through that node. Thus, for a given stream of examples, the first ones are used to determine the root test. Once the root attribute is chosen, the succeeding examples are passed down to the corresponding leaves and used to choose the appropriate attributes there, and so on recursively.

### Table 2

| Name                       | Type     | Description                                    |
|---------------------------|----------|------------------------------------------------|
| Length                    | Numeric  | Two features to count the number of words and the number of characters in the domain name. |
| Containing “-”            | Numeric  | Checks whether the domain contains a hyphen.   |
| Entropy of domain         | Numeric  | Three features to calculate Shannon’s Entropy on different parts of the domain name.           |
| Tranco rank               | Numeric  | Checks whether a domain is on the Tranco (Le Pochat et al., 2019) top 1 million list of domain names where domain names are most likely to be legitimate. We used the Tranco list generated on 06 February 2021. |
| Ratio of the longest word | Numeric  | This feature matches the longest word that a domain name contains and normalizes it by dividing by the length of the domain name. |
| Typo-squatting            | Numeric  | Checks whether a domain name contains typos by comparing it to a list of misspelled domain names. We generate the list of misspelled domain names from a given list of keywords using dnstwist. The generated list includes misspelled domain names with bitsquatting, addition, insertion, homoglyphs and so forth. Examples of misspelled domain names generated from “covid.com” are shown in Table 3. |
| Freenom TLD               | Numeric  | Checks whether the top-level domain belongs to Freenom, ”.gq”, ”.cf”, ”.ml”, ”.tk” and ”.ga”. Most of these free domain names are used for malicious intent. |
| Numbers other than “19”   | Numeric  | Checks whether a domain contains numbers other than 19. Hao et al. (Hao et al., 2016) observe that malicious domain names are more likely to use numerical characters than legitimate ones, but since we were looking at “COVID-19” related domain names, we excluded the number “19”.
| Number of subdomain levels| Numeric  | Counts the number of subdomains in the domain name. For example, if a domain name is “a19.b18.example.com”, here the number of subdomains for “example.com” is two: “a19” and “a18”. |
| Label                     | Numeric  | The label of the domain, Class 0 constitutes the malicious domain names and class 1, the legitimate domain names. |

With respect to the “entropy” feature, we calculated three entropy values:

1. The entropy of the domain name with the subdomain and suffix.
2. The entropy of the domain name with the subdomain but without the suffix.
3. The entropy of only the domain name, i.e., excluding both the subdomain and suffix.

With the exception of the Tranco rank, other web-based features requiring domain lookups were not used in this project because fetching them would have required excessive time; the process might take days for a huge data set.  

### Table 3

Examples of misspelled domain names generated from the domain “covid.com”.

| Original | Addition | Bitsquatting | Homoglyph | Insertion | Repetition |
|----------|----------|--------------|-----------|-----------|------------|
| covid.com | covidh.com | sovid.com | covid.com | copvid.com | ecovid.com |

With respect to the “entropy” feature, we calculated three entropy values:

1. The entropy of the domain name with the subdomain and suffix.
2. The entropy of the domain name with the subdomain but without the suffix.
3. The entropy of only the domain name, i.e., excluding both the subdomain and suffix.

With the exception of the Tranco rank, other web-based features requiring domain lookups were not used in this project because fetching them would have required excessive time; the process might take days for a huge data set.  

5. Experimental setup

We applied two architectures to our data set: λ, also referred to as batch learning, where the model was treated as a static object, i.e., trained on the available data set and then making predictions on new data and retrained from scratch in order to learn from the new data, and κ, where the model learned as instances arrived, also referred to as online learning. Online learning differed from batch learning in that the model was trained, in addition to making predictions for new data, was also able to learn from it. We did not have a fixed data set but a data stream that was continuous and temporal-ordered. Fig. 2 shows the workflows of the different architectures. In the λ architecture, we applied 6 machine learning algorithms: a decision tree classifier (DTC), a random forest classifier (RFC), the gradient boosting algorithm (GBM), Extreme Gradient Boosting (XGBoost), a support vector machine (SVM), and a multilayer perceptron (MLP). With the exception of XGBoost, which was developed in C++ and downloaded separately to use in the Python Jupyter NoteBook, all the algorithms were implemented in the scikit-learn library in Python. The data set required for conducting experiments is available, with a licence that allows free usage for research purposes in the interest of reproducibility, in our GitHub repository (https://github.com/womega/covid19-malicious-domain-names-classification/). In addition, a description of the features is provided in the repository. In the κ architecture, we applied one algorithm, the Very Fast Decision Tree (VFDT), or Hoeffding Tree classifier (HT), which, as stated by the authors Domingos and Hulten (2000), finds the best attribute to test at a given node by only considering a small subset of the training examples that pass through that node. Thus, for a given stream of examples, the first ones are used to determine the root test. Once the root attribute is chosen, the succeeding examples are passed down to the corresponding leaves and used to choose the appropriate attributes there, and so on recursively. Hoeffding trees solve the difficult problem of deciding exactly how many examples are necessary at each node by using the Hoeffding bound (HB, or additive Chernoff bound) (Hoeffding, 1963; Maron & Moore, 1993). Consider a real-valued random variable v whose range is R (e.g., for a probability, the range is one, and for an information gain, the range is log 2, where c is the number of classes). Suppose n independent observations of this variable have been made and their mean is v. The HB states that, with probability 1 − δ, the true mean of

15 https://scikit-multiflow.readthedocs.io/.
the variable is at least \( \tilde{r} - \epsilon \), where:

\[
\epsilon = \frac{\sqrt{R^2 \ln \left(\frac{1}{\delta}\right)}}{2n}
\]  

Eq. (2) shows that the HB is independent of the probability distribution generating the observations. Our choice of HTs was motivated by the fact that not only did they quickly achieve high accuracy with a small sample, but they were also incremental and did not perform multiple scans on the same data and therefore resolved the time and computational resources limitations of MLB approaches. We used the Hoeffding Tree classifier developed in scikit-multiflow. The major drawback of the HT is that it cannot handle changes in the data distribution, i.e., concept drift. The Decision Tree Classifier (DTC) is a powerful algorithm used for supervised learning, as it aids in selecting appropriate features for splitting the tree into subparts and finally identifying the target item. At the bottom of the tree, each leaf is assigned to a class. The DTC can be used for classification as well as regression. The attributes of the samples are assigned to each node, and each branch’s value corresponds to the attributes, (Flach, 2012). RFs are ensemble algorithms developed by Breiman (2001). RFs attain high accuracy, are able to handle outliers and noise in the data, and are powerful in the sense that the decision comes from the aggregate of all the decisions of the trees in the forest. Thus, they are less prone to over-fitting, and their accuracy is relatively high. The weaknesses of RFs include the following: When used for regression tasks, they have difficulty in predicting beyond the range in the training data, and they may over-fit data sets that are particularly noisy, Benyamin (2012). GBM (Friedman, 2001, 2002) is another ensemble method that uses decision trees as weak learners. The main objective is to minimize the loss function, or the difference between the actual class value of the training example and the predicted class value, using first order derivatives. XGBoost (Chen & Guestrin, 2016) is another boosting method that builds decision trees sequentially. It is a refined, customized version of a gradient boosting decision tree system that also utilizes the second derivatives and was created for performance, speed, and pushing the limit of what is possible for gradient boosting algorithms. Among the algorithm’s features, the regularized boosting that prevents overfitting, the ability to handle missing values automatically, and the tree pruning, which results in optimized trees, make the algorithm very powerful. GBM and XGBoost’s main weakness is that they are more likely to overfit the data due to those decision boundaries. Nevertheless, they often construct highly accurate models. The Support Vector Machine (SVM) (Vapnik, 1998), an important algorithm in machine learning, is a geometric model that finds hyperplanes to create boundaries between classes. SVM works well in high-dimensional spaces and when the classes are separated by a clear margin but does not perform well with large and noisy data sets. MLP is a class of feed-forward ANN consisting of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function to map the weighted inputs to the output; two common activation functions are sigmoid (\( \sigma \)) and hyperbolic tangent (\( \text{tanh} \)), shown in (3)-(4). MLP uses the backpropagation supervised learning technique for training, Rumelhart et al. (1988), Van Der Malburg (1986), a generalization of the least mean squares algorithm in the linear perceptron. The degree of error in an output node \( j \) at the \( n \)th sample is represented by \( e_j(n) = d_j(n) - y_j(n) \), where \( d \) is the target value and \( y \) is the value predicted by the perceptron. The node’s weights are then adjusted based on corrections minimizing the error of the output, given by (5); then, using gradient descent, the change in weight \( \Delta w_{ij}(n) \) is defined by (6), where \( y_j \) is the output of the previous neuron and \( \eta \) is the learning rate selected to ensure a quick convergence of the weights to a desired response without oscillations.

\[
\text{tanh}(v) = \frac{e^v}{\sum e^v}
\]  

\[
\sigma(v) = \frac{1}{1 + e^{-v}}
\]  

\[
e(n) = \frac{1}{2} \sum e_j^2(n)
\]  

\[
\Delta w_{ij}(n) = -\eta \frac{\partial e_j(n)}{\partial w_{ij}(n)} y_i(n)
\]

Neural networks are able to learn the function that maps the input to the output without it being explicitly provided and can also handle noisy data and missing values. However, their key disadvantages are that the answer that emerges from a neural network’s weights is difficult to understand, i.e., a black box. In addition, training often takes longer than it does for their competitors. In the \( \lambda \) architecture, we conducted a 10-fold cross-validation to assess the overall performance of the algorithms on the data set. For the online learning approach (\( \kappa \) architecture), in contrast, we trained the HT classifier with streams of different sizes and observed the performance. In total, we generated 6 streams:

1. 10:90 with subdomain level feature (ST1): the data stream consisted of 10% legitimate domain names and 90% malicious domain names, with the “subdomain level” feature.
2. 10:90 without subdomain level feature (ST2): the data stream consisted of 10% legitimate domain names and 90% malicious domain names, without the “subdomain level” feature.
3. 20:80 with subdomain level feature (ST3): the data stream consisted of 20% legitimate domain names and 80% malicious domain names, with the “subdomain level” feature.
4. 20:80 without subdomain level feature (ST4): the data stream consisted of 20% legitimate domain names and 80% malicious domain names, without the “subdomain level” feature; in other words, this was the same data set used in the batch learning architecture.
5. 50:50 with subdomain level feature (ST5): the data stream consisted of 50% legitimate domain names and 50% malicious domain names, with the “subdomain level” feature.
6. 50:50 without subdomain level feature (ST6): the data stream consisted of 50% legitimate domain names and 50% malicious domain names, without the “subdomain level” feature.
Experiments were carried out on a MacBook Pro (13-inch, 2017) with a 2.3 GHz Dual Core Intel Core i5 processor and 8 GB 2133MHz LPDDR3 RAM. Each test was executed with the different algorithms trained with the corresponding training data set and the default hyperparameters implementation.

6. Experimental results

Frequently, the performance of a model is evaluated by constructing a confusion matrix. From the values of the confusion matrix, metrics such as the accuracy, the F-1 score, the specificity (True Negative rate, recall of the negative class), the sensitivity (True Positive Rate, recall of the positive class), precision, and so forth can be calculated to evaluate the efficiency of the algorithms. Equations for calculating the metrics are shown in (7)–(11), where TP represents the true positives, the domain names predicted as legitimate that were truly legitimate, TN the true negatives, the domain names predicted as malicious that were truly malicious, FP the false positives, the domain names predicted as legitimate but were in fact malicious, and FN the false negatives, the domain names predicted as malicious but were in fact legitimate, as produced by the algorithms.

\[
\text{Sensitivity}(TPR) = \frac{TP}{TP + FN} \quad (7)
\]

\[
\text{Specificity}(TNR) = \frac{TN}{TN + FP} \quad (8)
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)
\]

6.1. Data set rebalancing

As mentioned in the previous sections, our data set was highly imbalanced; because the malicious class contained about 9 times more domain names than the minority class, we needed to resample the data set to change the class distribution and make it more balanced. Several sampling techniques have been proposed: some that add more samples in the minority class (oversampling), others that remove samples from the majority class (undersampling), and yet others that employ both oversampling and undersampling. Each of these approaches has associated advantages as well as disadvantages. Oversampling, by increasing the number of samples in the minority class, increases the chances of overfitting along with the learning time, as it makes the data set larger. In contrast, the main disadvantage of undersampling is that it removes data that can be useful. As we were in a time-sensitive environment and oversampling would increase the learning time, we defaulted to undersampling. Several undersampling methods have been proposed, including methods that select examples to keep (e.g., Near-Miss (NM), (Zhang & Mani, 2003), Condensed Nearest Neighbor (CNN), (Hart, 1968)), others that select which examples to delete (e.g., Tomek Links (TL) undersampling, (Tomek, 1976), Edited Nearest Neighbors (ENN), (Wilson, 1972)), and still others that do both (e.g., (Fernández, 2018), (Laurikkala, 2001)). Because we did not have enough training samples in the minority class, undersampling the majority class would have led to the loss of too much information and having a non-representative data set. On the other hand, the steps taken to compile the list of legitimate domain names detailed in Section 4.1., shows how difficult and time consuming the collection of more legitimate domain names is. Therefore, we aimed at rebalancing the data set to a ratio of 20:80. Table 4 shows the mean scores of a DTC trained on our data set that was resampled using the methods described above. For the comparison shown in Table 4, we set the number of neighbors parameter for NM, CNN and ENN to 3; as the CNN, ENN and TL methods did not allow us to explicitly set the ratio of the number of samples to retain in the majority class, only the NM was set to balance the data set with a 20:80 ratio. As the table shows, CNN took more time to resample the data set, removed too many samples from the majority and yielded poor results compared to the others. In contrast, ENN achieved the best results, but it took slightly longer than the NM to resample and did not achieve our desired ratio. We therefore defaulted to the NM, as it reached our sampling goal in a short amount of time while still yielding good results. Fig. 3 shows the approximate data distribution before and after undersampling the majority (malicious) class using NM.

6.2. \( \lambda \) architecture

In the \( \lambda \) architecture, we tested our models with and without the “Subdomain levels” feature, as most of the legitimate domain names had at least one subdomain. Moreover, we observed the feature had more importance than all the others after training and testing. Table 5 shows the average fit time, F-1, precision, Accuracy, TPR and TNR of the different classifiers after 10-fold cross-validation without the feature, while Table 6 shows those scores with the feature. One can observe from Tables 5 & 6 that overall, the DTC, RFC, and MLP yielded better scores when the feature was removed, and the situation was reversed for the XGBoost, GBM, and SVM. This outcome shows that the “Subdomain Levels” feature can have either positive or negative impacts, depending on the model. Therefore, to avoid over-fitting, we have focused more on the scores without the feature in this study, i.e., Table 5. Table 5 demonstrates that the DTC was the fastest to train (0.0667 s) and the MLP was the slowest (13.1045 s) per fold, as stated in the previous section, but yielded the highest F-1 (91.4848) and Accuracy (96.6445) on average. The SVM yielded the highest average Precision (95.5949) and Specificity (99.0011), while the RFC yielded the highest Sensitivity (TPR = 90.9333). Fig. 4 illustrates the Receiver Operating Characteristic (ROC) curves and mean Area Under Curve (AUC) after 10-fold cross-validation without the “Subdomain Levels” feature. The DTC yielded the lowest AUC (0.9198) with the highest standard deviation (0.0247), and the XGBoost yielded the highest AUC (0.9848) with a 0.0072 standard deviation. We further performed

---

**Table 4**

Different undersampling techniques.

| Method   | Ratio  | Fit-time (s) | G-Mean | F-1  | ROC  | AUC  |
|----------|--------|--------------|--------|------|------|------|
| NM       | 20:80  | 0.391        | 0.930  | 0.892| 0.935|
| CNN      | 65:35  | 2164.10      | 0.830  | 0.884| 0.836|
| ENN      | 10:90  | 0.961        | 0.956  | 0.921| 0.957|
| TL       | 10:90  | 0.897        | 0.933  | 0.880| 0.937|

**Table 5**

Average scores of classifiers without “Subdomain Levels” feature.

| Classifier | Fit time | F-1 | Precision | Accuracy | TPR | TNR |
|------------|----------|-----|-----------|----------|-----|-----|
| DTC        | 0.0667   | 85.9324 | 83.8446 | 93.8064 | 89.2683 | 94.9518 |
| RFC        | 1.3252   | 91.1245 | 91.7768 | 96.3627 | 90.9333 | 97.7206 |
| GBM        | 1.5327   | 90.0024 | 94.3979 | 96.1680 | 86.1179 | 98.6809 |
| XGBoost    | 0.9023   | 91.2849 | 92.1589 | 96.4447 | 90.8309 | 97.8487 |
| SVM        | 1.2799   | 90.2950 | 95.5949 | 96.3217 | 85.6049 | 99.0011 |
| MLP        | 13.1045  | 91.4848 | 95.8070 | 96.6445 | 89.4470 | 98.4441 |

**Table 6**

Average scores of classifiers with “Subdomain Levels” feature.

| Classifier | Fit time | F-1 | Precision | Accuracy | TPR | TNR |
|------------|----------|-----|-----------|----------|-----|-----|
| DTC        | 0.056535 | 84.8376 | 81.3734 | 93.1096 | 90.0364 | 93.8779 |
| RFC        | 1.170378 | 91.0747 | 90.3244 | 96.2602 | 92.3422 | 97.2400 |
| GBM        | 1.344766 | 92.3138 | 95.3655 | 97.0184 | 89.4950 | 98.8986 |
| XGBoost    | 0.889119 | 91.6890 | 91.8256 | 95.5984 | 91.8810 | 97.7780 |
| SVM        | 1.335580 | 91.9326 | 95.8724 | 96.8955 | 88.3454 | 99.0330 |
| MLP        | 9.574650 | 91.3232 | 92.5737 | 96.5318 | 90.3187 | 98.0853 |

---

7
the paired Student’s t-test (Vapnik, 1998), a statistical hypothesis test developed by William Sealy Gosset in 1908, to compare the models’ performance and determine whether there was a significant difference between the models. Table 7 shows the p-values for each pair of models calculated by means of the t-distribution with \(k - 1 = 9\) degrees of freedom. Observably, the SVM shows a significant difference with all the algorithms, except the DTC, at the \(\alpha = 0.05\) significance level. Thus, the null hypothesis is rejected in those cases (in bold in Table 7).

### 6.3. \(\kappa\) architecture

The Hoeffding Tree (HT) classifier was used in the \(\kappa\) architecture for the classification task. As with the \(\lambda\) architecture, we tested the model with and without the “Subdomain levels” features. In real data streams, the number of examples for each class might be changing and evolving. Therefore, the accuracy score is only useful when all classes have the same number of instances. \(\kappa\) is a more delicate measure for evaluating how streaming classifiers perform. The \(\kappa\) statistic was introduced by Cohen (1960) and is defined by:

\[
k = (p_0 - p_c)/(1 - p_c)
\]  

(12)

The quantity \(p_0\) is the classifier’s prequential accuracy, and \(p_c\) is the probability of randomly guessing a correct prediction. If the classifier is always correct, then \(\kappa = 1\). If the classifier’s predictions are similar to random guessing, then \(\kappa = 0\). Accordingly, we observed the value of \(\kappa\) for every test in addition to the accuracy, precision, recall, and Geometric Mean (G-Mean), measuring how balanced the classification performances were on both the majority and minority classes, defined by (13).

\[
G - \text{Mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}}
\]  

(13)

Table 8 presents the results obtained on each of these streams, all with a pre-trained size of 200 samples. The results show a mean prequential accuracy higher than 90 for all the streams, with ST1 having the highest accuracy (acc = 97.93) and ST2 coming second (acc = 97.15). In the case of mean \(\kappa\), the best performance was achieved on ST3 (\(\kappa = 90.06\)), with ST1 coming second (\(\kappa = 97.39\)). ST5 yielded the highest mean precision (precision = 97.39), with ST2 being the only one yielding a precision less than 90. In terms of mean recall and F-1, the best results were achieved on ST5 (recall = 89.85, F-1 = 93.47). Meanwhile, the highest G-Mean was also achieved on ST5 (G-Mean = 93.64), with ST2 again scoring below 90. According to the table, the “subdomain levels” feature highly influenced the results of models trained with
streams containing the same level of data distribution, i.e., ST1–ST2, ST3–ST4, and ST5–ST6. This outcome shows that even when instances were learned one at a time, the feature played a critical role in correctly predicting labels.

6.4. Analysis and discussion

In the batch learning method (s), the XGBoost algorithm outperformed the other five models (DTC, RFC, GBM, SVM and MLP) because it achieved relatively high and consistent scores in a short amount of time on the data set after 10-fold cross-validation, unlike the other models that seemed to focus on improving one metric at the expense of the others. Regarding the online learning method (s), after testing the HT Classifier with streams of different levels of data distribution and feature sets, we were able to conclude that the stream ST5 was the best, as the model achieved the highest mean recall, F-1, and precision, without to the other data streams. Comparing the XGBoost to HTs, without the “subdomain level” feature, reveals that the XGBoost performed better than HTs on streams ST2, ST4, and ST6, although ST2 yielded better accuracy than the XGBoost. This outcome shows that the HT’s accuracy kept improving with each sample it trained from at the expense of the other metrics. When the “subdomain level” feature was kept, however, ST3 and ST5 outperformed XGBoost in terms of F-1, precision, and ST1 and ST3 outperformed XGBoost in terms of accuracy. Contrariwise, XGBoost outperformed the rest (ST1, ST3, and ST5) in terms of mean Recall. This result indicates that the data distribution had an impact on the HT, in addition to the “subdomain level” feature, which also impacted the XGBoost’s scores.

The results obtained show that we can effectively detect malicious domain names associated to COVID-19. This is a significant outcome given the overwhelming panic caused by the pandemic, which left many individuals vulnerable.

7. Conclusion and future works

In this paper, we applied two learning methods, batch learning and online learning, to classify COVID-19-related domain names in our data set. The data included domain names from publicly available sources, comprised of our list of malicious domain names containing domain names from DomainTools and PhishLabs and our list of legitimate domain names with domain names from search engines. This study’s main contribution is its provision of an effective, fast-performing detection scheme to detect malicious domain names related to COVID-19. Indeed, our feature set consisted mainly of lexical features, easily and quickly extracted from the data we already had and did not require domain lookups, which would slow down the feature extraction steps; moreover, the algorithms we used attained relatively high scores. In addition, we have shown how one feature, namely the number of subdomain levels, can highly influence the performance of the different classifiers in both the batch and online learning frameworks. In the future, we plan on investigating effective feature sets for detecting domain names related to other attack campaigns as misspelled domain names can be generated from other keywords, not only those related to COVID-19. An additional area of interest would be looking at other parts of the URL, such as paths, query strings and fragments, for detection. In the batch learning approach, future works will involve tuning the algorithms for the best hyperparameters and selecting the model suiting our needs to make predictions on unseen data. Those predictions will then be compared to those made by the Hoeffding Trees pre-trained with different levels of data distributions by using such statistics as the inter-rater reliability, r, to see how the classifiers agree in the predictions made for each instance. Second, those predictions will be compared with labels from VirusTotal, a source that we consider reliable for this purpose, to properly evaluate each model. Finally, the best model can then be deployed as a browser add-on or plug-in to warn users that they are about to visit a malicious domain and flag those domain names in real-time.

CRediT authorship contribution statement

Paul K. Mvula: Conceptualization, Methodology, Software, Visualization, Writing – original draft. Paula Branco: Conceptualization, Supervision, Validation, Writing – review & editing. Guy-Vincent Jourdan: Conceptualization, Supervision, Validation, Writing – review & editing. Herna L. Viktor: Conceptualization, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the anonymous reviewers and the editor for their constructive comments and suggestions. We are also grateful to VirusTotal for access to their service and ZETAlitics, DomainTools, and PhishLabs for access to their data sets. This research was supported by the Natural Sciences and Engineering Research Council of Canada, the Vector Institute, and The IBM Center for Advanced Studies (CAS) Canada within Project 1059.

References

Abdelnabi, S., Krombholz, K., & Fritz, M. (2020). VisualPhishNet: Zero-day phishing website detection by visual similarity. (pp. 1681–1698). http://dx.doi.org/10.1145/3372973.3373235.

Almoman, A. (2013). Phishing dynamic evolving neural fuzzy framework for online detection “zero-day” phishing email. Indian Journal of Science and Technology, 6(1), 1–5. http://dx.doi.org/10.17485/ijst/2013/v6i11.18.

Basit, A. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques. Telecommunication Systems, 76(1), 139–154. http://dx.doi.org/10.1007/s11235-020-00735-2.

Benyamin, D. (2012). A gentle introduction to random forests, ensembles, and performance metrics in a commercial system. URL https://blog.citizenenet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146. http://dx.doi.org/10.1162/tacl_a_00065.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. http://dx.doi.org/10.1023/A:1010933404324.

Buber, E., Demir, O., & Sahingoz, O. K. (2017). Feature selections for the machine learning based detection of phishing websites. In 2017 International artificial intelligence and data processing symposium (pp. 1–5). http://dx.doi.org/10.1109/IDAP.2017.8090317.

Cao, Y., Han, W., & Le, Y. (2008). Anti-phishing based on automated individual white-list. In D.IM ’08: Proceedings of the 4th ACM workshop on digital identity management (pp. 51–66). Association for Computing Machinery, http://dx.doi.org/10.1145/1456424.1456434.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In KDD ’16: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785–794). Association for Computing Machinery, http://dx.doi.org/10.1145/2939672.2939785.

Chiew, K. L., Chang, E. H., Sze, S. N., & Tiong, W. K. (2015). Utilisation of website logo for phishing detection. Computers & Security, 54, 16–26. http://dx.doi.org/10.1016/j.cose.2015.07.006.

Cohn, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), 37–46.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining (pp. 71–80). ACM Press, http://dx.doi.org/10.1145/347068.347107.

Dunlop, M., Groat, S., & Shelly, D. (2010). GoldPhish: Using images for content-based phishing analysis. In 2010 Fifth international conference on internet monitoring and protection (pp. 123–128). http://dx.doi.org/10.1109/ICIMP.2010.24.

Fernández (2018). Learning from imbalanced data sets (1st ed. 2018 edition). Springer.

Flach, P. (2012). Machine learning: The art and science of algorithms that make sense of data (p. 416). Cambridge University Press.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189–1232. URL https://www.jstor.org/stable/2695986.

Friedman, J. (2002). Stochastic gradient boosting. Computational Statistics & Data Analysis, 38, 367–378. http://dx.doi.org/10.1016/S0167-9473(01)00065-2.
Han, W., Cao, Y., Bertino, E., & Yong, J. (2012). Using automated individual white-list to protect web digital identities. Expert Systems with Applications, 39(15), 11861–11869. http://dx.doi.org/10.1016/j.eswa.2012.02.020.

Hao, S., Kantchelian, A., Miller, B., Paxson, V., & Feamster, N. (2016). PREDATOR: Proactive recognition and elimination of domain abuse at time-of-registration. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 1568–1579). ACM, http://dx.doi.org/10.1145/2976749.2978317.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). IEEE Transactions on Information Theory, 14(3), 515–516. http://dx.doi.org/10.1109/TIT.1968.1054155, Conference Name: IEEE Transactions on Information Theory.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301), 13–30. http://dx.doi.org/10.2307/2282852.

Jain, A. K., & Gupta, B. B. (2017). Phishing detection: Analysis of visual similarity based approaches. Security and Communication Networks, 2017, 1–20. http://dx.doi.org/10.1155/2017/5421046.

Jain, A. K., & Gupta, B. B. (2018). Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems, 68(4), 687–700. http://dx.doi.org/10.1007/s11235-017-0414-0.

Korkmaz, M., Sahingoz, O. K., & Diri, B. (2020). Feature selections for the classification of webpages to detect phishing attacks: A survey. In 2020 International congress on human-computer interaction, optimisation and robotic applications (pp. 1–9). http://dx.doi.org/10.1109/HORA49412.2020.9152934.

Laurikila, J. (2001). Improving identification of difficult small classes by balancing class distribution. In S. Quaglini, P. Barahona, & S. Andreassen (Eds.), Lecture notes in computer Science, Artificial intelligence in medicine (pp. 63–66). Springer, http://dx.doi.org/10.1007/3-540-48229-6_9.

Le Page, S. (2019). Understanding the phishing ecosystem (Ph.D. thesis), Université d’Ottawa / Università di Ottawa / University of Ottawa, http://dx.doi.org/10.20381/ruor-23629, URL http://ruor.uottawa.ca/handle/10393/30385, Accepted: 2019-07-08T15:12:34Z.

Le Pochat, V., Van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., & Joosen, W. (2019). Tranco: A research-oriented top sites ranking hardened against manipulation. In Proceedings 2019 network and distributed system security symposium. Internet Society, http://dx.doi.org/10.14722/ndss.2019.23386.

Maron, O., & Moore, A. (1993). Hoeffding races: Accelerating model selection search for classification and function approximation. Advances in Neural Information Processing Systems, 6, URL https://proceedings.neurips.cc/paper/1993/hash/02a32ad266f96e629e60767ce0e1a0-Abstract.html.

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Intelligent rule-based phishing websites classification. IET Information Security, 8, 8.

Roselli, A. P. E., Kirda, E., Kruegel, C., & Ferrandi, F. (2007). A layout-similarity-based approach for detecting phishing pages. In Proceedings of the 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops, SecureComm 2007 (pp. 454–463). http://dx.doi.org/10.1109/SECCOM.2007.4550367.

Rumelhart, D., Hinton, G., & Williams, R. (1988). Learning internal representations by error propagation. In Readings in cognitive science (pp. 399–421). Elsevier, http://dx.doi.org/10.1016/8978-1-4832-1446-7.50035-2.

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, 345–357. http://dx.doi.org/10.1016/j.eswa.2018.09.029.

Schneier, B. (2000). Secrets and lies: Digital security in a networked world. New York: John Wiley.

Tajaddodianfar, F., Stokes, J. W., & Gururajan, A. (2020). Tception: A character/word-level deep learning model for phishing URL detection. (pp. 2857–2861). http://dx.doi.org/10.1109/ICASSP40776.2020.9053670.

Tanaka, Y., & Kashima, S. (2019). SeedMiner: Accurate URL blacklist-generation based on efficient OSINT seed collection. (pp. 250–255). http://dx.doi.org/10.1145/3358695.3361751.

Tomek, I. (1976). Two modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(11), 769–772. http://dx.doi.org/10.1109/TSMC.1976.4309452, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

Van Der Maasburg, C. (1986). Frank rosenblatt: Principles of neurodynamics: Perceptions and the theory of brain mechanisms. In G. Palm, & A. Aertsen (Eds.), Brain theory (pp. 245–248). Springer, http://dx.doi.org/10.1007/978-3-642-70911-1_20.

Vapnik, V. N. (1998). Adaptive and learning systems for signal processing, communications, and control, Statistical learning theory. Wiley.

Varshney, G., Misra, M., & Atrey, P. K. (2016). A survey and classification of web phishing detection schemes. Security and Communication Networks, 9(18), 6266–6284. http://dx.doi.org/10.1002/sec.1674.

Verma, R., & Das, A. (2017). What’s in a URL: Fast feature extraction and malicious URL detection. In IWSPA ’17, Proceedings of the 3rd ACM on international workshop on security and privacy analytics (pp. 55–63). Association for Computing Machinery, http://dx.doi.org/10.1145/3041108.3041016.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics, SMC-2(3), 408–421. http://dx.doi.org/10.1109/TSMC.1972.4309137, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. ACM Transactions on Information and System Security, 14(2), 21:1–21:28. http://dx.doi.org/10.1145/2019599.2019606.

Zhang, Y., Hong, J. L., & Cranor, L. F. (2007). Cantina: A content-based approach to detecting phishing web sites. In WWW ’07, Proceedings of the 16th international conference on world wide web (pp. 639–648). Association for Computing Machinery, http://dx.doi.org/10.1145/1242572.1242659.

Zhang, J., & Mani, I. (2003). KNN approach to unbalanced data distributions: A case study involving information extraction. In Proceedings of the ICMCI2003 workshop on learning from imbalanced datasets.

Zhu, E., Ju, Y., Chen, Z., Liu, F., & Fang, X. (2020). DTOF-ANN: An artificial neural network phishing detection model based on decision tree and optimal features. (p. 14). http://dx.doi.org/10.1016/j.select.2020.106505.