Scientific African 16 (2022) e01165

Contents lists available at ScienceDirect

Scientific African

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

Performance evaluation of machine learning tools for detection of phishing attacks on web pages

T.O. Ojewumi[a], G.O. Ogunleye[b,*], B.O. Oguntunde[a], O. Folorunsho[b], S.G. Fashoto[c], N. Ogbu[a]

[a] *Department of Computer Science, Redeemer's University, Ede, Osun State, Nigeria*
[b] *Department of Computer Science, Federal University, Oye-Ekiti, Ekiti State, Nigeria*
[c] *Department of Computer Science, Faculty of Science and Engineering, University of Eswatini, Kwaluseni M201, Eswatini*

A R T I C L E   I N F O

Article history:
Received 3 October 2020
Revised 19 April 2021
Accepted 19 March 2022

Editor DR B Gyampoh

Keywords:
Phishing
Attack
KNN
Random Forest
SVM

A B S T R A C T

This paper analyses and implements a rule-based approach for phishing detection using the three machine learning models trained on a dataset consisting of fourteen (14) features. The machine learning algorithms are; k-Nearest Neighbor (KNN), Random Forest, and Support Vector Machine (SVM). Among the three algorithms used, it was discovered that Random Forest model proved to deliver the best performance.

Rules were extracted from the Random Forest Model and embedded into a Google chrome browser extension called PhishNet. PhishNet is built during the course of this research using web technologies such as HTML, CSS, and Javascript. As a result, PhishNet facilitates highly efficient phishing detection for the web.

Introduction

Over the years since its inception, the internet has continued to grow exponentially in use, as well as its various use-cases. One of the popular use-cases of the internet is internet banking.

Internet banking allows customers to engage in financial transactions with financial institutions or banks over the internet. Internet banking is widespread and on the rise all over the world.

In Europe, about 51% of the adult population engage in internet banking [1], and about 60% of the United States population engaged in internet banking last year according to the Statista Global Consumer Survey [2,3]. The E-commerce industry influenced retail sales to the tune of $2.84 trillion in 2018. Cybercriminals have since realized the wealth in the internet banking industry and have invented several means of attack, one of which is phishing [3].

A phishing attack is based on deceiving a person to navigate to a fake original-look-alike website where his/her sensitive details can be stolen. Phishing attacks are on the rise because of their profitability. There was a 36% increase in phishing attacks, and the number of phishing sites grew by 220% over the course of 2018 [4]. Some of these phishing attacks are aimed at employees of certain companies. Big companies such as Google and Facebook were no exemption as they were

[*] Corresponding author.
E-mail address: gabriel.ogunleye@fuoye.edu.ng (G.O. Ogunleye).

https://doi.org/10.1016/j.sciaf.2022.e01165

confirmed to have fallen victim and lost up to $100 million combined in 2017 [5]. Many phishing attacks are aimed at customers/users of a particular service.

A famous phishing attack was a Gmail phishing scam that targeted almost 1 billion Gmail users worldwide in 2017. Such an attack is usually aimed at account takeover. Phishing attacks aim to deceive users/clients of services into revealing sensitive information by disguising trustworthy institutions over the internet. This disguise involves the creation of fake websites, which are look-alikes of original reputable sites. Users are then made to navigate to these fake websites, usually with emails. These emails are made to appear to have come from the reputable institution and target customers of that institution. They usually contain a hyperlink with the correct URL as text, while pointing to the fake website. The hope of the sender of the email is that the target will click such a link and fail to check or recognize the difference in the URL; and also reveal sensitive information such as login or credit card details. These attacks can be in a personalized form known as Spear phishing. Phishing attacks also affect businesses/brands. Phishers usually make use of a legitimate brand for their attack.

Once the phishing campaign is discovered and revealed, these brands become casualties, as this kind of publicity can hurt the brand's image. Customers of such a brand might even avoid the brand's legitimate website for fear of landing on the fake web page. Organizations might also lose confidence in that brand and drop it for a competitor. Phishing emails can be received at any time. Sometimes, these types of email messages may be preceded by technology disasters or newsworthy events. This makes the email seem authentic and creates a sense of urgency in the recipient. It is essential to solve the problem of phishing attacks as phishing attacks cause financial loss for victims and put their information at risk. It also creates a loss for business parties and can be a death blow for businesses.

The proposed solution to the issue of phishing will yield a safer environment for web users and reduce the risk of being caught by a phishing attack by alerting users when they land on a phishing site. This paper compares and implements a rule-based approach for phishing detection using the three machine learning models that are popular for phishing detection. The machine learning algorithms are; k-Nearest Neighbor (KNN), Random Forest, and Support Vector Machine (SVM). The models were trained on a dataset consisting of fourteen (14) features. Among the three algorithms used, it was discovered that Random Forest model proved to deliver the best performance.

Rules were extracted from the Random Forest Model and embedded into a Google chrome browser extension called PhishNet. PhishNet is built during the course of this research using web technologies such as HTML, CSS, and JavaScript. As a result, PhishNet facilitates highly efficient phishing detection for the web.

In this paper, we try to answer the following research questions:

1. Which of the three machine learning models that are being used for phishing detection would perform best?
2. How good is the rule-based machine learning model in predicting phishing websites?

Review of related literature

Mohigimi and Varjani [6] proposed a standard based phishing detection strategy. They proposed two new capabilities, which included making use of approximate string-matching algorithms and used relevant features from related works. They utilized the Support Vector Machine algorithm for training their model and later extracted the rules using SVM_DT calculation. Their proposed model had a True Positive Rate of 99.14% in the detection of web banking phishing site pages. The extracted rules were then implemented as a chrome browser extension, which they named PhishDetector for convenience. Their solution is autonomous of third-party services and can identify zero-day phishing attacks. The mishap of their proposed model is that it is excessively subject to page content and can't identify website page with non-HTML code [11]. Tan et al. [12] proposed a phishing detection technique dependent on character examination between the real and target website page. Their methodology named PhishWHO was in three phases;

- Keywords extraction: the extraction of identity keywords from the textual components of a website where a novel weighted URL tokens system based on the N-gram model was proposed.
- Target domain selection: the use of a search engine to find target domains which are then selected based on identity-relevant features.
- Identity matching: a three-tier identity matching system was proposed to determine the legitimacy of the query webpage.

PhishWHO accomplished a true positive rate of 99.68% and a true negative rate of 92.52%. PhishWHO doesn't perform well against the visual cloning procedure utilized by some phishers. Varshney et al. [13] utilized a search engine based anti phishing approach for phishing detection. They built up a lightweight phish detector as a Google Chrome extension augmented with Google as the search engine and called it Lightweight Phish Detector (LPD). They performed extensive testing and compared their solution to other web index-based antiphishing approaches in terms of performance. They used URLs from PhishTank and Alexa ranking for their examination and got a variable result of 92.4% to 100% for true negative rate and 99.5% for true positive rate. LPD was seen as proficient as far as calculation and capacity cost as it was estimated at 47 kb and utilized just 11mb worth of memory. It didn't require any dedicated server resources and could run on the client-side of the browser since it was lightweight. Unfortunately, LPD could not deal with languages other than English so accurately and yielded a couple of false positives [14].

Ali [4] proposed a phishing detection strategy utilizing different machine learning classification algorithms combined with wrapper feature selection techniques. The machine learning classifiers used incorporate Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Random Forest (RF), and Naïve Bayes (NB), among others. He had the ability to increase the accuracy of the model's wrapper feature selection and achieved a 97.3% True Positive Rate and a True Negative Rate of 97% with the Random Forest classifier on the detection of phishing sites. Wrapper selection feature requires extra computational overhead, thus wasting additional time.

Abutair et al. [10] proposed a Case-Based Reasoning phishing detection system (CBR-PDS). It is a system that recognizes phishing attacks by depending on past cases and uses a small dataset contrasted with machine learning approaches. Their solution utilized a two-phase feature selection and weighting procedure to choose informative features and assign them weights out of twenty-one initial features. Their proposed model delivered a 96% precision on their test dataset. The proposed model is versatile to new phishing strategies; in any case, it requires constant updating with new phishing URLs.

Yi et al. [15] proposed a phishing detection strategy that utilizes Deep Learning Frameworks. They grouped their extracted features into two types; original features and interactive features and then introduced a detection model dependent on Deep Belief Networks (DBN). Their solutiondelivered roughly 90% True Positive Rate and 99.4% True Negative rate. In any case, their model can end up being computationally costly.

Sahingoz et al. [8,9] proposed a model for detecting phishing pages using machine learning algorithms. They utilized Artificial Neural Networks (ANNs) and Deep Neural Networks (DNNs) machine learning algorithms and trained their models with a dataset containing around 60,000 site pages with 27 extracted features. Their solution achieved an accuracy of 92% for the ANN approach and 96% for the DNN approach. In any case, the execution time of their framework can, at present, be enhanced.

Muppavarapu et al. [7] proposed a novel strategy for phishing location utilizing Resource Description Framework (RDF) models and Random Forest classification algorithm. Their model included making decisions dependent on a search engine and using a Random Forest model trained with 21 features as a contingency for decision making should the main methodology fizzle. They executed their proposed model mostly with Java programming language and utilized Google as their search engine. The proposed strategy was effective as they accomplished a True Positive Rate of 98.8% and a True Negative Rate of 98.5%.

Mishra and Gupta [5] proposed a phishing detection system that included utilizing URI and CSS matching techniques with the assumption that phishers, by and large, utilize the same CSS style as the original website page. They executed their solutionas a desktop application called CUMP with Java programming language. Their proposed solution had a True Positive Rate of 93.27% and a True Negative Rate of 100%. The significant quality of their solution was its False Positive Rate, which was low to zero and furthermore it's capacity to deal with a broad scope of sites. Be that as it may, their solution has a high memory cost and fails when a CSS document has an enormous number of CSS rules.

Sahingoz et al. [9] proposed a real-time antiphishing framework that utilizes seven machine learning classification algorithms, and Natural Language Processing (NLP) based highlights. Their framework is autonomous of third-party services and can deal with new phishing sites. The classification algorithms utilized are Naïve Bayes, k-NN, Adaboost, Decision Tree, Random Forest, SMO, and K-star. Their best result was given by the Random Forest algorithm, which delivered an accuracy of 97.98%. Their proposed framework can perform well with various scope of sites as the models were prepared with a big dataset containing data that met multiple criteria. The framework does not perform as adequately when the URL contains short sub-domains and domains without any path. This is because of the use of NLP based highlights.

This paper aims to implement a solution that will make decisions based on the best performing machine learning classification model for phishing detection using a lightweight Google chrome extension, which will offer convenient and highly accurate phish detection for internet users.

Methodology

This section focuses on and analyzes the Extracted Rule-based method for phishing detection and how it can be implemented. In this paper, a total of 14 features were extracted from websites. Phishing websites were gotten from PhishTank (PhishTank.com is a community-based phish verification system where suspicious sites are submitted to be voted on by other users as a means of phish detection and verification), legitimate Internet-banking websites are gotten from various webpage directory services. The following outline methods were used to achieve this paper.

- Use of Supervised machine learning algorithms to train models:

  Support Vector Machine (SVM), Random Forest, and k-Nearest Neighbor (k-NN) classification algorithms were used to train three separate models. These algorithms were accessed from Python's Scikit-learn library.

- Assessment, evaluation, and comparison of the performance of trained models in the classification of test examples.
- Extraction of rules from the best model, selection, and evaluation of rules:

  A synthetic dataset based on the best model's correct predictions were used to train a Decision Tree model from which if-else decision rules can then be extracted.

- Embedding the selected rules into a browser extension called PhishNet for the detection of phishing webpages:

In this research paper, PhishNet was created as a Chrome browser extension in the form of content script using JavaScript, HTML, and CSS.

The steps taken to build the proposed system are as follows:

- Data collection
- Feature Extraction
- Model building and training
- Model assessment
- Rule Extraction
- Building of PhishNet

Fig. 1 illustrates the system methodology using a Block Diagram.

Data collection

This is the first phase of the project which involved the collection of phishing and legitimate webpage URLs which are required for feature extraction. A total of 1000 phishing webpages were gotten from Phishtank (www.phishtank.com). Four hundred legitimate webpages, which are internet banking and financial webpages were obtained from web directories like Jasmine directory, business, the Financial brand, Intechnic, and Similar web directories.

This phase was carried out with the aid of web scraping tools that access the DOM of webpages and extract the needed information.

Feature extraction

Collected data were represented in the form of features for training the machine learning models. A total of 14 features were extracted from the gathered webpages, six (6) of which were gotten from the webpage URL while the remaining eight from the DOM of the webpage. The features that were used are as follows:

IP address

Some phishing webpages mask their website name by representing the website in the form of IP address, e.g., "http://125.94.3.135/site.html". Another reason phishers use IP addresses is to avoid paying for a domain name and hosting. Feature 1 can be represented as follows:

$$\boldsymbol{f}1 = \begin{cases} 1, & IP\ address \\ 0, & not\ IP\ address \end{cases}$$

SSL security

SSL security is an important feature for identifying phishing webpages as phishers are more likely to use websites without SSL security in order to avoid extra costs. Use of SSL security reflects in the URL of websites as an extra 's' in the HyperText Transfer Protocol ('https'). Feature 2 can be extracted using string manipulation and is as follows:

$$f2 = \begin{cases} 1, & SSL\ security \\ 0, & no\ SSL\ security \end{cases}$$

Number of dots

The number of dots in the webpage URL is also another essential feature that can be used for classifying webpages. This is because some phishing webpages pretend to be legitimate by using popular legitimate domains as subdomains of their site. This, in turn, increases the number of dots in the URL webpage. Therefore, URLs containing more dots are more likely to be phishing webpages. Feature 3 is a count feature and can be represented as follows:

$$f3 = Number\ of\ dots$$

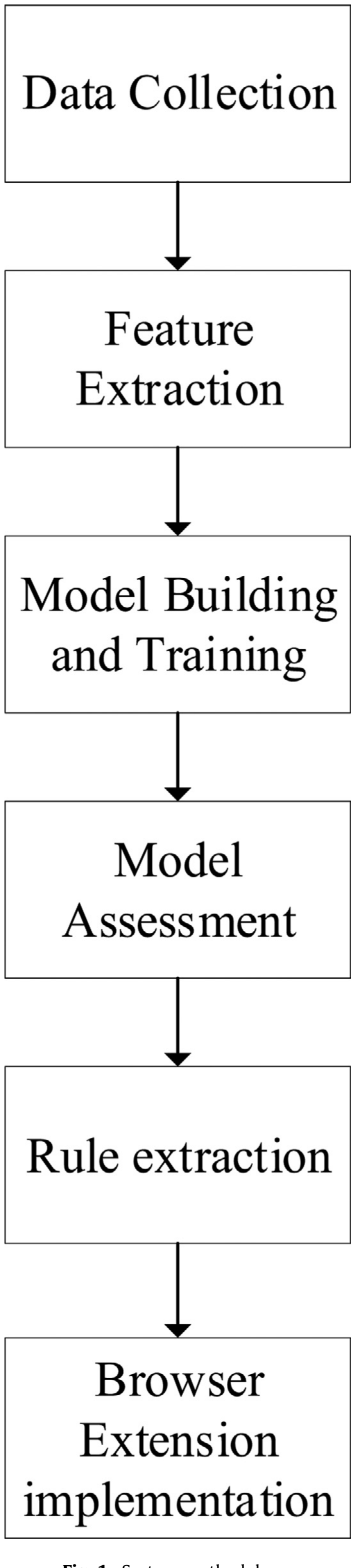


Fig. 1. System methodology.

Length of URL

Phishing webpages tend to have longer URLs as phishers sometimes try to mask suspicious URLs with very long URLs and legitimate-looking subdomains. For extraction purposes the URL will be broken down into three parts; the host, file path and query with three corresponding features for their lengths as follows:

$$\boldsymbol{f}4 = \boldsymbol{Length\ of\ host}$$

$$\boldsymbol{f}5 = \boldsymbol{Length\ of\ query}$$

$$\boldsymbol{f}6 = \boldsymbol{Length\ of\ the\ file\ path}$$

Resource domain difference feature set

Phishing webpages usually have resources pointing to or having their source from the legitimate website being masqueraded. Therefore, there is typically a difference between the domain of their URL and the domain of the URL of their resources.

The resources that were considered are links, styles, images and scripts which were represented by the HTML tags ⟨a⟩, ⟨link⟩, ⟨img⟩ and ⟨script⟩, respectively. These resources were gotten from the DOM of webpages, and their referenced URLs will be gotten from the adequate attributes.

Levenshtein distance string algorithm was employed to compute the difference between the domains of the webpage URL and resource URL for each URL in each resource category. The data would be normalized to allow for easy comparison of data from various webpages. The computed value would range from 0 to 1; Most phishing webpages would have values closer to 1. The features would be as follows as proposed by (Mohigimi & Varjani, 2016):

$$\mathrm{f7} = \frac{\sum_{i=0}^{n} \frac{\mathrm{LD(webpage\_domain,\ link_i\_domain)}}{\mathrm{max(|webpage\_domain|,\ |link_i\_domain|)}}}{n} \tag{1}$$

$$\mathrm{f8} = \frac{\sum_{i=0}^{n} \frac{\mathrm{LD(webpage\_domain,\ style_i\_domain)}}{\mathrm{max(|webpage\_domain|,\ |style_i\_domain|)}}}{n} \tag{2}$$

$$\mathrm{f9} = \frac{\sum_{i=0}^{n} \frac{\mathrm{LD(webpage\_domain,\ script_i\_domain)}}{\mathrm{max(|webpage\_domain|,\ |script_i\_domain|)}}}{n} \tag{3}$$

$$\mathrm{f10} = \frac{\sum_{i=0}^{n} \frac{\mathrm{LD(webpage\_domain,\ image_i\_domain)}}{\mathrm{max(|webpage\_domain|,\ |image_i\_domain|)}}}{n} \tag{4}$$

Resource access protocol feature set

This feature set will check if each URL resource in each resource category has SSL security. Legitimate webpages, especially internet banking and financial webpages, tend to have to access to their resources over secure protocols.

Links, styles, images, and scripts resources were considered and extracted from the DOM of the webpage. The features are represented with the aid of feature two as follows as proposed by (Mohigimi & Varjani, 2016):

$$\mathrm{f11} = \frac{\sum_{i=0}^{n} \mathrm{f2(link_i\_URL)}}{n} \tag{5}$$

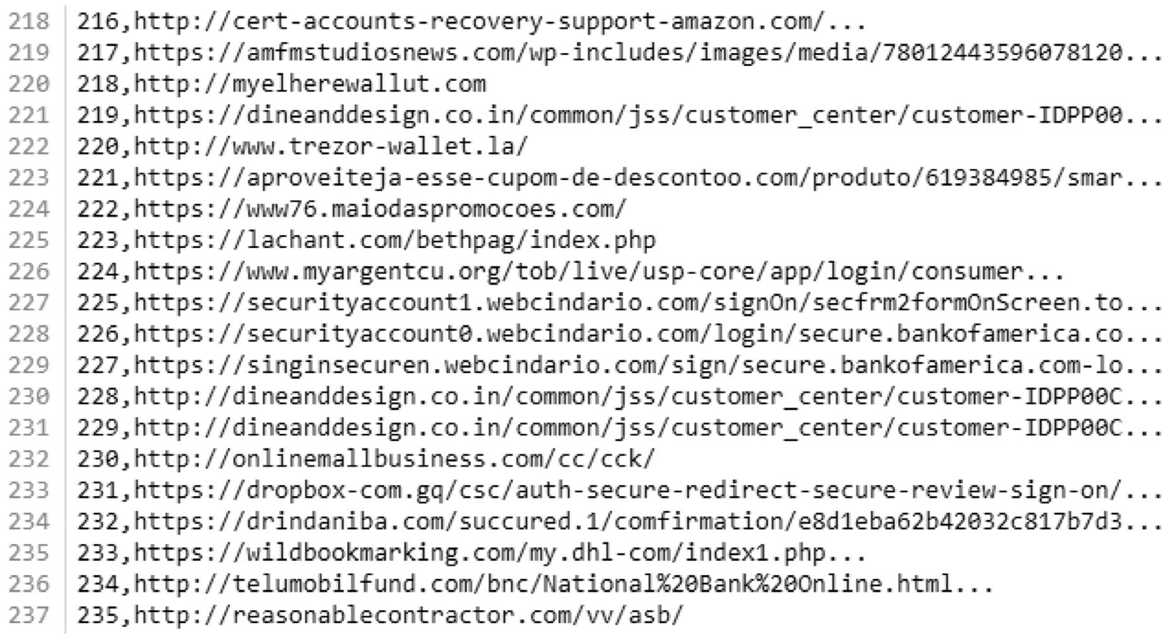
$$\mathrm{f12} = \frac{\sum_{i=0}^{n} \mathrm{f2(style_i\_URL)}}{n} \tag{6}$$

$$\mathrm{f13} = \frac{\sum_{i=0}^{n} \mathrm{f2(script_i\_URL)}}{n} \tag{7}$$

$$\mathrm{f14} = \frac{\sum_{i=0}^{n} \mathrm{f2(image_i\_URL)}}{n} \tag{8}$$

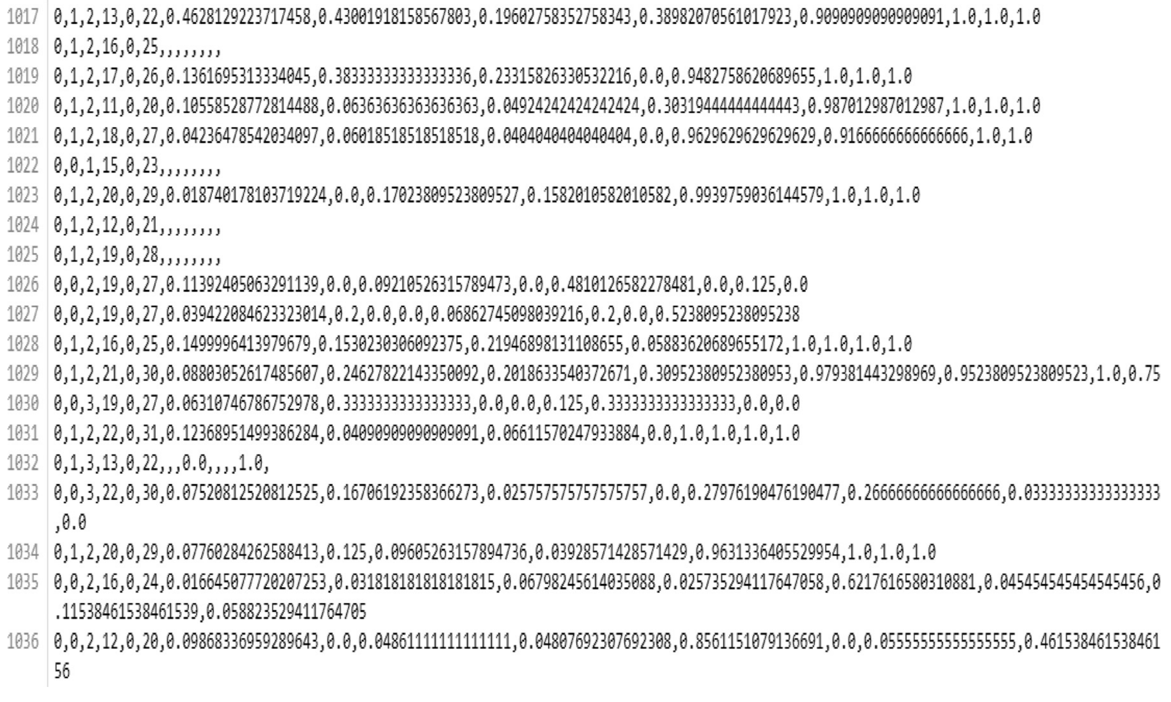
Model building and training

A total of two machine learning supervised classification algorithms were used to build and train classification models. The algorithms that were used are k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM). The models were trained with the dataset consisting of 14 features.



```
216,http://cert-accounts-recovery-support-amazon.com/...
217,https://amfmstudiosnews.com/wp-includes/images/media/78012443596078120...
218,http://myelherewallut.com
219,https://dineanddesign.co.in/common/jss/customer_center/customer-IDPP00...
220,http://www.trezor-wallet.la/
221,https://aproveiteja-esse-cupom-de-descontoo.com/produto/619384985/smar...
222,https://www76.maiodaspromocoes.com/
223,https://lachant.com/bethpag/index.php
224,https://www.myargentcu.org/tob/live/usp-core/app/login/consumer...
225,https://securityaccount1.webcindario.com/signOn/secfrm2formOnScreen.to...
226,https://securityaccount0.webcindario.com/login/secure.bankofamerica.co...
227,https://singinsecuren.webcindario.com/sign/secure.bankofamerica.com-lo...
228,http://dineanddesign.co.in/common/jss/customer_center/customer-IDPP00C...
229,http://dineanddesign.co.in/common/jss/customer_center/customer-IDPP00C...
230,http://onlinemallbusiness.com/cc/cck/
231,https://dropbox-com.gq/csc/auth-secure-redirect-secure-review-sign-on/...
232,https://drindaniba.com/succured.1/comfirmation/e8d1eba62b42032c817b7d3...
233,https://wildbookmarking.com/my.dhl-com/index1.php...
234,http://telumobilfund.com/bnc/National%20Bank%20Online.html...
235,http://reasonablecontractor.com/vv/asb/
```

Fig. 2. Screenshot sample of collected URLs.



```
0,1,2,13,0,22,0.4628129223717458,0.43001918158567803,0.19602758352758343,0.38982070561017923,0.9090909090909091,1.0,1.0,1.0
0,1,2,16,0,25,,,,,,,,
0,1,2,17,0,26,0.1361695313334045,0.38333333333333336,0.23315826330532216,0.0,0.9482758620689655,1.0,1.0,1.0
0,1,2,11,0,20,0.10558528772814488,0.06363636363636363,0.04924242424242424,0.30319444444444443,0.987012987012987,1.0,1.0,1.0
0,1,2,18,0,27,0.04236478542034097,0.06018518518518518,0.0404040404040404,0.0,0.9629629629629629,0.9166666666666666,1.0,1.0
0,0,1,15,0,23,,,,,,,,
0,1,2,20,0,29,0.018740178103719224,0.0,0.17023809523809527,0.1582010582010582,0.9939759036144579,1.0,1.0,1.0
0,1,2,12,0,21,,,,,,,,
0,1,2,19,0,28,,,,,,,,
0,0,2,19,0,27,0.11392405063291139,0.0,0.09210526315789473,0.0,0.4810126582278481,0.0,0.125,0.0
0,0,2,19,0,27,0.039422084623323014,0.2,0.0,0.0,0.06862745098039216,0.2,0.0,0.5238095238095238
0,1,2,16,0,25,0.1499996413979679,0.1530230306092375,0.21946898131108655,0.05883620689655172,1.0,1.0,1.0,1.0
0,1,2,21,0,30,0.08803052617485607,0.24627822143350092,0.2018633540372671,0.30952380952380953,0.979381443298969,0.9523809523809523,1.0,0.75
0,0,3,19,0,27,0.06310746786752978,0.3333333333333333,0.0,0.0,0.125,0.3333333333333333,0.0,0.0
0,1,2,22,0,31,0.12368951499386284,0.04090909090909091,0.06611570247933884,0.0,1.0,1.0,1.0,1.0
0,1,3,13,0,22,,,0.0,,,,1.0,
0,0,3,22,0,30,0.07520812520812525,0.16706192358366273,0.025757575757575757,0.0,0.27976190476190477,0.26666666666666666,0.03333333333333333
,0.0
0,1,2,20,0,29,0.07760284262588413,0.125,0.09605263157894736,0.03928571428571429,0.9631336405529954,1.0,1.0,1.0
0,0,2,16,0,24,0.01664507772020725З,0.031818181818181815,0.06798245614035088,0.025735294117647058,0.6217616580310881,0.045454545454545456,0
.11538461538461539,0.058823529411764705
0,0,2,12,0,20,0.09868336959289643,0.0,0.04861111111111111,0.04807692307692308,0.8561151079136691,0.0,0.05555555555555555,0.461538461538461
56
```

Fig. 3. Screenshot sample of the extracted output.

k-Nearest neighbors

k-Nearest Neighbor is a simple classification algorithm that trains a model simply by storing the dataset. It makes new predictions for a unique data point by looking for the datapoints closest to that datapoint and assigning it to the average class of those data points.

In its simplest form, the k-Nearest Neighbors algorithm considers only one neighbor and simply assigns the class of that neighbor to the new observation. The k in the name 'k-Nearest Neighbors' signifies an arbitrary number k of neighbors that can be considered.

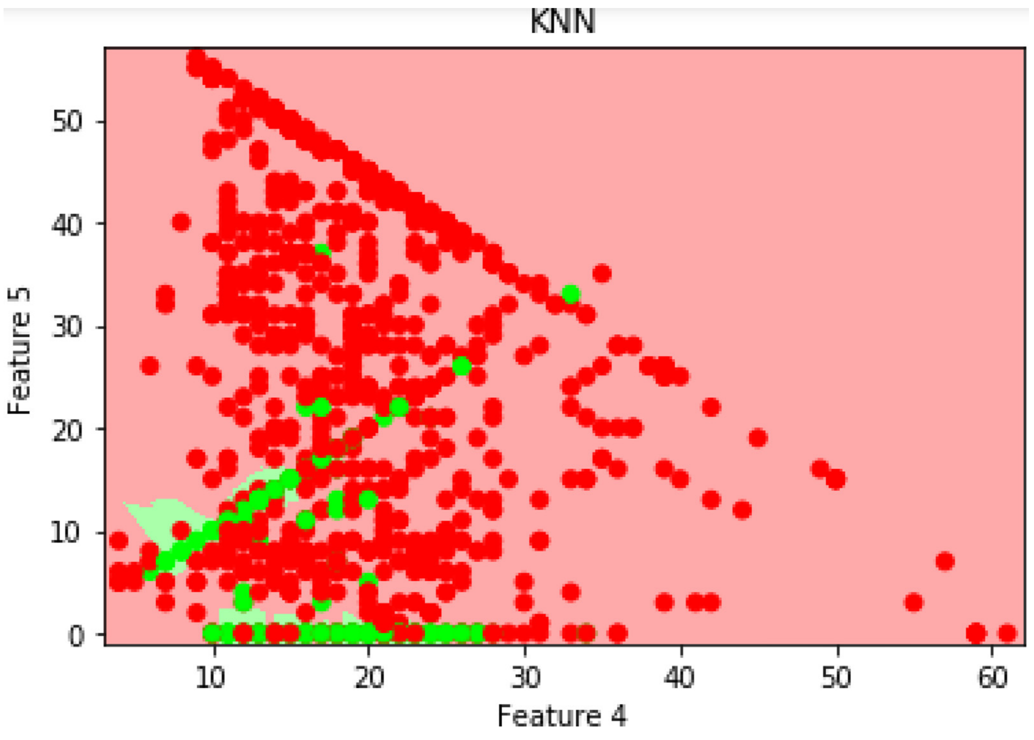
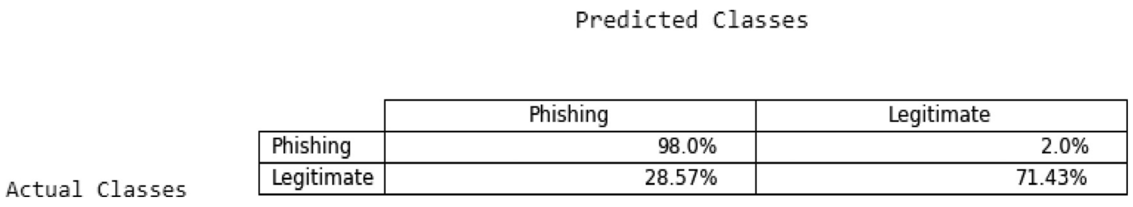


Fig. 4. Decision boundary for KNN model on two features.



Predicted Classes

Actual Classes

	Phishing	Legitimate
Phishing	98.0%	2.0%
Legitimate	28.57%	71.43%

Fig. 5. KNN model confusion matrix.

Support vector machine

Support Vector Machine is a linear model classification machine learning algorithm, i.e., it forms decision boundaries in a spatial plane using straight line segments, planes, or hyperplanes. New observations are then classified based on which class division they fall into. For instance, in a binary classifier, predictions are made using the general formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b > z \tag{9}$$

where,

w = regression coefficient
$x[i]$ = feature
b = intercept
$\hat{y}$ = regression prediction

The greater $> z$ at the end of the formula is the prediction aspect of the formula, which means values greater than z fall into one class, and those less than or equal to z fall into the other class.

Random Forest

A Random Forest is a collection of Decision trees with each tree differing slightly from the other. A prediction is made by averaging the result obtained from all individual Decision Trees. This helps to reduce the problem of overfitting, a problem peculiar to the Decision Tree Algorithm.

The algorithm is called Random Forest because of the way randomness is injected into the tree-building process to ensure each tree differs.

Table 1
KNN model classification report.

	Precision	Recall	F1-score	Support
Legitimate	0.88	0.71	0.79	21
Phishing	0.94	0.98	0.96	100
Micro avg	0.93	0.93	0.93	121
Macro avg	0.91	0.85	0.88	121
Weighted avg	0.93	0.93	0.93	121

Model assessment

In this phase, statistics such as True Positive, False Positive, True Negative, False Negative, Precision, F-score, Accuracy, and Recall would be computed for each model. These statistics were used as a standard for comparison, and the best performing model will be picked for the next phase, which is rule extraction.

Rule extraction

The method of rule extraction involves training a Decision Tree model with a synthetic dataset derived from the correct predictions made by the selected model. The best set rules for decision making can then be extracted from nodes in the Decision Tree.

Browser extension implementation

The selected ruleset will be embedded in a Google chrome web-browser extension called PhishNet. It would access the webpage DOM, analyze it, and extract the corresponding features required for classifying the webpage. The implementation of this tool are covered in more detail in the implementation.

Visual Studio code, Google Chrome, and the Python Scikit-learn library were used as tools to drive the implementation of this solution.

System implementation results

Data collection

This stage was executed with python scripts using a web scraping library called Beautiful soup. Phishing URLs were gotten from PhishTank (http://www.phishtank.com) while legitimate financial institution URLs were gotten from sites like Jasmine directory (https://www.jasminedirectory.com), Intechnic (https://www.intechnic.com), The Financial Brand (https://thefinancialbrand.com) and Similar web (https://www.similarweb.com). The URLs from these sites were consolidated and stored in an output CSV file as shown in Fig. 2.

Feature extraction

This stage was executed with a python script using a python web scraping library called Beautiful soup; each feature to be extricated was represented by a method. The script traverses through the list of URLs and visits each site, where it then accesses the Document Object Model (DOM) of each webpage in order to extract the required features and stores the output in CSV files. Fig. 3 is a snippet of output after feature extraction.

Model building and training

Three (3) Machine learning models were trained with the dataset with the aid of python's scientific library 'Scikit-learn'.

k-Nearest neighbor(KNN)

The k-Nearest Neighbor model generated the best outcomes with seven neighbors and was trained that way as shown in Fig. 4. Null-valued feature instances were assigned the mean value of that feature. The model had a high True positive estimation of 98%; however, it was hindered by the True Negative estimation of 71.43% as shown in Tables 1, 2, Figs. 5 and 6.

Support vector machine (SVM)

The SVM model was trained using a weight of 100. Null-valued feature instances were assigned with the mean estimation of that feature. The model reached a high True Positive estimation of 97% yet a low True Negative estimation of 66.67% as shown in Figs. 7 - 9, Tables 3 and 4.

Table 2
KNN summary of results.

STATISTIC	VALUE
Accuracy	93.39%
Error	6.61%
True Positive	98%
True Negative	71.43%
False Positive	28.57%
False Negative	2%
Precision	0.93
Recall	0.93
F-score	0.93

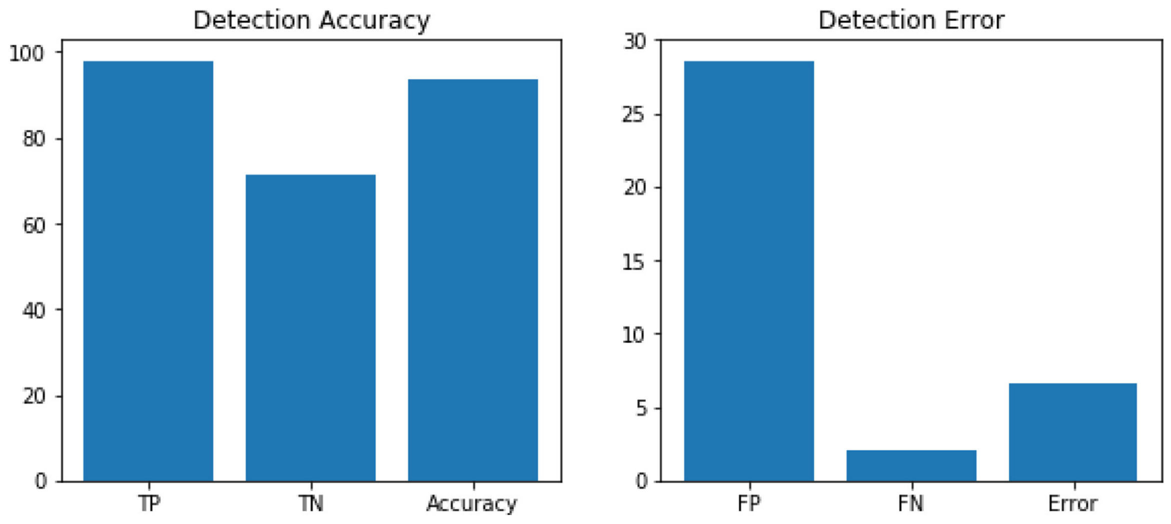


Fig. 6. Detection accuracy and detection error graph.

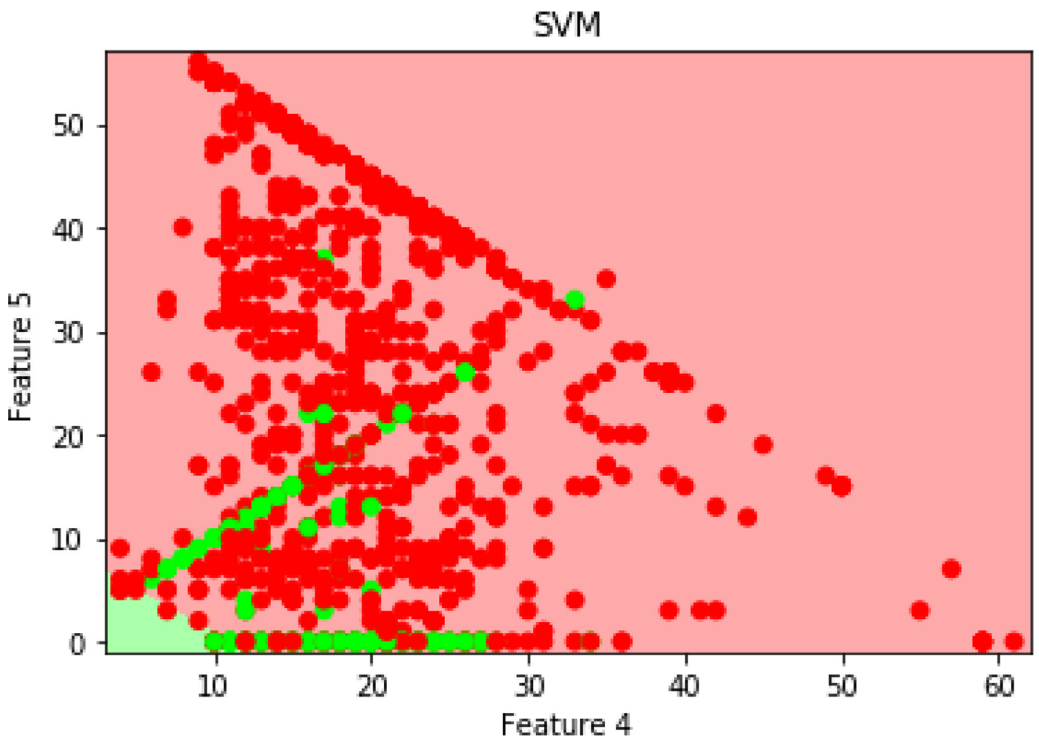
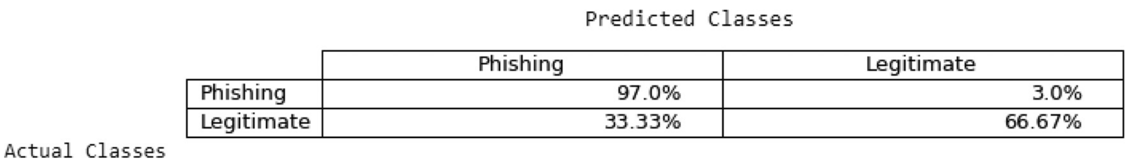


Fig. 7. Decision boundary for SVN model on two features.



Predicted Classes

Actual Classes	Phishing	Legitimate
Phishing	97.0%	3.0%
Legitimate	33.33%	66.67%

Fig. 8. SVM model confusion matrix.

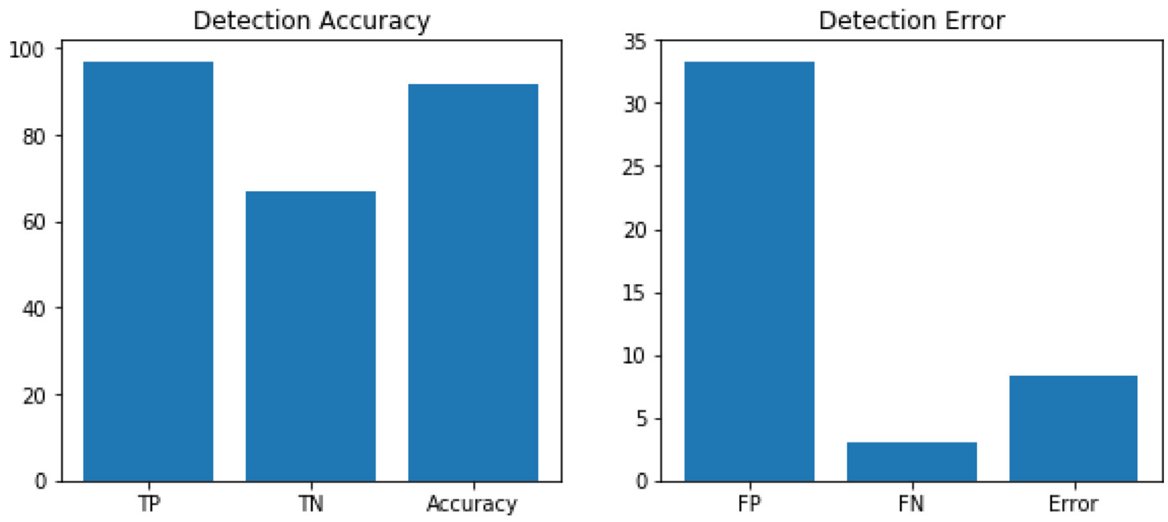


Fig. 9. Detection accuracy and detection error graph.

Table 3
SVM model classification report.

	Precision	Recall	F1-score	Support
Legitimate	0.82	0.67	0.74	21
Phishing	0.93	0.97	0.95	100
Micro avg	0.92	0.92	0.92	121
Macro avg	0.88	0.82	0.84	121
Weighted avg	0.91	0.92	0.91	121

Table 4
SVM summary of results.

STATISTIC	VALUE
Accuracy	91.74%
Error	8.26%
True Positive	97%
True Negative	66.67%
False Positive	33.33%
False Negative	3%
Precision	0.92
Recall	0.92
F-score	0.92

Random Forest

The Random Forest model was trained using five trees. Null-valued feature occurrences were assigned the value - 1. This model was the best performing model with a True Positive of 100%, a True Negative of 90.48%, and an accuracy of 98.35% on the test information as shown in Figs. 10 - 12, Tables 5 and 6.

Summary of all models

The Random Forest model performed best with an accuracy of 98.35% and a True Positive of 100%, followed by the k-Nearest neighbor model, which had an accuracy of 93.39%, True positive of 98% and a True Negative of 71.43% with the Support vector machine model coming in last as shown in Table 7.

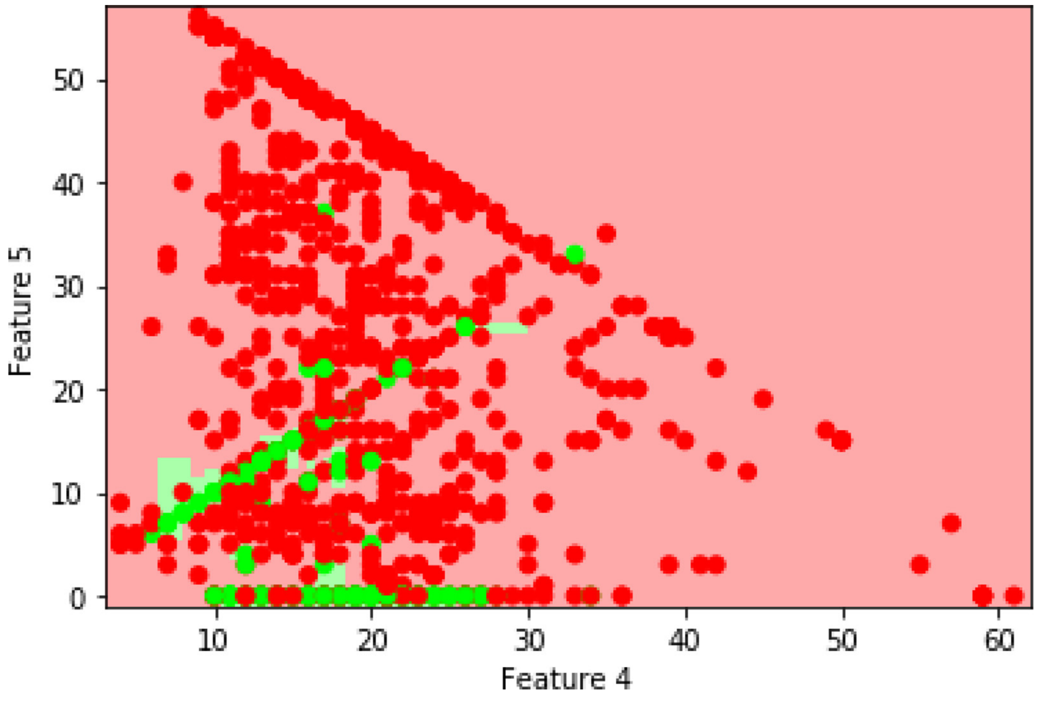


Fig. 10. Decision boundary for Random Forest on two features.

Actual Classes	Predicted Classes: Phishing	Predicted Classes: Legitimate
Phishing	100.0%	0.0%
Legitimate	9.52%	90.48%

Fig. 11. Random Forest confusion matrix.

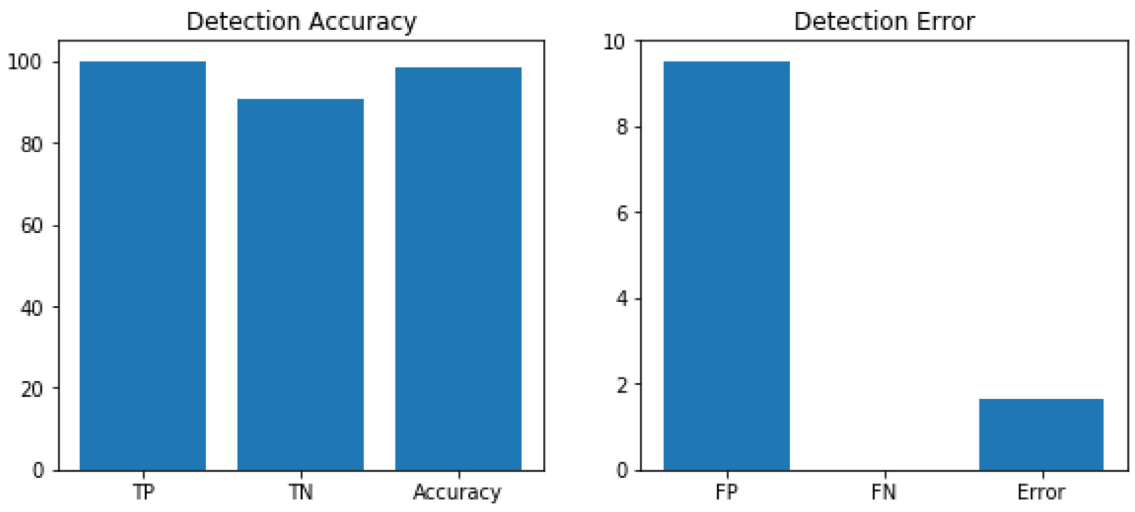


Fig. 12. Detection accuracy and detection error graph.

PhishNet implementation

PhishNet was implemented using technologies such as HTML, CSS, and Javascript to create a Google chrome web extension. The extension uses the rules extracted from the Random Forest model and prompts a user when the user lands on a phishing site as shown in Figs. 13 and 14.

Table 5
Random Forest classification report.

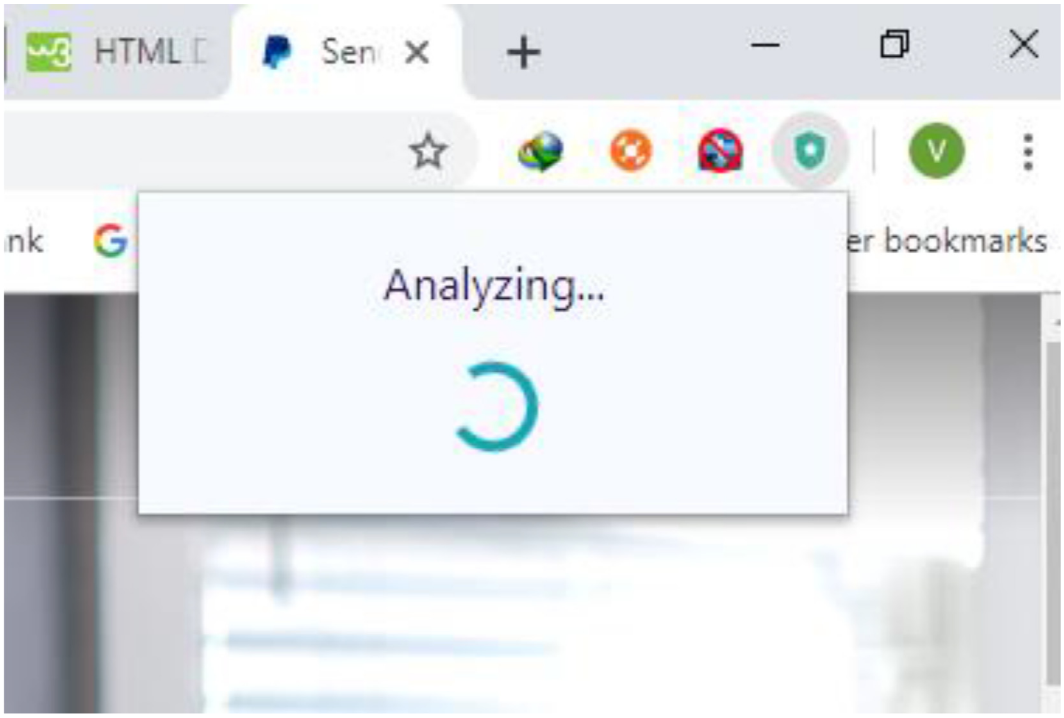
	Precision	Recall	F1-score	Support
Legitimate	1.00	0.90	0.95	21
Phishing	0.98	1.00	0.99	100
Micro avg	0.98	0.98	0.98	121
Macro avg	0.99	0.95	0.97	121
Weighted avg	0.98	0.98	0.98	121

Table 6
Random Forest summary of results.

STATISTIC	VALUE
Accuracy	98.35%
Error	1.65%
True Positive	100%
True Negative	90.48%
False Positive	9.52%
False Negative	0%
Precision	0.98
Recall	0.98
F-score	0.98

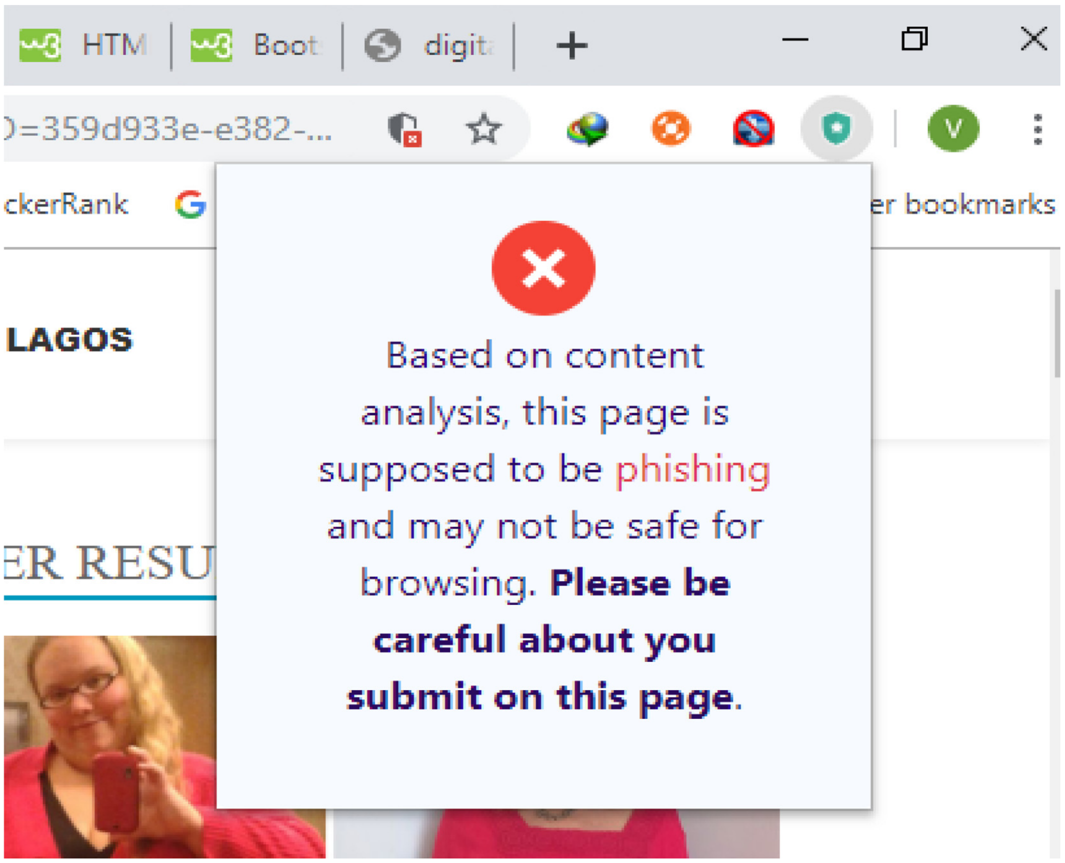
Table 7
Summary of results for all models.

STATISTIC	KNN	SVM	RF
Accuracy	93.39%	91.74%	98.35%
Error	6.61%	8.26%	1.65%
True Positive	98%	97%	100%
True Negative	71.43%	66.67%	90.48%
False Positive	28.57%	33.33%	9.52%
False Negative	2%	3%	0%
Precision	0.93	0.92	0.98
Recall	0.93	0.92	0.98
F-score	0.93	0.92	0.98



Analyzing...

Fig. 13. PhishNet analysing a page.



Based on content analysis, this page is supposed to be phishing and may not be safe for browsing. **Please be careful about you submit on this page.**

Fig. 14. PhishNet when it detects a phishing site.

Conclusion and future work

This paper presents the performance of machine learning classification model for phishing detection using a lightweight Google chrome extension. PhishNet was developed to detect phishing sites on the web. PhishNet provides a convenient solution to reduce the risk of phishing which in return would effectively alleviating the fear users feel when submitting sensitive information on the web.

The paper was able to evaluate the performance of three machine learning tools namely Random Forest, K-Nearest Neighbor and Support Vector Machine. The result given proved that Random Forest performed better than the other two machine learning tools. Consequently, our study has proffered a better solution to the issue of phishing attack on web pages.

The major shortcoming of this approach is its over-dependence on webpage content. Further work is hereby recommended in this area for future study.

Furthermore, the paper recommends more data to be gathered in the future to get more accurate results and the use of other machine learning tools could be tested.

Funding

This is to certify that this research work did not receive any funding.

Declaration of Competing Interest

The authors wish to declare that they have no conflict of interest.

References

[1] Eurostat, 2018. Internet banking on the rise.[Online] Available at: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20180115-1. [Accessed 22 August, 2020]
[2] Statista, 2018. Share of internet users who did online banking on a computer in the United States in 2018, by age.[Online] Available at: https://www.statista.com/statistics/228067/people-in-households-with-online-banking-usa/. [Accessed 20 June, 2020]

[3] Statista, 2018. Online industries most targeted by phishing attacks as of the 3rd quarter of 2018.[Online] Available at: https://www.statista.com/statistics/266161/websites-most-affected-by-phishing/ .[Accessed 10 July, 2020]
[4] W. Ali, Phishing website detection based on supervised machine learning with wrapper features selection, Int. J. Adv. Comput. Sci. Appl. 8 (2017) 72–78 January.
[5] A. Mishra, B.B. Gupta, Intelligent phishing detection system using similarity matching algorithms, Int. J. Inf. Commun. Technol. 12 (2018) 51–73.
[6] M. Mohigimi, A.Y. Varjani, New rule-based phishing detection method, Expert Syst. Appl. 53 (2016) 231–242.
[7] V. Muppavarapu, A. Rajendran, S. Vasudevan, Phishing detection using RDF and Random Forests, Int. Arab J. Inf. Technol. (2018) 817–824 September.
[8] O.K. Sahingoz, S.I. Baykal, D. Bulut, Phishing detection from urls by using neural networks, Comput. Sci. Inf. Technol. 8 (17) (2018) 41–54.
[9] O.K. Sahingoz, E. Buber, Ö. Demir, B. Diri, Machine learning-based phishing detection from URLs, Expert Syst Appl 117 (2019) 345–357 1 March.
[10] H. Abutair, A. Belghith, S. AlAhmadi, CBR-PDS: a case-based reasoning phishing detection system, J. Ambient Intell. Humaniz. Comput. 109 (2017) 281–288.
[11] Baraniuk, C., 2017. Google and Facebook duped in huge 'scam.'[Online] Available at: https://www.bbc.com/news/technology-39744007. [Accessed 10 September, 2020]
[12] C.L. Tan, K.L. Chiew, W. Koksheik, S.N. Sze, PhishWHO: phishing webpage detection via identity keywords extraction and target domain name finder, Decis. Support Syst. 88 (2016) 18–27.
[13] G. Varshney, M. Misra, P.K. Atrey, A phish detector using lightweight search features, Comput. Secur. 62 (C) (2016) 213–228.
[14] Webroot, 2019. 2019 Webroot Threat Report.[Online] Available at: https://www.webroot.com/us/en/about/press-room/releases/2019-webroot-threat-report. [Accessed 20 March, 2020]
[15] P. Yi, Web phishing detection using a deep learning framework, in: Wirel. Commun. Mob. Comput., 2018, Hindawi, 2018, pp. 1–9.