# Machine Learning Approach to Phishing Detection

Arvind Rekha Sura
Department of
Computer Science
University of Central Florida
Orlando, Florida
Email: arvindrs.gb@knights.ucf.edu

Jyoti Kini
Department of
Computer Science
University of Central Florida
Orlando, Florida
Email: jyoti.kini@knights.ucf.edu

Kishan Athrey
Department of
Computer Science
University of Central Florida
Orlando, Florida
Email: kishan.athrey@knights.ucf.edu

***Abstract*—The goal of our project is to implement a machine learning solution to the problem of detecting phishing and malicious web links. The end result of our project will be a software product which uses machine learning algorithm to detect malicious URLs. Phishing is the technique of extracting user credentials and sensitive data from users by masquerading as a genuine website. In phishing, the user is provided with a mirror website which is identical to the legitimate one but with malicious code to extract and send user credentials to phishers. Phishing attacks can lead to huge financial losses for customers of banking and financial services. The traditional approach to phishing detection has been to either to use a blacklist of known phishing links or heuristically evaluate the attributes in a suspected phishing page to detect the presence of malicious codes. The heuristic function relies on trial and error to define the threshold which is used to classify malicious links from benign ones. The drawback to this approach is poor accuracy and low adaptability to new phishing links. We plan to use machine learning to overcome these drawbacks by implementing some classification algorithms and comparing the performance of these algorithms on our dataset. We will test algorithms such as Logistic Regression, SVM, Decision Trees and Neural Networks on a dataset of phishing links from UCI Machine Learning repository and pick the best model to develop a browser plugin, which can be published as a chrome extension.**

***Keywords—Phishing Detection(PD), Chrome Extension(CE), Random Forest(RF), Support Vector Machine(SVM), Neural Networks(NN)***

## I. Introduction

Financial services such as banking are now easily available over the Internet making the lives of people easy. Thus it is very important that the security and safety of such services are maintained. One of the biggest threats to web security is Phishing. Phishing is the technique of extracting user credentials by masquerading as a genuine website or service over the web. There are various types of phishing attacks such as Spear phishing, which targets specific individuals or companies, Clone phishing is a type of phishing where an original mail with an attachment or link is copied into a new mail with a different (possibly malicious) attachment or link, Whaling, etc. Phishing can lead to huge financial losses. For example, the Microsoft Consumer Safer Index (MCSI) report for 2014 has estimated the annual worldwide impact of Phishing and other identity thefts to be nearly USD 5 Billion [1]. Similarly, the IRS has warned of a surge in phishing attacks with over 400% increase in reported cases [2].

Several solutions have been proposed to combat phishing ranging from educating the web users to stronger phishing detection techniques. The conventional approach to phishing detection has not been successful because of the diverse and evolving nature of phishing attacks. For instance, in January 2007, the total number of unique phishing reports submitted to the Anti-Phishing Working Group (APWG) was 29,930. Compared to the previous peak in June 2006, the number of submitted reports increased by 5% [3]. This happened despite taking preventive measure to thwart phishing. Upon investigation, it was found that each phishing attack was different from the other one. Thus it becomes imperative to find a way to adapt our phishing detection techniques as and when new attack patterns are uncovered.

Machine learning algorithms, which make a system learn new patterns from data, are an ideal solution to the problem of phishing detection. Although there have been many papers in recent years which have attempted to detect phishing attacks using machine learning, we intend to go one first step further and build a software tool which can be easily deployed in end user systems to detect phishing attacks.

For our project, we will experiment with three machine learning algorithms on a dataset of features that represent attributes commonly associated with phishing pages, choose the best model based on their performance and build a web browser plugin which will eventually be deployed to end users. The project report has been designed as follows; the Previous Work section describes the traditional approaches to phishing detection and some of the machine learning approaches attempted in recent years, the Proposed Approach section describes in detail our approach and what will be the end product of our project, the Dataset section describes the dataset that we are using for our project along with a list of features which will be used in our project, Machine Learning Algorithms section explains the different algorithms which we have tested with our dataset with their descriptions, the Chrome Plugin Implementation section describes the architecture of our phishing detection system and gives descriptions of the various software modules in the system, the Results section gives the results of our experiments with the algorithms with graphs plotting a comparison between the three algorithms on factors such as accuracy, sensitivity and false positive rate, and the Conclusion section summarizes our project with an outlook on future enhancements.

## II. Related Work

There are basically two approaches to phishing detection which are the blacklist approach and the heuristic evaluation of the source code to check for attributes commonly associated with phishing sites. The blacklist approach involves maintaining a publicly available list of reported phishing sites so that the number of victims of any new phishing site is kept very low. The main drawback with this approach is that a new phishing site has to be detected first before it can be reported. As phishers become more sophisticated with their phishing attacks, it becomes more difficult to detect those attacks. The other approach to phishing detection is to analyze the source code of a suspected phishing web page and identify attributes commonly associated with phishing sites. This approach is better than the blacklist method as we can potentially detect new phishing attacks within minutes of their launch. But as pointed out before, because of the diverse nature of phishing attacks it has become increasingly difficult to detect new attack patterns. Traditionally, the heuristic evaluation of the web page source code involved maintaining a count of identified phishing attributes and setting threshold for that count using trial and error, above which the page was considered a phishing site. As it is obvious, the hardcoding of such thresholds inherently make it difficult for the phishing detection system to adapt to new attack patterns. Thus, we use machine learning to train the system to detect phishing attacks and relearn from the data whenever a new phishing attack pattern is uncovered. There are sufficient literatures in recent years employing machine learning algorithms to the problem of phishing detection. In [3], for example, the work done by Chandrasekaran et al. describes a technique to classify phishing emails based on their structural properties such as style markers and the structure of the subject line, etc. They used Simulated Annealing for feature selection and Information Gain to rank the features based on their relevance on a dataset of 100 phishing emails and 100 legitimate emails. Later they used Support Vector Machines to classify the emails as phishing and legitimate based on their chosen features, getting an accuracy of 95%. The paper [3] also describes the approach taken by Fette et al., where they compared commonly used learning algorithms on a dataset of detected phishing emails composed of 860 phishing emails and 6950 legitimate emails. They hand picked ten features and used Random Forests algorithm on the dataset which provided an accuracy of 96% with a false positive rate of 0.1%.

## III. Proposed Approach

We propose to use machine learning to overcome the drawbacks associated with the traditional approaches to phishing detection. The problem of phishing detection is an ideal candidate for the application of machine learning solutions because of the easy availability of sufficient amounts of data on phishing attack patterns. The basic idea is to use machine learning algorithms on available dataset of phishing pages to generate a model which can be used to make classifications in real time if a given web page is a phishing page or a legitimate webpage. We intend to productionize the learned model into a software tool which can be deployed easily to end users for combating phishing attempts. For this purpose we have chosen to implement a machine learning algorithm from scratch using JavaScript and build a Chrome extension with it. A Chrome extension will enable us to deploy the learned model easily on the Chrome Web Store, from where anyone can download and use our product for phishing detection.

In order to successfully implement this project, we need to consider three constraints when choosing the machine learning algorithm for our product. First, the accuracy of the trained model should be high, as a product being used by end users in the real world should not give wrong results. Second, the algorithm which is being implemented should be able to make classifications in real-time; i.e. have very low execution time and also use less computational resources. Third, false positives and false negatives are important considerations when choosing a machine learning algorithm for the problem of phishing detection. This is because a user should not be wrongly led to believe that a phishing website is legitimate. Thus, we should look at these three constraints when selecting our phishing detection classifier.

## IV. Dataset

To evaluate our machine learning techniques, we have used the ‘Phishing Websites Dataset’ from UCI Machine learning repository. It consists of 11,055 URLs (instances) with 6157 phishing instances and 4898 legitimate instances. Each instance contains 30 features. Each feature is associated with a rule. If the rule satisfies, it is termed as phishing. If the rule doesn’t satisfy then it is termed as legitimate. The features take three discrete values. ‘1’ if the rule is satisfied, ‘0’ if the rule is partially satisfied, ‘-1’ if the rule is not satisfied.

The training dataset for our project is taken from the "Phishing Websites Data Set" of the UCI Machine learning repository. This dataset was compiled by [see acknowledgements]. The dataset consists of 11,055 entries with 6157 phishing instances and 4898 legitimate instances. Each instance consists of 30 features comprising of various attributes typically associated with phishing or suspicious web pages such as presence of IP address in the URL domain or presence of JavaScript code to modify the web browser address bar information. Each feature is associated with a rule. If the rule is satisfied, we take it as an indicator of phishing and legitimate otherwise. The dataset has been normalized to contain only discrete values. Each feature of each instance will contain ’1’ if the rule associated with that feature is satisfied, ’0’ if partially satisfied and ’-1’ if unsatisfied.

The features represented by the training dataset can be classified into four categories;
i) Address Bar based features
ii) Abnormal based features
iii) HTML and JavaScript based features
iv) Domain based features

### A. *Address bar based features*

1.1 Using IP address: If the domain of the URL of the suspected web page contains IP address, then we take it as a phishing page. eg: http:125.98.3.123fake.html or http:x58.0xCC.0xCA.0x622paypal.caindex.html

1.2 Long URL to hide suspicious part: It has been a common observance that phishing web pages usually have long URLs that attempt to hide malicious URL fragments from the user. We take the assumption that a web page with a long URL is necessarily a phishing or suspicious site. In the event the assertion fails, i.e, for a legitimate web page with valid long URLs, the absence of other phishing attributes on the web page will balance the wrong assumption and correctly classify a legitimate web page as non-phishing.

1.3 Use of URL shortening services: A shortened URL hides the real URL behind a redirection hop. A web page that uses a URL shortening service such as TinyURL is highly suspicious and is likely to be a phishing attempt. Therefore, we set the rule that if the URL has been shortened using a URL shortening service then it is a phishing page and legitimate otherwise.

1.4 Use of "@" symbol: Needs verification The "@" symbol is a reserved keyword according to Web standards. So the presence of "@" in a URL is suspicious and the web page is taken as phishing and legitimate otherwise.

1.5 Redirection with "": The presence of "//" in the URL path indicates the page will be redirected to another page. If the position of "//" in the URL is greater than seven then it is a phishing site and legitimate otherwise.

1.6 Adding prefix or suffix separated by "-" to the domain: Phishers tend to add a prefix or suffix to the domain with "-" to give the resemblence of a geniune site. Eg: www.a-paypal.com

1.7 Sub domains and multi sub domains: If a URL has more than three dots in the domain part then it is considered as a phishing site and legitimate otherwise.

### B. *Abnormal based features*

2.1 Request URL: A legitimate site usually has external page objects such as images, animations, files, etc. be accessed by a request URL which shares the same domain as the web page URL. We classify sites which fail this rule as phishing.

2.2 URL portion of anchor tag: We check if the domain in the URL portion of all anchor tags match the main URL of the page and if the anchor tag has only URL fragments or JavaScript functions.

2.3 Links in <meta>, <script> and <link> tags: We check if the domain of the links in the <meta>, <script> and <link> tags matches the domain in the mail URL.

2.4 Server Form Handler (SFH): When a form is submitted, some valid action must be taken. So if the action handler of a form is empty or "about:blank" or if the domain of the action URL is different from the domain of the main URL, then it is taken as a phishing site.

2.5 Submitting Information to Email: If the webpage contains a "mailto:" function then it is taken as a phishing site and legitimate otherwise.

### C. *HTML and Javascript based features*

3.1 Status bar customization: Phishers can modify the status bar using JavaScript to show a legitimate URL. By analyzing the "onMouseOver" events in the web page we can determine if such a modification has occurred.

3.2 Disabling right click option: Phishers can disable the right click option to prevent the user from checking the source code of the page. This is verified by analyzing the source code.

3.3 Using pop-up window: Legitimate sites rarely ask for user info on a pop-up window, whereas phishing sites generally use pop-up windows to get user info.

3.4 Iframe redirection: Phishers also use Iframe tags with invisible borders to get user info and redirect to the original site. We analyze the source code to check if Iframe tags are used.

## V. Machine Learning Implementation

We have trained and tested supervised machine learning algorithms on the training dataset. The following algorithms were chosen based on their performance on classification problems. The dataset was split into training and test set in the ratio 7:3. The results of our experiment are given in the results section.

### A. *Random Forests*

Random forests are classifiers that combine many tree predictors, where each tree depends on the values of a random vector sampled independently. Furthermore, all trees in the forest have the same distribution. To construct a tree, we assume that n is the number of training observations and p is the number of variables (features) in a training set. To determine the decision node at a tree we choose k « p as the number of variables to be selected. We select a bootstrap sample from the n observations in the training set and use the rest of the observations to estimate the error of the tree in the testing phase. Thus, we randomly choose k variables as a decision at a certain node in the tree and calculate the best split based on the k variables in the training set. Trees are always grown and never pruned compared to other tree algorithms. Random forests can handle large number of variables in a data set. Also, during the forest building process they generate an internal unbiased estimate of the generalization error. In addition, they can estimate missing data well. A major drawback of random forests is the lack of reproducibility, as the process of building the forest is random. Further, interpreting the final model and subsequent results is difficult, as it contains many independent decisions trees.

### B. *Artificial Neural Networks*

A neural network is structured as a set of interconnected identical units (neurons). The interconnections are used to send signals from one neuron to the other. In addition, the interconnections have weights to enhance the delivery among neurons. The neurons are not powerful by themselves, however, when connected to others they can perform complex computations. Weights on the interconnections are updated when the network is trained, hence significant interconnection play more role during the testing phase. Since interconnections do not loop back or skip other neurons, the network is called feedforward. The power of neural networks comes from the nonlinearity of the hidden neurons. In consequence, it is significant to introduce nonlinearity in the network to be able to learn complex mappings. The commonly used function in neural network research is the sigmoid function. Although competitive in learning ability, the fitting of neural network models requires some experience, since multiple local minima are standard and delicate regularization is required.

### C. Support Vector Machines

Support Vector Machine (SVM) is a supervised machine learning discriminative model, which conforms to the principle of drawing separating hyper-plane with maximum safety space, called margin, to minimize the risk of flawed predictions.

For linearly separable data points, Hard-Margin SVM is modelled on below primal optimization:

$$Objective(min) \quad \frac{1}{2}W^TW \tag{1}$$

$$Constraints \quad y_i(W^TX_i+b)>=1, \quad i=1,...n \tag{2}$$

Here, primarily, the hyperplanes, which satisfy the constraints are identified and only for the valid hyperplanes we tend to maximise the margin. The arithmatic computation involved is simplified using the dual form, which is as follows:

$$Objective(max) \quad q(\mu)=\sum_{i=1}^{n}\mu_i-\frac{1}{2}\sum_{i,j=1}^{n}\mu_i\mu_jy_iy_jX_i^TX_j \tag{3}$$

$$Constraints \quad \mu_i>=0, \quad i=1,...n, \quad \sum_{i=1}^{n}\mu_iy_i=0 \tag{4}$$

$$W^*=\sum\mu_i^*y_iX_i, \quad b^*=y_i-X_i^TW^*, \quad j \quad \mu_i>0 \tag{5}$$

However, the formulation fails to incorporate the outliers and therefore, cannot be used for generalisation.

Contradictory to the Hard-Margin SVM, the Soft-Margin SVM offers flexibility to select the optimal support vectors using C.The primal ojective for Soft-Margin SVM is as below:

$$Objective(min) \quad \frac{1}{2}W^TW+C\sum_{i=1}^{n}\xi_i \tag{6}$$

$$Constraints \quad to \quad y_i(W^TX_i+b)>=1-\xi_i, \quad i=1,...n \tag{7}$$

$$\xi_i>=0, \quad i=1,...n \tag{8}$$

While, the dual problem is represented as:

$$Objective(max) \quad q(\mu)=\sum_{i=1}^{n}\mu_i-\frac{1}{2}\sum_{i,j=1}^{n}\mu_i\mu_jy_iy_jX_i^TX_j \tag{9}$$

$$Constraints \quad C>=\mu_i>=0, \quad i=1,...n, \quad \sum_{i=1}^{n}\mu_iy_i=0 \tag{10}$$

$$W^*=\sum\mu_i^*y_iX_i, \quad b^*=y_i-X_i^TW^*, \quad j \quad C>\mu_i>0 \tag{11}$$

Furthermore, the data instances, which exhibit non-linear separation, are mapped to higher dimension space and classified using hyperplane in the higher dimension. Here, the kernel-trick plays a vital role in computational cost reduction by introducing kernel function as an alternative to computing higher dimension vectors.

$$(Z_i,y_i), \quad i=1,...n, \quad Z_i=\Phi(X_i) \tag{12}$$

$$Objective(max) \quad q(\mu)=\sum_{i=1}^{n}\mu_i-\frac{1}{2}\sum_{i,j=1}^{n}\mu_i\mu_jy_iy_jZ_i^TZ_j \tag{13}$$

$$Constraints \quad C>=\mu_i>=0, \quad i=1,...n, \quad \sum_{i=1}^{n}\mu_iy_i=0 \tag{14}$$

$$W^*=\sum\mu_i^*y_iX_i, \quad b^*=y_i-X_i^TW^*, \quad j \quad C>\mu_i>0 \tag{15}$$

Below are some of the most commonly used kernel functions:

$$Gaussian \quad K_g(X_i,X_j)=e^{-\frac{||X_i-X_j||^2}{\sigma^2}} \tag{16}$$

$$Sigmoid \quad K_s(X_i,X_j)=tanh(\alpha X_i^TX_j+\theta) \tag{17}$$

$$Polynomial \quad K_p(X_i,X_j)=(1+X_i^TX_j)^p \tag{18}$$

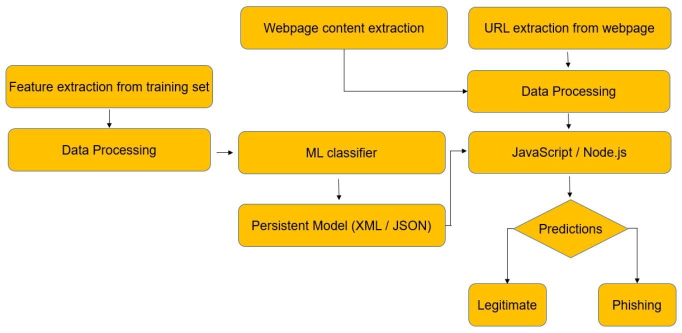
## VI. Technical Approach Details

The proposed approach aims at building a browser extension powered by state-of-the-art machine learning technique for phishing detection. Furthermore, given the flexibility of margin and reduced computational complexity offered by SVM, for classification problem statements, the implementation employs SVM trained persistent model to identify the malicious sites. The extension is packaged to support Chrome browser in specific, solely by the virtue of its popularity. Additionally, extensions exhibit minimal web-dependence, as it collates multiple files into single file for user to download, as one-time activity.

### A. Browser Extension Schematics

The solution deals with training the model with available data-set, using SVM discriminative classifier, followed by passing the persistent model to the extension, which further predicts the authenticity of the user accessed websites and provides alerts to notify the legitimacy of the browsed URL on every page load. The solution integrates Python-based training stage implementation with JavaScript-based testing module. The training component has been designed using Python, so as to make optimal utilisation of the available complex numeric computation libraries. Moreover, given the fact that the testing stage is centric to web-content and feature extraction, and has minimal heavy computation activities associated; the solution does face client-end computation performance lag concerns.

During the initial analysis of the project, the team analysed couple of approaches; and weighing the pros, cons and bandwidth of the resources, finalised the persistent model passing methodology as the favored methodology. One of the planned approaches aimed at developing Node.js enabled testing component, where the SVM model is structured as



Webpage content extraction
URL extraction from webpage
Feature extraction from training set
Data Processing
Data Processing
ML classifier
JavaScript / Node.js
Persistent Model (XML / JSON)
Predictions
Legitimate
Phishing

Fig. 1: Proposed phishing detection chrome extension implementation

scalable Web API for the testing module's consumption. Brython server-side architecture, which enables to run python in the browser; and Rapydscript client-side architecture, which supports compiling python to javascript; have been some of the other options considered during the implementation. However, due to the computation advancements offered by Python over Brython/Rapydscript, the solution has been designed with Python-based training module.

### B. Algorithm Details

The Chrome extension complies to the Google norms and, primarily, consists of three main files: manifest.json, content.js, background.js. The manifest file provides all the meta data information about the extension to Chrome browser. Additionally, it also specifies all the files and other resources associated to the extension. The content.js file loads on every page in the Chrome browser, post the extension deployment. However, it is an unprivileged module, which has direct access only to the DOM elements and needs supporting files to interact to external APIs and browser user interface manipulation. The supplementary file background.js aids the content script with these interactions, which is termed as message passing.

Multiple functions have been implemented in the content.js script for web-content and URL feature extraction. Below are the details used to identify phishing portals: isIPInURL(): Identify presence of IP address in the URL
isLongURL(): Validate if length of the URL is beyond 75 characters
isTinyURL(): Identify URLs smaller than 20 charaters
isAlphaNumericURL(): Check for alphanumeric '@' in URL
isRedirectingURL(): Verify if '//' existing within the URL more than once
isHypenURL(): Check for presence of '-' adjacent to domain name in URL
isMultiDomainURL(): Domain name should be confined to top-level domain, country-code and second-level domain.
isFaviconDomainUnidentical(): Verify if links on given web-page are loaded from other domains
isIllegalHttpsURL(): Identify presence of multiple 'https' in the URL string
isImgFromDifferentDomain(): Validate if images on given web-page are loaded from other domains
isAnchorFromDifferentDomain(): Detect if links on given web-page are loaded from other domains
isScLnkFromDifferentDomain(): Identify if scripts on given web-page are loaded from other domains
isFormActionInvalid(): Detect invalid/blank form submissions
isMailToAvailable(): Check for anchor tag incorporating mailto
isStatusBarTampered(): Validate if onmouseover manipulates the status bar display
isIframePresent(): Identify sites, which exhibit iframes in the DOM
The extracted features, further, passed through the SVM model identify hostile web-URLs.

## VII. Experimental Evaluation

This project compares the performance of all the classifiers described in section 5 on the phishing dataset. We have evaluated these algorithms on 3317 test samples using various performance metrics and this section contains the tabulated results with their graphs.

TABLE I: Random Forest Confusion Matrix

| | Predicted Phishing URLs | Predicted Legitimate URLs |
|---|---|---|
| Ground Truth Phishing URLs | 1249 | 162 |
| Ground Truth Legitimate URLs | 182 | 1680 |

Table 1 shows the confusion matrix for Random forests. With 1249 true positives, 182 false positives, 162 false negatives and 1680 true negatives.

TABLE II: Artificial Neural Network Confusion Matrix

| | Predicted Phishing URLs | Predicted Legitimate URLs |
|---|---|---|
| Ground Truth Phishing URLs | 1205 | 250 |
| Ground Truth Legitimate URLs | 170 | 1692 |

Table 2 shows the confusion matrix for Artificial neural network. With 1205 true positives, 170 false positives, 250 false negatives and 1692 true negatives.

TABLE III: SVM Confusion Matrix

| | Predicted Phishing URLs | Predicted Legitimate URLs |
|---|---|---|
| Ground Truth Phishing URLs | 1293 | 206 |
| Ground Truth Legitimate URLs | 131 | 1731 |

TABLE IV: Performance matrix of classifiers

| | Accuracy(%) | Specificity(%) | Sensitivity(%) |
|---|---|---|---|
| Artificial Neural Network | 87.34 | 91 | 83 |
| Random Forest | 89.63 | 90 | 86 |
| SVM | 89.84 | 93 | 89 |

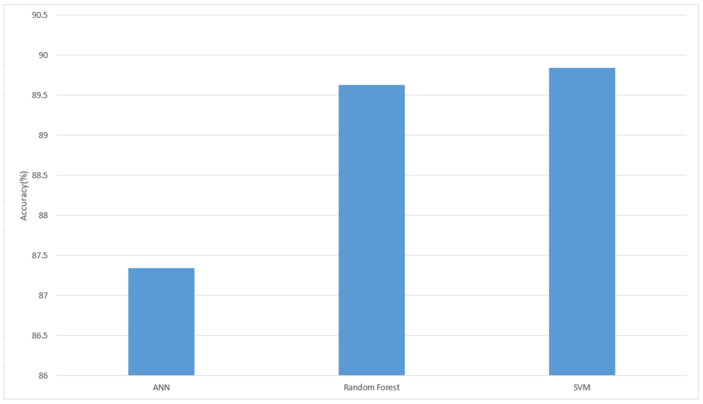


Fig. 2: Accuracy of classifiers

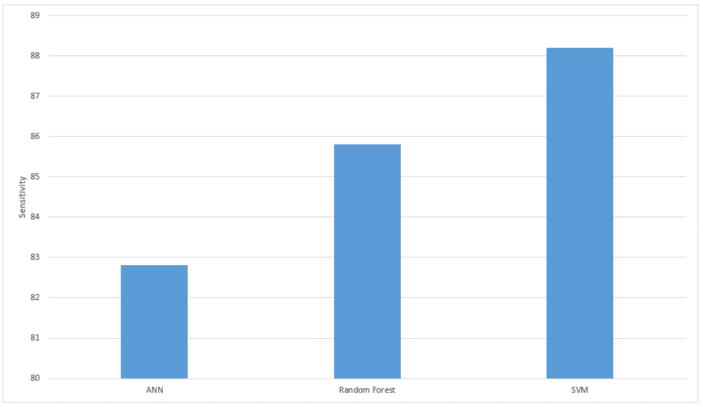


Fig. 3: Sensitivity of classifiers

Table 3 shows the confusion matrix for Support vector machine. With 1293 true positives, 206 false positives, 131 false negatives and 1731 true negatives.

The Figure 2 shows the accuracy of each classifier evaluated on 3317 test samples. From the above figure, it can be seen that SVM outperforms all the other algorithms based on accuracy in detection of Phishing URL. Also, Figure 3 shows the sensitivity of each classifier. Here sensitivity refers to the classifier's ability to correctly detect phishing URLs. It can be seen that SVM has the highest sensitivity among all the other classifiers. However, in phishing detection, false positives and false negatives are given more consideration when studying the performance (predictive accuracy) of a classifier. That is because false positives are more expensive than false negatives in the real world. Since we do not want to allow users to access the phishing URLs, false positives are considered to be important while deciding the best classifier. The Figure 4 shows the false positive rates of all the classifiers. It is evident that SVM has the least False positive rate among the three. Hence, SVM works best in classifying the phishing URL from the legitimate URLs.

## VIII. Conclusion

Thus to summarize, we have seen how phishing is a huge threat to the security and safety of the web and how phishing

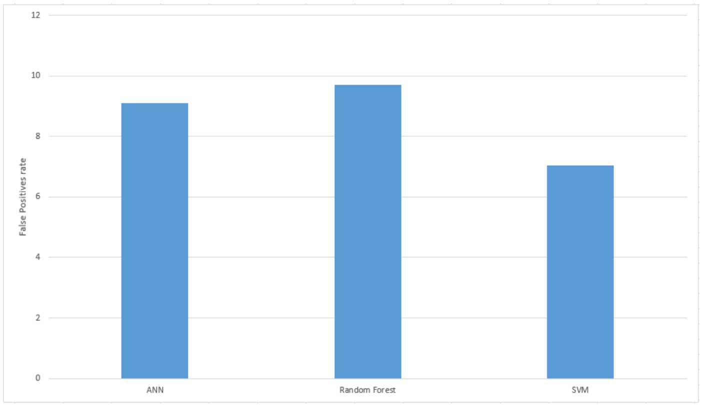


Fig. 4: False positive rate of classifiers

detection is an important problem domain. We have reviewed some of the traditional approaches to phishing detection; namely blacklist and heuristic evaluation methods, and their drawbacks. We have tested three machine learning algorithms on the 'Phishing Websites Dataset' from the UCI Machine Learning Repository and reviewed their results. We then selected the best algorithm based on it's performance and built a Chrome extension for detecting phishing web pages. The extension allows easy deployment of our phishing detection model to end users. For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

## References

[1] Microsoft, *Microsoft Consumer safety report*. Available at https://news.microsoft.com/en-sg/2014/02/11/microsoft-consumer-safety-index-reveals-impact-of-poor-online-safety-behaviours-in-singapore/sm.001xdu50tlxsej410r11kqvksu4nz

[2] Internal Revenue Service, *IRS E-mail Schemes*. Available at https://www.irs.gov/uac/newsroom/consumers-warned-of-new-surge-in-irs-email-schemes-during-2016-tax-season-tax-industry-also-targeted

[3] Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. (2007), *A comparison of machine learning techniques for phishing detection. Proceedings of the Anti-phishing Working Groups 2nd Annual ECrime Researchers Summit on - ECrime '07.* doi:10.1145/1299015.1299021

[4] E., B., K., T. (2015)., *Phishing URL Detection: A Machine Learning and Web Mining-based Approach. International Journal of Computer Applications,123(13), 46-50.* doi:10.5120/ijca2015905665

[5] Wang Wei-Hong, L V Yin-Jun, CHEN Hui-Bing, FANG Zhao-Lin., *A Static Malicious Javascript Detection Using SVM*, In Proceedings of the 2nd International Conference on Computer Science and Electrical Engineering(ICCSEE 2013)

[6] Ningxia Zhang, Yongqing Yuan, *Phishing Detection Using Neural Network*, In Proceedings of International Conference on Neural Information Processing, pp. 714–719. Springer, Heidelberg (2004)

[7] Ram Basnet, Srinivas Mukkamala et al, *Detection of Phishing Attacks: A Machine Learning Approach*, In Proceedings of the International World Wide Web Conference(WWW), 2003

[8] Sci-kit learn, *SVM library*. Available: http://scikit-learn.org/stable/modules/svm.html

[9] Sklearn, *ANN library*. Available: http://scikit-learn.org/stable/modules/ann.html

[10] Sklearn, *Random foresets library*. Available: http://scikit-learn.org/stable/modules/Randomforesets.html

[11] Lichman, M. (2013). *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences. Available: http://archive.ics.uci.edu/ml