VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

PHISHING DETECTION USING MACHINE LEARNING

Submitted by :

Arihant Bhardwaj 21BCE1008

Ashna Sachdeva 21BCE1693

Shradha Suman Jena 21BCE1764

Course Code : BCSE353E

Course Title : Information Security Analysis and Audit

Guided By: Mr. Joshan Athanesious J

PHISHING DETECTION USING MACHINE LEARNING

Abstract

Our project's purpose is to use machine learning to develop a solution for detecting phishing and harmful online links. Our study will culminate in a software solution that employs a machine learning algorithm to detect fraudulent URLs. Phishing isa technique for obtaining user passwords and sensitive data by impersonating a legitimate website. In phishing, the user is presented with a fake website that looks just like the real one but contains malicious code that extracts and sends the user's credentials to the phishers. Customers of banking and financial services may suffer significant financial losses as a result of phishing attempts. To detect the presence of harmful programmes,the typical method to phishing detection has been to either employa blacklist of known phishing URLs or heuristically evaluate the properties of a suspected phishing page. The heuristic algorithm uses trial and error to determine the threshold for classifying harmful and benign links. The disadvantages of this method include its lack of accuracy and flexibility to new phishing links. By building different categorization algorithms and comparingtheir performance on our dataset, we hope to apply machine learning to solve these disadvantages. On a dataset of phishingURLs from the UCI Machine Learning repository, we will evaluate methods such as Logistic Regression, SVM, DecisionTrees, and Neural Networks, and choose the best model to construct a browser plugin that may be distributed as a chromeextension.

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

INTRODUCTION

The availability of financial services, such as online banking, has simplified people's life. Therefore, it is crucial to preserve the security and safety of such services. One of the biggest threats to web security is phishing.

Phishing is a method for gaining user credentials online by pretending to be a trustworthy website or service. Spear phishing, which targets particular people or companies, Clone phishing, which involves copying an original email with an attachment or link into a new email with a different (possibly malicious) attachment or link, Whaling, and other types of phishing attacks are just a few examples. Phishing may cause huge financial losses. The 2014 Microsoft Consumer Safer Index (MCSI) survey estimates that the yearly global impact of phishing and other identity crimes will exceed USD 5 billion. Similarly, citing a 400% increase in reported incidents, the IRS has issued a warning about an increase in phishing attacks.

There are a number of ways to combat phishing, from better phishing detection tools to web user education. The conventional approach of phishing detection has been rendered ineffective due to the complex and dynamic nature of phishing assaults. For instance, the Anti-Phishing Working Group (APWG) received a total of 29,930 unique phishing reports in January 2007. 5% more reports were submitted than at the preceding peak in June 2006. Despite taking steps to prevent phishing, this still occurred. Further

investigation revealed that each phishing attempt was unique from the others. Finding a way to modify our phishing detection methods as new attack patterns are identified becomes crucial as a result.

Since machine learning methods enable a system to extract new patterns from data, they are a useful remedy for the issue of phishing detection. We aim to go a step further and develop a software application that can easily be installed in end user computers to detect phishing attempts, despite the fact that numerous publications have recently attempted to detect phishing attacks using machine learning.

For this project, we'll test three machine learning algorithms on a dataset of features that describe traits typically associated with phishing pages, pick the best model based on its performance, and construct a web browser plugin that will be distributed to end users. The project report is organised as follows: the Previous Work section describes traditional approaches to phishing detection and some of the machine learning approaches attempted in recent years, the Proposed Approach section describes our approach in detail 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 that will be used in our project, and the Dataset section describes the dataset that we are using for our project along with a list of features that will be used in our project, The Machine Learning Algorithms section explains the various algorithms that we have tested with our dataset and 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, and 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, segregation, and recall, and the Results section gives the results of our experiments with the algorithms with graphs plotting a comparison.

PROBLEM STATEMENT

Phishing attacks have become increasingly sophisticated, posing a significant threat to individuals and organizations alike. Detecting phishing attempts accurately and efficiently is crucial to protect sensitive information and prevent financial losses. Traditional rule-based approaches and manual inspection methods are often insufficient to keep up with evolving phishing techniques. Therefore, there is a need for an automated system that can effectively detect phishing emails and websites.

SOLUTION

Phishing detection using machine learning offers a promising approach to tackle the problem. Here is a potential solution:

1. Dataset Collection: Gather a comprehensive dataset consisting of labeled examples of both legitimate and phishing emails or websites. This dataset should include various features such as email headers, content, URLs, and other relevant information.

2. Feature Extraction: Extract meaningful features from the collected dataset. This can include analyzing the email sender, email content, URL characteristics (e.g., domain, length, SSL certificate), and other indicators of phishing attempts. Feature engineering techniques like TF-IDF, N-grams, and URL parsing can be employed to extract relevant information.

3. Data Preprocessing: Clean and preprocess the extracted features. This may involve removing irrelevant or noisy features, handling missing values, and normalizing or scaling the data to ensure uniformity.

4. Model Selection: Choose an appropriate machine learning algorithm for phishing detection. Several

algorithms can be considered, including decision trees, random forests, support vector machines (SVM), logistic regression, or deep learning models like convolutional neural networks (CNN) or recurrent neural networks (RNN).

5. Model Training: Split the preprocessed dataset into training and validation sets. Train the selected machine learning model using the training set. This involves feeding the labeled examples into the model and adjusting its parameters to learn the patterns and characteristics of phishing emails or websites.

6. Model Evaluation: Evaluate the trained model's performance using the validation set. Metrics such as accuracy, precision, recall, and F1-score can be used to assess its effectiveness in detecting phishing attempts. Adjust the model or experiment with different algorithms if necessary to improve the performance.

7. Deployment and Real-Time Detection: Once the model achieves satisfactory performance, deploy it in a real-time environment where it can analyze incoming emails or URLs and classify them as either legitimate or phishing. The system can then trigger appropriate actions, such as blocking or flagging suspicious emails or websites.

8. Ongoing Monitoring and Model Updating: Phishing techniques evolve over time, so it is essential to continuously monitor the system's performance and update the model periodically. Incorporate feedback mechanisms that allow users to report false positives or false negatives, which can be used to retrain and improve the model.

By developing a machine learning-based phishing detection system, organizations and individuals can enhance their defense against phishing attacks, reduce the risk of data breaches, and protect sensitive information. However, it's important to note that no solution can guarantee 100% accuracy, so it's crucial to combine it with user education and other security measures to create a robust defense against phishing threats.

Literature Survey

S.No.	Title	Methodology	Findings
1	Chapla, H., Kotak, R., & Joiser, M. (2019, July). A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier. In *2019 International Conference on Communication and Electronics Systems (ICCES)* (pp. 383-388). IEEE.	Web mining also includes web content mining, web usage mining, and web structure mining.Then there's Web URL Mining. Email, websites, and malware are the most common sources of phishing assaults. In this paper they designed a framework of phishing detection using URL. To overcome the problem of phishing we design a framework to detect it using the fuzzy logic as a classifier. Using MATLAB tool author codes the program which can extract the features from the entered URL. Using different feature sets, out of them were author extract some feature which can based on URL of website.	The set of features works well for detecting web phishing. The model was built using both phishing and genuine URLs, and it included the features gathered from both. The fuzzy classifier is built in MATLAB in this study, with the top results of 91.46 percent accuracy.

2	Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., & Bindhumadhava, B. S. (2020, January). Phishing website classification and detection using machine learning. In 2020 international conference on computer communication and informatics (iccci) (pp. 1-6). IEEE.	Comparing different machine learning techniques for the phishing URL classification task and achieved the highest accuracy of 98% for Naïve Bayes Classifier with a precision=1, recall = 0.95 and F1-Score= 0.97. A URL's lexical structure may reveal secret information about the URL. The protocol name, such as HTTP or HTTPs, is the first part of a URL. The FQDN (fully qualified domain name) is the full domain name of the web server, which is then translated into an IP address by DNS servers. The domain name is made up of a second-level domain (SLD) followed by the top level domain (TLD) to which it is assigned. A domain name is a registered name that is unique across the Internet and is registered with a domain registrar.	The performance metrics data from experiment shows that all used classifiers in the experiment are suitable suitable for the Phishing URL detection tasks. The classifiers Random Forest and Gaussian Naïve Bayes classifiers result in better accuracies of about 98%.
5	Zabihimayvan, M., & Doran, D. (2019, June). Fuzzy rough set feature selection to enhance phishing attack detection. In 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-6). IEEE.	Fuzzy Rough Set (FRS) theory as a tool to select most effective features from three benchmarked data sets. The selected features are fed into three often used. classifiers for phishing detection.To evaluate the FRS feature selection in developing a generalisable phishing detection, the classifiers are trained by a separate out of sample data set of 14,000 website samples. Rough Set (RS) theory, developed by Pawlak, is a method to decide how to separate a set of data, with a decision. boundary and an indiscernibility relation R.	Maximum performance of universal features (93%) competes with the maximum FRS performance(95%) over UCI1and Mendeley.

6	Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., & Zhu, T. (2018). Webphishing detection using a deep learning framework. Wireless Communications and Mobile Computing, 2018.	This paper mainly focuses on applying a deep learning framework to detect phishing websites. This paper first designs two types of features for web phishing: original features and interaction features. We define two kinds of features to detect web phishing, and they are an original feature and interactive feature. Detection Based on DBN. DBN is one of the deep learning models, each of which is a restricted type of Boltzmann machine that contains a layer of visible units that representing the data.	A detection model based on Deep Belief Networks (DBN) is then presented. The test using real IP flows from ISP (Internet Service Provider) shows that the detecting model based on DBN can achieve an approximately 90% true positive rate and0.6% false positive rate.
7	Aliyu, A. Y. L., Saudi, M. M., & Abdullah, I. (2017). A Review and Proof of Concept for Phishing Scam Detection and Response using Apoptosis. International Journal Of Advanced Computer Science And Applications.	Systematic review analysis on existing works related with the phishing detection and response techniques together with apoptosis have been further investigated and evaluated. Furthermore, one case study to show the proof of concept how the phishing works is also discussed in this paper. • Proof Point • CANTINA • Auntie Tuna • PhiGARO Anomaly Based Phishing detectiontool	Reverse engineering process and dynamic analysis were conducted to analyse the code using the architecture. The reverse engineering was performed in a controlled lab environment by using Windows 10 as the host for installing the testing tools. Windows 7 was used as the client and windows Server 2012(Mail Server) was utilised as the server. The outgoing network was not required. Further analysis showed that the phishing dataset attachment was able to successfully create a thread in which stolen information could be sent through. It also showed a buffer overflow in the result section which might cause loading of files to be slow with the following pathat the registry.

S.No.	Title	Methodology	Findings
8	Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: a literature survey. IEEE Communications Surveys & Tutorials, 15(4), 2091-2121.	This paper aims at surveying many of the recently. Proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also presented, such as: detection, offensive defence, correction, and prevention, which we belief is critical to present where the phishing detection techniques fit in the overall mitigation process. • User training approaches • Software classification approaches • Offensive defence • Correction • Prevention	The study concludes that Anti-Phish Phil training material reduced F N rate from 46% to 29%, which is not enough evidence to assume that it would also complement software solutions that, for example, achieve a FN rate of less than 1%. The unanswered question is what is the percentage of overlap between the classification performed by end-users following a user training phase, and the classification performed by a software classifier? If the overlap is 100%,then the addition of user training can be redundant and will not be worth the added cost and complexity. However, if the overlap is less than 100%, then they can be complementary to each other however, such a study is not available in the public literature.
9	Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., & Zhu, T. (2018). Web phishing detection using a deep learning framework. Wireless Communications and Mobile Computing, 2018.	This paper mainly focuses on applying a deep learning framework to detect phishing websites. This paper first designs two types of features for web phishing: original features and interaction features. We define two kinds of features to detect web phishing, and they are an original feature and interactive feature. Detection Based on DBN. DBN is one of the deep learning models, each of which is a restricted type of Boltzmann machine that contains a layer of visible units that representing the data.	A detection model based on Deep Belief Networks (DBN) is then presented. The test using real IP flows from ISP (Internet Service Provider) shows that the detecting model based on DBN can achieve an approximately 90% true positive rate and 0.6% false positive rate.

S.No.	Title	Methodology	Findings
10	Korkmaz, M., Sahingoz, O. K., & Diri, B. (2020, July). Detection of phishing websites by using machine learning-based URL analysis. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE.	Suggested a machine learning based phishing detection system that analyses URLs using eight distinct algorithms and compares the findings to previous studies using three distinct datasets. The experimental results show that the proposed models operate exceptionally well, with a high success rate. Using many algorithm technique to find the best accuracy algorithm for Web phishing. Here are some algorithm follows : • ANN • SVM • DT • NB • XGBOOST	In experiments conducted in first, it is reported that the model with the highest accuracy rate was obtained with RF algorithm in all datasets. This study was named as Experiment 2. In Experiment 2, the highest accuracy rate results were obtained with RF algorithm too. Dataset-1, 1.55% improvement in Dataset-2, and 0.98% improvement in Dataset-3 were achieved.
11	Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007, October). A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit* (pp. 60-69).	1. Logistic Regression Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction. We briefly present the classification methods that are used in our comparative study. Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction. As a member of generalized linear models it typically uses the logit function. Xp), y is the binary response variable, and β is a p × 1 vector of regression parameters. Logistic regression performs well when the relationship in the data is approximately linear. It performs poorly if complex nonlinear relationships exist between the variables. It requires more statistical assumptions before being applied than other techniques. The prediction rate gets affected if there is missing data in the data set	To find the error rate for each classifier we calculate the average error rate in all sub-samples during cross validation. To find the error rate for each classifier we calculate the average error rate in all sub-samples during cross validation. The WErr when $\lambda = 9$ for all classifiers is calculated by taking the average of all sub-samples during cross validation. It is noticeable that the average error rate changes after penalizing false positives more than false negatives

S.No.	Title	Methodology	Findings
12	Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, *117*, 345-357.	Pro device with 2.7 GHz Intel Core i5 processor and 8 GB of 1867 MHz DDR3 RAM. 10-fold Cross Validation and the default parameter values of all algorithms were used during the tests. Each test set is executed with seven different machine learning algorithms. The confusion matrix for the tested learning algorithms is constructed as depicted in Table 4 (Only the best test type (NLP based features, Word Vectors or Hybrid) is listed). By using the values in confusion matrix, 4 different statistics as precision, sensitivity, f-measure, and accuracy are calculated to measure the usefulness and efficiency of the algorithms. These statistics, whose formulation is depicted in Eqs. (2–5), are important for making a comparison between the tested machine These statistics, whose formulation is depicted in Eqs. (2–5), are important for making a comparison between the tested machine	We have used seven different classification algorithms (Naive Bayes, Random Forest, kNN (n = 3), Adaboost, K-star, SMO and Decision Tree) as machine learning mechanism of the proposed system and compared their performances. Efficiency and good performance, it is preferred in lots of application areas such as classification of texts, detection of spam emails/intrusions, etc. It is based on the Bayes theorem, which describes the relationship of conditional probabilities of statistical quantities. It is based on the assumption of independence between the attribute values.

13	Shahrivari, V., Darabi, M. M., & Izadi, M. (2020). Phishing Detection Using Machine Learning Techniques. *arXiv preprint arXiv:2009.11116*.	For evaluating phishing classification performance we use accuracy(acc) recall(r), precision(p), F1 score, test time, and train time of classifiers. Recall measures the percentage of phishing websites that the model manages to detect (models effectiveness). Precision measures the degree to which the phishing detected websites are indeed phishing (models safety). F1 score is the weighted harmonic mean of precision and recall. Let NL→L be the number of legitimate websites classified as legitimate, NL→P be the number of legitimate websites misclassified as phishing, NP →L be the number of phishing misclassified as legitimate and NP →P be the number of phishing websites classified as phishing.	In this research, we have implemented and evaluated twelve classifiers on the phishing website dataset that consists of 6157 legitimate websites and 4898 phishing websites. The examined classifiers are Logistic Regression, Decision Tree, Support Vector Machine, Ada Boost, Random Forest, Neural Networks, KNN, Gradient Boosting, and XGBoost. According to our result in Table III, we get very good performance in ensembling classifiers namely, Random Forest, XGBoost both on computation duration and accuracy. The main idea behind ensemble algorithms is to combine several weak learners into a stronger one, this is perhaps the primary reason why ensemblebased learning is used in practice for most of the classification problems.

S.No.	Title	Methodology	Findings
14	Almseidin, M., Zuraiq, A. A., Al-Kasassbeh, M., & Alnidami, N. (2019). Phishing detection based on machine learning and feature selection methods.	The 10-fold cross-validation technique is utilized in testing the models for the reason that it minimizes the estimation variance. By using this technique, the training dataset should be divided into 10 subsets, each of these subsets must be tested in the remaining nine subsets. Every test subset is employed once a time in all 10 repetitions. Algorithm J48 J48+infogain+top J48+infogain+top J48+infogain+top J48+ reliefF +top J48+ reliefF +top J48+ reliefF +top	For analysis and comparing between used classifiers, Weka 3.8.3 has been utilized. Weka is a set of machine learning algorithms used for different data mining functions such as data preparation, classification, regression, clustering, association rules mining, and visualization. Two different feature selection algorithms have been used in this study: InfoGain and ReliefF.
15	Alauthman, M., Almomani, A., Alweshah, M., Omoush, W., & Alieyan, K. (2019). Machine learning for phishing detection and mitigation. In *Machine Learning for Computer and Cyber Security* (pp. 48-74). CRC Press.	One of the methods used by phishers is represented in searching for victims and directing them to a fake websites. Through such websites, victims are asked to reveal their confidential information. However, there are learning-based methods for spam filtering and are characterized with several features. These features are classified into three types—latent topic model features, basic features and dynamic Markov chain features	It is a method of getting confidential information through dishonest e-mails that seem to be legitimate. We have a chapter on protection against these phishing e-mail attacks. This chapter improves the understanding of the phishing e-mails downside, this answer area and also the future scope to filter phishing e-mails. Approaches given within the literature still have abundant limitations regarding accuracy or performance, particularly with !zero day" phishing e-mail attacks. Most classifiers will not establish phishing e-mail area primarily based on supervised learning, i.e., they have to learn before they will find a replacement attack.

16	Ankit Kumar Jain, B. B. Gupta. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP Journal on Information Security (2016) 2016:9	In this paper, authors propose a novel approach to protect against phishing attacks using auto-updated white-list of legitimate sites accessed by the individual user. Their proposed approach has both fast access time and high detection rate. When users try to open a website which is not available in the white-list, the browser warns users not to disclose their sensitive information. Furthermore, our approach checks the legitimacy of a webpage using hyperlink features. For this, hyperlinks from the source code of a webpage are extracted and apply to the proposed phishing detection algorithm.	Experimental results show that the proposed approach is very effective for protecting against phishing attacks as it has 86.02 % true positive rate while less than 1.48 % false negative rate. Moreover, our proposed system is efficient to detect various other types of phishing attacks (i.e., Domain Name System (DNS) poisoning, embedded objects, zero-hour attack).
17	Ian Fette, Norman Sadeh, Anthony Tomasic. Learning to Detect Phishing Emails. WWW 2007 / Track: Security, Privacy, Reliability, and Ethics Session: Passwords and Phishing	PILFER, is a machine-learning based approach to classification. In this paper They present a collection of features that has been identified as being particularly successful at detecting phishing, given the current state of attacks. We expect that over time, as the attacks evolve, new sets of features will have to be identified combining information from both internal or external sources.	In this paper, they have shown that it is possible to detect phishing emails with high accuracy by using a specialized filter, using features that are more directly applicable to phishing emails than those employed by general purpose spam filters. Although phishing is a subset of spam, it is characterized by certain unique properties that we have identified

S.No.	Title	Methodology	Findings
18	Andr´e Bergholz, Gerhard Paaß, Frank Reichartz, Siehyun StrobelJeong-Ho Chang. Improved Phishing Detection using Model-Based Features. Conference Paper · January 2008, DBLP	The authors propose advanced email features generated by adaptively trained Dynamic Markov Chains and by novel latent Class-Topic Models. On a publicly available test corpus classifiers using these features are able to reduce the number of misclassified emails by two thirds compared to previous work.	They employed statistical classification methods to classify emails as legitimate (ham) or phishing emails. We introduce two new typesof features generated by adaptive Dynamic Markov Chains (DMC) and by latent Class-Topic Models (CLTOM).
19	Neda Abdelhamid; Fadi Thabtah; Hussein Abdel-jaber. Phishing detection: A recent intelligent machine learning comparison based on models content and features. 2017 IEEE InternationalConference on Intelligence and Security Informatics (ISI)	This article investigates ML techniques applicability to detect phishing attacks and describes their pros and cons. In particular, different types of ML techniques have been investigated to reveal the suitable options that can serve as anti-phishing tools. More importantly, we experimentally compare large numbers of ML techniques on real phishing datasets and with respect to different metrics.	The experimental results show that Covering approach models are more appropriate as anti-phishing solutions, especially for novice users, because of their simple yet effective knowledge bases in addition to their good phishing detection rate.
20	Website Phishing Detection using Heuristic Based Approach. IRJET Journal Volume: 03 Issue: 05 | May-2016	This paper proposes the model which identify the phishing site. System first extracts the features which clearly differentiate that whether website is phished or legitimate. Then they apply this feature to machine learning techniques it will identify website are phished or legitimate. In this way it will help towards our society.	In this paper, heuristic-based phishing detection technique proposed that employs URL- based features. Additionally, classifiers generated through machine learning algorithms and identify the legitimate and phishing websites. System have used SVM which showed accuracy of 96% and very low false- positive rate. The proposed model can reduce damage caused by phishing attacks because it can detect new and temporary phishing sites. System also implemented decision tree algorithm and generated tree for it.

PROPOSED METHODOLOGY

We propose that machine learning be used to overcome the limitations of traditional phishing detection methods. Because large volumes of data on phishing attack patterns are readily available, the problem of phishing detection is a perfect target for machine learning solutions. The main idea is to utilise machine learning algorithms on a dataset of phishing pages to create a model that can be used to determine if a given web page is a phishing page or a legitimate webpage in real time. We plan to turn the learned model into a software application that can be simply deployed to end users to counteract phishing attacks. For this, we choose to create a Chrome extension using JavaScript to design a machine learning algorithm from scratch. We will be able to quickly publish the learnt model on the Chrome Web Store, where anyone can download and utilise our phishing detection solution.

When selecting a machine learning method for our product, we must consider three criteria in order to complete this project successfully. First, the trained model's accuracy should be good, as a product utilised by end customers in the actual world should not produce incorrect results. Second, the algorithm being used should be able to produce classifications in real time, which means it should have a very short execution time and utilise very few computer resources. Third, while selecting a machine learning algorithm for the problem of phishing detection, false positives and false negatives must be taken into account. This is due to the fact that a user should not be induced to assume that a phishing website is authentic. As a result, when choosing a phishing detection classifier, we should consider these three restrictions.

DATASET

The training dataset for our project comes from the UCI Machine Learning repository's "Phishing Websites Data Set." There are 11,055 items in the database, with 6157 phishing scams and 4898 valid ones. Each instance contains 30 traits that are commonly linked with phishing or suspicious web pages, such as the inclusion of an IP address in the URL domain or the existence of JavaScript code that modifies the information in the web browser address bar. A rule is assigned to each feature. If the rule is met, we consider it to be a sign of phishing, unless it is otherwise valid. Only discrete values were included in the dataset after it was standardised. Each instance's feature will have a value of '1' if the rule associated with that feature is satisfied, '0' if it is partially satisfied, and '-1' if it is unsatisfied.

The features represented by the training dataset can be classified into four categories;

A. Address bar based features
B. Abnormal based features
C. HTML & JavaScript based features
D. Domain based features

A) Address bar based features

- Using IP address
- Long URL to hide suspicious part
- Use of URL shortening services
- Use of "@" symbol
- Redirection with "//"
- Adding prefix or suffix separated by "-" to the domain
- Sub domains and multi sub domains
- HTTPS
- Domain Registration Length
- Favicon
- Using Non-Standard Port
- The Existence of "HTTPS" Token in the Domain Part of theURL

B) Abnormal based features

- RequestURL
- URL portion of anchor tag
- Links in <meta>, <script> and <link> tags
- Server Form Handler (SFH)
- Submitting Information to Email
- AbnormalURL

C) HTML and Javascript based features

- Status bar customization
- Disabling right click option
- Using pop-up window
- IFrameRedirection

D. Domain based Features

- Age of Domain
- DNS Record
- Website Traffic
- PageRank
- Google Index
- Number of Links Pointing to Page
- Statistical-Reports BasedFeature

ML IMPLEMENTATION

We created and evaluated supervised machine learning algorithms on the training dataset. The following algorithms were selected based on how successfully they handled classification issues. The dataset was split between training and test sets with a ratio of 7:3. We report the findings of our experiment in the Results section.

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.

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, how- ever, 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.

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. Here, primarily, the hyperplanes, which satisfy the constraints are identified and only for the valid hyperplanes we tend to maximise the margin. The arithmetic computation involved is simplified using the dual form. 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 objective for Soft-Margin SVM. 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.

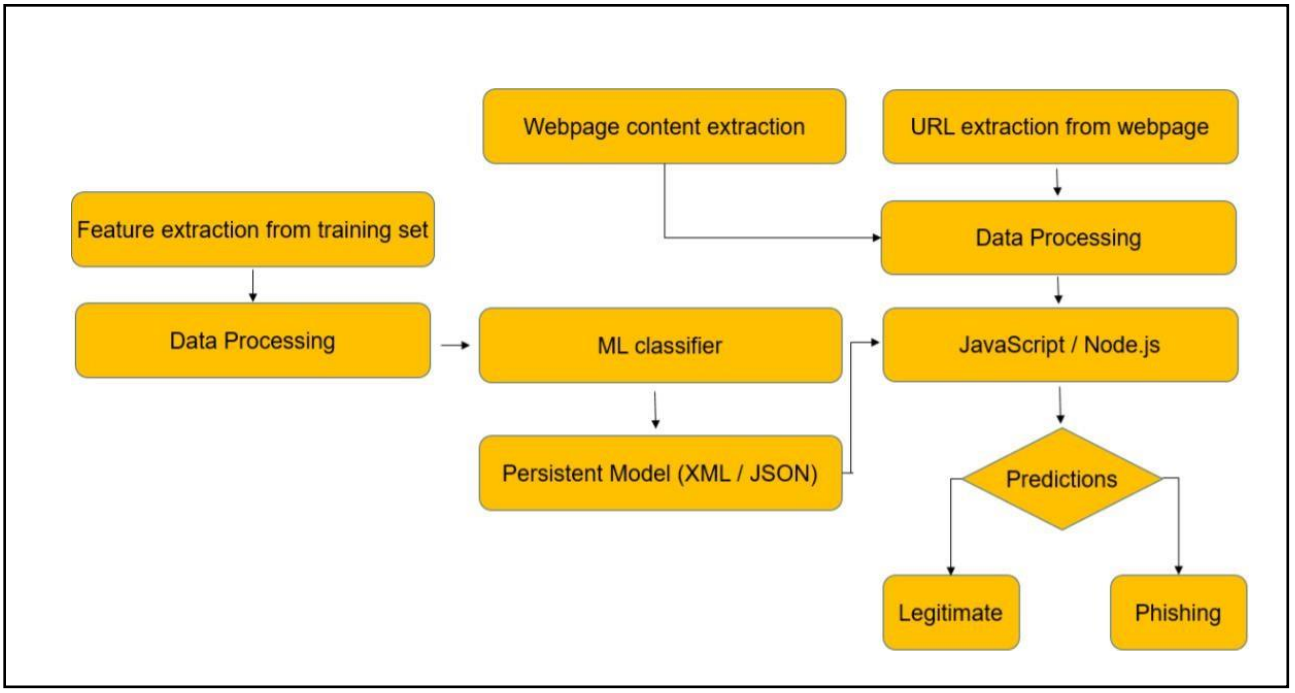
TECHNICALAPPROACH DETAILS

The suggested method seeks to create a phishing detection browser extension that uses cutting-edge machine learning techniques. Furthermore, because of the flexibility of margin and reduced computing complexity offered by SVM, the implementation uses an SVM trained persistent model to identify dangerous sites for classification issue statements. Because of its popularity, the extension is designed to support the Chrome browser in particular. Furthermore, extensions are web-independent because they combine many files into a single file for the user to download as a one-time action.

BROWSER EXTENSION SCHEMATICS

The solution entails training the model with available data using an SVM discriminative classifier, then passing the persistent model to the extension, which further predicts the authenticity of user-accessed websites and generates alerts to notify the legitimacy of the browsed URL on every page load. The solution combines the creation of a Python-based training stage with a JavaScript-based testing module. The training component was created in Python to make the best use of the complicated numeric computing modules that are available. Furthermore, because the testing stage is focused on site content and feature extraction with minimum heavy computation operations, the solution may have client-end computation performance latency.

Fig : Proposed Chrome extension implementation



During the project's initial research, the team looked at a few different methodologies and, after balancing the benefits, drawbacks, and bandwidth of the resources, decided on the persistent model passing methodology as the preferred methodology. One of the anticipated ways for establishing a Node.js-enabled testing component is to arrange the SVM model as a scalable Web API for consumption by the testing module.

Other options investigated throughout the implementation included Brython server-side architecture, which allows python to operate in the browser, and Rapydscript client-side architecture, which allows python to be compiled into javascript. The solution has been created utilising a Python-based training module due to the computation advancements afforded by Python over Brython/Rapydscript.

ALGORITHM DETAILS

The Chrome extension follows Google's guidelines and is composed of three files: manifest.json, content.js, and background.js. To the Chrome browser, the manifest file contains all meta data information about the extension. Furthermore, it lists all of the files and other resources linked with the extension. After the extension is installed, the content.js file is loaded on every page in the Chrome browser. It is, however, an unprivileged module with direct access only to the DOM components and requires supporting files to communicate with external APIs and manipulate the browser user experience. Background.js is a supplementary file that assists the content script with these interactions, also known as message forwarding.

Multiple functions have been implemented in the content.js script for web-content and URL feature extraction. Below are the detailsused 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 characters
- isAlphaNumericURL(): Check for alphanumeric '@' in URL
- isRedirectingURL(): Verify if '//' existing within the URL morethan once
- isHypenURL(): Check for presence of '-' adjacent to domainname 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 theURL 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 on mouseover manipulates the status bar display
- isIframePresent(): Identify sites, which exhibit iframes in theDOM

RESULTS

On the phishing dataset, this project compares the performance of all of the classifiers presented . We tested these algorithms on 3317 test samples using a variety of performance criteria, and the results are reported here with graphs.

	Predicted Phishing URLs	Predicted Legitimate URLs
Ground Truth Phishing URLs	1249 (37.7%)	162 (4.9%)
Ground Truth Legitimate URLs	182 (5.5%)	1680 (50.6%)

Table : Random Forest Confusion Matrix

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

	Predicted Phishing URLs	Predicted Legitimate URLs
Ground Truth Phishing URLs	1205 (36.3%)	250 (7.5%)
Ground Truth Legitimate URLs	170 (5.1%)	1692 (51.0%)

Table : Artificial Neural Network Confusion Matrix

This table shows the confusion matrix for Artificial neural networks. With 1205 true positives, 170 false positives, 250 false negatives and 1692 true negatives.

	Predicted Phishing URLs	Predicted Legitimate URLs
Ground Truth Phishing URLs	1293 (39.0%)	206 (6.2%)
Ground Truth Legitimate URLs	131 (3.9%)	1731 (52.2%)

Table : SVM Confusion Matrix

This table shows the confusion matrix for Artificial neural networks. With 1293 true positives, 131 false positives, 206 false negatives and 1731 true negatives.

Algorithms	Accuracy(%)	Specificity(%)	Sensitivity(%)
Artificial Neural Networks	87.34	91	83
Random Forests	89.61	90	86
SVM	89.93	93	89

Table : Performance matrix of classifiers

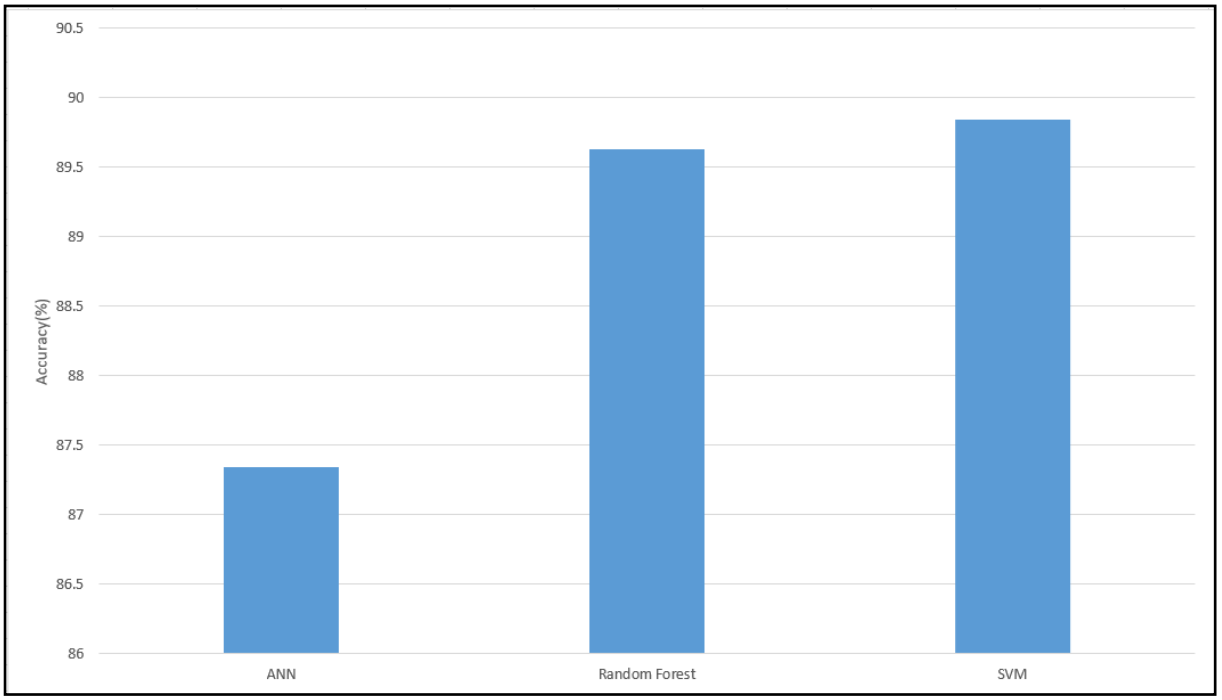


Fig 1 : Accuracy of classifiers

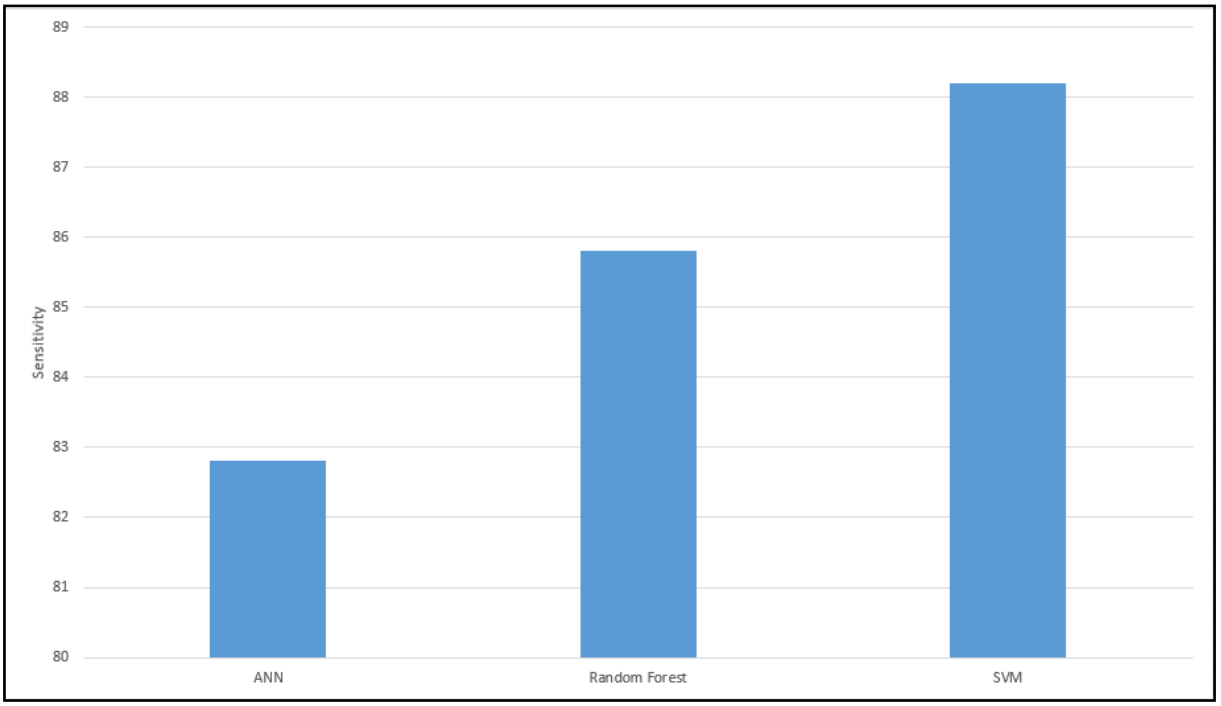


Fig 2 : Sensitivity of classifiers

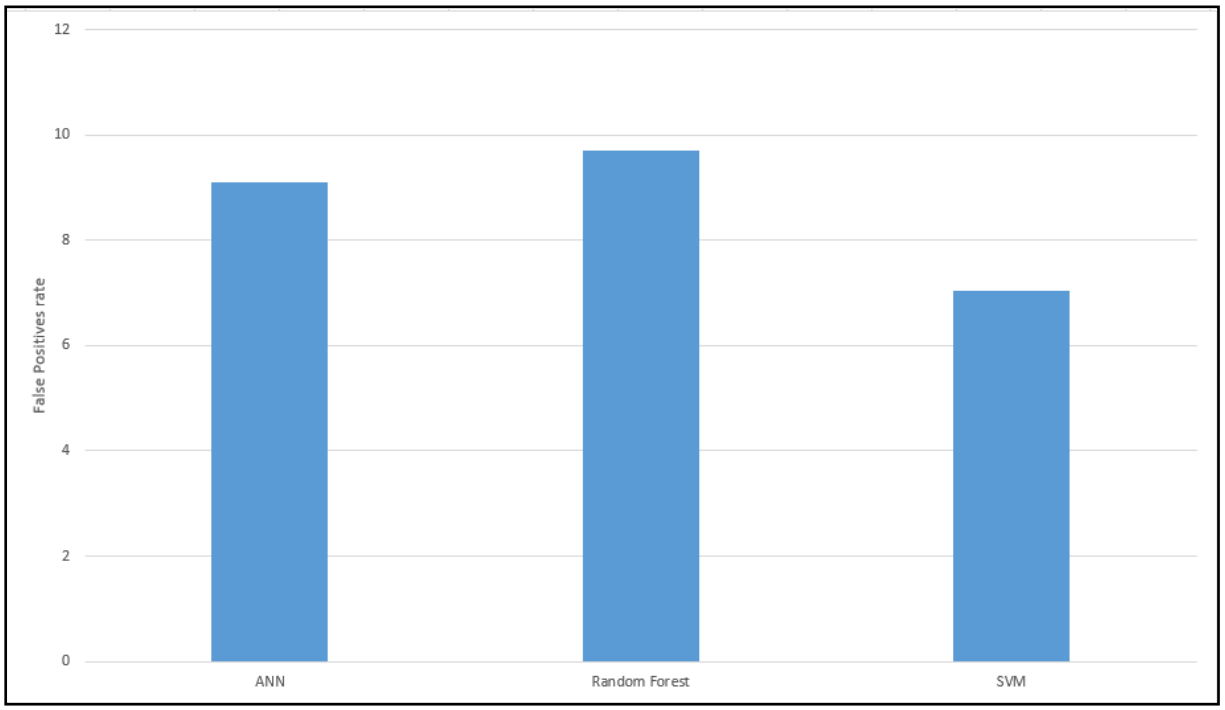
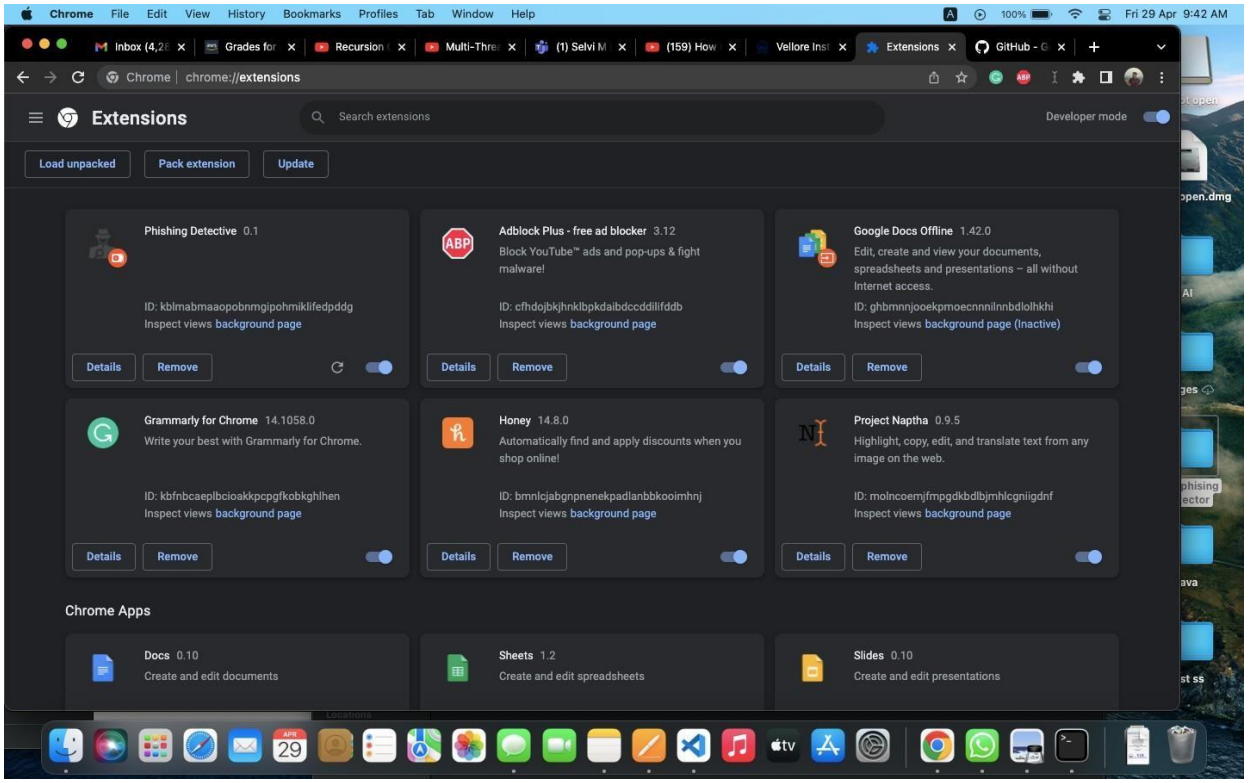
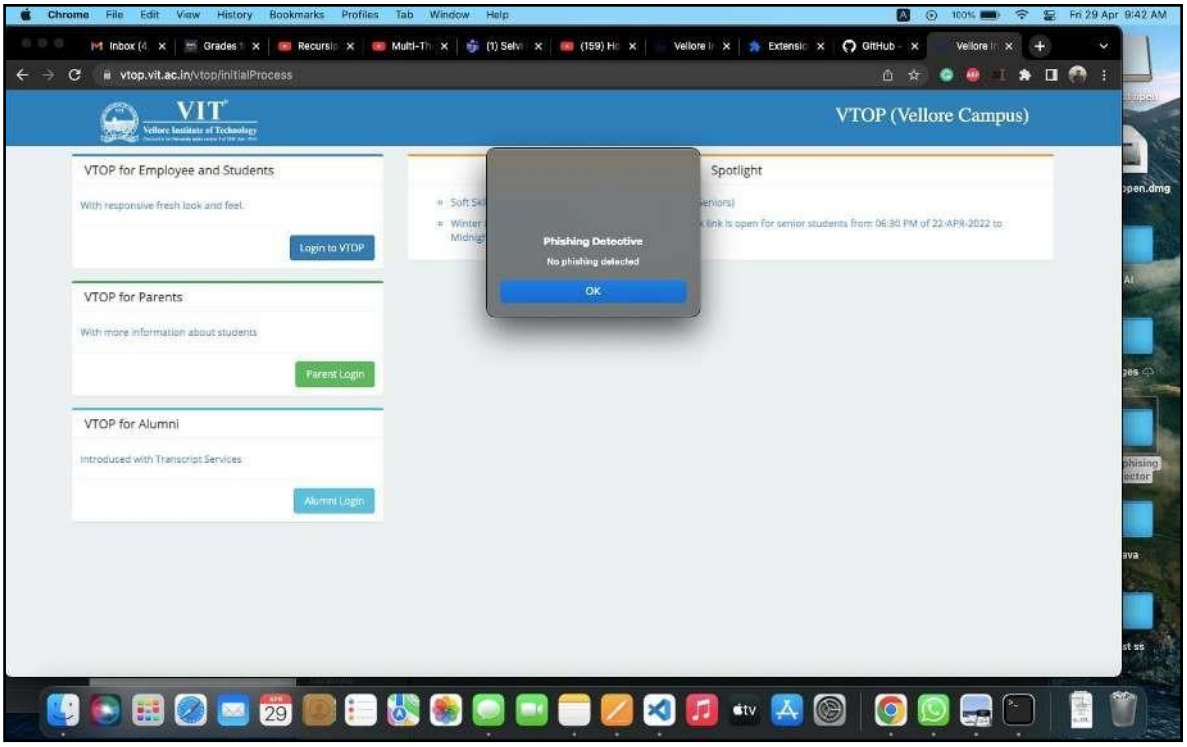


Fig 3 : False Positives of classifiers

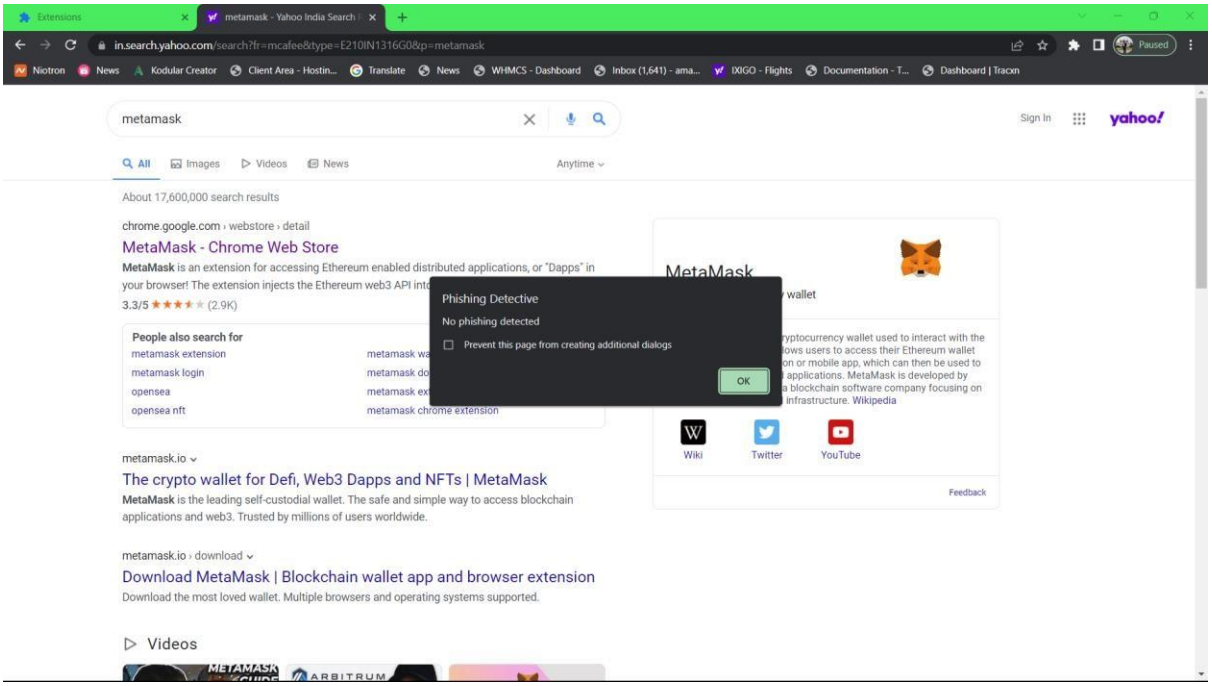
The Figure 1 above shows the accuracy of each classifier evaluated on 3317 test samples. From the above figure, it can be seen that SVM outperforms all the other algorithms based on accuracy in detection of Phishing URL. Also, Figure 2 shows the sensitivity of each classifier. Here, sensitivity refers 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 3 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.



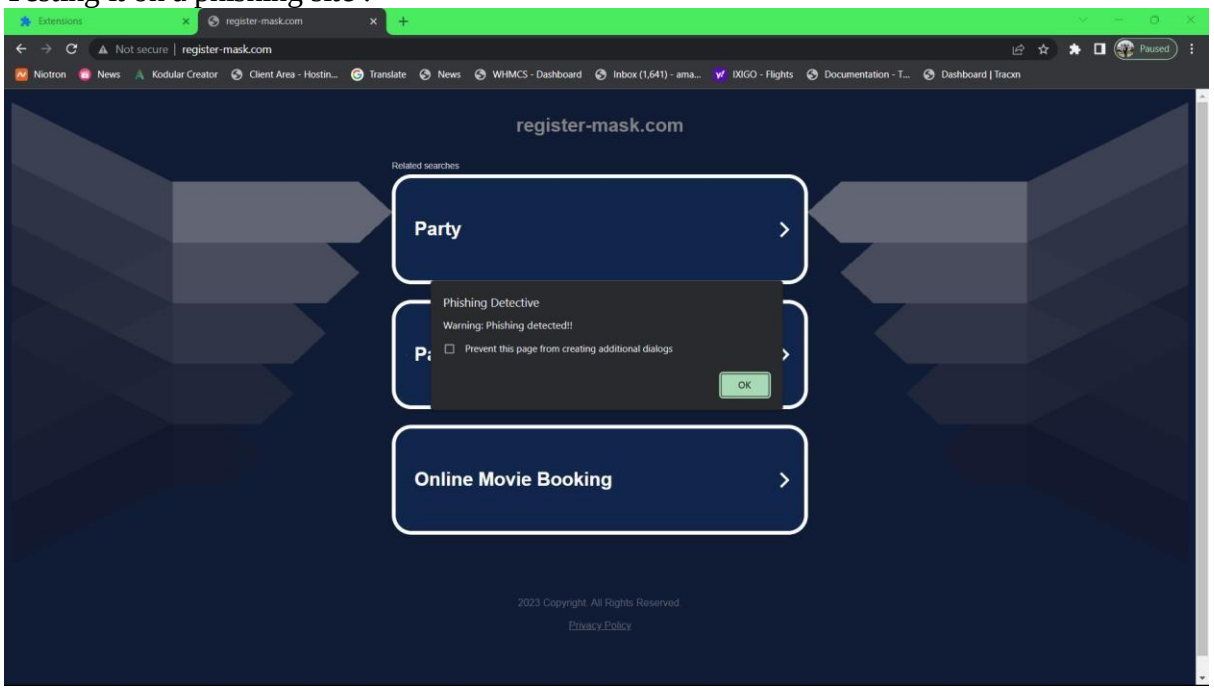
Testing it on a legitimate website :

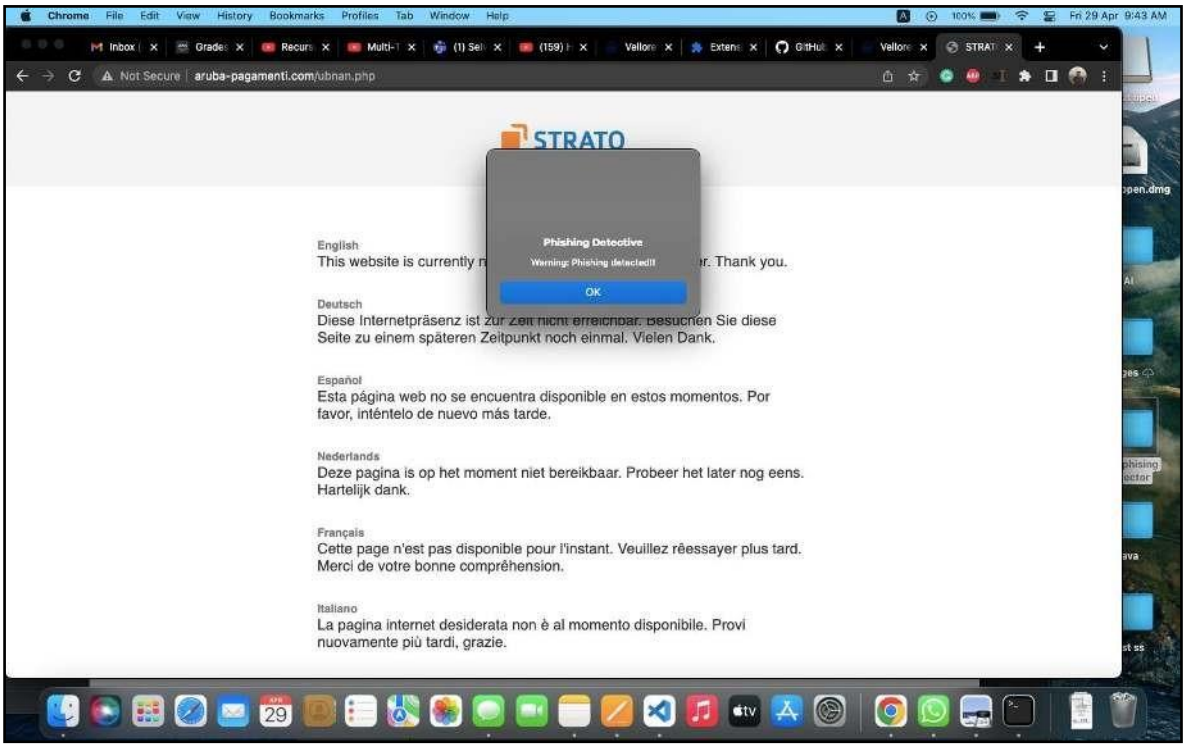


In the above screenshot you can see that when we opened VTop (a non-phishing website), there was a pop up in the top middle of the screen showing that there was “No phishing detected” which shows that the website youtube is safe and secure to browse.



Testing it on a phishing site :





In the above screenshot, we took a malicious website and as you can see that the pop-up now says “Warning : Phishing detected” which means thatthe phishing is detected and the website is not safe.

CONCLUSION

This leads us to the conclusion that phishing is a serious threat to the security and safety of the internet, and phishing detection is a major area of concern. We examined several of the common phishing detection techniques, including blacklist and heuristic evaluation approaches, and their drawbacks. We tested three machine learning techniques on the 'Phishing Websites Dataset' from the UCI Machine Learning Repository and studied the outcomes. After selecting the best algorithm based on performance, a Chrome plugin for detecting phishing websites was developed. Our phishing detection technology may be easily used by end users thanks to the plugin. We intend to develop the phishing detection system as a scalable web service with online learning in the coming improvements so that new phishing attack patterns can be readily taught and our models' accuracy can be enhanced with better feature extraction.

APPENDIX

Engineering Module(contains code related to extension) :Background.js -

```
chrome.extension.onRequest.addListener(function(prediction){
      if (prediction == 1){
            alert("Warning: Phishing detected!!");
      }
      else if (prediction == -1){
            alert("No phishing detected");
      }
});
```

Content.js -

```
var
testdata;
var
prediction;

function predict(data,weight){
      var f = 0;
      weight =
[3.33346292e-01,-1.11200396e-01,-7.77821806e-01,1.11058590e-01,3.8
9430647e-01,1.99992062e+00,4.44366975e-01,-2.77951957e-01,-6.00531
647e-05,3.33200243e-01,2.66644002e+00,6.66735991e-01,5.55496098e-0
1,5.57022408e-02,2.22225591e-01,-1.66678858e-01];
      for(var j=0;j<data.length;j++) {
         f += data[j] * weight[j];
      }
      return f > 0 ? 1 : -1;
}

function isIPInURL(){
      var reg =/\d{1,3}[\.]{1}\d{1,3}[\.]{1}\d{1,3}[\.]{1}\d{1,3}/;
      var url = window.location.href

      if(reg.exec(url)==null){
            console.log("NP");
            return -1;
      }
      else{
            console.log("P");
            return 1;
      }
}

function isLongURL(){
    var url = window.location.href;
      if(url.length<54){
            console.log("NP");
            return -1;
      }
      else if(url.length>=54 && url.length<=75){
            console.log("Maybe");
            return 0;
      }
      else{
```

```
            console.log("P");
            return 1;
        }
    }
    function isTinyURL(){
      var url = window.location.href;
        if(url.length>20){
            console.log("NP");
            return -1;
        }
        else{
            console.log("P");
            return 1;
        }
    }
    function isAlphaNumericURL(){
        var search ="@";
        var url = window.location.href;
        if(url.match(search)==null){
            console.log("NP");
            return -1;
        }
        else{
            console.log("P");
            return 1;
        }
    }
    function isRedirectingURL(){
        var reg1 = /^http:/
        var reg2 = /^https:/
        var srch ="//";

        var url = window.location.href;
        if(url.search(srch)==5 && reg1.exec(url)!=null &&
    (url.substring(7)).match(srch)==null){
            console.log("NP");
            return -1;
        }
        else if(url.search(srch)==6 && reg2.exec(url)!=null &&
    (url.substring(8)).match(srch)==null){
            console.log("NP");
            return -1;
        }
        else{
            console.log("P");
            return 1;
        }
    }
    function isHypenURL(){
        var reg = /[a-zA-Z]\//;
        var srch ="-";
        var url = window.location.href;
        if(((url.substring(0,url.search(reg)+1)).match(srch))==null){
            console.log("NP");
            return -1;
      }
      else{
            console.log("P");
```

```
            return 1;
        }
    }
    function isMultiDomainURL(){
        var reg = /[a-zA-Z]\//;
        var srch ="-";
        var url = window.location.href;
        if((url.substring(0,url.search(reg)+1)).split('.').length < 5)
    {
            console.log("NP");
            return -1;
      }
      else{
            console.log("P");
            return 1;
        }
    }
    function isFaviconDomainUnidentical(){
        var reg = /[a-zA-Z]\//;
        var url = window.location.href;
        if(document.querySelectorAll("link[rel*='shortcut
    icon']").length>0){
            var faviconurl =
    document.querySelectorAll("link[rel*='shortcut icon']")[0].href;
            if((url.substring(0,url.search(reg)
    +1))==(faviconurl.substring(0,faviconurl.search(reg)+1))){

                console.log("NP");
                return -1;
            }
            else{
                console.log("P");
                return 1;
            }
        }
        else{
            console.log("NP");
            return -1;
        }
    }

    function isIllegalHttpsURL(){
        var srch1 ="//";
        var srch2 = "https";
        var url = window.location.href;
        if(((url.substring(url.search(srch1))).match(srch2))==null){
            console.log("NP");
            return -1;
      }
      else{
            console.log("P");
            return 1;
        }
    }
    function isImgFromDifferentDomain(){
```

```
    var totalCount = document.querySelectorAll("img").length
    var identicalCount = getIdenticalDomainCount("img");
    if(((totalCount-identicalCount)/totalCount)<0.22){
        console.log("NP");
        return -1;
    }
    else if((((totalCount-identicalCount)/totalCount)>=0.22) &&
(((totalCount-identicalCount)/totalCount)<=0.61)){
        console.log("Maybe");
        return 0;
}
    else{
        console.log("P");
        return 1;
    }
}
function isAnchorFromDifferentDomain(){
    var totalCount = document.querySelectorAll("a").length
  var identicalCount = getIdenticalDomainCount("a");
  if(((totalCount-identicalCount)/totalCount)<0.31){
        console.log("NP");
        return -1;
    }

    else if((((totalCount-identicalCount)/totalCount)>=0.31) &&
(((totalCount-identicalCount)/totalCount)<=0.67)){
        console.log("Maybe");
        return 0;
}
    else{
        console.log("P");
        return 1;
    }
}
function isScLnkFromDifferentDomain(){
    var totalCount = document.querySelectorAll("script").length +
document.querySelectorAll("link").length
    var identicalCount = getIdenticalDomainCount("script") +
getIdenticalDomainCount("link");
    if(((totalCount-identicalCount)/totalCount)<0.17){
        console.log("NP");
        return -1;
    }
    else if((((totalCount-identicalCount)/totalCount)>=0.17) &&
(((totalCount-identicalCount)/totalCount)<=0.81)){
        console.log("Maybe");
        return 0;
}
    else{
        console.log("P");
        return 1;
    }
}

function isFormActionInvalid(){
    var totalCount = document.querySelectorAll("form").length
    var identicalCount = getIdenticalDomainCount("form");
```

```
        if(document.querySelectorAll('form[action]').length<=0){
                console.log("NP");
                return -1;
    }
        else if(identicalCount!=totalCount){
                console.log("Maybe");
                return 0;
    }
     else
    if(document.querySelectorAll('form[action*=""]').length>0){
                console.log("P");
                return 1;
        }
        else{
                console.log("NP");
                return -1;
        }
    }


    function isMailToAvailable(){
        if(document.querySelectorAll('a[href^=mailto]').length<=0){
                console.log("NP");
                return -1;
    }
        else{
                console.log("P");
                return 1;
        }
    }

    function isStatusBarTampered(){

    if((document.querySelectorAll("a[onmouseover*='window.status']").l
    ength<=0) ||
    (document.querySelectorAll("a[onclick*='location.href']").length<=0)){
                console.log("NP");
                return -1;
        }
        else{
                console.log("P");
                return 1;
        }
    }

    function isIframePresent(){
        if(document.querySelectorAll('iframe').length<=0){
                console.log("NP");
                return -1;
        }
        else{
                console.log("P");
                return 1;
        }
    }
```

```
function getIdenticalDomainCount(tag){
    var i;
    var identicalCount=0;
  var reg = /[a-zA-Z]\//;
    var url = window.location.href;
  var mainDomain = url.substring(0,url.search(reg)+1);
    var nodeList = document.querySelectorAll(tag);
    if(tag=="img" || tag=="script"){
      nodeList.forEach(function(element,index) {
          i = nodeList[index].src
          if(mainDomain==(i.substring(0,i.search(reg)+1))){
              identicalCount++;
      }
      });
    }


    else if(tag=="form"){
      nodeList.forEach(function(element,index) {
          i = nodeList[index].action
          if(mainDomain==(i.substring(0,i.search(reg)+1))){
              identicalCount++;
      }
      });
    }
    else if(tag=="a"){
      nodeList.forEach(function(element,index) {
          i = nodeList[index].href
          if((mainDomain==(i.substring(0,i.search(reg)+1))) &&
((i.substring(0,i.search(reg)+1))!=null) &&
((i.substring(0,i.search(reg)+1))!="")){
              identicalCount++;
      }
      });
    }
    else{
      nodeList.forEach(function(element,index) {
          i = nodeList[index].href
          if(mainDomain==(i.substring(0,i.search(reg)+1))){
              identicalCount++;
      }
      });
    }
    return identicalCount;
}

testdata =
[isIPInURL(),isLongURL(),isTinyURL(),isAlphaNumericURL(),isRedirec
tingURL(),isHypenURL(),isMultiDomainURL(),isFaviconDomainUnidentic
al(),isIllegalHttpsURL(),isImgFromDifferentDomain(),isAnchorFromDi
fferentDomain(),isScLnkFromDifferentDomain(),isFormActionInvalid()
,isMailToAvailable(),isStatusBarTampered(),isIframePresent()];

prediction = predict(testdata);
```

```
chrome.extension.sendRequest(prediction);
```

Manifest.json -

```
{
  "manifest_version": 2,
  "name": "Phishing Detective",
  "version": "0.1",
  "icons": {
          "16": "detective.png",
          "48": "detective.png",
            "128": "detective.png"

   },
  "background": {
          "scripts": ["background.js"]
   },
  "content_scripts": [
        {
          "matches": [
             "<all_urls>"
          ],
          "js": ["jquery-3.1.1.min.js","content.js"]
        }
   ]
}
```

Phishing.html -

```
<a href="http://www.google.com"
onmouseover="window.status='http://www.hpe.com';"
onmouseout="window.status='';">link here</a>

<a href="http://www.google.com" onclick="location.href='http://
www.hpe.com';return false">link here</a>
```

MLAlgorithm Evaluation :

```
#!/usr/bin/py

thonimport

time

def calculate_metrics(y_test,Y_predicted):

    from sklearn import metrics
    from sklearn.metrics import
classification_report,confusion_matrix

    accuracy = metrics.accuracy_score(y_test,Y_predicted)
    print("\naccuracy = "+str(round(accuracy * 100,2))+"%\n")
```

```
    confusion_mat = confusion_matrix(y_test,Y_predicted)

    print(confusion_mat)
    print(confusion_mat.shape)

    print("TP\tFP\tFN\tTN")
    for i in range(confusion_mat.shape[0]):
        # i means which class to choose to do one-vs-the-rest
calculation
        # rows are actual obs whereas columns are predictions

        TP = round(float(confusion_mat[i,i]),2) # correctly
labeled as i
        FP = round(float(confusion_mat[:,i].sum()),2) - TP #
incorrectly labeled as i
        FN = round(float(confusion_mat[i,:].sum()),2) - TP #
incorrectly labeled as non-i
        TN = round(float(confusion_mat.sum().sum()),2) - TP - FP -FN
        print(str(TP)+"\t"+str(FP)+"\t"+str(FN)+"\t"+str(TN))
        sensitivity = round(TP / (TP + FN),2)
        specificity = round(TN / (TN + FP),2)

    print("\tSensitivity : "+str(sensitivity)+"\t\tSpecificity:
"+str(specificity)+"\t\t")


    f_score = metrics.f1_score(y_test,Y_predicted)
    print(f_score)

def neural_network(dataset,class_labels,test_size):
    import warnings
    warnings.filterwarnings("ignore")
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.neural_network import MLPClassifier


    X = pd.read_csv(dataset)
    Y = pd.read_csv(class_labels)
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size= test_size, random_state=42)

    model = MLPClassifier(hidden_layer_sizes=(100),
activation='logistic',random_state = 42)
    model.fit(X_train,y_train)
    Y_predicted = model.predict(X_test)



    return y_test,Y_predicted


def random_forests(dataset,class_labels,test_size):
    import warnings
    warnings.filterwarnings("ignore")
    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
```

```
    from sklearn.ensemble import RandomForestClassifier
    from sklearn import metrics
    X = pd.read_csv(dataset)
    Y = pd.read_csv(class_labels)

    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size= test_size, random_state=42)
    model = RandomForestClassifier(n_estimators = 5, criterion =
'entropy',random_state = 42)
    model.fit(X_train,y_train)
    Y_predicted = model.predict(X_test)

    return y_test,Y_predicted

def support_vector_machines(dataset,class_labels,test_size):
    import warnings
    warnings.filterwarnings("ignore")
    import numpy as np
    from sklearn import svm
    import pandas as pd
    from sklearn.model_selection import train_test_split

    X = pd.read_csv(dataset)
    Y = pd.read_csv(class_labels)
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size= test_size, random_state=42)
    # 'rbf' value is the gaussian kernel, 'C' is the coefficient used
for regularization during training
    model = svm.SVC(kernel='rbf',C=2.0)
    model.fit(X_train,y_train)
    Y_predicted = model.predict(X_test)

    return

y_test,Y_predicteddef

main():

    dataset = "Dataset.csv"
    class_labels = "Target_Labels.csv"
    test_size = 0.3

    print("\nrunning neural networks...")
    start_time = time.time()
    y_test,Y_predicted =
neural_network(dataset,class_labels,test_size)
    calculate_metrics(y_test,Y_predicted)
    end_time = time.time()
    print("runtime = "+str(end_time - start_time)+" seconds")

    print("\nrunning random forests...")
    start_time = time.time()
    y_test,Y_predicted =
random_forests(dataset,class_labels,test_size)
    calculate_metrics(y_test,Y_predicted)
    end_time = time.time()
    print("runtime = "+str(end_time - start_time)+" seconds")
    print("\nrunning support vector machines...")
```

```
    start_time = time.time()
    y_test,Y_predicted =
support_vector_machines(dataset,class_labels,test_size)
    calculate_metrics(y_test,Y_predicted)
    end_time = time.time()
    print("runtime = "+str(end_time - start_time)+" seconds")


if __name__ == '__main__':
    start_time = time.time()
    main()
    end_time = time.time()
    print("runtime = "+str(end_time - start_time)+" seconds")
```

CONCLUSION

This leads us to the conclusion that phishing is a serious threat to the security and safety of the internet, and phishing detection is a major area of concern. We examined several of the common phishing detection techniques, including blacklist and heuristic evaluation approaches, and their drawbacks. We tested three machine learning techniques on the 'Phishing Websites Dataset' from the UCI Machine Learning Repository and studied the outcomes. After selecting the best algorithm based on performance, a Chrome plugin for detecting phishing websites was developed. Our phishing detection technology may be easily used by end users thanks to the plugin. We intend to develop the phishing detection system as a scalable web service with online learning in the coming improvements so that new phishing attack patterns can be readily taught and our models' accuracy can be enhanced with better feature extraction.

REFERENCES

I. Chapla, H., Kotak, R., & Joiser, M. “A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier.” In 2019 International Conference on Communication and Electronics Systems (ICCES) (pp. 383-388). IEEE.(2019)

II. Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., & Bindhumadhava, B. S. “Phishing website classification and detection using machine learning.” In 2020 international conference on computer communication and informatics (iccci) (pp. 1-6). IEEE. (2020)

III. Korkmaz, M., Sahingoz, O. K., & Diri, B. “Detection of phishing websites by using machine learning-based URL analysis.” In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE. (2020)

IV. Alswailem, A., Alabdullah, B., Alrumayh, N., & Alsedrani, A. “Detecting phishing websites using machine learning.” In 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS) (pp. 1-6). IEEE. (2019)

V. Zabihimayvan, M., & Doran, D. “Fuzzy rough set feature selection to enhance phishing attack detection.” In 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-6). IEEE. (2019)

VI. Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., & Zhu, T. “Web phishing detection using a deep learning framework.” Wireless Communications and Mobile Computing, 2018.

VII. Aliyu, A. Y. L., Saudi, M. M., & Abdullah, I. “A Review and Proof of Concept for Phishing Scam Detection and Response using Apoptosis.” International Journal Of Advanced Computer Science And Applications. (2017)

VIII. Khonji, M., Iraqi, Y., & Jones, A. “Phishing detection: a literature survey.” IEEE Communications Surveys & Tutorials, 15(4), 2091-2121. (2013)

IX. Korkmaz, M., Sahingoz, O. K., & Diri, B. “Detection of phishing websites by using machine learning-based URL analysis.”In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE. (2020)

X. Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. “A comparison of machine learning techniques for phishing detection”. In

Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit (pp. 60-69). (2007)

I. Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. “Machine learning based phishing detection from URLs.” Expert Systems with Applications, 117, 345-357. (2019)

II. Shahrivari, V., Darabi, M. M., & Izadi, M. “Phishing Detection Using Machine Learning Techniques.” arXiv preprint arXiv:2009.11116. (2020)

III. Almseidin, M., Zuraiq, A. A., Al- Kasassbeh, M., & Alnidami, N. “Phishing detection based on machine learning and feature selection methods”. (2019)

IV. Alauthman, M., Almomani, A., Alweshah, M., Omoush, W., & Alieyan, K. “Machine learning for phishing detection and mitigation.” In Machine Learning for Computer and Cyber Security (pp. 48-74). CRC Press. (2019)

V. Ankit Kumar Jain, B. B. Gupta. “A novel approach to protect against phishing attacks at client side using auto-updated white-list.” EURASIP Journal on Information Security (2016) 2016:9

VI. Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., & Zhu, T. (2018). “Web phishing detection using a deep learning framework.” Wireless Communications and Mobile Computing, 2018.

VII. Ian Fette, Norman Sadeh, Anthony Tomasic. “Learning to Detect Phishing Emails.” WWW / Track: Security, Privacy, Reliability, and Ethics Session: Passwords and Phishing (2007)

VIII. Andŕe Bergholz, Gerhard Paaß, Frank Reichartz, Siehyun Strobel Jeong-Ho Chang. “Improved Phishing Detection using Model- Based Features.” Conference Paper · January 2008, DBLP

IX. Neda Abdelhamid; Fadi Thabtah; Hussein Abdel-jaber. “Phishing detection: A recent intelligent machine learning comparison based on models content and features.” 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)

“Website Phishing Detection using Heuristic Based Approach.” IRJET Journal Volume: 03 Issue: 05 | May-2016