MASQUERADING WEBSITE RECOGNITION USING MACHINE LEARNING TECHNIQUE(S) VIA WEB-BROWSER

J COMPONENT PROJECT REPORT

for

INFORMATION SECURITY MANAGEMENT (CSE3502)

in

B.Tech (IT)

by

Soubhik Sinha (19BIT0303)

Muskan Agarwal (19BIT0169)

Prishita Raj (19BIT0288)

6th Sem, 2022 (3rd year)

Under the Guidance of

Dr. John Singh K

SITE

VIT®

Vellore Institute of Technology

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

1. TITLE AND TEAM DETAILS:

Title : Masquerading Website Recognition using Machine Learning Technique(s) via Web-Browser

Team Details: Soubhik Sinha (**19BIT0303**)

Muskan Agarwal (**19BIT0169**)

Prishita Raj (**19BIT0288**)

2. ABSTRACT:

We all are aware of various website attacks happening every then and so. Heavily used Websites like for Banking & Cloud - storage tops the list of easily getting hacked online resources and thus, fraudsters are fooling users by creating lookalike entities of the same to seek user credentials. Advances in Internet and cloud technology have resulted in a major expansion in electronic trade, which involves customers making purchases and transactions via the internet. This expansion leads to illegal access to sensitive information held by users, as well as harm to an organization's resources. Existing research shows that the phishing detection system's performance is restricted. An intelligent strategy to safeguard consumers from cyber-attacks is in high demand. Phishing assaults are especially vulnerable over the Internet because of its anonymous and unpredictable nature. The number of victims is increasing exponentially as a result of ineffective security systems. Phishing assaults are especially vulnerable over the Internet because of its anonymous and unpredictable nature. This project shall provide a considerable usage of Random Forest Classifier to mitigate the vulnerability level of user credentials & other vital online resources. Keeping in mind the lowering of mundaneness of the application & quick response , the activity will be executed within the browser itself - thus , the factor of network delay has no role to play - and also providing more reclusiveness to the user. The random forest classifier model will be trained via a dataset of vulnerable websites using Scikit-learn Library / Module. Advantages of browser-side execution and not including any distant server , can lead to greater emphasis on end-point application(s).

3. INTRODUCTION:

While the volume and sophistication of cybersecurity attacks continue to grow, social engineering techniques remain one of the simplest and most successful ways to get access to sensitive or secret information. Phishing is described as a type of social engineering that involves acting as a trustworthy organization or business and soliciting personal information from an individual or corporation via e-mails or malicious websites. While corporations could train personnel to spot phishing emails or links to assist guard against the aforementioned forms of attacks, software like HTTrack is widely available for anyone to replicate entire websites for their own reasons. As a consequence, even the most experienced users might be duped into disclosing private or sensitive information by visiting a rogue website that appears to be legitimate. As a result of the aforementioned issue, computer-based solutions to protect against phishing attempts, as well as user education, are required. A computer with such a solution would be able to recognise harmful websites and restrict people from engaging with them. The use of Uniform Resource Locators (URLs) is one way to identify phishing websites that aren't real (URLs). A URL is a document's worldwide address on the World Wide Web, and it is the most common way to find a document on the Internet. Even if the content of websites is replicated, websites may be utilized to identify legitimate sites from imposters.

Anti-virus companies have built a blacklist of harmful websites, which can be used as a remedy. The difficulty with this method is that the blacklist can never be complete since new malicious URLs appear on a regular basis. As a result, methods that can automatically classify a new, previously unknown URL as a phishing site or a valid one are required. Such solutions are often machine-learning-based techniques in which a system uses a model established from training sets of existing assaults to identify new phishing sites.

One of the major challenges in creating machine learning-based solutions for this problem is the scarcity of publicly available training data sets including phishing websites. As a result, research evaluating the efficacy of machine-learning algorithms based on existing data sets is required. This project tries to help meet that requirement. The purpose of this study is to assess the performance of widely used machine learning algorithms on the same phishing data set. We employ a data collection in which features from the data URLs have previously been extracted and the class labels have been made available. We used Random Forest Classifier to examine standard machine learning strategies for identifying URLs / Websites as spoofing entities.

4. MOTIVATION:

Phishing is a method of stealing people's personal information by deceiving them with bogus emails and websites. Phishing hinders people from doing their business via the internet. So, in order to tackle the threat and problem of phishing, we plan on successfully developing a phishing website detection system that focuses on client-side implementation with quick detection, so that users are informed before being phished.

We discovered via study and numerous literature surveys that comparable studies frequently employ web page attributes that are not practical to extract on the client side, resulting in detection being network-dependent. To address the aforementioned issue, we will build the system to only employ features that can be extracted on the client side, allowing it to deliver faster detection and greater privacy than other similar current systems.

5. OBJECTIVES:

The objective of the project is to convert the base paper into javascript so that it may be used as a browser plugin. Because javascript does not support many machine learning libraries, and because client computers' processing capability is restricted, the approach must be kept lightweight. The random forest classifier must first be trained on the phishing websites dataset using python scikit-learn, and then the learned model parameters must be exported into a portable format for usage in javascript.

6. PROBLEM STATEMENT:

Phishing is a method of stealing people's personal information by deceiving them with bogus emails and websites. Phishing hinders people from doing their business via the internet. Our problem statement is in order to tackle the threat and problem of phishing, developing a phishing website detection system that focuses on client-side implementation with quick detection, so that users are informed before being phished.

7. EXISTING SYSTEM:

I. LITERATURE SURVEY:

Title of the paper	Algorithm Used	Methodology Applied	Advantages	Issues/Drawbacks	Metrics Used to evaluate the work
Decision Tree Classifier for Classification of Phishing Website with Info Gain Feature Selection	Decision tree, random tree, random forest, Link Guard Algorithm	Comparative study	-The simplicity with which the feature correlations and interactions can be explained and interpreted. -The DT model is straightforward to analyse and understand since it generates simple IF–THEN statements.	-The DT does not support web - based learning and requires rebuilding the tree each time new samples are added; this necessitates a new process, which takes more time. -The classification result of the DT is low when compared to other ML approaches.	Accuracy, Sensitivity, Specificity
Phishing Website Detection using Machine Learning Algorithms	Machine learning algorithms.	Comparative study	-Machine Learning algorithms are adept at managing data that is multi-dimensional and multi-variety, and they can do so in dynamic or unpredictable environment. - -Machine Learning can analyse enormous amounts of data and uncover precise trends and patterns that	- A major limitation of this technology is that it is unable to detect zero-hour phishing attacks. - Machine Learning requires large data sets to train on, which must be inclusive, unbiased, and of high quality. They may have to wait for new data to be created at times.	Accuracy Score, False Negative Rate , False Positive Rate.

			people might miss.		
Detection of phishing websites using an efficient feature-based Machine learning framework	Random forest J48, logistic regression, Bayes network, multilayer perceptron, sequential minimal optimization, AdaBoostM1,SVM	Comparative study	- The developed techniques are highly efficient and robust, resulting in a significant boost in phishing detection. -The highest performance features are trained again over a Deep Neural Network (DNN) to determine the output status and distinguish between legal and phishing URLs, resulting in the best results.	- The detection rate can be increased by a factor of ten by inventing novel heuristics and upgrading calculating algorithms. -The heuristic technique has one important drawback: if the phishing web sites do not have heuristic elements, the detection rate drops; also, the technique can be readily avoided if the relevant algorithms or uncovered features are well-known ahead of time.	-Training Accuracy -Testing Accuracy

Phishing detection based Associative Classification data mining	Data mining algorithms	Comparative study	- Experiments using real data from various sources reveal that AC, specifically MCAC, detects phishing websites more accurately than other clever algorithms, resulting in the most accurate findings.	- The disadvantage of this strategy is that the blacklist typically cannot cover all phishing websites because adding a newly developed fraudulent website to the list takes time.	Accuracy
Survey on Malicious Web Pages	Batch machine learning algorithms	Comparative study and new proposed system.	-large batch gives stable results.	Mini-batch requires the configuration of an additional "mini-	Accuracy Score, False Negative

Detection Techniques			-The batched updates provide a computati-onally more efficient process than stochastic gradient descent.	batch size" hyperparameter for the learning algorithm.	Rate , False Positive Rate.
A new fast associative classification algorithm for detecting phishing websites	Data mining and machine learning Algorithms (Fast Associative Classification Algorithm)	-New proposed system	-It classified data using the association rule method, which has been demonstrated to be more accurate than the traditional method. -It uses a more efficient FP-tree than CBA, which uses less memory and space.	- It must deal with a support threshold sensitivity of the bare minimum. A high number of rules are generated when a low minimum support level is chosen.	Accuracy, Hamming Loss, Harmonic Mean, F1 evaluation.

Phishing Website Detection Using Ensemble Technique	Fast Associative Classification Algorithm, logistic model tree, machine learning algorithms.	Comparative study	The advantages of various algorithms can be combined using the ensemble technique, resulting in a single optimal output.	-A few more and various options for the classifiers within the voting procedure could increase this classifier's accuracy numbers.	Accuracy, True Positive Rate (*TPR*) False Positive Rate(FPR)
Phishing Website URL Detection using Machine Learning	A) logistic Regression B) Decision Tree C) Random Forest D) Adaboost E) Gradient Boosting F) Gaussian NB G) Fuzzy Pattern Tree classifier	Comparative Study	The accuracy of classification along with precision, recall and F1 Score were able to be determined on the examples in the testing set. The model could then be tested on real world phishing	Features such as favicons, meta tags, pop-up windows, redirect links on the web page were not considered as factors for determining the legitimacy of the websites. Number of training examples could	Accuracy, Precision, Recall, F1 score of the various algorithms were compared

			examples to determine its robustness.	have been increased to make it more robust	
Heuristic nonlinear regression strategy for detecting phishing websites	Two meta-heuristic algorithms used: ->Harmony Search (HS) ->Support Vector Machine (SVM)	Comparative study and a new-proposed system	The approach based on HS algorithm led to a high accuracy of 94.13% and 92.80% for train and test processes resp. The HS algo. allows for improvement of the method by allowing changes in the parameters or by being hybridized with a different algorithm.	One of the few drawbacks of the Harmony search algorithm is that in early iterations the value of PAR is low. Hence PA rule may not be able to satisfy the need of exploration of algorithm in early Iterations	Dynamic pitch adjustment rate of the harmony search algorithm as well as Accuracy and precision of predicting fraudulent websites using The Harmony search algorithm as compared to SVM

Detection of Phishing Websites Using Ensemble Machine Learning Approach	Random Forest algorithm and XGBoost algorithm	Comparative Study	Random forest aims to reduce the variance by training the model on different parts of the same training set using multiple deep decision trees. XGBoost is designed to be highly efficient, flexible, and portable. It uses parallelization, tree pruning, and weighted classifiers to produce superior results.	A large number of features have been used in training the classification model, leading to an increase in the search space dimension. The algorithms may be prone to overfitting or underfitting and thus might not produce optimum results. Machine Learning requires large data sets to train on, which must be inclusive, unbiased, and of high quality. They	>Accuracy >Precision >Recall >F-score

				may have to wait for new data to be created at times	
Phishing website detection using machine learning and deep learning techniques	->Logistic regression ->K nearest neighbour ->Decision Tree ->Random Forest ->XG Boost ->Ada Boost	Comparative Study	It made use of various flexible supervised machine learning techniques. We learnt that the decision trees model shows the best accuracy. The decision tree model could then be tested on real world phishing example, as a means of accurately detecting phishing websites.	Some common website features such as favicons, meta tags, pop-up windows, redirect links on the web page were not considered as part of the dataset that was taken, leaving chances of false positive predictions and overall lesser accuracy.	-> Training set accuracy -> Testing set accuracy -> Precision score

Phishing Website Detection System Using Machine Learning	Hybrid Algorithms approach, consisting of Naïve Bayes classifier, random forest and Support Vector Machine	Comparative study and a new proposed system	The hybrid algorithms approach led to amazing prediction rates and improved the accuracy of the system. The Random Forest turned out to be the best classifier as it gives the most exact precison, while the likes of Naïve bayes classifier and SVM proved very easy to utilize.	For the idea of the dataset utilized, any grouping calculations were referenced(not-ideal). As the square measures completely different applications, there is a tendency to not be able to differentiate that if the algorithms square measured are superior or not. Also, each of the classifiers has its own manner of classification	The accuracy of the various algorithms used were obtained and compared, to understand the relationship between classifiers and to find the most accurate algorithm

AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites	>ABET (Adaboost.M1+ Extra trees) >ROFET (Rotating Forest+Extra trees) >BET (Bagging+Extra trees) >LBET (LogiBoost+Extra trees)	Comparative Study	This study presented AI phishing detection methods that are interpretable, unlike other black-box AI methods. The proposed methods showed extremely high predictive accuracy of about 98% and low-false positive rate of about 0.018(by ABET) and 0.033(by LBET). There was usage of comprehensively featured and more recent phishing website data.	Decision tree algorithms apart from Extra trees could have been considered to provide a more interpretable model. Some hybridized models for detecting phishing website performed poorly when compared against single classifier Feature selection, extraction algo. could have been further explored with implemented meta-learner methods of this study models.	>Accuracy >False positive >ROC >False negative >F-measures
Detection of phishing websites using a novel twofold ensemble model	>Two-fold ensemble learner developed using integration of results from RF(random forest) classifier fed into a feedforward neural network(NN) >Decision Trees(DT) >Support vector machine(SVM) >Artificial Neural networks(ANN)	Comparative study and a new proposed system	The RF_NN model gave superior performance with an accuracy of 93.41% and a very less mean squared error of 0.000026. The RR_NF model reports if the website has any inherent threats based on its previous knowledgebase of websites. Such intelligent systems protect	Comparative analysis with other real-time data sets of recent origin must be performed to ensure generalization of the model against various security breaches. Different variants of phishing threats must be detected rather than focusing particularly toward phishing website detection. Machine Learning requires	> Accuracy >Precision >Recall >F-Measure >Mean square Error(MSE)

	>Random Forest >Bagging >Boosting		users from advanced spam and malicious attacks based on a prior evaluation. System can be used as a testing tool by LAN administrators on the server side to detect phishing websites from client architectures	large data sets to train on, which must be inclusive, unbiased, and of high quality. They may have to wait for new data to be created at times	
CatchPhish: detection of phishing websites by inspecting URLs	>TF-IDF algorithm (TF-term frequency & Inverse document frequency) >XGBoost >Random Forest(RF) >Logistic Regression(LR) >K-Nearest Neighbour >Support Vector machine(SVM) >Decision Tree	Comparative study and a new proposed system	The proposed model detects phishing sites based on the features extracted from the URL of a given website, so it is independent of third party services and source code. This behavior makes the technique adaptable at client side due to its low response time. The time taken for phishing website detection by the proposed CatchPhish system is only 0.68s on average. This application can	First limitation is that the model may misclassify some of the phishing sites hosted on free or compromised hosting servers since the features are extracted from the URL. There exist phishing URLs which do not follow patterns of the existing phishing sites. This also may lead to misclassifcation of such URLs as legitimate. Model may fail to detect the phishing URLs which use shortening URL and data URL services. It does not consider the visual mimic behavior existing	> Sensitivity or true positive rate(TPR) > Specificity or true negative rate(TNR) >False positive rate(FPR) >False negative rate(FNR) >Accuracy (Acc) >Precison (Pre) >F-measure (F)

			be used for the first level filtering of phishing websites without visiting the suspicious website with much higher accuracy.	in the source code or images.	

II. SUMMARIES OF LITERATURE SURVEY:

1. Information security has become a serious issue for all organizations and institutes as the demand for information and communication technology grows.[1] Phishing is one of the most prevalent methods for unauthorized individuals to obtain sensitive information. Data mining-based categorization intelligence techniques are crucial for classifying phishing and non-phishing attempts. They present a decision tree technique and an Info gain feature selection strategy (FST) for developing computationally efficient models for phishing website categorization based on different top selected feature subsets in this study. Our proposed Decision Tree (DT) technique improves classification accuracy by 99.80% employing 15 features in the case of Info gain FST.

2. Phishing is the most straightforward method of obtaining sensitive information from unsuspecting consumers. The goal of phishers is to obtain sensitive information such as usernames, passwords, and bank account numbers.[2] People working in cyber security are now looking for reliable and consistent detection strategies for phishing websites. The purpose of this work is to use machine learning to detect phishing URLs by extracting and evaluating various aspects of authentic and phishing URLs. Phishing websites are detected using Decision Tree, Random Forest, and Support Vector Machine algorithms. The goal of the paper is to detect phishing URLs and to narrow down the best machine learning method by analyzing each algorithm's accuracy rate, false positive rate, and false negative rate.

3. To date, several anti-phishing solutions have been presented, including blacklist or whitelist, heuristic and visual similarity-based methods, but internet users continue to be lured into providing important information on phishing websites. [3]To overcome the drawbacks of existing anti-phishing strategies, we offer a novel classification model based on heuristic features taken from URL, source code, and third-party services in this study. Our model was tested using eight different machine learning methods, with the Random Forest (RF) algorithm coming out on top with a 99.31 percent accuracy. To determine the best classifier for phishing website

detection, the trials were performed with different (orthogonal and oblique) random forest classifiers. With an accuracy of 99.55 percent, the principal component analysis Random Forest

(PCA-RF) outperformed all other oblique Random Forests (oRFs). We also evaluated our model with and without third-party-based features to see how effective third-party services are in classifying questionable websites. Our findings were also compared to the baseline models (CANTINA and CANTINA+). Our solution beat these methods and was also able to detect zero-day phishing attacks.

4. The Multi-label Classifier based on Associative Classification was developed by Abdelhamid et al. [4] for detecting phishing URLs (MCAC). They divided URLs into three categories based on sixteen features: phishing, legitimate, and suspect. The MCAC is a rule-based system that extracts several label rules from phishing data.

5. Millions of people utilize online services such as online banking, online shopping, social networking, e-commerce, and store and manage user sensitive information, and the World Wide Web has become an inseparable part of their lives. In reality, it is a widely used Internet tool for all types of users. To deliver such services, rich Web-based apps are accessible on the World Wide Web. At the same time, the Internet has evolved into a vital tool for individuals to communicate and conduct business.[5] This is the technology's good side. Regrettably, the Internet has become a more dangerous environment. Intruders and attackers have been lured to the popularity of the World Wide Web. These intruders take advantage of the Internet and its users by engaging in illegal behavior for financial gain. Harmful Web pages are those that contain such assaults or malicious code. While existing methods are effective in detecting fraudulent Web pages, there are still gaps in Web page feature selection and detection strategies. We present an in-depth analysis of existing malicious Web page detection algorithms and features in this study.

6. Associative categorization (AC) is a novel, powerful supervised learning method for predicting previously unknown events. AC successfully integrates association rule mining and classification, resulting in more accurate results than other data mining classification methods. The Fast Associative Classification Method is a novel AC algorithm proposed in this paper (FACA). On real-world phishing datasets, we compare our proposed method to four well-known AC algorithms (CBA, CMAR, MCAR, and ECAR) . [6]In our trials, we used classification accuracy and F1 evaluation metrics as the foundations for our research. In comparison to the other four well-known algorithms, the results show that FACA is quite successful in terms of the F1 evaluation metric (CBA, CMAR, MCAR, and ECAR). In terms of the accuracy evaluation metric, the FACA outperformed the other four AC algorithms.

7. Since there are so many online transactions made every day in the age of the internet, ensuring the security and privacy of online transactions and banking websites is a difficult challenge. Phishing attacks on websites are carried out by posing a phony website as a legitimate one in order to obtain private information and use it for non-genuine purposes.[7] The Bagging technique is utilized in this study with neural network and LMT classifiers as base classifiers in an ensemble to classify a group of URLs and identify whether they are phishing or authentic, allowing users to be protected from phishing assaults. We achieved a 90% accuracy rate in this project. 8. This paper explained the existing security problems in today’s digital world with respect to phishing and the process through which phishing is carried out. Phishing is a very

serious information security concern which may lead to loss of sensitive personal information due to clever disguising of phishing mails by attackers. This paper mainly focuses on trying to identify features useful for detecting phishing websites based on the URL of the website and applying machine learning algorithms to classify websites into legitimate and phishing ones respectively. The study discussed involves the comparison of results of 7 machine learning algorithms which were tested over a training set. The Random Forest algorithm emerged as the most accurate and hence, it may be considered the most suited algorithm for this binary classification. 9. In this paper, a method of phishing website detection was proposed, utilizing a meta-heuristic-based nonlinear regression algorithm together with a feature selection approach. This research utilized two feature selection methods: decision tree and wrapper to select the best feature subset. Post selection process, two meta-heuristic algorithms were successfully implemented to predict and detect the phishing websites: harmony search (HS) which was deployed based on nonlinear regression technique and support vector machine (SVM). The nonlinear regression approach was used to classify the websites, where the parameters of the proposed regression model were obtained using the HS algorithm. The nonlinear regression based on HS led to accuracy rates of 94.13 and 92.80% for train and test processes, respectively. As a result, the study finds that the nonlinear regression-based HS results in better performance compared to SVM. 10. Phishing is one of the most widely executed cybercrimes in the modern digital sphere where an attacker imitates an existing - and often trusted - person or entity in an attempt to capture a victim's login credentials, account information, and other sensitive data. Phishing websites are visually and semantically similar to real ones. The rise in online trading activities has resulted in a rise in the number of phishing scams. Machine Learning is one of the most feasible methods to approach this situation, as it is capable of handling the dynamic nature of phishing techniques, in addition to providing an accurate method of classification. In this paper, the use of Ensemble Machine Learning Methods such as Random Forest Algorithm and Extreme Gradient Boosting (XGBOOST) Algorithm for efficient and accurate phishing website detection based on its Uniform Resource Locator was proposed. It was seen that by using the Random Forest Classifier, a prediction accuracy of approximately 91% was obtained. While using XGBoost, a prediction accuracy of approximately 93% was obtained. Hence it was concluded that the algorithms produced highly accurate predictions with a satisfactory bias-variance trade-off in a timely and secure manner.

11. A phishing website is a website which is similar in name and appearance to an official website otherwise known as a spoofed website which is created to fool an individual and steal their personal credentials. So, to identify the websites which are fraud, the paper discussed the machine learning and deep learning algorithms and applied all these algorithms on a dataset and the algorithm having the best precision and accuracy could thus be selected for the phishing website detection. With the obtained result, it could be concluded that the Decision Tree model provided the best and highest accuracy. Whereas on the other hand ensemble algorithms have also been proved to be convenient as they are fast in speed and performance and are using more than one classifier for prediction and also give better performance. The methodology that was discovered can be considered a powerful technique to detect the phished websites and can provide more effective defenses for phishing attacks of the future.

12. It is discussed that phishing assaults are extremely harmful and it's significant for us to invite an instrument to distinguish it. This issue is handily understood by utilizing any of the AI calculations with the classifier. Thus through this paper we surveyed the capabilities used for the detection and detection techniques by the use of Machine learning. There are classifiers that give a decent expected pace of phishing. We've seen that the current framework gives less precision so a fresh out of the box strategy was proposed, that utilizes URL based highlights and furthermore, classifiers were created through a few AI calculations. The main findings of the preliminary work concluded that, Phishing URLs and domains show some characteristics that are different from other URLs and domains. So based on this the classifiers(Random forest, SVM, Naïve Bayes) were trained over the dataset of phishing websites. It was observed that the Random forest approach showed the most accuracy and could thus be considered the most optimal for the dataset.

13. The aim of this study is to provide an excellent solution to the menace of phishing in our modern society. By doing so, this study further aimed to solve the existing shortcomings already in place, the shortcomings like the inability of single classifier methods to detect phishing methods adequately, high false-positive rate, high false-negative rates, etc. As a result, this research work proposed, implemented and presented four (4) different AI-based meta-learner models using Extra-tree algorithm base learner for detecting phishing websites. The proposed methods (ABET, ROFET, BET and LBET) in this research work showcased the ability of AI meta-learners as an intelligent algorithm for developing models usable in detecting phishing websites. The methods produced extremely high predictive accuracy of approximately 98% by three of the proposed methods and also, a low false-positive rate of 0.018 by the ABET method and low false-negative rate of 0.033 from the LBET method. Thus, the results indicate the effectiveness and efficiency of the proposed methods whose false alarm rate is drastically low while achieving high accuracy and f-measure scores. The methods presented by this study resolved all problems mentioned above and set a new performance standard for phishing website detection methods.

14. Classifying phishing and non-phishing web content is a critical task in information security protocols, and full-proof mechanisms have yet to be implemented in practice. The purpose of this study was to present an ensemble machine learning model for classifying phishing websites. First there was focus on identifying significant attributes which lead to phishing attacks. The next step reflects the importance and evaluation of different variants of individual and ensemble machine learners toward detection of phishing websites. Lastly an intelligent detection system supported by RF_NN model for predicting the probability of phishing websites was proposed. To further fine-tune the predictive performance of individual classifiers, ensemble learners, such as the random forest, bagging and boosting, were adopted on the same data partitions. Of all the machine learning classifiers implemented in this study, RF_NN model gave the highest accuracy, whereas LR model resulted in lowest prediction rate. The best performing classifier (i.e., RF_NN) is selected to develop an intelligent detection system for tracking phishing websites for the data features identified from the study. This system helps in filtering malicious web URLs based on the behavioral patterns captured from the previous data samples. This system can be used as a testing tool by LAN administrators on the server side to detect phishing websites from

client architectures. 15. There exist many anti-phishing techniques which use source code-based features and third-party services to detect the phishing sites. These techniques, although somewhat successful, have some limitations and one of them is that they fail to handle drive-by-downloads. They also use third-party services for the detection of phishing URLs which delay the classification process. Hence, in this paper a light-weight application was proposed, called CatchPhish which predicts the URL legitimacy without visiting the website. The proposed technique uses hostname, full URL, Term Frequency-Inverse Document Frequency (TF-IDF) features and phish hinted words from the suspicious URL for the classification using the Random Forest classifier. Experiment with TF-IDF and hand-crafted features via the CatchPhish application achieved a significant accuracy of 94.26% on the dataset and an accuracy of 98.25%, 97.49% on benchmark datasets which is much better than the existing baseline models. The proposed model is implemented as an application using which the client can give a set of URLs as input and obtain the legitimacy of these URLs as output. The application is independent of third-party services and source code, so this behavior makes the technique adaptable at client side due to low response time.

III. SOME MORE EXISTING SYSTEMS:

1. PhishTank touts itself as a collaborative hub for data and information about four different forms of online spoofing. Additionally, PhishTank provides a free open API for developers and academics to incorporate anti-phishing data into their apps. As a result, PhishTank has created a directory of all vulnerable websites that have been identified and reported by people all across the internet, allowing developers to utilize their API to identify phishing sites.

2. In the same way as PhishTank employs a directory-based method and provides an open API, Google has a Safe Browsing API. Because new vulnerable web sites are continuously being created, and the directory cannot always be kept up to date, this technique is clearly unsuccessful. Because the URLs are sent to the PhishTank API, the users' browsing patterns are also revealed.

3. PhishDetector, a Chrome plugin, uses a rule-based approach to identify spoofing without requiring the use of an external internet service. While rule-based systems are easier to implement on the client side, they lack the precision of Machine Learning-based techniques. Similar work has been done by Shreeram. V on spoofing attack detection with a genetic algorithm identifies masquerading attacks using a genetic algorithm-generated rule.

4. One example is PhishNet, which uses predictive blacklisting algorithms. It used rules to match TLDs, directory structures, IP addresses, HTTP header replies, and other things. SpoofGuard by Stanford is a chrome extension that protects against spoofing by looking at DNS, URLs, images, and links using a similar rule-based manner..

5. Phishwish is a stateless spoofing filter with only the most fundamental criteria. The spoofing filter created by Debra L. Cook, Vijay K. Gurbani, and Michael Daniluk outperforms previous filters in the following ways: It does not need any training and does not rely on centralized white and black lists. They simply used 11 rules to determine the legitimacy of a website.

6. PhishBox: An Approach for Phishing Validation and Detection, by Jhen-Hao Li and Sheng-De Wang, discusses ensemble models for phishing detection. The average rate of false-positive phishing detection has dropped by 43.7 percent as a result.

7. The article Real time detection of phishing websites by Abdulghani Ali Ahmed and Nurul Amirah Abdullah presents an approach based on attributes from the website's URL alone. They were able to create a detection approach that can detect a wide range of phishing attacks with a low amount of false alarms.

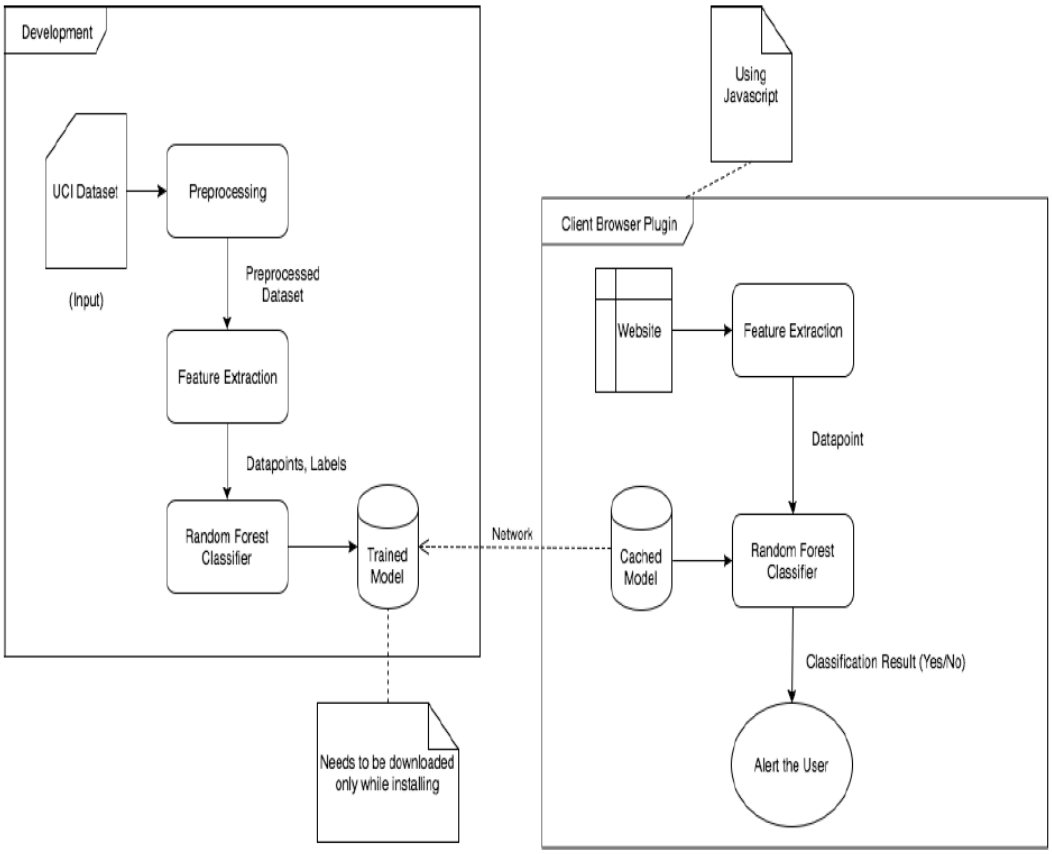
8. In Using Domain Top-page Similarity Feature in Machine LearningBased Web Phishing Detection, Nuttapong Sanglerdsinlapachai and Arnon Rungsawang provide a study on using a concept feature to identify web phishing. They incorporated a domain top-page similarity feature to a machine-learning-based phishing detection system.

8. PROPOSED TECHNIQUE:

Phishing is a method of stealing people's personal information by deceiving them with bogus emails and websites. Phishing hinders people from doing their business via the internet. Our problem statement is in order to tackle the threat and problem of phishing, developing a phishing website detection system that focuses on client-side implementation with quick detection, so that users are informed before being phished.

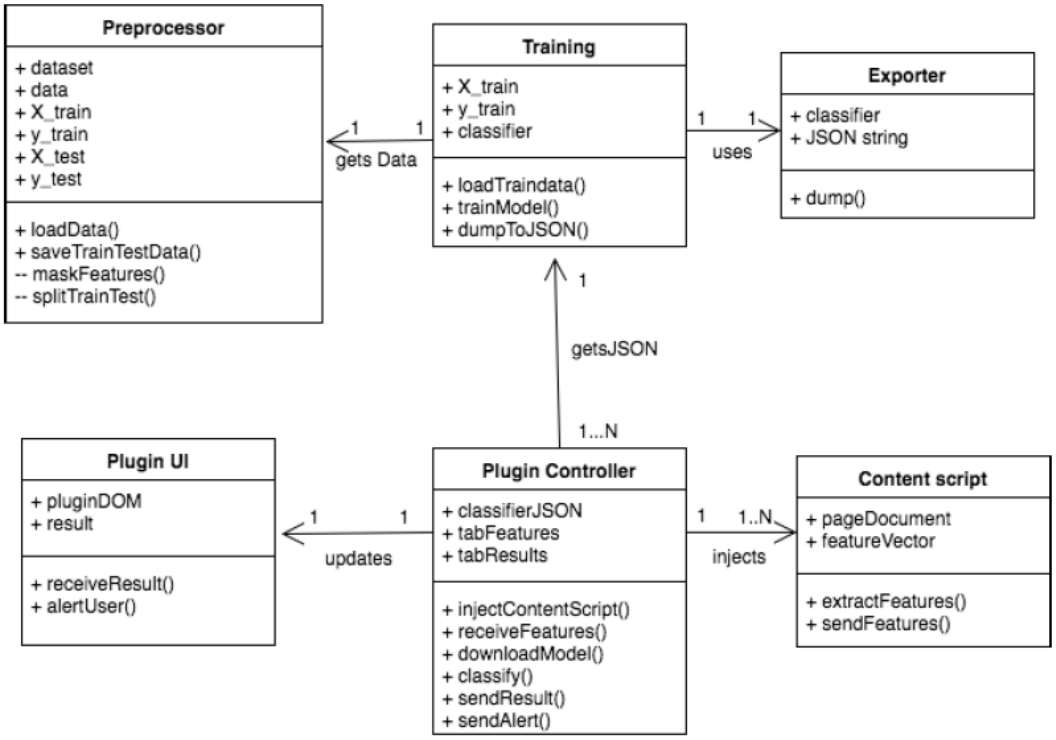
The objective of our project is to convert the base paper into javascript so that it may be used as a browser plugin. Because javascript does not support many machine learning libraries, and because client computers' processing capability is restricted, the approach must be kept lightweight. The random forest classifier must first be trained on the phishing websites dataset using python scikit-learn, and then the learned model parameters must be exported into a portable format for usage in javascript.

I. ARCHITECTURE DIAGRAM:

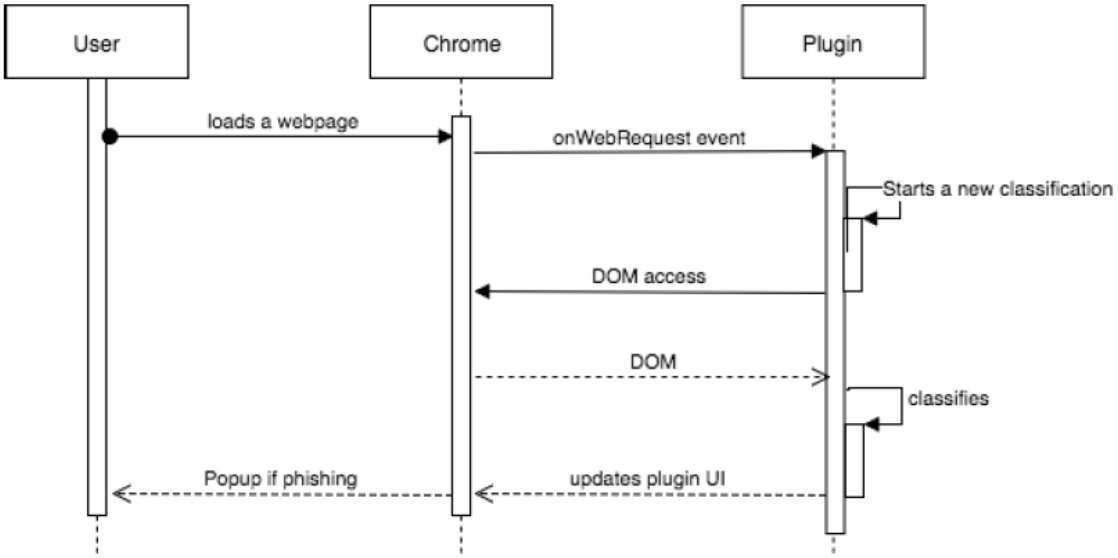


Our proposed system's entire operation is depicted in the architecture diagram. The phishing sites dataset is used to train a Random Forest classifier in Python scikit-learn. The learned classifier is sent to a JSON format that describes the random forest classifier. The exported model JSON was used to create a browser script that categorizes the current browser tab's webpage using the exported model JSON. If the user is being phished, the system will notify them. To assess whether a website is phishing or not, a Random Forest classifier is applied to 17 characteristics. The python arff module is used to load the dataset arff file, and 17 features are chosen from a list of 30. Features are chosen for their ability to be extracted completely offline on the client side without the need of a web service or a third party. After that, the dataset with the chosen characteristics is split into two halves for training and testing. After that, the Random Forest is trained on the training data before being exported to the JSON format that was previously defined. A URL is used to access the JSON file. On each page load, the client-side chrome plugin will execute a script and begin extracting and encoding the characteristics described above. If the exported model JSON isn't found in the cache after the features are encoded, the plugin downloads it again. Using the encoded feature vector and model JSON, the script can do classification. A warning is displayed to the user if the website is suspected of being phishing. The entire system is designed to be light and quick to detect.

II. FLOW DIAGRAM:



III. SEQUENCE DIAGRAM:



IV. PROPOSED METHODOLOGY:

A Random Forest classifier is trained on the fraudulent websites dataset using Python Scikit-learn Library / Module. The random forest classifier is described in JSON format, and the learned classifier is exported to it. A browser script has been written that categorizes the current browser tab's webpage using the exported model JSON. If the user is being spoofed, the system will notify them. To assess whether a website is fraudulent or not, a Random Forest classifier is applied to 17 characteristics. The python arff module is used to load the dataset arff file, and 17 features are chosen from a list of 30. Features are chosen for their ability to be extracted completely offline on the client side without the need of a web service or a third party. After that, the dataset with the chosen characteristics is split into two halves for training and testing. After that, the Random Forest is trained on the training data before being exported to the JSON format that was previously defined. A URL is used to access the JSON file. On each page load, the client-side Browser extension will execute a script and begin extracting and encoding the characteristics specified above. If the exported model JSON isn't found in the cache after the features are encoded, the plugin downloads it again. Using the encoded feature vector and model JSON, the script can do classification. A warning is displayed to the user if the website is suspected of being phishing. The entire system is designed to be light and quick to detect.

Let us explain the working of Random Forest Classifier — but first we will discuss Classification. Classification is the task of "classifying objects" into sub-categories, as the term implies. But that was done by a machine! If it doesn't seem impressive, consider your computer's ability to distinguish between you and a stranger. It's a cross between a potato and a tomato. Between an A and an F- on a scale of one to ten. Now let us jump into the concept of Random Forest - Random forest is a machine learning technique that mixes the results of several decision trees to produce a single outcome. Its popularity is due to its ease of use and adaptability, since it can handle both classification and regression issues. The three primary hyperparameters of random forest algorithms must be established before training. The size of the nodes, the number of trees, and the number of characteristics sampled are all factors to consider. The random forest classifier may then be used to tackle issues involving regression or classification.

Apart from the aforementioned , we shall also discuss some of the essential files and their content and the techniques enacted upon data taken —

Data Pre-processing : To load the dataset, we import python arff.library, and to divide the dataset, we import scikit learn train test split. Because the dataset is in arff (attribute relation file format), we must first read its properties using a Python function. After that, we divided the training and testing data in npy format (numpy) in a 70:30 ratio. Once that's done, we can get the npy files we'll need to train this dataset. As an assessment metric, cross validation is done on the training set.

Dump : In this section, we build the code to export the random forest model as a JSON file, so that it can be used from within the plugin by other background scripts for phishing computation while the plugin is running. The dump has two functions of code, one for converting the individual data of trees to Json format and the other for using the tree json data from the previous stage to create the random forest json file. When the trained random forest classifier model asks for it, the Random forest json file is the one that will be returned when this block of code has been fully executed.

Training : For the aim of training our dataset, we loaded numerous sklearn libraries, such as sklearn.ensemble, sklearn.tree, and sklearn.model selection. The npy files that had been produced during the preparation stage were then imported. Once the training data had been supplied to the classifier, we continued by generating the random forest classifier as clf and calculating the cross validation score for it. We used the model to predict whether the testing data had been set aside after it had been successfully trained. We discovered the accuracy of our trained model in this way. Finally, we used the Json.dump method to export our learned classifier data into JSON format, so that it could be accessible by other background Javascripts in our plug-in.

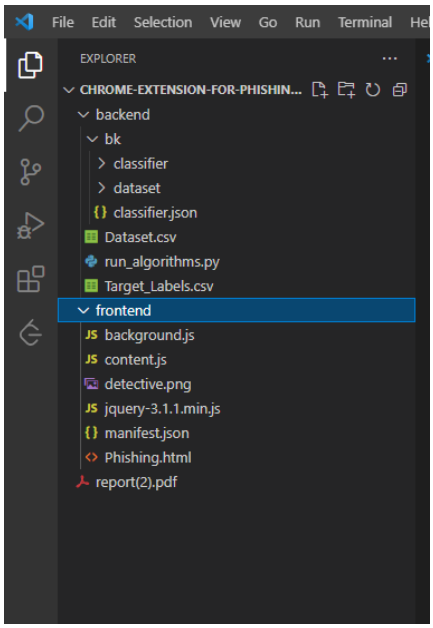
RandomForest.js : This was a critical component of the logic that underpins our background javascripts. We now proceed to define rules for our plugin, which will take a two-faceted approach, using the classifier.json data that was produced during the previous steps of training and preprocessing. The code uses the classifier JSON data and the feature information that is collected from the website when a user accesses a web page to make a prediction based on a single tree data from the JSON file, and then repeats the process for the remaining trees. Finally, the above-mentioned prediction data values would be fed into the random forest prediction function, which would evaluate the values and determine which values favored the website and which fell short, allowing it to produce a final prediction output for the website, which could then be accessed by the UI portion of the code and displayed to the user.

Feature Retrieval Plugin : When the plugin has been developed using the chrome developer tools and installed to the user's browser, the plugin acquires the URL and DOM features of the website whenever the user opens a webpage. It then extracts the url features before moving on to the direct DOM features. It then extracts the indirect features, bringing the feature retrieval procedure to a close.

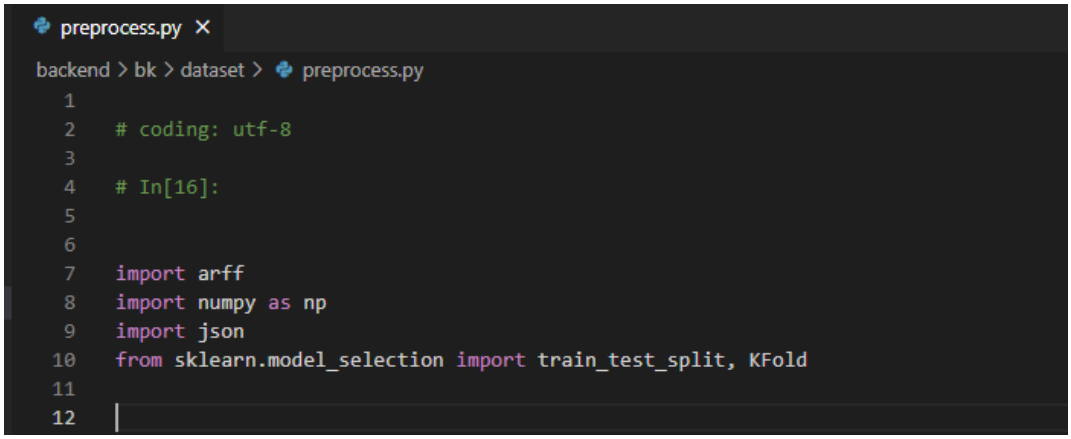
Prediction Plugin : Our plugin now loads the model parameters, followed by the model features, when the feature retrieval procedure has been completed. After that, sklearn prediction is reimplemented. If the plugin determines that the website is phishy based on the estimated projected value, the user will receive an alert before the page can fully load. However, if the plugin determines that the page is secure, no message is displayed, and the user may continue to browse without fear of being phished.

V. IMPLEMENTATION CODE:

Modules:



Data Preprocessing:



preprocess.py

backend > bk > dataset > preprocess.py

```
12
13  # In[17]:
14
15
16  dataset = arff.load(open('dataset.arff', 'r'))
17  data = np.array(dataset['data'])
18
19
20  # In[18]:
21
22
23  print('The dataset has {0} datapoints with {1} features'.format(data.shape[0], data.shape[1]-1))
24  print('Features: {0}'.format([feature[0] for feature in dataset['attributes']]))
25
```

preprocess.py

backend > bk > dataset > preprocess.py > ...

```
26
    Run Cell | Run Above | Debug Cell
27  # In[19]:
28
29
30  data = data[:, [0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 22, 30]]
31
32
    Run Cell | Run Above | Debug Cell
33  # In[20]:
34
35
36  X, y = data[:, :-1], data[:, -1]
37  y.reshape(y.shape[0])
38  print('Before spliting')
39  print('X:{0}, y:{1}'.format(X.shape, y.shape))
40  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
41  print('After spliting')
42  print('X_train:{0}, y_train:{1}, X_test:{2}, y_test:{3}'.format(X_train.shape, y_train.shape, X_test.shape, y_test.shape))
43
```

preprocess.py

backend > bk > dataset > preprocess.py > ...

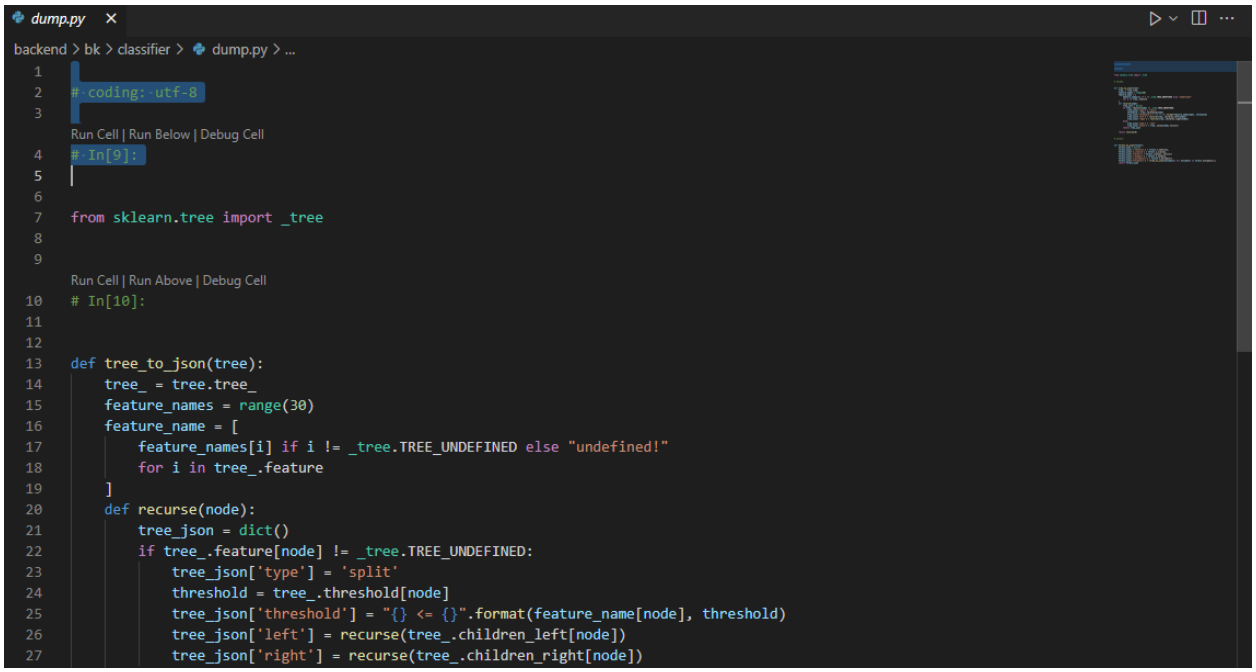
```
Run Cell | Run Above | Debug Cell
# In[21]:


np.save('X_train.npy', X_train)
np.save('X_test.npy', X_test)
np.save('y_train.npy', y_train)
np.save('y_test.npy', y_test)
print('Saved!')


Run Cell | Run Above | Debug Cell
# In[24]:


test_data = dict()
test_data['X_test'] = X_test.tolist()
test_data['y_test'] = y_test.tolist()
with open('../../static/testdata.json', 'w') as tdfile:
    json.dump(test_data, tdfile)
    print('Test Data written to testdata.json')

```

Data Preprocessing Results:

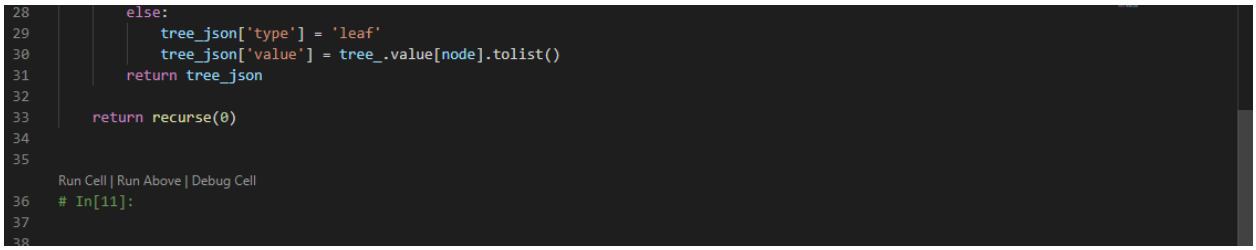
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell

```
PS C:\Users\sujal\Desktop\Chrome-Extension-for-Phishing-website-detection-master\backend\bk\dataset> python preprocess.py
The dataset has 11055 datapoints with 30 features
Features: ['having_IP_Address', 'URL_Length', 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix', 'having_Sub_Domain', 'SSL
final_State', 'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email',
'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index', '
Links_pointing_to_page', 'Statistical_report', 'Result']
Before spliting
X:(11055, 17), y:(11055,)
After spliting
X_train:(7738, 17), y_train:(7738,), X_test:(3317, 17), y_test:(3317,)
Saved!
Test Data written to testdata.json
PS C:\Users\sujal\Desktop\Chrome-Extension-for-Phishing-website-detection-master\backend\bk\dataset>
```

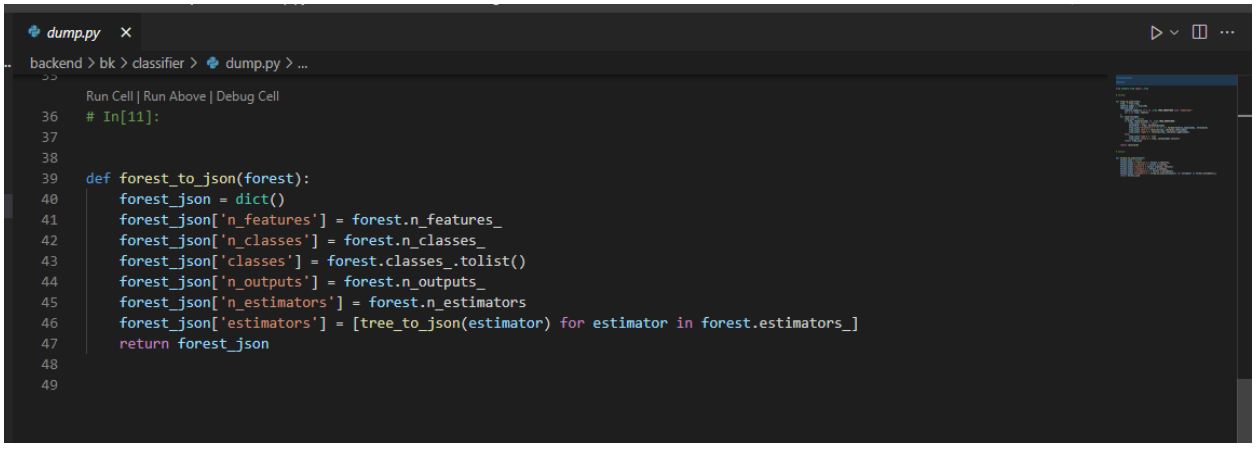
Dump.py



```
backend > bk > classifier > dump.py > ...
1
2   # -*- coding: utf-8 -*-
3
    Run Cell | Run Below | Debug Cell
4   # In[9]:
5
6
7   from sklearn.tree import _tree
8
9
    Run Cell | Run Above | Debug Cell
10  # In[10]:
11
12
13  def tree_to_json(tree):
14      tree_ = tree.tree_
15      feature_names = range(30)
16      feature_name = [
17          feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
18          for i in tree_.feature
19      ]
20      def recurse(node):
21          tree_json = dict()
22          if tree_.feature[node] != _tree.TREE_UNDEFINED:
23              tree_json['type'] = 'split'
24              threshold = tree_.threshold[node]
25              tree_json['threshold'] = "{} <= {}".format(feature_name[node], threshold)
26              tree_json['left'] = recurse(tree_.children_left[node])
27              tree_json['right'] = recurse(tree_.children_right[node])
```



```
28          else:
29              tree_json['type'] = 'leaf'
30              tree_json['value'] = tree_.value[node].tolist()
31          return tree_json
32
33      return recurse(0)
34
35
    Run Cell | Run Above | Debug Cell
36  # In[11]:
37
```



```
backend > bk > classifier > dump.py > ...
    Run Cell | Run Above | Debug Cell
36  # In[11]:
37
38
39  def forest_to_json(forest):
40      forest_json = dict()
41      forest_json['n_features'] = forest.n_features_
42      forest_json['n_classes'] = forest.n_classes_
43      forest_json['classes'] = forest.classes_.tolist()
44      forest_json['n_outputs'] = forest.n_outputs_
45      forest_json['n_estimators'] = forest.n_estimators
46      forest_json['estimators'] = [tree_to_json(estimator) for estimator in forest.estimators_]
47      return forest_json
48
49
```

Training:

training.py

backend > bk > classifier > training.py > ...

```
1
2   # coding: utf-8
3
    Run Cell | Run Below | Debug Cell
4   # In[1]:
5
6
7   from sklearn.ensemble import RandomForestClassifier
8   from sklearn.tree import DecisionTreeClassifier
9   from sklearn.model_selection import cross_val_score
10  from sklearn.metrics import accuracy_score
11  import numpy as np
12  import json
13  import dump
14
15
    Run Cell | Run Above | Debug Cell
16  # In[2]:
17
18
```

training.py

backend > bk > classifier > training.py > ...

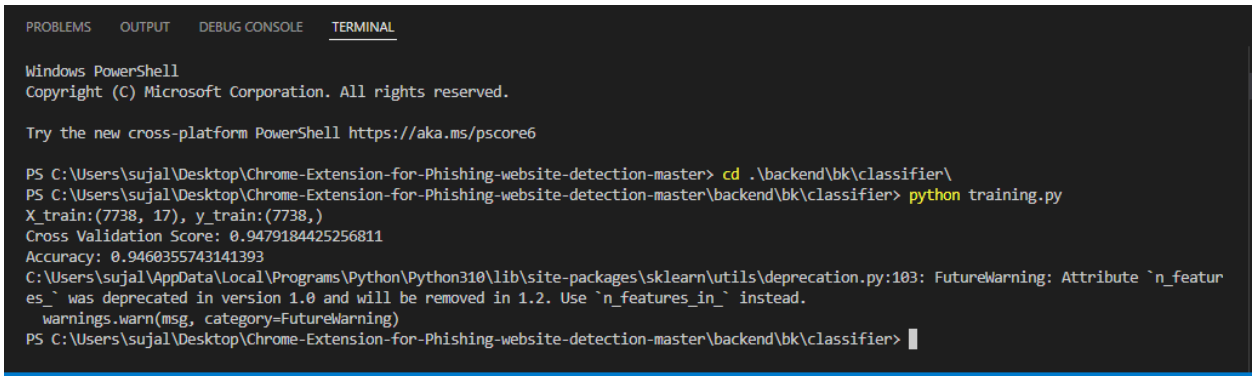
```
18
19  X_train = np.load('../dataset/X_train.npy')
20  y_train = np.load('../dataset/y_train.npy')
21  print('X_train:{0}, y_train:{1}'.format(X_train.shape, y_train.shape))
22
23
    Run Cell | Run Above | Debug Cell
24  # In[3]:
25
26
27  clf = RandomForestClassifier()
28  print('Cross Validation Score: {0}'.format(np.mean(cross_val_score(clf, X_train, y_train, cv=10))))
29
30
    Run Cell | Run Above | Debug Cell
31  # In[4]:
32
33
34  clf.fit(X_train, y_train)
35
36
```

training.py

backend > bk > classifier > training.py > ...

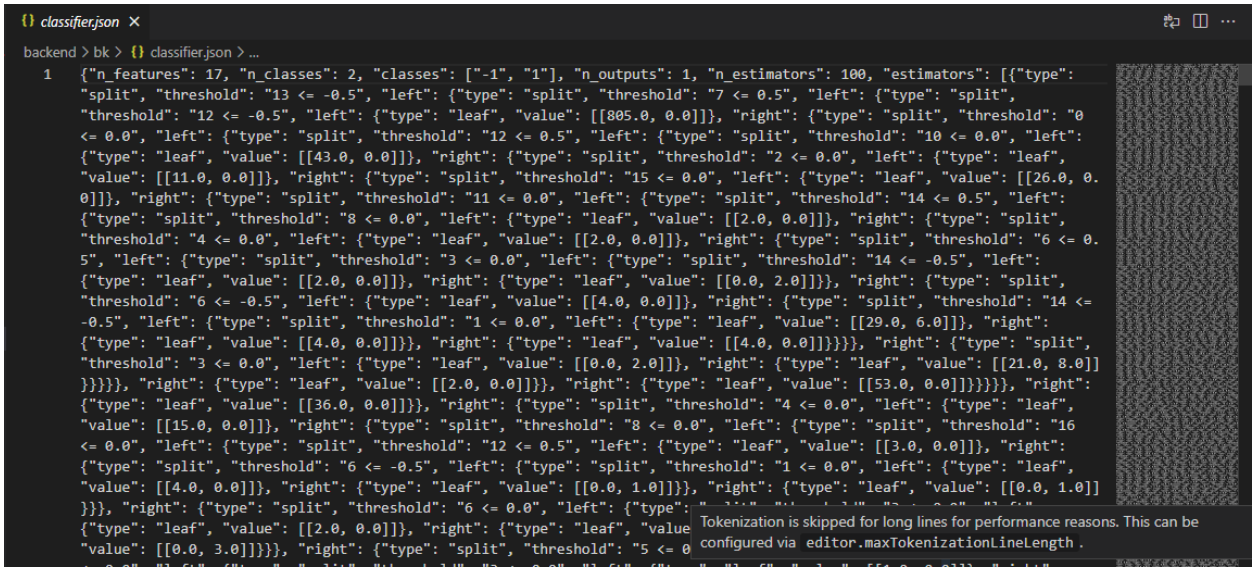
```
Run Cell | Run Above | Debug Cell
# In[4]:


clf.fit(X_train, y_train)


Run Cell | Run Above | Debug Cell
# In[5]:


X_test = np.load('../dataset/X_test.npy')
y_test = np.load('../dataset/y_test.npy')


Run Cell | Run Above | Debug Cell
# In[6]:


pred = clf.predict(X_test)
print('Accuracy: {}'.format(accuracy_score(y_test, pred)))


Run Cell | Run Above | Debug Cell
# In[7]:


#print(forest_to_json(clf))
json.dump(dump.forest_to_json(clf), open('../classifier.json', 'w'))


```

Training Results:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\sujal\Desktop\Chrome-Extension-for-Phishing-website-detection-master> cd .\backend\bk\classifier\
PS C:\Users\sujal\Desktop\Chrome-Extension-for-Phishing-website-detection-master\backend\bk\classifier> python training.py
X_train:(7738, 17), y_train:(7738,)
Cross Validation Score: 0.9479184425256811
Accuracy: 0.9460355743141393
C:\Users\sujal\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\deprecation.py:103: FutureWarning: Attribute `n_featur
es_` was deprecated in version 1.0 and will be removed in 1.2. Use `n_features_in_` instead.
  warnings.warn(msg, category=FutureWarning)
PS C:\Users\sujal\Desktop\Chrome-Extension-for-Phishing-website-detection-master\backend\bk\classifier>
```

Exporting Model:



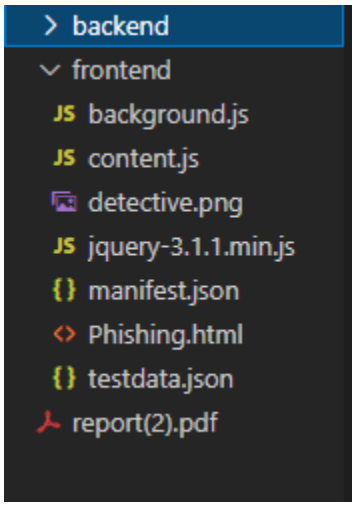
classifier.json

backend > bk > classifier.json > ...


```
{"n_features": 17, "n_classes": 2, "classes": ["-1", "1"], "n_outputs": 1, "n_estimators": 100, "estimators": [{"type":
"split", "threshold": "13 <= -0.5", "left": {"type": "split", "threshold": "7 <= 0.5", "left": {"type": "split",
"threshold": "12 <= -0.5", "left": {"type": "leaf", "value": [[805.0, 0.0]]}, "right": {"type": "split", "threshold": "0
<= 0.0", "left": {"type": "split", "threshold": "12 <= 0.5", "left": {"type": "split", "threshold": "10 <= 0.0", "left":
{"type": "leaf", "value": [[43.0, 0.0]]}, "right": {"type": "split", "threshold": "2 <= 0.0", "left": {"type": "leaf",
"value": [[11.0, 0.0]]}, "right": {"type": "split", "threshold": "15 <= 0.0", "left": {"type": "leaf", "value": [[26.0, 0.
0]]}, "right": {"type": "split", "threshold": "11 <= 0.0", "left": {"type": "split", "threshold": "14 <= 0.5", "left":
{"type": "split", "threshold": "8 <= 0.0", "left": {"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "split",
"threshold": "4 <= 0.0", "left": {"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "split", "threshold": "6 <= 0.
5", "left": {"type": "split", "threshold": "3 <= 0.0", "left": {"type": "split", "threshold": "14 <= -0.5", "left":
{"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "leaf", "value": [[0.0, 2.0]]}}, "right": {"type": "split",
"threshold": "6 <= -0.5", "left": {"type": "leaf", "value": [[4.0, 0.0]]}, "right": {"type": "split", "threshold": "14 <=
-0.5", "left": {"type": "split", "threshold": "1 <= 0.0", "left": {"type": "leaf", "value": [[29.0, 6.0]]}, "right":
{"type": "leaf", "value": [[4.0, 0.0]]}}, "right": {"type": "leaf", "value": [[4.0, 0.0]]}}}}, "right": {"type": "split",
"threshold": "3 <= 0.0", "left": {"type": "leaf", "value": [[0.0, 2.0]]}, "right": {"type": "leaf", "value": [[21.0, 8.0]]
}}}}}, "right": {"type": "leaf", "value": [[2.0, 0.0]]}}, "right": {"type": "leaf", "value": [[53.0, 0.0]]}}}}}}, "right":
{"type": "leaf", "value": [[36.0, 0.0]]}}, "right": {"type": "split", "threshold": "4 <= 0.0", "left": {"type": "leaf",
"value": [[15.0, 0.0]]}, "right": {"type": "split", "threshold": "8 <= 0.0", "left": {"type": "split", "threshold": "16
<= 0.0", "left": {"type": "split", "threshold": "12 <= 0.5", "left": {"type": "leaf", "value": [[3.0, 0.0]]}, "right":
{"type": "split", "threshold": "6 <= -0.5", "left": {"type": "split", "threshold": "1 <= 0.0", "left": {"type": "leaf",
"value": [[4.0, 0.0]]}, "right": {"type": "leaf", "value": [[0.0, 1.0]]}}, "right": {"type": "leaf", "value": [[0.0, 1.0]]
}}}, "right": {"type": "split", "threshold": "6 <= 0.0", "left": {"type":
{"type": "leaf", "value": [[2.0, 0.0]]}, "right": {"type": "leaf", "value
"value": [[0.0, 3.0]]}}}}, "right": {"type": "split", "threshold": "5 <= 0
```


Tokenization is skipped for long lines for performance reasons. This can be configured via editor.maxTokenizationLineLength .

Frontend Javascript Files:

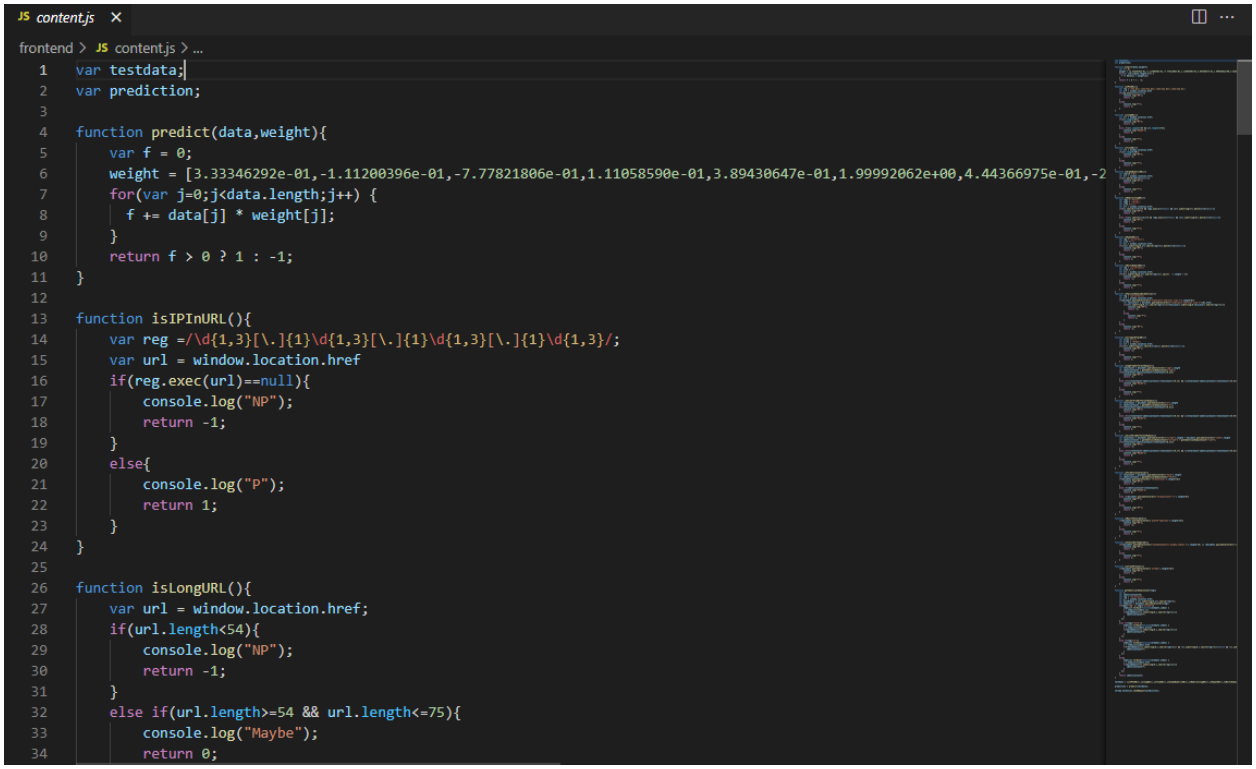


Manifest.json

Background.js

```
JS background.js ×
frontend > JS background.js > ...
1   chrome.extension.onRequest.addListener(function(prediction){
2       if (prediction == 1){
3           alert("Warning: Phishing detected!!");
4       }
5       else if (prediction == -1){
6           alert("No phishing detected");
7       }
8   });
9
```

Content.js



```
JS content.js ×
frontend > JS content.js > ...
1   var testdata;
2   var prediction;
3
4   function predict(data,weight){
5       var f = 0;
6       weight = [3.33346292e-01,-1.11200396e-01,-7.77821806e-01,1.11058590e-01,3.89430647e-01,1.99992062e+00,4.44366975e-01,-2
7       for(var j=0;j<data.length;j++) {
8         f += data[j] * weight[j];
9       }
10      return f > 0 ? 1 : -1;
11  }
12
13  function isIPInURL(){
14      var reg =/\d{1,3}[\.]{1}\d{1,3}[\.]{1}\d{1,3}[\.]{1}\d{1,3}/;
15      var url = window.location.href
16      if(reg.exec(url)==null){
17          console.log("NP");
18          return -1;
19      }
20      else{
21          console.log("P");
22          return 1;
23      }
24  }
25
26  function isLongURL(){
27      var url = window.location.href;
28      if(url.length<54){
29          console.log("NP");
30          return -1;
31      }
32      else if(url.length>=54 && url.length<=75){
33          console.log("Maybe");
34          return 0;
```

content.js

frontend > content.js > ...

```
    }
    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;
```

frontend > content.js > ...

```
    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;
    }
```

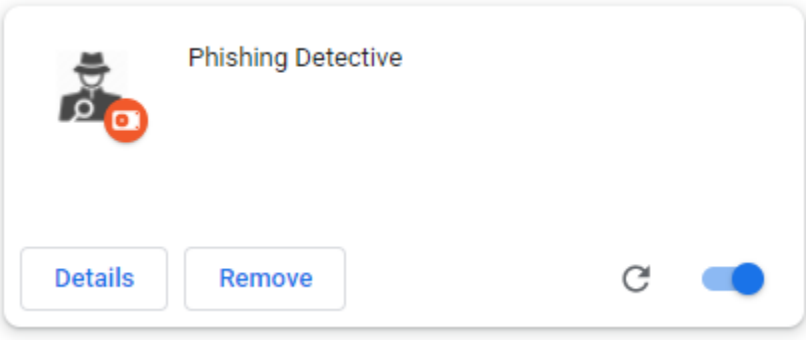
content.js

frontend > content.js > ...

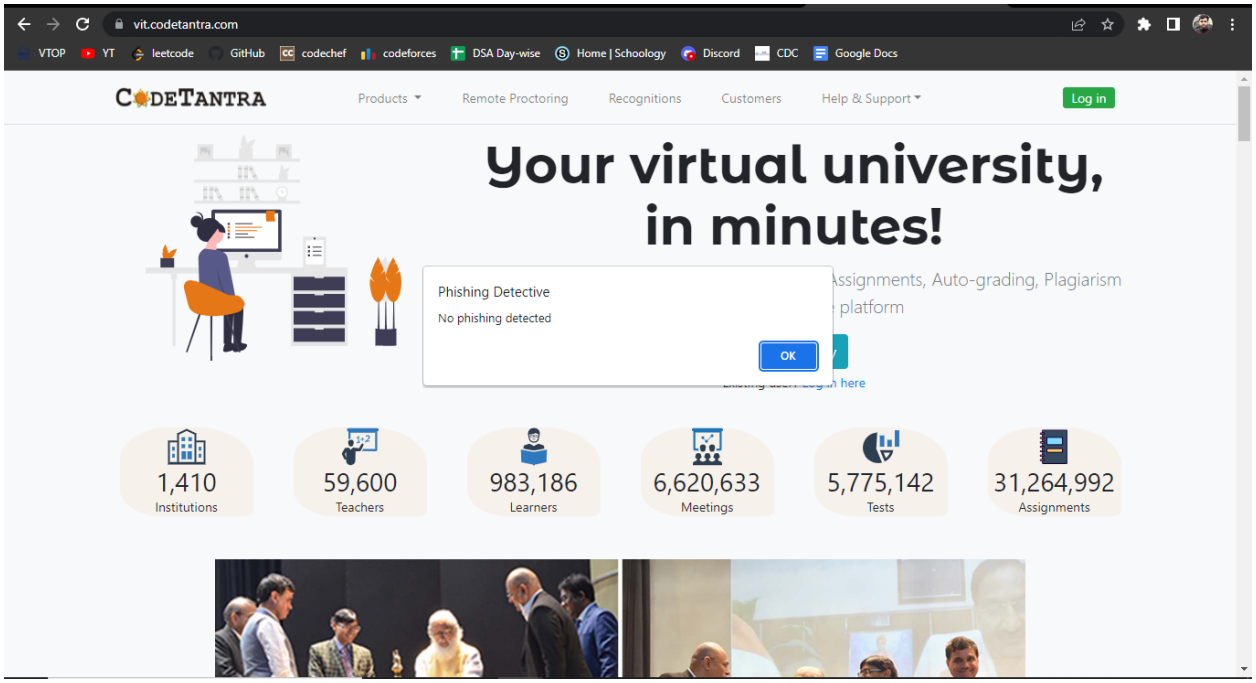
```
258     }
259     else if(tag=="form"){
260         nodeList.forEach(function(element,index) {
261         i = nodeList[index].action
262         if(mainDomain==(i.substring(0,i.search(reg)+1))){
263            identicalCount++;
264         }
265       });
266     }
267     else if(tag=="a"){
268         nodeList.forEach(function(element,index) {
269         i = nodeList[index].href
270         if((mainDomain==(i.substring(0,i.search(reg)+1))) && ((i.substring(0,i.search(reg)+1))!=null) && ((i.substring(0,i.
271            identicalCount++;
272         }
273       });
274     }
275     else{
276         nodeList.forEach(function(element,index) {
277         i = nodeList[index].href
278         if(mainDomain==(i.substring(0,i.search(reg)+1))){
279            identicalCount++;
280         }
281       });
282     }
283     return identicalCount;
284 }
285
286 testdata = [isIPInURL(),isLongURL(),isTinyURL(),isAlphaNumericURL(),isRedirectingURL(),isHypenURL(),isMultiDomainURL(),isFa
287
288 prediction = predict(testdata);
289
290 chrome.extension.sendRequest(prediction);
291
```

VI. IMPLEMENTATION OUTPUT:

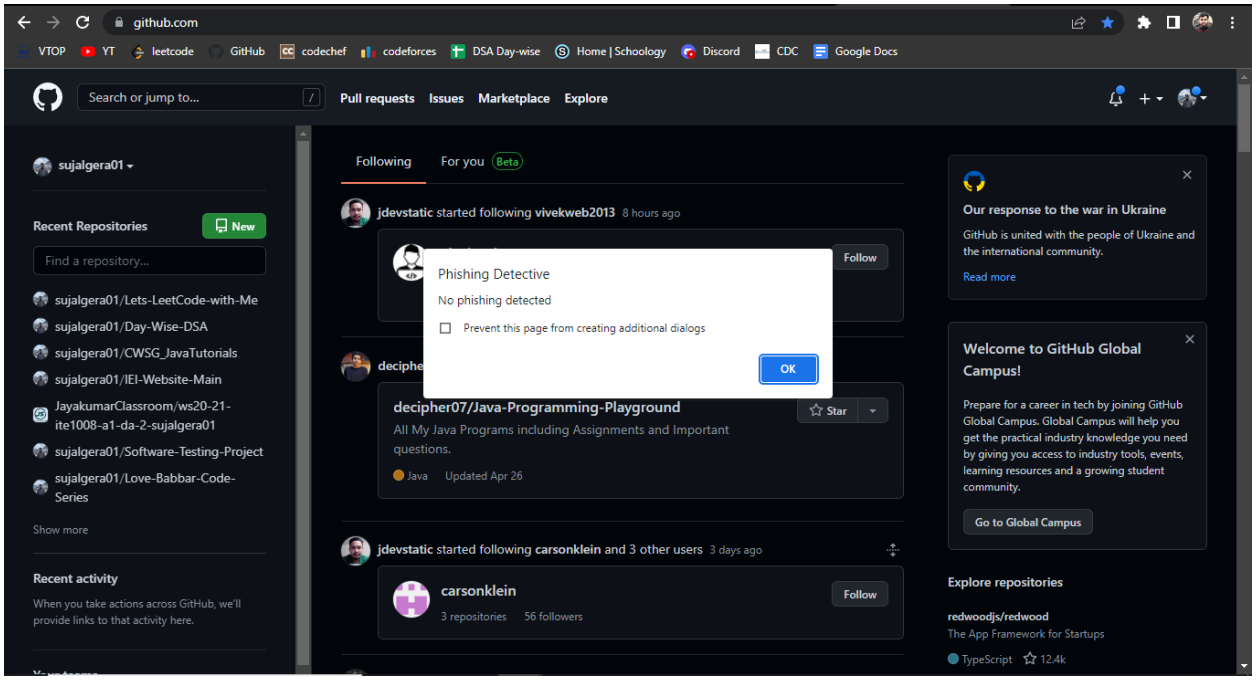
Extension Loaded in Chrome Browser:



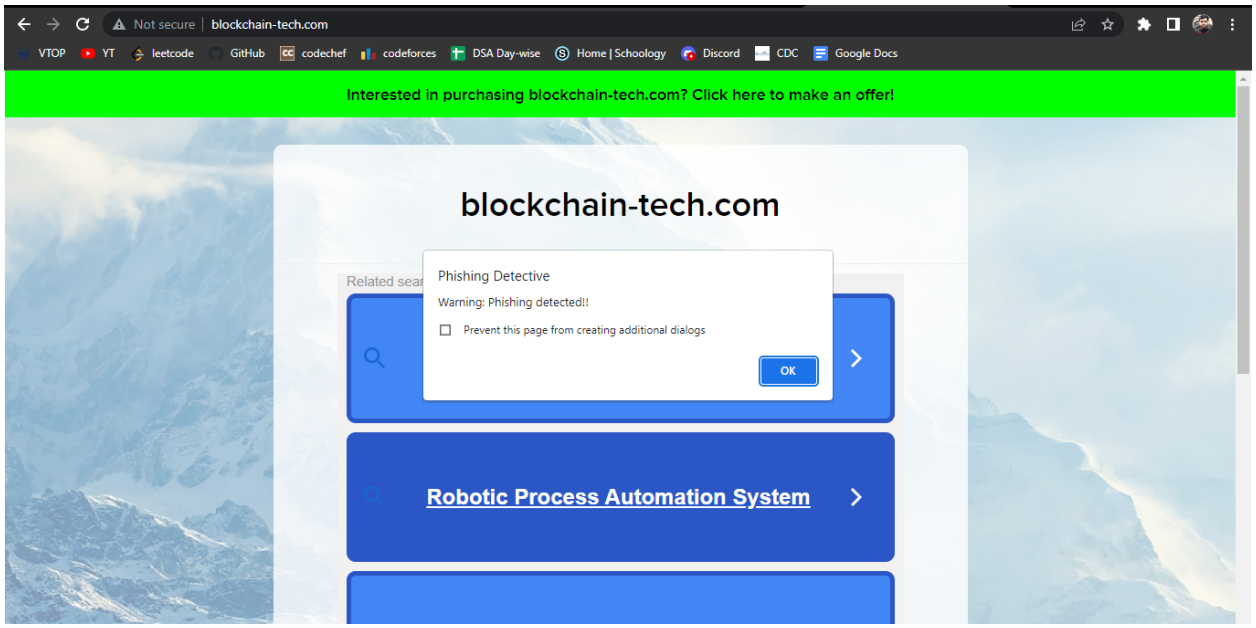
Tested on Codetantra platform:



Tested on Github:



Testing on any phishing website:



9. RESULTS AND DISCUSSIONS:

Thus , our system delivers an accuracy of **94.66%,** which is considerably higher. The websites tested were - CodeTantra.com , gitHub.com , and an intentionally made phishing website - blockchain-tech.com - Further we will see the comparative study to check how much our system is efficient than some of the significant existing systems (discussed earlier)

I. COMPARATIVE STUDY WITH EXISTING SYSTEMS:

When compared to ours, the current systems yielded the following set of accuracies:

Name of System		Accuracy
PhishChecker		0.960
Anti-Phish(Our system)		0.9466
PhishBox (PhishLogin Dataset)	BayesNet	0.8768
	Logistic Regression	0.9133
	Decision Tree	0.9211
	Random Forest	0.9491
PhishBox (PhishAlexa Dataset)	BayesNet	0.9336
	Logistic Regression	0.9546
	Decision Tree	0.9770
	Random Forest	0.9844
Netcraft		0.980

II. COMPARISON WITH OTHER ALGORITHMS:

Now let us compare the performance of our system's algorithm with other algorithms (present in existing system(s)) :

	ANN	K-NN	SVM	C4.5	RandomForest	RotationForest
ROC Area	0.995	0.989	0.97	0.984	0.996	0.994
F-measure	0.969	0.972	0.972	0.959	0.974	0.968
Accuracy	96.91%	97.18%	97.17%	95.88%	97.36%	96.79%

Out of the six methods evaluated, the Random Forest approach not only had the best accuracy of **97.36 percent**, but it also had the highest F-measure and ROC area values of **0.974** and **0.996**, respectively.

10. CONCLUSION AND FUTURE ENHANCEMENT:

Spoofing is a method of stealing personal information from people by deceiving them with bogus emails and websites. Masquerading hinders people from doing their business via the internet. So, in order to tackle the threat and problem of Spoofing , we were able to effectively design a Spoofing website detection system that focuses on client-side implementation and speedy detection, allowing users to be informed before being phished. We discovered via study and numerous literature surveys that comparable efforts frequently employ webpage characteristics that are difficult to extract on the client side, resulting in detection being network-dependent. To address the aforementioned issue, the system was designed to only employ features that can be extracted on the client side, allowing it to provide faster detection and greater privacy than similar existing systems. We chose the machine learning approach for this project after testing other algorithms on the same dataset based on data from Phishing websites. This allowed us to observe that the suggested RF classifier performs well in terms of classification accuracy, F-measure, and AUC. RF is also quicker, more robust, and more accurate than the other classifiers, according to our findings. As a result, the Random forest classifier was chosen as the best fit for our system. Also, because the JSON representation of the model and the classification script were created with temporal complexity in mind, the conversion from Python to Javascript and implementation of Random Forest in Javascript aided in speedy detection. Despite the fact that the developed plugin has many merits, there are still some criticisms about the plugin's inner workings that may be addressed in the future with more time and processing power. Overall, the development of the plugin for machine learning-based phishing website detection was a success.

We only trained the random forest classifier on 17 characteristics out of the 30 available in the dataset at the moment. However, this number can be increased as long as it does not slow down detection or endanger the privacy of the plugin's user; so, this is a future potential to consider. If feasible, the extension can store the results of sites that the user visits frequently, reducing the amount of processing necessary. However, this increases the chance of an undiscovered pharming attack, necessitating the development of a system for storing findings without compromising the ability to detect pharming. Worker Threads may also be used to categorize data in javascript, potentially speeding up the classification process. In conclusion, there are

various conceivable adjustments and additions to this system that may make it a more practical phishing detection tool, but the viability of such techniques is still debatable.

11. REFERENCES:

[1] Shrivas, A. K., & Suryawanshi, R. Decision Tree Classifier for Classification of Phishing Website with Info Gain Feature. Int. J. for Res.Appl. Sci. Eng. Technol., 5(5), 780–783

[2] Rishikesh Mahajan MTECH Information Technology K.J. Somaiya College of Engineering, Mumbai - 77 Irfan Siddavatam Professor, Dept. Information Technology K.J. Somaiya College of Engineering, Mumbai - 77International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 23, October 2018 45 Phishing Website Detection using Machine Learning Algorithms

[3] R.S., Pais, A.R.: Detection of phishing websites using an efficient feature-based machinelearning framework. Neural Comput. Appl., 1–23.

[4] Abdelhamid, N., Ayesh, A., & Thabtah, F. Phishing detection based Associative Classification data mining. Expert Systems with Applications, 41(13),5948–5959. https://doi.org/10.1016/j.eswa.2014.03.019.

[5] D. R. Patil and J. Patil, J., "Survey on malicious web pages detection techniques", International Journal of u-and e-Service, Science and Technology, vol. 8, no. 5, pp. 195–206.

[6] W. Hadi, F. Aburrub, and S, Alhawari, A new fast associative classification algorithm for detecting phishing websites, Applied Soft Computing 48 729-734.

[7] Priyanka Sharma, Rajni Ranjan Singh Makwana. Phishing Website Detection Using Ensemble Technique. International Journal of Computer Trends and Technology, 69(3), 26-29.

[8] Dipayan Sinha, Dipayan Sinha, Prof. Anitha Sandeep (2020) "Phishing Website URL Detection using Machine Learning", International Journal of Advanced Science and Technology, 29(7), pp. 2495-2504.

[9] Babagoli, M., Aghababa, M.P. & Solouk, V. Heuristic nonlinear regression strategy for detecting phishing websites. Soft Comput 23, 4315–4327. https://doi.org/10.1007/s00500-018-3084-2

[10] M. Dharani, Badkul Soumya, Gharat Kimaya, Vidhate Amarsinh & Bhosale Dhanashri 2021, 'Detection of Phishing Websites Using Ensemble Machine Learning Approach', ITM Web of Conferences, vol. 40, p. 03012, viewed 3 September 2021

[11] M Selvakumari et al 2021 J. Phys.: Conf. Ser. 1916 012169

[12] Manish Jain, Kanishk Rattan, Divya Sharma, Kriti Goel, Nidhi Gupta, Phishing Website Detection System Using Machine Learning, International Research Journal of Engineering and Technology (IRJET), Volume: 07, Issue: 05 May 2020

[13] Alsariera, Y. A. et al. (2020) 'AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites', IEEE Access, Access, IEEE, 8, pp. 142532–142542. doi: 10.1109/ACCESS.2020.3013699

[14] Kalyan Nagaraj et al. 'Detection of phishing websites using a novel twofold ensemble model', Journal of Systems and Information Technology, 20(3), pp. 321–357. doi: 10.1108/JSIT-09-2017-0074.

[15] Rao, R. S. (1), Pais, A. R. (1) and Vaishnavi, T. (2) (no date) 'CatchPhish: detection of phishing websites by inspecting URLs', Journal of Ambient Intelligence and Humanized Computing, 11(2), pp. 813–825. doi: 10.1007/s12652-019-01311-4.