PishShield: Machine Learning Plugin for

phishing detection

Kartik Patel  
Dept of Information Technology *Shah and Anchor Engineering College*Mumbai, India  
Kartik.patel16254@sakec.ac.in

Ashish Jha  
Dept of Information Technology *Shah and Anchor Engineering College*  
Mumbai, IndiaAshish.jha15641@sakec.ac.in

Pratham Bhadra  
Dept of Information Technology *Shah and Anchor Engineering College*  
Mumbai, Indial pratham.bhadra15579@sakec.ac.in

Prof. Chintal Gala  
Dept of Information Technology *Shah and Anchor Engineering College*  
*Mumbai, India*Chintal.gala@sakec.ac.in

Abstract— Phishing poses a significant threat to cybersecurity, with malicious actors constantly evolving tactics to deceive unsuspecting individuals into disclosing sensitive information. Detecting phishing websites in real-time is paramount to mitigating these risks. In this research, we propose a novel approach to address this challenge by developing a Chrome browser plugin that performs classification without relying on external servers, thereby preserving user privacy and minimizing latency. Leveraging machine learning techniques, particularly the Random Forest classifier, our plugin detects phishing websites promptly, providing users with timely alerts to prevent potential security breaches. Through comprehensive experimentation and analysis, we demonstrate the effectiveness of our approach in achieving a balance between computational efficiency and predictive accuracy. Our system offers a robust defense against phishing attacks, catering to the growing demand for secure web browsing experiences.

Keywords—Machine Learning, website extension, Phishing, cybersecurity, Random Forest classifier, Browser plugin, Real-time detection, Privacy Preservation

# Introduction

Phishing involves fraudulent attempts to acquire sensitive data, such as usernames, passwords, and credit card information, typically for malicious purposes. This is commonly executed via email spoofing or instant messaging, directing users to input personal details on a fake website resembling a legitimate one, with only the URL being the discernible difference. Phishing schemes frequently masquerade as communications from social media platforms, online auction sites, banks, or online payment processors to ensnare victims. These deceptive emails may also embed links to websites disseminating malware. Identifying phishing websites typically involves sifting through a database of known malicious sites. However, due to the ephemeral nature of phishing sites, maintaining an exhaustive directory is challenging, especially with the emergence of new ones. To tackle this challenge, machine learning methods can enhance phishing website detection, with the random forest classifier demonstrating notable efficacy. The optimal approach for end users to leverage this advancement is through implementing a browser plugin capable of real-time phishing site detection as users browse the web. Nevertheless, browser extensions have inherent constraints, being predominantly scripted in JavaScript and having restricted access to page URLs and resources. Commonly, existing plugins transmit URLs to a server for classification, raising privacy concerns and potentially introducing delays due to network latency. Furthermore, there's a possibility of the plugin failing to promptly alert the user. Given the gravity of this security concern and the privacy ramifications, we have opted to develop a Chrome browser plugin capable of classification sans reliance on an external server.

# Problem Statement

Phishing entails deceptive attempts to obtain sensitive information like usernames, passwords, and credit card details for malicious purposes. This often involves methods such as email spoofing or instant messaging, directing users to provide personal data on a fake website resembling a legitimate one, with the only notable difference being the URL. Phishing scams commonly pose as messages from social media platforms, online auction sites, banks, or online payment processors to dupe victims. Moreover, these phishing emails may include links to websites that distribute malware.  
  
Identifying phishing website usually requires searching a list of known malicious sites. However, due to the transient nature of phishing sites, it is difficult for the list to keep up with all of them, including new ones. To tackle this challenge, machine learning methods can be utilized to enhance phishing website detection. Among these methods, the random forest classifier has demonstrated promising results.  
  
The most effective way for end users to leverage this advancement is through the use of a browser plugin capable of identifying phishing sites in real time during web browsing. However, browser extensions have limitations, including being written exclusively in JavaScript and having limited access to page URLs and resources.  
  
Commonly, existing plugins transmit the URL to a server for analysis, raising privacy concerns and potentially causing delays due to network latency. Moreover, there's a risk that the plugin may not promptly alert the user. Recognizing the importance of this security issue and the associated privacy concerns, we've chosen to develop a Chrome browser plugin capable of conducting classification without relying on an external server.

# Objective

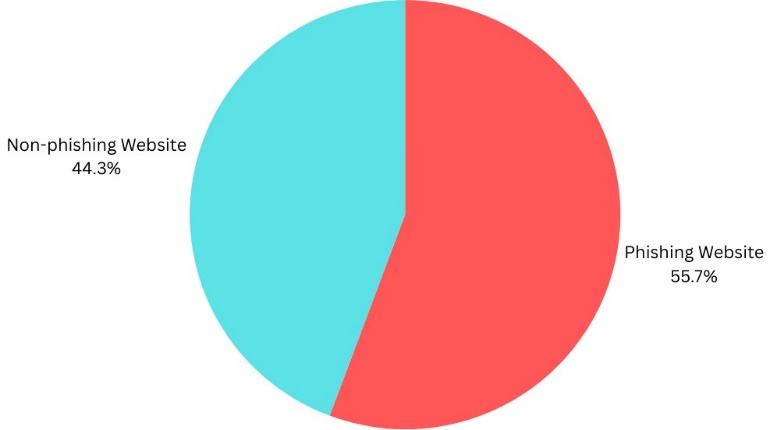
Our aim is to develop a browser plugin that swiftly alerts users when they visit a phishing website. It's crucial that the plugin operates without establishing connections with external web services to safeguard user browsing data. The detection process must be immediate, ensuring users are warned before divulging any sensitive information on the phishing site

# Scope

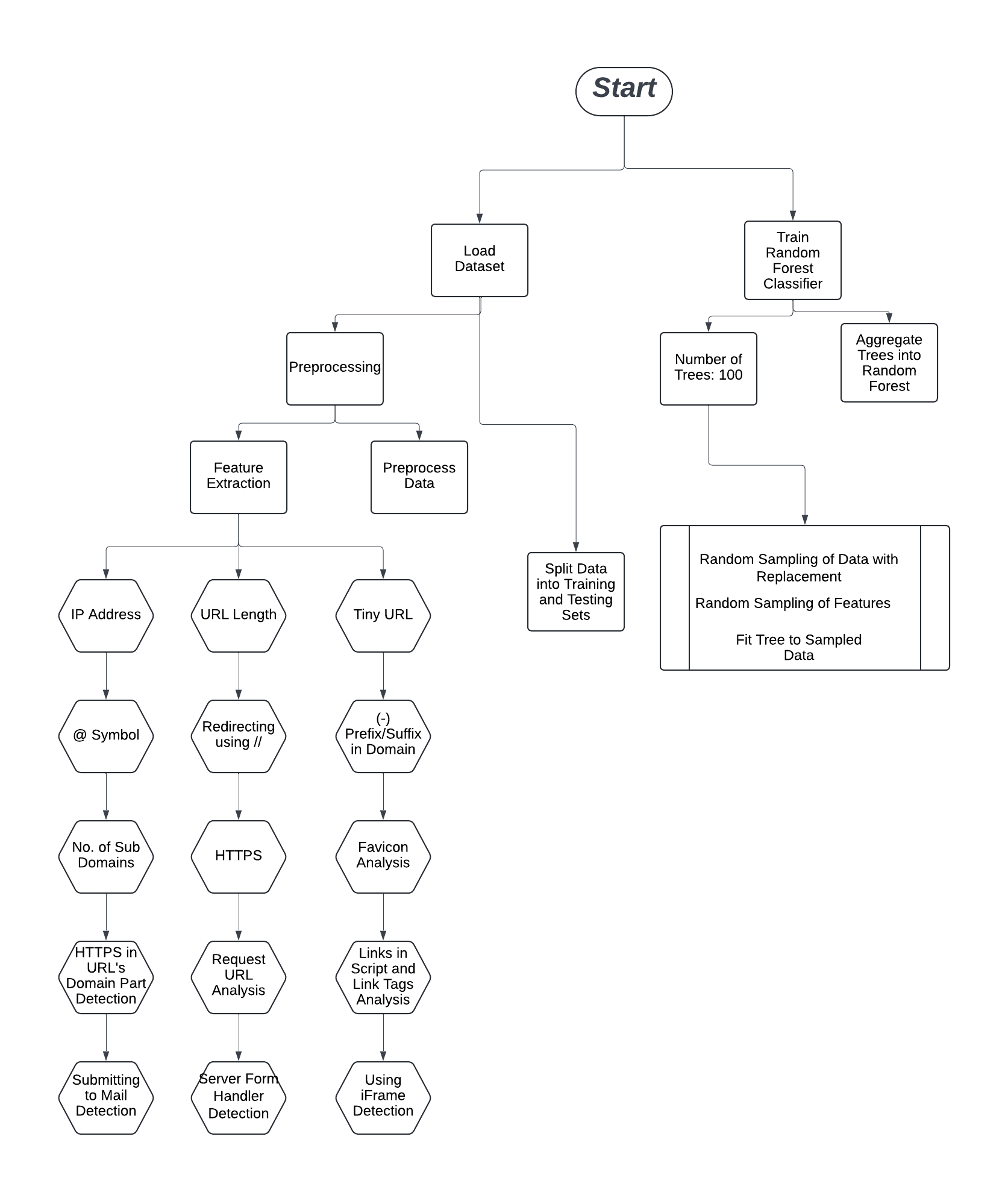
In 2021, a notable 76% of organizations reported encountering phishing attacks, aligning with the trend observed in 2017. Nearly half of the surveyed information security professionals acknowledged that the frequency of these attacks has consistently escalated since 2016. Over a three-month period in the first half of 2021, an alarming 93,570 phishing incidents targeted businesses and residents in Qatar. Considering the continuously growing number of internet users and the ongoing menace of phishing attacks, the demand for robust security solutions to protect Chrome users is pressing.

# DATASET

The test set comprises data points that are segregated from the dataset at a ratio of 70:30. Additionally, the plugin undergoes testing with websites enlisted in phishTank. Any newly discovered phishing sites are promptly incorporated into PhishTank. It is important to highlight that the plugin possesses the capability to identify new phishing sites. The results of both module testing and comprehensive system testing, along with this dataset containing 6157 phishing websites and 4898 non-phishing websites, are summarized below in a pie chart.



# Preproccing & Feature Extraction Flow chart



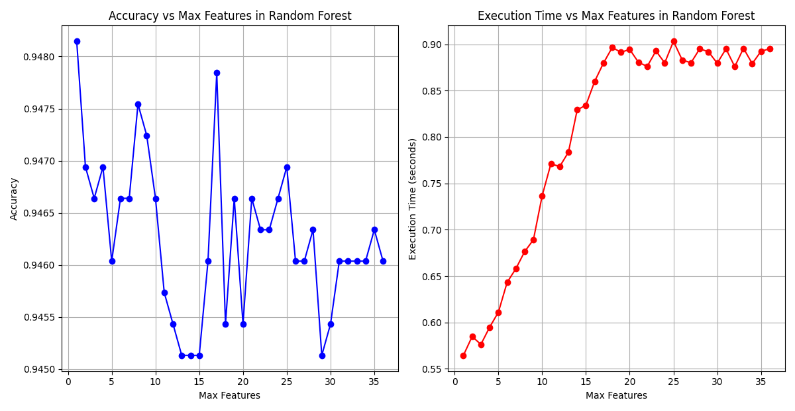
# Feature Extraction

In the feature extraction phase of our research, we meticulously analyzed the dataset to discern the impact of varying feature sets on the performance of our model. Our investigation revealed a compelling trade-off between computational efficiency and predictive accuracy.

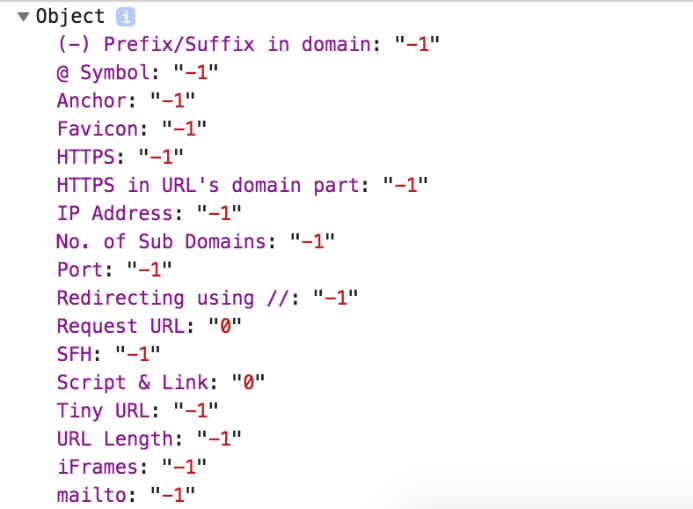
Upon scrutinizing the dataset, we observed a notable trend: when employing a reduced feature set, our model exhibited expedited computational time. However, this expedience came at the cost of a diminished accuracy rate. Conversely, by incorporating the entire feature set, computational time increased, albeit with a marked improvement in predictive accuracy.

Through rigorous experimentation, we identified a pivotal juncture, a "sweet spot," where we achieved a harmonious balance between computational efficiency and predictive accuracy. At this juncture, our model demonstrated commendable accuracy while maintaining reasonable computational efficiency, thus optimizing performance without compromising on predictive power.

The importance of balancing computational requirements and predictive accuracy is highlighted by this sophisticated comprehension of feature selection, emphasizing the strategic value in creating resilient and effective machine learning models..



In Feature Extraction phase we extracted well known 17 features out of 36 presents in dataset These features have been logged into the console. They are stored as key-value pairs, with the values being encoded within the range of -1 to 1, as previously mentioned. Here is selected feature

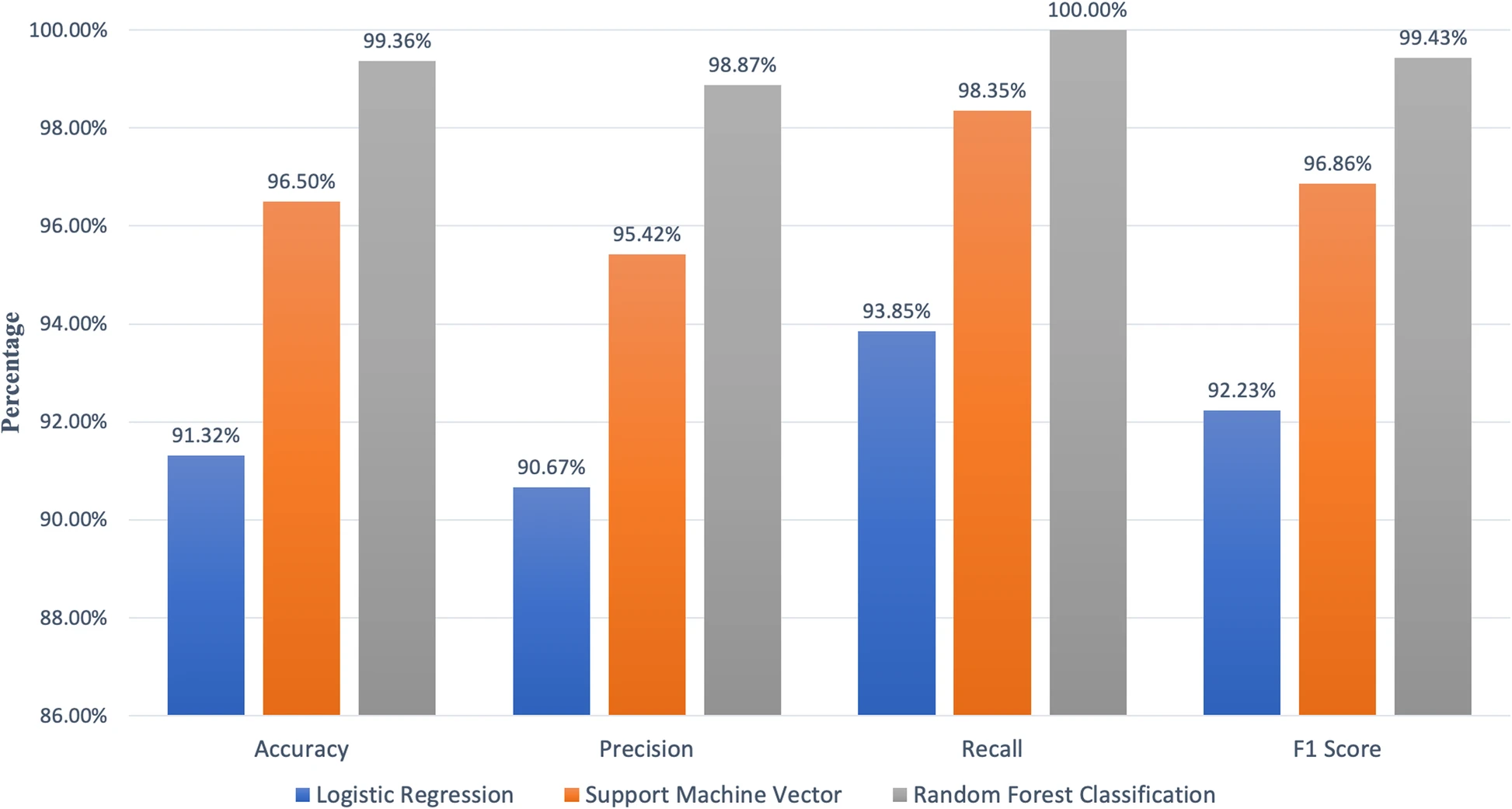


# Algorithms

In our study, we extensively explored the comparative analysis of three well-known algorithms: Logistic Regression, Random Forest Classification, and Support Vector Machine (SVM). Our objective was to uncover the strengths and weaknesses of each algorithm by evaluating their performance across multiple metrics such as recall, F1 score, accuracy, and precision. Notably, after carefully examining the generated graphs, it became apparent that Random Forest Classification outperformed the other algorithms in all metrics.

Specifically, Random Forest Classification consistently demonstrated higher recall, F1 score, accuracy, and precision scores in our experiments. This observation underscores the algorithm's robustness and effectiveness in handling the complexities inherent in the dataset. The ability of Random Forest Classification to capture nuanced patterns within the data and make accurate predictions was particularly striking. Its superior performance across multiple metrics suggests Its exceptional performance across various metrics indicates its potential suitability for a diverse range of classification tasks, spanning from healthcare diagnostics to financial forecasting.

Moreover, our findings highlight the significance of considering algorithm performance comprehensively, beyond just a single metric. While certain algorithms might excel in specific aspects, such as precision or recall, a holistic evaluation, as depicted in our graphs, provides a more nuanced understanding of their overall effectiveness. Thus, researchers and practitioners alike can leverage this insight to make informed decisions regarding algorithm selection based on the specific requirements and objectives of their applications.



**Random Forest: -**

Random forests operate as classifiers by amalgamating multiple tree predictors, where each tree's outcome hinges on a randomly chosen vector. Moreover, all trees in the forest adhere to the same distribution. When crafting a tree, 'n' signifies the number of training observations, while 'p' denotes the number of variables (features) in the training set. To establish a decision node within a tree, 'k' is selected as significantly smaller than 'p' to represent the number of variables to be considered. By employing a bootstrap sample from the 'n' observations in the training set, we gauge the tree's error during testing using the remaining observations. Consequently, 'k' variables are randomly chosen to make decisions at specific nodes within the tree, and the optimal split is determined based on these 'k' variables in the training set. Unlike other tree algorithms, random forests continually expand their trees without pruning, enabling them to adeptly handle a vast number of variables in a dataset. Additionally, during the forest construction process, they furnish an internally unbiased estimate of generalization error and can proficiently manage missing data. Nonetheless, a notable drawback of random forests is their lack of reproducibility due to the inherent randomness in the forest-building process. Furthermore, interpreting the final model and ensuing results can pose challenges due to the presence of multiple independent decision trees. In this model, we have employed 100 trees

The graph depicts how accuracy varies with the number of decision trees. Initially, accuracy rises sharply but eventually plateaus, suggesting diminishing returns with more trees. This highlights the trade-off between model complexity and accuracy, aiding informed decisions in model optimization.

# Cross Validation Score

Employing the same data for both learning the parameters of a prediction function and testing it leads to a methodological flaw known as overfitting. This occurs when a model simply memorizes the labels of the samples it has seen, achieving a perfect score but lacking the ability to make accurate predictions on new data. To mitigate overfitting, a portion of the dataset can be set aside as a validation set. Training the model on the training set and subsequently evaluating it on the validation set is a common practice. If the results meet expectations, a final evaluation is conducted on the test set.

However, dividing the data into three sets diminishes the sample size available for training and can result in differing outcomes due to the random selection of the training and validation sets

. To address this challenge, a technique called cross-validation (CV) can be employed. In standard k-fold CV, the training set is partitioned into k smaller sets. A model is trained using k-1 folds as training data and validated on the remaining portion, serving as a test set to assess performance metrics such as accuracy.

The result of a 10-fold cross-validation is presented below

= 0.94766021173256

# F1 Score

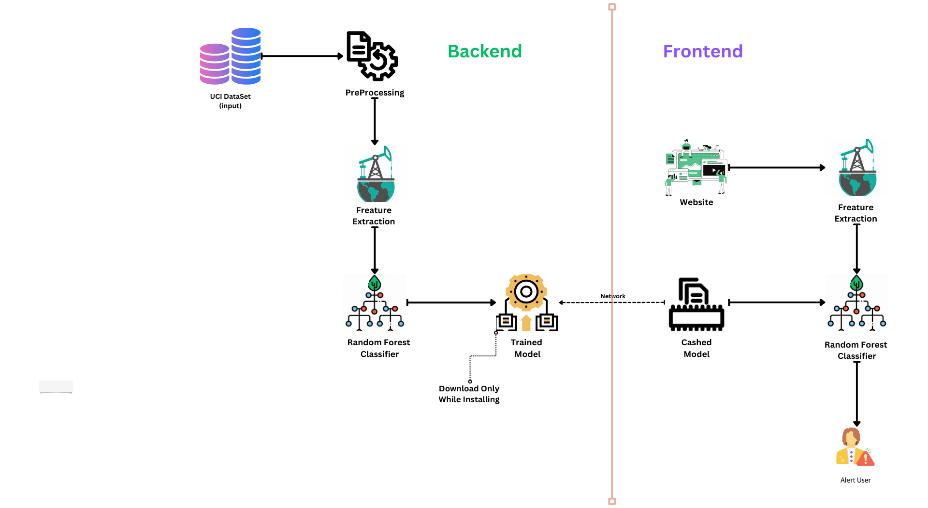
The F1 score serves as a metric for assessing the accuracy of a test, encompassing both precision and recall. Precision is calculated by dividing the number of correctly identified positive results by the total number of positive results returned by the classifier. On the other hand, recall is determined by dividing the number of correctly identified positive results by the total number of relevant samples, encompassing all samples that should have been identified as positive. The F1 score is then derived by taking the harmonic mean of precision and recall. A perfect F1 score of 1 signifies flawless precision and recall, whereas a score of 0 denotes the poorest performance. The F1 score can be construed as a balanced average of precision and recall, with equal weight given to both. Its optimal value is 1, while the lowest achievable value is 0. The formula for computing the F1 score is as follows:

F1 = 2 \* (precision \* recall) / (precision + recall). F1 = 2 x (0.8217636022514071 X 0.9631665750412315)

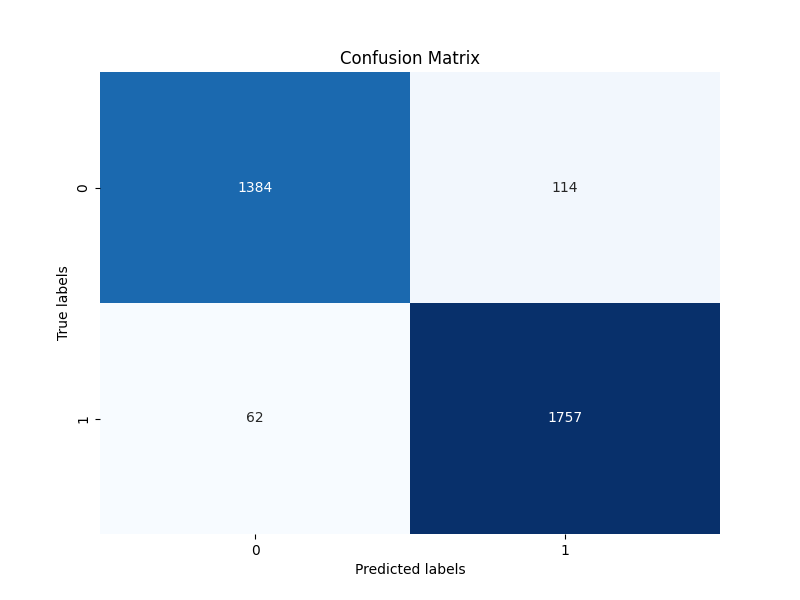
(0.8217636022514071 **+** 0.9631665750412315)

F1 = 0.8868640850417616

# WorkFLow



# Confusion Matrix



# Use Case

1. **Web Browsing Protection:** The main purpose of this system is to safeguard users during their web browsing activities. It can be seamlessly incorporated as a browser extension, providing users with timely warnings whenever they access websites that may pose a threat. By doing so, it effectively minimizes the chances of users becoming victims of phishing attacks.
2. **Real-Time Phishing Detection:** This system offers rapid detection of phishing attempts even before the web page finishes loading. It can be used to provide real-time alerts to users, ensuring they are informed about potential threats as quickly as possible.
3. **User Privacy:** One of the system's strengths is its focus on client-side implementation and the use of features that do not compromise user privacy. This makes it a suitable solution for individuals who are concerned about their online privacy.
4. **Balancing Accuracy and Usability:** While the system's accuracy may not be on par with some state-of-the-art solutions, it strikes a balance between accuracy and rapid detection. This makes it suitable for users who prioritize usability and quick warnings over absolute accuracy.
5. **Customizable Features:** The system can potentially be extended to include additional features for training. Users or organizations can tailor the system to their specific needs and threat landscape.
6. **Ongoing Development and Improvement:** The text mentions several areas for future work and enhancement, such as increasing the number of features for training, implementing result caching for frequently visited sites, and exploring the use of technologies like WorkerThreads to expedite the classification process. This implies ongoing development and improvement of the system to adapt to the evolving landscape of phishing threats.

# Conclusions

The system has the potential to become an even more effective solution in the ever-evolving landscape of phishing detection, offering users a robust defense against online threats while maintaining their privacy. Further enhancements could be achieved by expanding the feature set for training and implementing result caching for frequently visited sites, optimizing performance without compromising security. Moreover, the utilization of technologies like WorkerThreads could further expedite the classification process, ensuring prompt and efficient detection.

# Future Work

The classifier is currently trained on 17 features, which can potentially be increased without compromising the speed of detection or compromising privacy. To optimize computation, the extension can be modified to cache results of frequently visited sites. However, caution must be exercised as this may leave the system vulnerable to undetected pharming attacks. Therefore, a solution must be devised to enable result caching while maintaining the ability to detect pharming. Additionally, the classification in javascript can be enhanced by utilizing WorkerThreads, which can potentially improve the classification time. Consequently, this system offers numerous possibilities for improvements and enhancements, ultimately providing a more effective solution in the realm of phishing detection.

.

# Reference

[1] “Intelligent phishing website detection using random forest classifier,” IEEE Conference Publication | IEEE Xplore, Nov. 01, 2017. https://ieeexplore.ieee.org/abstract/document/8252051

[2] “Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation,” IEEE Conference Publication | IEEE Xplore, Oct. 09, 2020. https://ieeexplore.ieee.org/document/9277256

[3] “Phishing Detection and Prevention using Chrome Extension,” IEEE Conference Publication | IEEE Xplore, Jun. 06, 2022. https://ieeexplore.ieee.org/document/9800826

[4] A. H. Aljammal, S. Taamneh, A. Qawasmeh, and H. Bani-Salameh, “Machine learning based phishing attacks detection using multiple datasets,” International Journal of Interactive Mobile Technologies, vol. 17, no. 05, pp. 71–83, Mar. 2023, doi: 10.3991/ijim.v17i05.37575.

International Conference on Computer Applications & Information Security 2019.

[5] Dr. G. Ravi Kumar, Dr. S. Gunasekaran, Nivetha R., Sangeetha Prabha K, Shanthini G., Vignesh A. S., “URL Phishing Data Analysis and Detecting Phishing Attacks using Machine Learning in NLP”, International Journal of Engineering Applied Sciences and Technology (IJEAST) Vol. 3, Issue 8, ISSN No.2455-2143, Pages 70-75, Published: December 2018

[6] Muhammet Baykara, Zahit Ziya Gürel “Detection of Phishing Attacks”, 6th International Symposium on Digital Forensic and Security (ISDFS), 22-25 March, 2018

[7] Ishant Tyagi, Jatin Shad, Shubham Sharma, Sidharth Gaur & Gagandeep Kaur,” A Novel Machine Learning Approach to Detect Phishing Websites”.5th International Conference on Signal Processing and Integrated Networks (SPIN) Feb 2018.

[8] Tianrui Peng, Ian G. Harris, Yuki Sawa , “Detecting Phishing Attacks Using Natural Language Processing and Machine Learning” 12th IEEE International Conference on Semantic Computing 2018.

[9] S. Jagadeesan, Anchit Chaturvedi, Shashank Kumar, “URL Phishing Analysis using Random Forest”, International Journal of Pure and Applied Mathematics, 2018

[10] Ozgur Koray Sahingoz, Saide Işilay Baykal and Deniz Bulut, “Phishing Detection from URLs by using Neural Networks”, International Conference on Computer Science engineering and Applications, 2018

[11] “Phishing attacks and protection against them,” IEEE Conference Publication | IEEE Xplore, Jan. 26, 2021. <https://ieeexplore.ieee.org/document/9396693>