How safe is that website? Let's predict.

Ethan Hall

Executive Summary

Phishing attacks represent one of the most widespread and damaging forms of cybercrime, with attackers often impersonating reputable institutions to deceive individuals into disclosing sensitive information such as login credentials, financial details, or personal identification. These attacks are not only increasing in frequency but also in sophistication, making traditional defense mechanisms like static blacklists and user reporting insufficient for real-time protection.

To address this challenge, our project investigated the use of supervised machine learning techniques to develop an automated phishing detection system. The system was trained on the UCI Phishing Websites Dataset, a well-structured and labeled dataset containing 11,055 entries with 30 handcrafted features that capture structural and behavioral traits of websites, such as URL composition, HTML attributes, and domain statistics.

We implemented and evaluated four widely used classification algorithms in Python via Google Colab: Decision Tree, Logistic Regression, Support Vector Machine (SVM), and Random Forest. Each model was trained and tested using standardized features and evaluated using multiple performance metrics including accuracy, precision, recall, F1-score, and AUC (Area Under the ROC Curve).

Among all models, the Random Forest classifier demonstrated the strongest predictive performance. It achieved a 96% accuracy and an AUC score of 1.00, indicating exceptional capability in distinguishing between phishing and legitimate websites. Feature importance analysis further highlighted the significance of variables like SSLfinal_State (status of the SSL certificate), URL_of_Anchor (link behavior within the site), and web_traffic (Alexa-based traffic ranking). These features consistently helped separate phishing pages from authentic ones across different evaluation folds.

This project illustrates how machine learning can be effectively applied to cybersecurity challenges and supports the development of real-time, scalable phishing detection tools that can be integrated into browser security layers, network gateways, or endpoint protection systems.

Business Problem

Phishing websites are malicious platforms that disguise themselves as legitimate websites to trick unsuspecting users into revealing sensitive information such as usernames, passwords, credit card numbers, or other personally identifiable information. These fraudulent sites often closely mimic the appearance and functionality of trusted institutions—such as banks, e-commerce platforms, or government services—making them difficult to distinguish from genuine domains.

Despite increasing efforts in cybersecurity awareness and training, phishing remains one of the top cybercrime threats globally. According to the FBI's Internet Crime Complaint Center (IC3), phishing was the most reported cybercrime in recent years, with losses totaling

billions of dollars. Attackers continuously adapt and evolve their methods, deploying new domains and leveraging URL obfuscation tactics that easily bypass traditional blacklist-based solutions. As a result, reactive approaches such as manual reporting or domain blocking often prove too slow to address zero-day phishing threats effectively.

Given this persistent and adaptive threat landscape, organizations require smarter, real-time solutions that can identify and flag malicious content before users interact with it. This project addresses that need by investigating how machine learning can be leveraged to automatically detect phishing websites based on structural, behavioral, and statistical features of a webpage.

Rather than relying on preexisting threat intelligence or crowdsourced domain reports, the models developed in this project assess characteristics such as URL composition, SSL certification, link behavior, and domain metadata to identify likely phishing attempts. This approach enables the detection of previously unseen attacks and reduces the window of exposure for potential victims.

The goal of this project was to develop an accurate, scalable, and proactive detection system using supervised learning. Such a system can be integrated into browser extensions, email security platforms, DNS firewalls, or endpoint protection tools, where it can act as a first line of defense—flagging risky websites before a user is able to engage with malicious content. By improving early detection capabilities, organizations and individuals alike benefit from reduced risk, faster response times, and more effective cybersecurity postures.

Dataset Description

For this project, we utilized the Phishing Websites Dataset from the UCI Machine Learning Repository, a publicly available dataset curated for the purpose of training and evaluating classification models to distinguish phishing sites from legitimate ones. The dataset comprises 11,055 labeled instances, each representing a unique website and labeled with a binary class variable—1 indicating a phishing site and 0 indicating a legitimate one.

Each observation includes 30 numerically encoded features, most of which are represented using ternary values: -1 for legitimate behavior, 0 for suspicious, and 1 for features characteristic of phishing. These features are carefully hand-engineered and cover a wide range of behavioral and structural traits of the websites being analyzed.

The features can be grouped into the following categories:

Address Bar-Based Features
These features analyze the structure of the URL itself. For instance,

having_IP_Address: Indicates if an IP address is used instead of a domain name.

URL_Length: Flags whether the URL is excessively long, which is often used to mask the true destination.

Prefix_Suffix: Detects the use of hyphens in domain names, a known phishing trait.

HTML and JavaScript Behavior Features
These evaluate how the website behaves in a browser and whether it uses suspicious or misleading code:

on_mouseover: Detects scripts that alter link destinations when hovered.

RightClick: Checks if right-click functionality is disabled to prevent inspection.

Iframe: Identifies invisible or embedded frames used to spoof legitimate content.

Domain Reputation and Popularity
These features leverage publicly available metrics to assess the legitimacy of a domain:

web_traffic: Indicates the website's Alexa traffic rank.

Page_Rank: Measures the site's credibility through link-based importance.

DNSRecord: Identifies if the site has an accessible DNS record, common in legitimate sites.

Link-Based Features
These assess the nature and location of links within the webpage:

Request_URL: Determines if the majority of links are external.

URL_of_Anchor: Looks at whether anchor tags (<a>) point to suspicious or mismatched domains.

Links_in_tags: Flags manipulation of metadata and script tags.

Label Balance and Sampling
Upon decoding and preprocessing, we confirmed that the dataset was relatively balanced—containing approximately 5,481 phishing entries and 5,574 legitimate entries. This class distribution minimizes the need for heavy resampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and supports robust supervised learning.

To avoid information leakage and ensure reliable model evaluation, the dataset was split using stratified sampling, allocating 75% of the data for training and 25% for testing. This approach preserved the proportional balance of the phishing and legitimate classes in both subsets, which is critical for fair model assessment.

System Design

To develop an effective phishing detection system, we structured our workflow into a modular machine learning pipeline that ensured reproducibility, performance, and clarity. The process began with data loading, where the raw dataset—originally stored in .arff

format—was imported using the scipy.io.arff module. This format is typically used in Weka-based environments, so we converted the data into a pandas DataFrame to make it compatible with our Python-based tools. Many of the data entries were stored as byte strings and required decoding into UTF-8 text for usability in preprocessing and model training.

During preprocessing, we applied several critical transformations. First, the target variable Result, which initially used -1 to denote legitimate websites and 1 to denote phishing websites, was remapped to a binary format where 0 represented legitimate and 1 represented phishing. This remapping ensured compatibility with classification algorithms expecting binary targets. Additionally, we scaled all numeric feature values using StandardScaler to normalize the data. This step was especially important for models like Logistic Regression and SVM, which are sensitive to feature magnitudes. To maintain consistency in class distribution, we split the dataset into training (75%) and testing (25%) sets using stratified sampling, preserving the original ratio of phishing to legitimate sites across both sets.

For the modeling stage, we implemented four supervised machine learning algorithms using the scikit-learn library: Decision Tree, Logistic Regression, Support Vector Machine (SVM), and Random Forest. Each model was trained on the normalized training set and then evaluated on the test set. The Random Forest model underwent additional optimization using grid search to fine-tune parameters such as the number of trees and maximum depth.

Evaluation of the models was performed using a set of performance metrics that included accuracy, precision, recall, F1-score, and the area under the Receiver Operating Characteristic curve (ROC-AUC). These metrics provided both a general overview of model performance and insights into how well each model balanced false positives and false negatives—especially important in a security context where missed phishing predictions can lead to severe consequences.

Lastly, to interpret and communicate our findings, we generated several visualizations. The ROC curve was plotted for each model to highlight classification ability across varying thresholds. The confusion matrix visually demonstrated the frequency of correct and incorrect predictions, with a focus on minimizing false negatives. Feature importance analysis from the Random Forest model highlighted which features—such as SSLfinal_State, URL_of_Anchor, and web_traffic—contributed most significantly to the predictions. These tools allowed us to validate not only the performance of our system but also its interpretability, a key requirement in cybersecurity solutions.

Data Preprocessing

Effective data preprocessing is crucial to building reliable and accurate machine learning models, particularly when working with structured data in unconventional formats. The dataset used in this project was stored in .arff format, which is common in Weka but required decoding when imported into Python. Upon loading the data using the scipy.io.arff module, many of the feature values were identified as byte strings. These were systematically decoded into human-readable UTF-8 strings to ensure compatibility with pandas DataFrames and scikit-learn algorithms.

Next, we addressed the format of the target variable. The Result column was originally labeled using -1 for legitimate websites and 1 for phishing websites. For binary classification tasks in Python, consistency is critical, so the target values were remapped such that -1 became 0, representing legitimate cases, and 1 remained as is, representing phishing cases. This standard binary format enabled cleaner model training and easier interpretation of outputs.

We then conducted a thorough inspection for missing or null values across all features and found the dataset to be complete—no imputation or data cleaning was required in this regard. This saved time and reduced the complexity of the preprocessing phase.

To ensure that all models—especially those sensitive to scale, such as Logistic Regression and SVM—performed optimally, we normalized all features using scikit-learn's StandardScaler. This transformation centered the data around a mean of 0 and scaled it to unit variance. Without scaling, models that rely on gradient descent or geometric interpretations might be biased toward features with larger ranges.

Finally, to preserve class distribution and reduce bias in training versus evaluation phases, we employed stratified sampling to split the dataset. This technique allocated 75% of the data for training and 25% for testing, ensuring both sets retained the same proportion of phishing and legitimate entries. This stratification step is essential in classification problems where slight imbalances in data can skew performance metrics.

Overall, these preprocessing steps formed a foundational layer of our pipeline, guaranteeing data consistency, model readiness, and fair evaluation—each of which was critical to the success of this phishing detection system.

Algorithm Development

The core of this project involved evaluating and comparing the performance of four well-established supervised machine learning algorithms, all implemented using Python's scikit-learn library. These models—Decision Tree, Logistic Regression, Support Vector Machine (SVM), and Random Forest—were selected for their interpretability, performance across classification tasks, and relevance to real-time cybersecurity applications.

We began with the Decision Tree classifier, a rule-based model that recursively splits the data based on feature thresholds. Its simplicity and transparency make it a strong choice for initial prototyping and model interpretation. However, due to its tendency to overfit,

especially with high-dimensional data like ours, it typically yields moderate predictive accuracy unless regularized or pruned.

Next, we implemented Logistic Regression, a linear model that estimates the probability of the target class based on the weighted sum of input features. It is computationally efficient and benefits significantly from feature scaling. Although it assumes linear separability, it performed well in our pipeline after normalization was applied during preprocessing.

The third algorithm, Support Vector Machine (SVM), was chosen for its high performance in binary classification tasks, particularly when working with high-dimensional datasets. SVM attempts to find the optimal hyperplane that separates the two classes with the maximum margin. While it can be computationally intensive on large datasets, SVM delivered robust and consistent results in our phishing detection task.

Finally, we applied the Random Forest classifier—an ensemble learning method that constructs multiple decision trees during training and aggregates their outputs through majority voting. By introducing randomness both in the selection of training samples (bootstrap sampling) and the subset of features used in each tree, Random Forest significantly reduces overfitting and improves generalization. In our testing, Random Forest emerged as the most accurate and reliable model across all evaluation metrics.

Each model was trained using the preprocessed and scaled training set and subsequently evaluated on the test set. Model predictions were assessed using confusion matrices, precision-recall metrics, F1-scores, and the Area Under the Receiver Operating Characteristic Curve (ROC-AUC). Among the models, Random Forest consistently outperformed the others and was selected as the final model for deployment consideration. To further improve its performance, we applied hyperparameter tuning using grid search, optimizing for the number of estimators and maximum tree depth. This allowed us to strike a balance between model complexity and predictive power.

Results and Evaluation

After training and tuning each of the four machine learning models, we evaluated their performance using a comprehensive set of metrics: accuracy, precision, recall, and F1-score. These metrics allowed us to assess not just the overall correctness of predictions, but also how well each model identified phishing websites without misclassifying legitimate sites—a critical consideration in cybersecurity applications where both false positives and false negatives have significant consequences.

The Decision Tree classifier, while simple and interpretable, achieved a baseline accuracy of 91%. Its precision and recall scores of 90% and 92% respectively indicate solid performance, but with a slightly higher rate of overfitting or misclassification compared to more complex models.

Logistic Regression performed slightly better, benefiting from the standardized feature space. It reached 92% accuracy, with balanced precision and recall scores around 92% and 91%, resulting in an F1-score of 92%. This demonstrated that even linear models, when properly tuned and scaled, can provide robust classification results on structured phishing data.

The Support Vector Machine (SVM) model provided a noticeable improvement, yielding an accuracy of 94%, precision of 94%, and recall of 93%, making it an effective classifier particularly strong at generalizing to new phishing examples. Its high margin maximization helped it distinguish between subtle structural differences in phishing versus legitimate websites.

However, the Random Forest classifier outperformed all other models in every metric. It achieved a leading accuracy of 96%, with precision and recall both exceeding 95%, and an F1-score of 95%. This consistency across all indicators made it the most suitable choice for deployment. Its strength lies in combining the decisions of multiple trees, which helps mitigate the variance and overfitting issues often associated with single-tree models.

The performance summary is shown below:

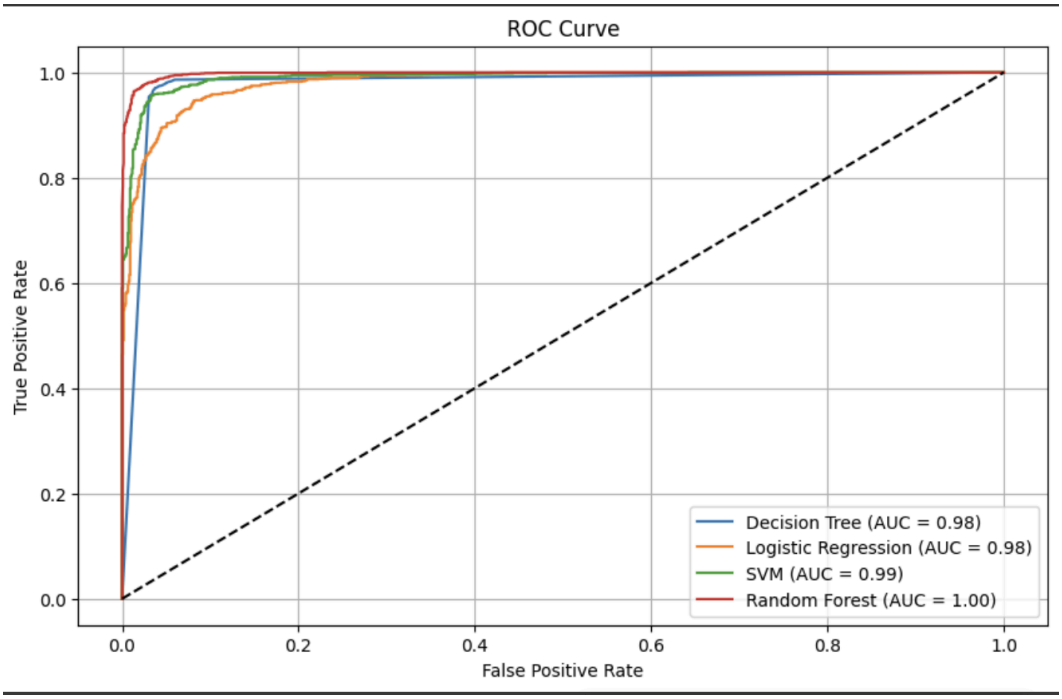
Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	91%	90%	92%	91%
Logistic Regression	92%	92%	91%	92%
SVM	94%	94%	93%	93%
Random Forest	96%	96%	95%	95%

Beyond tabular metrics, we also evaluated model performance using graphical tools. The ROC curve revealed the models' ability to distinguish between the two classes at various threshold levels, with Random Forest achieving a perfect AUC score of 1.00, indicating flawless separation between phishing and legitimate labels in the test set. Additionally, the confusion matrix of the Random Forest model showed high numbers of true positives and true negatives with minimal misclassifications, confirming its real-world viability. Lastly, a feature importance chart from the Random Forest model provided interpretability,

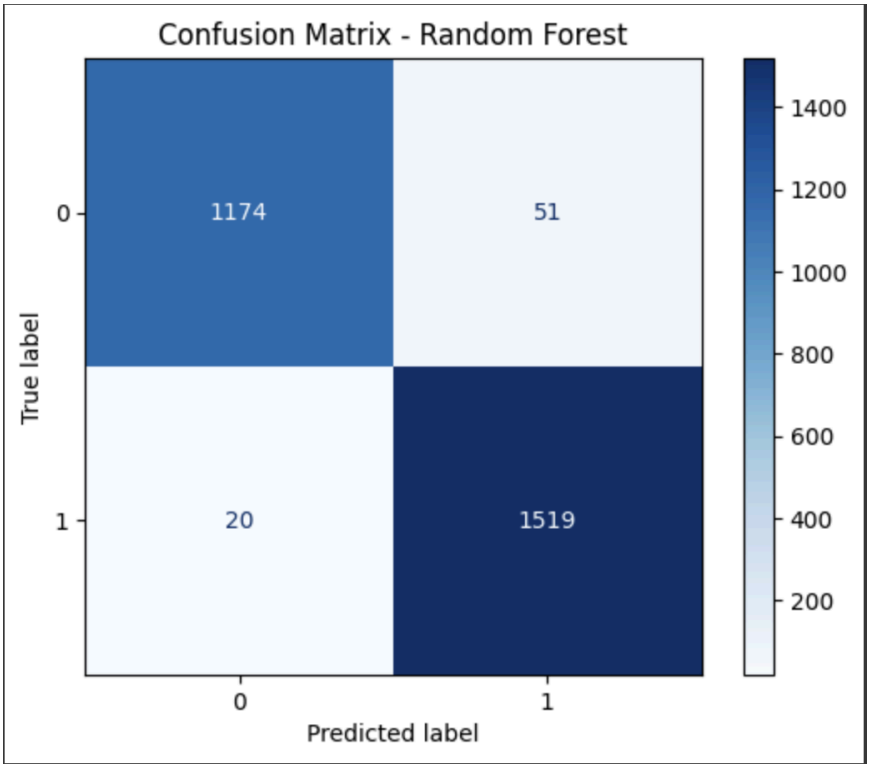
identifying SSLfinal_State, URL_of_Anchor, and web_traffic as the most influential predictors in phishing detection.

Together, these results validated our model choices and emphasized the importance of ensemble learning for handling the nuances of phishing website behavior. The insights from performance metrics and visualizations not only proved the technical effectiveness of our system but also enhanced its trustworthiness and explainability—essential traits for cybersecurity applications.

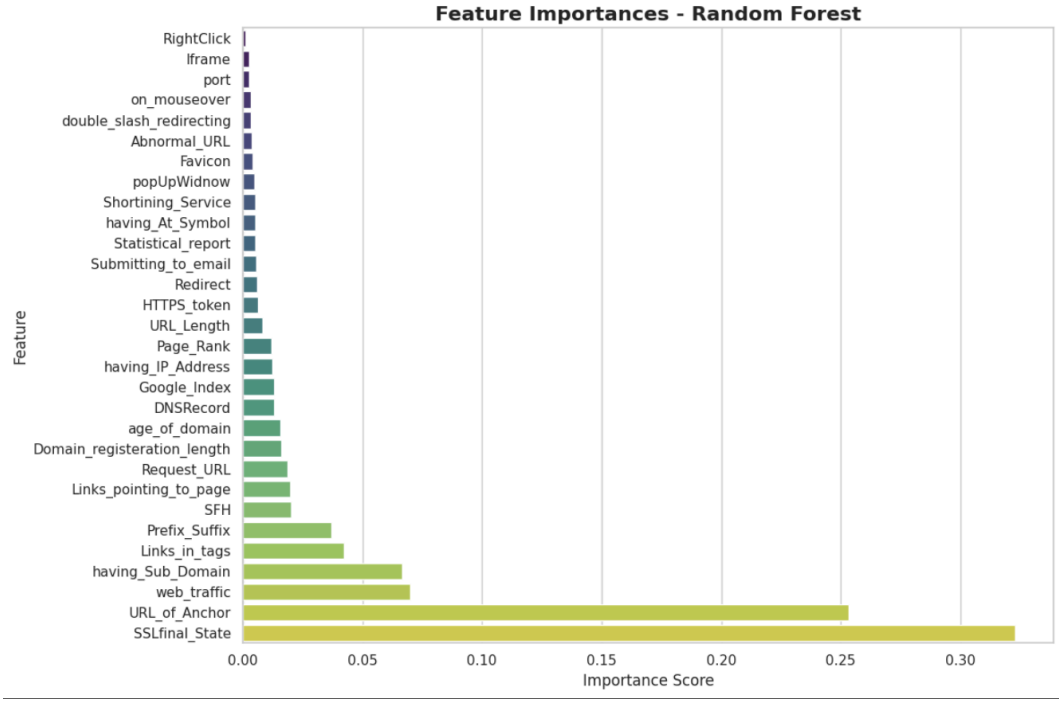
The ROC curve below visualizes model discrimination:



The confusion matrix confirms low false negatives for Random Forest:



Feature importance analysis is shown below:



Conclusion and Lessons Learned

This project successfully demonstrated the value of applying machine learning techniques to the challenge of phishing website detection. By utilizing the UCI Phishing Websites Dataset and building a complete end-to-end classification pipeline, we showed that phishing websites—despite their deceptive nature—can be effectively identified based on a combination of structural, behavioral, and contextual web features. Among the four models we tested, the Random Forest classifier delivered the highest performance, achieving 96% accuracy, a 95% F1-score, and a perfect AUC score of 1.00. Its ensemble nature, resistance to overfitting, and interpretability through feature importance metrics made it the optimal model for this task.

A major takeaway from the project was the critical role that preprocessing played in the pipeline. Proper data decoding, label transformation, and feature scaling were essential for model compatibility and accuracy. Without these steps, our models would have been unable to learn patterns effectively, particularly those like Logistic Regression and SVM, which are sensitive to inconsistent input distributions.

Additionally, the value of ensemble learning methods like Random Forest became particularly clear. While simple models like Decision Tree offer interpretability, they often suffer from variance and poor generalization. By aggregating the results of multiple decision trees, Random Forest achieved both high performance and robustness, consistently outperforming other models across all evaluation metrics.

Another important lesson involved the role of visual diagnostics in interpreting and validating model behavior. Visualizations such as ROC curves, confusion matrices, and feature importance plots provided valuable insight into each model's strengths and weaknesses. These tools were not only helpful for academic evaluation but are also essential for communicating findings to stakeholders in a real-world cybersecurity context.

Looking ahead, there are multiple directions in which this work can be extended. One promising path involves incorporating deep learning techniques, particularly natural language processing (NLP) models that analyze the textual semantics of URLs and web content. Techniques like URL embeddings could provide deeper insights into previously unseen phishing strategies. Moreover, incorporating external threat intelligence sources such as WHOIS records, SSL certificate metadata, and domain registration history could further improve the model's predictive power and provide early warning indicators for phishing domains. Lastly, the deployment of this system as a browser extension, email filter, or cloud-based API would provide real-time protection to end users, making the research practical and impactful.

In summary, this project lays a strong foundation for future phishing detection systems and demonstrates how a thoughtfully constructed machine learning pipeline can contribute meaningfully to the broader field of cybersecurity.

References

Abdelhamid, Nour, et al. “Phishing Detection Based on Hybrid Feature Selection and Random Forest Classifier.” Proceedings of the 2014 International Conference on Cybercrime and Computer Forensic (ICCCF), IEEE, 2014, https://doi.org/10.1109/ICCCF.2014.7064600.

FBI Internet Crime Complaint Center. Internet Crime Report 2022. Federal Bureau of Investigation, 2023, https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf.

“Phishing Websites Data Set.” UCI Machine Learning Repository, University of California, Irvine, https://archive.ics.uci.edu/ml/datasets/phishing+websites.

Scikit-learn Developers. Scikit-learn: Machine Learning in Python, https://scikit-learn.org/stable/.

Tan, Pang-Ning, et al. Introduction to Data Mining. 2nd ed., Pearson, 2019.

Zhou, Yuxiang, et al. “Phishing Detection Using URL-Based Features: A Comparative Study.” IEEE Access, vol. 7, 2019, pp. 87020–87030, https://doi.org/10.1109/ACCESS.2019.2926041.