Cybersecurity: Phishing Website Classification

Devananth V - EP20BTECH11004
Indian Institute of Technology, Hyderabad
iith.ac.in

Abstract

Phishing has been recognized as the easiest and most widespread cyber-security threat as it targets the weakest link in the security chain, the user. This paper is dedicated to the identification of phishing websites using machine learning techniques (MLTs) as a robust mechanism. The dataset utilized for this research is sourced from 'Mendeley data,' comprising 11,430 URLs with 87 extracted features. The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs. Our primary objective is to conduct a comparative analysis of various machine learning models, with the aim of identifying the optimal model through a systematic optimization process.

1. Introduction

Phishing is the practice of sending fraudulent communications that appear to come from a reputable source. The goal is to steal sensitive data such as credit card and login information or to install malware on the victim's machine. Hackers do not need to crack any complex cipher code nor breach a hard firewall. Instead, they simply send emotional, critical, or sensible e-mails urging recipients to introduce their personal credentials by clicking on a link. Recipients are then redirected to fake web that look very similar to those targeted authentic websites.

During the pandemic, there was a notable increase in phishing activities. This increase can be attributed to the increased demand for current information during this period, which presented an opportunity for malicious actors to impersonate trusted entities, such as health services and government websites.

To get a better understanding,, let us consider an incident in the IIT Hyderabad campus:

A simulation attack was conducted by the Computer Center, IIT Hyderabad in collaboration with Kludge, networking club, IIT Hyderabad as part of their Cybersecurity campaign. Their mode of attack was phishing emails. The subjects of the above mentioned phishing emails were:

- WiFi Password Expiration Notification.
- Avail the Chat GPT Plus Subscription.
- Reviewing the APAR Appraisal Form.
- Additional Funds Allocated to Your Department.
- Urgent Report Submission.

These emails were sent from the domain "iith-ac.c-0m.com".

The use of special characters in the above domain and its resemblance to an existing domain is an indication for it to be a phishing website. Similarly, the dataset includes approximately 87 features derived from the URL, its content, and external records to determine its legitimacy.

Thus, Machine learning could be used for classifying phishing websites as it offers adaptability to evolving attack methods, enables automated and large-scale analysis of datasets, excels in pattern recognition, provides real-time detection, reduces false positives, enhances accuracy through continuous learning, and allows for behavioral analysis, all of which contribute to effective and efficient identification of phishing threats.

2. Literature Review

"TOWARDS BENCHMARK DATASETS FOR MACHINE LEARNING BASED WEBSITE PHISHING DETECTION: AN EXPERIMENTAL STUDY"

Link: https://arxiv.org/pdf/2010.12847.pdf

2.1. Objective

To propose and experiment a general scheme for building reproducible and extensible datasets for website phishing detection, and to evaluate the performance of different machine learning techniques and features on the built dataset.

2.2. Methodology

URLs are collected from various sources and preprocessed, from which 87 features are extracted under different classes(URL, Content, External). The dataset is then balanced and shuffled. Two metrics are used for performance Evaluation, Accuracy and Macro F1 Score.

The classifiers used are:

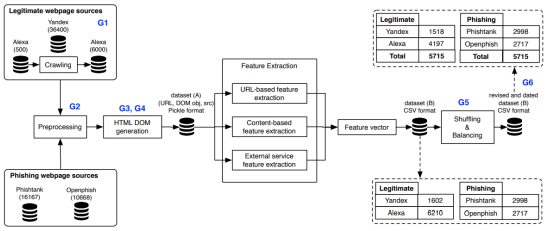

Figure 1. Dataset Preparation

- Naive Bayes
- Support Vector Model
- Logistic Regression
- Decision Tree
- Random Forest

The following experiments were done:

- Best model for single or combination of feature classes
- Combination of different models
- Feature selection
- Feature extraction runtime analysis

2.3. Results

A scheme for constructing benchmark datasets that canenable comparison, replication, and extension of website phishing detection systems was presented.
Random Forest was found to give the best accuracy for all the classes of features. External-based features provide the best accuracy, followed by URL-based and then Content-based. Using hybrid features was found to be better than using single class of features.
It was found that chi-square filter gave the best accuracy with less number of features and that filter ranking together with incremental removal of less important features improved the performance. Wrapper methods failed to improve the accuracy compared with filter methods and using 'Boruta algorithm' was much faster than 'WrapperSubsetEval'.
Extraction time of different features and feature classes was investigated. It was found that content-based features were the most time consuming and that some hyperlinkbased features also caused severe network delays.

3. Methodology

3.1. Dataset Description

The Dataset at 'Mendeley data' was designed to be used as a a benchmark for machine learning based phishing detection systems. The dataset includes 11430 URLs with 87 extracted features. The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs. The Datasets was constructed as of May 2020.

Link: https://data.mendeley.com/datasets/c2gw7fy2j4/3
It has features from three different classes:

- 56 extracted from the structure and syntax of URLs,
- 24 extracted from analyzing their HTML contents,
- and 7 were extracted by querying external services.

The datasets are divided into 2:

- Dataset A: contains a list a URLs together with their Document Object Model (DOM) tree objects
- Dataset B: contains the extracted feature values that can be used directly as input to classifiers for examination.

3.2. Machine Learning Models

3.2.1 Support Vector Model

It is a part of Supervised Learning paradigm.
The objective function has two parts: the first aims to maximize the separation between classes, while the second works to minimize misclassifications. The parameter "C" is the cost assigned to mistakes, treating errors in both positive and negative cases equally.
Objective Function:

$$\min \left(||w||^2 + C \sum_{i=1}^{n} \xi_i \right), \quad (1)$$

$$\text{where,} \quad y_i(w \cdot f(x_i) + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad (2)$$

3.2.2 Logistic Regression

It is a part of Supervised Learning paradigm.
The algorithm applies the logistic function (sigmoid) to a linear combination of input features, producing a probability score. A threshold is then set to classify instances into one of the two classes.
Objective Function:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] \quad (3)$$

$h_\theta(x^{(i)})$ is the sigmoid (logistic) function applied to the linear combination of input features x.

3.2.3 Decision Tree

It is a part of Supervised Learning paradigm
. It works by recursively partitioning the data into subsets based on the most significant attribute at each node. The goal is to create a tree-like structure where each internal node represents a decision based on a feature, each branch represents the outcome of that decision, and each leaf node represents the final prediction or value.
Objective Function: Decision tree does not have a single objective function that summarizes the entire tree-building

process, but you can express the impurity or information gain at a specific node using a criterion such as Gini impurity or entropy.

$$\text{Gini index } G = \sum_c \Pi_c(1 - \Pi_c) \tag{4}$$

$$\text{where } \Pi_c = \frac{\text{no.of points in the subspace with label c}}{\text{no.of points in the subspace}} \tag{5}$$

3.2.4 Random Forest

It is a part of Supervised Learning paradigm.
Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve overall performance and robustness. It operates by constructing a multitude of decision trees during training and outputs the mode (classification) or average (regression) prediction of the individual trees. Random Forests introduce randomness in two key ways: by using a random subset of the training data for each tree (bootstrapping) and by considering a random subset of features at each split. This helps reduce overfitting and increases the model's generalization ability.
Objective Function: As it is a ensemble of decision trees there isn't a single objective function that summarizes the entire Random Forest; instead, the focus is on the individual decision trees and their combination.

3.3. Performance Metrics

We use the following metrics to compare the models:

- **Time Taken:** Measures the computational efficiency of each model, indicating the duration required for training and prediction.
- **Accuracy:** Calculates the proportion of correctly classified instances, providing an overall measure of model correctness.
- **Macro F1 Score:** Computes the harmonic mean of precision and recall across all classes, offering a balanced assessment of performance, particularly in imbalanced datasets.
- **Recall:** Measures the model's ability to correctly identify all instances of a given class, indicating its sensitivity.
- **Log Loss:** Evaluates the model's probabilistic predictions, penalizing incorrect classifications with higher confidence.
- **Confusion Matrix:** Visualizes the model's classification performance, showing the counts of true positives, true negatives, false positives, and false negatives for each class.

3.4. Workflow

3.4.1 Extracting the data

We get the 'x' value by dropping the columns 'URL' and 'status' from the dataset. 'y' value from the status column of the dataset by assigning '1' for legitimate and '0' for phishing.

3.4.2 Splitting the data for Training and Testing

The data is split into training and testing/validation data, with 30% of the data as testing data. This is done using the *'train_test_split'* function available in the *sklearn.model_selection* library.

3.4.3 Raw Data vs Scaled Data

Data Scaling: Standardization is used to scale the 'X' values, centering it around a mean with a unit standard deviation. We use the 'fit trasfrom' and 'transform' function within 'StandardScalar' class to standardize the training data and test data respectively.
This analysis aims to evaluate the impact of data scaling on model performance relative to the use of raw data.

3.4.4 Cross Validation

K-fold cross-validation assesses a model's generalization by partitioning the data into 'k' subsets (folds). It iteratively trains and tests the model 'k' times, using each fold once as the test set and the rest as training data. The averaged performance across all iterations provides a more reliable estimate of generalization, effectively reducing overfitting, bias, and variance compared to a single train-test split. We employ 10-fold cross-validation.
Average Accuracy: Provides a more robust evaluation of model performance than standard accuracy, mitigating overfitting due to the train-test split.
Variance: Quantifies model sensitivity to training data variations, indicating the extent of overfitting introduced by the train-test split.

3.4.5 Hyperparameter Tuning

Support Vector Model:
kernel: ['linear', 'rbf']
C: [1, 10, 20]
gamma: ['auto', 'scale']

Logistic Regression:
penalty: ['l1', 'l2']
C: np.logspace(0, 10, 6)

Decision Tree:
max_depth: [25, 30, 25]
min_samples_leaf: [1, 5, 10]
min_samples_split: [25, 30, 35]

Random Forest:
max_depth: [20, 25, 30]
max_leaf_nodes: [400, 450, 475]
min_samples_split: [1, 5]

GridSearchCV is a technique in machine learning that systematically explores a predefined hyperparameter grid to find the optimal configuration for a model. It uses 'k-fold cross-validation', we use k = 10, to evaluate the performance of each combination of hyperparameters and selects the best parameters based on average performance.

3.4.6 Majority Voting

Majority voting, implemented using scikit-learn's "VotingClassifier", combines predictions from our four models. Hard voting selects the most frequently predicted class, while soft voting averages predicted probabilities, choosing the class with the highest average. In particular, soft voting allows for the application of weights to individual models, reflecting their relative confidence or performance.

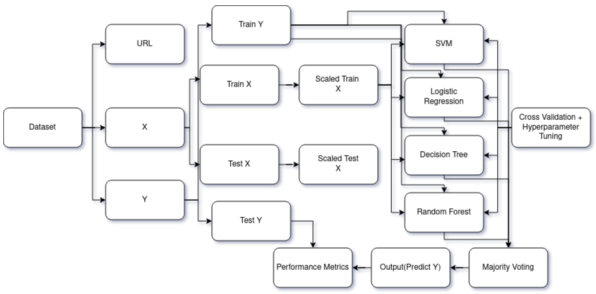

Figure 2. Workflow

4. Experimental Results

4.1. Raw Data vs Scaled Data

Raw Data

	Time Taken	Accuracy	Macro F1 Score	Recall	Log Loss
svm	4.553774	0.5917	0.4309	0.3060	14.7160
logistic_regression	0.075710	0.7927	0.7962	0.8020	7.4736
decision_tree	0.142416	0.9318	0.9326	0.9353	2.4597
random_forest	1.453395	0.9679	0.9683	0.9700	1.1563

Scaled Data

	Time Taken	Accuracy	Macro F1 Score	Recall	Log Loss
svm	1.353947	0.9603	0.9608	0.9613	1.4296
logistic_regression	0.338793	0.9484	0.9491	0.9521	1.8605
decision_tree	0.142673	0.9286	0.9296	0.9336	2.5753
random_forest	1.467035	0.9668	0.9672	0.9694	1.1983

Figure 3. Metrics for Raw and Scaled Data

The results demonstrate that data scaling significantly enhances the performance of SVM and Logistic Regression models, but has a negligible impact on Decision Tree and Random Forest models.

	Time Taken	Accuracy	Macro F1 Score	Recall	Log Loss	Average Accuracy	Variance
svm	1.012546	0.9598	0.9601	0.9590	1.4506	0.9532	0.00003
logistic_regression	0.334662	0.9487	0.9493	0.9521	1.8500	0.9415	0.00007
decision_tree	0.091805	0.9335	0.9343	0.9353	2.3966	0.9292	0.00007
random_forest	1.166574	0.9685	0.9689	0.9717	1.1352	0.9642	0.00002

Figure 4. Performance Metrics

4.2. Cross Validation

We used a 10-fold cross-validation for model evaluation. The first five metrics in the table represent the performance of the model, selected for its highest cross-validation accuracy, against the held-out test data (unseen data). The final two metrics reflect the model's overall performance on the entire training dataset.
Based on Average Accuracy, the performance hierarchy is: Random Forest > Support Vector Machine > Logistic Regression > Decision Tree.
The models, ranked by variance (overfitting susceptibility): Decision Tree > Logistic Regression > Support Vector Machine > Random Forest.

4.3. Hyperparameter Tuning

The best hyperparameters found via GridSearchCV are:

SVM:
C: 10
gamma: scale
kernel: rbf

Logistic Regression:
C: 100.0
penalty: l2

Decision Tree:
max_depth: 25
min_samples_leaf: 5
min_samples_split: 30

Random Forest:
max_depth: 25
max_leaf_nodes: 450
min_samples_split: 1

	Time Taken	Accuracy	Macro F1 Score	Recall	Log Loss
svm	461.824334	0.9615	0.9618	0.9602	1.3875
logistic_regression	58.042618	0.9495	0.9502	0.9532	1.8185
decision_tree	22.324550	0.9335	0.9337	0.9267	2.3966
random_forest	200.927664	0.9688	0.9692	0.9717	1.1247

Figure 5. Metrics of the refitted/tuned models

The *Time Taken* values indicate the time required to train all grid-searched hyperparameter combinations with cross-validation.

4.4. Majority Voting

	Time Taken	Accuracy	Macro F1 Score	Recall	Log Loss
hard	6.262948	0.9627	0.9627	0.9527	1.3455
soft	5.919712	0.9653	0.9657	0.9659	1.2509

Figure 6. Hard and Soft Majority Voting

As expected, soft majority voting's accuracy is higher than hard voting's, as shown in the table.

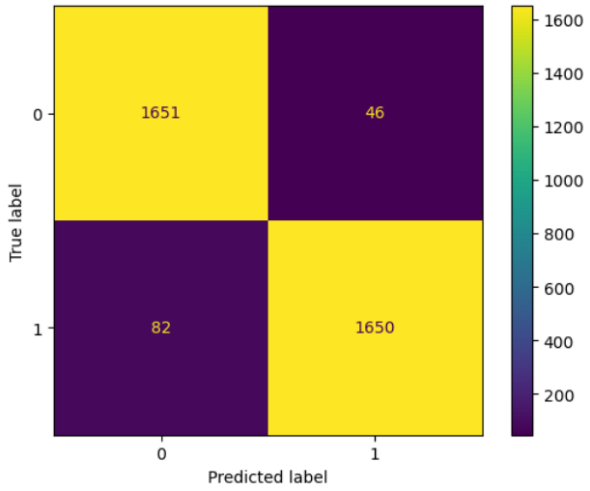

Figure 7. Confusion Matrix of Hard Majority

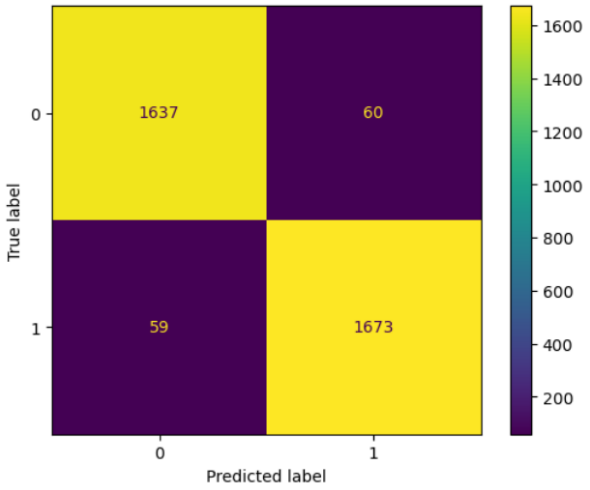

Figure 8. Confusion Matrix of Soft Majority

5. Discussion

- Random Forest exhibits the highest average accuracy and lowest variance, but requires increased training time.
- Support Vector Machine demonstrates performance comparable to Random Forest, with slightly lower accuracy but a slightly higher training time, and variance.
- Logistic Regression offers moderate performance with average training time.
- Decision Tree achieves the fastest training time, but suffers from the lowest accuracy and highest variance.

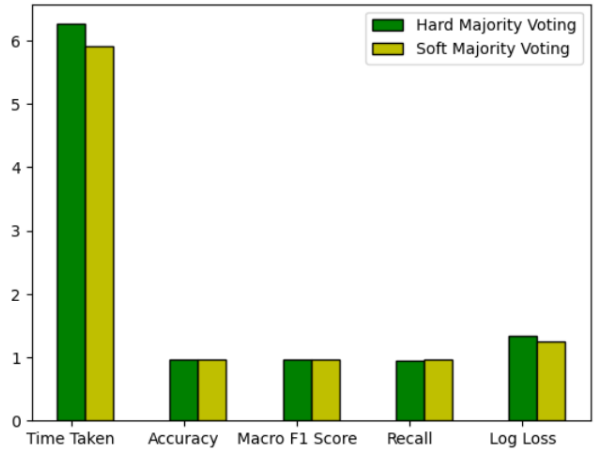

Figure 9. Hard vs Soft Majority

6. Improvements

- Further performance improvements could be explored by evaluating feature combinations, in addition to individual features.
- Feature selection techniques, such as chi-square or Pearson correlation, could be employed to identify and utilize the most relevant features, potentially enhancing model performance.
- Real-time classification could be implemented, similar to a website extension. However, this would necessitate optimizing feature extraction for reduced runtime.

7. References

- New DNS Threat Trend Report - Cisco Umbrella
- Web page phishing detection - Mendeley Data
- 2010.12847.pdf (arxiv.org)
- StandardScaler — scikit-learn
- GridSearchCV — scikit-learn
- Voting Classifier — GeeksforGeeks