

**DETECTING PHISHING URL USING MACHINE LEARNING MODELS**

MACHINE LEARNING FINAL PROJECT

COMP6577001 – MACHINE LEARNING

LE01 - LEC

2602160750 – Bryan Orville Audric

2602070206 – Michael Kurniawan

Even Semester 2023/2024

Table of Contents

1. Introduction

In recent years, the proliferation of online platforms has led to an increase in cyber threats, with phishing and malware attacks proving to be a significant threat to internet security. Phishing attacks can be defined as fraudulent activities that go beyond traditional data theft to encompass a range of malicious behaviors, including cyber terrorism, hacktivism, damaging reputations, espionage, and nation-state attack to trick individuals into divulging sensitive data, such as login credentials or bank information [1].

To address this, researchers have increasingly use machine learning models to automate phishing detection as machine learning models offer the capability of adapting and learning a specific pattern that can indicate phishing activity from vast amounts of data. This project focuses on the use of different machine learning models that will help detect phishing URL accurately.

1. Methodology
2. Project Process

There are several processes that need to be followed based on Fig. 1. The process starts with gathering a dataset that will then be extracted. The result would be a numerous feature that represents the data. The features are then filtered using Pearson correlation, which is able to produce a higher accuracy based on this [2]. The filtered features are then used for the machine learning models to train and predict. The machine learning models chosen for this project are Random Forest, XGBoost and LightBGM. These models are called an ensemble model and is known to be accurate in classifying classes based on this [3], in which the three classifier models are then evaluated and compared using the testing dataset.

A diagram of a data flow

Description automatically generated

Fig. 1. Project Process

1. Dataset

There are two datasets used for this research. The first dataset, taken from <https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset>, which consists of 651.191 URLs which are divided into 428.183 benign URLs, 96.457 defacement URLs, 94.111 phishing URLs, and 32520 malware URLs. Since the primary objective of this project is to detect phishing, other categories such as malware and defacement will be discarded. The second dataset, taken from <https://dataforseo.com/free-seo-stats/top-1000-websites>, which contains 1000 benign URLs.

1. Feature Extraction

Feature extractions are employed to extract several relevant characteristics or attributes that can represent URLs themselves to distinguish between benign or phishing. After the features are extracted, the dataset is resampled using under sampling method to balance out the phishing URLs and benign URLs. The list of features is shown in Table 1. Features 1 to 22 are extracted based on this [4] and the rest are taken based on this [2]

Table 1. List of Features

|  |  |  |
| --- | --- | --- |
| **No.** | **Feature** | **Description** |
| 1 | have\_ip\_address | IP Address in URL |
| 2 | url\_length | Number of Characters in URL |
| 3 | shortening\_service | URL shortening service used in URL |
| 4 | count\_attrate\_symbol | Count of at symbol (@) |
| 5 | count\_http\_url | HTTP used in URL |
| 6 | count\_https\_url | HTTPS used in URL |
| 7 | count\_dot\_url | Number of dots (.) in URL |
| 8 | count\_hyphen\_url | Number of hyphen (-) in URL |
| 9 | count\_underline\_url | Number of underline (\_) in URL |
| 10 | count\_question\_url | Number of question mark (?) in URL |
| 11 | count\_slash\_url | Number of path in URL ( / ) |
| 12 | count\_amp\_url | Number of ampersand symbol (&) in URL |
| 13 | count\_exclam\_url | Number of exclamation mark (!) in URL |
| 14 | count\_percent\_symbol | Number of percent symbol (%) in URL |
| 15 | count\_space | Number of whitespace ( ) in URL |
| 16 | count\_comma | Number of comma (,) in URL |
| 17 | count\_tilde | Number of tilde (~) in URL |
| 18 | count\_plus | Number of plus (+) in URL |
| 19 | count\_asterisk | Number of asterisk (\*) in URL |
| 20 | count\_hashtag | Number of hashtag (#) in URL |
| 21 | count\_dollar | Number of dollar symbol ($) in URL |
| 22 | count\_equal | Number of equal symbol (=) in URL |
| 23 | find\_email | Email used in URL or not |
| 24 | age\_of\_domain | The age of domain measure by subtracting current date by creation date |
| 25 | domain\_registration\_length | The registration length of domain measured by subtracting expiration date by creation date |
| 26 | SSL\_Certificate | Validity of SSL Certificate |

1. Feature Selection

Pearson’s Correlation is implemented during feature selection. Pearson's correlation coefficient, a statistical metric, is instrumental in the feature selection process. This coefficient quantifies the linear relationship between features and the target variable, discerning their relevance. By calculating correlation coefficients for each feature in the dataset with respect to the target variable, researchers can gauge their strength of association. Features exhibiting strong correlations, closer to 1 or -1, are considered more pertinent for distinguishing phishing from legitimate URLs. Conversely, features with coefficients near 0 indicate minimal linear relationship and are thus deemed less relevant for classification. The formula for calculating the Pearson correlation coefficient between two variables is given by:

Where rxy is the Pearson correlation coefficient between variables X and Y, Xi, and Yi are individual values of variable X and Y, and X̅ and Y̅ are the means of the values of X and Y, respectively. This formula measures the extent to which variables X and Y vary together, providing a deeper understanding of the relationship between features in the dataset and the target variable. Pearson correlation coefficient is broadly admitted in the feature selection algorithm to determine the best feature set. The features filtered using Pearson correlation are shown in Table 2.

Table 2. Feature Selected using Pearson’s Correlation

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| count\_http | 0.32100791187458777 |
| count\_https | 0.1310263357033035 |
| countdot | 0.3296993979703099 |
| count- | -0.2696988695694754 |
| count\_ | -0.11990573641394871 |
| counttilde | 0.10286000969916126 |
| domain\_age | -0.228017746506855 |
| regis\_length | -0.23760975867220246 |

1. Splitting Dataset into Training set and Testing set

The dataset is split into training and testing with a ratio of 70 to 30.

1. Modeling

*Random Forest*

Random Forest is a widely used ensemble learning model comprised of multiple decision trees. Each tree is constructed independently and randomly, with a subset of features selected randomly during the tree-building process. This feature selection technique helps mitigate overfitting and decorrelate the trees within the ensemble. Through bootstrap aggregating or bagging, Random Forest generates multiple bootstrap samples from the training data and builds a decision tree on each sample. During classification tasks, the model employs a majority voting mechanism, where each tree "votes" for the most prevalent class [18].

*XGBoost*

XGBoost is a scalable tree boosting system widely used in machine learning for its effectiveness and efficiency. It works by sequentially building an ensemble of decision trees, where each tree corrects the errors of the previous ones through boosting. By optimizing an objective function that combines a loss function and regularization term, XGBoost minimizes errors while preventing overfitting. Leveraging gradient boosting, XGBoost fits trees to the gradient of the loss function, learning from previous mistakes. With parallel and distributed computing capabilities, XGBoost can efficiently utilize multiple CPU cores and scale to large datasets. It also provides insights into feature importance and is designed for scalability, handling billions of examples with minimal resources [19].

*LightGBM Classifier*

LightGBM is a state-of-the-art Gradient Boosting Decision Tree (GBDT) algorithm that revolutionizes traditional implementations by introducing novel techniques like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS optimizes the training process by sampling data instances based on gradient values, focusing on instances with significant gradients to accurately estimate information gain while reducing computational overhead. EFB enhances efficiency by bundling related features together, improving cache hit rates and spatial locality. By leveraging these techniques, LightGBM accelerates model training, reduces memory consumption, and maintains high accuracy, making it a powerful tool for handling large feature dimensions and datasets in machine learning tasks [20].

1. Evaluation

*Accuracy*

Accuracy is one of the commonly used evaluation metrics in classification modeling to assess how well a model can classify data correctly overall. In this context, "correct" means the model's predictions match the true labels of the observed data. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions made by the model. The formula for calculating the accuracy of a classification model is as follows:

*Precision*

Precision is a crucial evaluation metric in classification tasks, particularly in scenarios where minimizing false positives is important. It measures the accuracy of positive predictions made by the model, i.e., the proportion of correctly predicted positive instances out of all instances predicted as positive. The formula for calculating precision in a binary classification setting is as follows:

*Recall Score*

Recall, also known as sensitivity or true positive rate, is an essential evaluation metric in classification tasks, especially when it's crucial to capture all positive instances. It measures the ability of the model to correctly identify positive instances from all actual positive instances in the dataset. The formula for calculating recall in a binary classification setting is as follows:

*F1-Score*

The F1 score is a commonly used evaluation metric in classification tasks, which combines precision and recall into a single metric. It provides a balance between precision and recall, making it particularly useful when both false positives and false negatives need to be minimized. The formula for calculating the F1 score is:

Where Precision and Recall are as previously defined. This formula computes the harmonic mean of precision and recall, emphasizing the balance between the two metrics. The F1 score ranges from 0 to 1, where a higher score indicates better performance. It is a useful metric for evaluating classification models, especially in scenarios where achieving a balance between precision and recall is important, such as in information retrieval systems or medical diagnosis.

*Confusion Matrix*

Confusion Matrix is a matrix or table that compares the actual label values with those predicted by the model. The basic structure for binary classification is as follows:

Table 3. Basic Structure of Confusion Matrix

|  |  |  |
| --- | --- | --- |
|  | **Predicted Positive** | **Predicted Negative** |
| **Actual Positive** | True Positive (TP) | False Negative (FN) |
| **Actual Negative** | False Positive (FP) | True Negative (TN) |

*ROC Curve*

The ROC (Receiver Operating Characteristic) curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots two parameters, which are:

*AUC Score*

The AUC (Area Under the Curve) represents the area under the ROC curve. It quantifies the overall ability of the classifier to discriminate between positive and negative classes. The value of AUC ranges from 0 to 1:

1. Result

After feature extraction and feature selection as well as implementing machine learning models such as Random Forest, XGBoost and LightGBM, the evaluation results are as follows:

Table 4. Classification Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Type** | **Precision** | **Recall** | **F1 Score** |
| Random Forest | 0.90 | Benign | 0.89 | 0.90 | 0.90 |
| Phishing | 0.90 | 0.89 | 0.90 |
| XGBoost | 0.90 | Benign | 0.89 | 0.91 | 0.90 |
| Phishing | 0.91 | 0.88 | 0.89 |
| LightGBM | 0.89 | Benign | 0.88 | 0.90 | 0.89 |
| Phishing | 0.90 | 0.88 | 0.89 |

Figure 1. Confusion Matrix of Random Forest Model

A yellow and purple squares with white text

Description automatically generated

Figure 2. Confusion Matrix of XGBoost

A yellow and purple squares with white text

Description automatically generated

Figure 3. Confusion Matrix of LightGBM

A yellow and purple squares with numbers

Description automatically generated

Figure 4. ROC AUC Score of Each Machine Learning Models

A graph with a line

Description automatically generated

1. Reference

[1] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, “Phishing Attacks: A Recent Comprehensive Study and a New Anatomy,” *Frontiers in Computer Science*, vol. 3. Frontiers Media S.A., Mar. 09, 2021. doi: 10.3389/fcomp.2021.563060.

[2] V. Aprelia Windarni, A. Ferdita Nugraha, S. Tri Atmaja Ramadhani, D. Anisa Istiqomah, F. Mahananing Puri, and A. Setiawan, “DETEKSI WEBSITE PHISHING MENGGUNAKAN TEKNIK FILTER PADA MODEL MACHINE LEARNING,” 2023.

[3] A. A. Ubing, S. Kamilia, B. Jasmi, A. Abdullah, N. Z. Jhanjhi, and M. Supramaniam, “Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning,” 2019. [Online]. Available: www.ijacsa.thesai.org

[4] M. W. Shaukat, R. Amin, M. M. A. Muslam, A. H. Alshehri, and J. Xie, “A Hybrid Approach for Alluring Ads Phishing Attack Detection Using Machine Learning,” *Sensors*, vol. 23, no. 19, Oct. 2023, doi: 10.3390/s23198070.