Detecting Phishing Websites

By

20AG1A6704 Ainakota Yaswanth

20AG1A6710 Bandapu Sai Chandu

20AG1A6717 Dachepally Kiran Kumar

20AG1A6757 Vinta Anshitha

Abstract

Data phishing has emerged as a prevalent and successful method employed by cybercriminals to deceive individuals and illicitly obtain their personal and financial information. In our increasingly internet-dependent world, where we conduct a significant portion of our daily activities online, fraudsters have found a fertile ground to execute targeted phishing attacks. These modern-day phishing attempts have become remarkably sophisticated, making them harder to detect. Astonishingly, a recent study conducted by Intel revealed that an overwhelming 97% of security experts failed to differentiate between genuine emails and phishing emails. This alarming statistic underscores the urgent need for a deeper understanding of data phishing and the implementation of robust countermeasures to safeguard ourselves from these insidious attacks.

Problem Statement

Develop a data science model to detect phishing websites accurately, aiming to protect users from online threats. The model should analyse various features of a website and classify it as either legitimate or phishing. Providing a reliable tool for online security and fraud detection.

Overview

- We used a variety of machine learning methods to solve the issue, including Logistic Regression, Decision Trees, Random Forests, SVM, LightGBM, and XGBoost. We had a dataset of URLs that had already been classified as phishing or real. where the attributes are based on HTML and JavaScript, as well as URL and domain-based bases. This allowed us to identify patterns that are typically present in phishing attempts.
- We put the models to the test by seeing how well they could distinguish between authentic and fraudulent websites. To determine how effective, they were at this task, we looked at their accuracy scores. The most effective model was the one with the highest accuracy score. This model will be used in real-time to quickly identify phishing websites and protect users from potential threats. Our system combines machine learning models with other techniques to produce a reliable and powerful defence against phishing attacks.

Procedure

➢ Preparation of Data:

- Import all the necessary libraries and load the dataset of various URLs into the Jupyter Notebook before beginning the analysis.
- Conducted a preliminary analysis of the data obtained, examining the kind and format of the various attributes.
- A dataset is created based on the categories of a URL's characteristics.
- From two key URL properties, we have taken into account the major attributes.
- Address-Based / URL-Based
- HTML & JavaScript Based
- The values 1 and 0 were assigned to the features. where 1 indicates maliciousness and 0 indicates purity.
- For analysis and model building, the newly created dataset is saved and imported into a Jupyter Notebook.
- There were no problems with the data after a thorough inspection of the whole data set.
- The datatypes of the columns were also examined, but no issues were identified.

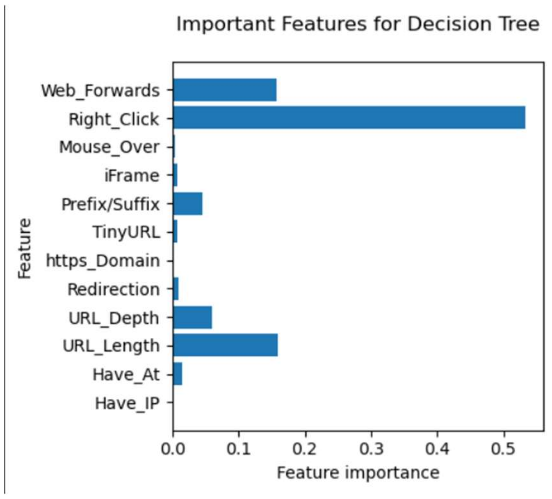
➢ Exploratory Data Analysis

- Data Collection: - The list of phishing URLs was collected using the open-source PhishTank website. This site offers a collection of phishing URLs that are updated hourly in a variety of formats, including csv, Json, etc. Visit https://www.phishtank.com/developer_info.php to download the data. The authentic URLs were extracted from the University of New Brunswick's open datasets at https://www.unb.ca/cic/datasets/url-2016.html. This dataset contains a collection of URLs that aren't malicious, spammy, phishing, or defacement. The harmless URL dataset is taken into consideration for this study out of all of these types. The dataset comprises of approximately 12,000 phishing URLs and 10,000 harmless URLs.

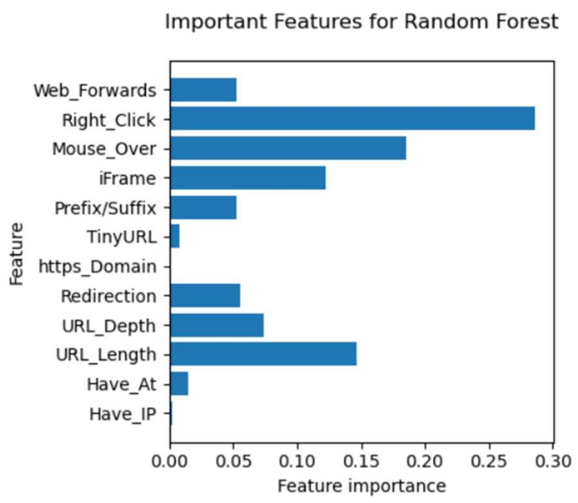
- Data Cleaning: - We first understand the types of attributes present in the dataset and next find if any null values exist in the dataset. The next step is to determine whether our data is balanced or not by looking at the "Label" column, which involves determining whether the dataset contains roughly equal numbers of phishing and safe URLs to prevent bias. We did univariate analysis to comprehend the distribution of attribute values.

Model selection

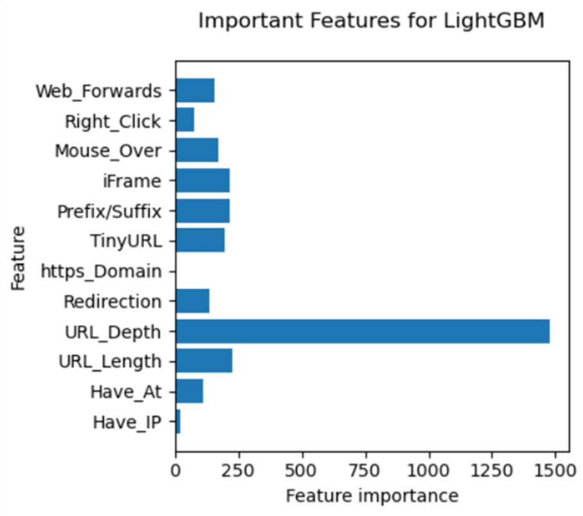
- For model building which could classify the given input into malicious or legitimate, we used sklearn libraries.
- We have split the data into two parts, one for training (70%), and other for testing (30%).
- We have used several machine learning algorithms to improve the accuracy.
- Models we used are:
    - **Decision Tree Algorithm:**
        - Models for classification and regression tasks frequently use decision trees. In essence, they pick up a hierarchy of if/else questions that lead to a choice. Finding the if/else set of questions that leads us to the correct solution the quickest is the key to understanding how to use a decision tree
        - Below are the important features the decision tree algorithm has considered.



- **Random Forest Algorithm:**
    - Random forest is a group of decision trees, where each tree differs somewhat from the others. The theory behind random forests is that while each tree may make somewhat accurate predictions, it will probably overfit on some portions of the data. They are quite effective, frequently operate well without much parameter adjusting, and don't call for data scaling.
    - Below are the important features the Random Forest algorithm has considered.



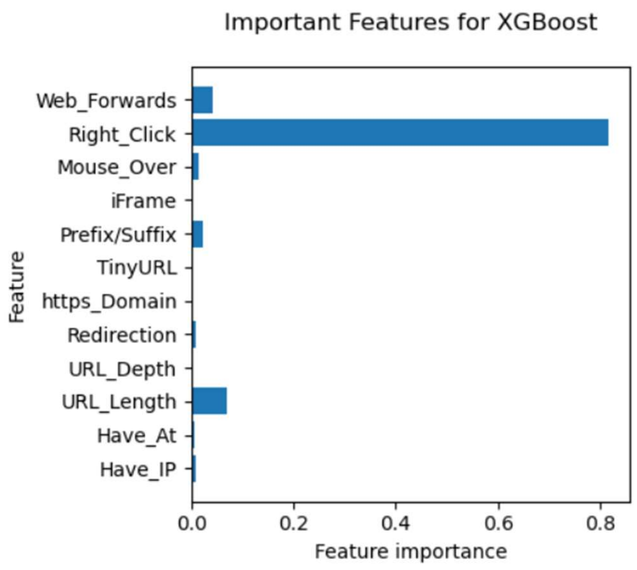
- **LightGBM:**
  - Python's LightGBM is a powerful gradient boosting method. It employs a leaf-wise tree growth strategy, which, in comparison to conventional gradient boosting techniques, provides quicker training times and greater accuracy. LightGBM was created with the explicit purpose of handling huge datasets well while maximizing memory consumption. To further improve its efficiency, it includes elements like parallel computation.
  - Below are the important features the LightGBM algorithm has considered.



- **Support Vector Machine (SVM):**
  - Support-vector machines (SVMs, also known as support-vector networks) are supervised learning models with corresponding learning algorithms that examine data used for regression analysis and classification.
  - An SVM training method creates a model that categorises fresh examples according to one of two categories given a series of

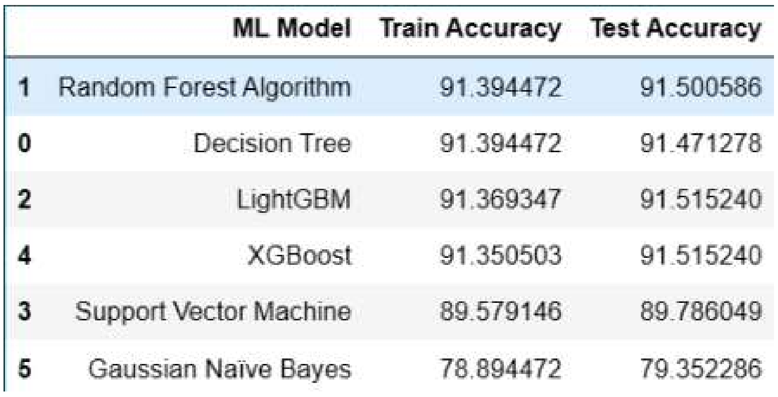
training examples that have each been tagged as belonging to one of the categories. This makes the algorithm a non-probabilistic binary linear classifier.

- **Extreme Gradient Boost Algorithm (XGBoost):**
  - XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.



- **Naïve Bayes:**
  - Naive Bayes is a simple and popular machine learning algorithm used for classification tasks. It assumes that features are independent and calculates the probability of a class given the feature values. It is easy to implement and performs well on large datasets. Despite its simplifying assumptions, Naive Bayes can be surprisingly accurate and is widely used for spam filtering, text classification, and other applications.

Results & Future Scope: -



	ML Model	Train Accuracy	Test Accuracy
1	Random Forest Algorithm	91.394472	91.500586
0	Decision Tree	91.394472	91.471278
2	LightGBM	91.369347	91.515240
4	XGBoost	91.350503	91.515240
3	Support Vector Machine	89.579146	89.786049
5	Gaussian Naïve Bayes	78.894472	79.352286

- The models are evaluated, and considered metric is accuracy.
- Although the accuracy score of Random Forest, Decision Tree, LightGBM and XGBoost Algorithms are similar, we have considered to use Random Forest Algorithm. As the Random Forest Algorithm considers and gives significant importance to the features present in the data.
- Rest of the algorithms consider one feature and giving more importance to that attribute compared to other attributes.
- By making the improved model into a browser extension, it can be used. which would prompt the user to inform about the website's malicious nature.
- We can use the model for user awareness by introducing the deployed project into the text service the user uses because there are no prompts or indicators of spam or phishing websites when shared in text messages.

References: -

- https://github.com/shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques
- https://jpinfotech.org/detection-of-phishing-websites-using-machine-learning
- https://www.ieeexpert.com/python-projects/phishing-website-detection-using-machine-learning