Malicious URL Classification

Jianfeng Liu V00969381 ***University of Victoria***
jianfengliu@uvic.ca

Abstract

I used different machine learning methods to classify over tens of thousands of URLs as malicious or non-malicious. The performance was compared to see which method worked best for this task. Data processing was first performed. For feature selection, I used feature extraction based on URL structural components. Then the prepared data were applied to decision tree, random forest and KNN algorithm models respectively. Finally, by comparison, the KNN algorithm ran the longest time, but in terms of accuracy, the KNN algorithm gave the best results, while the random forest and decision tree performed close, but with less satisfactory results.

1 Introduction

1.1 Malicious URL Classification

Phishing sites are a common method of social engineering; they are fronts for the distribution of viruses, Trojans and various malicious codes that can infect users and lead to data theft and money laundering. These URLs are usually distributed through email links, pop-up ads and embedded downloads. It mimics trusted Uniform Resource Locators (URLs) and web pages. The goal of this project is to train machine learning models on the provided dataset, apply multiple machine learning algorithms, compare the results, and finally select the model with the best results to predict phishing websites.

1.2 Problem definition and Input data

The main objective of this project is to classify URL into a spam (malicious) and normal. In order to achieve this, I have to extract the useful features from dataset so that machine learning algorithms can perform better. Several algorithms would be explored to find a relative optimal one to achieve our final goal.

The data I am going to use is provided by professor The data is stored in 2 txt files, Only urls and lable are included. There are more than 3 million labeled samples. Data only includes a low percentage of missing or null values. Hence, it could be used to apply ML algorithms.

2 Methodology

2.1 Data Cleaning

To prepare the text data for the model building, we have to do data cleaning to match the requirement of models. The source data is 2 txt documents containing url list, I read out the list from the file as dataframe and remove the irregular content. Then additionally added the list column. Finally, the two dataframes are merged to form one dataframe, and the data is distributed as shown in the figure1(0 stands for normal urls,1 stands for malicious urls).

Preprint. Under review.

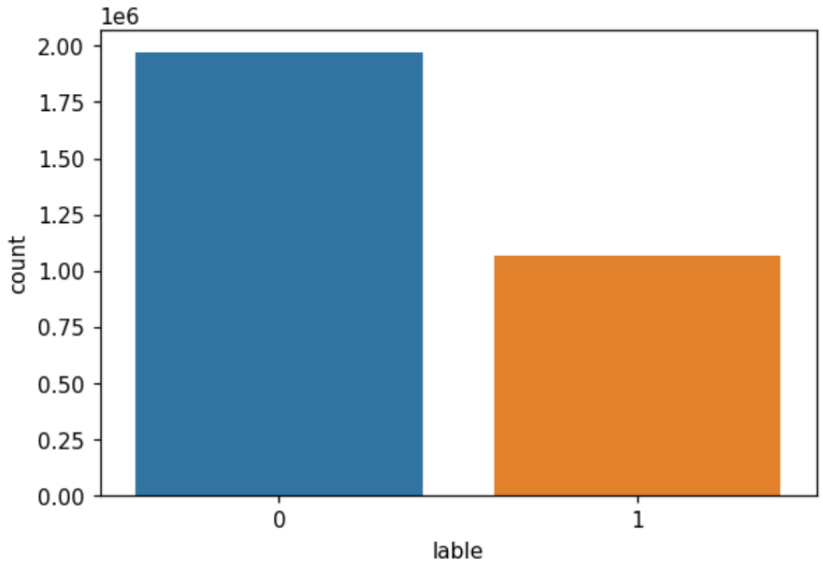

Figure 1: Source data

2.2 Features Extraction

Features are the most critical element in machine learning, and the selection of features is often directly related to the good or bad results. In malicious URL detection, features are mainly divided into two categories: static features and dynamic features. Static features refer to the acquisition of feature representations such as IP addresses, domain lengths, geographic information, path lengths, etc. in the case of "URL requests"; dynamic features refer to the behavior of the system "during the execution of JS/ActiveX code" (logging). Static features are relatively safer because they do not require the execution of malicious code in advance.

The structure of the URL is shown in the figure2. This project uses the **urllib** library to segment the obtained URLs and then performs feature extraction based on the segmented part.

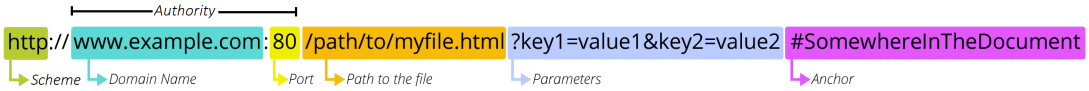

Figure 2: URL Components

In this project, domain-related features are extracted: domain depth (i.e., based on the number of dots), number of desh symbols in the domain, length of domain characters, whether to use short domain names, etc. In addition, path depth, whether to include IP addresses, whether to include @ symbols, url length, and whether to include the following special words(confirm, banking, secure, ebayisapi, webscr, eBay, PayPal, sulake, facebook, orkut, santander, mastercard, warcraft, visa, bradesco)(**?**). Lastly, we add lable feature, and there are 10 features in total which is showed in figure3.

	domaindept	domaindesh	pathDepth	domainlen	containIP	containAt	geturlLen	containWords	tinyURL	Label
0	2	0	3	1	0	0	0	0	0	0
1	2	16	3	1	0	0	1	0	0	0
2	2	1	6	1	0	0	0	0	0	0
3	1	2	0	1	0	0	0	0	1	0
4	1	4	2	1	0	0	0	0	0	0

Figure 3: Structure Based Features

2.3 Models and Performance

Figure 4 shows the general path that data will follow. Different classifiers will be test to be able to compare performance of models and conclude which approach is better.

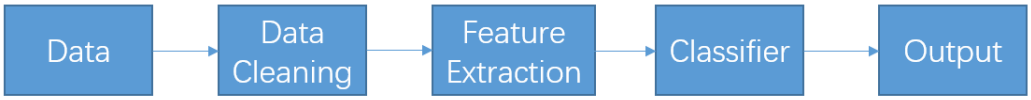

Figure 4: Pipeline for data

Also, after the data cleaning and feature extraction, it was necessary to split the data set into training and testing with 70% and 30% respectively.Therefore, classifiers could be evaluated and compared between them with metrics.

2.3.1 Decision Tree

Decision Tree is one of the most widely used machine learning algorithm for solving the supervised learning problems. It is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.For this classification, we used CART decision tree and set the max tree depth as 5. The image of ROC curve is shown below.

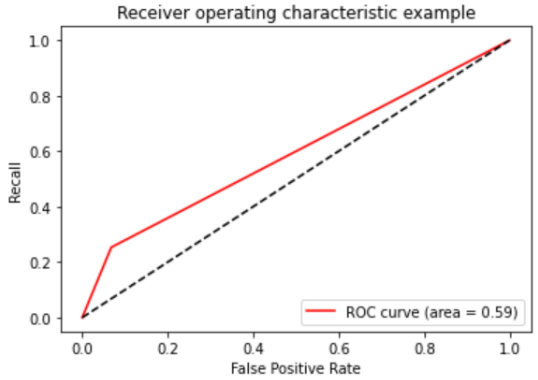

Figure 5: ROC curve of Decision Tree

2.3.2 Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.The image of ROC curve is shown below.

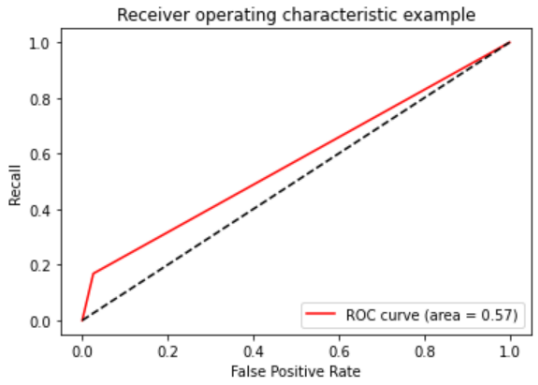

Figure 6: ROC curve of Random Forest

2.3.3 KNN

Based on the assumption "similar urls will have similar features" I decide to try nearest neighbors classifiers. For this classifier, I set number of neighbors as 3 which get the highest score. Shown as below.

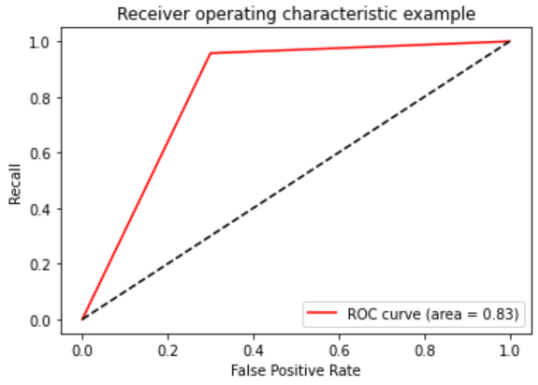

Figure 7: ROC curve of KNN

3 Results Comparison

Table 1 presents a summary of the best combination of hyper-parameters for each model after the cross validation. Also it presents the evaluation metrics of precision, recall and F1 for each model. Confusion matrices can be found in Appendix.

Table 1: Results Comparison

function	Training Score	Testing Score
Decision Tree	0.706	0.705
Random Forest	0.705	0.705
KNN	0.836	0.857

4 Discussion & Conclusions

By comparing the results from different models, we can see that the models perform not good enough. To find the reason, I found the distribution of all features on the data, as shown in the following two graphs, we can notice that the features I extracted based on the structural components are not particularly different between malicious and non-malicious URLs, which means that the data of the two types of lists are similar in terms of the performance of the selected features, thus leading to unsatisfactory results. And if we want to achieve better results, we must add features with more differences, such as dynamic URL features (e.g. domain registration To achieve better results, it is necessary to add more differentiated features, such as dynamic URL features

(e.g., domain name registration time, URL responce content, etc.), which is also an area for improvement in this project. I know there is another way to gather features, like phishing email/text classification, gaining tokens(words) from the URLs and set the vocabulary list as features Perhaps the ideal URL classifier might be one that fuses features extracted in two different ways to form a unified feature that can then be applied to different algorithmic models.

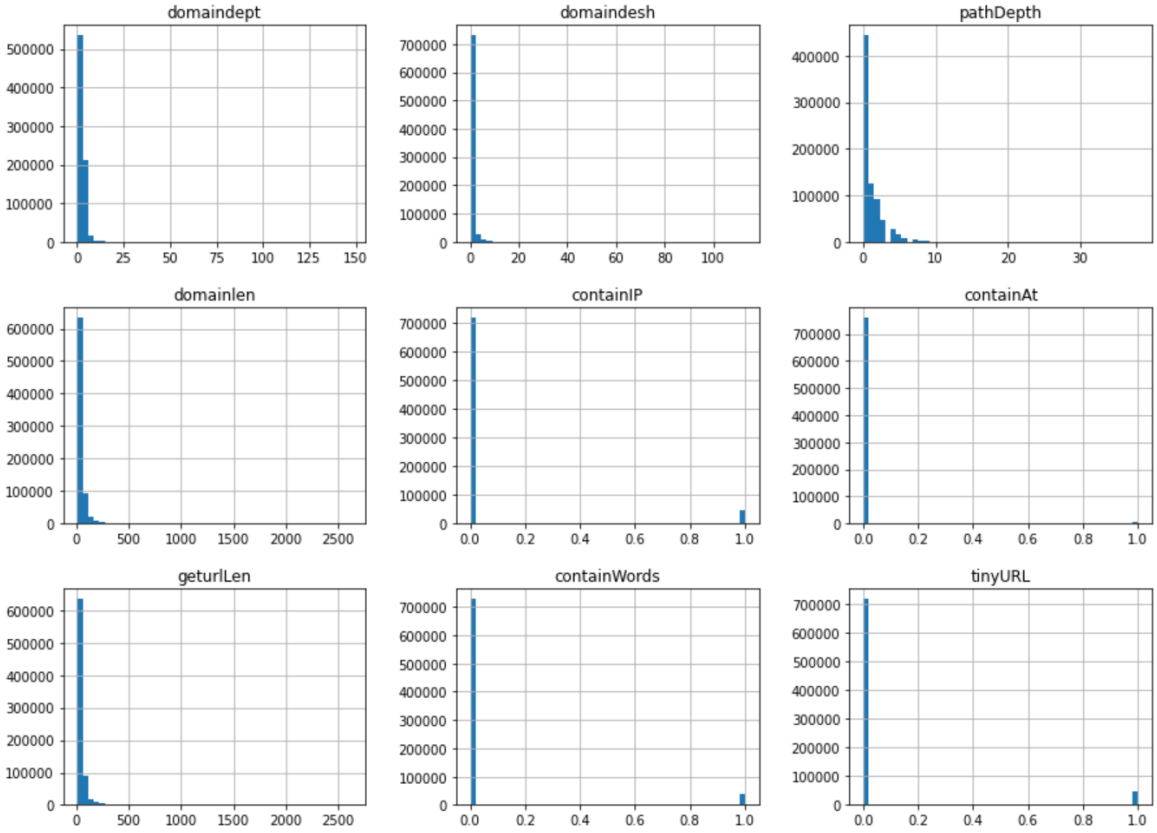

Figure 8: Feature Distribution of Black URLs

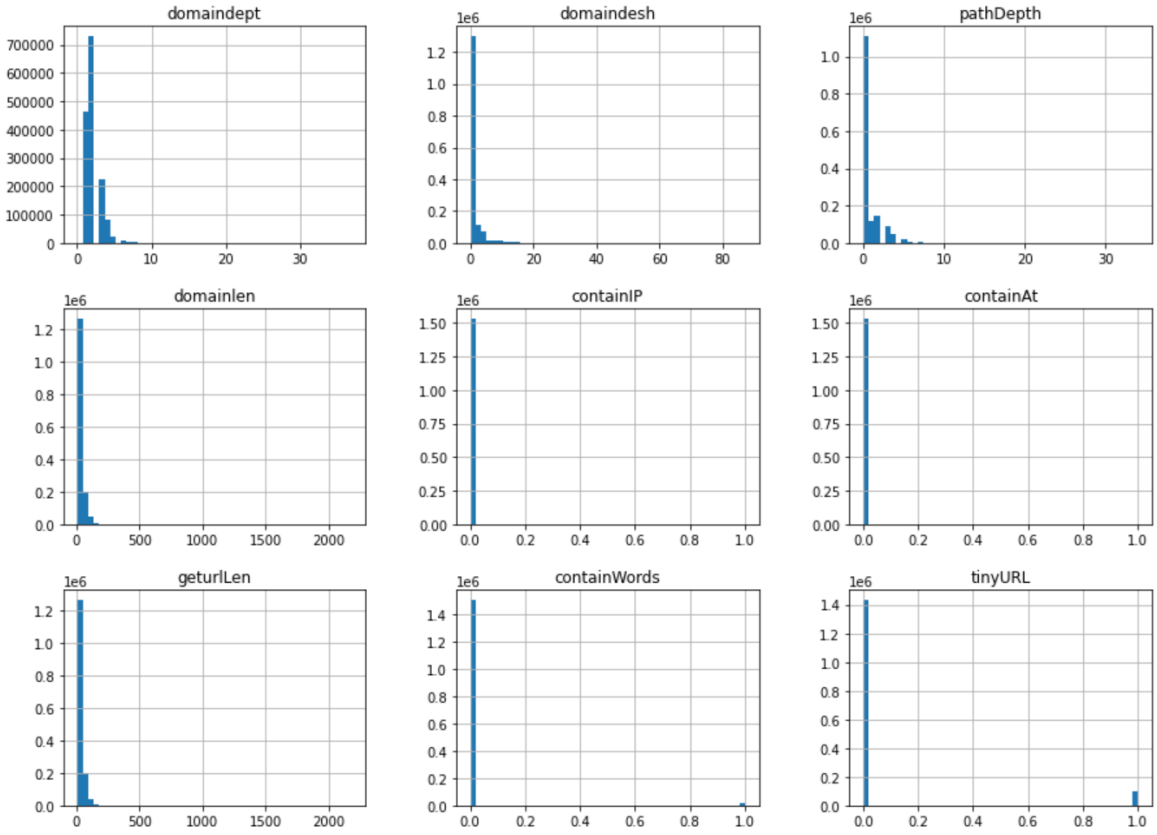

Figure 9: Feature Distribution of White URLs

References

[1] "Phishing URL Detection with ML," Feb 8, 2018. [Online]. Available https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5

[2] "Phishing Website Detection by Machine Learning Techniques,"May 11, 2020. [Online]. Available https://github.com/shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques

[3] "A SVM-based Technique to Detect Phishing URLs," May 29, 2012. Accessed On: Dec 9,2021 [Online] Available https://scialert.net/fulltext/?doi=itj.2012.921.925

[4] "Malicious URL Detection by Dynamically Mining Patterns without Pre-defined Elements," . Accessed On: Dec 1,2021 [Online] Available https://www.cs.sfu.ca/ jpei/publications/URLPatternMiningWWWJ.pdf

[5] "What's in a URL? Genre Classification from URLs," . Accessed On: Dec 1,2021 [Online] Available https://apps.dtic.mil/sti/pdfs/ADA599843.pdf

[6] "What is a URL?," . Accessed On: Dec 10, 2021 [Online]. Available https://developer.mozilla.org/en-US/docs/Learn/Common$_{q}uestions/What_{i}s_{a}URL$