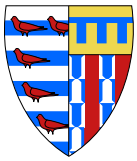
UNIVERSITY OF
CAMBRIDGE

Explain the ML models for better cyber phishing detection

Chuqiao Feng



The Chinese University of Hong Kong, shenzhen

This thesis is submitted as part of the Pembroke College International Summer School

Abstract

This project is to investigate a typical cyber incident response by some novel machine learning ideas. With the consequences of phishing scams becoming more complex and dangerous year by year, phishing website prediction has always been an important topic. However, direct phishing monitoring requires a lot of labor costs, while using machine learning can reduce a lot of labor use while ensuring high accuracy. This paper explores AI-based machine learning and deep learning phishing site monitoring methods for identifying potential phishing sites. In addition, in this study, I explored the practical significance of this model in phishing detection by further studying the interpretability of machine learning.

Keywords: Cyber incident, machine learning, phishing, natural language processing, explainable AI, hybrid method, classifier, SHAP, logistic regression, XGBoost, random forest, PCA, BERT, RoBERTa

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. This dissertation does not exceed the prescribed limit of 8 000 words.

Chuqiao Feng
July, 2023

Contents

Chapter 1

Introduction

With the consequences of phishing scams becoming more complex and dangerous year by year, phishing website prediction has always been an important topic. However, direct phishing monitoring requires a lot of labor costs, while using machine learning can reduce a lot of labor use while ensuring high accuracy. This paper explores AI-based machine learning and deep learning phishing site monitoring methods for identifying potential phishing sites. In addition, in this study, I explored the practical significance of this model in phishing detection by further studying the interpretability of machine learning.

Chapter 2

Review for cyber phishing incident

2.1 Why phishing

Phishing is a prevalent form of cybercrime in which the attacker dupes the victim into entering ostensibly regular but in reality, harmful websites. This deception typically involves sending emails, text messages, or advertisements that purport to originate from a trusted source, aiming to coax the recipient into revealing their personal or financial information. This scheme exposes the victim's private data and financial assets to potential compromise. Phishing often capitalizes on human psychological tendencies. It grabs the victim's attention with a warning or alert messages designed to create a sense of urgency, prompting swift action. In the rush, victims often overlook potential repercussions. In today's digital age, the Internet has become an integral part of our daily lives. It serves a multitude of purposes, from knowledge sharing and social communication to facilitating various financial activities like purchasing, selling, and money transfers. Malicious websites present a grave risk to Internet users, and any less vigilant user can easily become a victim of malicious URLs.

Phishing capitalizes on the inherent trust people place in the Internet—a tool that has, in many ways, become an extension of our personal and professional lives. Surprisingly, many individuals struggle to comprehend that they can be duped into disclosing their private information online. This naivety paves the way for well-intentioned people to become easy prey to phishing attacks. The efficacy of phishing lies in its precision and adaptability. The perpetrators can pinpoint specific individuals or demographics, thereby increasing their chances of success. Moreover, they employ an array of methods to deceive their victims into surrendering crucial information. One such approach involves sending an email that mimics correspondence from a reputable company, such as Google, prompting the recipient to log in to their account on a counterfeit website. This ruse, if successful, leads to the theft of usernames and passwords. A further aspect that amplifies the effectiveness of phishing as cybercrime is the lack of concrete legislation against it.

Currently, phishing attacks are categorized as online harassment or fraud, offering victims little to no legal recourse when their personal information is stolen through such scams. This lack of deterrent allows phishing to thrive relatively unchallenged. Lastly, the ease of execution adds to the appeal of phishing from the perpetrator's perspective. With nothing more than a computer and rudimentary knowledge of its use, anyone can launch a phishing attack. This low entry barrier makes phishing not only a widespread threat but also a cost-effective tactic for those with malicious intent. In an increasingly interconnected world, the dangers of phishing become more pronounced. It underlines the need for comprehensive cybersecurity measures and promotes a culture of online vigilance.

2.2 Why Machine Learning

Due to ubiquitous network access, big data, the IoT net, global digitization, and the widespread use of technologies such as ubiquitous social networking sites and mobile applications, huge institutional and personal security issues have arisen. Cybersecurity for businesses and individuals is always a growing concern, but traditional security systems often fall short. AI has the intelligence and adaptability to deal with the changing cybersecurity landscape. Controlling who has access to what, authenticating users, analyzing their activity, and detecting spam, malware, and botnets - all of these tasks greatly benefit from the application of AI.

The main challenge in detecting phishing attacks is discovering the techniques used. Phishers are constantly enhancing their tactics and can create web pages that protect themselves from many forms of detection. Therefore, the development of robust, effective, and up-to-date phishing detection methods is necessary to counter the adaptive techniques employed by phishers. As AI evolves and improves, it becomes an increasingly important part of the cybersecurity industry. Surveying the literature on phishing detection techniques, we can find a number of monitoring methods, each of which has its own characteristics and limitations.

In the past, the most popular method was to detect phishing pages based on the blacklist, but the disadvantage of this method is that it cannot identify non-blacklist or temporary phishing pages. As a result, more powerful and effective methods of detecting phishing attacks have been developed. First, the blacklist method maintains a list of suspicious or malicious URLs that have been collected using different methods such as Google Safe Browsing, phishing tanks, and user voting. Therefore, when a web page is launched, the browser searches for the page in the blacklist and alerts the user when the page is found. Finally, the blacklist can be memorized on the user's machine or on a server. Blacklists are often used to judge websites as malicious or legitimate. However, while these techniques have a lower false positive rate, they lack the ability to classify newly

generated malicious URLs.

In another approach, people analyze the web content, such as taking some words, such as brand names from a URL or HTML content and weighting them, extracting logo images and comparing them to the original image, or looking for consistency between URLs and web content. Content-based approach to in-depth analysis of page content. Build classifiers from page content and third-party services such as search engines and DNS servers. However, these methods are sometimes ineffective due to the high dimension of page features and their reliance on third-party servers that violate users' privacy by exposing their browsing history.[4]

2.3 Methodology

For the method of the detection process used in this paper, I use extracted salient features from the phishing page structure, such as features of URLs, domain names, page levels, and so on. Then I applied these features to machine learning algorithms to build accurate classifiers that can effectively distinguish phishing from legitimate web pages. Before starting, I used factor analysis and principle component analysis for the first step of the investigation of the dataset and feature. After that, I used logistic regression and random forest as ML classification methods. However, due to the large feature quantity, which means the high dimension of the data, I also used a binary classification hierarchical perceptron neural network method and used the interpretable library to analyze the feature. Moreover, the explainable structure I used in this paper was SHAP.

In practical applications, we usually have not extracted good web features, in order to make the model of the paper have a better universality, I set up another model based on the fuzzy rules method. Based on natural language processing, it is combined with various artificial intelligence algorithms to upgrade its functionality, such as classification algorithms and neural networks. By implementing and adopting a specific set of tokenization rules, such a model can not only deal with ambiguous variables but also enhance the degree of contribution to the relationship between obscure features. In addition, the approach incorporates interpretability architecture to elucidate these variables and their relationships, while further demonstrating the impact of certain phishing url features on the prediction results. I use the BERT model as the basic structure here because BERT mainly uses the encoder part of the Transformer. Also, its bidirectional nature means that it can consider both the preceding and the following text of a word, which is good for situations where the entire URL needs to be fully analyzed.

Chapter 3

Dataset and data preprocessing

3.1 Dataset Information

In this paper, two datasets were used for the whole experiment. In my machine learning method for feature classification work, I used an existing website feature dataset, each data from the first dataset is composed of an index number, a label(phishing or not), and 48 features related to the website URL and its domain. And the data from the second dataset only include text-form URLs with their labels (phishing or not phishing), which were used for the natural language processing work. The information on the two sets' division for machine learning and dataset samples is shown below:[5][3]

Table 3.1: Datasets number info

	Feature Set	URL Set
Training	8000	8000
Validation	2000	2000

Table 3.2: Feature set samples

id	NumDots	PathLevel	...	Label
1	3	5	...	1
2	3	3	...	1
3	3	2	...	1

Table 3.3: URL set samples

URL	Type
br-icloud.com.br	phishing
mp3raid.com/music/krizz.html	benign
bopsecrets.org/rexroth/cr/1.htm	benign

3.2 Data preprocessing

Firstly, in the feature dataset, I used standard normalization to transform the range of numbers into (0, 1), for a more accurate classification. Besides, I find that the feature called 'HttpsInHostname' are all 0, the reason might be that if "https" is in the hostname of the URL, it might be a phishing attempt trying to trick users into thinking the site

is secure. This shows that this feature only interferes with the construction of models based on this dataset, so I deleted this column before the experiment. As for the URL dataset, in most text analysis tasks, we will operate on those parts with special symbols or typo elements, but since these are often very important criteria in phishing URLs, here I did not do anything other than tokenization on the text in this dataset. And the specific tokenization will be mentioned later.

3.3 Statistics

Firstly for the URL dataset, we can use the word cloud to show how often words appear in the data set, so I divided the dataset into phishing parts and non-phishing parts, then drew the word cloud for each subset, and get the following figures. In these figures, we can see that positive URLs usually consist of words like "html", "wiki", "article", "org", "blog", and "gov", while for opposite subset words were mostly "index", "php", "option", "com-content" and so on.

Figure 3.1: Legitimate wordcloud

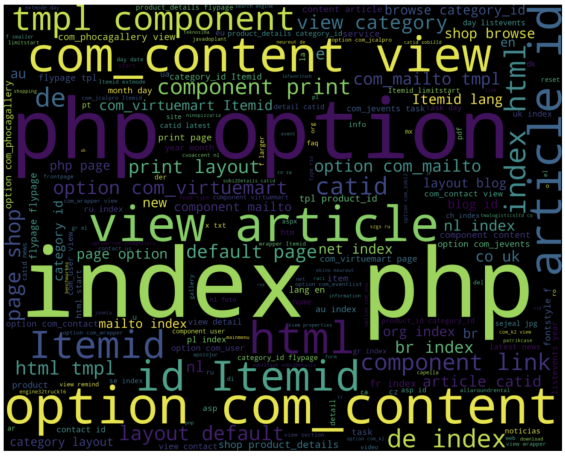


Figure 3.2: Phishing wordcloud

As for the feature dataset, I calculated the correlation between the two, from this, we can roughly interpret whether the correlation between features will affect the overall classification result. By calculating the correlation coefficient by the formula below, we can get the correlation relationship as shown in the figure below. It can be seen that there are many feature groups with correlation coefficients close to 1 among the relationships between different features. This also shows that in the process of using machine learning for classification, it is necessary to take into account the impact of the correlation between features on the overall result and use operations such as weighting and fuzziness to meet the accuracy and rationality of the model.

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Similarly, in order to explore whether there is a major feature that has a large influence

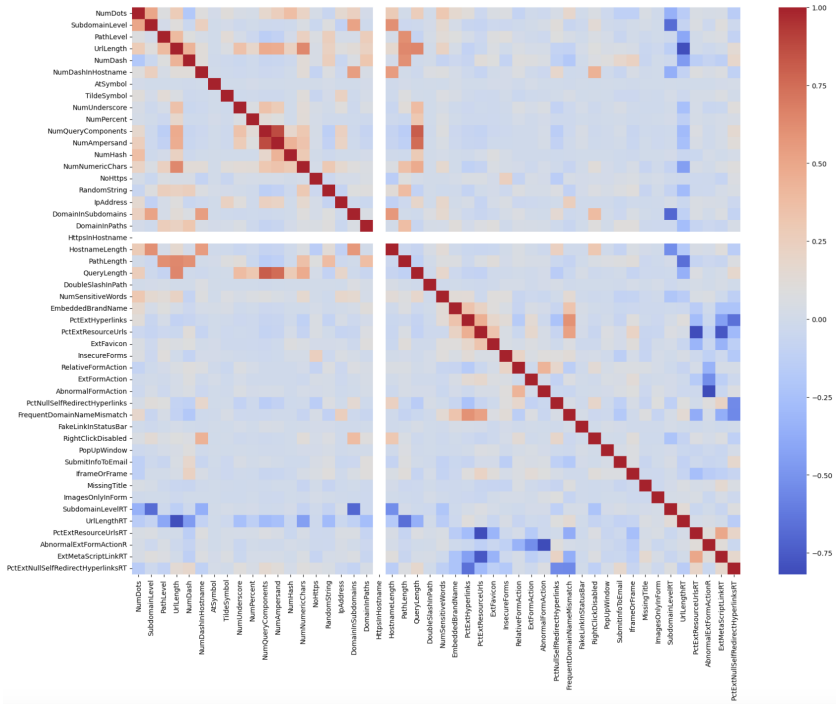


Figure 3.3: Correlation Map

on the results, I use principal component analysis and factor analysis here to explore the impact of the relationship between features and features on the overall results. This idea came from the high dimensions of the data set. PCA algorithm can increase the sampling density of samples by eliminating part of the information, which can alleviate the dimensional disaster to a certain extent. When the data is affected by noise, the eigenvectors corresponding to the minimum eigenvalues are often related to noise, so discarding them can reduce noise to a certain extent. However, PCA retains primary information, but this primary information is only specific to the training set, and this primary information is not necessarily important information. It is possible to discard some seemingly useless information, but this seemingly useless information happens to be important information, but it does not have a great representation on the training set, so PCA may also exacerbate overfitting.

Assuming the feature matrix as X, zero average the feature matrix and calculate the $C = \frac{1}{m}XX^T$, then calculate the eigenvalues of covariance matrix C and corresponding eigenvectors, arrange the eigenvectors into a matrix from top to bottom according to the corresponding eigenvalues, and take the first k rows to form a matrix P, $Y = PX$ is, the data after dimensionality reduction to k dimension. Here we take k to be 2 and 3 separately and visualize their results into 2D and 3D plots. It is easy to see from the figure below that due to the always large overlap area, it is difficult to find the greater impact of features on the results by separating two or three principal components.

Figure 3.4: 2c-PCA map

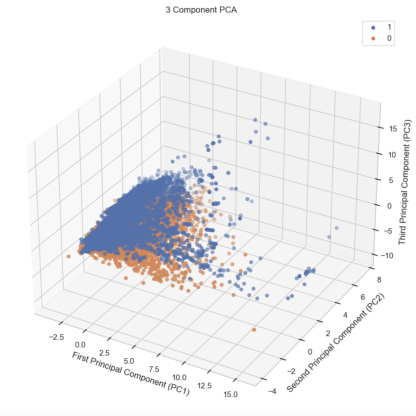


Figure 3.5: 3c-PCA map

Chapter 4

Machine learning method

4.1 Logistic Regression

First of all, as the most common classification method, I used logistic regression to establish the first model here. Logistic Regression, a statistical learning method, is applicable to binary classification tasks. Logistic Regression starts with the linear combination of input features and their corresponding weights to calculate the linear predictor $z = w^T x$. Then the Sigmoid function $\sigma(z) = \frac{1}{1+exp(-z)}$ maps the linear predictor to a probability value between 0 and 1. To train the regression model, the most commonly used loss function is the log loss or cross-entropy loss. Logistic Regression is a widely used algorithm for binary classification tasks because of its simplicity, efficiency, and ability to provide probabilistic outputs.

After the results of coefficient assignment in the code experiment, the statistical graph of coefficients as shown in the following figure was obtained. From the formula and principle of logistic regression, we can know that the greater the absolute value of the coefficient, the greater the contribution of the feature to the result, as can be seen from this figure. FrequentDomianNameMismatch ", "InsecureForm", and "NumDashInHostName" characteristics, and the results show a larger positive correlation, that several characteristics for the classification results and great influence, so the model has bigger weight for these a few characteristic parameters. Similarly, "SubmitInfoToEmail", "IframeOrFrame", "SubdomainLevel" and other factors have a large negative correlation with the results, indicating that they have the same but reverse effect on the results.

4.2 Random Forest

In addition, in the random forest model established for this data set. Random Forest is a popular ensemble learning method used for classification tasks. It combines multiple individual decision trees to make more accurate and robust predictions. The Decision

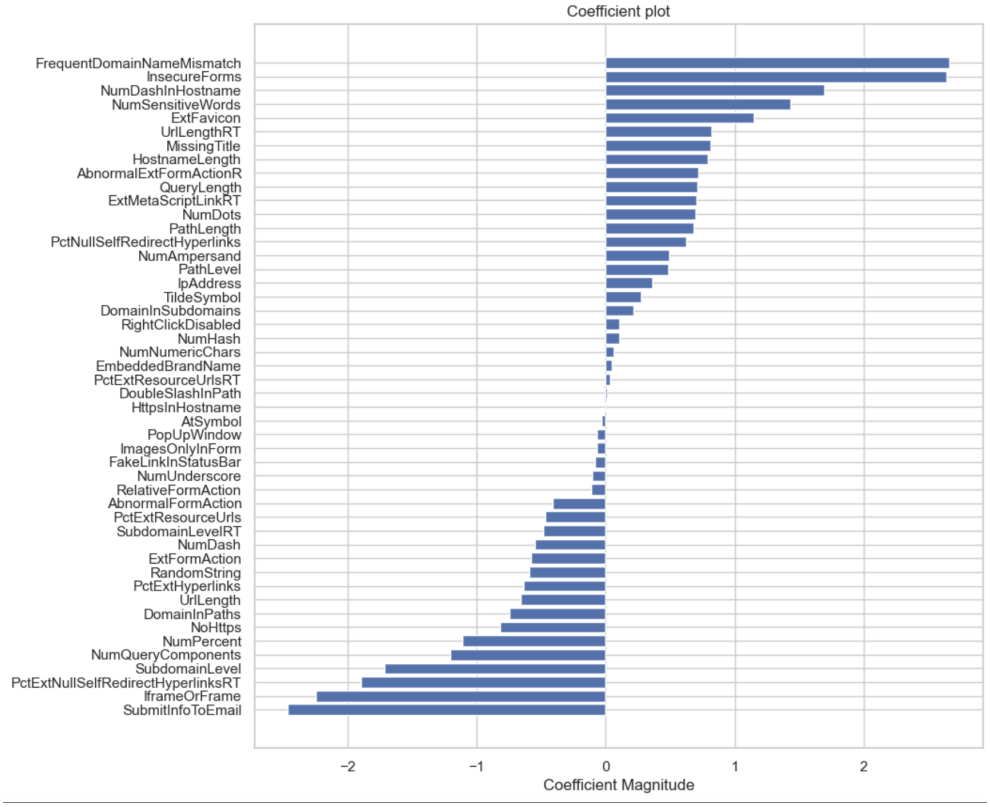


Figure 4.1: Logistic Regression - Coefficient Magnitude

tree is a flowchart-like structure where each internal node represents a test on an input feature, each branch represents the outcome of the test, and each leaf node represents a class label. Also similarly, I tried the XGBoost method, because XGBoost's strength lies in its ability to handle complex data, high-dimensional feature spaces, and large datasets. XGBoost introduces a unique regularized objective function that includes two parts - the loss function and the regularization term. The loss function quantifies the difference between the predicted and actual values, while the regularization term helps control the complexity of the model to avoid overfitting. XGBoost uses an approximate algorithm for finding the best split for each feature. It utilizes a "histogram-based" approach to quantize continuous features, which speeds up the training process significantly. To further speed up the feature splitting process, XGBoost uses a weighted quantile sketch to find the optimal split points efficiently. Besides, XGBoost implements L1 (Lasso) and L2 (Ridge) regularization terms in the objective function to control the complexity of the model and prevent overfitting.

$$\text{Random Forest Prediction} : y_{pred} = Mode/Mean(y_1, y_2, ..., y_n)$$

Then we drew the barplot of feature importance and found that the features with greater importance were very different from the model obtained by logistic regression. Then XGB

method model is used to draw the feature weight graph, which can also be found to be different from logistic regression. However, due to the similarity in principle, we find that the feature importance obtained by XGB is similar to the result of the random forest model. The reason is probably due to the fact that logistic regression cannot reflect the influence of elements themselves, so it cannot express the correlation between recessive factors and factors, and because of the high data dimension, it may cause the problem of overfitting.

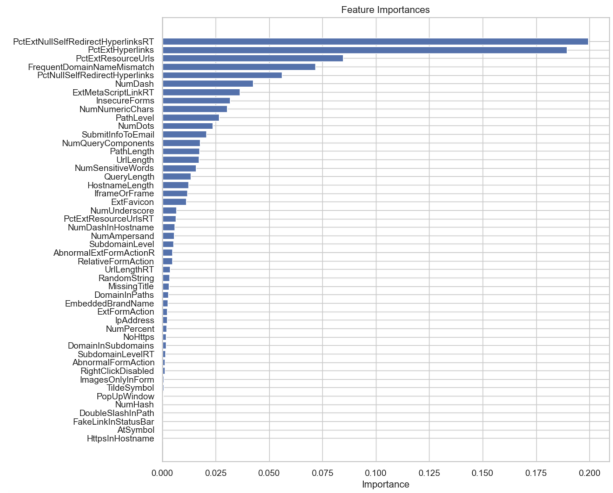


Figure 4.2: Random Forest

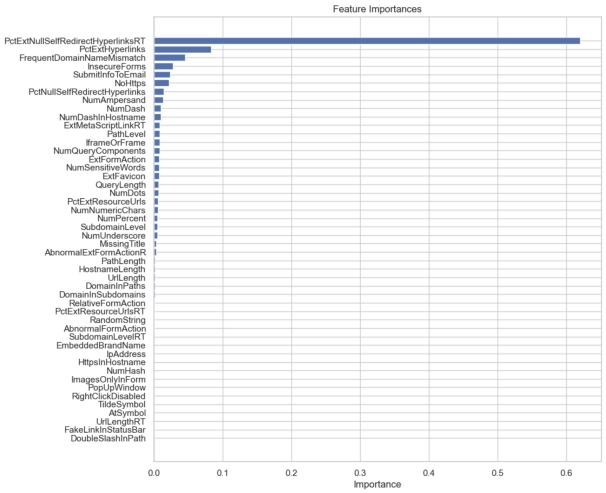


Figure 4.3: XGBoosting

4.3 MLP and Shap value

So, I brought out the neural network method to try. Due to the characteristics of this dataset, I used a binary classification multilayer perceptron as the infrastructure for classification. Binary classification multilayer perceptron is a supervised machine learning algorithm that learns the function $f : R^m \rightarrow R^o$ by training on a data set, where m is the dimension of the input and o is the dimension of the output. Given a set of features X and a label y, it can learn nonlinear functions for classification or regression, and unlike logistic regression, it can have one or more non-linear layers, called hidden layers, between the input and output layers. Its general structure is as follows: By collecting good data, we can train a machine-learning model. In future monitoring, the input URL is fed into a trained machine learning model and then outputs a category that can predict whether a website is a phishing address, thereby improving the network environment.

It is known to us all that a neural network is a kind of black box model because we can only know the network structure, transmission mode, model parameters, and other information, the specific internal operation mode is unknown. The utility of pure black box models for practical applications is very limited, so we need to introduce new ways to

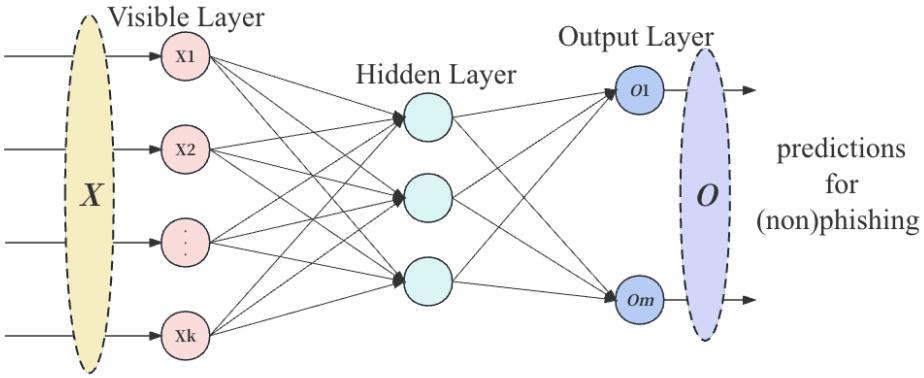


Figure 4.4: MLP structure

make the information in black box models available. Common interpretation models are Local interpretable model-agnostic explanations (LIME) and Shapley Additive exPlanation (SHAP). LIME approximately interprets the predictions of the black box model by building a local proxy model. Instead of just building the proxy model based on the local data of the model, some data perturbations are generated, and an interpretable white-box model is built based on the data of the original perturbation and the predicted value of the black box model, such as Lasso, decision tree, etc. But this article uses the SHAP model because LIME's biggest shortcoming when interpreting text models is the instability of the results. Because different local sampling brings different local perturbations, the final interpretation results will fluctuate. Moreover, LIME's use of a linear model as a proxy model means that it assumes a linear relationship between the predicted results and the inputs, which is too strong a prerequisite.

SHAP is a universal model interpretability framework that retrains models on feature subsets when calculating feature contributions. For feature i, the average of the feature i's contribution is calculated by first generating all feature sets containing i and removing i, and then retraining and calculating the predictions:

$$\text{shapley value:}\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_s(x_s)]$$

$$\text{explainer:}g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j$$

Here, g is the interpretation model, z' is the joint vector, M is the maximum length of the vector, phi is the contribution of feature j (Shapley values), and the joint vector represents which feature combinations the selected data point has, with 0 representing no feature and 1 representing the included feature. Mathematically it can be shown that only a single interpretation model g makes the interpretation result satisfy the satisfaction

accuracy, missing 0 and additivity. After using the code to implement the program calculation, we can get the following "summary plot" from the Python SHAP library.

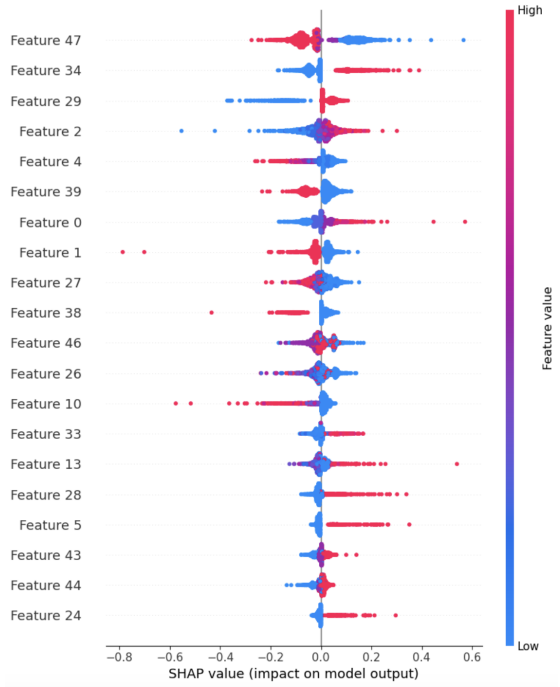


Figure 4.5: Summary plot

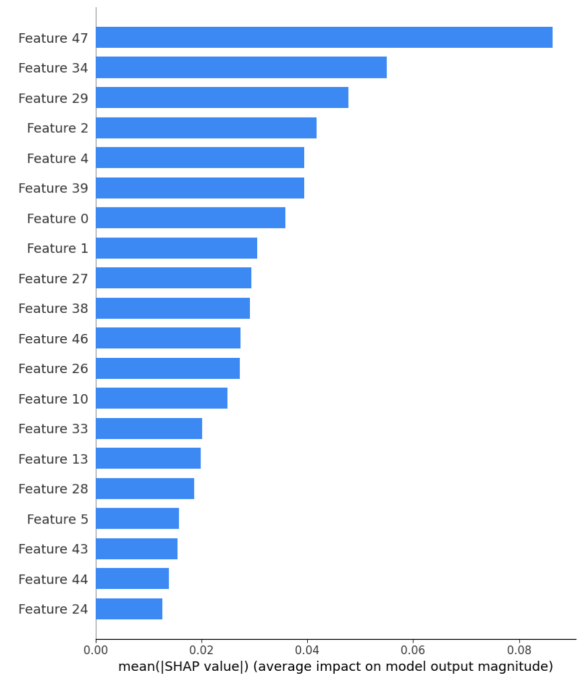


Figure 4.6: Bar plot

The above feature plot combines feature importance and feature effect. Summary-plot plots the Shapley value of each feature for each sample. It states which features are most important and their impact on the data set. Positions on the y-axis are determined by features Features with longer strips are more important for driving predictions, and colors indicate feature values (red high, blue low). The position on the x-axis is determined by the Shapley value, i.e. features to the right of the baseline have a positive effect and features to the left have a negative effect. Color allows us to match how changes in eigenvalues affect changes in risk. The overlap points are dithered in the y-axis direction, so we can understand the Shapley value distribution for each feature, and the features are ranked according to their importance. Calculate the Shapley value of the feature and plot a bar plot of the importance of the feature, which is consistent with the results implied in the summary plot.

Table 4.1: feature shapley value examples(top10 important)

index	column name	feature importance value
47	PctExtNullSelfRedirectHyperlinksRT	0.086388
34	FrequentDomainNameMismatch	0.055081
29	InsecureForms	0.047707
2	PathLevel	0.041805
4	NumDash	0.039373
39	IframeOrFrame	0.039372
0	NumDots	0.035864
1	SubdomainLevel	0.030492
27	PctExtResourceUrls	0.029454
38	SubmitInfoToEmail	0.029170
46	ExtMetaScriptLinkRT	0.027336
...	...	...

Chapter 5

Natural language processing method

5.1 BERT + logistic regression method

BERT, released by Google in 2018, is one of the most influential models of NLP, and in this article, I will further introduce and build a classification model based on BERT. The full name of BERT is Bidirectional Encoder Representation from Transformers, because the decoder cannot obtain the information to be predicted. The main innovation of the model is the pre-train method, which uses Masked LM and Next Sentence Prediction to capture the representation at the word and sentence level respectively. (Note: The images used below are all from the open source report of Google BERT model. The structure of BERT model is as follows: at the far left of the figure, compared with OpenAI GPT(Generative pre-trained transformer), BERT is a two-way Transformer block connection. Like the difference between one-way RNNS and two-way RNNS, it intuitively works better. In contrast to ELMo, although both are "two-way", the objective function is actually different. ELMo is named after $P(w_i|w_1, ..., w_{i-1})$and $P(w_i|w_{i+1}, ..., w_n)$as the objective function, the independent training of the two representations and then concatenated, while BERT is $P(w_i|w_1, ..., w_{i-1}, w_{i+1}, ..., w_n)$trains LM as an objective function.[1]

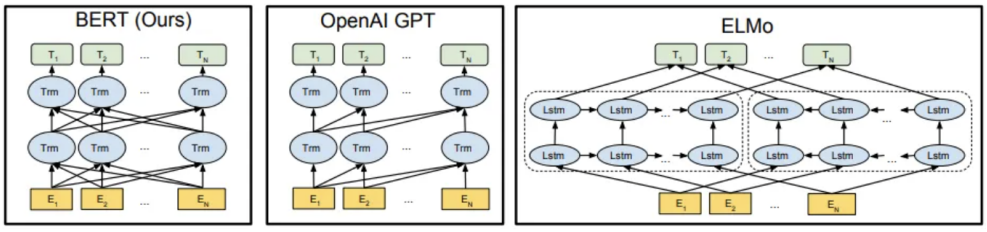


Figure 5.1: Google BERT structure

BERT's Embedding is the sum of three kinds of Embedding. Token Embeddings is a word vector, and the first word is the CLS flag, which can be used for later classification tasks. The Segment Embeddings are used to distinguish the two types of sentences because the pretraining is not only LM, but also a classification task with two sentences as input.

Position Embeddings are different from Transformer in the previous article. It is not a trigonometric function but a learned one.

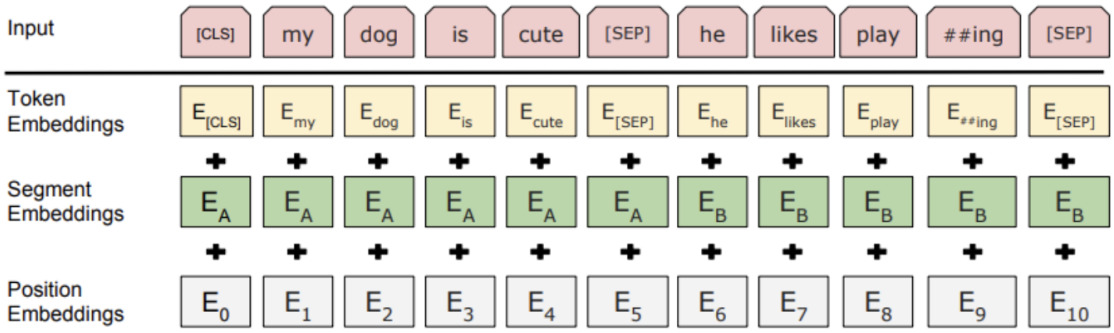


Figure 5.2: Google BERT embedding

For sequence-level classification tasks, which we need to complete in our article, BERT directly takes the final hidden state$C \in \mathbb{R}^H$of the first [CLS]token, softmax prediction label probability: $P = softmax(CW^T)$after adding a weight of $W \in \mathbb{R}^{K \times H}$

A binary classification model is implemented using a combination of BERT embedding and logistic regression. This model is used to classify URLs as benign or malicious. In this paper, the existing BERT model in hugging face is used, and the pre-trained BERT model and its word segmentation are loaded together. The "output-hidden-states" property is set to True to access the hidden state of the model. Then define a function to extract features from each URL. This function tags the URL, feeds it to the BERT model, and retrieves the hidden state of the tag. It then concatenates the last four hidden states of each tag and calculates their average. This produces a feature vector for each URL. After splitting the data set into the training set and the test set, SMOTE (synthetic minority oversampling technique) is used to balance the training set to deal with class imbalances. Training a logistic regression classifier on a balanced training set. Accuracy is then used as a metric to evaluate the performance of the model on the test set. Results can also be visualized using confusion matrices and classification reports. In a nutshell, BERT is used for feature extraction, and then a logistic regression model is trained on the extracted features for binary classification, resulting in the first model of our experiment.

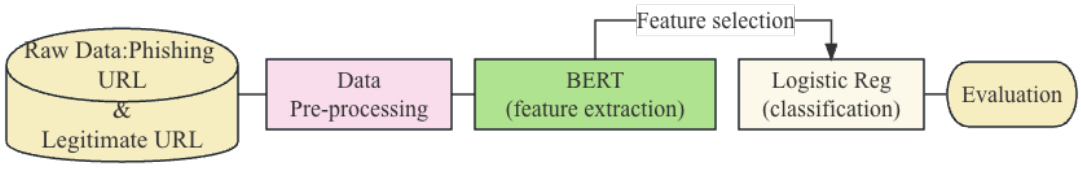


Figure 5.3: Our Model-1

5.2 RoBERTa method

In terms of model, RoBERTa basically has no outstanding innovations, but mainly makes several adjustments based on BERT: 1) longer training time, larger batch size, and more training data; 2) The next predict loss was removed; 3) The training sequence is longer; 4) Dynamically adjust Masking mechanism, etc. These modifications to BERT resulted in RoBERTa generally outperforming BERT on a range of NLP tasks. However, the "best" model can still vary depending on the specifics of the task and data. Therefore, it is always important to experiment with different models and configurations to find the best fit for a particular use case.

This article's model directly uses the RoBERTa model in the Transformer library to implement a text classification model for classifying URLs as benign or malicious. First, use the RoBERTa tag generator to text tag the URL. The word divider encodes the text into a format that the RoBERTa model can use. Apply padding and truncation to ensure that all sequences have the same length. After the data is prepared using the PyTorch dataset and DataLoader, the dataset receives the toked-up URL and its respective labels, and the data loader batches the data for training. The AdamW optimizer is used, along with the learning rate scheduler, and the mean square error (RMSE) is used as a loss function. Finally, the resulting data is used to train the model and evaluate its performance based on the validation data. If the performance of the validation data improves, the model parameters are saved. This process repeats the specified number of epochs. After training, the model's predictions are evaluated against actual labels, and the results are displayed using various metrics and graphs. Perform K-Fold cross-validation to evaluate the performance of the model. The data is divided into K folds, and the model is trained K times, each time using a different fold as a validation set.

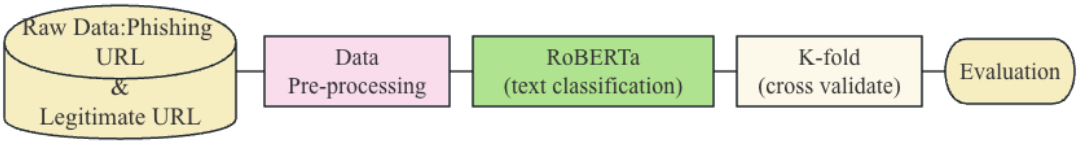


Figure 5.4: Our Model-2

5.3 Evaluation and result

For classification tasks, we usually consider the following indicators as evaluation criteria, and the results of two models from several experiments are also below.

- Accuracy: $acc = \frac{TP+TN}{TP+TN+FP+FN}$

- Precision: $prec = \frac{TP}{TP+TN}$
- Recall: $recall = \frac{TN}{FP+TN}$
- F1-score: $F_1 = 2\frac{precision \times recall}{precision+recall}$

Table 5.1: BERT

	Precision	Recall	f1-score
benign	0.98	0.99	0.99
defacement	0.95	0.95	0.95
malware	0.80	0.57	0.67
phishing	0.78	0.78	0.78
accuracy			0.96
macro avg	0.88	0.82	0.85
weighted avg	0.96	0.96	0.96

Table 5.2: RoBERTa

	Precision	Recall	f1-score
benign	0.9754	1.0000	0.9876
defacement	0.9600	0.8571	0.9057
malware	0.5714	1.0000	0.7273
phishing	1.0000	0.6667	0.8000
accuracy			0.9563
macro avg	0.8767	0.8810	0.8551
weighted avg	0.9640	0.9563	0.9562

For classification tasks, due to the particularity of the data set used in this experiment, the performance of model 2 is better than that of model 1. In fact, RoBERTa may perform better when the model is larger and the text content is more complex, and further experimental exploration is needed to determine the specific results. Next, I continue to study the relationship between the interpretability of the model and the characteristics of the text content.

5.4 SHAP explainer

Here, we also use Python's SHAP interpretability structure as the interpretation model. (Note: Due to the large computational cost and time consumption of SHAP library, only 100 pieces of data are used in the SHAP model as the model establishment standard for SHAP visualization, which is slightly different from the model used in the previous article. The performance of the classification model established here for RoBERTa model based on 50-shot is much lower than the previous results.) [2]

When we pass a single instance to a text graph, we get the importance of each tag overlaid on the original text corresponding to that tag. Red areas correspond to parts of the text that increase the output of the model when they are included, while blue areas decrease the output of the model when they are included. In the context of the sentiment analysis model, red corresponds to more positive comments and blue corresponds to more negative comments.

At the same time, use the bar chart visualization to get the following degree of contribution of each character to the result. We find that the results obtained may be somewhat different from intuition or reality because the explanatory model here uses a

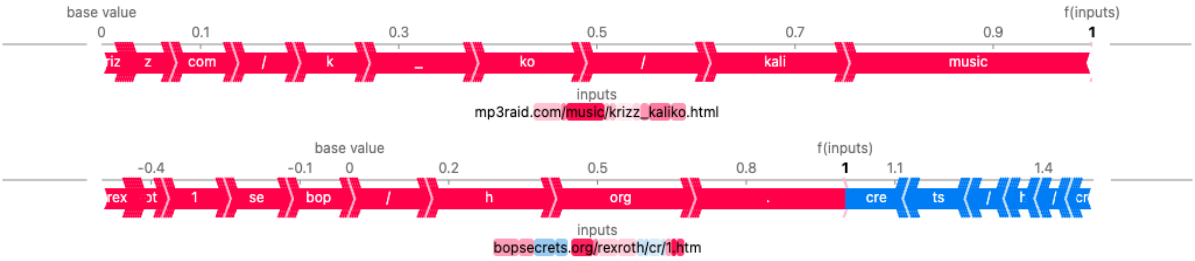


Figure 5.5: Examples on URLs labeled non-phishing

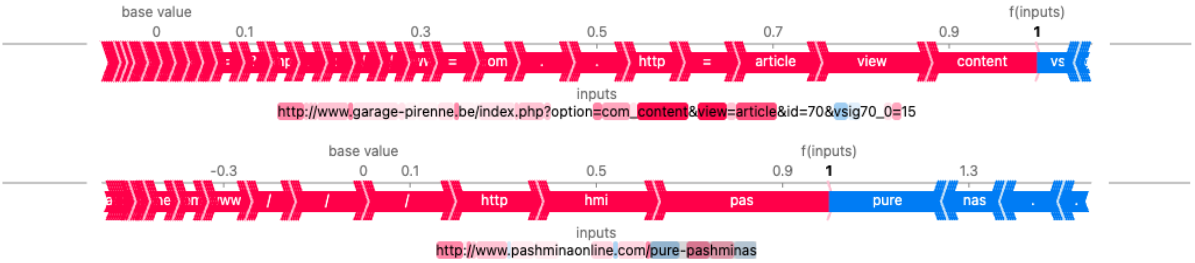


Figure 5.6: Examples on URLs labeled phishing

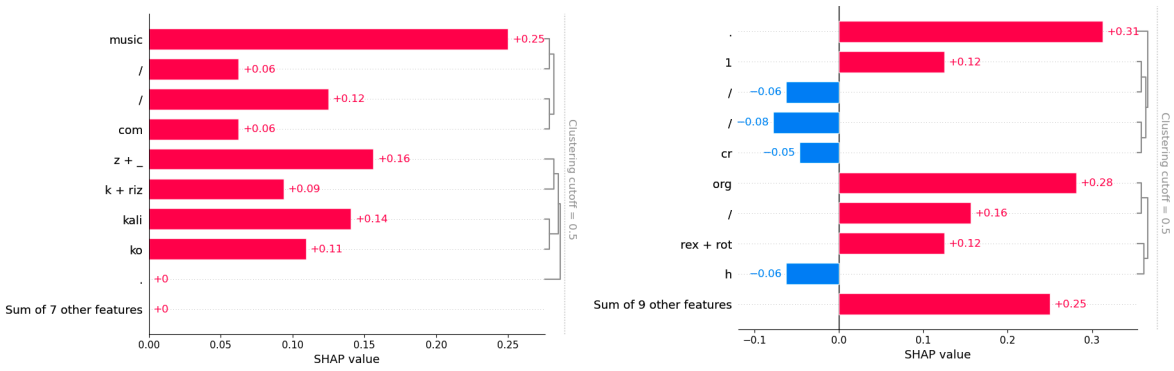


Figure 5.7: Bar-plots on URLs labeled non-phishing

small amount of data and needs to be further optimized.

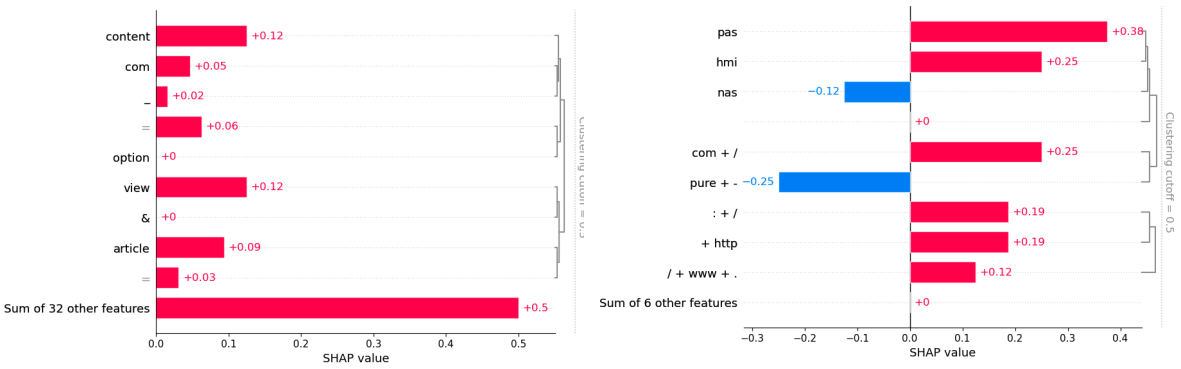


Figure 5.8: Bar-plots on URLs labeled phishing

Chapter 6

Limitation, future work, and Conclusion

First, the data set used in this article is too ideal, there are too few classifications, and the actual situation is often more complex. Secondly, the amount of data and model parameters used in this paper is small, so it is necessary to use larger data sets and more complex content for further research. For future work, because reality monitoring often does not have enough labeled data, I hope to add more unsupervised machine learning ideas in the subsequent research for a more realistic operation.

In addition, inspired by the characteristics of network security, since the network is an object that interacts with humans in real-time, I think more human influence factors can be added in subsequent research. For example, consider adding the idea of Reinforcement Learning with Human Feedback (RLHF). In addition, for security and privacy reasons, you can try to join the federal learning knowledge for further research. Finally, for the practical effect, we hope to find a hybrid method that can better combine the principle of feature extraction and classification.

Conslution

This paper analyzes the common phishing forms and detection methods and finds many applications of machine learning methods in them. For the general machine learning classifier, this paper classifies the website data with extracted features by means of logistic regression, random forest, and other methods. In addition, for plain text data, two classifiers are built using BERT and RoBERTa models, and the interpretability framework is used to explain the practical significance of neural networks in phishing.

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Scott Lundberg. Welcome to the shap documentatio. https://shap.readthedocs.io/en/latest/index.html, 2018.

[3] University of New Brunswick. Phishing dataset for machine learning: Feature evaluation. https://www.unb.ca/cic/datasets/url-2016.html, 2016.

[4] Santhosh Raminedi, Trilok Nath Pandey, Venkat Amith Woonna, Sletzer Concy Mascarenhas, and Arjun Bharani. Classification of phishing websites using machine learning models. In *2023 3rd International Conference on Artificial Intelligence and Signal Processing (AISP)*, pages 1–5, 2023. doi: 10.1109/AISP57993.2023.10134944.

[5] Choon Lin Tan. Phishing dataset for machine learning: Feature evaluation. https://data.mendeley.com/datasets/h3cgnj8hft/1, 2018. Accessed: 2018-03-24.

.1 Appendix A - Features of the feature dataset

1. NumDots: Phishing websites often use multiple subdomains to try to confuse users, each subdomain is separated by a dot, so a higher number of dots may indicate a phishing site.
2. SubdomainLevel: Subdomain levels might indicate a phishing site trying to mimic a legitimate site.
3. PathLevel: Phishing sites may use longer paths in an attempt to hide their true nature, so a relatively high path level should be suspicious.
4. UrlLength: Longer URLs might indicate that the site is trying to hide something in the URL, such as a misleading domain name or a malicious query string.
5. NumDash: Dashes are often used in domain names of phishing websites to mimic legitimate websites.

6. NumDashInHostname: More dashes in the hostname might be an attempt to mimic a legitimate site.
7. AtSymbol: The presence of '@' in a URL often indicates a phishing attempt because it can lead to a misleading domain name.
8. TildeSymbol: The tilde is rarely used in legitimate URLs, so its presence might indicate a phishing site.
9. NumUnderscore: Underscores are not allowed in domain names according to domain name standards (although they are allowed in paths and parameters), so their presence could be suspicious.
10. NumPercent: The percent symbol is often used in URL encoding, and a high number of them might indicate an attempt to hide something.
11. NumQueryComponents: A high number of query components can indicate a complex URL that might be trying to hide something.
12. NumAmpersand: Ampersands are used in query strings, and a high number might indicate a complex URL that's trying to hide something.
13. NumHash: The hash symbol is used in URLs to refer to a specific part of a webpage, and a high number of them might be suspicious.
14. NumNumericChars: A high number of numeric characters in a URL can be a sign of a suspicious site, as many phishing sites use IP addresses or random numeric strings in their URLs.
15. NoHttps: HTTPS indicates a secure connection. If a site doesn't use HTTPS, it might be less trustworthy.
16. RandomString: Phishing sites often use random strings in their URLs to avoid being blocked by filters.
17. IpAddress: If a URL uses an IP address instead of a domain name, it's often a sign of a phishing site.
18. DomainInSubdomains: A phishing site might include a legitimate domain name in its subdomains to try to trick users.
19. DomainInPaths: A phishing site might include a legitimate domain name in its paths.
20. HttpsInHostname: If "https" is in the hostname of the URL, it might be a phishing attempt trying to trick users into thinking the site is secure.
21. HostnameLength: A longer hostname might be an attempt to hide something and could indicate a phishing site.
22. PathLength: A longer path might be an attempt to hide something.
23. QueryLength: A longer query string can indicate a complex URL that's trying to hide something.
24. DoubleSlashInPath: Double slashes are not typically used in legitimate URLs, so their presence could be suspicious.

25. NumSensitiveWords: If a URL contains sensitive words like "login", "password", etc., it might be a phishing attempt.
26. EmbeddedBrandName: If a URL has a popular brand name embedded in it, it could be a phishing attempt trying to impersonate the brand.
27. PctExtHyperlinks: A high percentage of external hyperlinks could indicate a site that's trying to redirect users to malicious sites.
28. PctExtResourceUrls: A high percentage of external resource URLs could be suspicious.
29. ExtFavicon: A favicon loaded from an external URL might indicate a phishing site trying to mimic a legitimate site.
30. InsecureForms: If a site uses forms that don't use a secure connection, it might be a phishing site.
31. RelativeFormAction: If a form action uses a relative URL, it might be an attempt to hide the actual destination of the form data.
32. ExtFormAction: If a form action uses an external URL, it might be a phishing site trying to collect form data.
33. AbnormalFormAction: If a form action is different from the site domain, it might be a phishing site trying to collect form data.
34. PctNullSelfRedirectHyperlinks: A high percentage of self-redirecting hyperlinks with a null target might indicate a phishing site.
35. FrequentDomainNameMismatch: If the domain name frequently mismatches between the site and its content, it might be a phishing site.
36. FakeLinkInStatusBar: If a site uses JavaScript to display fake links in the status bar, it might be a phishing site.
37. RightClickDisabled: If a site disables right-clicking, it might be trying to prevent users from inspecting its content, which could indicate a phishing site.
38. PopUpWindow: Pop-up windows can be used to trick users, so a site that uses them might be a phishing site.
39. SubmitInfoToEmail: If a site submits form data to an email address, it might be a phishing site trying to collect personal information.
40. IframeOrFrame: Iframes can be used to include content from other sites, so a site that uses them might be a phishing site.
41. MissingTitle: A missing title might indicate a hastily constructed phishing site.
42. ImagesOnlyInForm: If a form only contains images, it might be an attempt to trick users into thinking the form is legitimate.
43. SubdomainLevelRT, UrlLengthRT, PctExtResourceUrlsRT, AbnormalExtFormActionR, ExtMetaScriptLinkRT, PctExtNullSelfRedirectHyperlinksRT: These features seem to be real-time (RT) or reactive features. They are likely similar to their non-RT counterparts but measured in real-time as a user interacts with the site. A higher or abnormal

value in real-time compared to the initial load could indicate a phishing site.

.2 Appendix B - Code

For the large space of code, you can refer to all project code at my GitHub repository: https://github.com/Carrief-0908/CamOSRP-submission