Website Phishing ML Project

**Group Members**

Dino Anastasopoulos: 1900661

Timothy Walters: 1855167

Razeen Gani: 1842173

Reece James Peters: 1924514

**Dataset Information**

We chose to study a dataset about Website Phishing.

The dataset consists of 9 attributes, 3 class types and 1353 datapoints. The goal is to predict the classification, that is whether the website is legitimate, suspicious or phishy.

The attributes and their corresponding values are as follows:

(Note: 1 = Legit, 0 = Suspicious, -1 = Phishy)

* **Server Form Handler:**

Once the user submitted his information; the webpage will transfer the information to a server so that it can process it. Normally, the information is processed from the same domain where the webpage is being loaded. Phishers resort to make the server form handler either empty or the information is transferred to somewhere different than the legitimate domain.

* + 1: otherwise
  + 0: SFH redirects to different domain
  + -1: SFH is ‘about : blank’ or empty
* **PopUp Window:**

Usually authenticated sites do not ask users to submit their credentials via a popup window.

* + 1: otherwise
  + 0: rightClick alert showing
  + -1: rightClick disabled
* **Fake HTTPs protocol/SSL final:**

The existence of HTTPs protocol every time sensitive information is being transferred reflects that the user is certainly connected with an honest website. However, phishers may use a fake HTTPs protocol so that users may be deceived, so it is recommended to check that the HTTPs protocol is offered by a trusted issuer.

* + 1: use of https and trusted issuer and age ≥ 2 years
  + 0: using https and issuer is not trusted
  + -1: otherwise
* **Request URL:**

A webpage usually consists of text and some objects such as images and videos. Typically, these objects are loaded into the webpage from the same server of the webpage. If the objects are loaded from a domain other than the one typed in the URL address bar, the webpage is potentially suspicious.

* + 1: request URL < 22%
  + 0: 22% ≤ request URL < 61%
  + -1: otherwise
* **URL of Anchor:**

Similar to the URL feature, but here the links within the webpage may point to a domain different from the domain typed in the URL address bar.

* + 1: URL anchor % < 31%
  + 0: 31% ≤ URL anchor < 67%
  + -1: otherwise
* **Web Traffic:**

Legitimate websites usually have high traffic since they are being visited regularly. Since phishing websites normally have a relatively short life, they have low web traffic.

* + 1: Web Traffic > 150 000
  + 0: 150 000 ≥ Web Traffic ≥ 50 000
  + -1: otherwise
* **URL Length:**

Phishers hide the suspicious part of the URL to redirect the information submitted by users or redirect the uploaded page to a suspicious domain.

* + 1: URL Length < 54
  + 0: 54 ≤ URL Length ≤ 75
  + -1: URL Length > 75
* **Age of Domain:**

Websites that have an online presence of less than 1 year, can be considered risky.

* + 1: age ≤ 6 months
  + -1: otherwise
* **IP Address in URL:**

Using an IP address in the domain name of the URL is an indicator someone is trying to access the personal information

* + 1: otherwise
  + 0: IP Address exists in URL
* **Classification:**
  + 1: Legitimate
  + 0: Suspicious
  + -1: Phishy

**Extra Algorithm – Artificial Neural Network**

Since our group spent a lot of time working on the other 2 algorithms, but were very interested in NNs, we decided to implement a very simple NN as a bonus algorithm to see what they are capable of without any of the optimizations available. We decided on using a NN with 9 input neurons (1 for each attribute), 1 hidden layer with 6 neurons (as this is the mean of the size of the input and output layers), and 3 neurons on our output layer, corresponding to phishy, suspicious and legitimate.

Due to the simplicity of our model, we were not tweaking hyperparameters and therefore had no need for validation data, so we used 60% of our data to train the neural network and 40% to test it. After trying several different ratios of training to testing data we decided that this was the best to avoid over/underfitting the model. A larger testing data size allowed mistakes in the prediction to have a smaller negative impact on the accuracy.

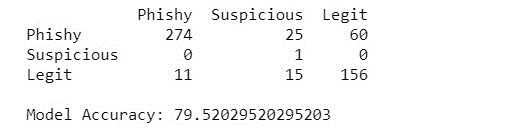
The data’s attributes and classes consist of the same values so there was no need to normalize the data. However, we attempted to change the phishy labels(-1) to 0, suspicious(0) to 0.5, and legit(1) remained as 1, as we thought that this more accurately scales the data to be either weakly (0-0.5) or strongly (0.5 – 1) correlated to legitimacy. Although, this gave us a problem in that all phishy attributes had zero effect on the changing of weights rather than a negative effect as we needed, so we decided to leave the values as -1, 0 and 1.

Our data set had insufficient records classified as “suspicious” and this resulted in our model being more prone to classifying as phishy or legitimate, and unable to classify examples as suspicious.

The training errors were a little bit all over the place because on every iteration we were only changing the weights. If we had added biases and regularization, the error would’ve decreased more normally. However, they did gradually decrease over time. We also noticed that an excessive number of iterations does not have any benefits, and results in overfitting.

The best performance we reached with the Neural Network was around 85%, which was remarkable since we barely added any optimizations. The average accuracy was around 70%, which we were pretty happy with, and we encountered a few outliers below 40%. The reason for this inconsistency comes from the overfitting to our training data which resulted from the simplicity of our model. Increasing the accuracy would be possible by adding optimizations such as momentum, and adding regularization and biases to the backpropagation.

* Bonus algorithm, wanted to keep it basic to see what would happen
* Not enough data points of suspicious web pages to train data
* We tried to normalize by shifting between 0-1 but then all phishy records don’t have an effect on the changing of weights, so we left it as -1,0,1
* Overfitting due to simplicity of model – leads to inconsistency as sometimes the model is accurate and sometimes extremely off
* Errors whack because of overfitting and no use of biases, regularization, momentum, etc
* Excessive number of iterations doesn’t actually help much, and results in overfitting
* One hidden layer with number of neurons equal to mean of input and output. This was good
* Best performance was around 80%, average was around 70% with few outliers under 40%



**True**

**Predicted**

A close up of a map

Description automatically generated

**Resources**

Dataset Information:

<http://fadifayez.com/wp-content/uploads/2017/11/Phishing-detection-based-Associative-Classification-data-mining.pdf>

Numpy:

<https://numpy.org/doc/stable/reference/>

Pandas:

<https://pandas.pydata.org/pandas-docs/stable/reference/frame.html>