Malicious Web Page Prediction

Nadine Safwat - 900212508

Link to notebook and dataset dump: 🗀 Machine Learning Project

DATASET CHOICE

Dataset of Malicious and Benign Webpages

This is the chosen dataset for this project as it has the required features as well as an appropriate size. On top of the statistics there are also references as to where the data has been gathered from which shows that the data available is reliable. It has also conveniently been split into respective testing and training sets. The dataset contains extracted attributes from websites that can be used for Classification of webpages as malicious or benign. There are 11 features, some of which are the URL, the IP address, the domain, if the website uses HTTPS or not, and most importantly whether the webpage is malicious or not, which is the label. There are 1.2 Million records in the training set and just over 350K records in the testing set. Kaggle has given the dataset a usability rating of 8.82

This dataset is relevant as it has almost all the features that can accurately predict if the web page is malicious or not and it has more than enough points for the models to train on. The only limitation I found in the set is that only 2% of the data points are classified as malicious which will cause changes in the next two phases. However, this dataset was the most suitable for the goal of this project.

ORIGINAL FEATURES

- URL: The URL of the webpage. (string)
- URL_LEN: The length of URL. (int64)
- IP_ADD: IP Address of the webpage. (string)
- GEO_LOC: The geographic location where the webpage is hosted. (categorical)
- TLD: The Top Level Domain of the webpage. (categorical)
- WHO_IS: Whether the WHO IS domain information is complete or not. (Bool)
- HTTPS: Whether the site uses https or http. (Bool)
- JS_LEN: Length of JavaScript code on the webpage. (int64)
- JS_OBF_LEN: Length of obfuscated JavaScript code. (int64)
- CONTENT: The raw webpage content including JavaScript code. (text)
- LABEL: The class label for benign or malicious webpage. (1 for benign and 0 for malicious)

CLEANING AND PRE-PROCESSING

STEP 1:

To begin, a quick analysis of the existing features is done, like noting which features are numerical and which are categorical, and easy fixes are applied. These include:

- Removing the index column from Kaggle
- Making all boolean columns have 1 and 0 representation

We can also note that there are no NULL values that need to be handled in the dataset

STEP 2:

Next we can see that there are two categorical columns: GEO_LOC and TLD and these can be engineered using one hot encoding. This is done using OneHotEncoder from the SkLearn library. Using the nunique() function in pandas we know that GEO_LOC has 234 unique categories and TLD has 1246.

STEP 3:

In order to analyze and normalize the numerical features we will first turn the IP_ADD feature into its corresponding decimal number and extract multiple features from the URL [1] as the URL itself cannot be compared against new sets for similarities. The features we will extract include:

- Usage of IP address in domain
- Entropy of the url [2]
- The number of digits in the URL
- The number of query parameters (separated by ?)
- The number of fragments (separated by #)
- The number of hexadecimal spaces (%20)

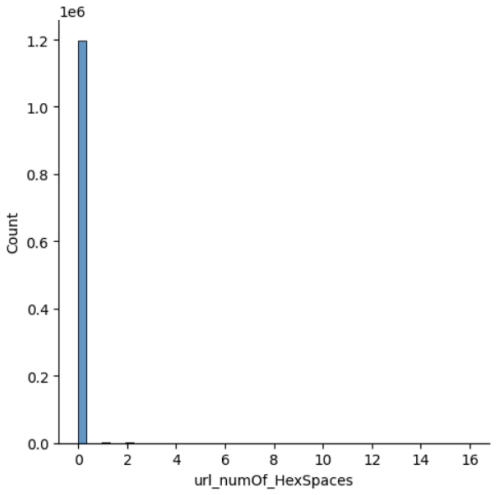
- The number of at (@) characters

Now, we can get the correlation between each numerical feature and the label to ensure that no one feature is in direct correlation.

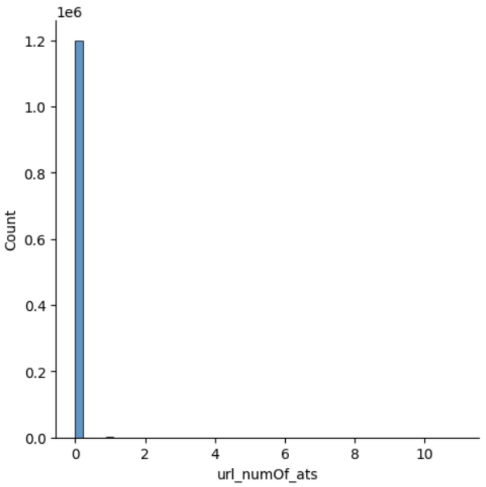
```
Correlation between url_numOf_fragments and the label:  0.46
Correlation between url_numOf_query_paramters and the label:  2.27
Correlation between url_numOf_ats and the label:  0.17
Correlation between url_numOf_HexSpaces and the label:  0.45
Correlation between url_numOf_digits and the label:  3.27
Correlation between url_entropy and the label:  1.05
Correlation between url_len and the label:  -1.37
Correlation between ip_add and the label:  0.04
Correlation between js_len and the label:  -73.61
Correlation between js_obf_len and the label:  -89.12
```

This shows that all features other than JS_OBF_LEN are not directly correlated thus we will remove this feature. The next step is to them plot each feature and find the distribution for each so that they can be normalized

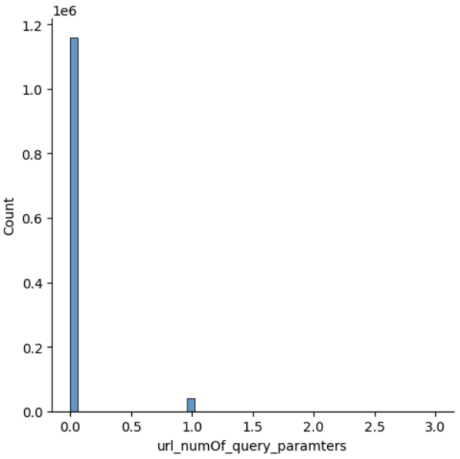
Hexadecimal Space (%20)



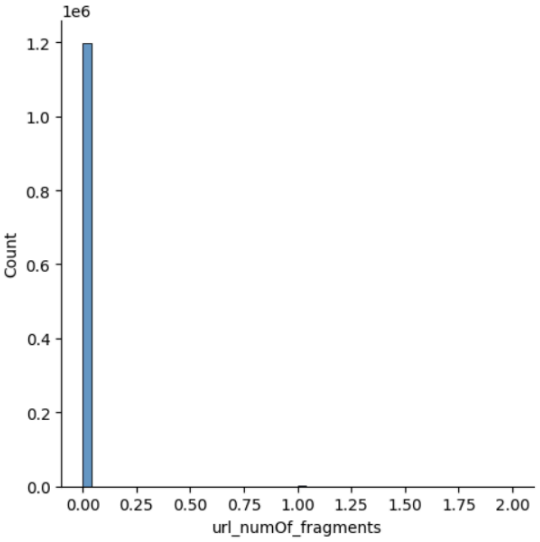
Ats (@)



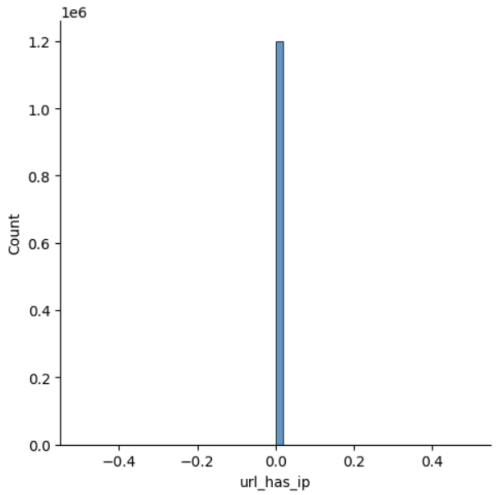
Query Parameters (?)



Fragments (#)

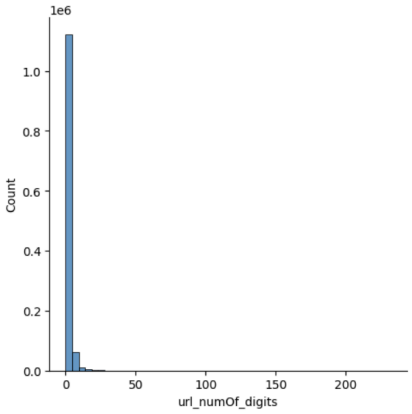


IP in Domain

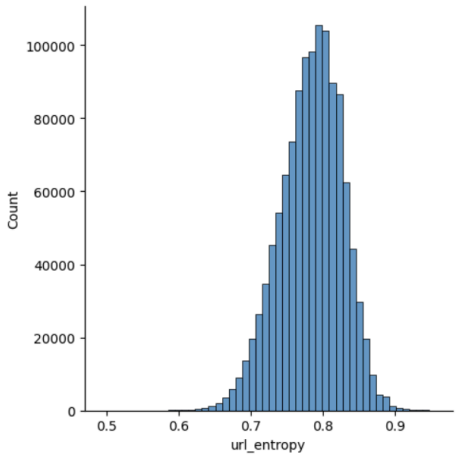


All these features have very little to no distribution, So they will not be helpful when comparing similarities or distances, so we will disregard them.

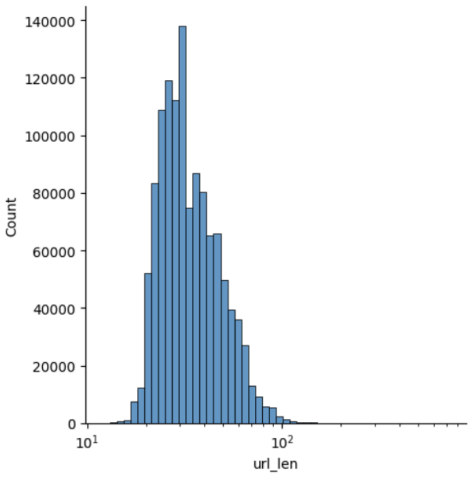
Digits:



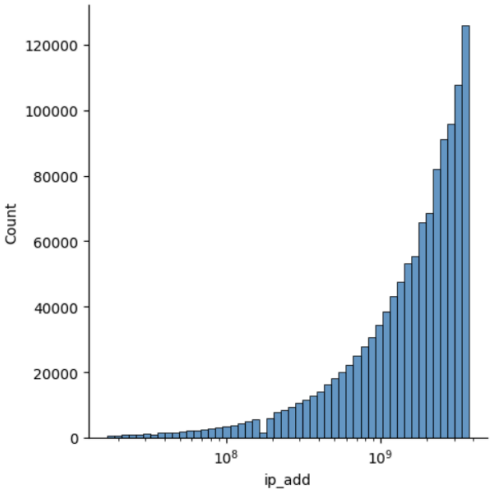
Entropy:



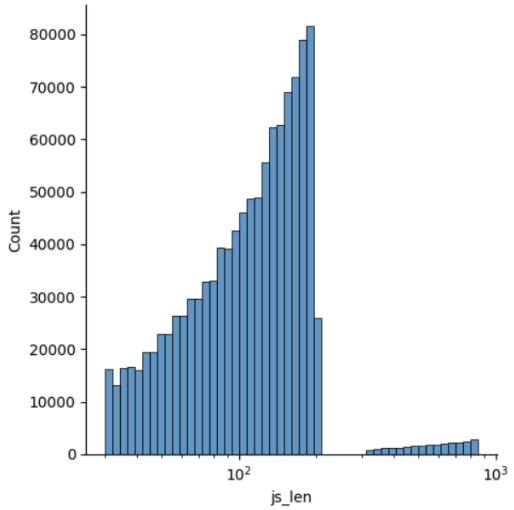
URL length:



IP address



Javascript length



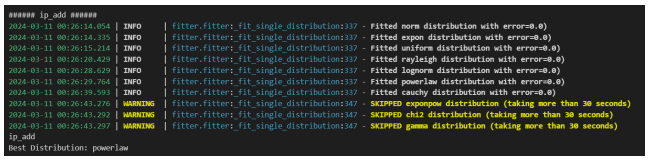
It is clear that all these features (other than entropy) need to be normalized, so we will use the library

Fitter to find their distributions and choose the best way to normalize accordingly

Digits:

URL length:

IP address



Javascript length

From this data I chose to normalize the url length with log scaling, as there it has lognormal scaling and there are no 0s in the data. For the number of digits, the javascript length and the ip address, the best scaling method was z scaling as there are 0s in the data or it works best with its given distribution. After applying the scaling we get:

```
Mean of url_numOf_digits:  5.3752557960251577e-17
SD of url_numOf_digits:  1.000000416666927

Mean of url_entropy:  0.7831284275583331
SD of url_entropy:  0.04299186607911006

Mean of url_len:  1.5269976183262748
SD of url_len:  0.1487179227025909

Mean of ip_add:  -1.8024100730447875e-17
SD of ip_add:  1.0000004166669276

Mean of js_len:  4.933505456013639e-16
SD of js_len:  1.000000416666927
```

STEP 4:

The only column now left to engineer is the content feature. This has the content of the webpage as well as some javascript code that can be used to detect if the webpage is malicious or not. To encode this text we will follow a set of given steps:

- Remove stop words (the, an, …)
- Stemming (reduce words to their simplest form)
- Apply TF-IDF

Luckily there are pre existing libraries in python that can do all this for us. For removing stop words and stemming we will use the library nltk and to apply TF_IDF we can use TfidfVectorizer from sklearn.

FINAL FEATURES

Compared to the beginning features, we have not added too many new features, but the number of rows has increased greatly due to the text and one hot encoding. In summary the final features include:

- Number of digits in the URL (int64)
- URL entropy (int64)
- URL length (int64)
- IP address (int64)
- Is the Who_is documentation complete (1/0)
- Does the web page use Https (1/0)
- Javascript length (int64)
- Geographical locations (each location its own feature (1/0))
- TLD (each domain its own feature (1/0))
- Content Words

Removed Features:

- Javascript obfuscated length
- URL

REFERENCES:

[1] Hong, J., Kim, T., Liu, J., Park, N., & Kim, S.-W. (2020)2. Phishing URL Detection with Lexical Features and Blacklisted Domains1. In S. Jajodia, G. Cybenko, V.S. Subrahmanian, V. Swarup, C. Wang, & M. Wellman (Eds.), Adaptive Autonomous Secure Cyber Systems (pp. 253–267)3. Cham: Springer. https://doi.org/10.1007/978-3-030-33432-1_124

[2] Ramakrishnan, R. (2021). URL Feature Engineering and Classification1. Nerd For Tech. Retrieved from https://medium.com/nerd-for-tech/url-feature-engineering-and-classification-66c0512fb34d