**Hammad Zahoor**

**20L-0971**

**8A**

**Assignment # 3**

**Cyber Security**

**Dataset :**

# **Malicious URLs dataset**

Cybersecurity is seriously threatened by rogue websites or URLs. Malicious URLs cause billions of dollars' worth of losses annually by hosting unwanted content (spam, phishing, drive-by downloads, etc.) and tricking gullible visitors into falling for frauds (money loss, theft of personal information, and malware installation).

This dataset has **651,191 URLs**, out of which 428103 benign or safe URLs, 96457 defacement URLs, 94111 phishing URLs, and 32520 malware URLs.

**Benign URLs:** These are the URLs that are normal or non-malicious. They belong to normal websites, services and users.

**Defacement URLs:** These are the URLs that are changed or altered by malicious activities and can have bad links, images or text.

**Phishing URLs:** These URLs take you to the website that can take your authentication details, personal and financial details.

**Malware URLs:** These are the URLs that take you to malware sites which include trojan horses, viruses, spyware, ransomware and more.

This dataset is overall collected from different sources like [URL dataset (ISCX-URL-2016)](https://www.unb.ca/cic/datasets/url-2016.html) **[1]**, [Malware domain black list dataset](http://www.malwaredomains.com/wordpress/?page_id=66), Faizan git repos **[2]**, Phishtank datasets **[3]** and PhishStorm datasets **[4].**

As we go further, we extract more features from urls using different techniques and got these named as url\_type (like benign, phishing, defacement and malware), url\_len (length of URL), letters\_count (count of letters in URL) , digits\_count (count of digits in URL), special\_chars\_count (count of special characters), shortened (1 if URL uses shortening service), abnormal\_url (1 if URL is abnormal), secure\_http (1 if URL is secured), have\_ip (1 if it has IP), url\_region (region of URL), root\_domain (domain of URL).

**Machine Learning Approaches:**

**DecisionTreeClassifier**

It is suitable for multi-class problems and has some nodes and branches where each node has a decision based on features and branches have outcomes. They are way simpler than other approaches and can handle both numerical and categorical data as ours and based on a specific threshold.

**RandomForestClassifier**

It is also suitable for multi-class problems and uses multiple decision trees to calculate outcomes. It uses a random subset of data and then averages the predictions. It reduces overfitting and can also handle multi-features data easily.

**AdaBoostClassifier**

It uses many classifiers to make a string classifier and it is also suitable for multi-class problems like the one which we are using. It iteratively trains all classifiers with weights and also handles class imbalance.

**KNeighborsClassifier**

It is a lazy learning algorithm and classifies data based on nearest neighbors with multiclass data. It is also suitable for our problem and it can handle multi-class data easily. It is simple and can be handled easily using clustered data.

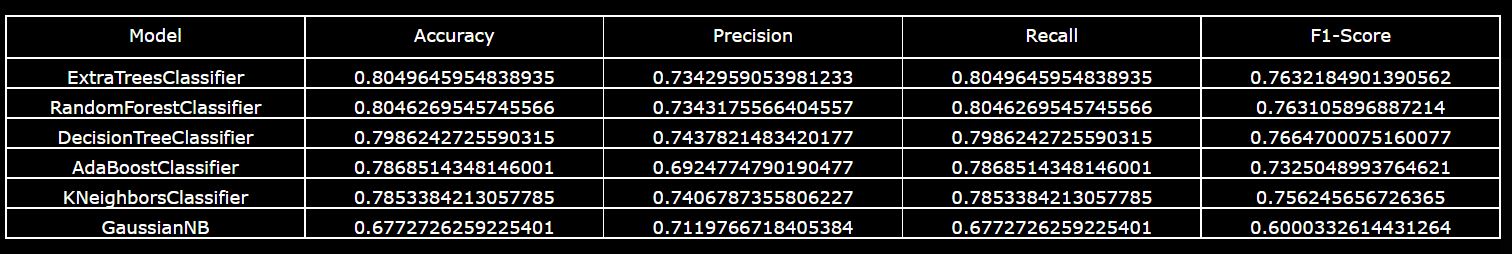
**ExtraTreesClassifier**

It is similar to a random forest classifier but adds some randomness in features of the data. It is also suitable for multi-class problems and reduces variance and overfitting. It is also robust to outliers and noisy data.

**GaussianNB**

It is suitable for multi-class problems and assumes features are independent and normally distributed. It is based on Bayes theorem and it is computationally efficient. It is simple and performs well on classification tasks.

Each of these classifiers has its own strengths and weaknesses, but we tried them all to differentiate between them and to check which is best for our dataset and model predictions.



**Related Work**

# **Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions**

M. Aljabri *et al.*, “Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions,” *IEEE Access*, vol. 10, pp. 121395–121417, 2022, doi: <https://doi.org/10.1109/access.2022.3222307>.

In this journal, there is a comprehensive analysis of the literature emphasizing the primary methods for machine learning-based malicious URL identification, while accounting for detection technologies, feature kinds, datasets, and constraints in the literature. Furthermore, it also highlights the directions of research in this area because there aren't many studies on the identification of harmful Arabic websites.

# **Detecting malicious URLs using machine learning techniques**

F. Vanhoenshoven, G. Nápoles, R. Falcon, K. Vanhoof and M. Köppen, "Detecting malicious URLs using machine learning techniques," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, Greece, 2016, pp. 1-8, doi: 10.1109/SSCI.2016.7850079.

In this work, it tackles the binary classification issue of malicious URL identification and examine the effectiveness of many popular classifiers, including Random Forest, Naïve Bayes, Support Vector Machines, Multi-Layer Perceptrons, Decision Trees, and k-Nearest Neighbors. In addition, it embraced a publicly available dataset with 3.2 million attributes and 2.4 million URLs. Numerical simulations have demonstrated that the majority of classification strategies attain satisfactory prediction rates without the need for complex feature selection techniques or domain expert involvement. Specifically, Multi-Layer Perceptrons and Random Forests achieve the maximum accuracy.

**Pre-Processing**

1. df.isnull().sum() - Checks for null values in the dataset and we have no null values in it.
2. urls\_data['url'] = urls\_data['url'].replace('www.', '', regex=True) - Replaces “www” with “” to perform other operations
3. Replaces 'benign' with 0, defacement with 1, phishing with 2, malware with 3 in our dataset to categorize and have numerical data for further processing.
4. We hash the root\_domain and url\_region feature using a hash function to have a numerical value for them.
5. df.drop\_duplicates(inplace=True) - delete the duplicate rows from dataset.
6. There are no outliers, skewed data, null values, or any other issue in our dataset.

**Feature Extraction**

1. urls\_data['url\_len'] = urls\_data['url'].apply(lambda x: get\_url\_length(str(x)))

It gives the length of URLs.

1. num\_letters = sum(char.isalpha() for char **in** url)

It gives the count of alphabets in URLs.

1. num\_digits = sum(char.isdigit() for char **in** url)

It gives the count of numbers in URLs.

1. num\_special\_chars = sum(char **in** special\_chars for char **in** url)

It gives the count of special characters in URLs.

1. if domain.lower() **in** common\_shortening\_services: return 1

It gives 1 if the URL has shortening services means it can be converted to a short URL so can be processed easily. It is used for easy computation and analysis purposes in our dataset.

1. match = re.search(netloc, url)

It gives 1 if the URL is abnormal means that the network location part of the URL is not found within the URL itself using regular expression search.

1. int(urlparse(url).scheme == 'https')

It gives 1 if the URL is secure means using https and 0 if http is used.

1. ip = ipaddress.ip\_address(parsed\_url.hostname)

if the URL has a hostname then it calculates its IPv4 or IPv6 address. If the address is made then it returns 1.

1. primary\_domain.endswith(ccTLD) return region

It gives the region by checking the country code in the URL and then hashes it to have a numerical value.

1. root\_domain = extracted.domain

It uses **tldextract.extract(url)** to extract the subdomain, domain, and top-level domain (TLD) parts of the URL and return domain of the URL. We hashes it to have a numerical value.

**Final Features**

There are a total of 11 features that we have extracted from URLs and they are used to predict malicious websites.

*'url\_type', 'url\_len', 'letters\_count', 'digits\_count',*

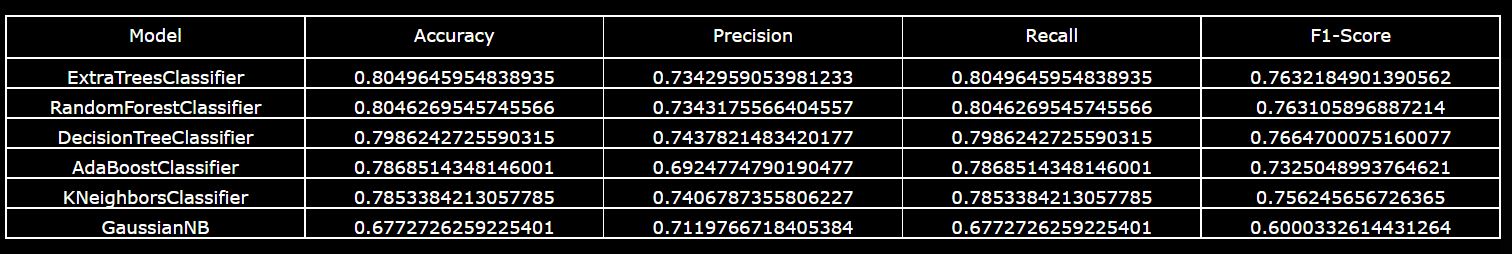
*'special\_chars\_count', 'shortened', 'abnormal\_url', 'secure\_http',*

*'have\_ip', 'url\_region', 'root\_domain'*

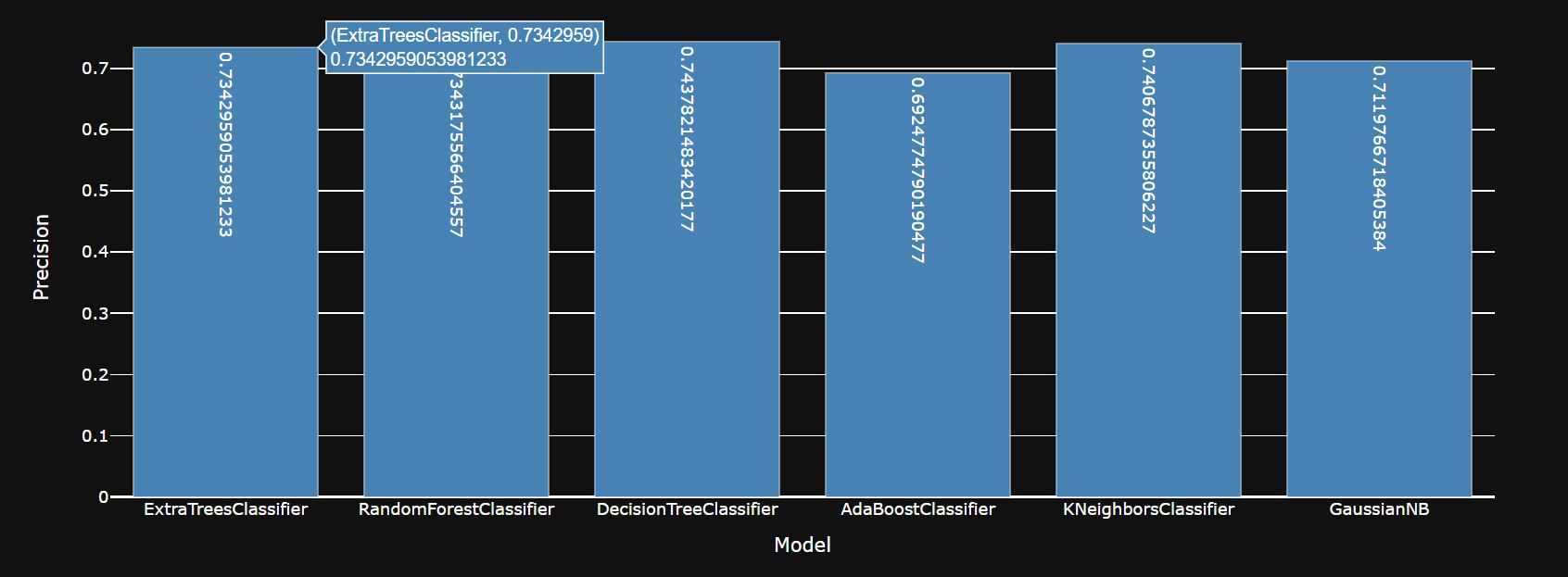
Before moving to results, we should know

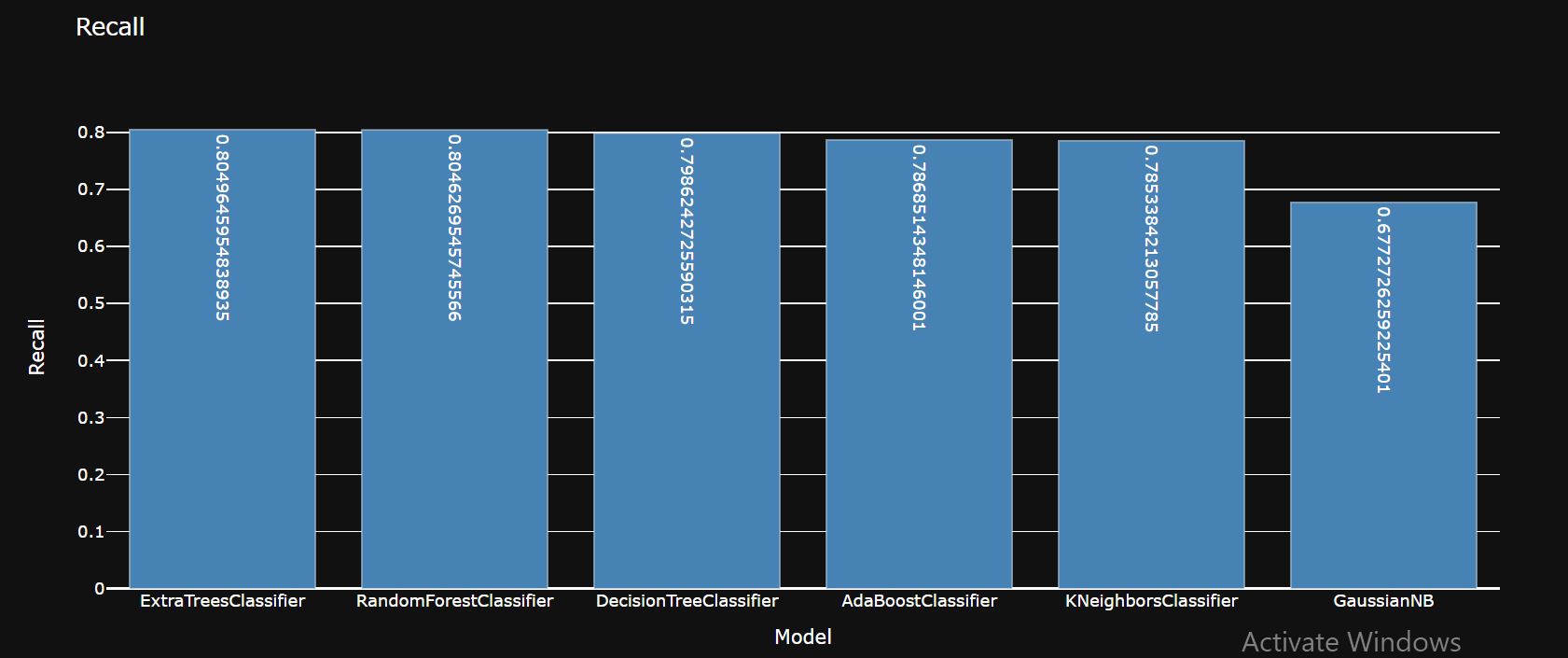
* **Accuracy:** The proportion of correctly classified instances among all instances, indicating overall model correctness.
* **Recall (Sensitivity):** The proportion of true positives correctly identified, measuring the model's ability to capture positive instances.
* **Precision:** The proportion of true positives among instances predicted as positive, showing the accuracy of positive predictions.
* **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of model performance for imbalanced classes.

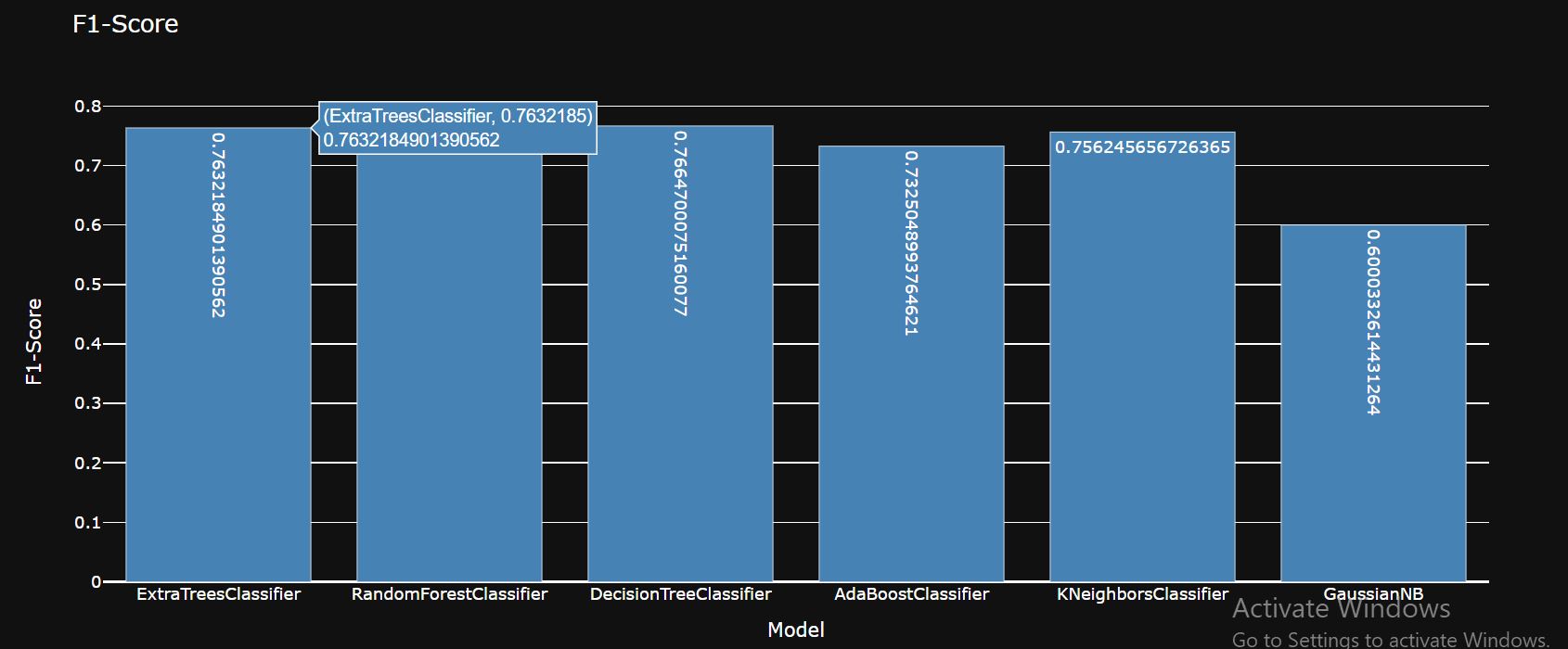
**Results**

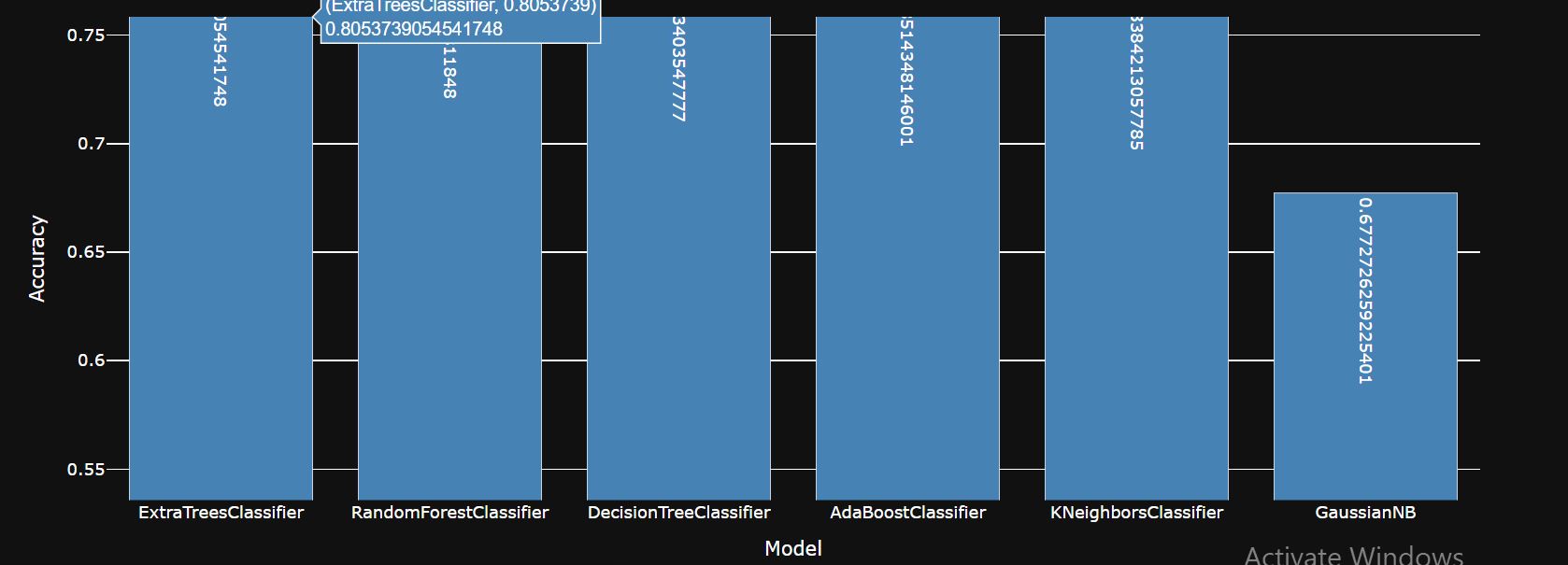


There are results of different Machine Learning algorithms and we can see that RandomForestClassifer, ExtraTreesClassifier gives best results with respect to accuracy and DecisionTreeClassifier and KNearestNeighbor have best results with respect to precision. So, overall these Machine Learning Algorithms i.e RandomForestClassifer, ExtraTreesClassifier gives the best results.

****

****

****

****

**References :**

1. <https://www.unb.ca/cic/datasets/url-2016.html>
2. https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs/tree/master/data
3. <https://www.phishtank.com/developer_info.php>
4. https://research.aalto.fi/en/datasets/phishstorm-phishing-legitimate-url-dataset