# **DETECTION OF PHISHING WEBSITES**

# **INCOGNITO GROUP MEMBERS**

Anne Nyambura

Brian Onyango

Gathoni Njoroge

Matilda Kadzo

Victoria Mbaka

# 

**Supervisors**

Nikita Njoroge

William Okomba

Moringa School

Data Science Core

Cohort 14 A

# **BUSINESS UNDERSTANDING**

## **Business Overview**

The internet has become a huge part of our lives, in the sense that most if not all of the information that we gather and disseminate, especially through social media platforms, is collected or obtained via the internet. It is considered to be a network of computers and servers that has valuable and sometimes sensitive data, therefore there are numerous security measures in place to protect the data. However there is one weak link, the human user. When a user freely shares information or even their computer access, all security measures can only go so far when it comes to data and device protection.

Phishing (pronounced: fishing), the most common type of social engineering, is an attack that attempts to steal money and/or identity, by getting a person to reveal their personal information -- such as credit card numbers, bank information, or passwords -- on websites that masquerade as legitimate. Cybercriminals typically pretend to be reputable companies, friends, or acquaintances in a fake message or advertisement that contains a link to a phishing website.

For a clear understanding of how attackers think when they create a phishing domain let's check the URL structure. Uniform Resource Locator (URL) is created to address web pages. Below are the relevant parts within the structure of a URL.

Uniform Resource Locator (URL) is created to address web pages. The figure below shows relevant parts in the structure of a typical URL.

# 

It begins with:

* A **protocol/scheme** used to access the page on the Internet. Eg http, https,ftps,file etc.
* A **Host name** consists of a subdomain name and a domain name.
* Subdomain indicates which particular page of your website the web browser should serve up. Eg. **www**.exampleurl.com, **blog**.hubspot.com
* Domain name consists of a second-level domain which is the name of your website , and top level domain that specifies the type of entity your organization registers as on the Internet. Eg. .com, .edu, .ke
* The **path** consists of a directory that helps people as well as web crawlers understand the particular section of the webpage they are in.

With the increase of such a threat that increases the vulnerability of humans to exploitation in recent years, there has been a rapid adaptation of technology improvements where users need to know how phishers do it (phishing), and be aware of techniques to protect themselves from being phished.

## **Problem Statement**

In recent years, phishing has become increasingly complex, frequent, and unfortunately successful. It is very difficult to determine if a website URL is actually legitimate or not, hence this project seeks to address fake URLs and domain names through the identification of phishing website links before the user actually clicks on the link for access. This is being designed to allow all users to be able to check for the legitimacy of a URL which will hopefully in turn, reduce security risks to individuals and organizations at large.

## **Proposed Solution**

The expectation is that this project will give us better insight on phishing, i.e how to distinguish phishing websites from legitimate websites by selecting the best algorithm and have it embedded in browsers as an extension that detects the phishing sites. Through this, we will be able to prevent and educate internet users on the deceptive ways of phishers through URLs and thus reduce the rate of financial theft from users and organizations online.

## **Justification of the Study**

With the presence of numerous websites on the World Wide Web, there have been several fraudulent websites that have been developed to resemble reputable websites. The main intention behind the forged websites is to trick the victims by requesting them to submit personal information such as their credit card numbers, passwords, etc., leading to the loss of financial assets from users and organizations that has cost the stakeholders so many financial resources. According to the third Microsoft Computing Safer Index Report, released in February 2014, the annual worldwide impact of phishing could be as high as $5 billion.

With the information above we seek to identify robust countermeasures through Machine Learning and neural networks that can point out phishing sites through their URL detection, thus justifying the need to conduct this study.

## **Specific Objectives**

* To identify phishing sites using a TensorFlow Model
* To Implement feature extraction to detect fake websites
* To identify the trend/similarity between phishing URLs

**Research Questions**

* Can we identify phishing websites to prevent attacks?
* How to apply Machine Learning and necessary classification methods to classify malicious vs legitimate websites?
* Are there components for detection and classification of phishing websites?

**Success Criteria**

Generating a model that will assess URL link and tag it as legitimate or not at 85% accuracy.

## **Project Plan**

Our project will consist of:

* Cross-Industry Standard Process For Data mining*(*CRISP-DM) will be used for conducting this research.
* JIRA Kanban board to manage and track the different tasks involved in this project.
* TensorFlow to view the Neural Network’s creation
* Streamlit for deployment
* A GitHub repository
* Presentation slides for the project

**Scope of the Study**

This study explores machine learning models that use datasets obtained from open source platforms in order to analyze website links and distinguish between phishing and legitimate URL links.

# **DATA UNDERSTANDING**

## **Overview**

In this project, a dataset containing information for Phishing sites was collected from [Kaggle data for Phishing Site URLs.](https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls?select=phishing_site_urls.csv)

## **Exploring Data**

The dataset includes fields that represent all of our primary data related to Phishing Sites.

* Data contains 549,346 entries.
* There are two columns.
* Label column is prediction col which has 2 categories:  
  A. Good - which means the URL does not contain ‘bait’ and therefore is not a Phishing Site.  
  B. Bad - which means the URL contains ‘bait’ therefore is a Phishing Site.
* There is no missing value in the dataset

## **Verifying Data Quality**

The data set does not require much cleaning. Detailed cleaning may be done during data preparation.

# **DATA PREPARATION**

These are the steps followed in preparing the data

#### **Importing Libraries**

All the necessary libraries were imported into the notebook.

#### **Loading Data**

Loaded the dataset from the CSV and then created a python notebook from it.

#### **Cleaning Data**

The data cleaning involved several steps;

* Missing Values - The dataset has no missing values.
* Duplicates - The dataset was found to have 42145 duplicate values which were dropped
* Column names - All the columns are named appropriately and in a homogenous manner

1. **Data Types**

The dataset contains categorical variables: url and labels

#### **Assumptions**

The data provided is correct and up to date.

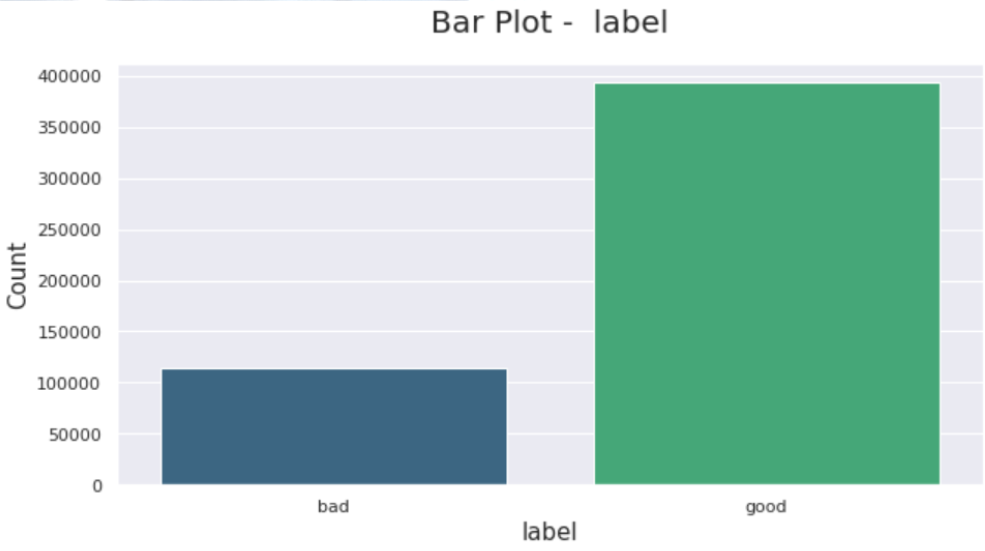
# **DATA ANALYSIS [**[**Link**](https://github.com/stogaja/Moringa-Project-WK5)**]**

# **Categorical Analysis**

The identification of a necessary and applicable model is significant because the model will be needed in the determination of a proper strategy and to assist us to correctly give recommendations and perform implementations. This can only be done when all the information is presented and the data to be used has already been cleaned and prepared. Analysis begins with the importation of libraries.

There are 2 columns in the dataset, URL and Label. URL consists of the List of URLs totalling to 507195 entries after cleaning, while Labels consist of 2 unique values, good and bad.

The good entries within the column consist of approximately, 37,000 URLs while the bad entries consist of 13,000 URLs. Therefore, majority of the sites in the dataset are good, i.e they are safe sites.

****

#### **Numerical Analysis**

There are different kinds of features used in machine learning algorithms in the detection process. The collected features are as below:

* URL-Based Features
* Domain-Based Features
* Page-Based Features
* Content-Based Features

We will primarily focus on URL based features in this project.

**URL- Based Features**

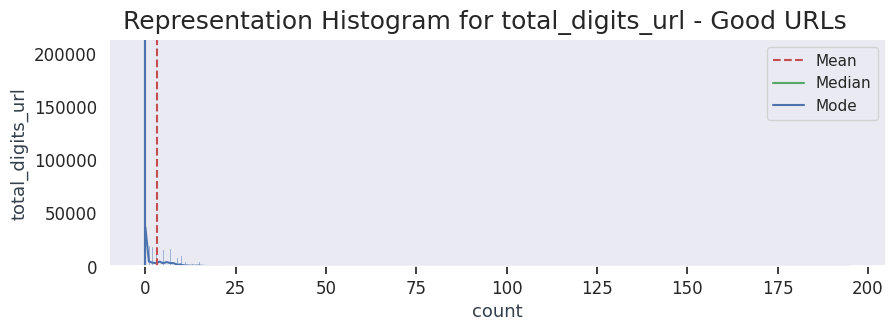
To analyze a website, to decide whether the site is a phishing site or not, the URL is the first thing to analyze. URLs of phishing domains have distinctive points. The features related to these points are obtained during the processing of the URL.

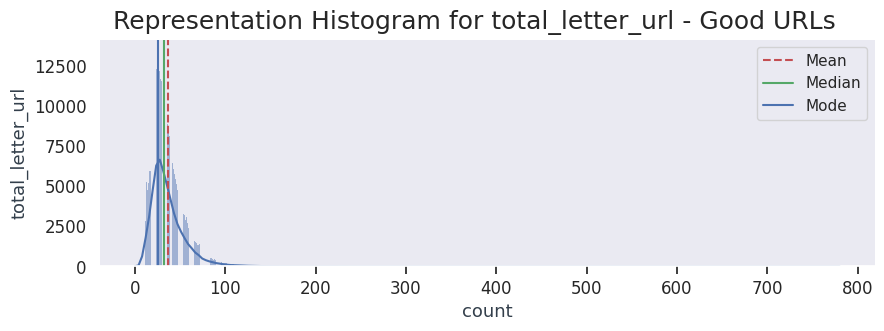
Some of the URL-Based Features are given below:

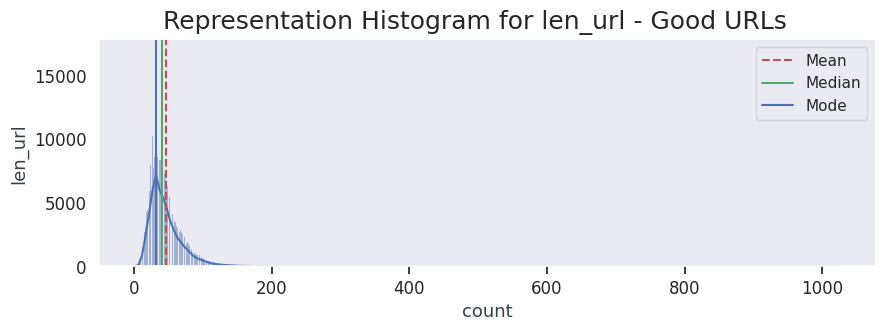
* Digit count in the URL
* Total length of URL
* Checking whether the URL is Typosquatted or not. (google.com → goggle.com)
* Checking whether it includes a legitimate brand name or not (apple-icloud-login.com)
* Number of subdomains in URL
* Is Top Level Domain (TLD) one of the commonly used ones?

In our project, there were defined functions that count the total digits, letters, and length of the domain, path and urls of the urls. This has been depicted in the histogram below:

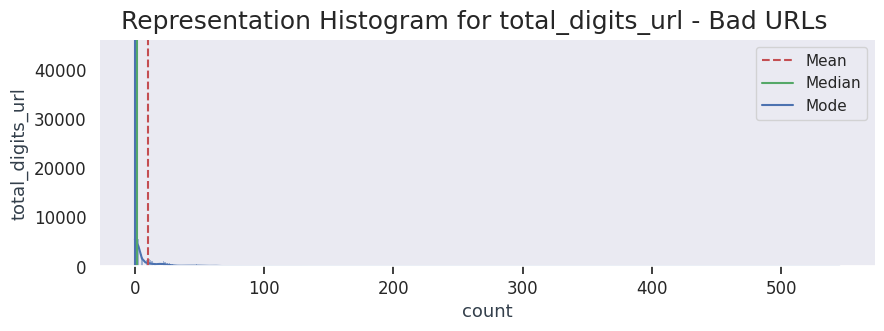
**Good URLs:**

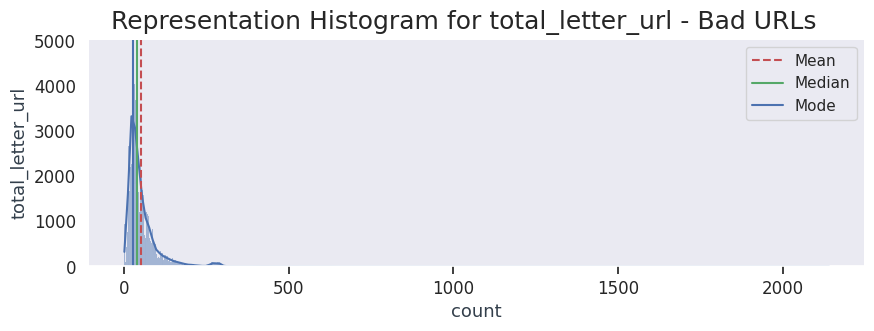


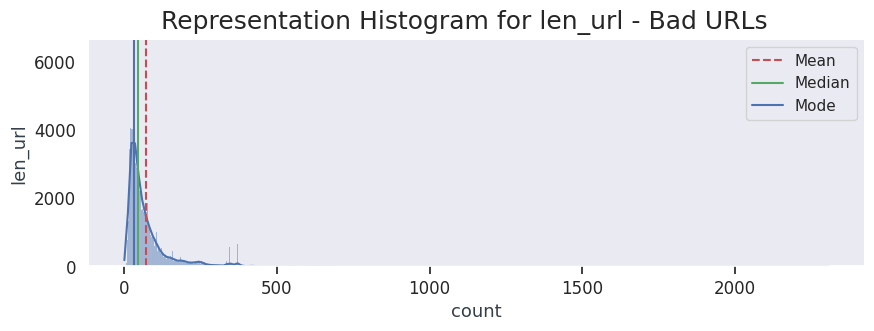




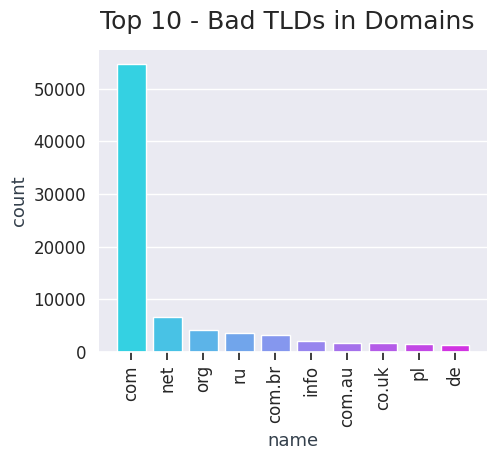
**Bad URLs:**





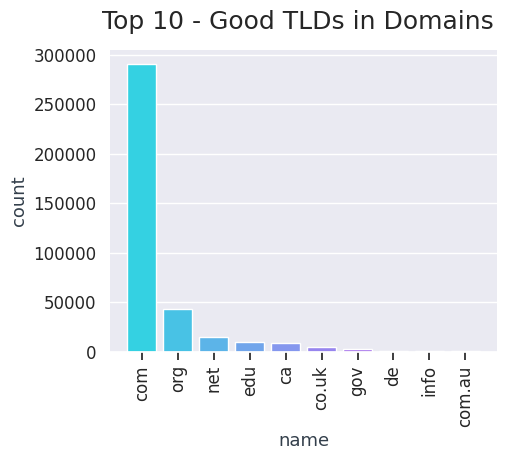


Some analysis was conducted on the top-level domain of the domain name to find out the most common to least common top level domains in both the good and bad URLs.



The top 3 URLs used for bad domains are:

* .com at 70,000 URLs
* .net at 6000 URLs
* .org at 4500 URLs



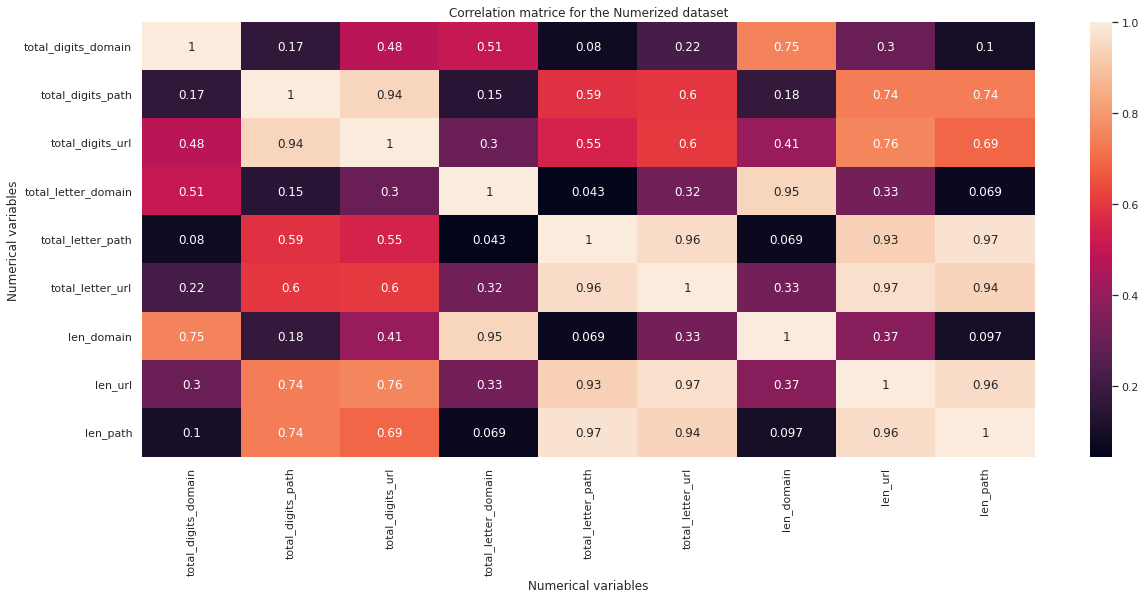
The top 3 URLs used for good domains are:

* .com at 280,000 URLs
* .org at 48000 URLs
* .net at 20000 URLs

#### Correlation Matrix

We used a correlation matrix to help us display the correlation coefficients for different variables as well as summarize the dataset we currently have, as well as to identify and visualize patterns in the given data.

From the correlation matrix, we were able to plot a correlation heatmap that gave us a visual interpretation of the correlation matrix.



**Reading a Correlation Matrix**

* -1 indicates a perfectly negative linear correlation between two variables
* 0 indicates no linear correlation between two variables
* 1 indicates a perfectly positive linear correlation between two variables

The correlation matrix and correlation heatmap above show that:

* *“Total\_letter\_domain”* and “len\_domain”, *“total\_letter\_path”* and “*len\_path”*, *“total\_letter\_path”* and *“total\_letter\_url”* are strongly correlated.
* “*Total\_digits\_domain*” and *“total\_letter\_path”*, *“total\_letter\_domain”* and *“len\_path”* are strongly negatively correlated.

#### **Multicollinearity check**

Multicollinearity occurs when there are two or more independent variables in a multiple regression model, which have a high correlation among themselves. When some features are highly correlated, we might have difficulty in distinguishing between their individual effects on the dependent variable. Multicollinearity can be detected using various techniques, one such technique being the Variance Inflation Factor(VIF).

In the VIF method, we pick each feature and regress it against all of the other features. For each regression, the factor is calculated as :

VIF=\frac{1}{1-R^2}

Where, R-squared is the coefficient of determination in linear regression. Its value lies between 0 and 1.

As we see from the formula, greater the value of R-squared, greater is the VIF. Hence, greater VIF denotes greater correlation. This is in agreement with the fact that a higher R-squared value denotes a stronger collinearity. Generally, a VIF above 5 indicates a high multicollinearity.

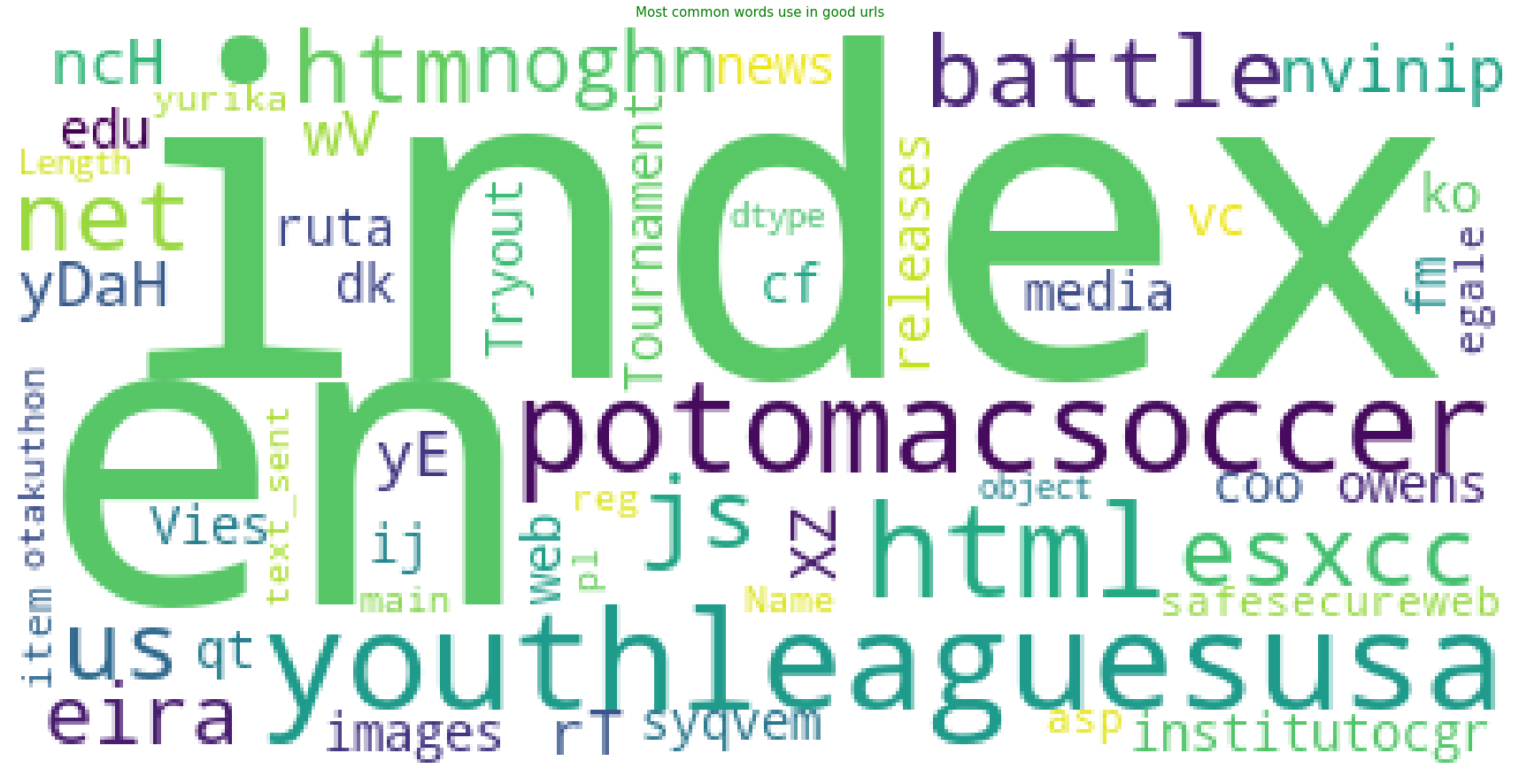
From our dataset, the features are highly collinear hence the reduction techniques should be applied when using ML models. For this model, Tensor flow with vectorized independent variables will be used.

#### **Visualization**

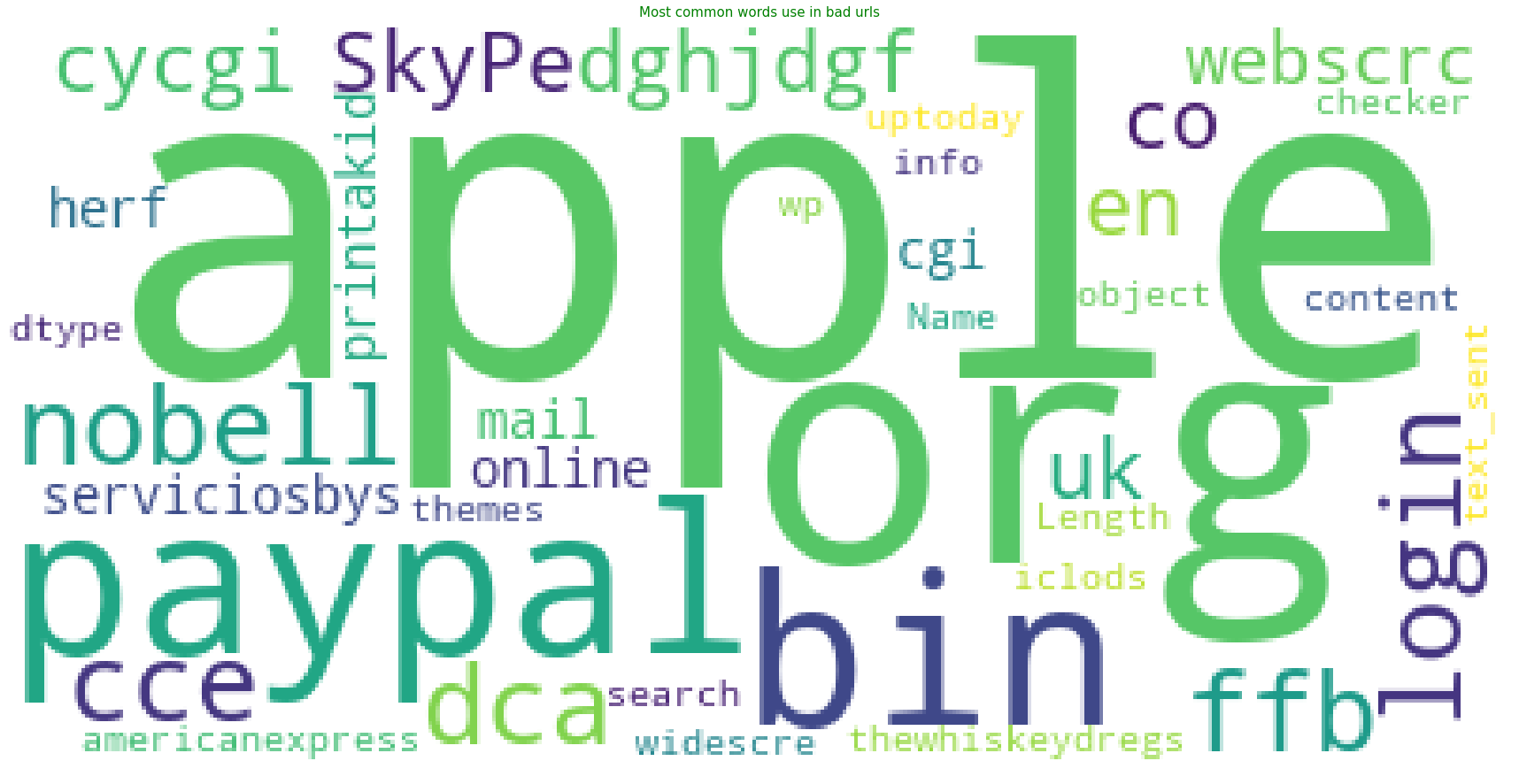
A tokenizer splits a string using a regular expression, which matches either the tokens or the separators between tokens. We, therefore, tokenized the entire data set.

##### **Visualizing some important keys**

The entire dataset was sliced and categorized into good and bad sites and a function was created to visualize the important keys from the URLs using word count.

The most common words used in good URLs are:

The most common words used in bad URLs are :



##### **Visualize internal links**

First, we set up the Chrome webdriver, which is used for automating the test of webapps across many browsers. It provides capabilities for navigation to web pages. User input and more so we can scrape dynamic web pages.

After the set up the Chrome driver then creates two lists:

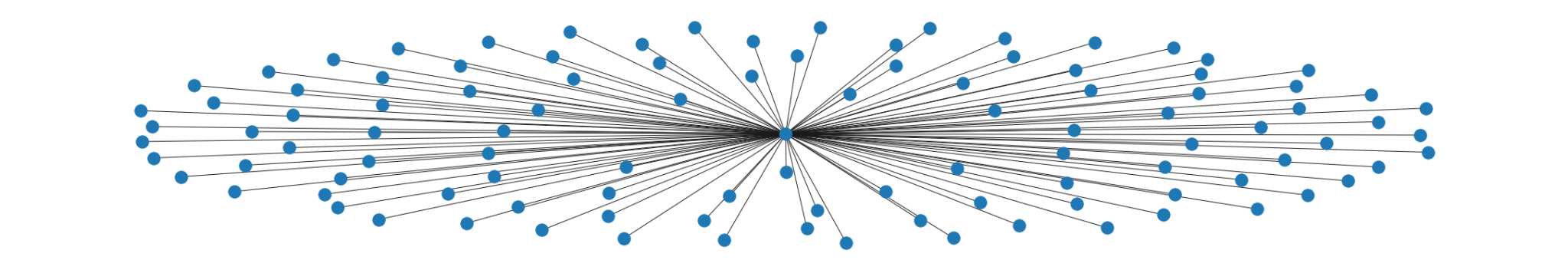
* First list: Named list\_urls holds all the pages you’d like to scrape.
* Second list: Creates an empty list where you’ll append links from each page.

BeautifulSoup Library is used to extract relevant hyperlinks for Google, i.e. links only with <a> tags with href attributes. It’s also used for getting data out of HTML, XML and other markup languages.

After getting the list of websites with hyperlinks, they are turned into a Pandas DataFrame with two columns;

* **From :** URL where the link resides
* **To :** Link destination of URL

Finally, using the DataFrame mentioned above to visualize an internal link structure by feeding it to the Networkx method from the from\_pandas\_edgelist first and it’s drawn by calling nx.draw.



#### **Modeling**

For Modeling, we used TensorFlow, which is an open source library for numerical computation and large-scale machine learning. It bundles a slew of both machine learning and deep learning (Neural Networks) models and algorithms and makes them useful.

For a start, we note that we have a huge dataset and the class is imbalance, with our dataset having around 130,000 bad links and approximately 270,000 good links.

To balance this out, we will use SMOTE (Synthetic Minority Oversampling Technique), which has the minority class over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors

**Step 1: Text Tokenization**

First step was to tokenize the texts as we mentioned before. The Tokenizer function will be used for that. By default, it removed all the punctuations and set the texts into space-separated organized forms. Each word became an integer by the tokenizer function. The value of oov\_token was set to be 'OOV' meaning unknown words were replaced by oov\_token.

**Step 2: Split data into Test and Train set**

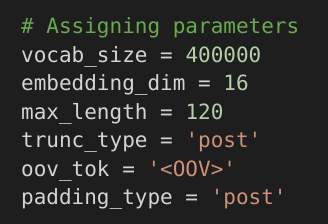
Then 80% of the data was set out for training and 20% was set for testing purposes.

**Step 3: Changing data into string values.**

Data was changed to string values to avoid there being an error, in case of there being data not in string format.

**Step 4: Set parameters**

Parameters were then assigned as shown:



* vocab \_size are the unique words to train the network
* Embedding\_dimension = dimensional vector representing each word
* Max\_length = the length of each review.
* Trunc\_type = ‘post’ meaning the data will be truncated once past the max\_length.
* Padding \_type=’post’ means padding will be applied at the end.

**Step 5: Tokenizing the text data and converting sentences**

Texts data was tokenized and the review sentences were converted into a sequence of words words and the pads if necessary

**Step 6: Building the model**

* The first layer is the embedding layer and has been defined and explained above
* The second layer is ‘GlobalAveragePooling1D()’ which flattens the vector. Originally the data is three-dimensional (batch\_size x steps x features). GlobalAveragePooling1D makes it (batch\_size x features).
* The third layer is a Dense layer called the hidden layer where a ‘relu’ activation function is used, neurons are also used here.
* The last layer uses the sigmoid activation function or logistic function.

**Step 7: Compiling the model**

We chose to use binary\_crossentropy as a loss function, since this is a probabilistic loss function. The optimizer used was “adam’.

**Step 8: Converting labels to an array**

The training and testing sets are converted from labels to an array.

**Step 9: Using SMOTE to deal with class imbalance**

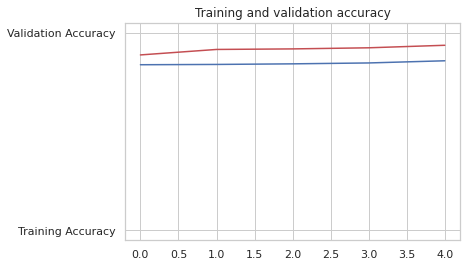
To sort out the class imbalance in our dataset, we used SMOTE, to oversample the train dataset. The dataset is then counted before and after using SMOTE and we note that the data is at 50% good URLs and 50% bad URLs.

**Step 10: Training the dataset**

The data set was trained using time.perf\_counter() which returns a float value of time in seconds.

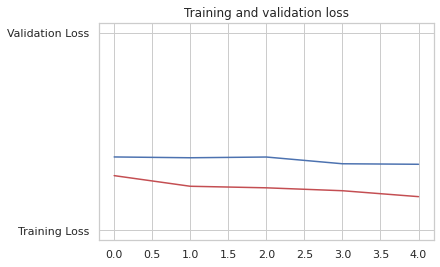
Num\_epochs which indicates the number of times in which the model would be trained.

There was a plot of accuracy:



From the above, the model could probably be trained a little more as the trend for accuracy on both datasets is still rising for the last few epochs. The model has not yet over-learned the training dataset, showing comparable skill on both datasets.

From the plot of loss, we note that the model has comparable performance on both train and validation datasets (red). Once these parallel plots depart consistently, it might be a sign to stop training at an earlier epoch.



Following above steps, our model was also evaluated using a confusion matrix and it garnered an **accuracy of 85.82%**

#### **Implementing the Solution**

With the trained model, we prompted the user to input a URL they had and our model would detect whether the link is a legitimate link or not.

If legitimate, the user would receive confirmation that the link is secure and they can proceed to explore the content within the link.

Otherwise, the user would receive a warning that the URL could probably be a phishing URL and accompanied with the alert is a set of recommendations for the user to read through. The recommendations are several ways in which the user can protect the computer from intrusion;

* Keep Your Firewall Turned On: A firewall helps protect your computer from hackers who might try to gain access to crash it, delete information, or even steal passwords or other sensitive information. Software firewalls are widely recommended for single computers. The software is prepackaged on some operating systems or can be purchased for individual computers. For multiple networked computers, hardware routers typically provide firewall protection.
* Install or Update Your Antivirus Software: Antivirus software is designed to prevent malicious software programs from embedding on your computer. If it detects malicious code, like a virus or a worm, it works to disarm or remove it. Viruses can infect computers without users’ knowledge. Most types of antivirus software can be set up to update automatically.
* Install or Update Your Antispyware Technology: Spyware is just what it sounds like—software that is surreptitiously installed on your computer to let others peer into your activities on the computer. Some spyware collects information about you without your consent or produces unwanted pop-up ads on your web browser. Some operating systems offer free spyware protection, and inexpensive software is readily available for download on the Internet or at your local computer store. Be wary of ads on the Internet offering downloadable antispyware—in some cases these products may be fake and may actually contain spyware or other malicious code. It’s like buying groceries — shop where you trust.
* Keep Your Operating System Up to Date: Computer operating systems are periodically updated to stay in tune with technology requirements and to fix security holes. Be sure to install the updates to ensure your computer has the latest protection.
* Be Careful What You Download: Carelessly downloading email attachments can circumvent even the most vigilant anti-virus software. Never open an e-mail attachment from someone you don’t know, and be wary of forwarded attachments from people you do know. They may have unwittingly advanced malicious code.
* Turn Off Your Computer: With the growth of high-speed Internet connections, many opt to leave their computers on and ready for action. The downside is that being “always on” renders computers more susceptible. Beyond firewall protection, which is designed to fend off unwanted attacks, turning the computer off effectively severs an attacker’s connection—be it spyware or a botnet that employs your computer’s resources to reach out to other unwitting users.

# **CONCLUSION**

Based on the data analysis the following conclusions were deduced:

The model developed detects if a URL link is phishing or legitimate by using machine learning models and deep neural network algorithms.

The feature extraction and the models used on the dataset helped to uniquely identify phishing URLs and also the performance accuracy of the models used.

# 

# **RECOMMENDATIONS**

There are existing websites that provide phishing detection currently,

* However, this project can be taken further by creating a browser extension that can be installed on any web browser to detect phishing URL Links. If the site is a phishing URL, browsing is blocked and the user is notified that the site is phishing.
* Removal of posts with links to phishing websites on social networks, forums or blogs. In the same way that some web content is processed for previews, it could be processed to detect malicious websites.