PhishGuard: Advanced Analysis & Prediction of Malicious URLs Using Unsupervised & Supervised Machine Learning

Abstract

Preventing phishing attacks can improve cybersecurity for companies and keep user or corporate data safe from hackers. This research investigates if there are characteristics of uniform resource locators (URLs) that distinguish legitimate websites from phishing websites designed to steal user credentials, and if these features can be used to predict a URL is for a phishing website quickly in an application, like email software. Unsupervised learning was performed on features of the URLs to find patterns, and learning algorithms like Decision Trees, Random Forests, and Support Vector Machines were used to determine how well phishing URLs could be predicted and what variables were most important for those predictions. The results showed that the variables about URL length, number of digits, number of letters, number of special characters like ampersands described the variance in the URLs the most and contributed the most to predicting if URLs were legitimate or not. Specifically, longer URLs with more complex characters tend to be phishing URLs. In addition, the Random Forest model was able to predict phishing URLs with 91% accuracy.

Introduction

Most systems and companies today gather their user's personal information. It is important to try to protect this information from cybersecurity attacks. One common attack method is to steal a user's username and password through a phishing website that pretends to be a legitimate website[1]. It would be beneficial to be able to detect a phishing URL for a user before they click on a link just by examining the features of the URL. The first goal was to discover any patterns in the URL dataset and try to examine the variability between the URLs in general. So, PCA was performed and the influence of the top 5 variables was examined on the first two

principal components. Next, clustering was used to examine the groups that were occurring in the dataset and what variables formed these groups. Once the dataset was examined, the next question to address was if it is possible to predict phishing URLs just on features about the URL. Linear logistic regression was used to set the base accuracy value to judge the other learning algorithm models. A decision tree, random forest, boosting, and support vector machine was built and tested to see how well the model could detect phishing URLs. Finally, variable importance was explored to gain a better understanding of how the models were differentiating between legitimate and phishing URLs.

The URL dataset was extracted from Seattle University's Microsoft 365 Security Center threat detection report that identifies phishing emails and the URLs in those messages[2]. The URLs from the emails labeled as phishing threats were gathered from that report for a 48 hour period. The safe URLs were gathered when no threat was detected for the same 48 hour period. Duplicate URLs were removed, which resulted in 3218 legitimate links and 1986 phishing links being collected. Features about the URLs were generated afterward, based on similar techniques discussed in the PhiUSIIL Phishing URL study[3]. This resulted in features that included the URL, domain, URL length, domain length, number of digits, number of letters, number of equal signs and ampersands, and the number of special characters not specifically counted. Then, variables were generated stating if the URL supported HTTPS, and stating the top level domain (TLD) value. The TLD is the highest domain hierarchy value, it is usually .com, .gov, .edu, etc. These features were used to perform unsupervised learning on the dataset to learn more about how websites could be grouped together and in the prediction models to identify phishing URLs.

Background

Principal Component Analysis

In Principal Component Analysis (PCA), the goal is to represent the data in low dimensional form by including the information as much as possible. Each of the components Z_j that represents the original data vectors $X_i, \ldots, X_p$ is a normalized linear combination of them[4]:

$$Z_j = \phi_{1j} X_1 + \phi_{2j} X_2 + \cdots + \phi_{pj} X_p$$

The optimization problem that PCA solves is iteratively finding components that maximize the variance of the linear combination of $X_i, \ldots, X_p$ that are uncorrelated to the components that come before it. As it happens, constraining each Z_j to be uncorrelated to each Z_{j-1} is equivalent to constraining ϕ_j to be orthogonal to ϕ_{j-1}.

In PCA, each component signals the direction that captures maximal variance. A principal component may explain a proportion of the total variance, and following the first principal component all proportion of variance explained (PVE) will monotonically decrease until the accumulation of them sum up to 1. The variance explained by the mth component is as shown as below[4]:

$$\frac{1}{n} \sum_{i=1}^{n} Z_{im}^2 = \frac{1}{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2$$

and it will often be divided by the total variance to show the PVE of each component. Overall, the principal components let us summarize the dataset with a condensed down set of variables..

Clustering

Under the categorization of unsupervised learning, two approaches that may compliment each other are PCA and Clustering. However, clustering is different from PCA in that PCA aims to find low dimensional representation of the original dataset; whereas clustering is looking for

groupings of the data examples that represent a homogeneous subgraph. In clustering, the two most well known methods are K-Means and Hierarchical Clustering. The K-Means clustering tries to minimize the within cluster variation[4]:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \epsilon C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

by subdividing the observations into K clusters. Each iteration of the algorithm shifts the cluster assignments so the centroids' positions continue to change. Until the K-Means algorithm converges, the within cluster variation will continue to be minimized, which can reveal similarities and patterns in the dataset.

Decision Trees

Decision trees models split the data into segments based on the best predictor at each step, so that each step will lead to the most likely outcome[4]. The models produced by decision trees can be represented in a visual graph of the decisions or choices that led to the predicted outcomes at each node. This is appropriate for the purposes of finding the most influential variables along with the values that split the decisions, so inferences about the data can be made from the model. Binary classification trees and regression trees can be tuned based on the number of terminal nodes, which can be considered the tree size. So, the tree with different sizes is compared against its other sizes. This is done by using the error rate or mean square error for each sized tree and comparing it using K-fold cross validation that breaks the training data into K segments to validate the models[4]. Then, the tree can be pruned using a compromise of complexity and error rate to pick a model that is easy to understand and doesn't overfit the training data.

Random Forests

Random forests create multiple trees to create the best decision tree. However, at each split, only a sample of the predictors are used, so the primary predictor isn't always chosen, and

secondary predictors can contribute more to the model[4]. Ignoring some variables when making the split ensures the strongest parameter doesn't dominate the decisions. Random forests can be tuned based on the number of trees generated. However, random forests can also be tuned by the number of predictors considered at each split, which we will call m. So, we can create models with different numbers of m and then measure their error against each other for best performance.

Support Vector Machines

There are various concepts in introducing a Support Vector Machine. First of all, in most of the literature about Support Vector Machines, the task that is at the focal point is mainly the classification task between two different classes (This is indeed what is being the main task for this project). To find a model that predicts a dataset in a p-dimensional feature space between two different classes, each of the observations in the p-dimensional feature space is being colored to one of two colored classes. The main way to classify the two classes is to use a hyperplane that is of dimension p-1 to split the p-dimensional feature space to two different sides. A separating hyperplane has the property that:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$$

By using the above, the correct class observations should be on each side of the separating hyperplane.

Another important topic in SVM (Support Vector Machines) is the concept of margin. With the margin M is being maximized, each observation is being ensured to have been classified correctly. However, the Maximal Margin Classifier does not cover the cases of non linearly separable dataset or provides the possibility of using a non-linear separating hyperplane. The weakness of such a classifier is that this overly constrained assumption that the dataset is linearly separable may lead to overfitting when the feature dimension p is large, and the inability to find a

hyperplane that can truly linearly separate the dataset. Therefore, certain measures are taken to tackle the difficulties in using a Maximal Margin Classifier.

In order to train SVM on linearly non-separable datasets, the concepts of soft-margin and non-linear kernels are introduced. The kernel considered in this project particularly is the rbf kernel, used to measure how distance between the observations have effect on the final decision boundary.

Gradient Boosting

"The gradient boosting technique consists of three simple steps:

- An initial model F0 is defined to predict the target variable y. This model will be associated with a residual (y – F0)
- A new model h1 is fit to the residuals from the previous step
- Now, F0 and h1 are combined to give F1, the boosted version of F0. The mean squared error from F1 will be lower than that from F0:

$$F_1(x) \leftarrow F_0(x) + h_1(x)$$

For performance of F1, we could model after the residuals of F1 and create a new model F2:

$$F_2(x) \leftarrow F_1(x) + h_2(x)$$

This can be done for *'m'* iterations, until residuals have been minimized as much as possible:

$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x)$$

Here, the additive learners do not disturb the functions created in the previous steps. Instead, they impart information of their own to bring down the errors."[5]

Methodology

Data Processing

The original SU URL dataset just consisted of the URL strings. The characters in the strings were parsed using a Python script to count the number of digits, letters, question marks, equal signs, ampersands, and other special character values. These generated fields were checked for any missing values and there were no missing values found in them. The script appropriately labeled any counts as zero when there weren't any specific character types found. In addition, the dataset was checked to see if scaling was necessary before performing principal component analysis. The mean and variance for the variables were calculated and examined to ensure that a specific field with large values wouldn't skew the results of the PCA. The fields representing the url length and domain length were much larger than some of the other variables. Based on this information, the URL dataset was scaled before being analyzed. The dataset was not scaled before creating the decision tree model, so inferences could be made on the actual values.

Computations

All qualitative columns and non-continuous numerical columns were dropped. Then, PCA was calculated on the remaining fields in the dataset. The results from PCA were used to determine what fields were contributing to the most variance in the dataset and what fields had the most influence on the principal components. K-means clustering was used to break the dataset into groups of 2 groups to identify patterns that may be existing in the dataset. To judge the performance of the prediction models, linear logistic regression was first calculated as a baseline accuracy score. A decision tree was calculated and pruned using cross validation to determine how the model was interpreting the variables in the dataset. Next, Random Forests, Boosting, and Support Vector Machine models were created and their parameters were tuned in order to predict if a URL was phishing or not. In addition, these models were created to see how

high our accuracy could get. To calculate the accuracy of the models, the dataset was split into 70% training data, and 30% testing data. The accuracy was based on how well the model's predicted labels matched the testing dataset labels.

Results

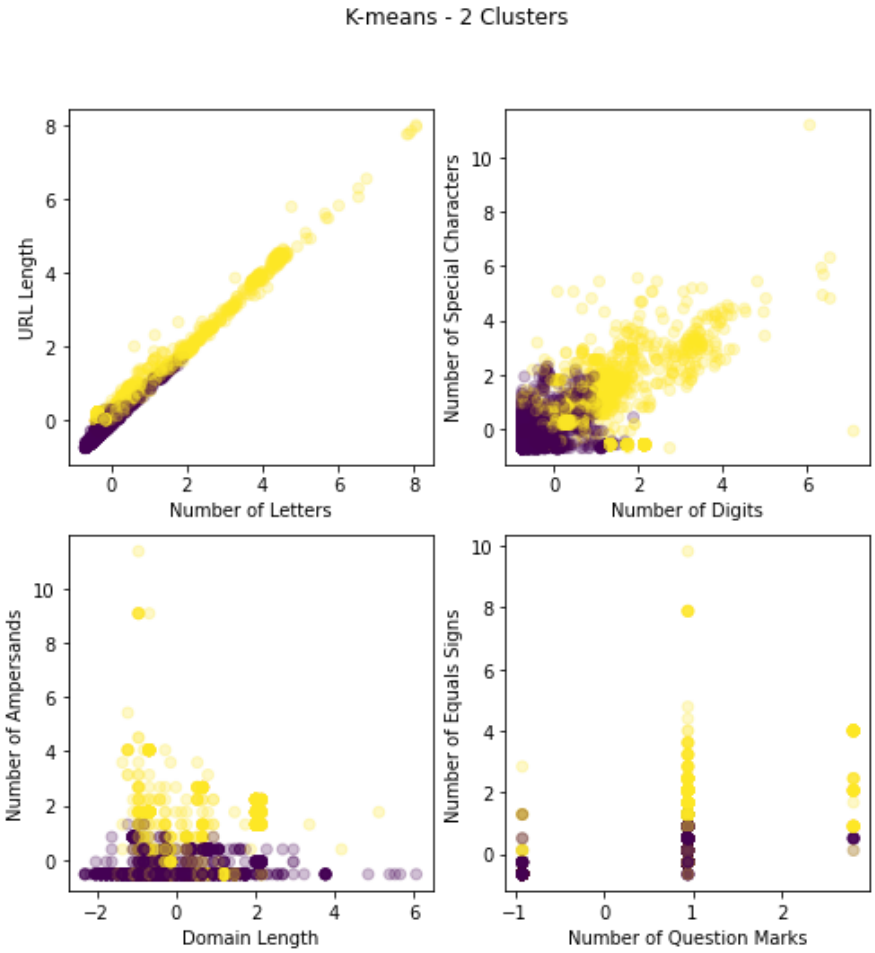
➢ Principal Component Analysis

- Top 5 most influential variables for principal component 1 and principal component 2 are shown in the table below. The top most influential variable is the number of digits, URL length, and number of letters for PC1. For PC2, the most influential variables or special characters. Many of the most impactful variables for the principal components have to deal with the number of character types in the URL.

PC1		**PC2**	
Top Features	**Vector Values**	**Top Features**	**Vector Values**
NoOfDegitsInURL	0.461515	NoOfEqualsInURL	-0.515345
URLLength	0.434051	NoOfAmpersandInURL	-0.513835
NoOfLettersInURL	0.394067	NoOfLettersInURL	0.383566
NoOfOtherSpecialCharsInURL	0.386374	NoOfQMarkInURL	-0.381205
NoOfEqualsInURL	0.301401	URLLength	0.314049

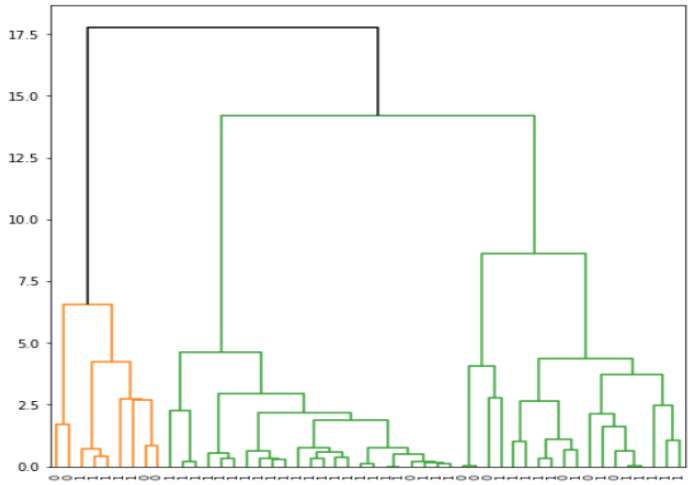
➢ K-Means Clustering

- Below plot shows K-means clusters for two clusters based on useful feature combination pairs. The 2 clusters created by K-means are color encoded, one is yellow and the other is dark blue to show how the URLs are being divided. The dataset is scaled, so that is why there are some negative values.



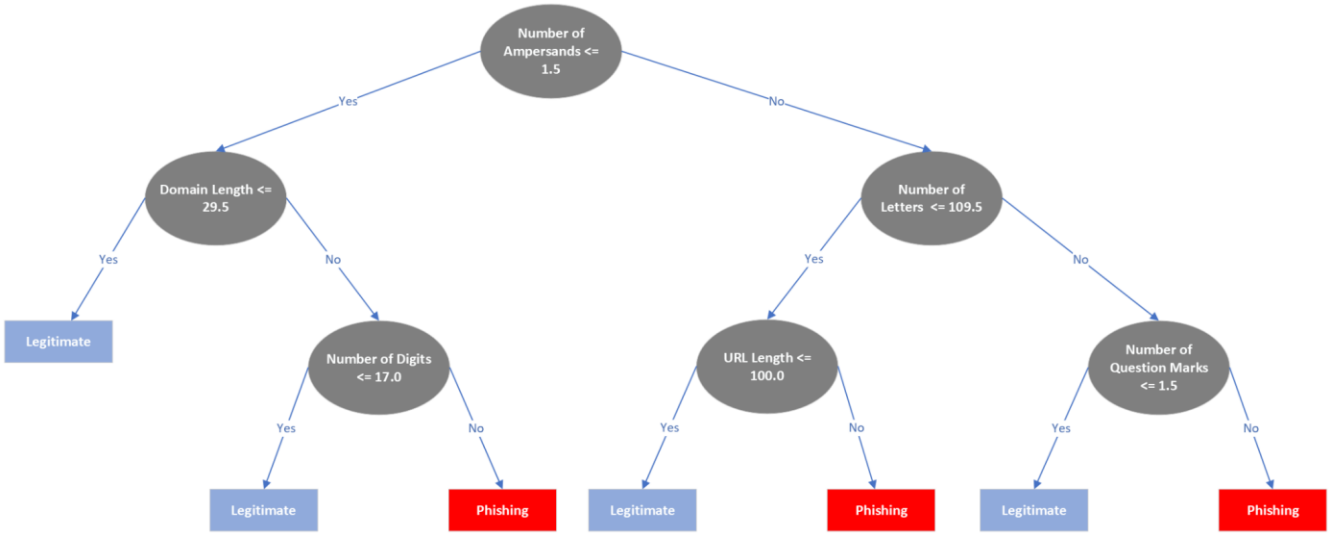
➢ Hierarchical clustering

- The plot below shows the Hierarchical clusters for two groups with ward linkage. The clusters seem to be well separated revealing legitimate and phishing URLs.



➢ Decision Tree

- The decision tree below outlines the most important variables for defining whether a URL is phishing or legitimate.

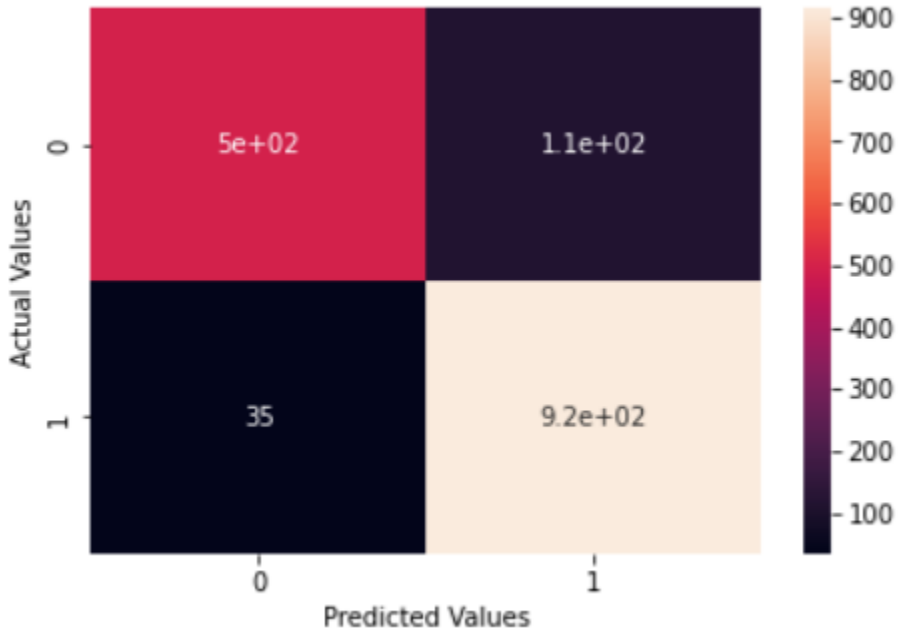


➢ Random Forests

- The cross validation results: maximum features is 4 and number of trees is 200
- Variable Importance:

	importance
URLLength	0.179561
DomainLength	0.165935
NoOfLettersInURL	0.161780
NoOfDegitsInURL	0.145390
NoOfEqualsInURL	0.095231
NoOfOtherSpecialCharsInURL	0.091227
NoOfAmpersandInURL	0.071683
NoOfQMarkInURL	0.062616
IsHTTPs_True	0.026576

- Confusion Matrix: Model predicted 899 safe urls correctly out of 950, while predicted 504 safe urls correctly out of 612

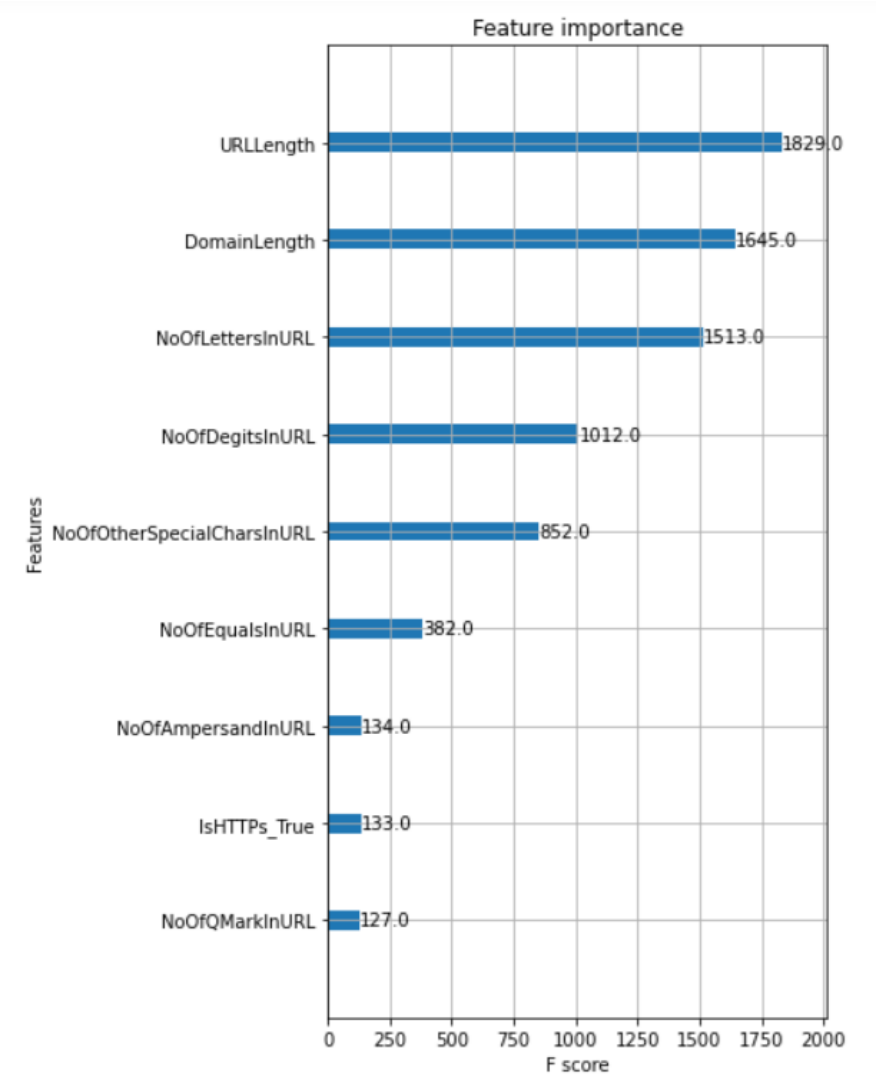


- Accuracy: 90%

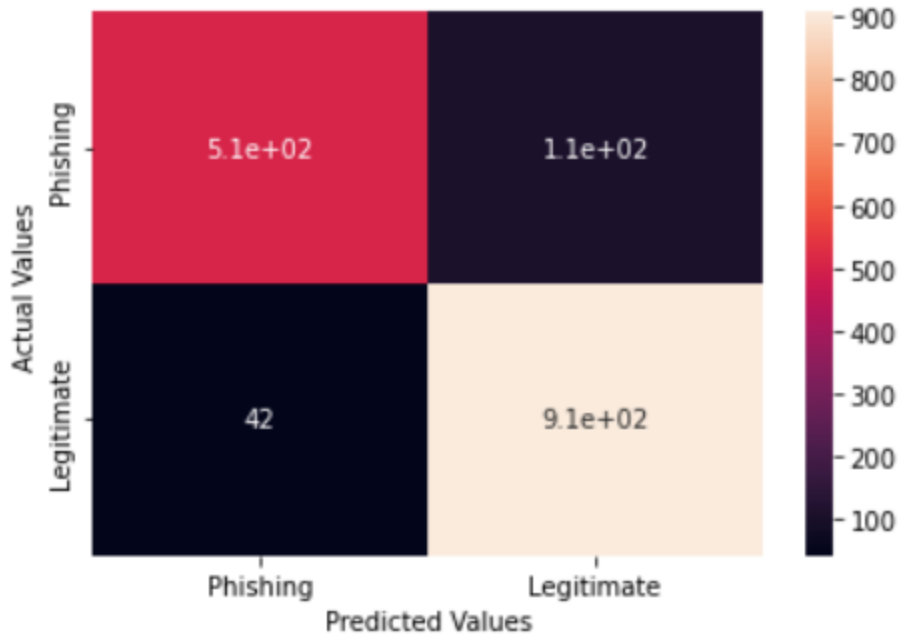
➢ XGB Classifier

- Cross validation results: colsample_bytree: 0.8589674088633774, gamma: 0.4258361373682714, Learning rate: 0.08014470711610436, Maximum depth: 7, No. of trees: 595, subsample: 0.7030892181592572

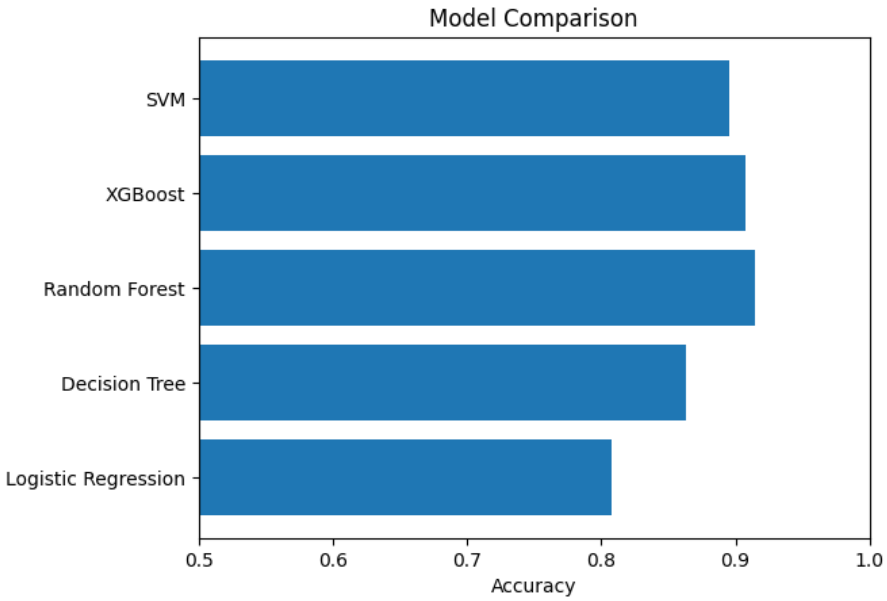
- Variable Importance:



- Confusion Matrix: Model predicted 897 safe urls correctly out of 950, while predicted 510 safe urls correctly out of 612



- Accuracy: 90%
    - ➢ Model Accuracy
- The model accuracy for each learning algorithm was calculated on the test set. The results are shown in the table below, the Random Forest model had the highest accuracy with 90.5%.



Discussion

The principal components were calculated using PCA and the dataset points were plotted on the first two principal component axes. In addition, the loading vector values of the top 5 variables of principal component 1 and 2 were examined, and the 2 cluster graphs on useful features were also plotted. These charts indicated that the dataset consisted of two groups, long complex URLs with multiple special characters, and shorter simpler URLs that were more structured. In addition, the difference in the principal component variable values for PC1 described longer URLs with number of letters and digits and URL length being the highest contributors. For PC2, the values for equal signs, ampersands, and question marks were all negative, so more special characters led to a reduced score, but URL length and number of letters were positive. Again, indicating that the variance in the dataset is described by the size of the URL and then by its use of special characters.

Multiple learning algorithms were used to determine if it is possible to reasonably predict whether a URL is phishing or legitimate. The variance in the data has proved to be significant in predicting whether the urls are phishing. With the two most flexible models in tree ensembles able to get above the accuracy of 90%. However, model flexibility can also wear on the accuracy of the results. Comparing between the random forest and the boosting classifier, the averaged prediction provided by the random forest although is preferred for its lower variance and higher bias comparing to boosting classifier; however, in the current dataset, the random forest's classification accuracy is higher than boosting. The SVM classifier was also able to capture non-linear relationships between the labels and features and improve in accuracy on the test set; this means that there is an underlying nonlinear pattern in the dataset improving the results over linear logistic regression..

When examining the variable importance and decision tree diagram, there are structures shared by phishing URLs and structures shared by legitimate URLs, which the models relied upon to make the predictions. In the decision tree, the number of ampersands is the most important variable, more than 1 tends to indicate a phishing URL. Then, the number of letters, if it is lower compared to the URL length, then it is more likely to be phishing. The algorithms used length and character type counts to make the determinations. Thus, confirms the patterns seen in the unsupervised learning section that phishing urls are usually more complex and have longer characters, while legitimate URLs are shorter and are less complex. Plus, if legitimate URLs are long, then they appear to contain less special characters and have a more standard structure with more letters and digits. Overall, it is possible to predict with moderately high accuracy whether or not a URL is a phishing URL. Plus, the calculations were on simple characteristics of the URLs that could be quickly calculated in email software. Therefore, it could potentially be used to label URLs as suspicious to users for emails not caught by the spam filters.

As for limitations, the dataset was built on labels provided by Microsoft's threat detection report. So, the accuracy of the phishing label that was assigned to the URLs is dependent on the accuracy of Microsoft's threat detection tools and logic. It could be possible that some phishing URLs were missed. Also, the report only pulled data from a 48 hour period, so it may not contain a completely diverse collection of URLs. Furthermore, some phishing URLs contain links to Google Docs or SharePoint sites. These websites are normally legitimate, it is just the content in that particular Google or Microsoft form that is malicious. However, in this case the phishing label for malicious Google docs cannot be based on URL features alone, this would require examining the email text and sender information to improve accuracy.

Conclusions

Understanding similar characteristics in phishing websites could contribute to efforts in preventing phishing attacks. Using unsupervised learning on the phishing dataset did identify the

websites that could be grouped into sites with lengthy complicated URLs and sites with smaller human readable URLs. The models showed that URLs for phishing attacks could be identified with high accuracy of 90%. This means that some email software could potentially be enhanced with features to notify a user if a link looks suspicious. So, if an account within the organization gets compromised and starts sending phishing emails, instead of relying on the users' skills to be able to catch it, they would be assisted by this feature before clicking the link.

Citations

[1] National Cyber Security Centre. (2018) *Phishing: How to Recognise and Avoid Phishing Attacks*. NCSC, https://www.ncsc.gov.uk/guidance/phishing.

[2] Microsoft. *About Threat Explorer and Real-Time detections in Microsoft Defender for Office 365*. https://learn.microsoft.com/en-us/defender-office-365/threat-explorer-real-time-detections-about

[3] Prasad,Arvind and Chandra,Shalini. (2024). PhiUSIIL Phishing URL (Website). UCI Machine Learning Repository. https://doi.org/10.1016/j.cose.2023.103545.

[4] James, G., Hastie, T., Tibshirani, R. and D. Witten. *An Introduction to Statistical Learning with Applications in R.* 2nd ed., Springer 2023.

[5] Analytics Vidhya, Introduction to XGBoost Algorithm in Machine Learning

https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behindxgboost/#:~:text=XGBoost%20Classifier%20is%20a%20gradient,used%20for%20structured%20data%20tasks.

In [ ]:
```
# Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.datasets import get_rdataset
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression, LassoCV, LogisticRegression
from sklearn.preprocessing import LabelBinarizer, StandardScaler
from sklearn.metrics import mean_absolute_error, accuracy_score, confusion_matrix, classification_report
from ISLP import load_data
from sklearn.cluster import \
     (KMeans,
      AgglomerativeClustering)
from scipy.cluster.hierarchy import \
     (dendrogram,
      cut_tree)
from ISLP.cluster import compute_linkage

np.random.seed(2)
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree, DecisionTreeRegressor
```

Import Datasets

In [ ]:
```
# Load the New dataset
new_safe_urls = pd.read_csv("C:\\Users\\david\\Downloads\\safeURLs.csv")
```

In [ ]:
```
# New phishing dataset requires extra processing
import csv

def process_urls(input_csv):
    # Read the input CSV file
    with open(input_csv, mode='r', newline='', encoding='utf-8') as infile:
        reader = csv.reader(infile)
        next(reader)  # Skip the header row
        urls = [row[0] for row in reader]

    # Split URLs by pipe and remove duplicates
    unique_urls = set()
    for url in urls:
        parts = url.split('|')
        for part in parts:
            cleaned_url = part.strip()
            if cleaned_url:
                unique_urls.add(cleaned_url)

    return list(unique_urls)

new_phish_urls = process_urls("C:\\Users\\david\\Downloads\\phishingURLs.csv")
```

In [ ]:
```
import re
from urllib.parse import urlparse

def analyze_url(url, label):
    # Parse the URL
    parsed_url = urlparse(url)

    # Calculate length
    url_length = len(url)

    # Count letters, digits, and special characters
    num_letters = sum(c.isalpha() for c in url)
    num_digits = sum(c.isdigit() for c in url)
    num_equals = url.count('=')
    num_question_marks = url.count('?')
    num_ampersands = url.count('&')

    # Count periods, ignoring the first two
    num_periods = max(0, url.count('.') - 2)
```

```
    # Count special characters excluding slashes and periods
    num_special_chars = sum(not c.isalnum() and c not in ('/', '.') for c in url) - (num_equals + num_question_m

    # Extract domain and TLD
    domain = parsed_url.netloc
    tld = domain.split('.')[-1] if '.' in domain else ''
    domain_length = len(domain)

    # Check if URL uses HTTPS
    has_https = parsed_url.scheme == 'https'

    # Populate dictionary
    url_analysis = {
        'URL': url,
        'URLLength': url_length,
        'NoOfLettersInURL': num_letters,
        'NoOfDegitsInURL': num_digits,
        'Domain': domain,
        'TLD': tld,
        'DomainLength': domain_length,
        'NoOfEqualsInURL': num_equals,
        'NoOfQMarkInURL': num_question_marks,
        'NoOfAmpersandInURL': num_ampersands,
        'NoOfOtherSpecialCharsInURL': num_special_chars,
        'IsHTTPs': has_https,
        'label': label
    }

    return url_analysis

results = []

# process safe urls
for index, row in new_safe_urls.iterrows():
    url_analysis = analyze_url(row['URL'], 1)
    results.append(url_analysis)

print(len(results))

# process phishing urls
for url in new_phish_urls:
    url_analysis = analyze_url(url, 0)
    results.append(url_analysis)

print(len(results))

url_df = pd.DataFrame(results)
```

```
3218
5204
```

1.1 Check for Missing Values

In [ ]:

```
url_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5204 entries, 0 to 5203
Data columns (total 13 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   URL                          5204 non-null   object
 1   URLLength                    5204 non-null   int64
 2   NoOfLettersInURL             5204 non-null   int64
 3   NoOfDegitsInURL              5204 non-null   int64
 4   Domain                       5204 non-null   object
 5   TLD                          5204 non-null   object
 6   DomainLength                 5204 non-null   int64
 7   NoOfEqualsInURL              5204 non-null   int64
 8   NoOfQMarkInURL               5204 non-null   int64
 9   NoOfAmpersandInURL           5204 non-null   int64
 10  NoOfOtherSpecialCharsInURL   5204 non-null   int64
 11  IsHTTPs                      5204 non-null   bool
 12  label                        5204 non-null   int64
dtypes: bool(1), int64(9), object(3)
memory usage: 493.1+ KB
```

There appears to be no missing values in the dataset. This because all values were calculated by a script and give 0 when a character isn't found.

1.2 Check the Mean and Variance for Scaling

In [ ]:
```
# Set index equal to the domain field
url_df.set_index('URL', inplace=True)

# Drop text fields and boolean fields
url_data = url_df.drop(['Domain', 'TLD', 'IsHTTPs'],axis = 1)
```

In [ ]:
```
url_data.head()
```

Out[ ]:

URL	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEquals
https://docs.google.com/forms/d/1hM9Hkb1jP3rqqJ8o2-HTolSKRqGbqlGNcFfBK0S3RYo/prefill	84	66	8	15	
https://seattleu.zoom.us/j/95478319044	38	20	11	16	
https://seattleu.zoom.us/my/earthmonth	38	31	0	16	
https://www.seattleu.edu/cejs/campus-sustainability/what-su-is-doing/climate-action-plan/	89	73	0	16	
https://www.seattleu.edu/staff-council/meetings/	48	39	0	16	

In [ ]:
```
# Check the Mean
url_data.mean()
```

Out[ ]:
```
URLLength                    159.626826
NoOfLettersInURL             113.398155
NoOfDegitsInURL               29.563605
DomainLength                  21.039201
NoOfEqualsInURL                1.696387
NoOfQMarkInURL                 0.492890
NoOfAmpersandInURL             1.165450
NoOfOtherSpecialCharsInURL     6.665642
label                          0.618370
dtype: float64
```

In [ ]:
```
# Check the Variance
url_data.var()
```

Out[ ]:
```
URLLength                    35095.559734
NoOfLettersInURL             22039.684225
NoOfDegitsInURL               1559.368431
DomainLength                    55.057467
NoOfEqualsInURL                  6.635459
NoOfQMarkInURL                   0.287284
NoOfAmpersandInURL               4.767739
NoOfOtherSpecialCharsInURL      63.466119
label                            0.236034
dtype: float64
```

The columns for letters in the URL and URL length are quite large compared to the other values, so it is probably best to scale the dataset.

1.3 Scaling the Data

In [ ]:
```
# Drop true / false values
X = url_data.drop('label',axis = 1)

# Scale the dataset
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X),columns=X.columns,index=X.index)
```

In [ ]:
```
X_scaled.head()
```

Out[ ]:

URL	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEquals
https://docs.google.com/forms/d/1hM9Hkb1jP3rqqJ8o2-HToISKRqGbqlGNcFfBK0S3RYo/prefill	-0.403730	-0.319301	-0.546121	-0.813979	-0.(
https://seattleu.zoom.us/j/95478319044	-0.649299	-0.629184	-0.470143	-0.679196	-0.(
https://seattleu.zoom.us/my/earthmonth	-0.649299	-0.555081	-0.748729	-0.679196	-0.(
https://www.seattleu.edu/cejs/campus-sustainability/what-su-is-doing/climate-action-plan/	-0.377038	-0.272145	-0.748729	-0.679196	-0.(
https://www.seattleu.edu/staff-council/meetings/	-0.595915	-0.501189	-0.748729	-0.679196	-0.(

In [ ]:
```
# Check the Mean
X_scaled.mean()
```

Out[ ]:
```
URLLength                   -4.272567e-16
NoOfLettersInURL             3.472114e-16
NoOfDegitsInURL              4.817387e-15
DomainLength                -1.715000e-15
NoOfEqualsInURL              1.423124e-14
NoOfQMarkInURL              -2.835195e-14
NoOfAmpersandInURL          -2.136881e-14
NoOfOtherSpecialCharsInURL   4.660953e-15
dtype: float64
```

In [ ]:
```
# Check the Mean
X_scaled.var()
```

Out[ ]:
```
URLLength                    1.000192
NoOfLettersInURL             1.000192
NoOfDegitsInURL              1.000192
DomainLength                 1.000192
NoOfEqualsInURL              1.000192
NoOfQMarkInURL               1.000192
NoOfAmpersandInURL           1.000192
NoOfOtherSpecialCharsInURL   1.000192
dtype: float64
```

2.0 PCA

In [ ]:
```
# Perform PCA
pca = PCA()
```

```
pca_out = pca.fit_transform(X_scaled)
```

2.1 PCA Principal Components

In [ ]:
```
pd.DataFrame({'Center': scaler.mean_
                  , 'Scale': scaler.scale_}
                 , index=X_scaled.columns)
```

Out[ ]:

	Center	Scale
URLLength	159.626826	187.320089
NoOfLettersInURL	113.398155	148.443420
NoOfDegitsInURL	29.563605	39.485045
DomainLength	21.039201	7.419359
NoOfEqualsInURL	1.696387	2.575691
NoOfQMarkInURL	0.492890	0.535937
NoOfAmpersandInURL	1.165450	2.183306
NoOfOtherSpecialCharsInURL	6.665642	7.965797

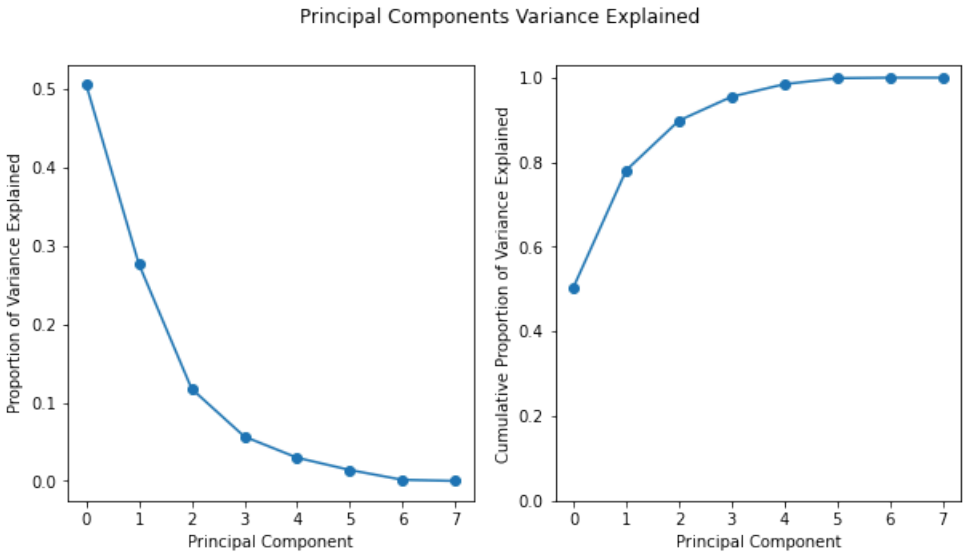
In [ ]:
```
print("Number of Principal Components:", pca.n_components_)
```

```
Number of Principal Components: 8
```

2.1 Plot the Principal Component Explained Variance

In [ ]:
```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

# Plot of proportion of variance explained
ax[0].plot(pca.explained_variance_ratio_, marker='o')
ax[0].set_xlabel('Principal Component')
ax[0].set_ylabel('Proportion of Variance Explained')

# Plot of cumulative proportion of variance explained
ax[1].plot(np.cumsum(pca.explained_variance_ratio_), marker='o')
ax[1].set_xlabel('Principal Component')
ax[1].set_ylabel('Cumulative Proportion of Variance Explained')
ax[1].set_ylim(0, 1.03)
fig.suptitle("Principal Components Variance Explained")
```

Out[ ]:
```
Text(0.5, 0.98, 'Principal Components Variance Explained')
```



2.2 PCA Examine the Loading Vector Values

In [ ]:
```
pc2_df = pd.DataFrame(pca.components_[:2].T,
            index=X_scaled.columns,
            columns=['PC1', 'PC2'])

# Top 5 PC1 variables
top_5_pc1 = pc2_df.loc[pc2_df['PC1'].abs().sort_values(ascending=False).index].head(5)
# Top 5 PC2 variables
top_5_pc2 = pc2_df.loc[pc2_df['PC2'].abs().sort_values(ascending=False).index].head(5)

print(top_5_pc1['PC1'])
print(top_5_pc2['PC2'])
```

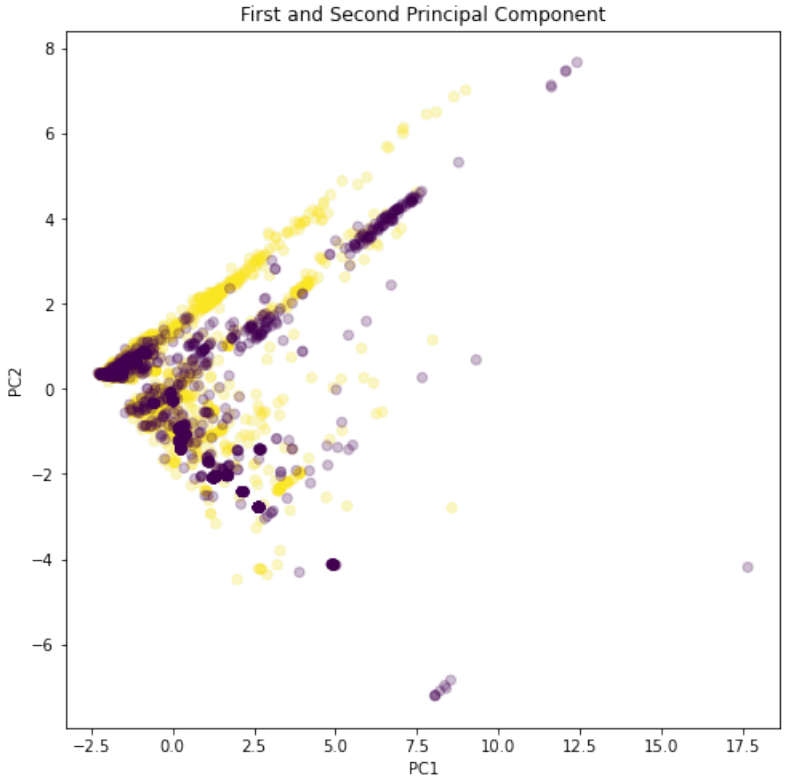
```
NoOfDegitsInURL               0.461515
URLLength                     0.434051
NoOfLettersInURL              0.394067
NoOfOtherSpecialCharsInURL    0.386374
NoOfEqualsInURL               0.301401
Name: PC1, dtype: float64
NoOfEqualsInURL      -0.515345
NoOfAmpersandInURL   -0.513835
NoOfLettersInURL      0.383566
NoOfQMarkInURL       -0.381205
URLLength             0.314049
Name: PC2, dtype: float64
```

Looking at the loading vectors, we can see that URLs with higher values of PC1 are longer and contain more digits, letters, and special characters. The URLs seem to be separated between long complex URLs and short simple URLs, with more complexity having a higher PC1 value. Then, URLs with higher PC2 scores have more letters and are longer, but have fewer special characters like equals signs, ampersands, and question marks.

2.3 1st and 2nd Principal Component Plot

In [ ]:
```
fig , ax = plt.subplots(1, 1, figsize=(8,8))

ax.scatter(pca_out[:,0], pca_out[:,1], alpha=0.25, c=url_data['label'])

ax.set_xlabel("PC1")
ax.set_ylabel("PC2")
ax.set_title("First and Second Principal Component")
```

Out[ ]:
```
Text(0.5, 1.0, 'First and Second Principal Component')
```



Phishing URLs might have lower PC2 scores because they often use more special characters (equals signs, ampersands, question marks) to encode information, track user data, or redirect to different pages. They might also be shorter but dense with these characters.

Legitimate URLs might have moderate to higher PC2 scores, reflecting a balanced use of letters and fewer special characters. They are often well-structured and easier to read.

Phishing URLs might exhibit higher PC1 scores due to their tendency to be overly complex. They often include many digits, letters, and special characters to obfuscate their true nature and appear more legitimate.

3.0 SVD

In [ ]:
```
# Perform SVD
U, s, V = np.linalg.svd(X_scaled, full_matrices=False)
```

In [ ]:
```
s.shape
```

Out[ ]:
```
(8,)
```

3.0.1 SVD Interpretation on U and V*

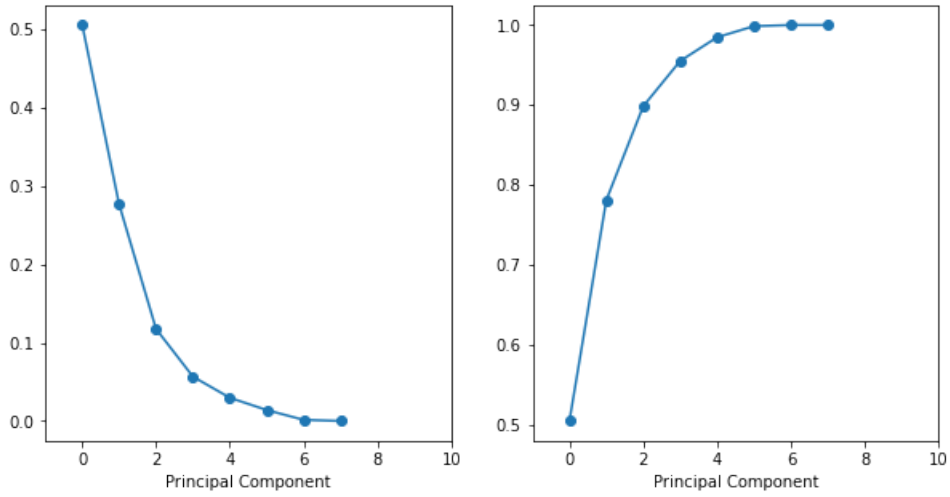
1. Since we are performing SVD on the scaled data of the original dataset, it is equivalent to performing PCA on the original dataset
2. The right singular vectors are the principal component axis.
3. The left singular vectors are the principal component scores divided by the singular values.

3.1 SVD Principal Components

In [ ]:
```
# Find the explained variance
explained_variance_ratio = (s ** 2) / np.sum(s ** 2)

fig, ax = plt.subplots(1, 2, figsize=(10, 5))

# Plot of singular values
ax[0].plot(explained_variance_ratio, marker='o')
```

```
ax[0].set_xlim(-1, 10)
ax[0].set_xlabel('Principal Component')
ax[1].plot(np.cumsum(explained_variance_ratio), marker='o')
ax[1].set_xlim(-1, 10)
ax[1].set_xlabel('Principal Component')
```

Out[ ]:
```
Text(0.5, 0, 'Principal Component')
```



We need at 4 principal components to explain over 90% of the variance on our dataset.

In [ ]:
```
U.shape
```

Out[ ]:
```
(5204, 8)
```

4.0 Kmeans

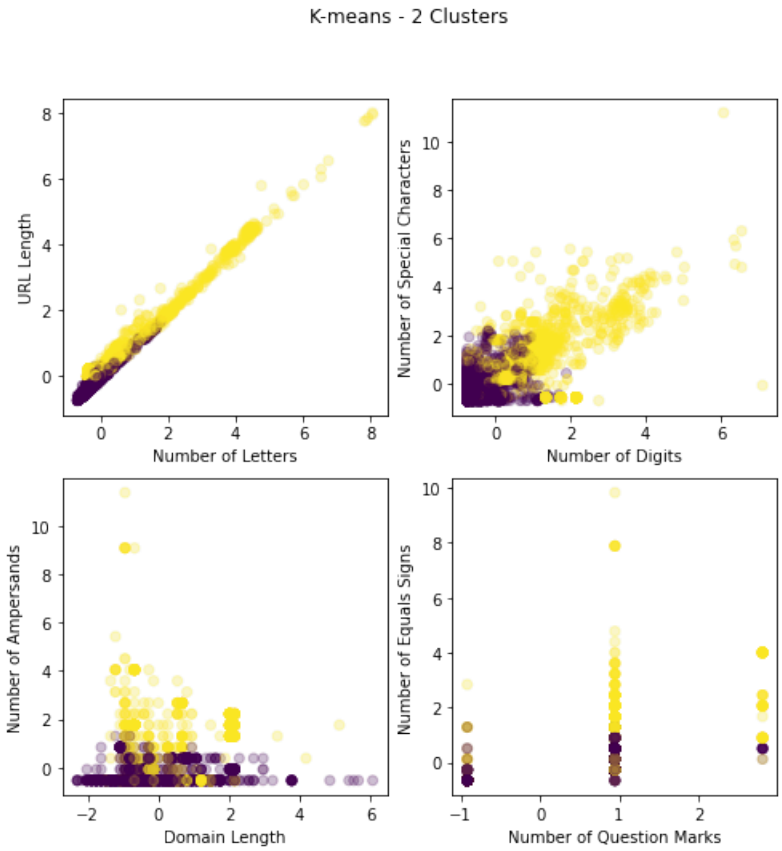
4.1 Kmeans 2 Clusters

4.1.1 Perform Kmeans

In [ ]:
```
kmeans = KMeans(n_clusters=2,
                random_state=2,
                n_init=20).fit(X_scaled)
```

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(8,8))

ax[0,0].scatter(X_scaled['NoOfLettersInURL'], X_scaled['URLLength'],
                c=kmeans.labels_, alpha=0.25)
ax[0,0].set_xlabel('Number of Letters')
ax[0,0].set_ylabel('URL Length')

ax[0,1].scatter(X_scaled['NoOfDegitsInURL'], X_scaled['NoOfOtherSpecialCharsInURL'],
                c=kmeans.labels_, alpha=0.25)
ax[0,1].set_xlabel('Number of Digits')
ax[0,1].set_ylabel('Number of Special Characters')

ax[1,0].scatter(X_scaled['DomainLength'], X_scaled['NoOfAmpersandInURL'],
                c=kmeans.labels_, alpha=0.25)
ax[1,0].set_xlabel('Domain Length')
ax[1,0].set_ylabel('Number of Ampersands')

ax[1,1].scatter(X_scaled['NoOfQMarkInURL'], X_scaled['NoOfEqualsInURL'],
                c=kmeans.labels_, alpha=0.25)
ax[1,1].set_xlabel('Number of Question Marks')
ax[1,1].set_ylabel('Number of Equals Signs')

fig.suptitle("K-means - 2 Clusters")
```

Out[ ]:
```
Text(0.5, 0.98, 'K-means - 2 Clusters')
```



4.1.2 Examine the 2 cluster items

In [ ]:
```
# Get examples of URLs from the two different clusters
zero_indexes = np.where(kmeans.labels_ == 0)[0]
one_indexes = np.where(kmeans.labels_ == 1)[0]

group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
print(group0_random_indexes)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
print(group1_random_indexes)
```

```
[1785 3043  112 4897 2673]
[2885 3676 4664 5059   95]
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

	https://my.usc
	https://click.linl qs=793db0cd0cefa631181e806b053fad89d751aa8c0ae4955f910ed47c677e28d8d354d20bd5d71f041bee8f32cb79f224326f20534eadce9ce78d
	https://publichealthinsider.com/2022/02/16/vaccination-verification-policy-to-end-a
	https://url.us.m.mimecastprotect.com/s/w9EeCjRNMAhR5x4RfWk6fC?domain=linkprotect.cudasvc.com#emhhbmdhQHNlYXR0bG
	https://www.justice.gov/crt/volunteer-and-paid-stude

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

Out[ ]:

https://u23540048.ct.sendgrid.net/ls/click?upn=u001.I3F-2F7WcQv1b6XTqsDw3pqlt-2Fwp08lstKkfHCWOCY-2F8AfOehBjBSOuY 2BBQvLxXMCNOJFJ1drwz0ei4QRlg7fsdhcMp7Za6gdzVioUv4uF1LHlSRv0AAUEScpbArWciXlpkdj2XmCd5DtsXn83RKVZf4xatdkXsRyhZTGi6bCp 2BOYPj3Abl8loi89PZyixjXi4HyLUiYFgiZoSSkIX4YwHAlWH-2BZ4T39-2BSXLcob0dupKxVAYq2omkOn2g-Z 2FjKAy0MUDF2LYcGgXlzw1hUsfJHEcO7vDRumMQWzFdfwBMpleN9KOrLPmy8OWdLe 2FO2g9scleBObWeTHdMVSwbttGhtisE5nCaTXPCOIpRnIHSR6zQN0-2BzUSe6j74FbZidMbYjTvrR5fcTR7iFb-2BlEc3FwJ-2BoVD53 2BNK4w1EnFeDzrIcGtBnzH0E-2BP2lBnN9U8bxizA3Kr-2BQ7HrzOf8ddXwQJOrGzNqOBwhjvRnKVZKOpkJF1EhqLn3-2B3nv8BXXNV6BdekJSVh 2FqH98ImmmIaGLt3jRvwICkJ-2BLg9vn2q3T 2BcOSumUUSUNbfaXKkYi6Xa1GlVWcpe1hJYLVa1w8Gf9UA9eb7kscgexCkVJo8s7P1fNj18nb430lmljQprxbdTveYNjZ-2F-2F8BUm0rJQf0wuJMAI
https://link.mediaoutreach.meltwater.com/ls/click?upn=RlCwVYFJgVP13fjU-2Btu-2FvrvWRai7McfaP-2BmOnb4AzSE0jh7 2FMwm6cPAOyZ8uUp-2BvWa-2FGjAeH2O1gpr2h0tg6eIsMKYdM-3D2xUv_YoOSKr1NP1W8kCUEqdLsS6BO3DKa49mFN8VSLDHpcw-Z 2BL6dLk97AOWGWciHMtHIrHb-2BS0I-2FCnE7rZQtimRYe8MWzZcwzSVEdRG8vxSfhrnknII8BAn4KKnYK 2FtA27CqVoFBYERPHwS0ErWqNGBPAjkWvyL-2FkU3jovr2X7PHqRD6fQzauC78dSfbiocdsZwlr0d-2B2-2FyXUH9B 2FXCurtx4Nvqw9AmbdTmxLWNSh3v3V2gD6hyoYZmtWq9sNYEAht-2F-2BRjh4l57Lwn79-2FVK5rsHE6sqTSavfazOr 2BSTKlc2p7pzxeO31fuVlUYRR9lUgDdhhNntE9

Looking at the two groups, we see short URLs with less ampersands, and long URLs with a lot more special characters.

4.1.3 Compare Kmean Results to the Actual Grouping

In [ ]:
```
def getAccuracy(cfMatrix):
    accuracy = (cfMatrix[0][0]+cfMatrix[1][1])/(cfMatrix[0][0]+cfMatrix[0][1]+cfMatrix[1][0]+cfMatrix[1][1])
    return accuracy
```

In [ ]:
```
# flip the label from 0 to 1, to get the correct sign
url_data['label'] = 1-url_data['label']

# create confusion matrix
confusion_matrix = pd.crosstab(index=kmeans.labels_, columns=url_data['label'], rownames=[''])
print(confusion_matrix)
acc = getAccuracy(confusion_matrix)
print("Kmeans 2 cluster Accuracy: {:.2f}%".format(acc*100))
```

```
label     0     1

0      2741  1084
1       477   902
Kmeans 2 cluster Accuracy: 70.00%
```

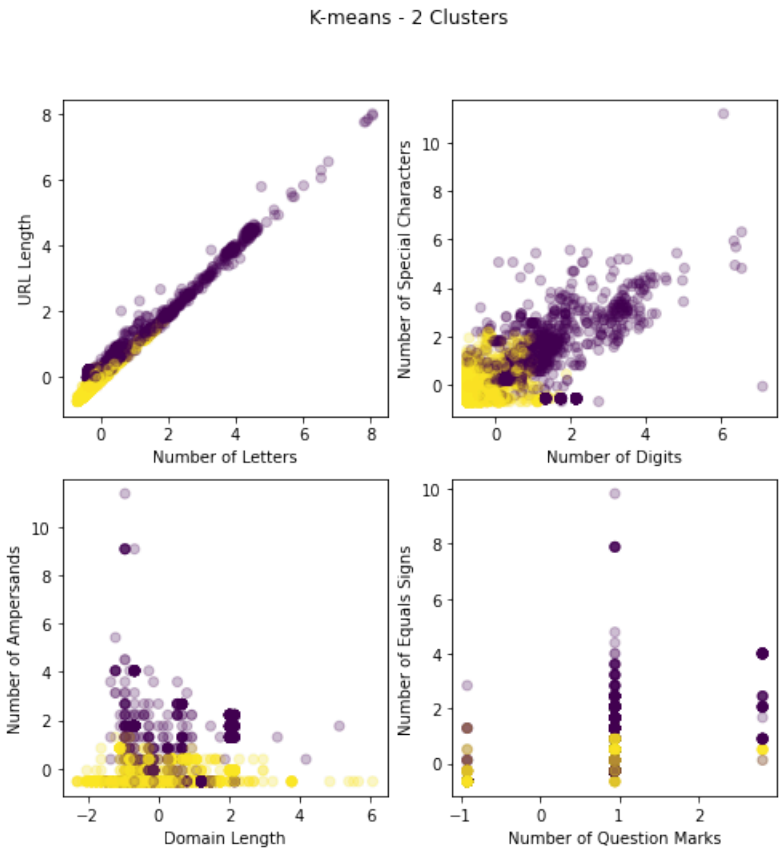
4.1.4 Kmeans on 18 Principal Components

In [ ]:
```
# Number of principal components
r = 4

singular_vector_space = U[:,0:r] * s[0:r]

kmeans2 = KMeans(n_clusters=2,
                 random_state=2,
                 n_init=20).fit(singular_vector_space)
```

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(8,8))

ax[0,0].scatter(X_scaled['NoOfLettersInURL'], X_scaled['URLLength'],
                c=kmeans2.labels_, alpha=0.25)
ax[0,0].set_xlabel('Number of Letters')
ax[0,0].set_ylabel('URL Length')

ax[0,1].scatter(X_scaled['NoOfDegitsInURL'], X_scaled['NoOfOtherSpecialCharsInURL'],
                c=kmeans2.labels_, alpha=0.25)
```

```
ax[0,1].set_xlabel('Number of Digits')
ax[0,1].set_ylabel('Number of Special Characters')

ax[1,0].scatter(X_scaled['DomainLength'], X_scaled['NoOfAmpersandInURL'],
                c=kmeans2.labels_, alpha=0.25)
ax[1,0].set_xlabel('Domain Length')
ax[1,0].set_ylabel('Number of Ampersands')

ax[1,1].scatter(X_scaled['NoOfQMarkInURL'], X_scaled['NoOfEqualsInURL'],
                c=kmeans2.labels_, alpha=0.25)
ax[1,1].set_xlabel('Number of Question Marks')
ax[1,1].set_ylabel('Number of Equals Signs')

fig.suptitle("K-means - 2 Clusters")
```

Out[ ]:
```
Text(0.5, 0.98, 'K-means - 2 Clusters')
```



In [ ]:
```
# flip the label from 0 to 1, to get the correct sign
url_data['label'] = 1-url_data['label']

# create confusion matrix
confusion_matrix = pd.crosstab(index=kmeans2.labels_, columns=url_data['label'], rownames=[''])
print(confusion_matrix)
acc = getAccuracy(confusion_matrix)
print("Kmeans 2 cluster Accuracy: {:.2f}%".format(acc*100))
```

```
label     0     1

0       902   477
1      1084  2741
Kmeans 2 cluster Accuracy: 70.00%
```

4.2 Kmeans 3 clusters

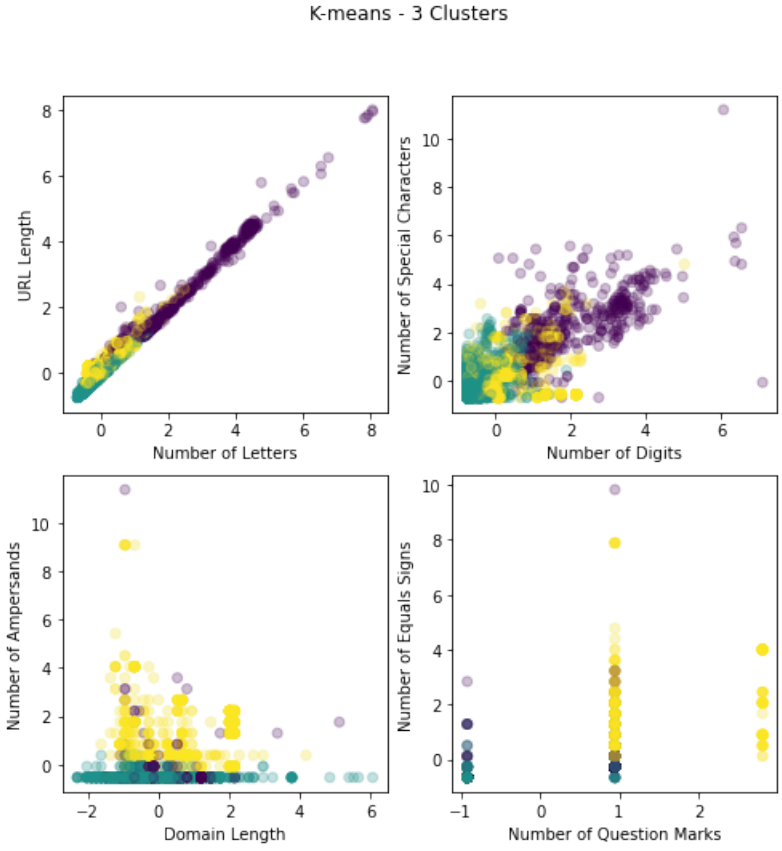
4.2.1 Calculate Kmeans for 3 clusters

In [ ]:
```
kmeans3 = KMeans(n_clusters=3,
                 random_state=2,
                 n_init=20).fit(X_scaled)
```

4.2.2 Plot the 3 clusters on some variables

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(8,8))

ax[0,0].scatter(X_scaled['NoOfLettersInURL'], X_scaled['URLLength'],
                c=kmeans3.labels_, alpha=0.25)
ax[0,0].set_xlabel('Number of Letters')
ax[0,0].set_ylabel('URL Length')

ax[0,1].scatter(X_scaled['NoOfDegitsInURL'], X_scaled['NoOfOtherSpecialCharsInURL'],
                c=kmeans3.labels_, alpha=0.25)
ax[0,1].set_xlabel('Number of Digits')
ax[0,1].set_ylabel('Number of Special Characters')

ax[1,0].scatter(X_scaled['DomainLength'], X_scaled['NoOfAmpersandInURL'],
                c=kmeans3.labels_, alpha=0.25)
ax[1,0].set_xlabel('Domain Length')
ax[1,0].set_ylabel('Number of Ampersands')

ax[1,1].scatter(X_scaled['NoOfQMarkInURL'], X_scaled['NoOfEqualsInURL'],
                c=kmeans3.labels_, alpha=0.25)
ax[1,1].set_xlabel('Number of Question Marks')
ax[1,1].set_ylabel('Number of Equals Signs')

fig.suptitle("K-means - 3 Clusters")
```

Out[ ]:
```
Text(0.5, 0.98, 'K-means - 3 Clusters')
```



4.2.3 Examine differences between the 3 groups

In [ ]:
```
# Get examples of URLs from the three different clusters
zero_indexes = np.where(kmeans3.labels_ == 0)[0]
one_indexes = np.where(kmeans3.labels_ == 1)[0]
two_indexes = np.where(kmeans3.labels_ == 2)[0]

group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
print(group0_random_indexes)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
print(group1_random_indexes)
```

```
group2_random_indexes = np.random.choice(two_indexes, size=5, replace=False)
print(group2_random_indexes)
```

```
[1158 2043 2369  425  785]
[1589  273   46 1116  832]
[2817 3692 3671 4600 4589]
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

---

yUOzD6C7JVJwLGGVZVmmh1bTUCk6AciRm1RB3tardN7mKJhlnqN5Tiu2bjjhh0A38WScYe93jtt7kq_4e2xEIzz5B6KHTuSTqfgu8303GIedi7iQ0l1_

gwHGAwYXHvZ7Ntts58N4xD4i_YrR1lRJmzV794W0mFcO_1zLtznZG2qAmYr
7MeR7JWbW92JFRHrb5RzlpYkMMLhC66Ta86qhLR7OaiihvOKr8jlfgB4LT7h9LpCxFm2R

https://email.axioshq.seattleu.edu/c/eJyNkL1OxTAMRp-mWVCq_DcdMrDwAsCMnNi9N6JqIMlVgacnDAxsSJZlfdInH51SL3
dG0PtOt5nwxq5htVFZhzFFikajjSjGx9WSRB2NBbaHa-9vbdL3k3oYc57n_Joa1DSXehkBHWOxRjVT-0FfJDmppOVS08oNQe

https://links.paperlesspost.com/ls/click?upn=u001.fp1Y-2B-2B6EsH5Pp5ZB1HM8Q9NOaIRTPFTb2Sy8VZ2
2F6iGyo7wmr0SIYcLGnfr6zMTC8ctuwXsbFSbOZlByHyg-2FJ
PC_0TtkN1H4U0f6Pz92RxcmQTWeFRLYGQekOoF3Tt
2BWv7XsWPDX94xl7AFlbVpTbit6LsK5oWa1v1qtTiM2Xywozj1N1CKbUzBwCIFA5iz9er
2BkWjSjD7WtEZlHnr7LztcbEd0i9CUINKgFbk9JQoqPp

https://email.axioshq.seattleu.edu/c/eJyNkL1uwzAMhJ8mXgIZEq3fQUOXzH2DgjKpWkBst5aCtH36KkOHbAUIkPiAO9xx
VzrlytXfk2Mt2GJepsw_QQBc0yAWmFzkoTDLoJtU3DNS6tfdTT9HKCS5_7_f7

https://email.axioshq.seattleu.edu/c/eJyNU01vo0AM_TXhshrk-WQ4cGgTdZWokbrb9GsvKw9jGhoCS
sT4qMwFCcOCWa62kjAE4SRQJKK8kJmaiAN_Kpl3v45aw6yo6xuSP0TpLlU14gYXNCXma6JQLI5K0njzlwvuoy
KIdk1V5iW1ozNoYxxAstujazusc2IX7tgSu_lIY1h7dnOgV6qHQgJsng-4jXe-mAjjscNQBujAlkyBj69uTRSq1Ar-KSOZyqA9NFzkkpMCgIILA
Jh_tKUy6nqr18u-uXsoryeLn4MsXtRHZen-eBfrsq5mW_4Gh_6wX-8n87Nsh6ulLOzmqBgBJ-NaT-aMgTE5ap-Sunpcb7A1fbt8V40twvcXxez8SF

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

Out[ ]:

URL	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLe
https://www.seattleu.edu/policies/copyright-policy/	51	42	0	
https://seattleu.instructure.com/courses/1615281/assignments/7223571	68	45	14	
https://supsychology.sona-systems.com/	38	31	0	
https://sodolabs.com/	21	16	0	
https://docs.google.com/document/d/1WS6LVmCgILS44SXxciqX4PY8Mb6J-2LN1C9TBifWzEc/edit?usp=sharing	96	74	10	

In [ ]:
```
# Third Group
X.iloc[group2_random_indexes]
```

Out[ ]:

https://na3.docusign.net/Signing/EmailStart.aspx?a=06bf9d8e-dc6b-4b36-bfff-7c8fcf788fe2&acct=69145e49-6ad7-4386-b87d-a321e3f26b9
4be

https://doctorsofnursingpractice1
lid=5744467107905536&nid=6378616556879872&c=5842015976161280&a=5471085795737600&ae=5115680206880768&e_i

https://doctorsofnursingpractice1.ebtrk6.com/unsubscribe?nid=6378616556879872&

https://github.us11.list-manage.com/track/click?u=9d7ced8c4bbd6c2f238673f0f&id=19a143

http://www.linkedin.com/shareArticle?url=https%3A%2F%2Fmailchi.mp%2Ffusicc
2024&mini=true&title=Best+of+May+%2B+New+Music+from+Meshell+Ndegeocello%2C+IAMTHELIVING+ft.+Braxton+Cook%2C+Lucky+

In [ ]:
```
# Cluster sizes
print(len(zero_indexes))
print(len(one_indexes))
print(len(two_indexes))
```

```
572
3145
1487
```

4.3 Kmeans 5 Cluster

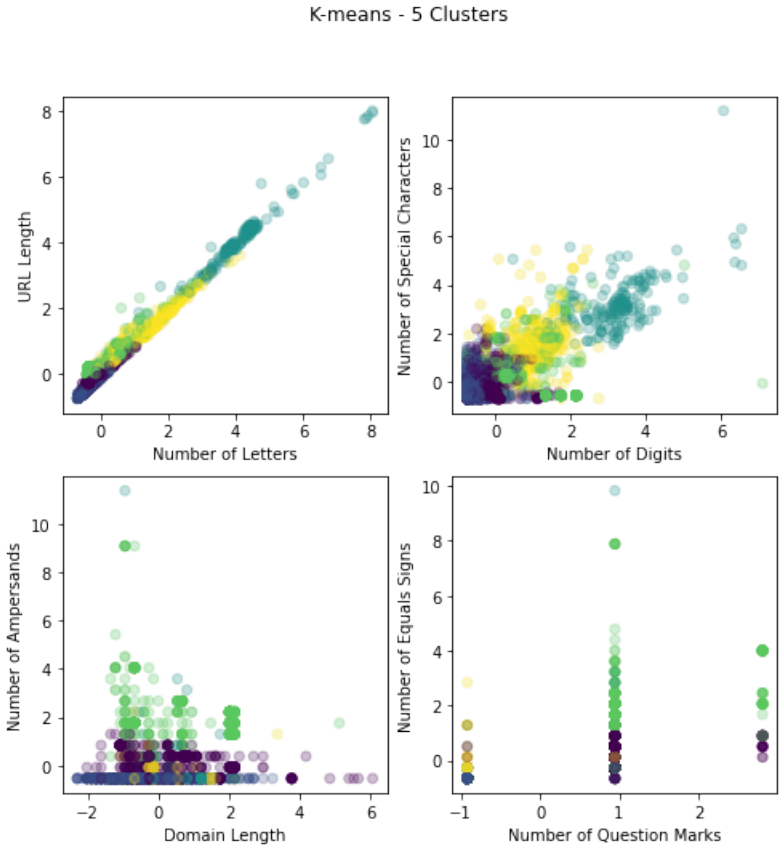
4.3.1 Perform Kmeans with 5 clusters

In [ ]:
```
kmeans5 = KMeans(n_clusters=5,
                 random_state=2,
                 n_init=20).fit(X_scaled)
```

4.3.2 Graph the 5 clusters on variables

In [ ]:
```
fig, ax = plt.subplots(2, 2, figsize=(8,8))

ax[0,0].scatter(X_scaled['NoOfLettersInURL'], X_scaled['URLLength'],
                c=kmeans5.labels_, alpha=0.25)
ax[0,0].set_xlabel('Number of Letters')
ax[0,0].set_ylabel('URL Length')

ax[0,1].scatter(X_scaled['NoOfDegitsInURL'], X_scaled['NoOfOtherSpecialCharsInURL'],
                c=kmeans5.labels_, alpha=0.25)
ax[0,1].set_xlabel('Number of Digits')
ax[0,1].set_ylabel('Number of Special Characters')

ax[1,0].scatter(X_scaled['DomainLength'], X_scaled['NoOfAmpersandInURL'],
                c=kmeans5.labels_, alpha=0.25)
ax[1,0].set_xlabel('Domain Length')
ax[1,0].set_ylabel('Number of Ampersands')

ax[1,1].scatter(X_scaled['NoOfQMarkInURL'], X_scaled['NoOfEqualsInURL'],
                c=kmeans5.labels_, alpha=0.25)
ax[1,1].set_xlabel('Number of Question Marks')
ax[1,1].set_ylabel('Number of Equals Signs')

fig.suptitle("K-means - 5 Clusters")
```

Out[ ]:
```
Text(0.5, 0.98, 'K-means - 5 Clusters')
```



In [ ]:
```
# Get examples of URLs from the three different clusters
zero_indexes = np.where(kmeans5.labels_ == 0)[0]
one_indexes = np.where(kmeans5.labels_ == 1)[0]
two_indexes = np.where(kmeans5.labels_ == 2)[0]
three_indexes = np.where(kmeans5.labels_ == 3)[0]
four_indexes = np.where(kmeans5.labels_ == 4)[0]

group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
print(group0_random_indexes)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
print(group1_random_indexes)
group2_random_indexes = np.random.choice(two_indexes, size=5, replace=False)
print(group2_random_indexes)
group3_random_indexes = np.random.choice(three_indexes, size=5, replace=False)
print(group3_random_indexes)
group4_random_indexes = np.random.choice(four_indexes, size=5, replace=False)
print(group4_random_indexes)
```

```
[2253 2833 2026  130 2127]
[2629  131  133  991 2265]
[4446 2637 4603 3640  546]
[3571 5199 3780 3881 4633]
[ 427 4054 4444 1708 2943]
```

In [ ]:
```
# Cluster sizes
print(len(zero_indexes))
print(len(one_indexes))
print(len(two_indexes))
print(len(three_indexes))
print(len(four_indexes))
```

```
1401
2368
177
792
466
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

URL	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEquals
https://redhawks.sharepoint.com/sites/Intranet-Home/SitePages/Staff.aspx?OR=Teams-HL&CT=1711743166541	101	72	13	23	
https://calendar.google.com/calendar/render?mode=day&date=20240522T180030	73	48	14	19	
https://seattleu.csod.com/ats/careersite/JobDetails.aspx?id=2331&site=2	71	53	5	17	
https://www.eventbrite.com/e/building-a-career-in-sustainability-advice-from-impact-finance-careers-registration-120301532121?utm_campaign=2020-Jobs-Webinar-Impact-Finance&utm_source=Website	190	146	16	18	
https://seattleu.csod.com/ats/careersite/JobDetails.aspx?id=1757&site=2	71	53	5	17	

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

Out[ ]:

URL
https://cta.narvar.com/f/a/fWkwzNPnM1qviihJf6fcUA~~/AAG0DAA~/RgRoOtVdP0UNT3JkZXIgRGV0YWlsc0RaaHR0cHM6Ly9hdGhsZXRhLmM

In [ ]:
```
# Third Group
X.iloc[group2_random_indexes]
```

Out[ ]:

https://ablink.transactions.earnin.com/uni/ls/click?upn=u001.SUmKYsSeX26nJj3C26bNx1l7WIYJ 3DZHf9_Nmuxu6LuKttWZ2PGLNUJbTC3k2oFmPGNjtVd1IM05R4O88E3OTzkFZiOGK35lgj2EsKUlODz894JjsYGOY1qVd-2BuwhR3 2FJSnW4l3TLcdmORukmbzNjgD2k0MsGdlJOadrhO46wCEx0ygHV21xNiI1knwBnFa4ZFdLbVCQe2EkJ3dB-2F8mcq-2B0vDRB-2BTI 2BQh0mjWhYalDNyfUkcdz37wBWnU6bN 2BdJnvE8AQpOlXNXVWWq6ARMHQS75cV5ijLaqaNTvmzGXwOOSboi4gaNi7ghxAYssSUsaoCQZB1Ra9Jp11NpfI6itAzJd609HFqIf5AiT196XG 2BQ-2B6-2BleO9Zhi-2FsMRfQZGwTqH8zyGHfJsZ0torewN0tQFoXF-2FgoiJP00DHft
https://mx.technolutions.net/ss/c/u001.I7QH9QJlytPYeTGKoyJp41GKnfLCHVxPYsCo7cbZb8sKffTw73DcxMyEOy8GI0P8A7nXu8V1jW2uDL_ iTKilJOBXbSYmBGx1S3HM99cOjvL9qutJZozMqu8MwyCvvoksqKEt1GkntMjb3OvRroAEVjsdl0 njx7R9nHByZ_pi3UfzzRkjmGS_s2rrCzXldcepOry0mUltwj2pa-Qd8 RhJxeeU_ZpnfFJ7lgjH8l5_2lyLz0g2hF96pSsPOivz27TKDOF2Wn3rNRqnJKGLwT8FDU0-WXEi1uuJIEAdNrx9-npEKNQV2t KY9JTKzDzGzLs6xItNMiZKyKyg/46j/JJ-Ugnn
https://ablink.transactions.earnin.com/uni/ls/ 2BYiEXMqnedYwe6Wv__2leoliQ6CNyHLiS2I0NGY6MkcGEJJq7d8JYsA37YkkmPCdC3 2FwiXZ64QVweFPhwEkvxPjmVgSByEgv5FUegYFDDN6FpLljD00232Q-2F1YeGtWf2DChnb3bUzsUe4A7t14XgYDnnq8dM 2BAWe4DnclgOvNqRZtRwNnrVMkPxUCd22LgxOHY3R1-2BaI5EDAUdc7-2FRTL0Klnli7Ga68ZxKV 2BwmlCcM6hIZ2yAVbRugMdGVKL7ab0q1XeAqv81vuJvmUoNCEXx-2F-2BSARH5Uy7ZcrV1GhKJQy17yr17MqCd4bpqJwOqahrYxlnKj5oPs2izV 2BzGFDe9rWyu-2FK8LighhdeCaYmtwliZwK5TtQCv6mIVJEkl5B-2FsZvGUGA8F1L-2Be9b4-2BFN4yuwlSa26_
https://ablink.transactions.earnin.com/ls/click?upn=u001.SUmKYsSeX26nJj3C26bNxxlY8MOV 3DjE2a_3qhYZEJDP4UB7uGKsXgUKE7VWqVSZNT2TRN-2BXX 2Bnui1DI49kohXkfBFkAw3p0BySoRYuGdVSL6qpLGOq83aFuiBl8C0 2BAZrw4gsGvf5Fn1gHqalq6hekkRPMf6eqlA479Udp6AsdUAcaUuRJ7zYotOh-2Bj1KHi-2F7K 2BlvkpODEmSh9wJj9YbUOb6m4wwdccohD1u7XKidB2nKwfSJpNAd5-2Fv1CSZmqp8hytYKlY-2Fy5ZA3gzRJVN7mBM6uZDSp7INa-2FxlQ 2BLRFRbB3PUBak7M9WzDu1O9N7BP9zPgdmFPGqWgQx-2FTDVGuAKsPJIKry0uHP51ysNeC-2FRz3ILexCLOyH28tS74avmFqL0EWvKZtb9V
https://email.axioshq.seattleu.edu/c/eJyNUk1v2 A239Dr7u2j81jjkKyUHoIBzyVCKK1JZKpaq0mtBo4kwmOAZAYzGpx5xxzgS3ueFa6kzljuWlJiMrh5a5kWTwWnfD9jkbCLxv6JQRnpLt2LiCoaqE1ag
ZzFRVU1AjG1wdq3hz0dGWH0GdjI64RPIREJsODmbL8cvktUWRK9vULlycnKy0cL01VoRSuKsqcM4vAECpDWFxIpcsZMcVLljtpZVEU2hAZcnFIBdp m66-BVjj7w5Ld_n0b9Z1XM93-dbWJ-j__Q4netlG0uK2UVUMBY8PXxMgd43nt-LHyuYPu3W-viyC7uhFkMpn39u7g6vk4I

In [ ]:
```
# Fourth Group
X.iloc[group3_random_indexes]
```

Out[ ]:

https://www.paypal.com/us/smarthelp/home?v=1&utm_source=unp&utm_medium=email&utr 40a6b72dfb90&ppid=RT000186&cnac=US&rsta=en_US%28en-US%29&cust=&unptid=15e919c0-2020-11ef-865a-40a6b72dfb90 requestee&page=main%3Aemail%3ART000186&pgrp=main%3Aemail&e=cl&mchn=em&s=ci&mail=sys&appVersion=1.256.0&tenant_name
s=66594ccfb78810097f226370&u=50535997&v=3&key=1163&skey=b69a8b9970&url=https%3A%2F%2Fblacklawrencepr s=66594ccfb78810097f226370&u=50535999&v=3&key=59b7&skey=b69a8b9970&url=https%3A%2F%
https://doctorsofnursingpractice1.ebtrk6.com/openurl?lid=5572115841679360&nid=6378616556879872&c=63130192697098 24&a=54710
http://report.mnb.email/t.js?s=66594ccfb78810097f226377&u=50535987&v=3&key=d023&skey=68a88720e2&url=h
https://doctorsofnursingpractice1.ebtrk6.com/openurl?lid=5572115841679360&nid=6378616556879872&c=52421471213977 60&a=54710

In [ ]:
```
# Fifth Group
X.iloc[group4_random_indexes]
```

Out[ ]:

https://email.axioshq.seattleu.edu/c/eJyNkLFuAyEQRL_GNBYW7HHAFRRpXOcPooVdckjOXQLYjvL1IZFSuIu0zYx2R292
4WvJ6arWAM6R0RjV5M1SMnOEbNVyDEr5aIRl7D2_t4O09MBzmPu9_tDxrDSXvcNb6Ve21A3TKlsv02GEo1r4fbTaeElOwtaIqAenF

d=0Y1vxLyk1pJP8FIDrQU57PKf%2C0%2C5%2Chttps%3A%2F%2Fwww.jobcase.com%2Fr%2Fprofile%3F_esk%3D0Y1vxLyk1pJP8FIDrQU57PKf%

https://u14887607.ct.sendgrid.net/ls/click?upn=u001.ybCfF9Qcjr5VEmnOs-2FxQ-2Fqg6HYiomLB5irsEWxbbpoWs1K7dm-2BI2709Q8FBOMzc
2BbzpoxGXgc-2FGvboP0cERXDLebCyQrsy6-2FZxtPoutt8JBQigwCQvU-2BYpM8uX3ILWBD8QrkE9hnhRZcZoc27igGGOigI5QrzW9UD1On7A168

https://email.axioshq.seattleu.edu/c/eJyNkMFOwzAQRL-mviBH9q7tOAcfuPADwBlt7G3jqk1K7FDg
iHOI4tCQQwaNoZRe95KdNrV5hqPfHWcdrEFLyJkTygoj4Oo0OvW7vuIyqwEE0SpzDVeik7vN_BQ7vr9Xrr6A7L

https
fD0h65gHk4vFS3qjHOmh4GSSbUxYtg1_YK27
07Js6zOBz9Q3leR2hW8kSV8vMYRmtMWHZV1h39neNdZgEMEEP89X_X97FCxrgqWqrESqF6c0Y85CcmcM-m

5.0 Hierarchical Clustering

In [ ]:
```
# There are too many records for these scripts to run in a reasonable amount of time
# So, we take a random sample of 50 records
url_sample = url_df.sample(50, random_state=57)

# Set index equal to the domain field
url_sample.set_index('Domain', inplace=True)

# Drop text fields and boolean fields
url_sample_data = url_sample.drop(['TLD', 'IsHTTPs'],axis = 1)
```

In [ ]:
```
# Drop true / false values
X = url_sample_data.drop(['label'],axis = 1)

# Scale the dataset
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X),columns=X.columns,index=X.index)
```

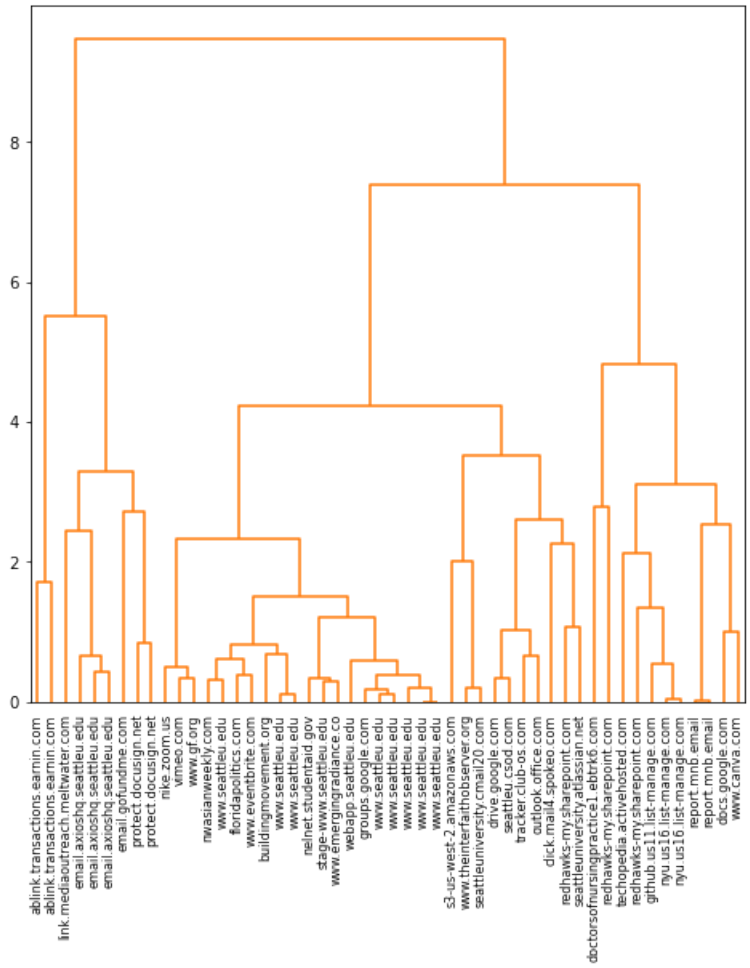
5.1 Hierarchical clustering 2 Clusters

5.1.1 Complete linkage clustering

In [ ]:
```
HClust = AgglomerativeClustering
hc_comp = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='complete')
hc_comp.fit(X_scaled)
```

Out[ ]:
```
AgglomerativeClustering(distance_threshold=0, linkage='complete',
                        n_clusters=None)
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_comp)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=15,
           above_threshold_color='black',labels=X.index.tolist());
```

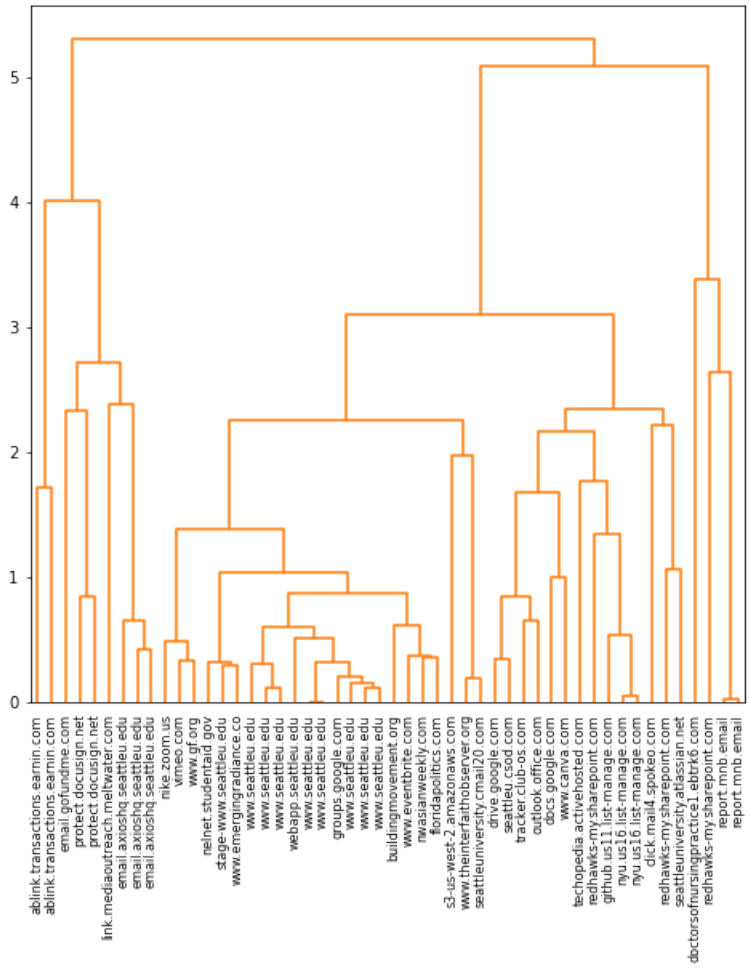


5.1.2 Average linkage clustering

In [ ]:
```
hc_avg = HClust(distance_threshold=0,
                n_clusters=None,
                linkage='average');
hc_avg.fit(X_scaled)
```

Out[ ]:
```
AgglomerativeClustering(distance_threshold=0, linkage='average',
                        n_clusters=None)
```

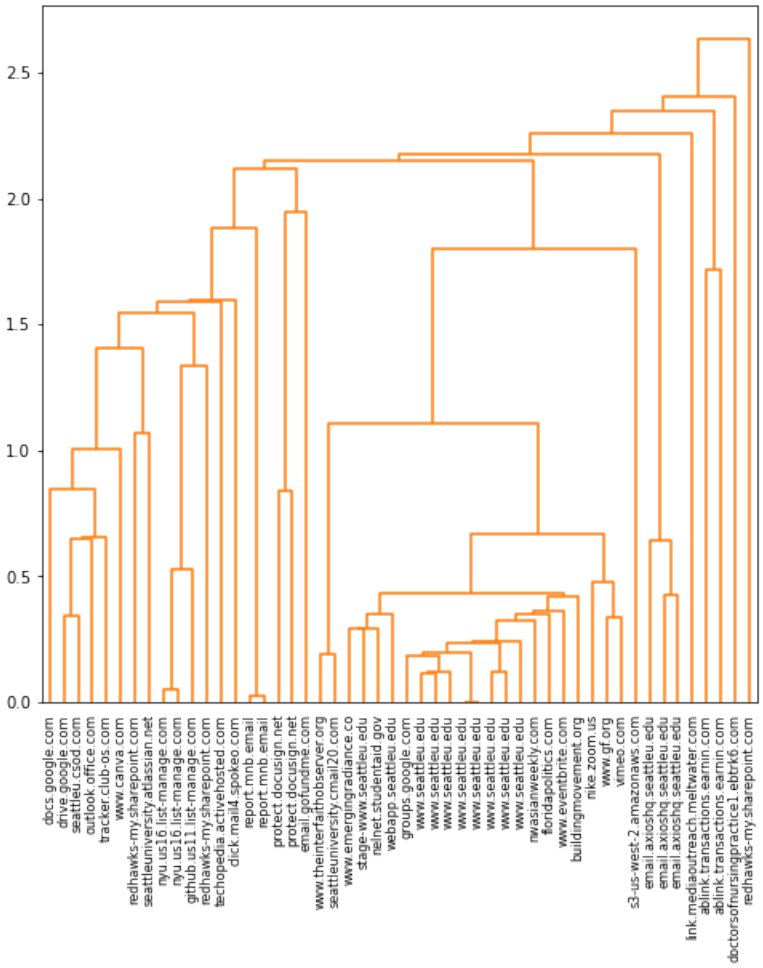
In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_avg)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=16,
           above_threshold_color='black',labels=X.index.tolist());
```



5.1.3 Single linkage clustering

In [ ]:
```
hc_sing = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='single');
hc_sing.fit(X_scaled);
```

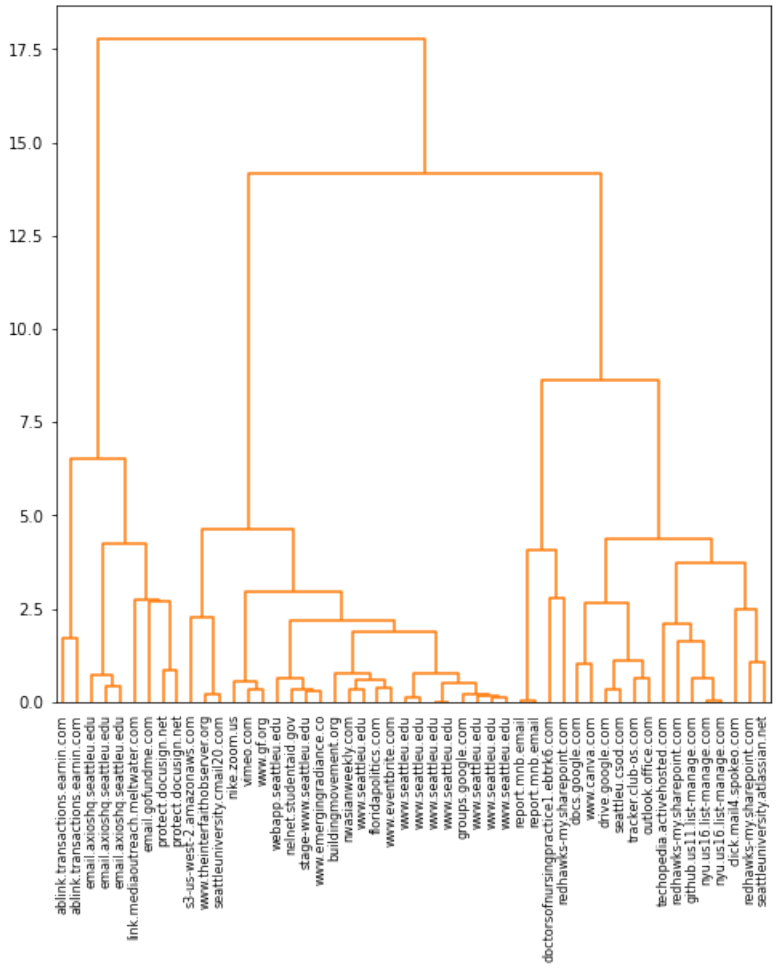
In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_sing)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=5,
           above_threshold_color='black',labels=X.index.tolist());
```



5.1.4 Ward linkage clustering

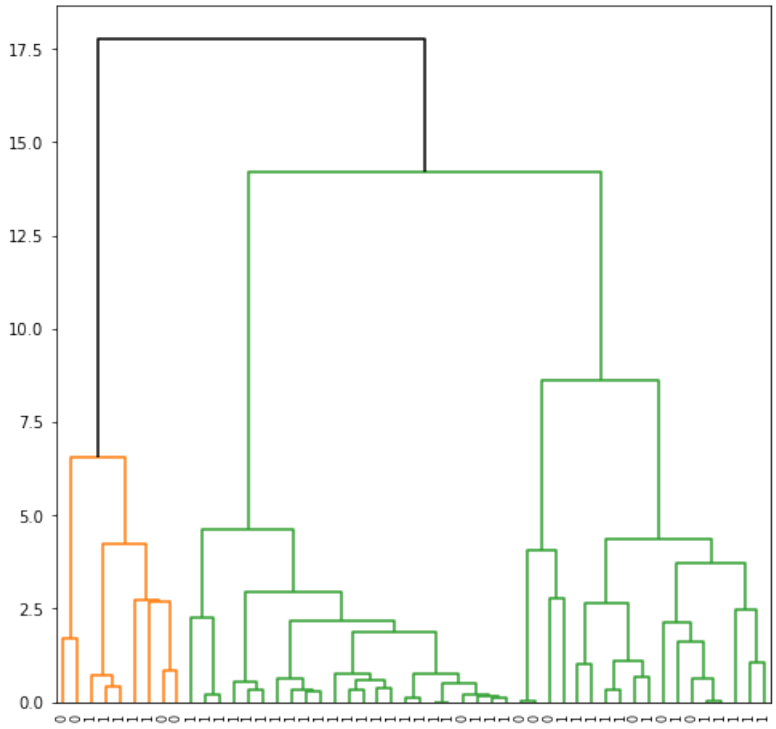
In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=20,
           above_threshold_color='black',labels=X.index.tolist());
```



In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':15,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=15,
           above_threshold_color='black',labels=url_sample_data.label.tolist());
```



In [ ]:
```
clusters_hc = cut_tree(linkage_comp, n_clusters=2).flatten()
```

In [ ]:
```
# Get examples of URLs from the two different clusters
zero_indexes = np.where(clusters_hc == 0)[0]
one_indexes = np.where(clusters_hc == 1)[0]

group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL	NoOfAn
nyu.us16.list-manage.com	99	48	36	24	3	1	
drive.google.com	102	74	14	16	2	1	
seattleu.csod.com	71	53	5	17	2	1	
www.eventbrite.com	95	60	19	18	0	0	
webapp.seattleu.edu	44	37	0	19	0	0	

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

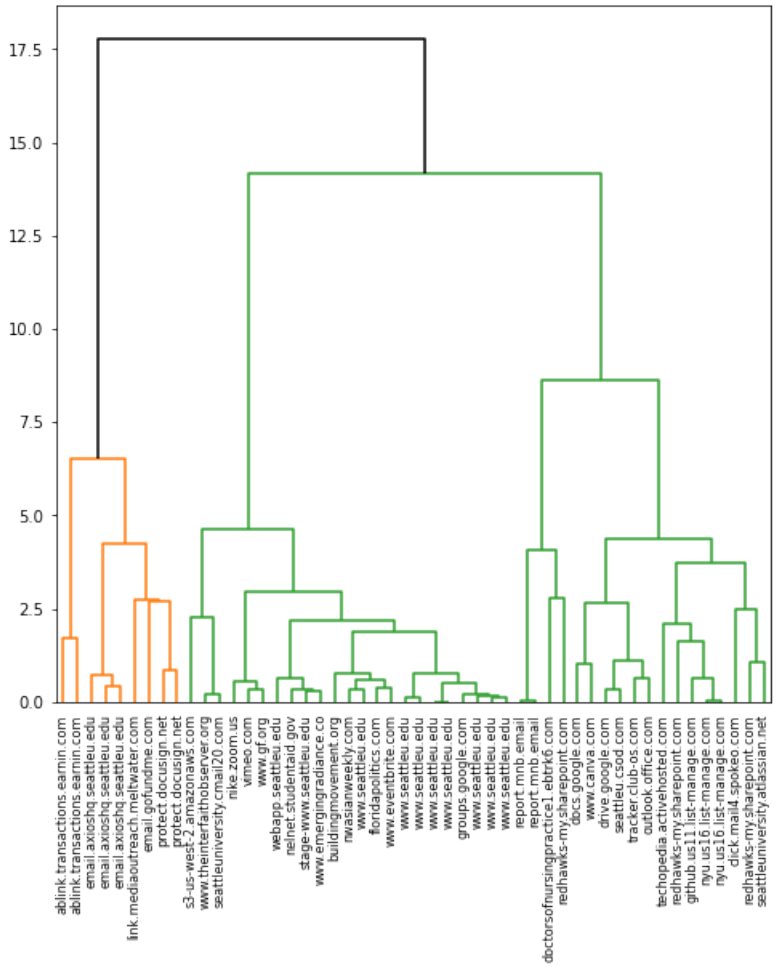
Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkIn
email.axioshq.seattleu.edu	561	461	75	26	0	
email.axioshq.seattleu.edu	552	437	91	26	0	
email.axioshq.seattleu.edu	468	368	76	26	0	
link.mediaoutreach.meltwater.com	567	439	99	32	1	
email.gofundme.com	742	597	110	18	1	

5.2 Hierarchical clustering 3 Clusters

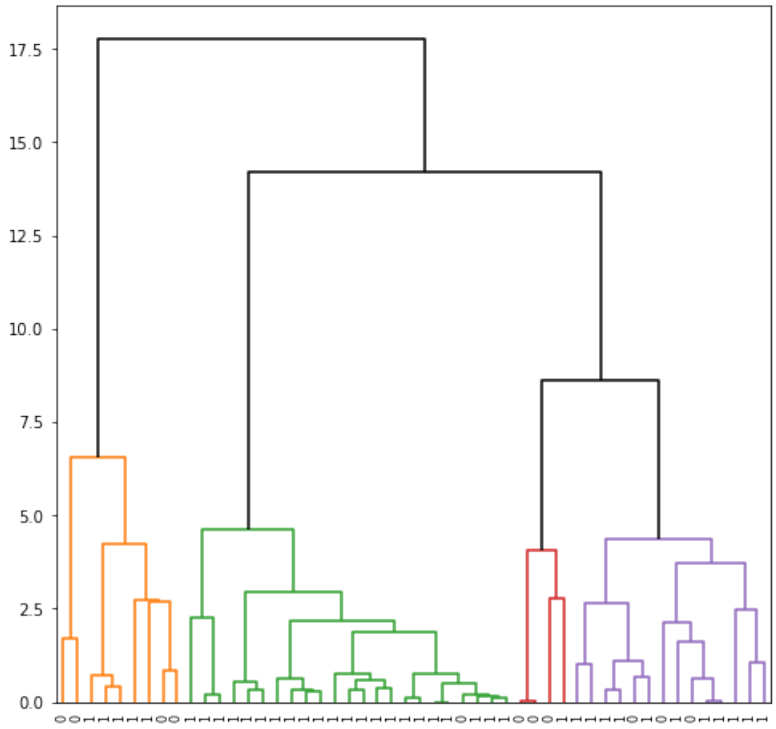
In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=17,
           above_threshold_color='black',labels=X.index.tolist());
```



In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=8,
           above_threshold_color='black',labels=url_sample_data.label.tolist());
```



In [ ]:
```
clusters_hc = cut_tree(linkage_comp, n_clusters=3).flatten()
clusters_hc
```

Out[ ]:
```
array([0, 1, 2, 1, 2, 2, 2, 1, 2, 0, 2, 2, 0, 0, 0, 2, 0, 0, 2, 2, 2, 0,
       1, 2, 2, 2, 1, 0, 2, 2, 0, 0, 2, 2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1,
       0, 0, 1, 0, 2, 0])
```

In [ ]:
```
np.where(clusters_hc == 1)[0]
```

Out[ ]:
```
array([ 1,  3,  7, 22, 26, 38, 39, 43, 46], dtype=int64)
```

In [ ]:
```
# Get examples of URLs from the two different clusters
zero_indexes = np.where(clusters_hc == 0)[0]
one_indexes = np.where(clusters_hc == 1)[0]
two_indexes = np.where(clusters_hc == 2)[0]
group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
group2_random_indexes = np.random.choice(two_indexes, size=5, replace=False)
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL
outlook.office.com	107	71	23	18	1	1
seattleuniversity.atlassian.net	102	69	19	31	1	1
redhawks-my.sharepoint.com	123	98	7	26	1	1
techopedia.activehosted.com	107	62	28	27	5	1
drive.google.com	102	74	14	16	2	1

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkIn
email.axioshq.seattleu.edu	468	368	76	26	0	
ablink.transactions.earnin.com	908	734	136	30	1	
link.mediaoutreach.meltwater.com	567	439	99	32	1	
ablink.transactions.earnin.com	994	764	181	30	1	
email.axioshq.seattleu.edu	561	461	75	26	0	

In [ ]:
```
X.iloc[group2_random_indexes]
```

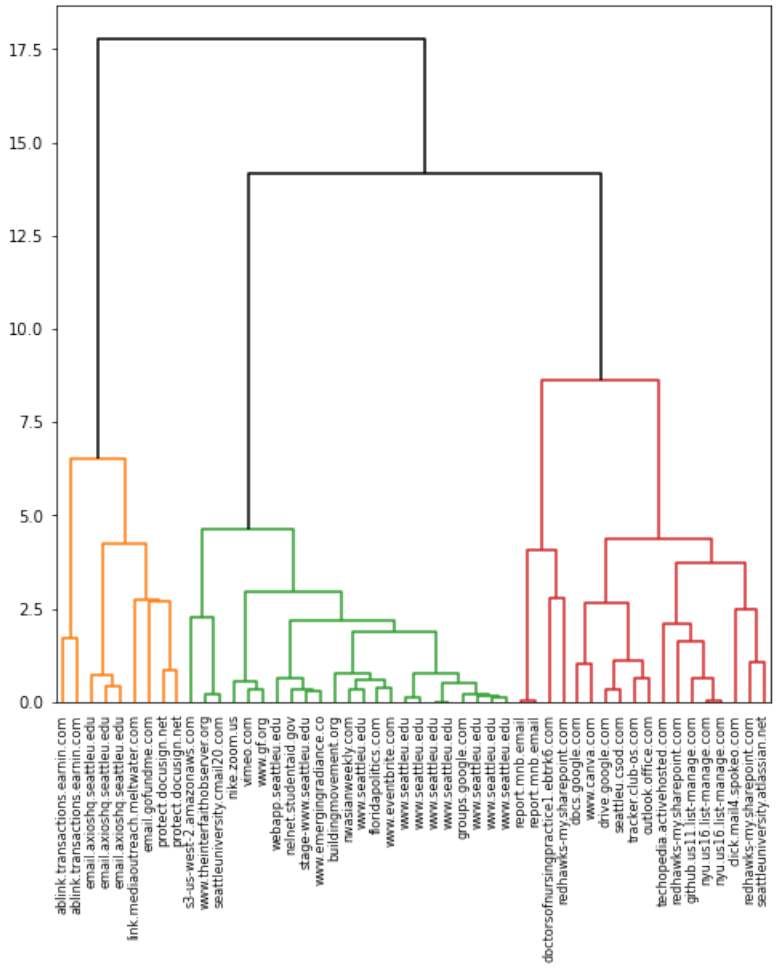
Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL
buildingmovement.org	84	71	0	20	0	0
www.gf.org	58	47	0	10	0	0
www.seattleu.edu	52	43	0	16	0	0
www.theinterfaithobserver.org	83	68	4	29	0	0
webapp.seattleu.edu	44	37	0	19	0	0

5.3 Hierarchical clustering 5 Clusters

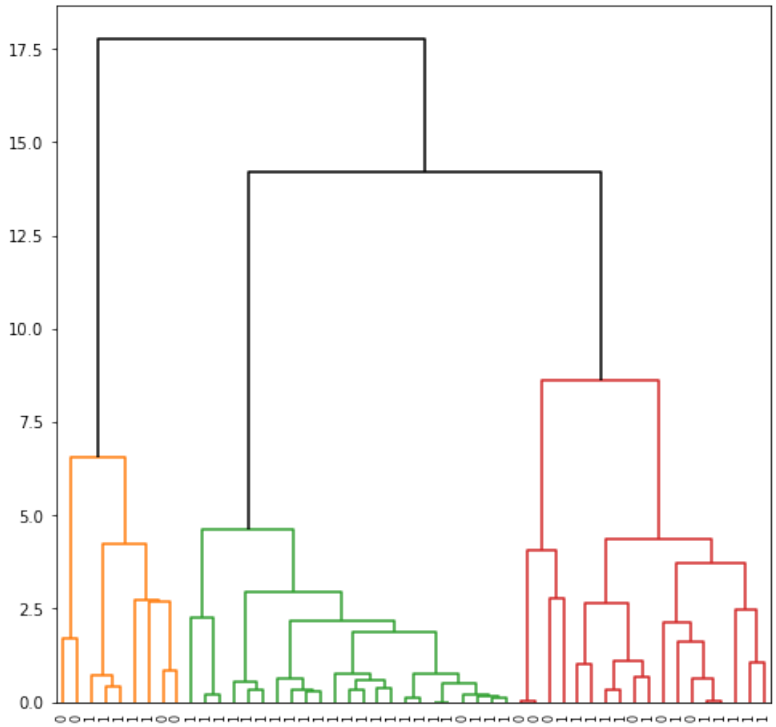
In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=14,
           above_threshold_color='black',labels=X.index.tolist());
```



In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
dendrogram(linkage_comp,
           ax=ax,
           color_threshold=14,
           above_threshold_color='black',labels=url_sample_data.label.tolist());
```



In [ ]:
```
clusters_hc = cut_tree(linkage_comp, n_clusters=5).flatten()
```

In [ ]:
```
clusters_hc
```

Out[ ]:
```
array([0, 1, 2, 1, 2, 2, 2, 1, 2, 0, 2, 2, 3, 0, 3, 2, 0, 3, 2, 2, 2, 3,
       1, 2, 2, 2, 4, 0, 2, 2, 0, 0, 2, 2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1,
       0, 0, 4, 0, 2, 0])
```

In [ ]:
```
# Get examples of URLs from the two different clusters
zero_indexes = np.where(clusters_hc == 0)[0]
one_indexes = np.where(clusters_hc == 1)[0]
two_indexes = np.where(clusters_hc == 2)[0]
three_indexes = np.where(clusters_hc == 3)[0]
four_indexes = np.where(clusters_hc == 4)[0]

group0_random_indexes = np.random.choice(zero_indexes, size=5, replace=False)
group1_random_indexes = np.random.choice(one_indexes, size=5, replace=False)
group2_random_indexes = np.random.choice(two_indexes, size=5, replace=False)
group3_random_indexes = np.random.choice(three_indexes, size=4, replace=False)
group4_random_indexes = np.random.choice(four_indexes, size=1, replace=False)
```

In [ ]:
```
# First Group
X.iloc[group0_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL
redhawks-my.sharepoint.com	192	121	43	26	3	1
drive.google.com	102	74	14	16	2	1
tracker.club-os.com	96	59	27	19	1	1
techopedia.activehosted.com	107	62	28	27	5	1
nyu.us16.list-manage.com	99	48	36	24	3	1

In [ ]:
```
# Second Group
X.iloc[group1_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkIn
email.axioshq.seattleu.edu	552	437	91	26	0	
email.gofundme.com	742	597	110	18	1	
protect.docusign.net	416	332	56	20	2	
email.axioshq.seattleu.edu	561	461	75	26	0	
link.mediaoutreach.meltwater.com	567	439	99	32	1	

In [ ]:
```
X.iloc[group2_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL	No
www.seattleu.edu	87	73	0	16	0	0	
vimeo.com	29	24	0	9	0	0	
www.seattleu.edu	28	21	0	16	0	0	
www.seattleu.edu	83	70	0	16	0	0	
www.emergingradiance.co	32	26	0	23	0	0	

In [ ]:
```
X.iloc[group3_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMar
report.mnb.email	160	91	43	16	6	
redhawks-my.sharepoint.com	277	153	84	26	8	
report.mnb.email	160	92	42	16	6	
doctorsofnursingpractice1.ebtrk6.com	174	57	98	36	6	

In [ ]:
```
X.iloc[group4_random_indexes]
```

Out[ ]:

Domain	URLLength	NoOfLettersInURL	NoOfDegitsInURL	DomainLength	NoOfEqualsInURL	NoOfQMarkInURL
ablink.transactions.earnin.com	994	764	181	30	1	1

5.4 Hierarchical clustering 2 Clusters, computation of confusion matrix

In [ ]:
```
hc_ward = HClust(distance_threshold=0,
                 n_clusters=None,
                 linkage='ward');
hc_ward.fit(X_scaled);
```

In [ ]:
```
cargs = {'color_threshold':-np.inf,
         'above_threshold_color':'black'}
linkage_comp = compute_linkage(hc_ward)
```

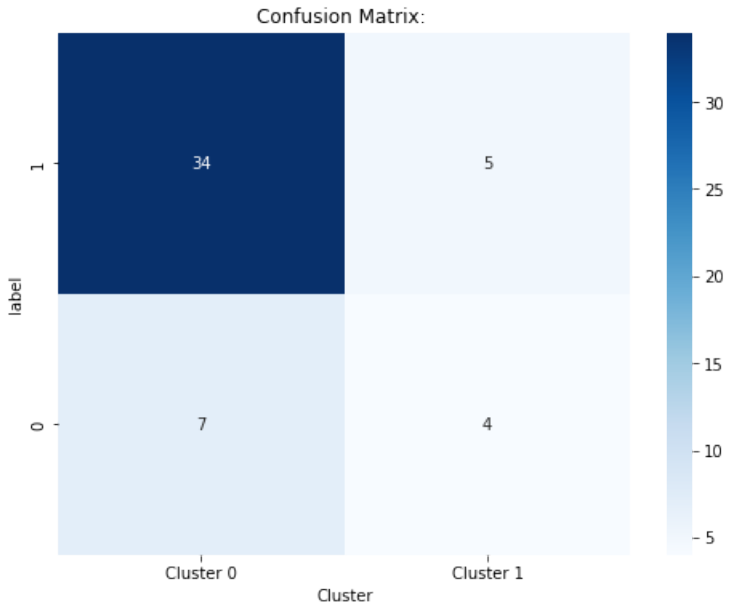
In [ ]:
```
clusters_hc = cut_tree(linkage_comp, n_clusters=2).flatten()
clusters_hc
```

Out[ ]:
```
array([0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0])
```

In [ ]:
```
url_sample_data['label'].unique()
```

Out[ ]:
```
array([0, 1], dtype=int64)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

# Switch the 0 and 1 label to match the cluster label
url_sample_data['label'] = 1 - url_sample_data['label']

# Create a confusion matrix
conf_matrix = confusion_matrix(url_sample_data['label'], clusters_hc)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Cluster 0', 'Cluster 1'], yticklabels=
plt.xlabel('Cluster')
plt.ylabel('label')
plt.title('Confusion Matrix: ')
plt.show()
```



In [ ]:
```
accuracy = (34+4)/(34+7+5+4)
print("Hierarchical Dendrogram Accuracy: {:.2f}%".format(accuracy*100))
```

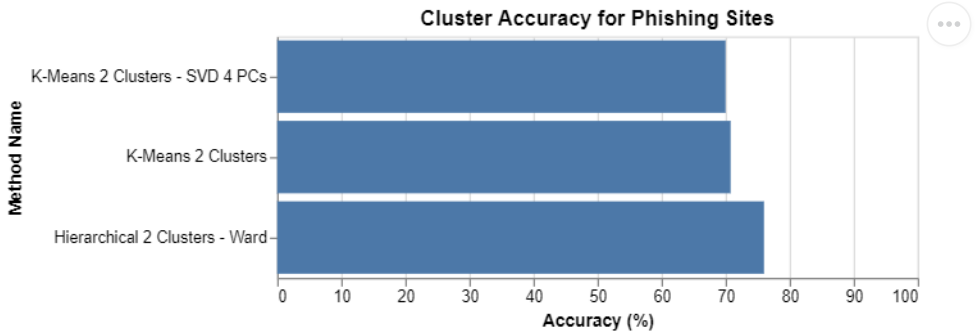
```
Hierarchical Dendrogram Accuracy: 76.00%
```

6.1 Accuracy Chart

In [ ]:
```
# Import Altair library
import altair as alt

# Phishing label accuracy chart
phishing_clusters = {'Method Name': ['K-Means 2 Clusters', 'K-Means 2 Clusters - SVD 4 PCs', 'Hierarchical 2 Clu
                'Accuracy': [70.79, 70.00, 76.00]}

accuracy_chart_df = pd.DataFrame(phishing_clusters)
```

In [ ]:
```
alt.Chart(accuracy_chart_df, title='Cluster Accuracy for Phishing Sites').mark_bar().encode(
    alt.X('Accuracy:Q', title = 'Accuracy (%)', scale=alt.Scale(domain=[0,100])),
    alt.Y('Method Name:N', sort='x'),
).properties(
    width=400,
    height=150
)
```

Out[ ]:


Cluster Accuracy for Phishing Sites
K-Means 2 Clusters - SVD 4 PCs
K-Means 2 Clusters
Hierarchical 2 Clusters - Ward
Method Name
0
10
20
30
40
50
60
70
80
90
100
Accuracy (%)

In [ ]:

In [1]:
```
pip install ISLP
```

```
Collecting ISLP
  Downloading ISLP-0.4.0-py3-none-any.whl (3.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 13.6 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.25.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.11.4)
Requirement already satisfied: pandas>=0.20 in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.0.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from ISLP) (4.9.4)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.4.2)
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.14.
2)
Collecting lifelines (from ISLP)
  Downloading lifelines-0.28.0-py3-none-any.whl (349 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 349.2/349.2 kB 22.3 MB/s eta 0:00:00
Collecting pygam (from ISLP)
  Downloading pygam-0.9.1-py3-none-any.whl (522 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 522.0/522.0 kB 28.4 MB/s eta 0:00:00
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.3.0+cu121)
Collecting pytorch-lightning (from ISLP)
  Downloading pytorch_lightning-2.2.5-py3-none-any.whl (802 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 802.3/802.3 kB 42.3 MB/s eta 0:00:00
Collecting torchmetrics (from ISLP)
  Downloading torchmetrics-1.4.0.post0-py3-none-any.whl (868 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 868.8/868.8 kB 23.8 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=
0.20->ISLP) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.20->ISL
P) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.20->IS
LP) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lea
rn>=1.2->ISLP) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13-
>ISLP) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.
13->ISLP) (24.0)
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISL
P) (3.7.1)
Requirement already satisfied: autograd>=1.5 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP)
(1.6.2)
Collecting autograd-gamma>=0.3 (from lifelines->ISLP)
  Downloading autograd-gamma-0.5.0.tar.gz (4.0 kB)
  Preparing metadata (setup.py) ... done
Collecting formulaic>=0.2.2 (from lifelines->ISLP)
  Downloading formulaic-1.0.1-py3-none-any.whl (94 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.2/94.2 kB 1.2 MB/s eta 0:00:00
Requirement already satisfied: progressbar2<5.0.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pyga
m->ISLP) (4.2.0)
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning-
>ISLP) (4.66.4)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->
ISLP) (6.0.1)
Requirement already satisfied: fsspec[http]>=2022.5.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-
lightning->ISLP) (2023.6.0)
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.10/dist-packages (from pytorc
h-lightning->ISLP) (4.12.1)
Collecting lightning-utilities>=0.8.0 (from pytorch-lightning->ISLP)
  Downloading lightning_utilities-0.11.2-py3-none-any.whl (26 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.14.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (1.12.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.1.4)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch->ISLP)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch->ISLP)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch->ISLP)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch->ISLP)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch->ISLP)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch->ISLP)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch->ISLP)
```

```
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch->ISLP)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch->ISLP)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch->ISLP)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch->ISLP)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (2.
3.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch->ISLP)
  Downloading nvidia_nvjitlink_cu12-12.5.40-py3-none-manylinux2014_x86_64.whl (21.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 50.1 MB/s eta 0:00:00
Requirement already satisfied: future>=0.15.2 in /usr/local/lib/python3.10/dist-packages (from autograd>=1.5->l
ifelines->ISLP) (0.18.3)
Collecting interface-meta>=1.2.0 (from formulaic>=0.2.2->lifelines->ISLP)
  Downloading interface_meta-1.3.0-py3-none-any.whl (14 kB)
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.2.2->li
felines->ISLP) (1.14.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0
->pytorch-lightning->ISLP) (2.31.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from fssp
ec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.9.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>
=0.8.0->pytorch-lightning->ISLP) (67.7.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.
0->lifelines->ISLP) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->l
ifelines->ISLP) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=
3.0->lifelines->ISLP) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=
3.0->lifelines->ISLP) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->
lifelines->ISLP) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.
0->lifelines->ISLP) (3.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels>=
0.13->ISLP) (1.16.0)
Requirement already satisfied: python-utils>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from progressbar
2<5.0.0,>=4.2.0->pygam->ISLP) (3.8.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch->
ISLP) (2.1.5)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->tor
ch->ISLP) (1.3.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0
a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a
0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.
0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.
0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a
0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohtt
p!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (4.0.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from reques
ts->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[h
ttp]>=2022.5.0->pytorch-lightning->ISLP) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->fs
spec[http]>=2022.5.0->pytorch-lightning->ISLP) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->fs
spec[http]>=2022.5.0->pytorch-lightning->ISLP) (2024.6.2)
Building wheels for collected packages: autograd-gamma
  Building wheel for autograd-gamma (setup.py) ... done
  Created wheel for autograd-gamma: filename=autograd_gamma-0.5.0-py3-none-any.whl size=4030 sha256=9e8462efce9
3b28e26319146a137ac31ead8f6fcfa1d47c778aacf1af2057a97
  Stored in directory: /root/.cache/pip/wheels/25/cc/e0/ef2969164144c899fedb22b338f6703e2b9cf46eeebf254991
Successfully built autograd-gamma
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, n
vidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12,
lightning-utilities, interface-meta, nvidia-cusparse-cu12, nvidia-cudnn-cu12, autograd-gamma, pygam, nvidia-cus
```

```
olver-cu12, formulaic, lifelines, torchmetrics, pytorch-lightning, ISLP
Successfully installed ISLP-0.4.0 autograd-gamma-0.5.0 formulaic-1.0.1 interface-meta-1.3.0 lifelines-0.28.0 li
ghtning-utilities-0.11.2 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.
1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26 nvidia-cufft-cu12-11.0.2.54 nvidia-curand-cu
12-10.3.2.106 nvidia-cusolver-cu12-11.4.5.107 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-cu12-2.20.5 nvidia-nv
jitlink-cu12-12.5.40 nvidia-nvtx-cu12-12.1.105 pygam-0.9.1 pytorch-lightning-2.2.5 torchmetrics-1.4.0.post0
```

In [ ]:
```
# Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.datasets import get_rdataset
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression, LassoCV, LogisticRegression
from sklearn.preprocessing import LabelBinarizer, StandardScaler
from sklearn.metrics import mean_absolute_error, accuracy_score, confusion_matrix, classification_report
from ISLP import load_data
from sklearn.cluster import \
     (KMeans,
      AgglomerativeClustering)
from scipy.cluster.hierarchy import \
     (dendrogram,
      cut_tree)
from ISLP.cluster import compute_linkage

np.random.seed(2)
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree, DecisionTreeRegressor
from xgboost import XGBClassifier
import xgboost as xgb
from scipy.stats import uniform, randint
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.model_selection import GridSearchCV
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

1.1 Import Datasets

In [ ]:
```
# Load the New dataset
new_safe_urls = pd.read_csv("/content/drive/MyDrive/All_Documents/SEATTLE_UNIVERSITY_Files/Masters_In_DataScienc
```

In [ ]:
```
# New phishing dataset requires extra processing
import csv

def process_urls(input_csv):
    # Read the input CSV file
    with open(input_csv, mode='r', newline='', encoding='utf-8') as infile:
        reader = csv.reader(infile)
        next(reader)  # Skip the header row
        urls = [row[0] for row in reader]

    # Split URLs by pipe and remove duplicates
    unique_urls = set()
    for url in urls:
        parts = url.split('|')
        for part in parts:
            cleaned_url = part.strip()
            if cleaned_url:
                unique_urls.add(cleaned_url)

    return list(unique_urls)

new_phish_urls = process_urls("/content/drive/MyDrive/All_Documents/SEATTLE_UNIVERSITY_Files/Masters_In_DataScie
```

1.2 Process the URLs

In [ ]:
```
import re
from urllib.parse import urlparse
```

```
def analyze_url(url, label):
    # Parse the URL
    parsed_url = urlparse(url)

    # Calculate length
    url_length = len(url)

    # Count letters, digits, and special characters
    num_letters = sum(c.isalpha() for c in url)
    num_digits = sum(c.isdigit() for c in url)
    num_equals = url.count('=')
    num_question_marks = url.count('?')
    num_ampersands = url.count('&')

    # Count periods, ignoring the first two
    num_periods = max(0, url.count('.') - 2)

    # Count special characters excluding slashes and periods
    num_special_chars = sum(not c.isalnum() and c not in ('/', '.') for c in url) - (num_equals + num_question_r

    # Extract domain and TLD
    domain = parsed_url.netloc
    tld = domain.split('.')[-1] if '.' in domain else ''
    domain_length = len(domain)

    # Check if URL uses HTTPS
    has_https = parsed_url.scheme == 'https'

    # Populate dictionary
    url_analysis = {
        'URL': url,
        'URLLength': url_length,
        'NoOfLettersInURL': num_letters,
        'NoOfDegitsInURL': num_digits,
        'Domain': domain,
        'TLD': tld,
        'DomainLength': domain_length,
        'NoOfEqualsInURL': num_equals,
        'NoOfQMarkInURL': num_question_marks,
        'NoOfAmpersandInURL': num_ampersands,
        'NoOfOtherSpecialCharsInURL': num_special_chars,
        'IsHTTPs': has_https,
        'label': label
    }

    return url_analysis

results = []

# process safe urls
for index, row in new_safe_urls.iterrows():
    url_analysis = analyze_url(row['URL'], 1)
    results.append(url_analysis)

print(len(results))

# process phishing urls
for url in new_phish_urls:
    url_analysis = analyze_url(url, 0)
    results.append(url_analysis)

print(len(results))

new_urls = pd.DataFrame(results)
```

```
3218
5204
```

In [ ]:
```
new_urls.tail()
```

Out[ ]:

	URL	URLLength	NoOfLettersInURL	NoOfDegitsInURL	D
5199	https://github.us11.list-manage.com/track/clic...	102	59	28	github.us11.list-mana
5200	https://doctorsofnursingpractice1.ebtrk6.com/o...	154	55	82	doctorsofnursingpractice1.ebtr
5201	https://tracker.club-os.com/campaign/click?msg...	159	111	26	tracker.club-
5202	https://docs.google.com/drawings/d/189dhm1NB_h...	87	68	9	docs.goog
5203	https://docs.google.com/drawings/d/1nbtjhe7IEl...	87	74	4	docs.goog

1.3 Split the Datasets

In [ ]:
```
new_sample = new_urls[['URLLength', 'NoOfLettersInURL', 'NoOfDegitsInURL', 'NoOfEqualsInURL', 'NoOfQMarkInURL',

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(new_sample.drop('label',axis = 1)
                                                    , new_sample['label']
                                                    , test_size=0.3, random_state=13)
```

In [ ]:
```
# Check the Mean
new_sample.mean()
```

Out[ ]:
```
URLLength                   159.626826
NoOfLettersInURL            113.398155
NoOfDegitsInURL              29.563605
NoOfEqualsInURL               1.696387
NoOfQMarkInURL                0.492890
NoOfAmpersandInURL            1.165450
NoOfOtherSpecialCharsInURL    6.665642
IsHTTPs                       0.891622
label                         0.618370
dtype: float64
```

In [ ]:
```
# Check the Variance
new_sample.var()
```

Out[ ]:
```
URLLength                   35095.559734
NoOfLettersInURL            22039.684225
NoOfDegitsInURL              1559.368431
NoOfEqualsInURL                 6.635459
NoOfQMarkInURL                  0.287284
NoOfAmpersandInURL              4.767739
NoOfOtherSpecialCharsInURL     63.466119
IsHTTPs                         0.096651
label                           0.236034
dtype: float64
```

In [ ]:
```
#scaler = StandardScaler().fit(X_train)
#X_train = scaler.transform(X_train)
#X_test = scaler.transform(X_test)
```

1.4 Linear Logistic Regression

In [ ]:
```
# Fit linear regression model on train set
model = LogisticRegression(solver='liblinear', random_state=0)
lfit = model.fit(X_train, y_train)

# Predict on test set and calculate accuracy
lpred = lfit.predict(X_test)
acc_score = accuracy_score(lpred , y_test)

print("Accuracy: ",acc_score)
```

```
Accuracy:  0.7874519846350833
```

2.1 Decision Trees

In [ ]:
```
# fit decision tree model
tree_phishing = DecisionTreeClassifier()
tree_phishing.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeClassifier()
```

In [ ]:
```
tree_summary = export_text(tree_phishing, feature_names=X_train.columns.tolist())
print(tree_summary)
```

```
|--- NoOfAmpersandInURL <= 1.50
|   |--- NoOfDegitsInURL <= 17.50
|   |   |--- NoOfLettersInURL <= 113.50
|   |   |   |--- URLLength <= 22.50
|   |   |   |   |--- NoOfDegitsInURL <= 2.00
|   |   |   |   |   |--- NoOfLettersInURL <= 15.50
|   |   |   |   |   |   |--- URLLength <= 20.50
|   |   |   |   |   |   |   |--- NoOfLettersInURL <= 12.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfLettersInURL >  12.00
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 14.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 19.50
|   |   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- URLLength >  19.50
|   |   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  14.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- URLLength >  20.50
|   |   |   |   |   |   |   |--- URLLength <= 21.50
|   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 14.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  14.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- URLLength >  21.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfLettersInURL >  15.50
|   |   |   |   |   |   |--- URLLength <= 21.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- URLLength >  21.50
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 0.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 16.50
|   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  16.50
|   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  0.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- NoOfDegitsInURL >  2.00
|   |   |   |   |   |--- class: 0
|   |   |   |--- URLLength >  22.50
|   |   |   |   |--- URLLength <= 93.50
|   |   |   |   |   |--- URLLength <= 86.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 4.50
|   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 5.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 62.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 65.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 28.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 19
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  28.50
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 21
|   |   |   |   |   |   |   |   |--- URLLength >  65.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 53.50
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  53.50
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 12
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  62.50
|   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 65.00
```

```
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  65.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  5.50
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 67.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 63.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  63.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 5
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  67.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 6.50
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength <= 77.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength >  77.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  6.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 63.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  63.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  4.50
|   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 6.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 33.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  33.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  6.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 2.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 53.50
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength <= 67.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 7
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength >  67.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  53.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 54.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  54.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  2.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 10.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  10.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 5
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- URLLength >  86.50
|   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 3.50
|   |   |   |   |   |   |   |   |--- URLLength <= 88.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 76.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  76.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- URLLength >  88.50
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 5.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  5.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  3.50
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 0.50
|   |   |   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  0.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 66.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  66.50
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 87.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 68.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  68.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  87.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 70.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  70.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |--- URLLength >  93.50
|   |   |   |   |   |   |--- NoOfEqualsInURL <= 2.50
|   |   |   |   |   |   |   |--- URLLength <= 120.50
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 15.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 70.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 7.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  7.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  70.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 96.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 8
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  96.50
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength <= 108.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength >  108.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  15.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- URLLength >  120.50
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 13.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 123.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 103.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  103.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  123.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 5.50
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength <= 133.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength >  133.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  5.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  13.50
|   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  0.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 15.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  15.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfEqualsInURL >  2.50
|   |   |   |   |   |   |   |--- NoOfLettersInURL <= 102.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfLettersInURL >  102.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- NoOfLettersInURL >  113.50
|   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 9.00
|   |   |   |   |   |--- NoOfEqualsInURL <= 0.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfEqualsInURL >  0.50
```

```
|   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |--- URLLength <= 136.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- URLLength >  136.00
|   |   |   |   |   |   |   |--- NoOfLettersInURL <= 157.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfLettersInURL >  157.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- NoOfOtherSpecialCharsInURL >  9.00
|   |   |   |   |--- class: 1
|   |--- NoOfDegitsInURL >  17.50
|   |   |--- NoOfEqualsInURL <= 0.50
|   |   |   |--- NoOfLettersInURL <= 137.50
|   |   |   |   |--- NoOfDegitsInURL <= 28.50
|   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |--- NoOfLettersInURL <= 108.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfLettersInURL >  108.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 19.50
|   |   |   |   |   |   |   |--- NoOfLettersInURL <= 58.50
|   |   |   |   |   |   |   |   |--- URLLength <= 70.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 65.00
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 55.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  55.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  65.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- URLLength >  70.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 48.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  48.00
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 86.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  86.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  58.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  19.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 135.50
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 83.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  83.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 51.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  51.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  9.50
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 108.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 51.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  51.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  108.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  135.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfDegitsInURL >  28.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 55.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfDegitsInURL >  55.50
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 60.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  60.00
|   |   |   |   |   |   |   |   |--- URLLength <= 154.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- URLLength >  154.00
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 13.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  13.00
```

```
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- NoOfLettersInURL >  137.50
|   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 11.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  11.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |--- NoOfQMarkInURL <= 0.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfQMarkInURL >  0.50
|   |   |   |   |   |   |--- class: 0
|   |   |--- NoOfEqualsInURL >  0.50
|   |   |   |--- URLLength <= 128.00
|   |   |   |   |--- URLLength <= 94.50
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 2.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  2.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 20.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfDegitsInURL >  20.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- URLLength >  94.50
|   |   |   |   |   |--- NoOfLettersInURL <= 65.50
|   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 8.00
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 31.50
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 30.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  30.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  31.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  8.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfLettersInURL >  65.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 19.50
|   |   |   |   |   |   |   |--- URLLength <= 102.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- URLLength >  102.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfDegitsInURL >  19.50
|   |   |   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |   |   |--- URLLength <= 99.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- URLLength >  99.00
|   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL >  0.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 5.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  5.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |--- URLLength >  128.00
|   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 9.50
|   |   |   |   |   |--- IsHTTPs <= 0.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- IsHTTPs >  0.50
|   |   |   |   |   |   |--- NoOfDegitsInURL <= 91.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfDegitsInURL >  91.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- NoOfOtherSpecialCharsInURL >  9.50
|   |   |   |   |   |--- NoOfLettersInURL <= 658.00
|   |   |   |   |   |   |--- URLLength <= 526.50
|   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 2.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 141.00
|   |   |   |   |   |   |   |   |   |--- URLLength <= 201.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 112.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  112.50
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |--- URLLength >  201.50
|   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  141.00
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 11.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  11.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 75.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  75.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 5
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  2.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 228.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  228.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- URLLength >  526.50
|   |   |   |   |   |   |   |   |--- URLLength <= 751.00
|   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  1.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 22.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  22.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- URLLength >  751.00
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 607.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  607.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfLettersInURL >  658.00
|   |   |   |   |   |   |   |--- NoOfAmpersandInURL <= 0.50
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 22.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 898.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  898.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  22.50
|   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfQMarkInURL >  0.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 134.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 30.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  30.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  134.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 742.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  742.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfAmpersandInURL >  0.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |--- NoOfAmpersandInURL >  1.50
|   |   |--- NoOfLettersInURL <= 115.50
|   |   |   |--- URLLength <= 100.00
|   |   |   |   |--- NoOfLettersInURL <= 72.00
|   |   |   |   |   |--- NoOfLettersInURL <= 53.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfLettersInURL >  53.50
|   |   |   |   |   |   |--- NoOfLettersInURL <= 56.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfLettersInURL >  56.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- NoOfLettersInURL >  72.00
|   |   |   |   |   |--- class: 0
|   |   |   |--- URLLength >  100.00
|   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 9.50
|   |   |   |   |   |--- NoOfDegitsInURL <= 18.50
|   |   |   |   |   |   |--- URLLength <= 134.50
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 6.00
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 3.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  3.50
|   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  6.00
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 5.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  5.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 132.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  132.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- URLLength >  134.50
|   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 4.00
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 4.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  4.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  4.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfDegitsInURL >  18.50
|   |   |   |   |   |   |--- NoOfAmpersandInURL <= 8.00
|   |   |   |   |   |   |   |--- URLLength <= 152.00
|   |   |   |   |   |   |   |   |--- URLLength <= 102.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 101.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 57.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  57.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- URLLength >  101.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- URLLength >  102.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 110.00
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 29.00
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength <= 105.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- URLLength >  105.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  29.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- URLLength >  110.00
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 6.50
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 6.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  6.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  6.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- URLLength >  152.00
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 4.50
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 37.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  37.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  4.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfAmpersandInURL >  8.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- NoOfOtherSpecialCharsInURL >  9.50
|   |   |   |   |   |--- NoOfDegitsInURL <= 50.50
|   |   |   |   |   |   |--- NoOfLettersInURL <= 114.50
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 27.50
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 25.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  25.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  27.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfLettersInURL >  114.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfDegitsInURL >  50.50
|   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 15.00
|   |   |   |   |   |   |   |--- NoOfLettersInURL <= 81.00
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 65.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  65.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfLettersInURL >  81.00
|   |   |   |   |   |   |   |   |--- class: 0
```

```
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  15.00
|   |   |   |   |   |   |--- class: 1
|   |--- NoOfLettersInURL >  115.50
|   |   |--- NoOfQMarkInURL <= 1.50
|   |   |   |--- NoOfDegitsInURL <= 97.50
|   |   |   |   |--- NoOfDegitsInURL <= 4.50
|   |   |   |   |   |--- NoOfLettersInURL <= 129.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- NoOfLettersInURL >  129.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- NoOfDegitsInURL >  4.50
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 22.50
|   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 9.50
|   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 5.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 116.50
|   |   |   |   |   |   |   |   |   |--- NoOfAmpersandInURL <= 3.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfAmpersandInURL >  3.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  116.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 148.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  148.50
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL <= 3.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfEqualsInURL >  3.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- NoOfEqualsInURL >  5.50
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 5.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  5.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  9.50
|   |   |   |   |   |   |   |--- URLLength <= 390.50
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 21.50
|   |   |   |   |   |   |   |   |   |--- URLLength <= 243.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- URLLength >  243.00
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 16.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  16.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- NoOfDegitsInURL >  21.50
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 132.50
|   |   |   |   |   |   |   |   |   |   |--- URLLength <= 204.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- URLLength >  204.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  132.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- URLLength >  390.50
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL <= 325.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- NoOfLettersInURL >  325.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  22.50
|   |   |   |   |   |   |--- NoOfLettersInURL <= 229.00
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 80.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  80.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfLettersInURL >  229.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- NoOfDegitsInURL >  97.50
|   |   |   |   |--- NoOfLettersInURL <= 237.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- NoOfLettersInURL >  237.50
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL <= 44.50
|   |   |   |   |   |   |--- NoOfEqualsInURL <= 3.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfEqualsInURL >  3.50
|   |   |   |   |   |   |   |--- NoOfDegitsInURL <= 190.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- NoOfDegitsInURL >  190.50
|   |   |   |   |   |   |   |   |--- class: 1
```

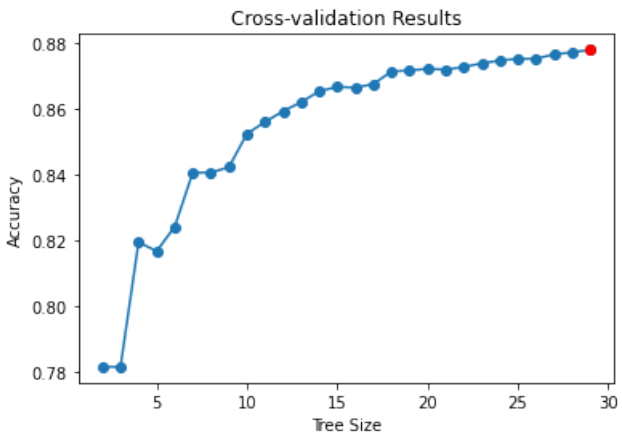
```
|   |   |   |   |   |--- NoOfOtherSpecialCharsInURL >  44.50
|   |   |   |   |   |   |--- NoOfLettersInURL <= 351.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- NoOfLettersInURL >  351.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |--- NoOfQMarkInURL >  1.50
|   |   |   |--- class: 0
```

In [ ]:
```
# predict on test data
tree_pred = tree_phishing.predict(X_test)

# create confusion matrix
confusion_matrix = pd.crosstab(index=tree_pred, columns=y_test, rownames=['']) 
print(confusion_matrix)

print(classification_report(y_test, tree_pred))
```

```
label    0    1

0      502  108
1      110  842
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       612
           1       0.88      0.89      0.89       950

    accuracy                           0.86      1562
   macro avg       0.85      0.85      0.85      1562
weighted avg       0.86      0.86      0.86      1562
```

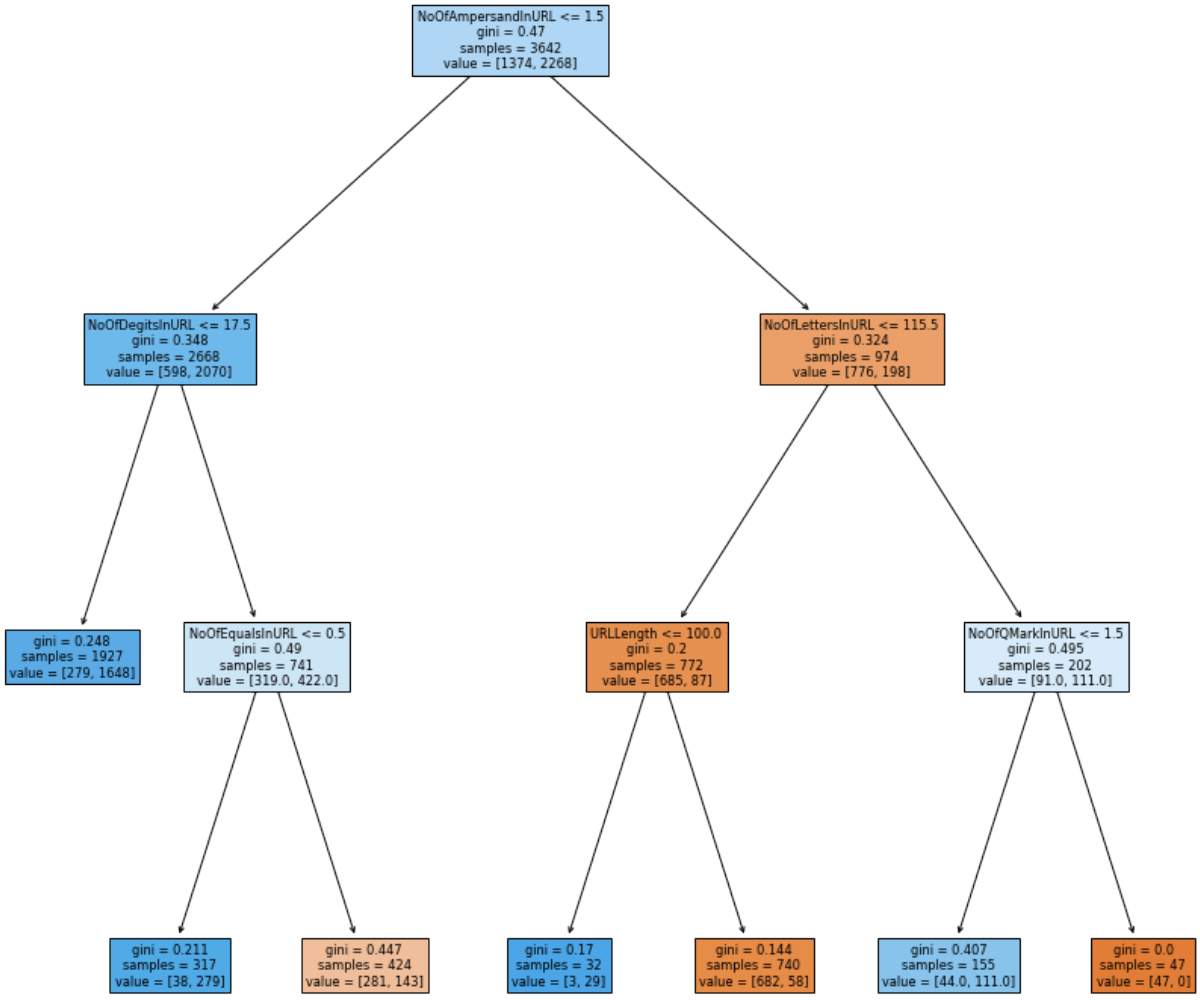
2.2 Cross Validation

In [ ]:
```
# fit decision tree model
tree_phishing = DecisionTreeClassifier(random_state=7)
tree_phishing.fit(X_train, y_train)

# cross-validation to determine optimal tree size
params = {'max_leaf_nodes': range(2, 30)}
cv_phishing = GridSearchCV(tree_phishing, params, cv=10)
cv_phishing.fit(X_train, y_train)
cv_results = cv_phishing.cv_results_

# find the best score for max leaf nodes
best_size = cv_phishing.best_params_['max_leaf_nodes']
best_score = cv_phishing.best_score_

# plot results of cross-validation
plt.figure(figsize=(6, 4))
plt.plot(cv_results["param_max_leaf_nodes"], cv_results["mean_test_score"], 'o-')
plt.plot(best_size, best_score, 'ro-')
plt.xlabel('Tree Size')
plt.ylabel('Accuracy')
plt.title('Cross-validation Results');
```



2.3 Pruning the Tree

In [ ]:
```
# prune tree using optimal size
prune_phishing = DecisionTreeClassifier(max_leaf_nodes=7, random_state=13)
prune_phishing.fit(X_train, y_train)

# plot pruned tree
plt.figure(figsize=(15,15))
plt.title('Pruned Tree')
plot_tree(prune_phishing, feature_names=X_train.columns, filled=True);
```



In [ ]:
```
prune_summary = export_text(prune_phishing, feature_names=X_train.columns.tolist())
print(prune_summary)
```

```
|--- NoOfAmpersandInURL <= 1.50
|   |--- NoOfDegitsInURL <= 17.50
|   |   |--- class: 1
|   |--- NoOfDegitsInURL >  17.50
|   |   |--- NoOfEqualsInURL <= 0.50
|   |   |   |--- class: 1
|   |   |--- NoOfEqualsInURL >  0.50
|   |   |   |--- class: 0
|--- NoOfAmpersandInURL >  1.50
|   |--- NoOfLettersInURL <= 115.50
|   |   |--- URLLength <= 100.00
|   |   |   |--- class: 1
|   |   |--- URLLength >  100.00
|   |   |   |--- class: 0
|   |--- NoOfLettersInURL >  115.50
|   |   |--- NoOfQMarkInURL <= 1.50
|   |   |   |--- class: 1
|   |   |--- NoOfQMarkInURL >  1.50
|   |   |   |--- class: 0
```

In [ ]:
```
# obtain predicted labels for test set
y_pred = prune_phishing.predict(X_test)

# create confusion matrix
confusion_matrix = pd.crosstab(index=y_pred, columns=y_test, rownames=[''])
print(confusion_matrix)
```

```
label    0    1

0      437   97
1      175  853
```

In [ ]:
```
(437+853)/(437+97+175+853)
```

Out[ ]:
```
0.8258642765685019
```

3.1 Random Forests

In [ ]:
```
# Drop is https
new_urls = pd.get_dummies(new_urls, columns=['IsHTTPs'], drop_first=True)
```

In [ ]:
```
new_sample = new_urls[['URLLength', 'NoOfLettersInURL', 'NoOfDegitsInURL', 'NoOfEqualsInURL', 'NoOfQMarkInURL',

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(new_sample.drop('label',axis = 1)
                                                    , new_sample['label']
                                                    , test_size=0.3, random_state=13)
```

In [ ]:
```
y_test.unique()
```

Out[ ]:
```
array([1, 0], dtype=int64)
```

In [ ]:
```
# Scale the dataset
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3.2 Tuning Parameters

In [ ]:
```
param_grid = {
    'max_features': [1,2,3,4,5,6,7,8,9],
    'n_estimators': [200,400,600,800,1000]
}

rf = RFC(random_state=0)

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')

# Fit GridSearchCV to the data
```

```
grid_search.fit(X_train_scaled, y_train)

# Get the best parameters and score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f"Best parameters: {best_params}")
print(f"Best cross-validated score: {best_score}")
```

```
Best parameters: {'max_features': 4, 'n_estimators': 200}
Best cross-validated score: 0.9093922880959917
```

In [ ]:
```
# Train the model with the best parameters
best_rf = grid_search.best_estimator_

# Predict on the test set
y_hat_bag = best_rf.predict(X_test_scaled)

# Calculate the mean squared error
accuracy = np.mean(y_hat_bag == y_test)
print(f"Accuracy on test set: {accuracy}")
```

```
Accuracy on test set: 0.8982074263764405
```

3.3 Variable Importance

In [ ]:
```
# Feature Importance
feature_imp = pd.DataFrame(
    {'importance':best_rf.feature_importances_},
    index=['URLLength', 'NoOfLettersInURL', 'NoOfDegitsInURL', 'NoOfEqualsInURL', 'NoOfQMarkInURL', 'NoOfAmpersa
feature_imp.sort_values(by='importance', ascending=False)
```

Out[ ]:

	importance
DomainLength	0.152871
URLLength	0.149398
NoOfDegitsInURL	0.143194
NoOfLettersInURL	0.140283
NoOfAmpersandInURL	0.137556
NoOfEqualsInURL	0.130596
NoOfOtherSpecialCharsInURL	0.078744
NoOfQMarkInURL	0.051005
IsHTTPs_True	0.016354

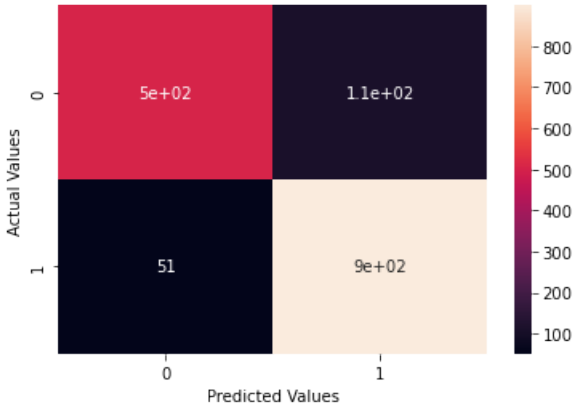
3.4 Random Forest Results

In [ ]:
```
mat_mgb = confusion_matrix(y_test, y_hat_bag)
mat_mgb
```

Out[ ]:
```
array([[504, 108],
       [ 51, 899]], dtype=int64)
```

In [ ]:
```
sns.heatmap(mat_mgb, annot=True)
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
```

Out[ ]:
```
Text(0.5, 15.0, 'Predicted Values')
```



In [ ]:
```
sns.heatmap(mat_mgb, annot=True,xticklabels=["Phishing","Legitimate"], yticklabels=["Phishing","Legitimate"])
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
```

Out[ ]:
```
Text(0.5, 15.0, 'Predicted Values')
```

4.1 Boosting

In [ ]:
```
xgb_model = xgb.XGBClassifier()

params = {
    "colsample_bytree": uniform(0.5, 0.5),
    "gamma": uniform(0, 0.5),
    "learning_rate": uniform(0.01, 0.2),
    "max_depth": randint(3, 10),
    "n_estimators": randint(100, 1000),
    "subsample": uniform(0.5, 0.5)
}

search = RandomizedSearchCV(xgb_model,
                            param_distributions=params,
                            n_iter=200,
                            cv=5,
                            verbose=1,
                            n_jobs=1,
                            return_train_score=True)

search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 200 candidates, totalling 1000 fits
```

Out[ ]:

▸ **RandomizedSearchCV** ⓘ ?

▸ **estimator: XGBClassifier**

▸ XGBClassifier

In [ ]:
```
search.best_params_
```

Out[ ]:
```
{'colsample_bytree': 0.8589674088633774,
 'gamma': 0.4258361373682714,
 'learning_rate': 0.08014470711610436,
 'max_depth': 7,
 'n_estimators': 595,
 'subsample': 0.7030892181592572}
```

In [ ]:
```
xgb_model = xgb.XGBClassifier()
xgb_model.set_params(**search.best_params_)
```

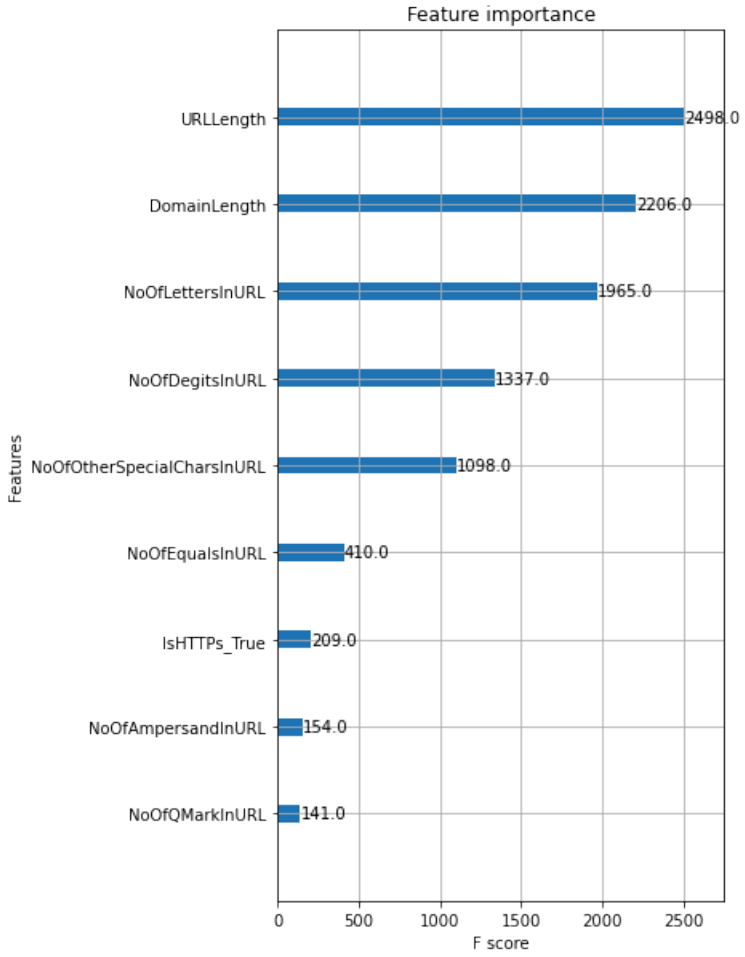
Out[ ]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8589674088633774, device=None,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0.4258361373682714,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.08014470711610436,
              max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=595, n_jobs=None,
```

In [ ]:
```
xgb_model.fit(X_train_scaled, y_train)
```

Out[ ]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8589674088633774, device=None,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0.4258361373682714,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.08014470711610436,
              max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=595, n_jobs=None,
```

4.2 Variable Importance

In [ ]:
```
feature_names = ['URLLength', 'NoOfLettersInURL', 'NoOfDegitsInURL', 'NoOfEqualsInURL', 'NoOfQMarkInURL', 'NoOfA
# Create a mapping of feature indices to actual feature names
feature_importance_dict = {i: feature_names[i] for i in range(len(feature_names))}

# Plot feature importance
fig, ax = plt.subplots(figsize=(5, 10))
xgb.plot_importance(xgb_model, ax=ax)
ax.set_yticklabels([feature_importance_dict[int(i.get_text().strip('f'))] for i in ax.get_yticklabels()])
plt.show()
```



4.3 Boosting Results

In [ ]:
```
y_test.value_counts()
```

Out[ ]:
```
1    950
0    612
Name: label, dtype: int64
```

In [ ]:
```
pred_mgb = xgb_model.predict(X_test_scaled)
```

In [ ]:
```
mat_mgb = confusion_matrix(y_test, pred_mgb)
mat_mgb
```

Out[ ]:
```
array([[510, 102],
       [ 53, 897]], dtype=int64)
```

In [ ]:
```
sns.heatmap(mat_mgb, annot=True,xticklabels=["Phishing","Legitimate"], yticklabels=["Phishing","Legitimate"])
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
```

Out[ ]:
```
Text(0.5, 15.0, 'Predicted Values')
```

In [ ]:
```
print(classification_report(y_test, pred_mgb))
```

```
              precision    recall  f1-score   support

           0       0.91      0.83      0.87       612
           1       0.90      0.94      0.92       950

    accuracy                           0.90      1562
   macro avg       0.90      0.89      0.89      1562
weighted avg       0.90      0.90      0.90      1562
```

5.1 SVM

In [ ]:
```
# train svm
from sklearn.svm import SVC

# grid search with SVC
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf']}
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=3)
grid.fit(X_train_scaled, y_train)

print(grid.best_params_)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.846 total time=   0.6s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.866 total time=   0.4s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.839 total time=   0.4s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.853 total time=   0.5s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.839 total time=   0.5s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.826 total time=   0.5s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.842 total time=   0.4s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.839 total time=   0.4s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.835 total time=   0.5s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.820 total time=   0.4s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.791 total time=   0.5s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.745 total time=   0.5s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.795 total time=   0.5s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.815 total time=   0.5s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.791 total time=   0.5s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.748 total time=   0.8s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.730 total time=   0.7s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.729 total time=   0.8s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.723 total time=   0.7s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.738 total time=   0.6s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.623 total time=   0.8s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.623 total time=   0.7s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.8s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.8s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.624 total time=   1.0s
[CV 1/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.885 total time=   0.4s
[CV 2/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.896 total time=   0.5s
[CV 3/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.885 total time=   0.6s
[CV 4/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.885 total time=   0.6s
[CV 5/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.872 total time=   0.5s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.844 total time=   0.6s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.859 total time=   0.4s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.857 total time=   0.5s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.865 total time=   0.3s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.834 total time=   0.3s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.801 total time=   0.5s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.809 total time=   0.6s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.827 total time=   0.4s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.816 total time=   0.4s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.804 total time=   0.3s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.783 total time=   0.4s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.796 total time=   0.5s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.780 total time=   0.4s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.812 total time=   0.4s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.802 total time=   0.5s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.748 total time=   0.6s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.731 total time=   0.6s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.728 total time=   0.6s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.725 total time=   0.5s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.738 total time=   0.5s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.898 total time=   0.4s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.901 total time=   0.3s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.889 total time=   0.5s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.886 total time=   0.5s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.886 total time=   0.8s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.863 total time=   0.8s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.872 total time=   0.6s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.867 total time=   0.5s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.876 total time=   0.4s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.848 total time=   0.4s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.785 total time=   0.4s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.824 total time=   0.4s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.841 total time=   0.3s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.826 total time=   0.3s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.815 total time=   0.4s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.778 total time=   0.4s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.808 total time=   0.4s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.809 total time=   0.4s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.816 total time=   0.4s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.790 total time=   0.4s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.768 total time=   0.4s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.789 total time=   0.4s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.775 total time=   0.4s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.804 total time=   0.4s
```

```
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.795 total time=   0.4s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.900 total time=   0.6s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.903 total time=   0.4s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.905 total time=   0.5s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.886 total time=   0.7s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.890 total time=   0.7s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.864 total time=   0.6s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.881 total time=   0.5s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.871 total time=   0.7s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.876 total time=   0.6s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.859 total time=   0.4s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.820 total time=   0.4s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.844 total time=   0.5s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.848 total time=   0.4s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.853 total time=   0.4s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.831 total time=   0.4s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.789 total time=   0.3s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.807 total time=   0.4s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.819 total time=   0.4s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.815 total time=   0.4s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.783 total time=   0.3s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.774 total time=   0.4s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.812 total time=   0.4s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.810 total time=   0.4s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.815 total time=   0.4s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.793 total time=   0.5s
[CV 1/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.892 total time=   1.9s
[CV 2/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.897 total time=   2.3s
[CV 3/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.919 total time=   2.2s
[CV 4/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.897 total time=   2.4s
[CV 5/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.896 total time=   1.7s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.877 total time=   1.5s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.892 total time=   1.4s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.876 total time=   1.7s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.883 total time=   1.4s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.867 total time=   1.4s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.855 total time=   1.2s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.863 total time=   1.2s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.856 total time=   1.4s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.870 total time=   1.5s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.841 total time=   1.3s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.782 total time=   0.5s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.805 total time=   0.6s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.830 total time=   0.5s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.810 total time=   0.5s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.786 total time=   0.5s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.789 total time=   0.4s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.804 total time=   0.5s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.824 total time=   0.4s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.820 total time=   0.4s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.795 total time=   0.4s
{'C': 1000, 'gamma': 1, 'kernel': 'rbf'}
```

5.2 SVM Results

In [ ]:

```
svm_pred = grid.best_estimator_.predict(X_test_scaled)

# create confusion matrix
confusion_matrix = pd.crosstab(index=svm_pred, columns=y_test, rownames=[''])
print(confusion_matrix)
print(classification_report(y_test, svm_pred))
```

```
label    0    1

0      486   46
1      126  904
              precision    recall  f1-score   support

           0       0.91      0.79      0.85       612
           1       0.88      0.95      0.91       950

    accuracy                           0.89      1562
   macro avg       0.90      0.87      0.88      1562
weighted avg       0.89      0.89      0.89      1562
```

In [ ]: