Classifying URLs
A Machine Learning Approach

Springboard Data Science Program
Capstone II Project - Final Report
Author: Helga Wilde

1 Table of Contents

2 Executive Summary

Malicious web content is leveraged by threat actors in phishing and malware attacks to achieve objectives such as credential harvesting, data exfiltration or destruction. Accurate classification of URLs is critical for the security of computing systems and networks.

This project's goal was to develop a machine learning model that accurately classifies URLs related to a potential security incident. This is a multi-class classification problem, as we seek to accurately categorize URLs as benign, phishing or malicious.

Incident investigations typically involve analysis of activity that has been permitted by security defenses. An important consideration for this project: What URL characteristics should we train and build our model with, in order to re-classify truly malicious URLs that had bypassed our security controls? After considering available information on URLs and their underlying web infrastructure, it was decided that the project would rely on lexical characteristics, and also explore natural language processing techniques. These characteristics are not leveraged by signature, behavior and rule-based security tools. Thus, we may more accurately classify URLs in a post-incident stage.

Since we are most interested in post-incident URL classification, recall scores for phishing and malicious URLs were designated as the most important scoring mechanism for our model. A 95% or higher recall score was considered a good threshold for adopting a model and placing it into production.

The training data consisted of 30,000 unique URLs, with an equal balance from benign, phishing and malicious categories. Models were built using two unique feature sets. One lexical-based, the other leveraging natural language processing with TF-IDF vector representations of URL tokens (substrings). Multiple algorithms were explored before final algorithms were selected for each model. The model trained on the TF-IDF feature set outperformed the lexical-based model, and also meets our 95% threshold. Final results for each model is as follows:

Model 1. A Random Forest Classifier was trained on a lexical-based set of ninety-six predictor features. Overall accuracy score was .931. Recall scores: benign- .9727, phishing-.9055, and malicious-.9158.

Model 2. A Random Forest Classifier was trained on vectors consisting of URL string, token and TF-IDF scores. Overall accuracy score was .9625. Recall scores: benign-.9765, phishing-.9554, malicious-.955.

3 Background

3.1 Identifying Malicious URL Content

Organizations may deploy multiple security measures to prevent malicious web content from making it through their security infrastructure to computing systems. But enterprises are incapable of blocking 100% of malicious activity due to constant changes in threat actor tactics and techniques. Preventive measures are the best defense, but detective and investigative capabilities can fill in the gaps, and give enterprise IT and security teams the opportunity to recover from security incidents.

For the security industry, protections against malicious URLs has historically been based on:
1. Reputation checks on the URL and/or its underlying domain infrastructure
2. Analysis of the underlying web resource
    - Connection and evaluation of the web page's source code
    - Evaluation of the communication between web client and web server
    - Heuristic analysis of the follow-on activity on the client system

Ongoing URL analysis can and does support a multitude of detection and prevention-based security tools, either by identifying the integrity of specific domains and URLs, or by providing details on specific malicious tactics, techniques and procedures (TTPs) used by threat actors.

3.2 Historical Machine Learning Techniques for URL Classification

Machine learning techniques are being applied to cyber security problems, and several machine learning models have been built to classify URLs. These studies utilize one or more of the following sources to develop data feature sets used to train and test new models:

1. The URL itself. Lexical, aka textual properties, of the URL link.
2. Host-based characteristics
    a. Reputation lists for URL, domain and/or hosting IP address.
    b. Domain name registration information. WHOIS properties such as name servers, associated IP addresses, registrant and registrar records, and dates like domain creation, update and expiration.
    c. Domain name resolution information. DNS records pertaining to the hosting infrastructure, including A, MX, NS, PTR records, IP addresses across all records. Geographic location may be utilized as well.
    d. Connection speed to/from web client and host
    e. Link popularity. A measure of traffic to/from URL resource as compared to established, benign web resources.
    f. URL resource code. Assessment of web content such as links, tags, scripts.

4 Business Case & Project Goals

Incident investigations typically involve analysis of activity that has been permitted by existing security defenses. Perimeter and endpoint security protections rely on signatures, reputation lists, behavioral analysis and rule sets when analyzing urls in web and email traffic. Post-incident url analysis should leverage different url characteristics (versus characteristics reviewed in section 2.2), and thus may identify malicious links from reputable websites and infrastructure, identify malicious links that cannot be fully explored due to anti-forensic techniques, or identify malicious urls leveraging a new threat actor technique.

Therefore, the goals of this project include building feature set(s) and machine learning models that:

- Accurately classify a URL as benign, phishing or malicious
- Are self-reliant and not dependent on URL reputation data, domain registration, DNS information, or connection to a URL link.
- Are not limited or compromised by a threat actor's anti-forensic techniques.
- May be incorporated into an existing security operations and automation tool, to quickly classify a URL that has made it past preventive security measures.

5 Project Approach

5.1 Classification Model

This project's goal is to build one multiclass classification model and accurately classify a URL as benign, phishing or malicious.

5.2 Feature Set

With the above goals in mind, this project focused solely on a URL's lexical features to build a safe, efficient machine learning model.

Two approaches were explored. The first approach leveraged a large feature set based on the URL string and URL components (such as domain name, path, etc.). This is Feature Set 1.

The second approach relied solely on a feature set of URL tokens (substrings) and used natural language processing techniques prior to model training and testing. This data set is referenced as Feature Set 2.

6 Data Sources

Phishing URLs - Over 17,000 phishing URL links were retrieved from PhishTank.[1] PhishTank is a collaborative clearing house for data and information about phishing on the Web. Phishtank's URL lists are available to developers for integration into tools and applications.

Malicious URLs – Over 600,000 malicious URL links were retrieved from abuse.ch.[2] Abuse.ch operates the URLHAUS project, which collects and shares malware URLs to assist network administrators and security analysts in protecting their networks from cyber threats.

Benign URLs - Over 25,000 URLs were collected by crawling Alexa's list of the top 2500 websites.[3] Internal and external links were captured. In order to validate that each URL was 'benign', each URL's reputation was checked via Virus Total's reputation service.[4] VirusTotal inspects URLs with over 70

[1] http://phishtank.org/developer_info.php
[2] https://urlhaus.abuse.ch
[3] https://www.alexa.com/topsites
[4] https://www.virustotal.com

antivirus scanners and URL/domain blacklisting services, as well as other tools. Virus scans were requested in those instances where a URL had no previous scans or reporting available.

Final Dataset – A random sample of 10,000 benign, phishing and malicious URLs were taken for a total of 30,000 records for exploratory data and statistical analysis, and machine learning.

7 Data Wrangling

Phishing, malicious and benign URL records were merged together, each record consisting of a URL and its corresponding category.

Two features sets were constructed as follows.

7.1 Feature Creation – Feature Set 1

The Python urllib module's urlparse function was used to parse URLs into six features, representing the general structure of a URL.[5]

Feature	Value	Description	Value if not present
scheme	URL scheme specifier	Protocol to be used to access the resource on the Internet	scheme parameter
netloc	Network location part	The host name identifies the host where resource is located	empty string
path	Hierarchical path	The specific resource within the host that the user wants to access	empty string
params	Parameters	Parameters for last path element	empty string
query	Query component	Data to be passed to server-side scripts, running on the web server	empty string
fragment	Fragment identifier	Specifies a location within the page	empty string

Table 1: Initial Features Derived from Parsing URLs

[5] See Table 1 for initial features representing parsed URL segments

A domain feature was created by extracting the domain name from the netloc section.

Finally, additional features were created that reflect lexical and other characteristics of the entire URL link as well as its individual segments.[6]

Feature Type	Description	URL Section						
		URL	Domain	NetLoc	Path	Param	Query	Frag
Length	length	✓	✓	✓	✓	✓	✓	✓
	avg section/token length	✓	✓	✓	✓			
	shortest path length				✓			
	longest path length				✓			
Composition	list of all tokens	✓						
	number of sections		✓	✓	✓			
	number of letters	✓	✓	✓	✓	✓	✓	✓
	number of numbers	✓	✓	✓	✓	✓	✓	✓
	number of special characters	✓	✓	✓	✓	✓	✓	✓
	percent of letters	✓	✓	✓	✓	✓	✓	✓
	percent of numbers	✓	✓	✓	✓	✓	✓	✓
	percent of special char	✓	✓	✓	✓	✓	✓	✓
	percent of uppercase letters	✓			✓			
	percent of lowercase letters	✓			✓			
	location of last //	✓						
	location of last slashes as %	✓						
	number of @ signs	✓						
	number of underscores	✓						
	number of question marks	✓						
	number of %20				✓			
	entropy	✓	✓	✓	✓	✓	✓	✓
	number of masques	✓	✓	✓	✓	✓	✓	✓
	character continuity rate	✓						
	is domain an ip address		✓					
	is domain in Alexa top 500		✓					
	number of subdomains		✓					
	number of domain suffixes		✓					
	number of single character paths				✓			

Table 2: Feature Descriptions and Applicable URL Sections

[6] See Table 2 for additional feature descriptions

Following is a brief description of several key features found in Feature Set 1.

Entropy: A Shannon entropy score is calculated, reflecting the string's character distribution. Larger character distributions equate to a higher score.
Character continuity rate: Reflects the total of the following: length of the longest alphabetic string + length of the longest digit string + length of the longest special character string. This total is then divided by the length of the URL.
Number of masques: A masque reflects a letter + digit + letter combination. This characteristic may reflect masquerading, an attempt to spoof a legitimate string with a deceptive replacement. E.g. goog1e.
Percent of lowercase | Percent of upper case: These two features reflect the percent of these characters compared to all alphabet characters in the referenced string.
Is domain an ip address: It's not that uncommon for a URL to have an IP address in lieu of a domain, but malicious links have a *much higher* occurrence.
Is domain in Alexa top 500: Each domain was checked against Alexa's list of the current top 500 websites.[7]

To prepare for statistical analysis and machine learning, categorical features containing strings, such as URL and other URL elements, were dropped. Features with Boolean data types were changed to integer.

The final feature set consists of ninety-six predictor variables and one target variable.

7.2 Feature Creation – Feature Set 2

To create this feature set, URLs were simply parsed into separate substrings using NLTK's Tokenizer.[8] A regular-expression based tokenizer was used to separate URLs based on punctuation. For example www.google.com would be parsed into [www, ., google, ., com].

8 Exploratory Data Analysis

Exploratory data analysis techniques were used to investigate, analyze and summarize characteristics of the URL string and its components. This section covers key findings from feature set 1.

8.1 URL String Analysis

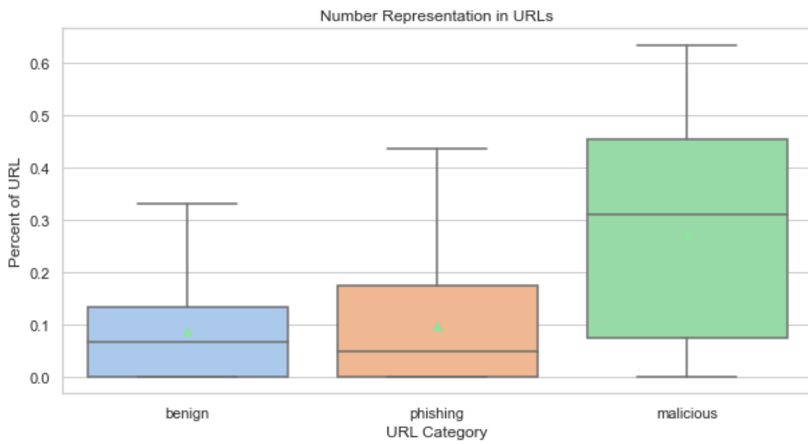
Malicious URLS - Approximately 50% of the malicious URLs in our dataset have an IP address in lieu of a domain name. This accounts for some of the notable differences in malicious URL statistics, in comparison to the other categories. For example, on average, only 50% of the malicious URL consists of letters, versus 71% and 73% for benign and phishing URLs.[9] Malicious URLs are, on average, shorter in length and have shorter token lengths.

Phishing URLs - The mean URL length score is 89.1 in comparison to 57.4 for benign URLs, and 44.9 for malicious URLs. Thus, it's not surprising that phishing URLs have the highest average number of tokens

[7] https://www.alexa.com/topsites
[8] https://www.nltk.org/api/nltk.tokenize.html
[9] See Figure 1: Number Representation in URLs

(20.8 versus 17.4 and 14.8) and longest average token length of 4.1 (versus 3.4 and 3.0). They also have the highest average entropy score and highest average masque count.

Figure 1: Number Representation in URLs



8.2 Registered Domain Analysis

Malicious URLs - 54% of the records in this category have an IP address, not a registered domain name.[10] Malicious domains are on average shorter than phishing domains, but longer than benign domains. This category had the highest entropy score of 1.072, in comparison to .576 for benign and .778 for phishing domains.[11]

Phishing URLS - Phishing domains are longer on average, with a longer token length.

Benign URLs - In comparison with the phishing and malicious domains, benign domains have shorter lengths overall, the lowest entropy score, a greater propensity to be on the Alexa Top 500 website list. They are the least likely URL type to contain an IP address.

[10] Figure 2: IP Addresses per Category
[11] Figure 3: Domain Entropy Scores

Figure 2: IP Addresses per Category

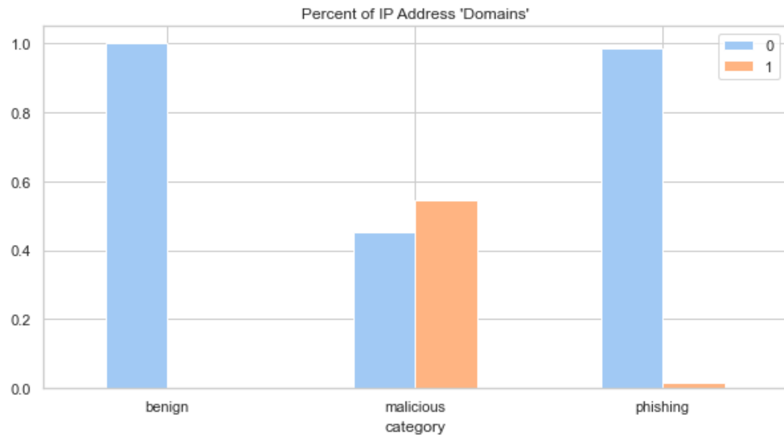
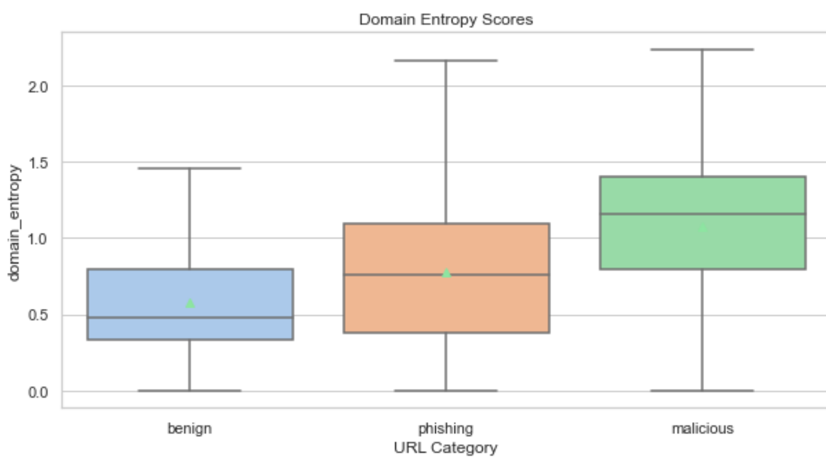


Figure 3: Domain Entropy Scores



8.3 Netloc String Analysis

Malicious URLS - For the malicious URL category, this section is, on average, longer than benign URLs and shorter than phishing URLs (18.07 characters versus 13.55 for benign and 22.59 for phishing). The entropy score of 1.32 is similar to phishing's score of 1.34, but greater than benign URLs' score of .85.

Phishing URLs - The phishing category trumps the others again with its high length scores. This category is also the most likely to have one or more subdomains.

Benign URLs - The benign netloc section has the lowest mean score for length (13.55 in comparison to phishing's 22.59 and malicious' 18.07). It has the lowest mean entropy and masque scores, lowest mean number of tokens, and shortest average token length. Subdomains are more common in benign URLs than malicious URLs.

8.4 Path String Analysis

Malicious URLs - In comparison to benign and phishing URLs, URLs in this category have shorter path sections on average (mean of 18.29 characters versus 30.26 for phishing and 36.46 for benign), and a smaller amount of path items (1.81 versus 2.47 for phishing and 2.26 for benign). They also have lower entropy scores (mean of .722 versus 1.33 and 1.52). In comparison to benign URLs, malicious URLs have a greater percentage of letters and special characters.

Phishing URLs - In comparison to benign and malicious URLs, these URLs have a greater percentage of special characters (29.3% versus 16.2% and 23.2%) and masques (.468 versus .257 and .109). The path sections are a little shorter than benign URL and have a greater propensity for single character paths.

Benign URLs - These URLs have the greatest tendency for numbers within path sections. Numbers comprise 14.3% of the path section, versus 8.7% and 7.6% for phishing and malware. This may be the reason for the highest average entropy score of all categories (1.529 versus 0.722 and 1.334). Its average path item length, 17.34, is much higher than the other categories which score 12.59 and 9.29.

8.5 Inferential Statistics

For feature set 1, it is important to identify strong correlations between pairs of predictor variables and between predictor variables and category, our target variable.

Pairwise Correlation Analysis: We identified a considerable amount of highly correlated features. 73 pairs of features have correlation scores of 80% or higher.

Predictor v. Target Correlation Analysis: Numeric-based domain/netloc features are the most highly correlated with the target feature. The features with the top eight scores may be reflective of the fact that 50+% of the malicious URLs in our dataset have numeric IP addresses instead of domain names. Since the domain and netloc features overlap, features may need to be tailored down before machine learning work.

9 Machine Learning

9.1 Model Evaluation Metrics

Metrics were gathered during machine learning and are included within jupyter notebooks. A few are represented in this report.

9.1.1 Overview of URL Classification Outcomes

Table 3 reviews each possible URL classification outcome

Classification Outcome	Description
True Positive (TP)	Model correctly predicts the positive class. URLs are correctly classified, e.g. A truly benign URL is classified as benign.
False Positive (FP)	The model incorrectly predicts the positive class. URLs are incorrectly classified, e.g. An actual malicious URL is misclassified as benign.
True Negative (TN)	The model correctly predicts the negative class. URLs are correctly classified, e.g. An actual malicious URL is not classified as benign.
False Negative (FN)	The model incorrectly predicts the negative class. URLs are incorrectly classified, e.g. An actual malicious URL is not classified as malicious.

Table 3: Classification Outcomes

9.1.2 Overview of score types

Log Loss - An important classification metric based on probabilities. Log Loss quantifies the accuracy of a classifier by penalizing false classifications. A good metric for comparing models, with lower log loss values meaning better predictions.

Accuracy - Fraction of the total samples that were correctly classified as benign, phishing or malicious. Overall accuracy of the model.

Classification Report

> Precision: The fraction of predictions as a positive class that were actually positive (TP/(TP + FP)).
>
> Recall: The True Positive Rate: the fraction of all positive samples that were correctly predicted as positive by the classifier (TP/(TP + FN))
>
> F1: A weighted averaged of the precision and recall scores, where an F1 score of 1 means it is 100% accurate. (2TP/(2TP + FP + FN))
>
> Note: Our primary goal is to accurately classify malicious or phishing URL links, attaining high true positive rates and low false negative rates. Therefore, recall scores for phishing and malicious URLs are highly valuable.

Confusion Matrix - Confusion matrices allow us to visualize TP / TN / FP / FN scores for each class. Confusion matrices for multi-class problems are not straight-forward. For example, in the matrix below, True positive classifications scores for malicious URLs are obvious. Other measurements like TN, FP, FN can be derived with manual calculations. One can also rely on precision, recall and F1 scoring, which are based on these TP / TN / FP / FN prediction results.

		PREDICTED		
		Benign	Phishing	Malicious
ACTUAL	Benign	TN	TN	FP
	Phishing	TN	TN	FP
	Malicious	FN	FN	TP

Figure 4: Confusion Matrix Guide for Malicious URL Classification

9.2 Feature Set 1 Machine Learning Model

Feature set 1 contains 96 predictor features created from lexical properties of the entire URL string or its components, such as scheme, netloc, domain, path, fragment, parameters and queries.

9.2.1 Baseline Models

Supervised learning algorithms capable of multi-class classification were explored. Prior to any pre-processing or normalization, several classifiers were trained on the feature set. The RandomForestClassifier achieved the best overall accuracy score and TP scores.[12]

			Benign	Phishing	Malicious
	Accuracy	**Log Loss**	**Confusion Matrix - TP Scores**		
KneighborsClassifier	0.852	1.587	0.932	0.817	0.807
DecisionTreeClassifier	0.885	3.961	0.936	0.867	0.853
RandomForestClassifier	0.928	0.222	0.974	0.903	0.906
AdaBoostClassifier	0.843	1.016	0.907	0.822	0.799
GradientBoostingClassifier	0.897	0.297	0.950	0.885	0.856
XGBClassifier	0.883	0.330	0.951	0.860	0.839

Table 4: Overview of Baseline Scores

9.2.2 Random Forest Classifier

Based on the baseline scores, the Random Forest Classifier was selected as the algorithm to develop for this model.

Step 1. Normalization of features

I adopted MinMaxScaler for normalization, constraining the range of all numeric values between 0 - 1. The shape of the original distribution is maintained, so outliers still have influence.

[12] See Table 4: Overview of Baseline Scores

Step 2. Parameter Tuning

I utilized GridSearchCV, which completes a thorough check for the parameter set that returns the best accuracy score. It does this by using all possible parameter combinations. I settled on these parameter settings:

1. Max depth of the tree = 30
2. Max features to consider when looking for the best split = 10
3. Min samples required to be at a leaf node = 1
4. Min samples split required to split an internal node = 2
5. Number of estimators (number of tress in the forest) = 800

Step 3. Validation of Scores

In the previous models, we split the dataset into 80/20 subsets, 80% used for training the model, 20% for testing. To ensure that our newly-tuned model will return similar results in real life applications (with new URLs), a validation test is run.

StratifiedKFold was selected as the validation technique. It is a variation of KFold that preserves the percentage of samples for each class. Both the train and test partitions have equal portions of benign, phishing and malicious URLs. Just like KFold, the validation test trains/tests on alternating sets of data. Accuracy scores are taken following each test.

The mean accuracy score following our StratifiedKFold cross-validation with 5 splits is 93.14, with a standard deviation of .0035. The accuracy scores and standard deviation prove that our model is stable and should perform well on unseen data.
Step 4. Prediction and Final Scores

Our final step is to split our dataset 80/20, fit and predict on the model, evaluate our scores and feature importance. Our final model scores are not higher across the board, but close to initial baseline and other models. [13]

			Benign	Phishing	Malicious
RandomForestClassifier	**Accuracy**	**Log Loss**	**Confusion Matrix – TP scores**		
Baseline Scores	0.928	0.222	0.974	0.903	0.906
Baseline w/ MinMaxScaler	0.932	0.205	0.971	0.907	0.919
Predict after Tuning	0.931		0.973	0.905	0.916

Table 5: Random Forest Classifier Score Review

[13] See Table 5: Random Forest Classifier Score Review

Our final Random Forest Classification report shows some variability in our model's classification accuracy for benign versus phishing and malicious URLs.

```
Classification Report

              precision    recall  f1-score   support

      Benign     0.9358    0.9727    0.9539      1979
    Phishing     0.9429    0.9055    0.9238      1989
   Malicious     0.9154    0.9158    0.9156      2032

    accuracy                         0.9312      6000
   macro avg     0.9314    0.9313    0.9311      6000
weighted avg     0.9313    0.9312    0.9310      6000
```

Domain and netloc features predominate in feature importance scores for our Random Forest Classifier.[14]

Feature	Score
n_netloc_let	0.0489
avg_netloc_tok_len	0.0409
pc_netloc_let	0.0338
n_domain_let	0.0336
len_netloc	0.0306
netloc_entropy	0.0290
pc_domain_let	0.0278
pc_domain_spec	0.0236
pc_path_spec	0.0217
longest_path_len	0.0217
pc_netloc_num	0.0202
avg_path_token_len	0.0192
avg_domain_tok_len	0.0189
n_domain_dots	0.0182
n_let	0.0178
entropy	0.0175
n_domain_tok	0.0174
pc_let	0.0173
pc_path_uppercase	0.0173
domain_entropy	0.0170

Table 6: Feature Importance Scores

[14] See Table 6: Feature Importance Scores; Figure 5: Feature Importance Scores

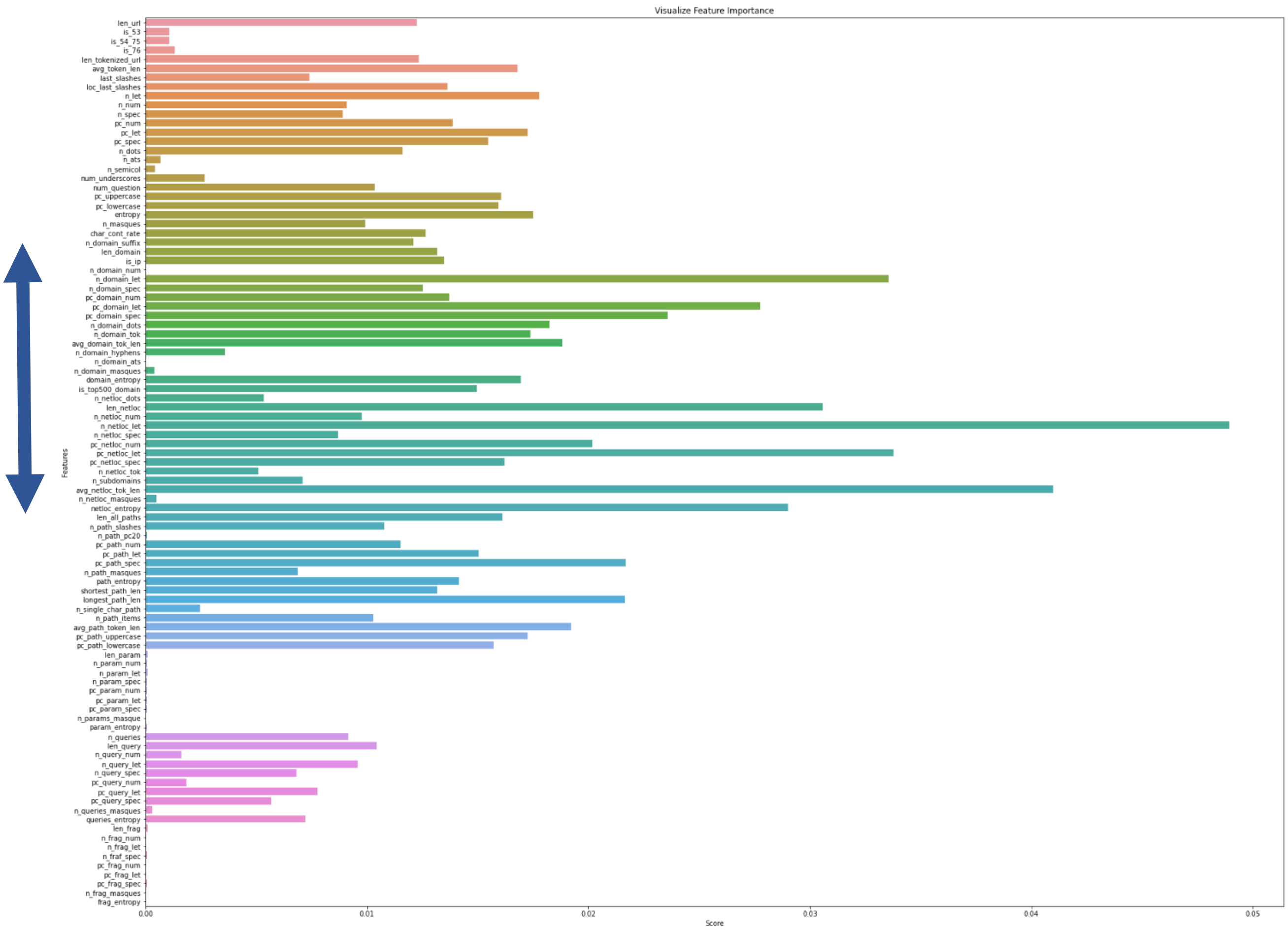


Figure 5: Feature Importance Scores

In order to improve the efficiency of our model and remove some redundancy in our dataset (as indicated with the feature importance scores), I decided to make alterations to the dataset. Details follow.

9.2.3 Random Forest Classifier with Reduced Dataset

As stated previously, netloc and domain features dominated the top 20 features of importance in our Random Forest Classifier. I decided to train models with a reduced feature set. Using the feature importance list and the pairwise correlation data, I carefully selected a set of twenty-two predictor variables that covered the essential URL characteristics.[15] I then completed the same machine learning workflow steps, as with the full dataset.

[15] See Table 7: Reduced Feature Set

Selected Predictor Features	
pc_num	len_netloc
pc_uppercase	n_netloc_tok
pc_spec	pc_netloc_spec
entropy	pc_netloc_num
len_domain	netloc_entropy
is_top500_domain	path_entropy
pc_domain_spec	avg_path_token_len
pc_domain_num	longest_path_len
n_domain_suffix	pc_path_uppercase
n_domain_tok	len_all_paths
domain_entropy	pc_path_num

Table 7: Reduced Feature Set

As the following results in Tables 8 and 9 show, we did not achieve higher scores for baseline models and the final Random Forest Classifier model. Multicollinearity does not negatively affect prediction accuracy.

			Benign	Phishing	Malicious
	Accuracy	**Log Loss**	**Confusion Matrix - TP Scores**		
KneighborsClassifier	81.813	1.874	0.88	0.79	0.78
DecisionTreeClassifier	87.453	4.333	0.91	0.87	0.84
RandomForestClassifier	92.147	0.247	0.97	0.89	0.91
AdaBoostClassifer	82.080	1.016	0.89	0.78	0.79
GradientBoostingClassifier	87.586	0.342	0.94	0.84	0.85
XGBClassifier	86.760	0.367	0.94	0.83	0.84

Table 8: Baseline Classifier Scores with Reduced Feature Set

			Benign	Phishing	Malicious
RandomForestClassifier	**Accuracy**	**Log Loss**	**Confusion Matrix – TP Scores**		
Baseline	0.920	0.245	0.964	0.884	0.911
Baseline w/ MinMaxScaler	0.928	0.222	0.966	0.897	0.920
Predict	0.922		0.967	0.885	0.914

Table 9: Random Forest Classifier Scores, Reduced Feature Set

The tables below compare feature importance scores for both the full and reduced feature sets.[16] Multicollinearity may affect the ability to interpret the parameters learned by our models. We cannot

[16] Table 10: Feature Importance Comparison

say that the features with the largest weights are the most important when features are highly correlated with each other.

The 96 predictor feature set's top scorers stem from the domain/netloc section of the URL. After reducing the feature set to 22 predictors, we still have a high proportion of domain/netloc features, but new features appear and features like entropy and the percent of special characters move up significantly in the importance list.

96 Predictors	**Score**
n_netloc_let	0.0489
avg_netloc_tok_len	0.0409
pc_netloc_let	0.0338
n_domain_let	0.0336
len_netloc	0.0306
netloc_entropy	0.0290
pc_domain_let	0.0278
pc_domain_spec	0.0236
pc_path_spec	0.0217
longest_path_len	0.0217
pc_netloc_num	0.0202
avg_path_token_len	0.0192
avg_domain_tok_len	0.0189
n_domain_dots	0.0182
n_let	0.0178
entropy	0.0175
n_domain_tok	0.0174
pc_let	0.0173
pc_path_uppercase	0.0173
domain_entropy	0.0170

22 Predictors	**Score**
len_netloc	0.0740
pc_domain_spec	0.0677
entropy	0.0639
pc_spec	0.0605
netloc_entropy	0.0602
pc_netloc_spec	0.0522
longest_path_tok	0.0495
n_domain_tok	0.0491
pc_netloc_num	0.0490
avg_path_token_len	0.0485
len_all_paths	0.0459
pc_num	0.0458
pc_uppercase	0.0416
domain_entropy	0.0401
n_domain_suffix	0.0392
pc_path_uppercase	0.0369
path_entropy	0.0362
len_domain	0.0333
pc_domain_num	0.0323
pc_path_num	0.0306

Indicates feature not represented in reduced feature set

Table 10: Feature Importance Comparison

9.3 Feature Set 2 Machine Learning Model

For this approach, natural language processing (NLP) techniques were used to train and test our models.

9.3.1 Feature Creation

For NLP we require a corpus, a collection of texts. For this URL classification project, the corpus consists of URL tokens. We start with just a dataframe of 30,000 URL strings, each classified with a benign, phishing or malicious label. With the help of the nltk.tokenize.WordPunctTokenizer() method, tokens are extracted from each URL string and stored in a new 'tokenized_url' feature. For example, applying the WordPunctTokenizer method on https://www.google.com returns the following tokens: https, ://, www, ., google, ., com.[17]

[17] https://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.regexp.WordPunctTokenizer

9.3.2 Vectorization & Transformation

Our URL tokens need to be transformed into vector representations since machine learning algorithms require a numeric feature set. Our URL token lists are of varying length, but after the numeric transformation step (vectorization), our vector representations will be of uniform length.

There are several vectorization methods and tools to perform the transformation. For this project we leverage TF-IDF and utilize scikit-learn's TfidVectorizer to do the conversion and return a matrix of TF-IDF features.

9.3.2.1 TF-IDF

TF-IDF stands for Term-Frequency-Inverse Document Frequency. With this approach each URL token is assigned a number that is proportional to its frequency in the URL string and inversely proportional to the number of URL strings in which it occurs.

Other encoding approaches exist, like bag-of-words representations. But they only take individual documents (URLs, in our case) into consideration, and not the context of the entire corpus.

Formulas:

N - number of URLs we have in our dataset
d – a given URL from the dataset
D - the collection of all URLs
W - a given token from a URL
f(w,d) - frequency of word w in document d

Term Frequency (TF) Formula

$$tf(w,d) = \log(1+f(w,d))$$

Inverse Document Frequency (IDF) Formula

$$idf(w, D) = \log(N/f(w,D))$$

TF-IDF formula

$$Tfidf(w,d,D) = tf(w,d)*idf(w,D)$$

Following TF-IDF scoring of our URL records, we are left with a matrix of td-idf scores with one row per URL, and as many columns as there are different tokens in the entire corpus.

9.3.2.2 Scikit_learn TfidVectorizer

TF-IDF scores may be calculated manually, but our models utilize Scikit-Learn's TfidVectorizer transformer, which uses an estimator to count the occurrences of tokens (TF), followed by TfidTransformer which normalizes these counts by the inverse document frequency(IDF). The vectorizer returns a sparse matrix representation in the form of ((doc, term), tf-idf score) where each key is a document and term pair.

9.3.3 Baseline Models

Our matrix is finally split into train and test subsets and multiple algorithms are fitted to gather baseline accuracy scores and confusion matrices. The LinearSVC model achieved the best overall accuracy score and true-positive rates (with the exception of TP scores for malicious URLs).[18]

			Benign	Phishing	Malicious
	Accuracy	**Log Loss**	**Confusion Matrix - TP Scores**		
Logistic Regression	94.8830	0.2222	0.9735	0.9478	0.9250
LinearSVC	96.2330		0.9805	0.9684	0.9380
MultinomialNB	93.2000	0.2573	0.9825	0.9188	0.8949
Random Forest	95.1500	0.1754	0.9705	0.9363	0.9475

Table 11: Baseline Classifier Scores

Since these baseline scores are higher than our model using a 96 predictor feature set, we continue to develop the top 3 classifiers - LinearSVC, Random Forest Classifier and Logistic Regression.

9.3.4 Random Forest Classifier – LinearSVC – Logistic Regression

For each of these classifiers, we followed these steps:

<u>Step 1. Normalization of features</u>

No further steps required.

<u>Step 2. Parameter Tuning</u>

We utilized GridSearchCV to test all parameter combinations and provide the best parameters

The following parameters[19] were returned:

[18] Table 11: Baseline Classifier Scores
[19] Table 12: Classifier Best Parameters

LinearSVC	Logisitic Regression	Random Forest Classifier
C = 1	C = 100	max depth = None
penalty = l2		max features = 30
		min samples leaf = 1
		number of estimators = 1000

Table 12: Best Parameters per Classifier

Step 3. Validation of Scores

StratifiedKFold was selected as the validation technique, to preserve the percentage of samples for each class. For each algorithm, our accuracy scores remained consistent across all folds. The mean accuracy scores and standard deviation following StratifiedKFold cross-validation with 5 splits[20]:

	Mean Accuracy	Standard Deviation
LinearSVC	96.31	0.0015
Logistic Regression	96.21	0.0021
Random Forest Classifier	96.21	0.0021

Table 13: StratifiedKFold Results

Step 4. Prediction and Final Scores

Our final step was to split our dataset 80/20, fit and predict on each model and evaluate our scores.

Results as shown in Table 14:

- LinearSVC and Random Forest Classifier tie on the overall accuracy score
- LinearSVC has the highest True Positive detection rate for benign and phishing URLs
- Random Forest Classifier has the best True Positive scoring for malicious URLs

		Benign	Phishing	Malicious
	Accuracy	**Confusion Matrix - TP Scores**		
Logistic Regression	94.8800	0.9735	0.9479	0.9250
LinearSVC	96.2500	0.9835	0.9649	0.9390
Random Forest	96.2500	0.9765	0.9555	0.9555

Table 14: Classifier Final Results

Our models' Classification Reports cover the following scores. See Table 15 for precision, recall and F1 scores for LinearSVC, Random Forest and Logistic Regression models.

Precision	TP / (TP + FP)	The fraction of predictions as a positive class that were actually positive.
Recall	TP / (TP + FP)	The fraction of all positive samples that were correctly predicted as positive by the classifier.
F1	2TP / (2TP + FP + FN)	A combined measure of both precision and recall.

[20] Table 13: StratifiedKFold Results

	LinearSVC		
	precision	recall	f1-score
benign	0.9825	0.9835	0.9830
phishing	0.9286	0.9649	0.9464
malicious	0.9781	0.9391	0.9582

Best Recall Scores:

The Random Forest Classifier model will accurately classify 95.55% of actual malicious urls as malicious.

The LinearSVC model will accurately classify 98.35% of actual benign urls as benign, and 96.49% of actual phishing urls as phishing.

	Random Forest		
	precision	recall	f1-score
benign	0.9839	0.9765	0.9802
phishing	0.9371	0.9554	0.9462
malicious	0.9671	0.9555	0.9613

	Logistic Regression		
	precision	recall	f1-score
benign	0.9659	0.9735	0.9697
phishing	0.9136	0.9479	0.9304
malicious	0.9691	0.9251	0.9466

Table 18: Classification Reports for All Models

9.3.5 Stacking

As a final step to potentially improve on the machine learning model performance utilizing TF-IDF scores, stacking was completed. Stacking is an ensemble learning technique which combines multiple classification models with a meta-classifier. The base-level models are trained on a complete training set. The meta-classifier is trained on the features that are outputs of the base-level model.

Our base models are heterogenous: Logistic Regression, Random Forest Classifier and LinearSV. Logistic Regression is used as the meta-classifier. As per the results in Table 16, our overall accuracy score and recall score for phishing improve.

			Benign	Phishing	Malicious
	Features	**Accuracy**	**Recall Scores**		
Logistic Regression meta-classifier	Base Predictions	0.9633	0.9805	0.9699	0.9396

Table 16 Stacking Results

10 Machine Learning Summary

While both features sets produced overall accuracy scores of 90% or higher, the feature set leveraging natural language processing achieved the best overall scores.[21] The stacked model returned the highest overall accuracy score of 96.33.

Focusing on recall scores, models trained on the TF-IDF scores returned superior results. The Random Forest Classifier model returned the highest score for classifying positive malicious URLs, the LinearSVC model returned the highest recall scores for classifying benign URLs, and the stacked model returned the highest recall score for phishing URLs.

			Benign	Phishing	Malicious
	Features	Accuracy	Recall Scores		
RandomForestClassifi	Lexical	93.3310	0.9727	0.9055	0.9158
LinearSVC	TF-IDF	96.2500	0.9835	0.9649	0.9391
RandomForestClassifi	TF-IDF	96.2500	0.9765	0.9554	0.9555
LogisticRegression	TF-IDF	94.8800	0.9735	0.9479	0.9251
Stacked	Base Models	96.3300	0.9805	0.9699	0.9396

Table 17: Overview of Models: Accuracy & Recall Scores

It is important to note the lightweight nature of each model created for this project. After accumulating the necessary phishing, malicious and benign URL strings, feature creation for both feature sets relied solely on URL strings and were built 'in-house'. No further data was necessary from third party sources. Achieving high scores with models built solely with lexical or TF-IDF-based features is tremendous.

11 Using a Model in a Production Environment

The goal of this project was to build a URL classification model to assist in the investigation of events involving a URL, and/or defend against (or prevent activity) related to phishing and malicious URLs.

The selection of a model is dependent on how it is leveraged in any environment. For example, for investigation of a past event, a model would classify a URL that had already passed a user or organization's security infrastructure. In this case, a model with high recall scores in classifying phishing or malicious URLs would be selected. In an organization was limited to just one model, I would select the Random Forest Classifier trained on the TF-IDF scores. The model could be leveraged in an automated security operations workflow that analyzes URLs links in emails or URL links involved in suspicious web traffic.

[21] Table 17: Overview of Models: Accuracy & Recall Scores

12 Next Steps

Here is a list of potential next steps:

- While models relying on a feature set built entirely from URL lexical properties did not perform as well as those relying on TF-IDF scores, they show promise. Additional features can be explored to supplement the feature set, to improve model performance.
- For TF-IDF based models leveraging stacking, additional base classifiers may be explored, such as GaussianNB and Decision Tree, to potentially improve overall scores.
- Separate models may be built to classify individual URL types. Separate models focusing on one URL type, e.g. malicious URLs, may achieve higher accuracy, precision and recall scores.