Identification of Malicious URL's using Data Mining

Christopher Garner

Florida Institute of Technology

CYB 5998

April 20, 2019

Final Report

Submitted to: Dr. Thomas Eskridge

Identification of Malicious URL's using Data Mining

Malicious websites and URL's

As more of the world becomes connected on through the internet, malicious websites are also on the rise. These websites are set up for a variety of purposes from financial crimes to spreading malware. Links to these malicious sites can be sent out through email, hyperlinks on posts on Facebook, Twitter and other social networks, sent through instant messages and various other means. (Basnet & Doleck, 2015).

Other than websites that have been crafted to steal financial information from unsuspecting users, another issue with these malicious websites is drive-by-downloads. These sites use attacks on vulnerable browsers to execute client-side code without input from the user and infect the user's machines. (Canali, Cova, Vigna & Kruegel, 2011)

Companies such as Google and Bright Cloud have developed plug-ins to blacklist websites, as well as anti-virus software by Avast and McAfee to try and block malicious sites via blacklists (Chakraborty & Lin, 2017) however they are only able to block known sites and are vulnerable to new URL links until they have been evaluated and added to the blacklist. (Jain & Gupta, 2017) The number of sites that need to be blocked continues to grow. In their 2014 annual survey, Cisco reported they blocked 80 million web requests every day that lead to malicious websites. (Mašetic, Subasi & Azemovic, 2016)

End users can be trained to spot suspicious URL's, but according to research (Jain & Gupta, 2017), below are a few reasons that people click on bad URL's.

1. Some people don't look at the entire URL, trusting that it has an HTTPS or seeing a known brand name somewhere in the URL

2. Imparting a sense of urgency in the recipient or invoking a sense of trust in them
3. Not having access to the actual URL, either due to obfuscation or hiding it in hypertext.
4. Feeling that if an email is from a friend or reputable business, it can be trusted.
5. Accidentally clicking on an URL

Due to the above issues, end users need help to protect themselves. Development of automated detection systems can go a long way in this direction. Blacklists are available through most virus protection software; however, blacklists are only effective on sites that have been analyzed and added to their lists. Malicious sites are created constantly making blacklists unable to block them all. The use of data mining techniques shows great progress towards being able to assist in this endeavor. Continued research and fine-tuning of the models used in automated detection systems will be able to help protect end-users from phishing and possible financial crimes.

Literature Review

Learning to Detect Malicious URLs

In this paper (Ma, Saul, Savage & Voelker, 2011) the authors used a binary classification system for malicious and benign websites, where a positive would indicate a malicious site and negative a benign one. They used lexical information from the URL itself as well as host-based information. They state that two additional types of information could be used which might improve accuracy, but also can add complexity to the equation. Additional information that could be used can be gained by downloading and analyzing page information itself, but this also introduces a safety issue. It also increases the time needed to create a prediction. One additional issue they noticed through their research, was the ability of webpages to "cloak" their

information, depending on the end user, such as sending a benign page to honeypot IP's, which makes analyzing the data itself a challenge. They found, however, that using the lexical and host-based information provided highly accurate results.

Some of the lexical information the researchers used is based on previous research by others that included a combination of data that included length of the entire URL, number of dots in the URL, length of the hostname and using a technique called "bag-of-words", a representation of each token in the hostname and path portion of the URL that is delimited by characters such as "/", "?", ".", "=", "-" and " ").

The Host-based features used include information gathered from WHOIS such as the registration date, registrant, registrar; information about the IP address from DNS servers such as if the IP of the A, MX and NS records are in the same Autonomous System, are they hosted on disreputable ISP's; information about the geographic location of the IP; as well as the connection speed and if the URL is already on a blacklist.

The URL data they collected was provided from a webmail provider for the malicious URL, and the benign URL's from Yahoo, collecting over 2 million URL's over 100 days.

These researchers ran tests, comparing batch learning to continuous online learning. They found that using a continuous learning model provided much better classifiers with an error rate on their best performing model of around 1%. Limitations were found with the batch methods, by the number of training examples that could be fit into memory in a single interval.

Towards detection of phishing websites on client-side using machine learning based approach

In this paper (Jain & Gupta, 2017) the authors proposed a slightly different approach than above, using URL features as well as information from the source code of the web page.

Overall, the authors used 19 features extracted from the two sources which fit into one of five categories. The categories and number of features the used are: URL Forgery – 8 features; Fake Login form – 1 feature; Hyperlink information – 6 features; Copied CSS – 1 feature and Fake Web identity – 3 features.

Some of the URL features that are used are the presence of suspicious words in the URL, such as security, login, signin, etc., that are used to gain user trust; presence of special symbols in the URL such as "@" and "-"; count of number of times HTTP appears in the URL; does a brand name appear in the URL, in a place that it should not, as well as some features in the above referenced paper "Learning to Detect Malicious URLs".

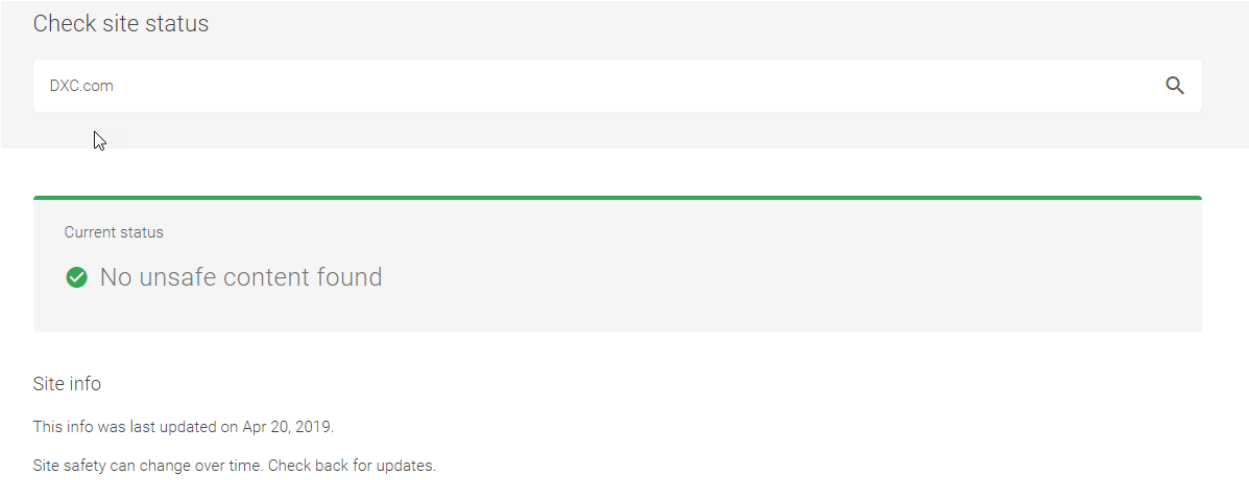
Some of the source code features used by the authors include the linking of CSS to an external web site, as this leads to malicious website being able to copy the CSS from the known benign site, for easier mimicry; number of hyperlinks on a site, as legitimate sites typically contain many separate links, whereas a phishing site would be smaller and not contain as many pages; does the page link to a different website and the presence of empty or null hyperlinks.

Training data consisted of 2141 phishing websites gathered from Phishtank and Openphish and 1918 benign sites gathered from Alexa, payment gateway links and other top banking websites, and included sites from various languages such as Portuguese, Hindi, English, etc.

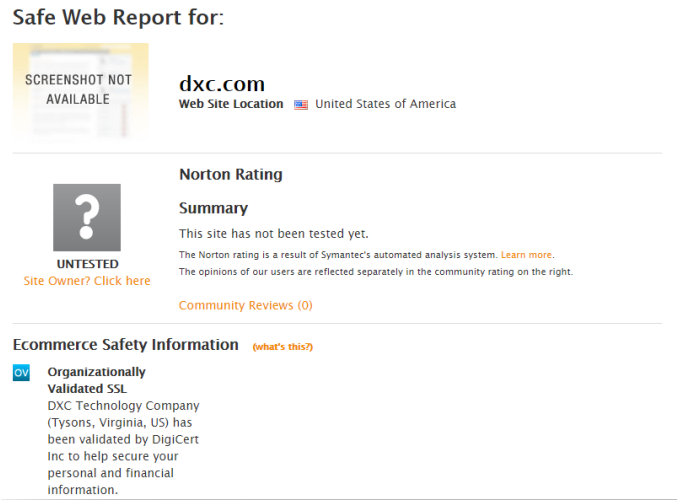
The researchers use WEKA with a 10-fold cross verification with a 90/10 split for training versus testing. Using several different classifiers, the researchers found that Random Forrest outperformed others, with a TPR of 99.39%, FPR of 1.25% and 99.09% Accuracy.

Online Resources for URL Checking

Several online resources exist to check the validity of URLs. Google provides "Safe Browsing." By navigating to the site https://transparencyreport.google.com/safe-browsing/search?url= , a user can enter in a URL and determine if Google has detected that it is an unsafe website. For example, testing a website DXC.com returns the following:



Norton also provides "Safe Web." It works in the same manner as Google Safe Browsing. A user enters in a URL and Norton will tell you if it is safe or not. Using the same URL as above, DXC.com returns:



As can be seen from these two examples, different information is returned. Google states that the site at DXC.com has "No unsafe content found" however, Norton says the site hasn't been tested yet but does provide that it has a valid SSL certificate. If the user only uses Norton, they cannot have much reliability into the site being malicious or not.

One of the problems with online systems, is they store the information of a site at any given time, and do not crawl and rank the site upon demand. This leads to the same issues as having a blacklist, they need to be updated and the prevalence of new malicious webpages each day can lead to the online URL checkers having outdated information. Having a machine learning model that has been trained and can predict on the fly provides a valuable resource to end-users. One of the benefits of using a model such as this is that it automates the checking for the end-user, possibly by intercepting the URL as they click on the link and making a determination, without the end-user needing to go to a third-party website, which can have outdated information, and entering the URL by hand. Automating this process can provide better protection for the user that is not trained to examine URL's beforehand and maybe forgetful and go to the URL without going to Google Safe Browsing or Norton Safe Web.

System Design and Implementation

This project was designed and executed inside of VM running in VMware Workstation. The VM is running Ubuntu 18.04 with 8GB of memory with dual 8-core processors. The code was written and executed using the PyCharm IDE. The code is written in Python and uses Scikit-learn to run the experiments. The output from the training and testing will be measured using Detection Rate, False Positive Rate, and Overall Accuracy will be calculated. Detection

Rate will be calculated as: TP / (TP + FN). False Positive Rate will be calculated as FP / TN+FP. The Overall Accuracy will be calculated as (TP + TN) / (TP + TN + FP + FN).

URLs for this project were obtained from "Canadian Institute for Cybersecurity" at the University of New Brunswick ("URL dataset", n.d.). The downloaded data contained several lists of URLs. I chose to use the ones for the Benign, Malware, Phishing and Spam lists. The benign lists contained 35,378 URLs, while the combined list malicious URLs contains 33,530 URLs. This set of data was created for the paper "Detecting Malicious URLs Using Lexical Analysis" (Mamun, Rathore, Lashkari, Stakhanova & Ghorbani, 2016)

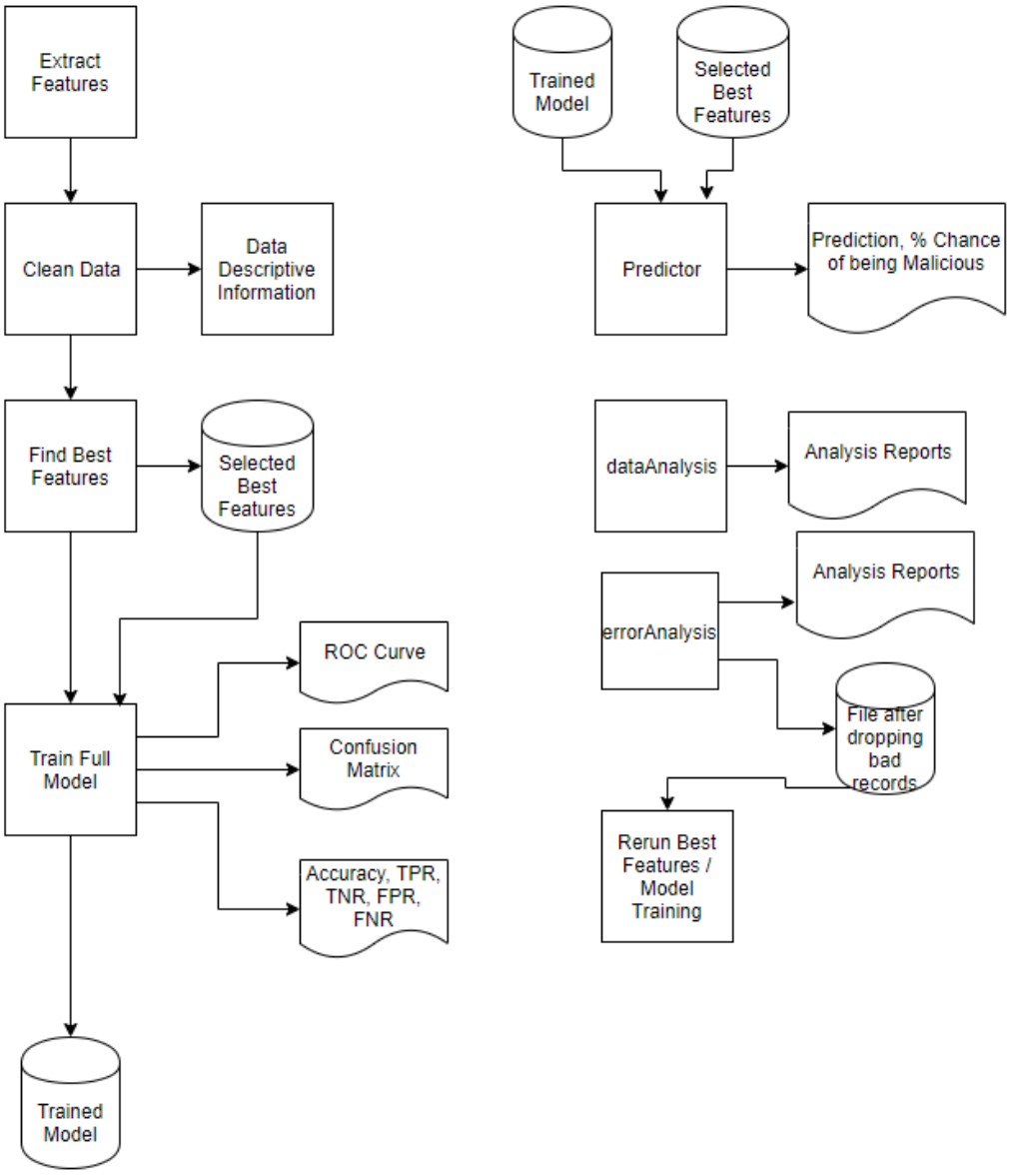
There are several modules involved in this project. The datasets and modules can be found at: https://github.com/garner007/url-classification-system-ml (Garner, 2019)

For this project, the goal is to predict the probability that a given URL will be either benign or malicious with a binary classifier. In the project, I used 10-fold cross-validation to train and test the model. Initially, 31 features are extracted from the URL, to include both lexical and host-based features. The initial base for the vectorizer came from the GitHub repo: https://github.com/Anmol-Sharma/URL_CLASSIFICATION_SYSTEM (Sharma, 2019) The initial vectorize included 22 features, which were expanded to include an additional 9.

The features selected include the presence of special characters that allow the malicious URL to attempt to mask its appearance. For example, using the "@" symbol that invalidates all the characters before it or the "- "to use a known brand name but appending additional words after it to make it appear legitimate. Other features selected for use include checking to see if the domain contains HTTPS, indicating that they may use SSL; checking the top 1 million pages on Alexa to see if the page is ranked, and checking the age of the domain, as phishing pages are typically short-lived. Research has been done into these various features and are included in

several different research articles including "Towards detection of phishing websites on client-side using machine learning based approach" (Jain & Gupta, 2017) and "Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages" (Canali, Cova, Vigna & Kruegel, 2011), among others.

The diagram below shows an overview of the training system and the predictor.



The program flows as follows:

The extract features module reads in the Malicious and Benign URLs and creates a combined vectorized file of them. All the URLs are read in from each of the files and randomized before being processed through the vectorizer, to shuffle the malicious and benign together. Following this, it was found that there was some bad data, such as "|" (pipe) symbols that we used as a field delimiter. Several other delimiters were tested for use, and each had a set of issues, so the "pipe" was used. Three records were dropped due to this issue during the clean data step. During this step, the country for the URL from the vectorizer is changed from a literal name to a numeric, and a new file is written out the "cleaned data."

From this, it was noticed that there are duplicate records present. To fix the duplicate records, a module was created to read in the "cleaned data" and load it into a Pandas data frame and remove the duplicates, writing out an additional new "de-duped clean data" file.

In the Find Best Features step, the cleaned data with the removed duplicates is processed through a training / testing step N times. The value for N is determined by the number of original features; for this exercise, there are 31. It will process iteratively over the number of features, to determine the best number of features to use to produce the best accuracy. This is determined by using 10-fold cross validation for each of the selected feature sets and selecting the one with highest accuracy. A mask of these selected features is written out, to be used to train the classifier model.

Using the features selected the classifier is run again, with 10-fold cross-validation to produce a model. This model is saved to be used in the predictor. The output from the full classifier module also includes the ROC curve, accuracy metrics of the classifier and confusion matrix.

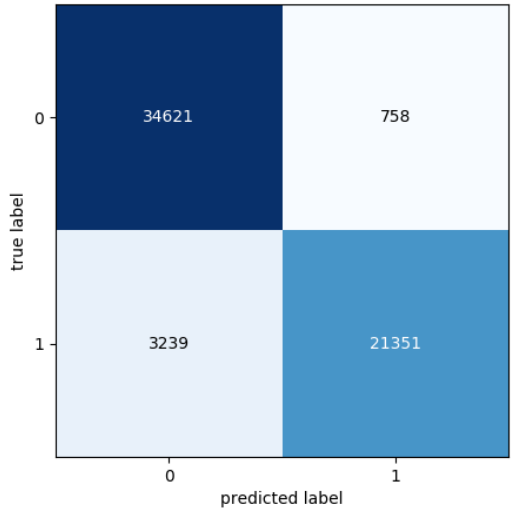
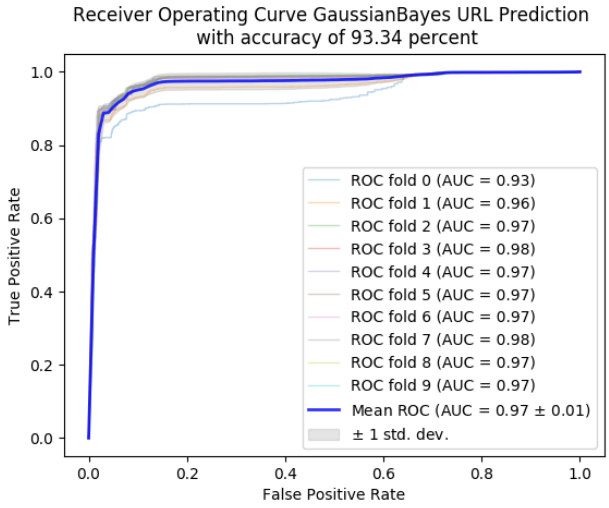
For the classifier, a Naïve Bayes classifier was used from the Scikit-learn. Naïve Bayes is a set of algorithms that apply Bayes theorem. It combines the probability that each of the features, independent of any of the other features could belong to the class, either benign or malicious. For this, the Gaussian Naïve Bayes was implemented. This version assumes that each of the features has a Gaussian distribution.

For comparison, the project will also run a decision tree classifier and random forest, also in Scikit-learn. A decision tree classifier creates what can be an easier to understand classifier, by creating a model of simple decision rules from the data can than be visualized and require less data preparation. A random forest classifier creates a number of decision trees on subsets of the data and averages out the predictions to make a decision. These two classifiers will be run on data that has been corrected of some of the misclassified data, that will be discussed later in the paper, in the **Analysis of Classification Errors** section.

Results

File Statistics:

	Original	After Duplicates Removed
	68905	59961
Benign	35375	35375
Malicious	33530	24586

Confusion Matrix	ROC
	

Testing Results – 16 Features		**Testing Results – All Features**	
True Positives	34621	True Positives	34535
False Positives	758	False Positives	844
True Negatives	21351	True Negatives	18620
False Negatives	3239	False Negatives	5970
Average Accuracy	93.34%	Average Accuracy	88.64%
Overall Accuracy	93.33%	Overall Accuracy	88.64%
True Positive Rate	91.44%	True Positive Rate	85.26%
True Negative Rate	96.57%	True Negative Rate	95.66%
False Positive Rate	3.43%	False Positive Rate	4.34%
False Negative Rate	8.56%	False Negative Rate	14.74%
Precision	97.86%	Precision	97.61%

During the process to find the best features, the system optimized and selected 16 features to include for the highest accuracy rate. Those features include:

length_of_url	number_of_dots	number_of_hyphens_in_domain
length_of_directory	length_of_domain	words_in_domain
has_alexa_rank	country_code	words
length_of_largest_domain_token	length_of_largest_path_token	len_of_filename
num_dot_in_filename	num_delims_in_filename	length_of_arguments
length_of_largest_variable		

Using these 16 features provided for accuracy of 93.34% compared to an accuracy of 88.64% using all the features. The False Positive using all features increases to 844, and the False Negatives increases significantly to 5970.

Analysis of classification errors

False positives (FP) occur when a benign URL is misclassified as a malicious one. In analyzing the statistical measurements of the benign and malicious URLs, the following appears to be the major causes of FP.

The largest number of misclassification issues come from the num_delims_in_filename and the country_code. Seven hundred twenty-one of the Benign URLs have a num_delims_in_filename less than 3, which is the mean for benign URLs. Two hundred seven of the misclassified benign URL's have something other than 225 for the country code.

False Negatives occur when a malicious URL is misclassified as a benign one. In analyzing the data, I found that 1528 have a country code of 225, which is the United States. I also found that 1934 of them are on URL's that are ranked in the top 1 million web pages on Alexa. There is an overlap of 1213 URL's that are classified as False Negatives that are in the United States and have an Alexa Ranking. Several possibilities can have occurred. In looking at the actual URLs that are in this overlap, several of them sprung out that belong to Amazon and to

Ancestry.com. Counting the URL's that belong to this group, I found 928 entries. This leads me to believe there is a possible issue with the data that I am using to build my classifier from, which is skewing the results of the False Negatives. A sampling of the URL's that belong to either Amazon or Ancestry are below, and can also be found on the GitHub repository (https://github.com/garner007/url-classification-system-ml) as a CSV titled 'FN_amazon_ancestry.csv'

http://amazon.co.uk/gp/redirect.html/ref=amb_link_18295865_6/202-2513040-1206232?location=http://www.
http://ancestry.co.uk/search/db.aspx?dbid=28596
http://amazon.co.uk/s/ref=amb_link_22202665_32/202-2513040-1206232?ie=UTF8&search-alias=kitchen&fi
http://ancestry.co.uk/search/db.aspx?dbid=15680
http://amazon.co.uk/b/ref=amb_link_39721465_3/202-2513040-1206232?ie=UTF8&node=293663011&pf_rd_
http://amazon.co.uk/b/ref=amb_link_36810965_26/202-2513040-1206232?ie=UTF8&node=577870&pf_rd_m
http://amazon.co.uk/s/ref=amb_link_38340765_10/202-2513040-1206232?ie=UTF8&search-alias=music&fie
http://ancestry.co.uk/search/db.aspx?dbid=27463
http://amazon.co.uk/gp/product/B000PDZHZ0/ref=br_nf_5_1/202-2513040-1206232?pf_rd_m=A3P5ROKL5A
http://ancestry.co.uk/search/db.aspx?dbid=18036
http://amazon.co.uk/b/ref=amb_link_18203665_25/202-2513040-1206232?ie=UTF8&node=560884&pf_rd_m
http://amazon.co.uk/gp/product/B000GFRIJA/ref=amb_link_36590765_4/202-2513040-1206232?pf_rd_m=A
http://ancestry.co.uk/search/db.aspx?dbid=19793
http://amazon.co.uk/s/ref=sr_nr_p_4_2/202-2513040-1206232?ie=UTF8&rh=n:560858

In researching further into the original source of data, there are other URL's that I find, belonging to Amazon such as "astore.amazon.co.uk" that are classified as spam / malicious, as shown below. While these are not being misclassified as False Negatives, it does lead me to believe the classifier is not being trained on good data. Including the "astore.amazon" and other Amazon links, the count rises to 1515.

http://astore.amazon.co.uk/allezvinsfrenchr/detail/1904010202/026-8324244-9330038
http://archive.salisburyjournal.co.uk/2007/3/6/306369.html

The original data that was used for the development of this project is available on the GitHub link in the Data folder. The False Negative and False Positive dataset can also be found on the GitHub link.

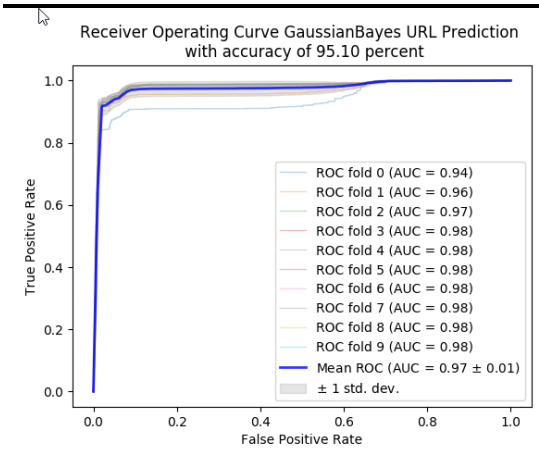
Removing the additional misclassified links, and rebuilding the classifier, the model produces much better results as shown below.

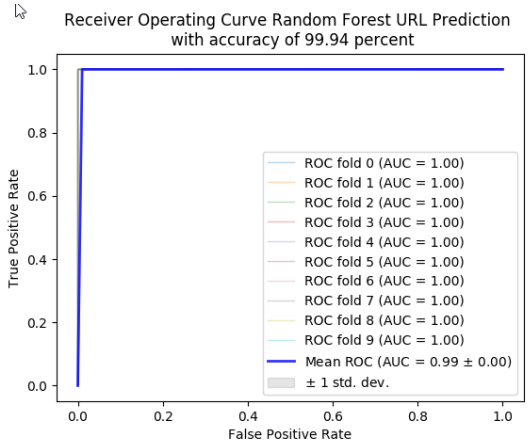
After Removal		**Original Results**	
True Positives	34844	True Positives	34621
False Positives	535	False Positives	758
True Negatives	20841	True Negatives	21351
False Negatives	2334	False Negatives	3239
Average Accuracy	95.10%	Average Accuracy	93.34%
Overall Accuracy	95.09%	Overall Accuracy	93.33%
True Positive Rate	93.72%	True Positive Rate	91.44%
True Negative Rate	97.49%	True Negative Rate	96.57%
False Positive Rate	2.51%	False Positive Rate	3.43%
False Negative Rate	6.28%	False Negative Rate	8.56%
Precision	98.49%	Precision	97.86%

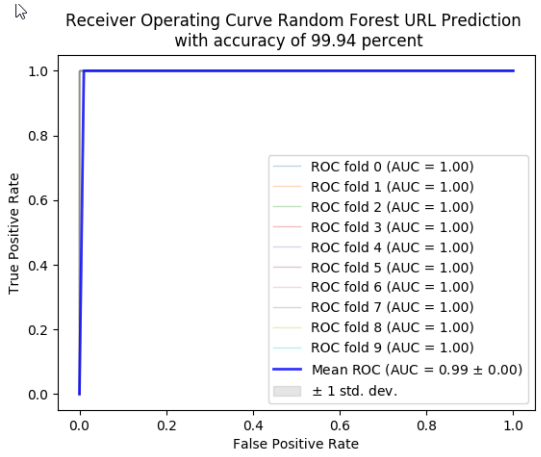
Comparison to Random Forest and Decision

After clearing the misclassified links that were found, a comparison was run against other two other classifiers that are available in Scikit-learn, Decision Tree and Random Forest.

ROC

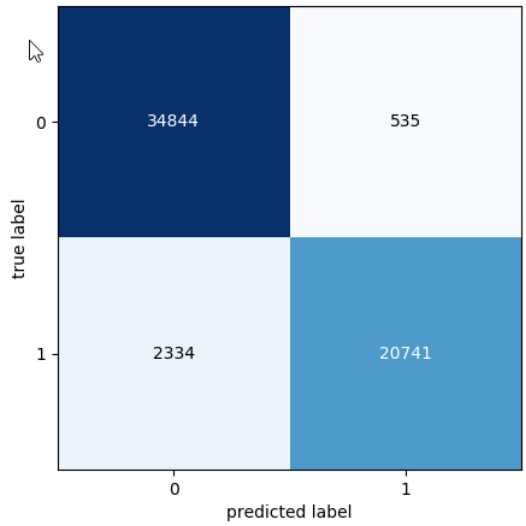




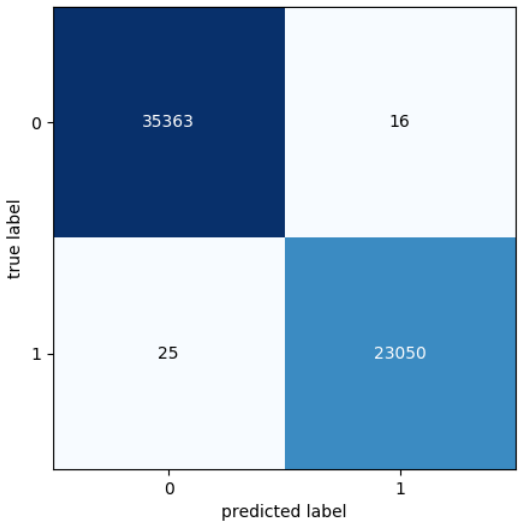


Confusion Matrices

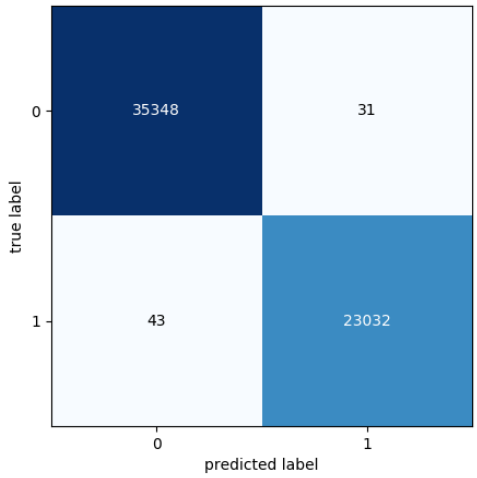
Gaussian Bayes



Random Forest



Decision Tree



Gaussian Bayes		Decision Tree		Random Forest	
True Positives	34844	True Positives	35347	True Positives	35363
False Positives	535	False Positives	32	False Positives	16
True Negatives	20841	True Negatives	23023	True Negatives	23050
False Negatives	2334	False Negatives	52	False Negatives	25
Average Accuracy	95.10%	Average Accuracy	99.88%	Average Accuracy	99.94%
Overall Accuracy	95.09%	Overall Accuracy	99.87%	Overall Accuracy	99.93%
True Positive Rate	93.72%	True Positive Rate	99.88%	True Positive Rate	99.93%
True Negative Rate	97.49%	True Negative Rate	99.87%	True Negative Rate	99.93%
False Positive Rate	2.51%	False Positive Rate	0.13%	False Positive Rate	0.07%
False Negative Rate	6.28%	False Negative Rate	0.12%	False Negative Rate	0.07%
Precision	98.49%	Precision	99.91%	Precision	99.95%
		Nodes	243		
Time to Run:	2.746 seconds	Time to Run:	9.421 seconds	Time to Run:	9.517 seconds

Discussion

The data originally contained 66,905 URLs, distributed almost evenly between malicious and benign. It was found that there were 8944 duplicate malicious URL's that were skewing results and were removed, leaving 24586 malicious URL's. In analyzing the results obtained, the Gaussian Bayes classifier was able to correctly classify malicious and benign URLs with an average accuracy of 93.34% with the data used. In the analysis of the False Positive and False Negative results, I found that there are URL's that are mislabeled as malicious which should have been labeled as benign. To test this, these were removed, and the testing was rerun. Overall accuracy increased from 93.34% to 95.10%; False Positive rate decreased from 3.43% to 2.51% and False Negative Rate decreased from 8.56% to 6.28%.

In comparison to the Random Forest and Decision tree, the Gaussian Bayes classifier did not perform as well, with an accuracy rate between 4.7% and 4.8% less than the Decision Tree

and Random Forest, respectively. These differences I believe are due to the need to better prepare the data for use in the Bayes classifier, as well as more analysis on the dataset to clear out mislabeled data. The Bayes classifier prefers data of one type, whereas neither the Random Forest nor Decision tree are susceptible to issue with different data types.

The downsides to the Random Forest and Decision tree models are the time it takes to train the classifier, however minimal, on this small set of data, it took around 7 seconds longer to create these two models. This timing would certainly increase with additional data points.

The discovery of mislabeled data shows the advantages of having an automated system available to analyze and make predictions on the data and shows the difficulties in trying to classify URL's by hand. With additional research and correctly classified data points, I believe the Bayes classifier would improve in accuracy to provide better results. The datasets used for this are available on the GitHub repository for additional research and analysis, to determine any additional misclassified data which may better tune the classifier.

Conclusion

In this report, I have attempted to build and test a data mining classifier to analyze URL's and make a prediction as to whether the URL points to a malicious or benign website. Information was provided on other research that has also been done in this area, using other classifiers with similar data features. Comparing my results to what the researchers found in "Towards detection of phishing websites on client-side using machine learning based approach" we can evaluate a comparable difference to what I found with my testing:

	Gaussian Bayes	Testing Random Forest	Researchers Random Forest
Average Accuracy	95.10%	99.94%	99.09%
True Positive Rate	93.72%	99.93%	99.39%
False Positive Rate	2.51%	0.07%	1.25%

Similar results were found by the researchers and their Random Forest classifier, with their classifier achieving around 4% better accuracy results as compared to the Gaussian Bayes that I created. Although I don't have their dataset to analyze, my belief remains that the quality of the data that is fed into the classifier has a greater impact on the Bayes classifier as well as the preparation of the data to clean and normalize it. The researcher in the paper "Learning to Detect Malicious URLs" do not provide exact results for a comparison such as above, but they were able to build classifiers that are continually trained and provide for an error rate of around 1%.

I found that a discrepancy existed in the data I used for training with data labeled as malicious. In the original researched that produced the data, the researchers included it as spam / malicious URLs. I believe that with more relevant data, this classifier could produce better results. Additional data should also be obtained and evaluated for use in future studies, with emphasis put onto classifying the input URL better.

References

Basnet, R., & Doleck, T. (2015). Towards Developing a Tool to Detect Phishing URLs: A Machine Learning Approach. 2015 IEEE International Conference On Computational Intelligence & Communication Technology. doi: 10.1109/cict.2015.63

Canali, D., Cova, M., Vigna, G., & Kruegel, C. (2011). Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages. Proceedings Of The 20Th International Conference On World Wide Web - WWW '11. doi: 10.1145/1963405.1963436

Chakraborty, G., & Lin, T. (2017). A URL address aware classification of malicious websites for online security during web-surfing. 2017 IEEE International Conference On Advanced Networks And Telecommunications Systems (ANTS). doi: 10.1109/ants.2017.8384155

Garner, Christopher, URL-Classification-System-ML (2019), GitHub repository, https://github.com/garner007/url-classification-system-ml

Jain, A., & Gupta, B. (2017). Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems, 68(4), 687-700. doi: 10.1007/s11235-017-0414-0

Ma, J., Saul, L., Savage, S., & Voelker, G. (2011). Learning to detect malicious URLs. ACM Transactions On Intelligent Systems And Technology, 2(3), 1-24. doi: 10.1145/1961189.1961202

Mamun, M., Rathore, M., Lashkari, A., Stakhanova, N., & Ghorbani, A. (2016). Detecting Malicious URLs Using Lexical Analysis. Network And System Security, 467-482. doi: 10.1007/978-3-319-46298-1_30

Mašetic, Z., Subasi, A., & Azemovic, J. (2016). Malicious Web Sites Detection using C4.5 Decision Tree. Southeast Europe Journal Of Soft Computing, 5(1). doi: 10.21533/scjournal.v5i1.109

Sharma, A. (2019). Anmol-Sharma/URL_CLASSIFICATION_SYSTEM. Retrieved from https://github.com/Anmol-Sharma/URL_CLASSIFICATION_SYSTEM

Shi, Y., Chen, G., & Li, J. (2017). Malicious Domain Name Detection Based on Extreme Machine Learning. Neural Processing Letters, 48(3), 1347-1357. doi: 10.1007/s11063-017-9666-7

URL dataset. Retrieved from https://www.unb.ca/cic/datasets/url-2016.html