Malicious URL Detection Using Supervised Machine Learning Techniques

Vara Vundavalli
Department of Information Technology, Kennesaw State University

Farhat Barsha
Department of Computer Science and Engineering, Military Institute of Science and Technology

Mohammad Masum
Analytics and Data Science Institute, Kennesaw State University

Hossain Shahriar
Department of Information Technology, Kennesaw State University

Hisham Haddad
Department of Computer Science, Kennesaw State University

CCS CONCEPTS

• **Security systems** → URL detect using machine learning; • **Computing methodologies** → Logistic Regression, Naïve Bayes, Neural Networks.

KEYWORDS

Blacklist, Malicious URL detection, Logistic Regression, naïve Bayes, Neural Networks

ACM Reference Format:
Vara Vundavalli, Farhat Barsha, Mohammad Masum, Hossain Shahriar, and Hisham Haddad. 2020. Malicious URL Detection Using Supervised Machine Learning Techniques. In *13th International Conference on Security of Information and Networks (SIN 2020), November 04–07, 2020, Merkez, Turkey*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3433174.3433592

1 INTRODUCTION

Malicious Websites are a primary means of Internet crimes. Attackers may attempt to get private information through malicious URLs [1]. These malicious URLs can cause untrustworthy exercises, for example, burglary of private and secret information, ransomware installation on the client gadgets that brings about massive misfortunes consistently all around.

With the progression of interacting social platforms, many permit its customers to broadcast unauthorized URLs—a lot of these URLs identified with the advancement of business and self-ad. However, a portion of these exceptional source locators can represent a susceptible risk to inexperienced client. The naive customers who utilize malicious URLs [3] are going to confront significant security dangers started by the rival. The check of URL is fundamental to guarantee that the client must keep from visiting malicious websites. Numerous procedures have proposed to distinguish malicious URLs. One of the main characteristics is - a component must permit the benign URLs that are mentioned by the customer, avert the malicious URLs before reaching to users and alert users. Instead of depending on syntactical properties of the URLs, a system should consider properties of URL. Conventional procedures, for example, Black-Listing [2] and Heuristic Classification [3], can recognize these URLs and identify them before reaching to the client.

Blacklisting [1] is one of the main approaches in identifying malicious URLs. Black-List is a database that contains all URLs which are known as malicious. A database query played out for each opportunity. The structure goes over another URL. Here, the new URL will be organized and analyzed with each recently known malicious URL in the blacklist. The update must make in blacklist at whatever point the structure runs over another malicious URL. The procedure is recurrent, slow, and computationally increased with ever-expanding new URLs.

The leading Internet organizations have started one of the cooperative work, for example, Google, Facebook, alongside a large number of the new businesses to manufacture a single stage that works all together for one reason for keeping the inexperienced clients from the malicious URLs [1]. Vast numbers of these online organizations utilize databases, which can store many URLs [25], and refine URLs usually. Our motive is to find the benign and malicious websites from datasets, which has hundreds and thousands of URLs. For detecting the malicious websites, we have used different types of machine learning techniques.

The paper is organized as follows: Section II discusses related work. Section III shows the dataset and discussed methodologies used to identify Malicious URLs. Section IV evaluates the approaches with datasets and compares accuracies of all the three methods. Finally, Section V concludes the paper.

2 RELATED WORK

Following the guidelines throughout this template will also improve the accessibility of your manuscript and increase the audience for your work. Ensure that heading styles are applied as instructed, tables are created using Word's table feature (rather than an image), figures have a text equivalent, and list styles are applied as instructed.

Different technologies have been developed to detect malicious URL for the purpose of securing client's information. Below is an overview of related works. In "Fake website detection using regression" authors represent the complete literature overview of

SIN 2020, November 04–07, 2020, Merkez, Turkey

ACM ISBN 978-1-4503-8751-4/20/11…$15.00
https://doi.org/10.1145/3433174.3433592

AI-based malicious URL recognition, which talks about the most significant progress towards malignant URL recognition [5]. Towards the starting stages blacklisting, regular expression, and signature matching methodologies are most normally utilized for malicious URL identification. These techniques fail to recognize new or variations of the current URL. Besides, the signature database must be refreshed now and again to deal with new examples of malicious URLs. Afterward, machine learning calculations were utilized to adequately recognize new kinds of malicious URL.

Ordinary machine learning calculations rely upon include designing to remove a rundown of highlights from the URL. This element building requires broad space information on URL in cyber security and a rundown of good highlights must be deliberately picked through component determination. There are different sorts of highlights were utilized in the distributed works for malicious URL identification. This includes blacklist features [6, 7], lexical highlights [7–9] have based highlights [10, 11], content highlights, setting and notoriety-based highlights [12–14]. Boycott highlights are assessed through checking its quality of a URL in a boycott. This could fill in as a solid element in recognizing malicious URLs. Lexical highlights are evaluated through the string properties of the URL - number of exceptional characters, length of URL, and so forth.

This incorporates data identified with WHOIS data, IP Address, Geographic area, and so on. The substance highlight is gotten from the HTML and JavaScript when a clueless client visits a page through the malicious URL. Content highlights incorporate data identified with their positioning, prevalence scores, and sources of sharing. Many existing investigations have utilized separate component classification and just as a blend of these features which was persistently decided through domain experts.

Feature engineering is a daunting task with considering the security threats. For example, obtaining context-based features consumes more time and it is high risky too. Moreover, feature selection requires extensive domain knowledge. The information which is obtained directly from the raw URL was well-known approach [7, 15]. From the published results, obtaining the lexical feature is easier in comparison to other features and it gave good performances [16]. Analytical characteristics of the website URL sequence like the length of the URL, phase-frequency methods such as TDM TF-IDF and n-gram characteristics [7, 16, 17] are also considerable. All these features are not useful in extracting sequential order and semantics of URL. This overlooks information from unseen characters. Moreover, detection solution based on feature engineering with machine learning can be damaged by an adversary.

Deep learning with character level embedding has been used for malicious URL detection. In [18] comparison of a comprehensive review of deep learning with character level embedding and conventional machine learning with feature engineering methods for malicious and phishing URL detection was discussed. Deep learning designs executed considerably in contrast to the conventional methods. For phishing URL detection, Recurrent Neural Network and Long Short-Term Memory is used [19]. For comparative analysis, lexical features and statistical URL analysis was used with random forest classifier. Both models performed well; but compared to machine learning, long short-term memory performed well.

In [20] a Convolution Neural Network was used with character level Keras embedding for identifying pseudo URLs, tracking the file, and registry keys. This study showed how a unique deep learning architecture could be used on different cyber security problems. We estimate the execution of different deep learning designs for malicious URL detection in this paper.

For getting more understanding about the URL, without excavating too profound gaps at one spot a few assets are beneficial, one can utilize the Lexical Specialties as the characterizing constraints in the Detection of Malicious URLs [4] by utilizing the Noticeable Characteristics, it is conceivable to analyze the Malicious Small URLs. The Social Network trolls, for example, Twitter and Facebook, use these sorts of disturbing aspects to know whether to check; in fact, these frameworks are called Recommendation frameworks. We can infer four distinct classifications of confusing systems so we can recognize the considerate from malicious [6]. The four Obfuscation Characteristics are (1) Obfuscating the host with an IP address, (2) Obfuscating the host with another space, (3) Obfuscating with the huge hostnames and, (4) Domain misspelled [22].

Though various approaches were applied to detect malicious URL there was always some shortcomings. This research proposes a system based on Machine learning where three different techniques are used to detect malicious URL.

3 DATASET

The next subsections provide instructions on how to insert figures, tables, and equations in your document.

For this work, we have used the dataset from Kaggle website [23]. The principal task was gathering information. We have discovered some sites offering malicious links while surfing. The following step was about discovering clear URLs. We have used a dataset that was accessible and there was not any requirement for slithering. We also have assembled URLs out of which a large portion is malicious, and the remaining non malicious. We then use Logistic regression, Naive Bayes and Convolution neural network system to detect malicious URL and identify which algorithm gives better accuracy. Thus, the input of this experiment is a set of URLs and the output is good or bad based on the accuracy.

4 METHODOLOGIES

4.1 Logistic Regression

Logistic regression is a strategy for performing regression on a dataset that has clear cut objective values [7]. The logistic part is applied to change the linear sequence of the informative factors into probabilities. Logistic regression is the correct regression evaluation to perform when the dependent variable is combined. Like all regression examinations, logistic regression is a perceptive interpretation which is used to describe the information and to demonstrate the relationship between a binary term and nominal, ordinal, interval, or extent level independent elements. On the other hand, most of the logistic regressions are difficult to scrutinize. However, the Intellectuals Statistical tool will help us evaluate the analysis of the output effectively. The logistic function is described below in equation 1. Here, σ is the standard logistic function t is a linear

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	URL	URL_LENG	NUMBER_	CHARSET	SERVER	CONTENT_	WHOIS_C	WHOIS_ST	WHOIS_REGDATE	WHOIS_UI	TCP_CON	DIST_REM	REMOTE_I	APP_BYTE	SOURCE_A	REMOTE_	SOURCE_A	REMOTE_	APP_PACK	DNS_QUE	Type
2	M0_109	16	7	iso-8859-1	nginx	263	None	None	10/10/2015 18:21	None	7	0	2	700	9	10	1153	832	9	2	1
3	B0_2314	16	6	UTF-8	Apache/2.	15087	None	None	None	None	17	7	4	1230	17	19	1265	1230	17	0	0
4	B0_911	16	6	us-ascii	Microsoft-	324	None	None	None	None	0	0	0	0	0	0	0	0	0	0	0
5	B0_113	17	6	ISO-8859-	nginx	162	US	AK	7/10/1997 4:00	########	31	22	3	3812	39	37	18784	4380	39	8	0
6	B0_403	17	6	UTF-8	None	124140	US	TX	12/5/1996 0:00	########	57	2	5	4278	61	62	129889	4586	61	4	0
7	B0_2064	18	7	UTF-8	nginx	NA	SC	Mahe	3/8/2016 14:30	########	11	6	9	894	11	13	838	894	11	0	0
8	B0_462	18	6	iso-8859-1	Apache/2	345	US	CO	29/07/2002 0:00	########	12	0	3	1189	14	13	8559	1327	14	2	0
9	B0_1128	19	6	us-ascii	Microsoft-	324	US	FL	18/03/1997 0:00	19/03/201	0	0	0	0	0	0	0	0	0	0	0
10	M2_17	20	5	utf-8	nginx/1.10	NA	None	None	8/11/2014 7:41	None	0	0	0	0	2	3	213	146	2	2	1
11	M3_75	20	5	utf-8	nginx/1.10	NA	None	None	8/11/2014 7:41	None	0	0	0	0	2	1	62	146	2	2	1
12	B0_1013	20	6	utf-8	Apache	NA	US	Kansas	14/09/2007 0:00	########	0	0	0	0	0	0	0	0	0	0	0
13	B0_1102	20	6	us-ascii	Microsoft-	324	US	CO	22/11/2016 0:00	23/11/201	0	0	0	0	0	0	0	0	0	0	0
14	B0_22	20	7	utf-8	None	13716	GB	None	11/10/2002 0:00	########	16	6	8	1492	20	20	2334	1784	20	4	0
15	B0_482	20	6	ISO-8859-	nginx	3692	None	None	14/11/2002 0:00	19/04/201	25	19	4	3946	35	29	16408	4746	35	10	0

Figure 1: Snapshot of the data-set from Kaggle

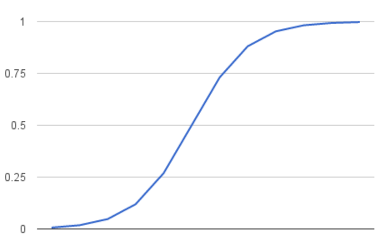

Figure 2: Graphical Representation of Logistic Function

function of a singular explanatory variable x.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Equation 1: The Logistic Function

The formula below is used for transforming the typical linear regression, where x is the predictor variable; the β values are the linear parameters.

$$f(x) = \beta_0 + \beta_1 x$$

Equation 2: Linear Regression Equation

The subsequent condition is given in Equation 3. In this equation, p(x) gives the probability that equation 1 represents the input sample.

$$p(x) = \frac{1}{1 + e^{-\beta_0 - \beta_1 x}}$$

Equation 3: Logistic Regression

The target tests are the rows of a sparse matrix with an enormous number of sections. These segments encode the calculations of all the conceivable system calls, activities, ways, and results that can happen. Utilizing sparse matrices advances the presentation and memory utilization of the model. With sparse matrices, the runtime and memory use is bound by the quantity of non-zero sections of the matrix and not the components of the matrix. Due to this only being 1 malicious application on the computer, the classes 0 and 1 are imbalanced.

4.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) will have one convolutional layer (generally with a subsampling step). It is later followed by one correlated layer in a standard multi-layer neural network. The designing of a CNN is expected to misuse the 2D structure of a data picture (or other 2D data, for instance, a discourse signal). This is encouraged with nearby connections and tied loads followed by pooling that realizes understanding invariant features. Another benefit of CNNs is that they are easier to formulate and have various fewer parameter than totally connected networks with a comparable number of hidden layers. At present, we will look at the structure of a CNN and the back-propagation estimation to calculate the incline concerning the parameters of the model to use inclination-based improvement.

CNN contains a couple of convolutional and subsampling layers, on the other hand, followed by totally related layers. The involvement of a convolutional layer is a (m x m x r) where r is the number of channels, m is the length and breadth of the model. The extent of the channels attempts to raise the associated confidential structure, which is each convolved with the image to make k incorporate maps of size m−n+1. Each guide is then subsampled regularly with mean or max pooling over (p x p) adjacent units where p reaches out between 2 for little pictures (for instance, MNIST) and is typically not more than 5 for more significant sources of information. An extra inclination tendency and sigmoidal nonlinearity is applied to every part of the map, before or after the subsampling layer.

Back propagation:

Let $\delta^{(l+1)}$ = error term for the (l+1)-st layer in the network

Cost function = J(W,b;x,y)

Where (W,b) = The parameters

(x,y) = the training data and label pairs.

If the l-th layer is densely connected to the (l+1)-st layer, then the error for the l-th layer is computed as $\delta^{(1)} = ((W^{(l)^T}) \, \delta^{(l+1)}) \,.\, f' \, (z^{(l)})$ and the gradients are

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T$$

If the l-th layer is a convolutional and subsampling layer, then the error is propagated through as

$$\delta_k^{(l)} = upsample \left(\left({W^{l}}_k^{(T)} \right) \delta_k^{(l+1)} \right) \,.\, z_k^{(l)} f'$$

Where k indexes the filter number and $z_k^{(l)} f'$ is the derivative of the activation function. The upsample activity must produce the error through the pooling layer by ascertaining the error with respect to every unit approaching the pooling layer. For instance, if we have mean pooling, at that point, the upsample consistently propagates the error for an individual pooling unit among the units which support in the previous layer. In max pooling, the unit which was picked as the maximum gets all the blunder since minimal changes in information would irritate the outcome just through that unit.

At long last, to ascertain the angle with respect to the channel maps, we depend on the fringe dealing with convolution activity again and flip the mistake lattice δ(l)k a similar way we flip the channels in the convolutional layer.

$$\nabla W_k^{(l)} J(W, b; x, y) = \sum_{i=1}^{m} a_i^{(l)} rot90\left(\delta_k^{(l+1)}, 2\right),$$

$$\nabla b_k^{(l)} J(W, b; x, y) = \sum_{a,b} (\delta_k^{(l+1)})\text{a, b}$$

where $a^{(l)}$ is the input to the l-th layer, and $a^{(l)}$ is the input image. The operation $a_i^{(l)} * \delta_k^{(l+1)}$ is the "valid" convolution between i-th input in the l-th layer and the error with respect to the k-th filter [21].

4.3 Naïve Bayes

Naive Bayes Rule is the base for machine learning and data mining strategies. This calculation is utilized to make designs with cautious measures. It gives better approaches for reviewing and getting information. It is utilized when information is large, and we need proficient yield contrasted with different strategies. The probability model for a classifier is a restrictive model over a reliant class variable. $P(C|F_1, \ldots\ldots F_n)$

Using Bayes' theorem

$$p\left(C|F_1, \ldots\ldots F_n\right) = p\left(C\right) p\left(\frac{F_1 \ldots\ldots F_n}{C}\right) \Big/ p\left(F_1 \ldots\ldots F_n\right)$$

4.3.1 Gaussian Naïve Bayes.

While managing constant information, standard speculation is that the constant characters identified with each class are scattered by a standard (or Gaussian) scattering. For example, accept the preparation information contains a consistent characteristic, x. The first part of the information by the class, and a short time later, process the mean and fluctuation of x in each class. Let μ_k be the mean of the characters in x related to class C_k, and let σ_k^2 be the Bessel variance of the qualities in x related to class C_k. Assume we have gathered some value v. At that point, the probability dissemination of v given a class C_k, $p(x = v \mid C_k)$ can be registered by connecting v to the condition for a typical dispersion parameterized by μ_k and σ_k^2.

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

Another typical method for managing persistent qualities is to use binning to discretize the component values, to obtain another plan of Bernoulli-appropriated highlights. A few works, in reality, prescribe this is as essential to apply Naive Bayes. However, the discretization may dispose of discriminative information [21].

4.3.2 Multinomial Naïve Bayes.

With a multinomial event model, examples address the frequencies with which certain events have been produced by a multinomial $(p_1, \ldots\ldots p_n)$ where p_i is the probability that event i happens). A component vector $x = (x_1, \ldots\ldots x_n)$is then a histogram, with x_i checking the occasion's event we observed in an example. This is the event model commonly utilized for report grouping, with events speaking to the event of a word in a single record. The probability of watching a histogram x is given by

$$p(x|C_k) = \frac{(\sum i\, x_i)!}{\prod i\, x_i} \prod_i pki^{x_i}$$

when stated in log-space, the multinomial naive Bayes classifier turn into a linear classifier:

$$Logp(\,C_k|x)\alpha log(p(C_k) \prod_{i=1} pki^{x_i})$$
$$= logp(C_k) + \sum_{i=1}^{n} x_i \,.\, \log p_{ki}$$
$$= \; b + w_k^T x$$

where $b = log\ p(C_k)\ and\ w_{ki} = \log p_{ki}$

In the training data, if the chances of occurring given class and feature value are zero, at that point, the recurrence-based probability gauge will be zero. This is because the likelihood gauge is legitimately corresponding to the number of events of an element's value. This is risky because it will clear out all data in different probabilities when they are duplicated. In this manner, it is frequently appropriate to consolidate a small example of occurring, called pseudo count. More likely, there will not be any chance of setting it to zero. This method of normalizing Naive Bayes is called Laplace smoothing when the pseudo count is 1, and in general case it is known as Lidstone smoothing [21].

4.3.3 Bernoulli Naïve Bayes.

In the multivariate Bernoulli event model, inputs represent the characteristics of independent Booleans. Like the multinomial model, Bernoulli Naive Bayes is famous for document classification tasks [10] in which using binary term occurrence is used instead of using term frequencies. If x_i is a Boolean describing the event or nonappearance of the i'th term from the dialect, at that point, the probability of a document given a class C_k is given by

$$p(x \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x^i} \,(1 - p_{ki})^{(1-x_i)}$$

where

p_{ki} = probability of class
C_k = generating the termx_i.

5 EVALUATION

We evaluated the dataset with three algorithms. They are Logistic regression, Neural Networks and Naïve Bayes (Gaussian, Multinomial, and Bernoulli). We have started by scaling the data then split it into training and test sets. A simple logistic regression model is used to baseline the accuracy to see if the performance is better using a neural network. We get Accuracy of 86.25%; Precision of 33.33%; Recall of 5.80% and F1 of 9.88%. So, our base model has 86% accuracy. However, poor precision and recall. From the result, we can see that while it is pretty good at predicting benign websites, it is poor at predicting the malicious websites. Now we see if we can improve this with a neural network. We willl use the standard SciKit learn class, starting off by using all the default values, before attempting to optimize it by adjusting its parameters. The result

Table 1: Confusion Matrix for Malicious URL website detection using Gaussian Naive Bayes

	Precision	Recall	f1-score	Support
0	1.00	0.16	0.28	475
1	0.13	1.00	0.23	60
Accuracy	0.67	0.27	0.25	535
Macro avg.	0.57	0.58	0.25	535
Weighted avg.	0.90	0.25	0.27	535

Table 2: Confusion Matrix for Malicious URL website detection using Multinomial Naive Bayes

	Precision	Recall	f1-score	Support
0	0.93	0.98	0.95	630
1	0.72	0.43	0.54	83
Accuracy	0.90	0.90	0.91	713
Macro avg.	0.82	0.71	0.75	713
Weighted avg.	0.90	0.91	0.90	713

Table 3: Confusion Matrix for Malicious URL website detection using Bernoulli Naive Bayes

	Precision	recall	f1-score	Support
0	0.99	1.00	1.00	475
1	1.00	0.95	0.97	60
Accuracy	0.99	0.91	0.99	535
Macro avg	1.00	0.97	0.99	535
Weighted avg	0.99	0.99	0.99	535

for neural network is as follows Accuracy = 88.32%; Precision = 61.29%; Recall = 27.54%; F1 = 38.00%.

While using Naïve Bayes method, we have used three methods to compare which method is having more accuracy. The results of confusion matrix are in Table 1 below.

This event model is particularly well known for short grouping messages. It has the advantage of unequivocally modeling the lack of terms. After this, it is evident that a multinomial NB classifier with recurrence counts shortened to one is not equivalent to a Naive Bayes classifier with a Bernoulli event model [21].

From the above table we can see that Gaussian Naive Bayes accuracy is quite slow. It seems BernoulliNB has the highest accuracy. It is because the data contains large categorical data, so Gaussian is not suitable in this case. Then, we compared between two modals BernoulliNB and MultinomialNB. MultinomialNB cares about how many times X value appears in our dataset while BernoulliNB only cares about whether X value appear or not. Some numerical columns make MultinomialNB accuracy decrease by the difference between their units.

First, we need to test the dataset in order to perform the techniques. Therefore, we examined the dataset if it is eligible for the testing

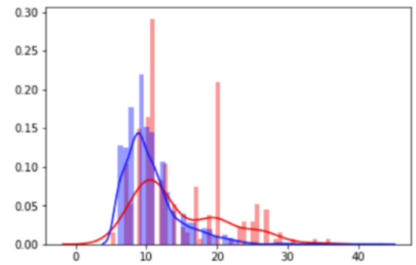

Figure 3: Dataset Eligibility Test

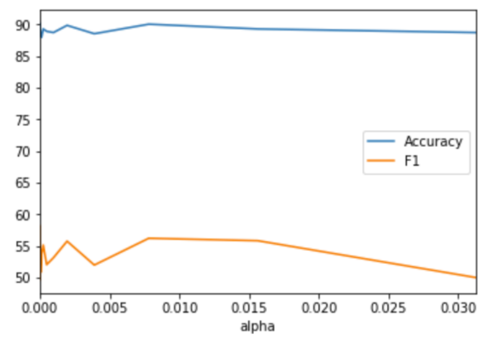

Figure 4: Logistic Regression

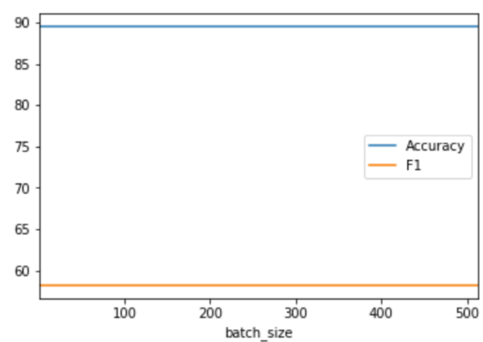

Figure 5: Accuracy and F1 score

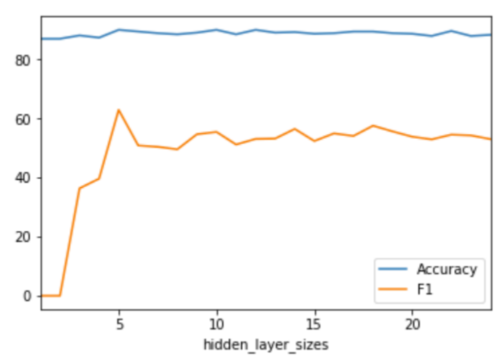

Figure 6: Accuracy after adding first layer

Table 4: Accuracy Comparison of Logistic Regression, Neural Networks and Naive Bayes.

	Logistic Regression	Neural Network	Naive Bayes
Accuracy	0.86	0.88	0.91

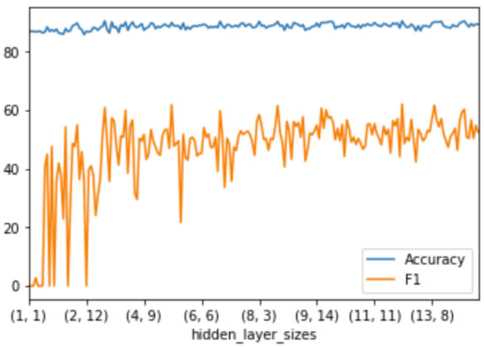

Figure 7: Accuracy after adding second layer

or not. In Figure 3, red bars indicate the malicious websites and we can also see some odd spikes. So, the dataset we have taken is eligible for testing. After that we applied logistic regression to find the accuracy. We took one set of data and performed logistic regression. The result is shown in the Figure 4. And then we take another set of data and find the accuracy which is shown in Figure 5. The set we took is not affecting the accuracy pretty much. So, we add layers to it to check if there are any changes taking place in accuracy. Figures 6 and 7 show accuracy after adding more layers.

6 CONCLUSION

Malicious URL detection plays a crucial role to secure cyber security applications and machine learning approaches are promising. In this paper, we led a complete and careful study on Malicious URL Detection utilizing AI strategies. Specifically, we offered a precise plan of Malicious URL detection from Machine learning viewpoint, and afterward point by point the discussions of existing tests for malicious URL detection, especially in the types of building new element representations, and structuring new systematic algorithms for improving the malicious URL detections.

In this work, we used datasets to classify malicious URL websites by using Logistic Regression, Neural Networks and different types of Naive Bayes algorithms. The results show that Naive Bayes algorithm performed better than logistic regression and neural networks in the dataset with extremely challenging distribution. Our future work includes exploring more datasets and comparing other machine learning techniques.

REFERENCES

[1] Immadisetti Naga Venkata Durga Naveen, Manamohana K, Rohit Verma, Detection of Malicious URLs using Machine Learning Techniques, March 2019, Retrieved from https://www.ijitee.org/wp-content/uploads/papers/v8i4s2/D1S0085028419.pdf
[2] Justin. Ma, Lawerence. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious websites from suspicious URLs," Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA: ACM, 2009, pp. 1245–1254.
[3] Mohammed Al-Janabi, Ed de Quincey, Peter Andras, "Using Supervised Machine Learning Algorithms to Detect suspicious URLs in online social networks", Proc. of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, https://dl.acm.org/citation.cfm?id=3116201.
[4] R.k. Nepali and Y. Wang, "You Look suspicious!!" Leveraging the visible attributes to classify the malicious short URLs on Twitter, Proc. of 49th Hawaii International Conference on System Sciences (HICSS), IEEE, 2016, pp. 2648-2655.
[5] G kumari, M Naveen kumar, A Mary Sowjanya, "Fake website detection using regression", International Journal of Advance Research in Science and Engineering, Vol. 6, Issue 08, 2017, Accessed from https://www.ijarse.com/images/fullpdf/1502607327_ijarse155.pdf
[6] Anh Le, Athina Markopoulou, Michalis Faloutsos, "PhishDef: URL Names Say It All", Proc. of IEEE International Conference on Computer Communications (INFOCOM), DOI: 10.1109/INFCOM.2011.593499, Published in 2011.
[7] Statistic solutions Advancement Through Clarity, "What is Logistic Regression?" https://www.statisticssolutions.com/what-is-logistic-regression/
[8] Kolari, P., Finin, T., & Joshi, A., SVMs for the Blogosphere: Blog Identification and Splog Detection, Proc. of AAAI spring symposium: Computational approaches to analyzing weblogs, March 2006, pp. 92-99.
[9] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M., "Identifying suspicious URLs: an application of large-scale online learning," Proc. of the 26th Annual International Conference on Machine Learning, June 2009, pp. 681-688. ACM.
[10] Chiba, D., Tobe, K., Mori, T., & Goto, S., "Detecting malicious websites by learning IP address features," Proc. of 2012 IEEE/IPSJ Applications and the Internet (SAINT), 12th International Symposium on, July 2012, pp. 29-39.
[11] McGrath, D. K., & Gupta, M., Behind Phishing: An Examination of Phisher Modi Operandi. LEET, 2008, 8, 4.
[12] Cao, J., Li, Q., Ji, Y., He, Y., & Guo, D., "Detection of forwarding based malicious urls in online social networks," International Journal of Parallel Programming, 2016, 44(1), 163-180.
[13] Choi, H., Zhu, B. B., & Lee, H., "Detecting Malicious Web Links and Identifying Their Attack Types," Proc. of 2nd USENIX Conf. on Web App Development, 2011, 11-11.
[14] Lee, S., & Kim, J., "WarningBird: Detecting Suspicious URLs in Twitter Stream," In NDSS, Feb 2012, Vol. 12, pp. 1-13.
[15] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2009. Identifying suspicious URLs: an application of large-scale online learning. Proc. of the Annual International Conference on Machine Learning. ACM, 681–688.
[16] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. 2010. Lexical feature-based phishing URL detection using online learning. Proc. of the 3rd ACM Workshop on Artificial Intelligence and Security. ACM, 54–60.
[17] Pranam Kolari, Tim Finin, and Anupam Joshi. 2006. SVMs for the Blogosphere: Blog Identification and Splog Detection. In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 92–99.
[18] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating deep learning approaches to characterize and classify malicious URL's. Journal of Intelligent & Fuzzy Systems, 34(3), 1333-1343.
[19] Saxe, J., & Berlin, K. (2017). eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. arXiv preprint arXiv:1702.08568
[20] UFLDL Tutorial, "Convolutional Neural Network", http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
[21] Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., & Gonzalez, ´ F. A. (2017, April). Classifying phishing URLs using recurrent neural networks. In Electronic Crime Research (eCrime), 2017 APWG Symposium on (pp. 1-8). IEEE.
[22] Justin Tung Ma, Lawrence K.Saul, Stefan savage, Geoffrey M.Voelker, Pamela Cosman, Gert Lanckriet, University of California, San Diego, "Learning to Detect Malicious URLs", A dissertation submitted in partial satisfaction of the requirements for the degree, https://escholarship.org/uc/item/4j86t705
[23] Kaggle malicious dataset, Retrieved from https://www.kaggle.com/search?q=malicious+in%3Adataset.