Structural Analysis of URL For Malicious URL Detection Using Machine Learning

A. Saleem Raja [a], S. Peerbasha[b*], Y. Mohammed Iqbal[b], B. Sundarvadivazhagan[a], M. Mohamed Surputheen[b]
[a]Information Technology Department, University of Technology and Applied Sciences, Sultanate of Oman.
[b]Department of Computer Science, Jamal Mohamed College, Affiliated to Bharathidasan University, Trichy, Tamilnadu, India.

Abstract

Malicious websites are intentionally created websites that aid online criminals in carrying out illicit actions. They commit crimes like installing malware on the victim's computer, stealing private data from the victim's system, and exposing the victim online. Malicious codes can also be found on legitimate websites. Therefore, locating such a website in cyberspace is a difficult operation that demands the utilization of an automated detection tool. Currently, machine learning/deep learning technologies are employed to detect such malicious websites. However, the problem persists since the attack vector is constantly changing. Most research solutions use a limited number of URL lexical features, DNS information, global ranking information, and webpage content features. Combining several derived features involves computation time and security risk. Additionally, the dataset's minimal features don't maximize its potential. This paper exclusively uses URLs to address this problem and blends linguistic and vectorized URL features. Complete potential of the URL is utilized through vectorization. Six machine learning algorithms are examined. The results indicate that the proposed approach performs better for the count vectorizer with random forest algorithm.

Keywords: Malicious Link, Phishing, Natural Language Processing, Machine learning, ngram, Random Forest, Lexical features of URL, TFIDF vectorizer, Count vectorizer , Hashing vectorizer .

*Corresponding author Email: bashapeer2003@gmail.com

I. Introduction

Each year, the number of devices linked to the internet increases significantly. Smart apps are becoming more prevalent on the web and on mobile devices. As a result, cyberattacks surged as well. Cybercrime costs are rising 15% per year globally, according to a cyber-venture research [1]. Cybersecurity is seriously threatened by malicious websites or URLs. These kinds of URLs include malicious content that can result in a phishing scam or a drive-by download attack. Malicious website’s URL use appealing terms to resemble legitimate URLs. Malicious URLs are typically injected into legitimate websites or disseminated by SMS or social media. The intentional or accidental use of such URLs may compromise the security of a user or business and have legal consequences. These links expose the user's system vulnerabilities in cyberspace by downloading malware or stealing sensitive information from the user's system once the user clicks on them. Approximately 22 billion records were compromised as a result of the more than 4,100 publicly reported data breaches in 2021[2]. Recognizing such malicious URLs in cyberspace is a difficult challenge for the user. Majority of security systems use the stored list of URLs to block harmful link access. But recent attacks use short URLs and URLs that are generated by an algorithm can easily get around the security system. Therefore, it is becoming increasingly important to have an automated system in place that makes use of machine learning techniques to detect and prevent such attacks. The performance of the machine learning-based detection method is dependent on the feature that is utilized throughout the detection process. Features are usually taken from different datasets and are used in the classification process. Researchers investigated many types of features generated from the URLs, the content of the webpages, and some information taken from the domain server, as well as the rank of the webpage from external servers. Certain visual similarities are also taken into account when detecting malicious webpages [3]. Aside from the linguistic feature of the URL, other features necessitate either access to external resources or the processing of potentially hazardous content on the webpage. This increases computation time and creates security risks. Therefore, researchers prefer to solely process URLs for malicious URL detection. However, identifying a sufficient number of features from URLs is a difficult aspect of processing URLs. Sometimes the retrieved features might not make use of all the potential possibilities of the URL This paper parse the URLs to generate tokens, clean the tokens, vectorize the tokens using natural language processing methods, which turns the text into vectors. These vectors are then coupled with selected lexical features of the URL, which serves as the dataset for machine learning algorithms. Six different machine learning algorithms are tested for the better detection accuracy. Three different vectorizer are used to vectorize the data such as count vectorizer, TFIDF and hashing vectorizer. The result demonstrates that count vectorizer with random forest algorithm achieves the higher accuracy.

The paper is organized as follows. Section 1 provides the significance of the problem and overview of the solution strategy. Part II discusses existing research. Part III describes the machine learning classifiers and NLP methods. The proposed method is in Section IV, and the experimental findings are in Section V. The paper is concluded in Section VI.

II. Related Work

Many current security studies aim to prevent attacks by identifying potentially harmful websites. Most research examines the linguistic properties of URLs since they are risk-free, and very few features are extracted from the dataset that increase the processing speed. This section highlights some of the most recent works. Boukhalfa et al.[4] provided a method for locating fraudulent URLs on social media sites like Twitter. Around 25 features were gathered from the URL, webpage, and DNS server for the experiment using the UCI phishing, Phishtank, and MillerSmiles datasets. The experiments' use of LR, SVM, and RF produced accuracy results of 90.28%, 93.43%, and 95.51%, respectively. By combining NLP and machine learning techniques, Lakshmanarao et al. [5] proposed a method of malicious website detection. More than 500000 URLs from the Kaggle URL dataset were utilized for the studies. The trial employed four distinct machine learning algorithms (LR, KNN, DT, RF), with RF achieving the highest accuracy of 97.5%. Joshi et al[6] claim that machine learning and deep learning aid in the detection of malicious URLs sent over email. The proposed method used the linguistic features of URLs to classify them. Openphish, Alexa, and Fire Eye datasets were used in the experiment. 23 lexical characteristics are extracted from the URL, and the Random Forest algorithm achieves 92% accuracy. Lexical analysis and feature quantification are used by Hong et al. [7] to identify harmful domain names. For successful and precise detection, the approach consists of two stages. The first stage matches a domain name to a blacklist of malicious domain links. It is either malicious or possibly malicious depending on how closely the domain name alterations match the blacklist. In the second stage, an N-gram model's reputation value is used to evaluate a suspected malicious domain name. The top 100,000 regular Alexa domain names are used to extract a collection of whitelist/blacklist substrings using the N-gram technique. Substring weights are based on their whitelist/blacklist frequency. Finally, the potential harmful domain name's reputation value determines its legitimacy. Selvi et al. [8] utilized Random Forest to recognize algorithmically generated domains. The dataset included ordinary and algorithmically generated domain names from numerous malware families for testing. Masked N-grams and other domain name data were extracted. Result shows that masked N-grams provide improved performance and detection accuracy compared to state-of-the-art methods. Saleem et al. [9] proposed a machine-learning-based method for identifying malicious URLs based on the linguistic features of the URL. The UNB dataset 2016 was used in the experiment. 20 of the 27 collected features are used in the experiment. Findings show that the random forest classifier performs better than the other classifiers.

III. Background

URL is a string and a unique identifier to locate the resource in the cyberspace. It is composed of several elements that are presented in the figure 1. Generally, machine learning models operated on numerical features that are extracted from the raw dataset (list of URL) and represented as 2-D array. The majority of recent studies just extract a few features from the dataset. In table 2, various types of features are presented. In addition to conventional features, this paper vectorizes URL texts using natural language processing (NLP) techniques.

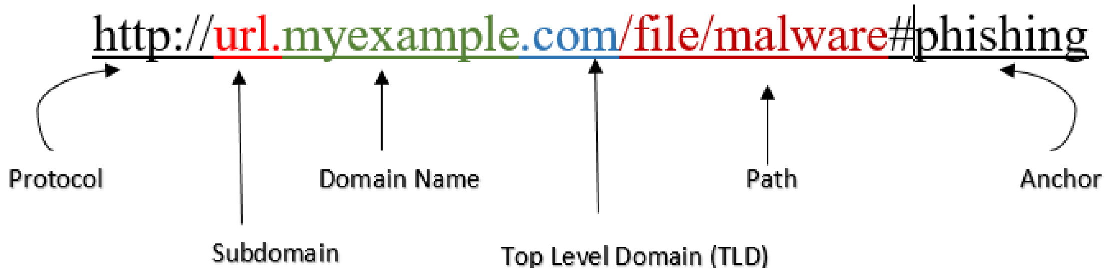


Fig 1: URL

NLP has a set of vectorization methods. such as Count Vectorizer, Hashing Vectorizer, and TFIDF. Vectorization converts text into numerical features. Then the vectorized data will be used in machine learning algorithms. Table 1 listed the machine learning algorithms for our experiment.

- *Count Vectorizer [10]:* It is easy to convert text data into a numerical vector that counts word occurrences in a document. This method produces sparse matrix.
- *TFIDF [10]:* It is used to reflect the relevance of a word in a corpus. It indicates word significance in a corpus. The equation 1 use to calculates TF, which is a document's word frequency.

$$TF = \frac{Number\ of\ occurence\ of\ the\ word\ in\ a\ document}{Total\ number\ of\ the\ words\ in\ the\ document} (1)$$

The equation 2 calculates IDF, which is the word's frequency across documents. IDF denotes word significance. In general, rarer words are more informative.

$$\mathrm{IDF} = log_{10} \frac{Total\ number\ of\ documents}{Number\ of\ document\ that\ contain\ the\ word} (2)$$

At last, the score of the TFIDF is computed with the equation 3.

$$TFIDF = TF * IDF (3)$$

- *Hashing Vectorizer [10]: S*imilar to count vectorizer, that converts raw text into numerical vector (matrix) but hold less memory because of not storing the resulting vocabulary.

Table 1: Machine learning algorithms [11]

Algorithm	Explanation
Logistic Regression (LR)	Used to predict the binary target class based on the independent features. It uses sigmoid function to transform the real value into discrete class.
Multinomial Naive Bayes (MNB)	Binary and multi-class classification problems are solved with Naive Bayes. In order to address problems with document or text categorization, the Multinomial Naive Bayes (MNB) classifier uses a multinomial distribution.
Decision Tree (DT)	Construct the tree, based on the input data and are divided based on the conditions in the internal node. It can be used for regression and classification problems
K Nearest Neighbors (KNN)	The closest data points to the selected point are grouped together. Group size is determined by K.
Random Forest (RF)	Algorithm construct groups of decision trees. The algorithm's output is the mean prediction of all decision trees.
Gradient Boosting (GB)	Combines several weak models to get better performance model.

IV. Proposed Method

The proposed method (CONVEC) utilizes the potential of the raw dataset by exploiting through **con**ventional feature extraction and **vec**torization through NLP methods. Process flow of the CONVEC is depicted in the figure 2.

A. *Raw Dataset*

The dataset for the experiment was collected from the Kaggle URL dataset [12], which includes both benign and malicious urls. There are 450176 urls in the dataset. Imbalanced dataset affects the classification process which gives a skewed result [13]. To avoid such issue, the experiment uses 10000 benign and 10000 malicious urls.

B. *Conventional Feature Extraction*

Most of the research done so far uses the four categories of features shown in table 2 to classify malicious URLs [9]. For our experiments, 8 common features are extracted from the raw dataset, that are listed in table 3 and are stored in a separate file (fs.csv) as shown in figure 2.

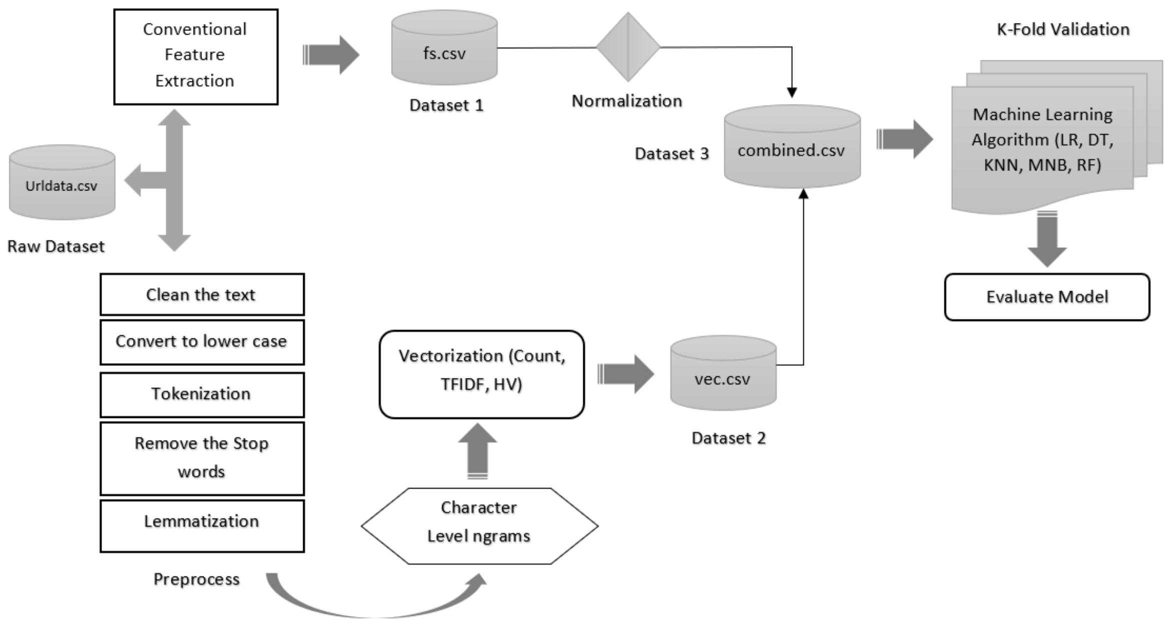


Fig 2. Proposed method(CONVEC) process flow

Table 2: Categories of Features [9]

Category	Description
Features extracted from URL string	Count the various elements in the URL such as counting number of dot, hyphens etc.
Features extracted from webpage content	Count the various elements in the webpage such as counting number of script tags, paragraph tags etc. Counting specific words in the webpage
Features extracted from Domain name server	IP address, registration date, expiry date, hosted server and other related information of the webpage
Features extracted from third party server	Ranking of webpage from google or alexa etc.

Table 3: URL Features

No	Features	Description
1	urllen	URL Length
2	domlen	Domain name length
3	countDoT	Count the number of dots in domain name
4	counthyphen	Count the number of hyphens in domain name
5	tldlen	TLD length
6	tlddotcount	Number of dots in TLD
7	pathlen	Path Length
8	pathslashcount	Number of slashes in the path

C. *Preprocess*

Preprocessing encompasses all operations required for text vectorization. Cleaning text removes special characters, unwanted parts of the url such as protocol and considers only text and numbers. After cleaning the text, will convert to lowercase, remove stop words, and lemmatize to reduce features during vectorizing. The Algorithm 1 summarizes the preprocessing steps.

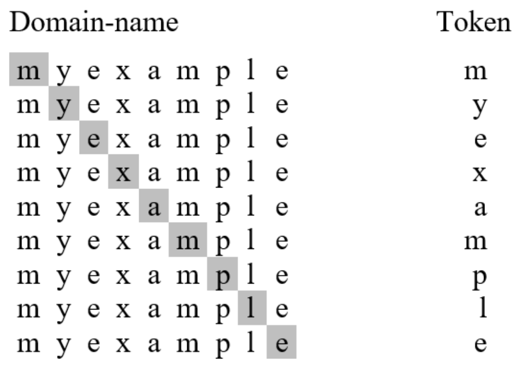
Algorithm 1: Data Preprocessing

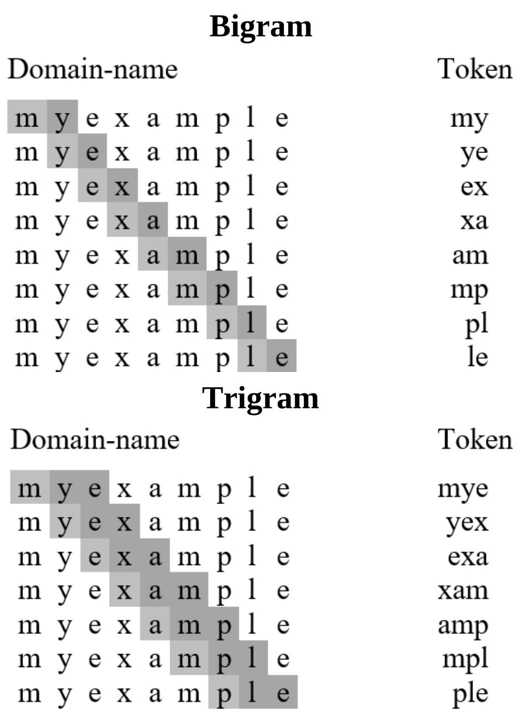
```
Input: Web content both phishing and benign (urldata.csv)
Output: Corpus ($T_2$ )
Function preProcess(urldata.csv)
$T_{org}$=Read_text(data.csv)
$T_1$=Cleaning($T_{org}$)    Remove the unwanted elements from $T_{org}$
$T_1$ = Lower(T1)
For token in $T_1$ do
    if token is not a STOPWORDS in ENGLISH then
        $T_2 = T_2$ U Lemmatize(token)
    End if
end for
return $T_2$
```

D. *Character Level ngrams*

N-grams are ‘n’ items from a text sample. The two most prevalent varieties of N-Grams are character-level and word-level [14]. N determines token size. For our experiment, character level N-grams are used to generate number of tokens for vectorization. Process of the character level ngrams for the domain name are depicted in figure 3. We set the N value for the N-gram range from 3 to 7 to reduce the number of tokens in a URL.

Unigram





Bigram

Domain-name	Token
m y e x a m p l e	my
m y e x a m p l e	ye
m y e x a m p l e	ex
m y e x a m p l e	xa
m y e x a m p l e	am
m y e x a m p l e	mp
m y e x a m p l e	pl
m y e x a m p l e	le

Trigram

Domain-name	Token
m y e x a m p l e	mye
m y e x a m p l e	yex
m y e x a m p l e	exa
m y e x a m p l e	xam
m y e x a m p l e	amp
m y e x a m p l e	mpl
m y e x a m p l e	ple

Fig 3. Generating N-gram from the domain name

E. *Vectorization*

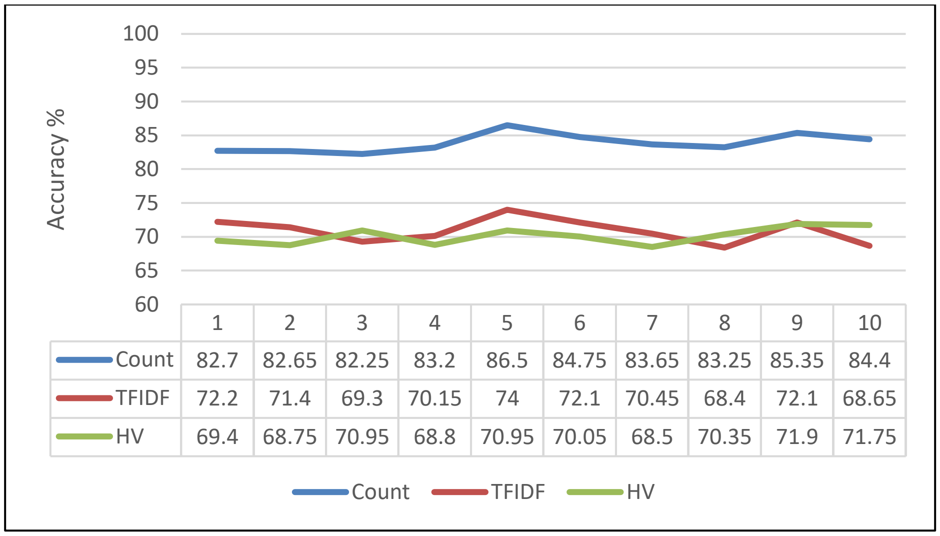
By employing various NLP techniques, such as count vectorizer, TFIDF, and hashing vectorizer, the generated tokens of text are converted into a real-valued vector. To limit the size of the 2-D array, we set the vectorizer's features to 2500 and store the output in a csv file (vec.csv) as shown in figure 1.

F. *K-Fold Validation & Model Evaluation*

Machine learning models are tested on a small data sample using cross-validation. Its primary purpose is to provide an estimate of the performance of a machine learning model on data that has not yet been examined [15]. In our experiment, combine dataset of Dataset 1 (fs.csv) and Dataset 2 (vec.csv) is fed in to machine learning algorithm with 10-fold cross validation. Performance of the machine algorithm is evaluated with four key metrics such as accuracy, precision, recall and f1-score.

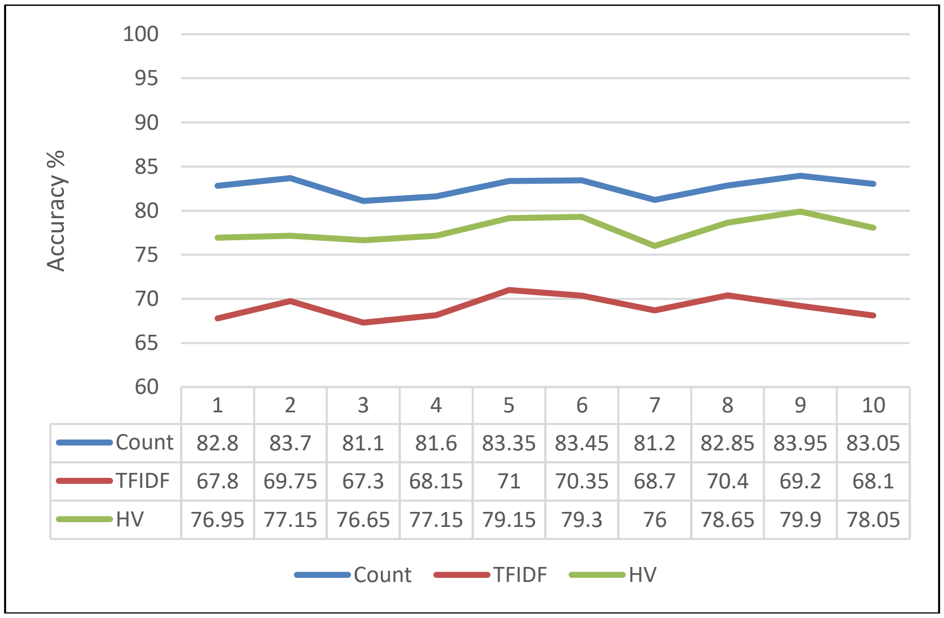
V. Experimental Result

The experimental configuration includes Windows 10, an I5 (3.2 GHz) processor, and 8 GB of RAM. Python with sklearn is used for programming. Three different vectorizers, such as a count vectorizer, an IFIDF vectorizer, and a hash vectorizer, are tested in the experiments. Experiments use six machine learning algorithms. Maximum number of features generated by the vectorizers are limited to 2500. The performance criteria that are taken into consideration are accuracy, precision, recall and f1-score. Cross-validation is the most appropriate method for evaluating the performance of machine learning models using a small data sample. It prevents overfitting and generalizes the model in order to obtain optimal performance. Figure 4 illustrates the performance (Accuracy) of various machine learning models with three distinct vectorizers under 10-fold cross validation.



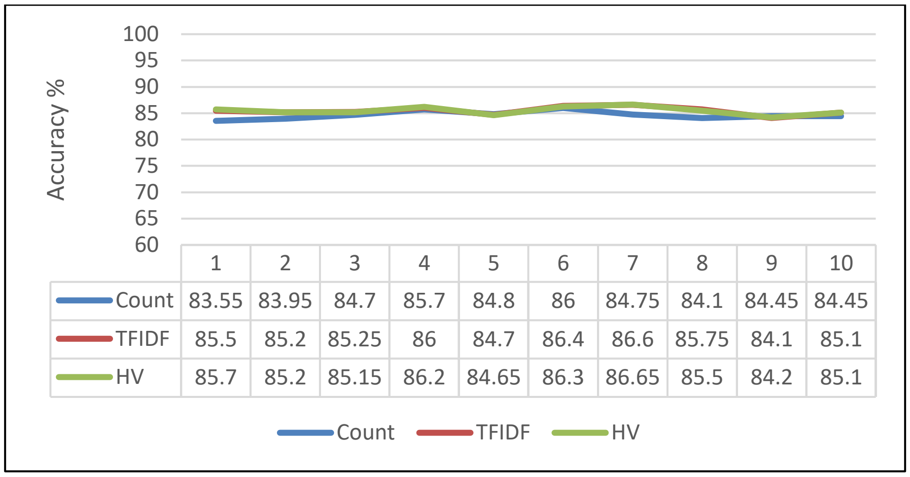
	1	2	3	4	5	6	7	8	9	10
Count	82.7	82.65	82.25	83.2	86.5	84.75	83.65	83.25	85.35	84.4
TFIDF	72.2	71.4	69.3	70.15	74	72.1	70.45	68.4	72.1	68.65
HV	69.4	68.75	70.95	68.8	70.95	70.05	68.5	70.35	71.9	71.75

LR



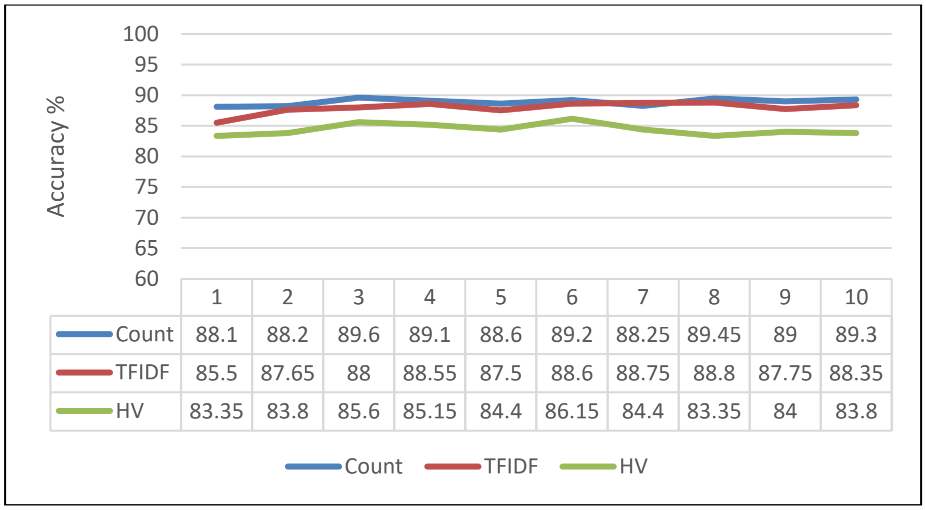
	1	2	3	4	5	6	7	8	9	10
Count	82.8	83.7	81.1	81.6	83.35	83.45	81.2	82.85	83.95	83.05
TFIDF	67.8	69.75	67.3	68.15	71	70.35	68.7	70.4	69.2	68.1
HV	76.95	77.15	76.65	77.15	79.15	79.3	76	78.65	79.9	78.05

MNB



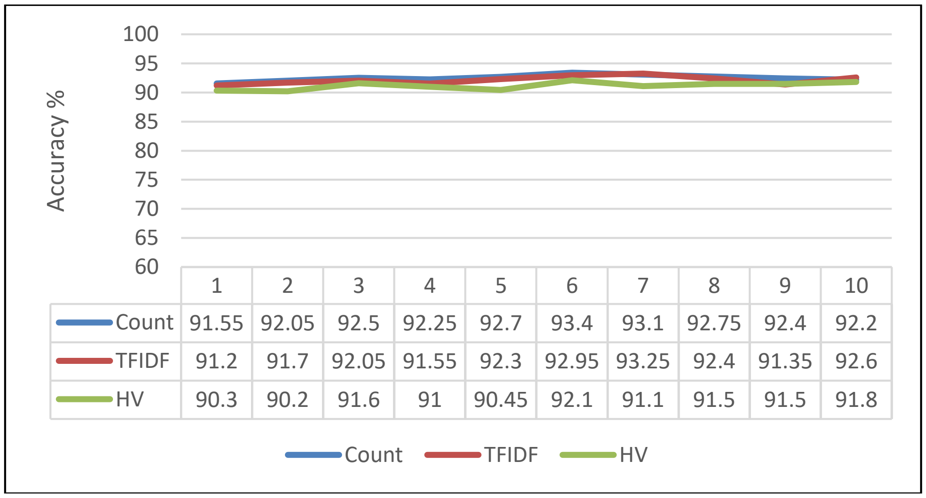
	1	2	3	4	5	6	7	8	9	10
Count	83.55	83.95	84.7	85.7	84.8	86	84.75	84.1	84.45	84.45
TFIDF	85.5	85.2	85.25	86	84.7	86.4	86.6	85.75	84.1	85.1
HV	85.7	85.2	85.15	86.2	84.65	86.3	86.65	85.5	84.2	85.1

KNN



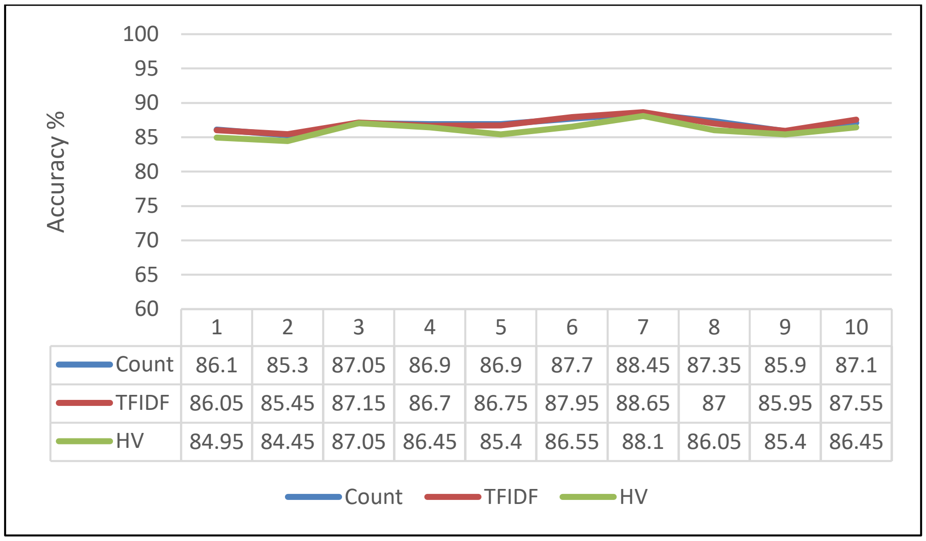
	1	2	3	4	5	6	7	8	9	10
Count	88.1	88.2	89.6	89.1	88.6	89.2	88.25	89.45	89	89.3
TFIDF	85.5	87.65	88	88.55	87.5	88.6	88.75	88.8	87.75	88.35
HV	83.35	83.8	85.6	85.15	84.4	86.15	84.4	83.35	84	83.8

DT



	1	2	3	4	5	6	7	8	9	10
Count	91.55	92.05	92.5	92.25	92.7	93.4	93.1	92.75	92.4	92.2
TFIDF	91.2	91.7	92.05	91.55	92.3	92.95	93.25	92.4	91.35	92.6
HV	90.3	90.2	91.6	91	90.45	92.1	91.1	91.5	91.5	91.8

RF



	1	2	3	4	5	6	7	8	9	10
Count	86.1	85.3	87.05	86.9	86.9	87.7	88.45	87.35	85.9	87.1
TFIDF	86.05	85.45	87.15	86.7	86.75	87.95	88.65	87	85.95	87.55
HV	84.95	84.45	87.05	86.45	85.4	86.55	88.1	86.05	85.4	86.45

GB

Fig 4. Performance (Accuracy) of Machine Learning Algorithms under 10-Fold Cross Validation

The results of the various trials are shown in the figure 5-7 and table 5-7. The outcomes indicated that Count vectorizer with RF obtains the higher accuracy of 92.49%.

Table 5. Performance of Count Vectorizer

Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
LR	83.87	84.57	82.92	83.71
MNB	82.71	84.59	79.99	82.22
KNN	84.65	89.31	78.71	83.67
DT	88.88	89.04	88.69	88.86
RF	92.49	92.58	92.39	92.48
GB	86.88	91.34	81.48	86.12

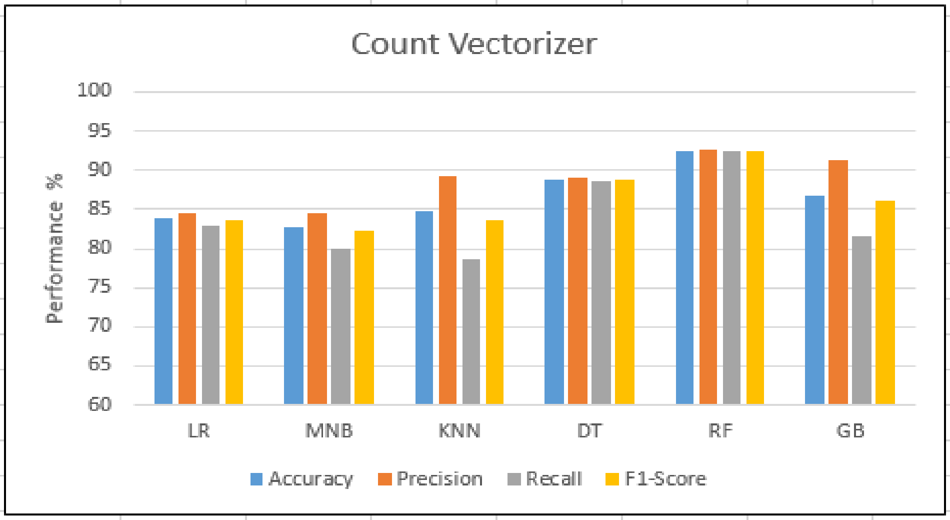


Fig 5. Performance of Count Vectorizer

Table 6. Performance of TFIDF Vectorizer

Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
LR	70.88	73.55	65.22	69.12
MNB	69.08	69.95	66.91	68.39
KNN	85.46	89.63	80.20	84.65
DT	87.95	88.01	87.87	87.94
RF	92.14	92.29	91.96	92.12
GB	86.92	91.46	81.45	86.16

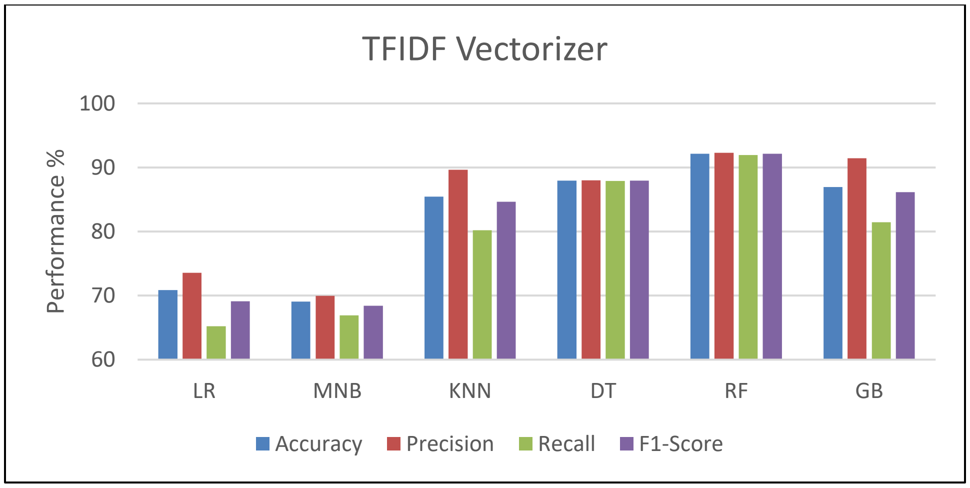


Fig 6. Performance of TFIDF Vectorizer

Table 7. Performance of Hashing Vectorizer

Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
LR	70.14	72.46	65.05	68.53
MNB	77.90	79.61	75.01	77.24
KNN	85.47	89.49	80.37	84.68
DT	84.40	84.29	84.56	84.42
RF	91.16	92.69	89.36	90.99
GB	86.09	90.18	80.99	85.33

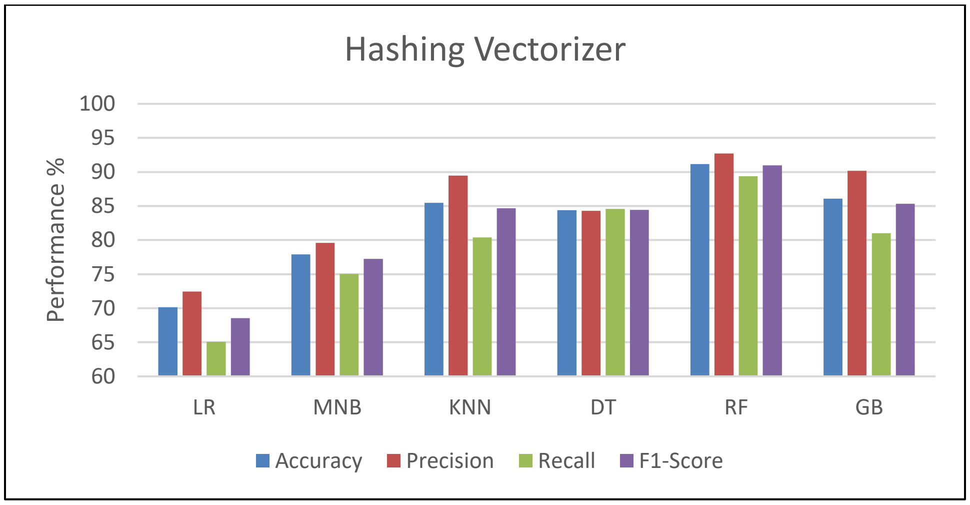


Fig 7. Performance of Hashing Vectorizer

VI. Conclusion

Websites provide the ideal environment for cybercriminals to infect naïve visitors. They deceived the trusting users into becoming their target using a range of social engineering strategies. To identify such dangerous websites on the internet, AI-based automated solutions are required. This study makes use of linguistic features in the URL to identify malicious links. Six different machine learning algorithms are used to test three different NLP-based vectorizers. Results show that the proposed method with count vectorizer + RF algorithm provides a higher level of accuracy.

References:

[1]. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/
[2]. https://www.cshub.com/attacks/news/the-top-10-hacks-and-cyber-security-threats-of-2022.

[3]. Saleem, Pradeepa, Arulkumar., Mudhr: Malicious URL detection using heuristic rules based approach, AIP Conference Proceedings, vol 2393, Issue 1, 2022.

[4]. Boukhalfa, Djaballah, Ghalem, Boukerma, A new approach for the detection and analysis of phishing in social networks: the case of Twitter,Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS), pp. 1-8, 2020. https://doi.org/10.1109/SNAMS52053.2020.9336572

[5]. Lakshmanarao, Babu, Bala, Malicious URL Detection using NLP, Machine Learning and FLASK, International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp. 1-4, 2021. https://doi.org/10.1109/ICSES52305.2021.9633889

[6]. Joshi, Lloyd, Westin, Seethapathy., Using Lexical Features for Malicious URL Detection - A Machine Learning Approach, arXiv:1910.06277.

[7]. Hong, Zhaobin, Weijie, Xiangyan, Malicious Domain Names Detection Algorithm Based on Lexical Analysis and Feature Quantification, IEEE Access, vol. 7, pp. 128990-128999, 2019. https://doi.org/10.1109/ACCESS.2019.2940554

[8]. Selvi, Rodríguez, Soria-Olivas, Detection of algorithmically generated malicious domain names using masked N-grams, Expert Systems with Applications,vol.124,pp.156-163,2019. https://doi.org/10.1016/j.eswa.2019.01.050

[9]. Saleem, Vinodini, Kavitha, Lexical features based malicious URL detection using machine learning techniques, Materials Today: Proceedings,vol 47, 1,pp.163-166, 2021, https://doi.org/10.1016/j.matpr.2021.04.041.

[10]. Benjamin, Rebecca, Tony., Applied Text Analysis with Python Enabling Language-Aware Data Products with Machine Learning, O'Reilly Media, Inc, 2018.

[11]. Aurélien Géron , Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, 2017.

[12]. Dataset: https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls.

[13]. Jadhav, Mostafa, Elmannai, Khalid, An Empirical Assessment of Performance of Data Balancing Techniques in Classification Task, Applied Sciences, 12, 3928,2022, https://doi.org/10.3390/app12083928.

[14]. Li, Aletras, Improving Graph-Based Text Representations with Character and Word Level N-grams, arXiv:2210.05999. https://doi.org/10.48550/arXiv.2210.05999.

[15]. Saleem, Sundaravadivazhagan, Pradeepa, Justin, Karthikeyan., Weighted ensemble classifier for malicious link detection using natural language processing, International Journal of Pervasive Computing and Communications,2023. https://doi.org/10.1108/IJPCC-09-2022-0312