Automated Detection of Malevolent Domains in Cyberspace Using Natural Language Processing and Machine Learning

Saleem Raja Abdul Samad[1], Pradeepa Ganesan[2], Amna Salim Al-Kaabi[3], Justin Rajasekaran[4], Singaravelan M[5], Peerbasha Shebbeer Basha[6]

IT Department, College of Computing and Information Sciences, Shinas, University of Technology and Applied Sciences, Sultanate of Oman[1, 2, 3, 4]

Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R & D Institute of Science and Technology, Chennai, Tamil Nadu, India[5]

Department of Computer Science, Jamal Mohammed College, Tiruchirappalli, Tamil Nadu, India[6]

***Abstract*—Cyberattacks are intentional attacks on computer systems, networks, and devices. Malware, phishing, drive-by downloads, and injection are popular cyberattacks that can harm individuals, businesses, and organizations. Most of these attacks trick internet users by using malicious links or webpages. Malicious webpages can be used to distribute malware, steal personal information, conduct phishing attacks, or perform other malicious activities. Detecting such malicious websites is a tedious task for internet users. Therefore, locating such a website in cyberspace requires an automated detection tool. Currently, machine learning techniques are being used to detect such malicious websites. The majority of recent studies derive a limited number of features from webpages (both benign and malicious) and use machine learning (ML) algorithms to detect fraudulent webpages. However, these constrained capabilities might not use the full potential of the dataset. This study addresses this issue by identifying malicious websites using both the URL and webpage content features. To maximize detection accuracy, both ngrams and vectorization methods in natural language processing are adopted with minimum feature-set. To exploit the full potential of the dataset, the proposed approach derives the 22 common linguistic features of the URL and generates ngrams from the domain name of the URL. The textual content of the webpages was also used. The research employs seven machine learning algorithms with three vectorization methods. The outcome reveals that the proposed method outperformed the results of previous studies.**

***Keywords*—Machine learning; N-gram; linguistic features; natural language processing (NLP); malicious webpage**

I. Introduction

A malicious webpage is a website intentionally crafted to harm users or their devices, often orchestrated by hackers and cybercriminals utilizing various techniques to exploit vulnerabilities. These webpages pose significant threats, serving as conduits for malware distribution, personal information theft, phishing attacks, and other malicious activities. Cybercriminals employ common techniques like phishing, where deceptive URLs (Uniform Resource Locators) or webpages mimic legitimate sites to trick users into divulging login credentials or personal data, facilitating identity theft or financial fraud. Another tactic involves drive-by downloads, concealing malicious code within a webpage to exploit vulnerabilities in users' browsers or operating systems, and installing malware without their knowledge. Malvertising utilizes legitimate online advertising networks to redirect users to malicious pages, aiming to infect their computers with malware. Cryptojacking involves hijacking a user's computer processing power to mine cryptocurrency for attackers. Once users access a malicious webpage, they risk downloading malware, suffering personal information theft, or falling victim to various attacks. The consequences extend to severe impacts on individuals, businesses, and organizations, encompassing data theft, financial losses, reputational damage, and even physical harm.

The cost of cyber-attacks continues to rise and is a significant concern for organizations across all industries. According to IBM's 2024 Cost of a Data Breach Report, the global average cost of a breach has increased by 10% since 2023, reaching $4.88 million [1]. Global cybercrime expenditures are anticipated to increase from $3 trillion in 2015 to $10.5 trillion annually by 2025, according to Cybersecurity Ventures research [2]. Phishing attacks, which targeted 1339 brands, accounted for 36% of US data breaches in 2023. It became the second-most expensive source of compromised credentials with 5 million unique phishing sites, signaling a significant change in the dynamics of breaches across industries [3]. The financial impact of a cyber-attack depends on factors such as its severity, organizational size, and the compromised data. To mitigate these impacts, proactive measures like implementing robust security systems and conducting employee training programs are essential. While conventional security systems often rely on stored lists of URLs to block harmful links, recent attacks exploit vulnerabilities using short URLs and algorithmically generated ones, evading traditional security measures. Consequently, the necessity for an automated system employing machine learning techniques becomes increasingly critical to detect and prevent these evolving cyber threats effectively. Timely detection plays a pivotal role in ensuring a proactive defense against cyber

Funding: This research work is supported by the Research Internal Funding Program (RIFP) 2024, of the University of Technology and Applied Sciences-Shinas, Sultanate of Oman.

threats, effectively safeguarding users and their devices from various malicious activities.

Machine learning-based detection relies on feature engineering. The researcher identifies and uses six types of features for the detection of malicious webpages such as:

- Linguistic analysis of URL [3][4]: URL consists of several parts such as protocol, domain name, path, query-string, etc. Using these parts for the detection process such as a number of subdomains in the URLs, protocol used in the URL length of the path, etc.
- Information obtained Domain Name System (DNS) [3][5][6]: DNS is a critical component of the internet infrastructure, and it plays an important role in the functioning of URLs. To generate features, researchers obtain information from the DNS server such as server location, registration and expiry date of the website, etc.
- Structural information of webpage [7]: The structural information of a webpage refers to the underlying organization of the webpage's content, including its layout, formatting, and elements of the page. For malicious website detection, the researcher analyzes HTML tags, DOM objects, CSS, and JavaScript elements, and other web page components.
- Linguistic analysis of webpage contents [8]: This is the written content of the webpage, including headings, paragraphs, and other text elements. The linguistic analysis involves character and word-level analysis to find harmful websites, such as finding probable keywords, tokens, and keyword weight, etc.
- Website Reputation [9]: Website reputation or ranking refers to the perceived level of trustworthiness, authority, and relevance of a particular website, as assessed by search engines, users, and other entities. Websites with a high reputation or ranking are generally viewed as more credible and valuable and may receive more traffic and engagement as a result. Ranking of the website globally and locally is considered as a feature for malicious webpage detection.
- Visual Similarity of the Webpage [10]: Visual similarity of a webpage can be a useful tool in malicious webpage detection. Malicious actors may attempt to create webpages that are visually similar to legitimate sites to trick users into providing sensitive information or installing malware. One approach to detecting visual similarity in webpages is to use computer vision techniques such as image processing, feature extraction, and machine learning. These techniques can be used to identify and compare visual elements of webpages, such as logos, fonts, and layouts.

Among all these features, researchers preferably use the linguistic analysis of URLs because it doesn't involve any risks. Web page content, on the other hand, gets less attention due to its risky nature. However, in comparison, webpage contents are a rich source of malicious webpage detection. The majority of recent studies derive a limited number of features from the URL and use machine learning algorithms (ML) to detect fraudulent webpages. However, these constrained capabilities might not use the URL's full potential. This study addresses this issue by identifying malicious websites using both the URL and webpage content features. To maximize detection accuracy with a minimum hybrid feature set, the proposed approach uses both URL and webpage contents. To exploit the full potential of the dataset, the proposed approach derives the 22 common linguistic features of the URL and generates ngrams from the domain name of the URL. Moreover, it extracts text from the webpage's paragraph tags. In the experiment, three distinct vectorization techniques (Count vectorizer, TFIDF, and Hashing vectorizer) are used to convert text into real numbered 2-D vectors. Lastly, the vectorized features and linguistic features are integrated to generate the experimental dataset. The incorporation of vectorized contents with URL linguistic features (hybrid approach) eliminates the necessity for generating and selecting a limited number of features. This ensures the full exploitation of the dataset's potential. The research employs seven machine learning algorithms. The outcome reveals that the proposed method outperformed the results of previous studies.

Contribution of the paper:

- Both URL and Webpage contents are utilized to generate features.
- Two different feature sets are generated using URLs of the webpages namely linguistic features and character-level ngrams of the URL
- Textual contents are extracted from the paragraph tags of the webpages.
- Three different NLP methods (Count vectorizer, TFIDF, and Hashing vectorizer) are used to vectorize the textual contents such as ngrams of the URL and textual contents from the webpages.
- NLP methods are used for feature generation instead of manual feature extraction.
- A balanced dataset (malicious and benign) is used for the experiments.
- Seven machine learning algorithms are evaluated for better detection accuracy.
- Achieve higher accuracy and f1-score with minimum feature set.

The structure of this paper is organized as follows. Section I outlines the significance of the problem and an overview of the solution strategy. Existing research is presented in Section II. The NLP techniques and machine learning classifiers are presented in Section III. Section IV presents the proposed method, and Section V reports the experimental results. Section VI provides a conclusion of the paper.

II. Related Works

Malicious webpage detection is a critical component of web security, as it can help to protect users from a variety of online threats, such as malware, phishing, and other types of

malicious activity. Extensive research has been conducted in this area. Table I summarizes recent research studies in this field. Saleem et al. [7] introduced a technique for classifying malicious webpages by analyzing their content through machine learning and deep learning approaches. The presented method exclusively uses of tags, events, keywords, scripts in the website for classification. The study utilizes the Kaggle dataset. From the dataset, over 206 features are retrieved. The selectKbest approach selects the highest-scoring features for the experiment. Scores in selectKbest are based on chi-square tests. Support vector machine (SVM), random forest (RF), and convolutional neural network (CNN) are used to test the system. The results demonstrate that random forest and support vector machine obtains the accuracy of 93% and 88% respectively. Saleem et al. [11] suggested a simple technique for detecting malicious URLs. The authors note that traditional approaches to malicious URL detection, which rely on blacklists and reputation-based systems, are not effective against new or previously unseen threats. To address this, the authors propose a new approach that uses a set of lexical features, including the length of the URL, the presence of certain characters, and the number of subdomains, to train a machine learning model to detect malicious URLs. The authors evaluate their method using a UNB dataset containing over 68851 URLs, including both malicious (33473) and benign (35378) URLs. From the dataset, 27 lexical features were taken, and 20 of those features were used in the experiment. The study's findings indicate that the suggested method achieves high accuracy in detecting malicious URLs, with 99% accuracy for random forest (RF) and 98% accuracy for k-nearest neighbor (k-NN) algorithms, respectively. Malak et al. [12] evaluate different feature sets for detecting malicious URLs using machine learning and deep learning models. The authors note that existing approaches to malicious URL detection often rely on a single feature set, which may limit their effectiveness. To address this, the authors evaluate three different feature sets: lexical, network-based, and content-based. Many feature selection procedures, including correlation analysis, ANOVA, and chi-square, were utilized to select features from the dataset. Naive Bayes (NB) was shown to be the most appropriate method for identifying malicious URLs in the used dataset, with an accuracy of 96%. Kamel [13] proposes a new approach for detecting and analyzing phishing attacks on social networks, specifically on Twitter. The authors note that phishing attacks on social networks are a growing problem, and traditional approaches to detecting them, such as manual analysis or keyword-based filtering, are not effective. To address this, the authors proposed a new approach by using machine learning algorithms to identify phishing attacks on Twitter. The approach involves analyzing the content of tweets and using machine learning to classify them as either phishing or non-phishing. For this experiment, a UCI phishing, Phishtank, and MillerSmiles datasets were utilized, and roughly 25 features were retrieved from the URL, webpage, and DNS server. For the studies, LR, SVM and RF were applied, yielding 90.28%, 93.43%, and 95.51% accuracy, respectively. Lakshmanarao et. al [14] proposes a web application for detecting malicious URLs using natural language processing (NLP) and machine learning techniques. This method makes use of a count, a TFIDF, and a hashing vectorizer. The outcome demonstrates that hashing vectorizer and random forest obtained 97.5% accuracy. Machine and deep learning help detect email-delivered malicious URLs, according to Joshi et al [15]. To classify URLs, the proposed method relied on their lexical features. The experiment employed a dataset from openphish, alexa, and fire eye. The proposed method used 23 distinct lexical characteristics to distinguish between malicious and benign URLs. Combining the lexical features with 1000 trigram-based features yielded 1023-long numerical vectors to represent the URLs. To identify the most important traits, correlation and scatter matrices were used. The results show that 92% accuracy is achieved by the random forest method. Hong et al. [16] employ lexical analysis and feature quantification to identify potentially dangerous domain names. The method comprises of two stages for accurate and successful detection. A domain name is compared in the first step to a blacklist of harmful domain links. The degree to which the domain name modifications closely resemble the blacklist determines whether it is malicious or possibly malicious. In the second step, a suspected malicious domain name is assessed using the reputation value of an N-gram model. Using the N-gram approach, a collection of whitelist/blacklist substrings is extracted from the top 100,000 regular Alexa domain names. The frequency of substrings on whitelists and blacklists determines their weights. Lastly, the authenticity of the possibly dangerous domain name is determined by its reputation value. The outcome shows that the accuracy rate for the proposed method, LA-FQ, is 94.16%. Josh et al. [17] used random forest to detect algorithmically generated domains. For testing, the dataset comprised regular and algorithmically generated domain names from several malware families. Masked N-grams and other domain name data were extracted. Results show that masked N-grams provide improved performance and detection accuracy compared to state-of-the-art methods. To detect harmful web links using NLP, Saleem et al. [18] suggested an ensemble classifier. The author observed that feature generation needs effort and topic expertise. Sometimes the generated features don't maximize the data set. Hence, the suggested method generates a feature set from the URL using the NLP method and classifies using an ensemble classifier. Two datasets (D1 and D2) were used in the experiment. D1 and D2 had 91.4% and 99.1% accuracy, respectively. The phishing URL detection method used by Ozgur et al. [19] is based on machine learning and a natural language processing (NLP) feature set. URLs were broken up and features were taken from them. For the experiment, seven different machine learning algorithms were used. The author states that the proposed method is language-independent and uses a large dataset. The outcome demonstrates that random forest with NLP features has an accuracy of 97.98%.

TABLE I. SUMMARY OF THE EXISTING WORK

Author	Dataset/Data Source	Features	Accuracy	Issues
Saleem et al. [7]	Kaggle dataset	Web content-based features	SVM: 88%, RF: 93%	URL-based features and webpage textual contents were ignored.
Saleem et al. [11]	UNB Dataset	Lexical features of URL	RF:99%, KNN:98%	Limited number of features. Web content features were ignored
Malak et.al [12]	Singh & Kumar Dataset	Lexical, network-based, and content-based features	NB:96%	Limited features. The full potential of the dataset was not used.
Kamel et.al [13]	UCI phishing, Phishtank, MillerSmiles datasets	Lexical, content based and DNS server features	LR: 90.28%, SVM: 93.43%, RF: 95.51%	Limited features. The full potential of the dataset was not used.
Lakshmanarao et.al[14]	Kaggle dataset	NLP -Vectorizer Methods	Hashing vectorizer with RF: 97.5%	Potentials of web contents were not utilized.
Joshi et.al [15]	Openphish, Alexa, Fire eye	Lexical Features & N-Gram Method	RF:92%	Web content features were ignored.
Hong et al. [16]	Alexa, Anquan organization, Malwaredomains.com, Malicious domain list, Zeus Tracker, Conficker, Torping, Symmni, PhishTank	N-gram Method	LA-FQ:94.16%	Web content & Lexical URL features were ignored.
Josh et.al [17]	Alexa, Bader repo extended	N-Gram Method	RF: 98.91%	Web content & Lexical URL features were ignored.
Saleem et.al[18]	URL, UNB, Phistank dataset	NLP -Vectorizer Methods	Weighted Soft Voting Classifier: D1:91.4% , D2:99.1%	Web content & Lexical URL features were ignored.
Ozgur et.al [19]	Phistank dataset, Yandex Search API	NLP based features	RF: 97.98%	Web content and other features were ignored.

III. BACKGROUND

The URL of the webpage and the webpage's content are both valuable resources for webpage classification, particularly for the detection of malicious webpages. A URL is a string that is used as a specific identifier to find resources online. It is made up of several components as shown in Fig. 1. Due to its risk-free nature, researchers employ the URL of the webpage for detection. Two distinct methods are used to extract features from the URLs.

1) Numerical/presence measurement: check the URL for the presence of specific words or characters and count the occurrence of the specific word or character, URL/domain/path length, etc.

2) Decompose the URL string using the ngram method and identify interesting features such as the amount of 2-grams and the average length of 2-gram tokens, etc.

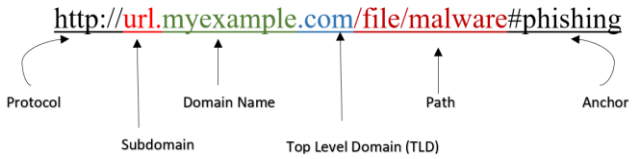

Fig. 1. Components of an URL.

Malicious websites can include a wide range of content that is intended to trick or exploit users, including fake login pages that steal login credentials, malware links that download malware onto the victim's computer, phishing forms that steal personal information, pop-up windows that direct users to phishing sites, redirect links that open phishing sites, malware scripts, fake surveys, and more. The textual content of a website serves as source for malicious activities. Generally, machine learning models operated on numerical features that are extracted from the raw dataset (list of url) and represented as 2-D array. However, in addition to a collection of URL features, this paper vectorizes URL text using natural language processing (NLP) methods. NLP has a set of vectorization methods. such as Count, Hashing, and TFIDF vectorizer. Vectorization converts text into numerical features. Then the vectorized data will be used in machine learning algorithms. Table II lists the machine learning algorithms used for the experiment.

a) Count vectorizer [20]: It works by first tokenizing the text corpus into individual words or n-grams (contiguous sequences of words). It then creates a vocabulary of unique tokens in the corpus and assigns an index to each token in the vocabulary. Finally, it counts the occurrence of each token in each document in the corpus and constructs a matrix of token counts, with one row per document and one column per token. The resulting matrix can be used as input to a machine learning algorithm for various NLP tasks.

b) TFIDF [20]: It measure the importance of each term in a document or a corpus. It considers both the frequency of a term in a document and the frequency of the term across the entire corpus. The basic idea behind TF-IDF is that a term is considered important if it appears frequently in a document but is not so common that it appears in every document in the corpus. The TF-IDF score of a term in a document is calculated by multiplying the term frequency (TF) of the term in the document by the inverse document frequency (IDF) of the term in the corpus.

The TF-IDF formula (1,2) can be expressed as follows:

$$TF(t,d) = (Number\ of\ times\ term\ t\ appears\ in\ document\ d) / (Total\ number\ of\ terms\ in\ document\ d) \quad (1)$$

$$IDF(t,D) = log\_e(Total\ number\ of\ documents\ in\ the\ corpus / Number\ of\ documents\ with\ term\ t\ in\ it)$$

$$TF-IDF(t,d,D) = TF(t,d) * IDF(t,D) \quad (2)$$

$where$:

- t: $a\ term$
- d: $a\ document$
- D: $the\ corpus$

c) Hashing vectorizer [20]: It is a technique used to convert a collection of text documents to a matrix of token occurrences. It is similar to Count-Vectorizer in that it creates a document-term matrix, but uses a hash function to convert each token to a numerical index, rather than storing the tokens as strings. This can be useful in situations where the vocabulary is very large, and the memory requirements of storing all the tokens are prohibitively high. The Hashing-Vectorizer works by mapping each token to a numerical index using a hash function. The resulting numerical index is then used as the column index in the document-term matrix. The size of the matrix is fixed in advance and depends on the number of features (i.e. the number of hashed tokens) that are required. This means that the hashing-vectorizer is a stateless transformer, and does not need to keep track of the mapping between tokens and numerical indices.

TABLE II. MACHINE LEARNING ALGORITHMS [21]

Algorithm	Explanation
Logistic Regression (LogR)	The logistic function, also referred to as the sigmoid function, is applied to a linear combination of the input variables via the logistic regression procedure. Any input value is converted by the logistic function to a probability value between 0 and 1. The method predicts the positive class (i.e., 1) if the probability is greater than a predetermined threshold (often 0.5), otherwise it predicts the negative class (i.e., 0).
Gaussian Naive Bayes (GNB)	Gaussian Naive Bayes is a variant of the Naive Bayes algorithm in machine learning that assumes that the features follow a Gaussian (normal) distribution. In Gaussian Naive Bayes, the likelihood of the features given the class is modeled as a normal distribution with mean μ and standard deviation σ for each feature and class.
Decision Tree (DT)	The Decision Tree algorithm iteratively splits data by feature or attribute value, creating a tree-like structure. At each node in the tree, the algorithm chooses the feature that delivers the best split based on some criterion, such as the Gini index or entropy. The procedure is repeated until a stopping requirement, such as a maximum depth or minimum number of samples per leaf, is fulfilled.
K Nearest Neighbors (KNN)	The number of nearest neighbors to take into account for each prediction is the initial step in the KNN algorithm's operation. The algorithm then determines the k training samples that are most similar to each new input based on a distance metric (e.g., Euclidean distance or Manhattan distance). The algorithm then calculates a forecast by averaging the target values of the k-nearest neighbors (for regression) or obtaining the majority vote (for classification).
Random Forest (RF)	Random Forest is a popular machine learning algorithm for both classification and regression tasks. It belongs to the family of ensemble learning methods, which combine multiple individual models to improve the overall predictive performance.
Gradient Boosting (GB)	Gradient Boosting is an ensemble learning method that combines weak models, usually decision trees, to generate a more accurate model. Each decision tree in the method is trained to rectify the faults of the previous tree. All tree predictions are added to make the final projection.
Extreme Gradient Boosting (XGB)	Extreme Gradient Boosting is a popular implementation of the Gradient Boosting algorithm that is optimized for speed and performance. It uses a technique called "gradient boosting with regularization," which adds a penalty term to the loss function to reduce overfitting. XGB is known for its high performance, scalability, and ability to handle large and complex data sets.

IV. PROPOSED METHOD

The proposed method utilizes both URLs and the textual contents of webpages from the raw dataset. Three distinct feature sets, ULF (URL linguistic features), CNF (Character level ngram features), and WCF (Web content features), are created.

- ULF-URL linguistic features, which include the conventional URL features.
- CNF-URL character level ngram features, which tokenize the URL's domain name using character level ngram.
- WCF-Web content features include webpage text.

Three separate NLP vectorization algorithms are employed to vectorize CNF and WCF. Subsequently, ULF, CNF, and WCF are amalgamated to form the final dataset for training and testing. The process flow of the proposed method is shown in Fig. 2.

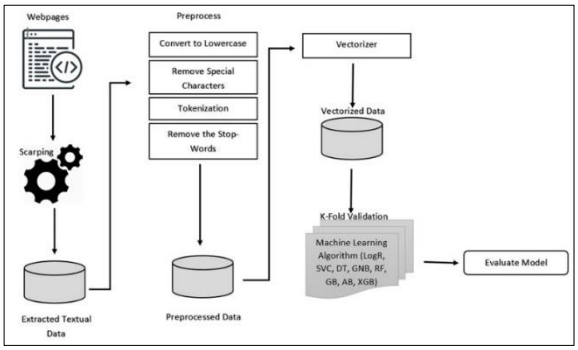

Fig. 2. Process flow.

TABLE III. URL FEATURES

No	Features	Description	No	Features	Description
1	ip_status	Presence of IP address in the URL	12	hyp_dom	Count the hyphens in domain name
2	dots_url	Count the dots in the URL.	13	at_dom	Count the @ in domain name.
3	slash_url	Count the / in the URL	14	underscr_dom	Count the underscores in the domain name
4	hyp_url	Count the hyphens in the URL	15	urllen	Length of the URL
5	hash_url	Count the # in the URL	16	num_url	Count the numbers in the URL
6	semi_url	Count the semicolons in the URL	17	alpha_url	Count the alphabet in the URL
7	and_url	Count the & in the URL	18	spl_url	Count the special symbols in the URL
8	underscr_url	Count the underscores in the URL	19	domlen	Length of domain name
9	http_url	Presence of http in the URL	20	num_dom	Count the numbers in the domain name
10	https_url	Presence of https in the URL	21	alpha_dom	Count the alphabet in the domain name
11	dots_dom	Count the dots in the domain name	22	spl_dom	Count the special symbols in the domain name
			23	url_class	Class of the URL either malicious (1)/benign (0)

A. Raw Dataset

The experiment's dataset, which contains both malicious and benign URLs, was gathered from the Kaggle URLs dataset [22]. The collection contains 450176 URLs. Classification is influenced by imbalanced datasets [23]. To prevent this problem, the experiment employs 6504 benign and 6478 malicious URLs.

B. Conventional Feature Extraction (URL Linguistic Features (ULF))

Conventional URL features include the dots in the URL, numbers in the URL, etc. Our experiments use only 23 lexical features (22 independent features and 1 dependent feature) of URLs, which are listed in Table III and saved as a separate file (conv fs.csv) as shown in Fig. 2.

C. Generating Character Level ngrams (Character level ngram features (CNF))

Character-level n-gram and word-level n-gram are two types of n-gram models used in natural language processing and machine learning. Both of these models are used for text analysis, but they operate at different levels of granularity [24]. To fully utilize the domain name of the URL, character level ngram is used in our experiment. A character-level n-gram model looks at sequences of characters in a text, regardless of the words they form. The model divides the text into n-grams, or consecutive groups of n characters called tokens, as shown in Fig. 3. Number of tokens for ngram is calculated by using the Eq. (3):

$$T_n = L - n + 1 \quad (3)$$

Where

- T_n=Total number of Tokens
- L=Length of the text
- n=Size of the ngram

For this experiment, the N value of the ngram is set between 3 and 7.

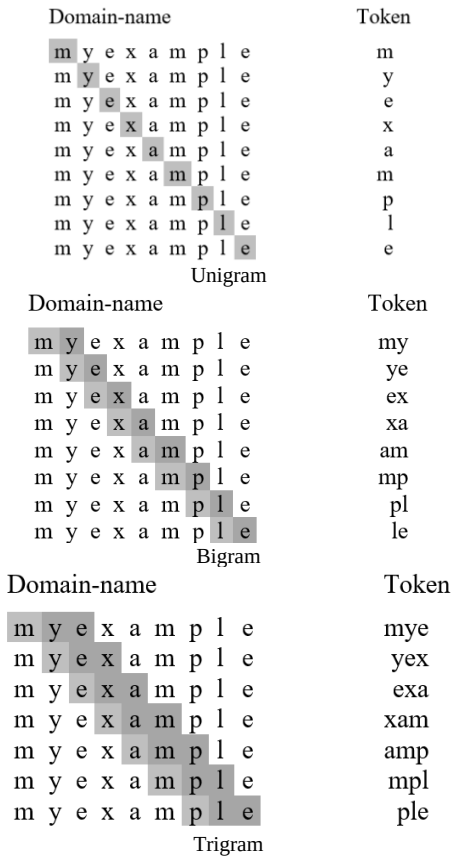

Fig. 3. Generating N-gram from the domain name.

D. Webpage Text Extraction

The process of extracting content from websites focuses on paragraph tags. The paragraph tag is a reasonable and straightforward way to organize the content of a web page because it is meant to denote a block of text that is unique from other blocks of text. After text extraction, text preprocessing begins. The preprocessing procedures are outlined in Algorithm 1.

E. Preprocess

The entire process of text vectorization is included in preprocessing. Text is cleansed by removing special characters and unnecessary components of the URL, such as the protocol, leaving only text and numbers. After text cleaning, it will be converted to lowercase, stop words will be removed, and lemmatized to reduce number of features during vectorization process. The procedures for preprocessing are summarized in Algorithm 2.

Algorithm 1: Text Extractions

```
Input: List of URLs (W)
Output: doc
Function Cont_Extraction (W, TagName)
Con=Connect(W)
Tag List = get Tag (Con, TagName)
For tag in TagList do
     doc = doc U getText(tag)
end for
                    return doc
```

The pseudo code of the method Cont_Extract () is used to extract textual contents from particular tags on the webpage. The Connect() function is used to establish the connection to the appropriate website. Following the establishment of the connection, the content of the given tag "TagName" is extracted and added to the string "doc". This "doc" will be used for preprocessing.

Algorithm 2: Data Preprocessing

```
Input: doc
Output: Corpus (T2 )
Function preProcess(doc)
Torg=read_Text(doc)
T1=clean_Text(Torg)
T1 = lower_Text(T1)
For token in T1 do
     if token is not in STOPWORDS of ENGLISH then
          T2 = T2 U Lemmatize(token)
     End if
end for
return T2
```

The pseudo-code of the preprocess() method is used to complete the preprocessing task for the given text document. The contents of the "doc" are read using the read_Text () function. The clean_Text() and lower_Text() routines convert text to uppercase and lowercase correspondingly. Stop words in a text are eliminated and lemmatized using the "for loop". The resultant corpus is to be vectorized.

F. Vectorization

Most machine learning algorithms take numeric feature vectors as input. Consequently, while working with text documents, required to convert each document into a numeric vector. This method is referred to as text vectorization. By employing various NLP techniques, such as count vectorizer, TFIDF, and hashing vectorizer, the generated tokens of text in the preprocessing and character level ngram are converted into a real-valued vector. The output of vectorization is a two-dimensional (2D) array. The vectorizer's features are set to 2500 and stored in a CSV file to limit the 2D array's size. As illustrated in Fig. 2, two files are created in our experiment, one for character level ngram tokens and another for textual content of the webpage (char level vec.csv & para text vec.csv).

G. Merging Files

After vectorization, three separate files (conv fs.csv, char level vec.csv & para text vec.csv) are combined into a single data file (combined.csv) as shown in Fig. 2. This file serves as a data file for the machine learning algorithm.

H. K-Fold Validation and Model Evaluation

The process uses a single parameter, K, to partition a data sample into k groups. When a particular number for k is selected, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation [15]. In our experiment, a machine learning algorithm using 10-fold cross validation is fed the combined dataset (combined.csv). Four important metrics—precision, recall, accuracy, and f1-score—are used to assess the machine algorithm's performance.

V. Experimental Result

The experimental configuration consists of Windows 10, an I5 processor (3.2 GHz), and 8 GB of RAM. For programming, Python and sklearn package are utilized. Three distinct features, including URL Linguistic Features (ULF), Character-level Ngram Features (CNF), and Web Content Features (WCF), are generated for the experiments. Table II lists the seven most popular machine learning techniques used in the experiments. Each feature set and feature set combination (ULF+CNF+WCF) is examined independently for performance evaluation. Three different vectorizers were used to generate the features. A range of features between 250 and 2000 was taken for each trial. This range is known as "feature base."

1) URL linguistic feature (ULF): The features are retrieved from the URL alone by counting some characters in the URL and checking for the presence of the required pattern or characters in the URL. Table III provides the 22 features extracted from the URL. These are the most common features seen in the majority of existing research works. Fig. 2 shows the conventional feature extraction module extracting features from the Raw dataset and preparing a 2D array of values where rows represent URLs and columns represent the 22 features. The last column is the dependent feature called "class" which shows if the URL is benign (0) or malicious (1). The experiment's outcomes reveal that the random forest

achieves an accuracy of 98.31%, as depicted in both Table IV and Fig. 4. This outcome distinctly underscores the appropriateness of the selected features for the experiment.

TABLE IV. PERFORMANCE OF ULF

Type	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
URL Linguistic Features	LogR	95.06	99.48	90.58	94.82
	KNN	96.80	97.67	95.91	96.77
	GNB	91.19	99.44	82.82	90.36
	DT	96.66	95.76	97.84	96.74
	RF	**98.31**	**98.74**	**97.90**	**98.31**
	GB	96.83	99.06	94.55	96.74
	XGB	97.03	96.99	97.39	97.12

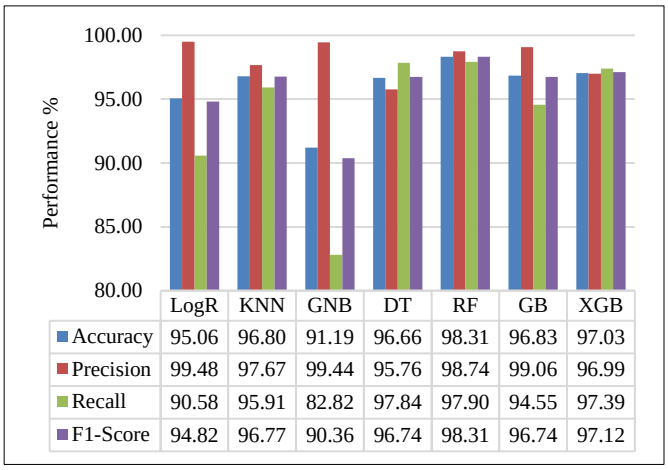

Fig. 4. Performance of ULF.

2) Character level-n-gram features (CNF): Character-level ngrams are generated by breaking a text into sequences of characters of a predetermined length. Character-level ngrams are useful for identifying patterns in text, specifically URL processing. In this experiment, character level ngram processing simply takes the URL's domain name into account. Character-level ngrams produce tokens. Following the generation of tokens, the tokens are vectorized using three distinct vectorizers for feature generation Experiments use seven machine learning algorithms to test the generated features and evaluate the performance. Tables V to VII show the outcomes of the various trials for the count, TFIDF, and hashing vectorizer, respectively. The results showed that using a 2000 feature base, count vectorizer + random forest achieves an accuracy of 90.87%, TFIDF vectorizer + random forest achieves an accuracy of 90.04%, and Hashing vectorizer + random forest achieves an accuracy of 92.95%. Fig. 5 depicts the performance comparison of three different vectorizers.

TABLE V. PERFORMANCE OF CNF WITH COUNT VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	70.11	66.55	92.84	76.65
	KNN	68.88	70.82	69.28	69.11
	GNB	67.32	63.47	99.35	76.50
	DT	72.78	68.72	92.08	78.00
	RF	72.38	68.55	91.94	77.70
	GB	70.65	66.64	94.38	77.28
	XGB	71.72	67.86	92.37	77.48
500	LogR	71.78	70.42	82.53	75.16
	KNN	76.65	75.60	82.19	78.28
	GNB	68.77	64.59	98.83	77.20
	DT	81.12	78.45	88.75	82.77
	RF	81.80	78.75	89.69	83.40
	GB	70.80	67.25	92.62	77.01
	XGB	76.27	73.45	87.47	79.23
1000	LogR	77.57	76.19	83.78	79.29
	KNN	77.33	88.71	62.86	73.51
	GNB	67.97	72.96	61.44	65.96
	DT	86.37	83.12	93.27	87.59
	RF	88.71	85.82	93.56	89.37
	GB	70.98	67.27	93.32	77.28
	XGB	78.37	77.64	84.59	80.17
1500	LogR	79.56	77.78	85.66	81.09
	KNN	78.27	87.31	66.59	75.43
	GNB	66.75	81.46	44.30	57.13
	DT	87.97	84.40	94.72	89.00
	RF	89.93	87.04	94.69	90.54
	GB	71.38	67.43	93.92	77.63
	XGB	78.23	76.77	86.54	80.47
2000	LogR	81.35	79.29	87.11	82.66
	KNN	79.16	86.77	69.42	76.98
	GNB	68.40	83.38	46.62	59.58
	DT	88.97	85.70	94.81	89.81
	RF	**90.87**	**88.37**	**94.69**	**91.30**
	GB	71.48	67.43	94.04	77.70
	XGB	78.82	77.53	86.66	80.93

Configuring the count vectorizer with 2000 features results in an elevated accuracy of 90.87% and an F1-Score of 91.30% when applied in conjunction with the random forest algorithm. By incorporating n-grams, the count vectorizer not only captures individual words but also preserves contextual information regarding word combinations.

TABLE VI. PERFORMANCE OF CNF WITH TFIDF VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	69.37	66.23	92.53	76.19
	KNN	69.84	69.60	75.97	71.87
	GNB	67.44	63.54	99.40	76.57
	DT	70.37	66.79	92.17	76.63
	RF	72.80	68.65	92.64	78.14
	GB	68.29	64.87	94.23	75.92
	XGB	72.62	68.57	92.76	78.10
500	LogR	71.38	70.61	80.73	74.41
	KNN	72.84	71.96	79.81	75.08
	GNB	68.87	64.67	98.64	77.20
	DT	78.71	75.75	88.79	81.11
	RF	79.54	76.41	89.61	81.87
	GB	72.43	68.32	93.35	78.07
	XGB	76.76	73.68	89.32	80.00
1000	LogR	76.57	74.83	84.05	78.64
	KNN	76.26	74.24	84.87	78.66
	GNB	69.97	73.48	66.84	69.21
	DT	85.18	81.71	93.32	86.74
	RF	85.68	82.14	94.09	87.29
	GB	72.96	69.63	91.23	77.90
	XGB	79.83	76.37	90.61	82.33
1500	LogR	77.62	75.61	85.61	79.74
	KNN	78.14	83.74	70.73	76.47
	GNB	67.86	82.42	46.14	58.90
	DT	87.41	84.21	94.29	88.62
	RF	87.09	83.48	94.92	88.46
	GB	72.02	69.52	89.67	77.08
	XGB	79.89	76.57	90.78	82.44
2000	LogR	78.49	76.05	86.92	80.59
	KNN	78.58	84.48	71.01	76.94
	GNB	69.49	84.35	48.27	61.18
	DT	86.54	83.12	94.75	88.10
	RF	90.04	87.01	95.17	90.72
	GB	69.44	67.31	88.18	74.98
	XGB	80.35	77.11	90.34	82.63

Configuring the TF-IDF vectorizer with 2000 features yields improved performance, achieving an accuracy of 90.04% and an F1-Score of 90.72% when coupled with the random forest algorithm. The incorporation of n-grams enhances the TF-IDF vectorizer, providing a more comprehensive representation of textual data.

TABLE VII. PERFORMANCE OF CNF WITH HASHING VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	67.97	67.40	79.14	71.94
	KNN	66.61	62.76	99.41	76.01
	GNB	62.36	61.28	88.25	71.06
	DT	88.79	85.36	95.85	89.94
	RF	90.51	87.86	95.51	91.26
	GB	76.04	72.25	90.26	79.61
	XGB	88.49	85.04	94.81	89.43
500	LogR	71.01	69.73	82.11	74.61
	KNN	66.38	62.67	99.35	75.90
	GNB	65.35	63.39	90.45	73.31
	DT	88.28	84.76	95.45	89.45
	RF	90.84	88.34	95.48	91.51
	GB	76.80	73.01	89.73	79.96
	XGB	87.31	83.54	94.37	88.39
1000	LogR	77.38	75.37	84.93	79.33
	KNN	66.18	62.54	99.38	75.81
	GNB	65.78	63.47	92.34	74.04
	DT	90.34	87.56	95.45	91.08
	RF	92.45	91.03	94.98	92.80
	GB	77.16	73.99	88.45	79.97
	XGB	87.75	84.24	93.89	88.63
1500	LogR	79.51	76.81	87.53	81.40
	KNN	66.09	62.48	99.34	75.75
	GNB	67.15	64.00	93.33	74.90
	DT	90.56	87.50	95.83	91.27
	RF	91.92	89.92	95.11	92.30
	GB	77.82	74.07	90.37	80.77
	XGB	86.73	82.76	93.92	87.80
2000	LogR	80.27	76.94	89.27	82.23
	KNN	66.08	62.48	99.40	75.76
	GNB	70.83	67.06	94.27	77.33
	DT	90.93	88.10	95.91	91.60
	RF	**92.95**	**91.84**	**94.86**	**93.20**
	GB	78.00	74.35	89.80	80.78
	XGB	87.31	83.80	93.62	88.23

Configuring the Hashing vectorizer with 2000 features yields an elevated accuracy of 92.95% and an F1-Score of 93.20% when coupled with the random forest algorithm. The incorporation of n-grams into the Hashing vectorizer enhances the representation of sequential word combinations, capturing contextual information and thereby improving the overall effectiveness of classification.

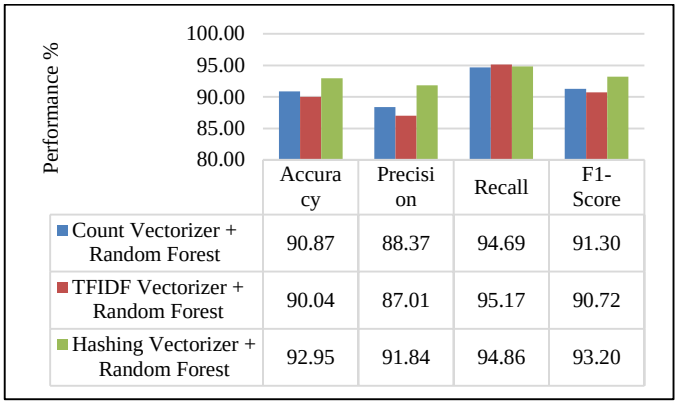

Fig. 5. Performance comparison of three vectorizers (2000 Features).

3) Web content features (WCF): The content of a webpage is the most informative option for analysis. Our experiment uses Beautiful Soup and Request in Python to analyze paragraph tag text. Text contents were preprocessed to get rid of stop words, special characters, etc. After that split the words into sentences to produce a corpus. This corpus serves as the input for the vectorizer, which produces features. To examine the generated features and assess performance, experiments utilize seven machine learning techniques. The results of the various trials for count, TFIDF, and hashing vectorizer, are displayed in Tables VIII to X. A performance comparison of three different vectorizers is shown in Fig. 6.

TABLE VIII. PERFORMANCE OF WCF WITH COUNT VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	77.28	72.72	91.60	80.58
	KNN	78.35	74.87	88.90	80.76
	GNB	69.36	64.26	98.21	77.02
	DT	87.69	82.71	97.04	89.04
	RF	90.71	86.94	97.10	91.53
	GB	78.43	74.47	91.23	81.41
	XGB	85.82	81.37	95.20	87.39
500	LogR	79.58	75.04	91.83	82.16
	KNN	80.21	76.47	90.03	82.26
	GNB	70.89	65.51	98.18	77.92
	DT	88.91	84.55	97.30	90.16
	RF	91.54	88.22	97.05	92.23
	GB	79.20	75.15	91.45	81.96
	XGB	86.37	82.07	95.28	87.84
1000	LogR	82.98	78.65	93.13	84.89
	KNN	80.54	75.54	93.27	83.10
	GNB	73.36	67.53	98.26	79.41
	DT	90.17	86.00	96.85	90.94
	RF	93.15	90.27	97.18	93.50
	GB	79.53	75.46	91.62	82.22
	XGB	88.41	83.86	96.33	89.45
1500	LogR	84.13	79.64	93.86	85.84
	KNN	79.78	74.94	92.64	82.43
	GNB	73.77	67.82	98.50	79.71
	DT	90.73	86.51	97.41	91.47
	RF	93.58	90.97	97.19	93.88
	GB	79.58	75.57	91.71	82.30
	XGB	88.75	84.25	96.42	89.73
2000	LogR	85.76	81.11	94.91	87.21
	KNN	80.06	75.33	92.48	82.61
	GNB	75.31	69.08	98.36	80.60
	DT	92.44	88.99	97.39	92.90
	RF	**94.01**	**91.58**	**97.36**	**94.29**
	GB	79.58	75.73	91.28	82.22
	XGB	88.85	84.40	96.34	89.79

TABLE IX. PERFORMANCE OF WCF WITH TFIDF VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	78.28	75.63	87.62	80.64
	KNN	77.98	73.51	91.94	81.18
	GNB	73.61	68.36	95.49	79.06
	DT	87.24	82.38	96.74	88.69
	RF	89.75	85.86	96.68	90.70
	GB	79.83	76.38	90.40	82.25
	XGB	86.04	81.87	94.74	87.50
500	LogR	80.89	78.40	88.58	82.68
	KNN	76.03	70.79	93.76	80.18
	GNB	77.05	71.43	95.45	81.19
	DT	88.29	83.51	97.22	89.57
	RF	91.78	88.68	96.96	92.44
	GB	80.80	77.42	90.60	82.97
	XGB	80.89	78.40	88.58	82.68
1000	LogR	83.34	80.96	89.58	84.65
	KNN	84.76	89.55	79.10	83.87
	GNB	77.60	71.70	96.71	81.81
	DT	90.46	86.08	97.31	91.20
	RF	93.27	90.52	97.10	93.60
	GB	80.18	76.59	91.11	82.65
	XGB	88.41	83.86	96.33	89.45
1500	LogR	85.03	82.53	90.65	86.07
	KNN	81.26	80.91	83.31	81.82
	GNB	79.19	73.32	97.16	83.00
	DT	92.42	89.07	97.19	92.86
	RF	93.70	91.12	97.25	94.00

	GB	80.75	77.16	91.37	83.11
	XGB	88.75	84.25	96.42	89.73
2000	LogR	86.13	83.92	91.02	87.01
	KNN	81.64	81.44	83.39	82.13
	GNB	81.59	75.71	97.04	84.59
	DT	92.20	88.62	97.31	92.67
	RF	**93.95**	**91.59**	**97.22**	**94.23**
	GB	79.54	75.69	91.31	82.21
	XGB	88.85	84.40	96.34	89.79

TABLE X. PERFORMANCE OF WCF WITH HASHING VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	76.18	75.93	80.69	77.65
	KNN	83.65	80.93	89.23	84.67
	GNB	75.37	77.30	73.99	75.22
	DT	92.06	89.50	95.92	92.47
	RF	95.16	94.97	95.55	95.22
	GB	82.40	79.80	88.85	83.73
	XGB	91.53	88.74	95.80	92.00
500	LogR	81.94	81.21	84.75	82.62
	KNN	84.69	82.20	89.56	85.53
	GNB	78.55	81.39	75.36	77.95
	DT	92.66	90.22	96.37	93.05
	RF	94.92	94.21	95.94	95.02
	GB	81.81	78.89	89.53	83.44
	XGB	92.18	89.63	95.71	92.50
1000	LogR	85.39	83.96	88.67	86.01
	KNN	85.52	83.11	89.95	86.23
	GNB	79.35	82.89	75.42	78.65
	DT	91.64	88.60	96.47	92.20
	RF	94.98	93.87	96.53	95.11
	GB	80.81	77.38	90.20	82.84
	XGB	90.65	87.25	95.89	91.23
1500	LogR	86.17	84.56	89.43	86.72
	KNN	86.01	83.98	89.78	86.62
	GNB	82.32	86.06	77.83	81.50
	DT	91.57	88.61	96.33	92.13
	RF	**95.21**	**94.02**	**96.80**	**95.33**
	GB	80.71	77.14	90.61	82.88
	XGB	90.10	86.23	96.16	90.78
2000	LogR	86.83	85.21	89.98	87.34
	KNN	86.50	84.88	89.36	86.94
	GNB	83.42	88.21	77.54	82.40
	DT	92.41	89.67	96.60	92.86
	RF	94.97	93.62	96.84	95.13
	GB	80.97	77.44	90.71	83.07
	XGB	90.72	87.04	96.31	91.31

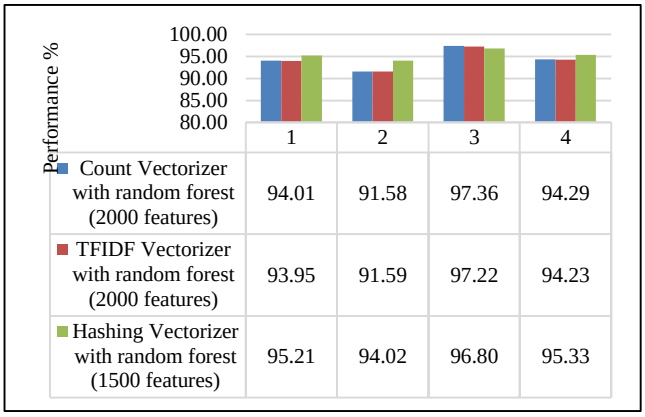

Fig. 6. Performance comparison of three vectorizers.

The findings demonstrated that when employing the 2000 feature base, the Count vectorizer with random forest achieves an accuracy of 94.01% and F1-Score of 94.29%. The TFIDF vectorizer with random forest reached an accuracy of 93.95% and an F1-score of 94.23%. However, the Hashing Vectorizer with Random Forest used a 1500 feature base and produced an accuracy of 95.21 % and an F1-score of 95.33%.

4) Combined features (CF): The combined feature set includes the features of ULF+CNF+WCF. Features are combined in row-wise. The main objective of combining feature sets is to reduce the feature set size and achieve higher accuracy. A range of features between 250 and 2000 was taken for each trial. So, the dataset for the trial is formed based on the following Eq. (4)

$$CF_n = ULF + CNF_n + WCF_n \quad (4)$$

Where n ϵ [250, 2000]

The features are merged in row-wise to create a single unified feature set. Tables XI to XIII show the outcomes of the various trials for the count, TFIDF, and hashing vectorizer, respectively. Fig. 7 depicts the performance comparison of three different vectorizers.

TABLE XI. PERFORMANCE OF CF WITH COUNT VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	94.48	94.49	94.67	94.53
	KNN	86.59	82.70	93.66	87.64
	GNB	73.54	68.14	98.77	79.87
	DT	96.79	94.83	99.32	96.95
	RF	98.84	98.51	99.23	98.85
	GB	97.42	99.63	95.18	97.35
	XGB	98.51	98.09	99.03	98.54
500	LogR	94.94	93.92	96.39	95.07

	KNN	85.47	82.87	90.71	86.37
	GNB	76.14	70.30	98.61	81.40
	DT	98.24	97.23	99.37	98.27
	RF	**99.38**	**99.63**	**99.12**	**99.37**
	GB	97.81	99.83	95.79	97.76
	XGB	99.12	99.14	99.10	99.12
1000	LogR	96.03	94.34	98.15	96.15
	KNN	85.19	82.17	91.43	86.27
	GNB	78.51	72.52	98.44	82.88
	DT	98.64	97.91	99.44	98.66
	RF	**99.38**	**99.57**	**99.18**	**99.37**
	GB	97.94	99.90	95.97	97.90
	XGB	99.18	99.19	99.18	99.18
1500	LogR	96.17	94.59	98.10	96.28
	KNN	85.94	82.80	92.24	86.98
	GNB	79.17	73.08	98.63	83.34
	DT	98.24	97.19	99.44	98.28
	RF	99.33	99.63	99.03	99.33
	GB	97.91	99.89	95.92	97.86
	XGB	99.15	99.10	99.21	99.15
2000	LogR	96.47	94.89	98.38	96.57
	KNN	86.22	82.68	92.99	87.29
	GNB	79.96	73.92	98.53	83.87
	DT	98.68	97.96	99.44	98.69
	RF	99.23	99.46	99.00	99.23
	GB	97.93	99.87	95.97	97.88
	XGB	99.14	99.14	99.15	99.14

TABLE XII. PERFORMANCE OF CF WITH TFIDF VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	95.76	96.84	94.66	95.72
	KNN	89.05	85.13	95.62	89.90
	GNB	73.40	67.83	99.75	79.97
	DT	95.65	93.79	99.24	96.17
	RF	99.27	99.38	99.15	99.27
	GB	97.48	99.75	95.20	97.42
	XGB	98.98	98.93	99.03	98.98
500	LogR	96.63	97.60	95.68	96.61
	KNN	89.79	88.37	92.03	90.06
	GNB	79.73	73.38	99.66	83.90
	DT	98.73	98.12	99.38	98.74
	RF	**99.33**	**99.54**	**99.12**	**99.33**
	GB	97.74	99.84	95.63	97.69
	XGB	99.07	99.08	99.06	99.07
1000	LogR	97.51	97.97	97.07	97.51
	KNN	86.96	83.04	94.75	88.23
	GNB	83.74	77.42	99.23	86.51
	DT	98.60	97.94	99.35	98.62
	RF	**99.35**	**99.48**	**99.21**	**99.34**
	GB	97.78	99.73	95.82	97.73
	XGB	99.26	99.30	99.23	99.26
1500	LogR	97.86	98.49	97.22	97.85
	KNN	87.98	84.95	95.52	89.42
	GNB	85.08	78.99	99.31	87.52
	DT	98.56	97.84	99.35	98.58
	RF	**99.38**	**99.68**	**99.07**	**99.37**
	GB	97.78	99.76	95.79	97.73
	XGB	98.98	98.93	99.03	98.98
2000	LogR	98.13	98.83	97.44	98.12
	KNN	78.09	73.13	97.48	82.73
	GNB	86.56	80.75	99.12	88.57
	DT	98.60	97.94	99.31	98.61
	RF	99.32	99.52	99.12	99.32
	GB	97.86	99.83	95.88	97.81
	XGB	99.18	99.13	99.23	99.18

TABLE XIII. PERFORMANCE OF CF WITH HASHING VECTORIZER

Feature Base	Machine Learning Algorithm	Accuracy	Precision	Recall	F1-Score
250	LogR	96.00	97.11	94.92	95.97
	KNN	91.71	90.58	93.32	91.87
	GNB	93.95	97.54	90.20	93.71
	DT	98.38	97.71	99.14	98.40
	RF	99.45	99.82	99.09	99.45
	GB	97.65	99.59	95.69	97.60
	XGB	99.41	99.63	99.20	99.41
500	LogR	97.19	97.87	96.53	97.18
	KNN	92.00	90.85	93.59	92.15
	GNB	91.93	90.71	93.89	92.16
	DT	98.26	97.43	99.18	98.28
	RF	99.15	99.29	99.03	99.15
	GB	97.68	99.87	95.48	97.62
	XGB	**99.50**	**99.68**	**99.32**	**99.50**
1000	LogR	97.57	98.43	96.71	97.55
	KNN	92.45	91.55	93.73	92.57
	GNB	90.58	87.13	96.30	91.28
	DT	98.57	97.96	99.23	98.58
	RF	99.33	99.68	98.98	99.33
	GB	97.80	99.79	95.79	97.75

	XGB	99.46	99.65	99.27	99.46
1500	LogR	98.17	99.25	97.07	98.14
	KNN	92.55	91.97	93.42	92.63
	GNB	90.27	86.22	96.70	90.99
	DT	98.52	97.75	99.37	98.54
	RF	99.29	99.54	99.04	99.29
	GB	97.97	99.87	96.06	97.93
	XGB	99.50	99.69	99.31	99.50
2000	LogR	97.91	98.71	97.11	97.90
	KNN	92.51	91.71	93.67	92.63
	GNB	91.97	88.63	96.90	92.46
	DT	98.44	97.63	99.35	98.47
	RF	99.03	99.06	99.03	99.03
	GB	97.86	99.70	96.00	97.81
	XGB	99.46	99.65	99.27	99.46

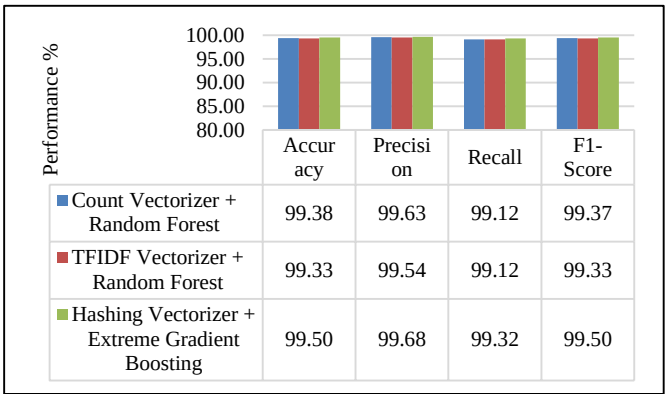

Fig. 7. Performance comparison of three vectorizers (500 Feature base).

The results showed that using a 500 features base, the Count vectorizer with random forest achieves an accuracy of 99.38% and an F1-score of 99.37%. The TFIDF vectorizer with random forest achieves an accuracy of 99.33% and an F1-score of 99.33% Hashing Vectorizer with extreme gradient boosting achieves an accuracy and F1-score of 99.50%.

All of the vectorizers employed in the experiment obtain higher accuracy and F1-score with 500 feature bases, especially the hashing vectorizer with extreme gradient boosting algorithm achieves the highest accuracy at 99.5%, as shown in Table XIV and Fig. 8. As compared to previous work, the proposed method significantly improves performance.

TABLE XIV. PERFORMANCE COMPARISON OF EXISTING AND PROPOSED METHOD

Author	Machine Learning Algorithm	Highest Accuracy
Saleem et al. [7]	Random Forest	93.0%
Saleem et al. [11]	Random Forest	99.0%
Malak et.al [12]	Naïve Bayes	96.0%
Kamel et.al [13]	Random Forest	95.51%
Lakshmanarao et.al[14]	Hashing vectorizer with Random Forest	97.50%
Joshi et.al [15]	Random Forest	92.0%
Hong et al. [16]	LA-FQ	94.16%
Josh et.al [17]	Random Forest	98.91%
Saleem et.al[18]	Weighted Soft Voting Classifier	99.10%
Ozgur et.al [19]	Random Forest	97.98%
Proposed Method	Hashing vectorizer with Extreme Gradient Boosting	99.50%

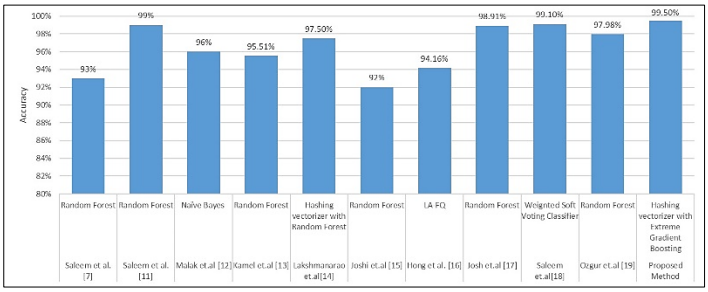

Fig. 8. Performance comparison of existing and proposed method.

VI. CONCLUSION

The majority of cybercrimes are committed using malicious links or malicious websites. Unintentionally visiting these websites or clicking on malicious links can have more serious effects, including the theft of private, sensitive information, security breaches, financial loss, and reputational damage. To find these kinds of dangerous websites on the internet, AI-based automated solutions are needed. For the detection method, this paper makes use of both URLs and web contents. Two different feature types, including ULF and CNF, are generated using URL of the webpage. Moreover, web page content is processed to generate features (WCF) for the detection procedure. Seven different machine learning methods are combined with three different vectorizers. Results of the study show that the proposed method, which combines an extreme gradient boosting algorithm with a hashing vectorizer, offers a better level of accuracy.

REFERENCES

[1] Data Breach Report. Internet: https://www.ibm.com/reports/data-breach. (Last access 18 oct 2024)

[2] Cybercrime magazine Internet: https://cybersecurityventures.com/cybercrime-damage-costs-10-trillion-by-2025/. (Last access 18 oct 2024)

[3] Hamadouche, Boudraa, Gasmi. Combining Lexical, Host, and Content-based features for Phishing Websites detection using Machine Learning Models. EAI Endorsed Transactions on Scalable Information Systems, vol.11, no.6, 2024. https://publications.eai.eu/index.php/sis/article/view/4421

[4] Saleem, Madhubala, Rajesh, Shaheetha, Arulkumar. Survey on Malicious URL Detection Techniques, 6th International Conference on Trends in Electronics and Informatics (ICOEI), pp. 778-781, 2022. DOI: 10.1109/ICOEI53556.2022.9777221.

[5] MohammadMoein, Arash, Hardhik,Unveiling malicious DNS behavior profiling and generating benchmark dataset through application layer traffic analysis,Computers and Electrical Engineering, vol.118, 2024. https://doi.org/10.1016/j.compeleceng.2024.109436..

[6] Shivika, Feature-Rich Models and Feature Reduction for Malicious URLs Classification and Prediction, Iowa State University, 2019. https://dr.lib.iastate.edu/handle/20.500.12876/16711.

[7] Saleem, Sundarvadivazhagan, Vijayarangan, Veeramani., Malicious Webpage Classification Based on Web Content Features using Machine Learning and Deep Learning, International Conference on Green Energy, Computing and Sustainable Technology (GECOST), pp. 314-319, 2022. DOI: 10.1109/GECOST55694.2022.10010386.

[8] Rong, Yan, Jiefan, Binbin., Detection of malicious web pages based on hybrid analysis, Journal of Information Security and Applications, vol 35, pp. 68-74, 2017. (https://doi.org/10.1016/j.jisa.2017.05.008).

[9] Cho Do, Hoa Dinh, Tisenko, Malicious URL Detection based on Machine Learning, International Journal of Advanced Computer Science and Applications (IJACSA), vol. 11, no. 1, 2020. DOI: 10.14569/IJACSA.2020.0110119.

[10] Jiann, Yi, Kuan., Intelligent Visual Similarity-Based Phishing Websites Detection, Symmetry, 12, 1681. 2020. (https://doi.org/10.3390/sym12101681).

[11] 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.

[12] Malak, Fahd, Rami, Samiha, Dina, Hanan, Sara., An Assessment of Lexical, Network, and Content-Based Features for Detecting Malicious URLs Using Machine Learning and Deep Learning Models. Computational Intelligence and Neuroscience., 2022. DOI: 10.1155/2022/3241216.

[13] Kamel, Boukhalfa, Zakaria , Oussama., 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. DOI: 10.1109/SNAMS52053.2020.9336572.

[14] 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. DOI: 10.1109/ICSES52305.2021.9633889.

[15] Saleem, Pradeepa, Justin, Madhubala, Hariraman, Vinodhini, SmishGuard: Leveraging Machine Learning and Natural Language Processing for Smishing Detection, International Journal of Advanced Computer Science and Applications, vol. 14, no. 11, 2023. DOI: 10.14569/IJACSA.2023.0141160.

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

[17] Jose, Ricardo , Emilio., 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.

[18] 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.

[19] Ozgur, Ebubekir, Onder, Banu., Machine learning based phishing detection from URLs, Expert Systems with Applications, vol.117, pp.345-357, 2019. https://doi.org/10.1016/j.eswa.2018.09.029.

[20] Benjamin, Rebecca, Tony., Applied Text Analysis with Python Enabling Language-Aware Data Products with Machine Learning, O'Reilly Media, Inc, 2018. ISBN-13: 978-1491963043.

[21] Aurélien Géron , Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, 2017. ISBN: 9781492032649.

[22] Malicious and Benign URLs: https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls.

[23] 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.

[24] 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.