American Journal of Interdisciplinary Research and Innovation (AJIRI)

ISSN: 2833-2237 (ONLINE

VOLUME 5 ISSUE 3 (2026)

PUBLISHED BY

E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Interdisciplinary Research and Innovation (AJIRI)

Volume 5 Issue 3, Year 2026
ISSN: 2833-2237 (Online)
DOI: https://doi.org/10.54536/ajiri.v5i3.8109
https://journals.e-palli.com/home/index.php/ajiri

Design and Implementation of a Machine Learning-Based Malicious URL Detection System

Oludele Adeleke[1], Jimoh Abdulhakeem Kuranga[2], Samuel Adeolu Ogunbiyi[2*], Endurance .O. Aneke[3]

Article Information

Received: April 21, 2026

Accepted: June 29, 2026

Published: September 01, 2026

Keywords

Cyber Security, Machine Learning, Malicious Url Detection, Phishing Detection, Random Forest, Tf-Idf, Url Classification, Xgboost

ABSTRACT

The internet has rapidly evolved in its communication, commerce and information sharing making it a huge platform for cyber threats, particularly malicious URLs. They pose a serious threat to individuals and to organisations. Phishing attacks, malware distribution and other types of cybercrime frequently are carried out through malicious URLs. In this research, we have created and tested the machine learning models to detect malicious URLs. The labeled URLs used were obtained from a public dataset with more than 651,000 labeled URLs. The dataset was prepared for classification by applying data pre-processing techniques like stratified sampling, label encoding and Term Frequency–Inverse Document Frequency (TF-IDF) vectorization. To train and test the algorithms, five machine learning were used: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Random Forest (RF) and Extreme Gradient Boosting (XGBoost), which were trained and evaluated by the metrics of accuracy, precision, recall and F1 score. The results indicated that the Random Forest model had the highest classification accuracy (95%) as compared to the other models. Moreover, a web based malicious URL detection system was developed to demonstrate the actual application of the developed models in real time cyber security scenarios. The study provides a conclusion that the machine learning techniques, particularly ensemble learning techniques can be considered an effective and reliable technique to detect malicious URL.

INTRODUCTION

Internet technologies have had a tremendous impact on the way people communicate, conduct business and access information globally and at a much faster rate. But the surge in reliance on online platforms has given a rise to the chances of cybercriminals attacking unsuspecting users through different cyber-attacks. The most frequently-used technique used by the attackers is the usage of malicious Uniform Resource Locators (URLs). Malicious URLs are fake URLs that are used to spread malware, aid phishing, steal sensitive information or hack computer systems (Mehndiratta *et al.*, 2023).

Generally, when users are tricked into revealing sensitive data through malicious links, they are likely to provide their username, password and financial information, often by mimicking legitimate websites like banks, government agencies, e-commerce sites and social networking sites (Geldenhuys, 2024). A lot of cyber threats including phishing, spreading malware, ransomware, identity theft and credential compromise are associated with them. They can cause a lot of financial loss, data breach, and damage to the reputation of individuals and organizations (Ogunbiyi *et al.*, 2026).

Various traditional malicious URL detection methods have been used to fight these threats, such as blacklist-based, heuristic-like, DNS monitoring, and email filtering methods. These approaches are efficient in terms of computation, but they fail to capture newly generated malicious URLs and/or those that constantly change (Kailas & Roopalakshmi, 2025). Likewise, heuristic techniques can also generate false positive and can be overcome by simply changing the structures of URLs (Osmanoğlu *et al.*, 2026). There has been a need for more adaptive and intelligent detection methods due to these restrictions.

The capabilities of detecting malicious URLs have greatly increased recently through the use of artificial intelligence (AI) and machine learning (AI). With machine learning algorithms, the patterns can be learned from a large dataset and known and unknown threats can be detected without using predefined patterns (Catak *et al.*, 2021). Some of the algorithms that have shown good results in this field of malicious URL classification are Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, Random Forest and Extreme Gradient Boosting (XGBoost) (Mehndiratta *et al.*, 2023; Imani *et al.*, 2025).

In this study, the authors have created and tested various machine-learning models, using a public data set of over 650,000 labeled URLs, to solve the issue of detecting malicious URLs. This study utilizes the data preprocessing, TF-IDF vectorization, model training and performance measurement using some metrics such as accuracy, precision, recall, and F1 score. In addition,

[1] Department of Information communication Technology, University of Ibadan, Nigeria
[2] Department of Computer Science, University of Ibadan, Nigeria
[3] Laboratory Technology, University of Ibadan, Nigeria
[*] Corresponding author's e-mail: ogunbiyisamuel001@gmail.com

the research shows how to practically implement a real time, web based malicious URL detection system that can classify these in real time. The results will be applied to cybersecurity to find the best way to use machine learning to identify malicious URLs and safeguard users against phishing, malware and other threats on the web.

LITERATURE REVIEW

This chapter summarizes the literature that covers malicious URL detection via machine learning algorithms. The review covers the malicious URL types, cybersecurity risks of malicious URLs, traditional methods for discovering malicious URLs, machine learning methods of detection for malicious URLs, machine learning algorithms used in malicious URL detection, and empirical review.

Types of Malicious URLs and Associated Cybersecurity Risks

The malicious URLs can be categorized based on how the attackers intend to use them and what they will do with them. A phishing URL is a URL that is posing as a legitimate site to fool the user into supplying sensitive information. The attackers build fake logon pages that appear to be from a known institution such as a bank, social networking site or online retailer. Users are encouraged to enter their username, password and financial information which is then collected by the cybercriminal (Dawood & Ibrahim, 2020).

Malware URLs spread malicious software such as viruses, worms, ransomware, spyware and trojans. Users can unknowingly be exposed to the risk of downloading malicious software when clicking on links, which may damage the security of the device, steal information or disrupt device operations (Aslan *et al.*, 2023).

Defacement URLs are URLs, which are modified by the attackers to direct to another site. Unapproved materials, propaganda or inflammatory items exist on the sites and are targeted at harming the reputation of organizations and institutions.

Popular methods for sharing spam URLs include spam email, social media postings and online ads. Their first goal is to bring users to their site, to peddle scams or to send users to a phishing or malware hosting site (Zhou *et al.*, 2024).

Individuals, companies and governments are at a high risk of being exposed to harmful URLs. Loss of sensitive information is the worst of all damage. Phishing attacks can result in identity theft, financial loss and unauthorized access to confidential systems. According to Cisco (2022), one of the primary reasons of cyber incidents around the world is still due to phishing.

Malicious URLs can also be used to infect with malware which can cause data damage, disruption and ransomware attacks. According to Aslan *et al.* (2023), malware distributed via malicious URLs is still a key generator of cybercrime losses. Targeted malicious URLs usually cause reputational damage, disruption of operations and regulatory penalties to organizations as a result of data breach. With the advancement of cyber-attacks, these concerns have grown even louder. Bad URLs detection is an integral component of any cybersecurity program as attackers are consistently coming up with creative methods to bypass existing detection mechanisms.

Traditional and Machine Learning-Based Approaches to Malicious URL Detection

In the traditional world, there are several ways to catch harmful URLs before they reach end users. Blacklist-based detection depends on databases of previously detected malicious URLs. When a user tries to access a URL, it is checked against the blacklist and blocked if it is listed. Conversely, whitelisting permits access to sites that are trusted and on known databases (Li *et al.*, 2024). The blacklist and whitelists are easy to develop and efficient from a computing perspective; however, they are not effective against newly formed malicious URLs because fraudsters are always creating new domains.

Heuristic detection systems examine URLs for specific properties that are typical of malicious URLs, such as suspicious keywords, unusual domain structures, an excessive number of special characters and uncommon top-level domains (Hasan, 2024). This approach makes it possible to identify unknown threats that are not present in blacklist databases. However, it can be easily bypassed by making minor changes to the URL syntax and is prone to false positives.

DNS-based detection solutions monitor DNS operations to detect malicious redirections and attempts to manipulate domains. The detection of DNS poisoning is done by identifying unauthorized changes to the DNS (Alsad & Abu Al-Haija, 2024), thus preventing users from being redirected to phishing sites. Email spoofing detection verifies the authenticity of email senders and identifies emails with malicious links (Alzahrani *et al.*, 2020). Although these methods are useful, they have limitations in detecting all forms of malicious URLs.

One of the most promising ways to find malicious URLs is to use machine learning. It can analyze large amounts of data and find patterns of malicious activity that were not detected before (Aryee *et al.*, 2025). Machine learning models can adapt to changing attack methods and increase detection capabilities over time as opposed to traditional rule-based systems (Catak *et al.*, 2021).

Machine learning-based detection approaches generally consist of several steps, such as data collection, data preprocessing, feature extraction, learning and testing (Ogunbiyi *et al.*, 2026). The various properties of the URL are translated into numeric values that can be used by classification procedures. The main advantage of machine learning is that it can find complex relationships between URL features that would not be apparent from manual inspection. This leads to higher accuracy in identifying both known and unknown malicious URLs.

Machine Learning Algorithms Used in Malicious URL Detection

There are multiple machine learning algorithms used for malicious URL detection. Support Vector Machine (SVM) is a supervised learning algorithm commonly used for classification problems (Ogunbiyi *et al.*, 2026). It categorizes harmful and benign URLs by computing an optimal hyperplane with a maximum classification margin.

K-Nearest Neighbors (KNN) categorizes a URL according to the category of the closest neighbors of the URL in the feature space. Easy to understand and highly interpretable, but high computational complexity with large datasets, so it is not scalable (Zhou *et al.*, 2024).

Naïve Bayes is a Bayesian naive classification algorithm. It has been demonstrated to be effective on text classification tasks and because of its simplicity and efficiency, it is a suitable baseline model for the malicious URL detection problem (Catak *et al.*, 2021).

Random Forest is an ensemble learning technique that relies on several decision trees through a random selection process to reduce overfitting and increase the accuracy of the prediction (Ogunbiyi *et al.*, 2026) demonstrated that the Random Forest classification algorithm is suitable for detecting malicious URLs with a satisfactory classification accuracy.

Extreme Gradient Boosting (XGBoost) is an enhanced ensemble learning approach (Ijiga *et al.*, 2024) which sequentially builds the models and fixes the weaknesses of the trees. The method incorporates regularization techniques to prevent overfitting and achieve good generalization. XGBoost is one of the latest algorithms that has proven to be one of the top algorithms for classification problem in cybersecurity as shown by Olatunji *et al.* (2026).

Empirical Review

There are a number of studies done for the malicious URL detection using machine learning algorithms. In the research done by Mehndiratta *et al.* (2023), they have analyzed which machine learning models are more accurate than traditional machine learning models and the results showed that ensemble learning techniques are more accurate than traditional classifiers. Similarly, Mosa *et al.*, (2023) used machine learning algorithms having lexical and host-based features and achieved a high accuracy in phishing URL detection.

Catak *et al.* (2021) compared the various classification techniques and observed that random forest and gradient boosting techniques outperformed Naïve bayes and KNN technique in all cases. Li *et al.* (2023) also demonstrated that other models such as machine learning and deep learning models can be further improved to detect malicious urls with greater accuracy, and can automatically learn complex feature representations.

The literature shows that ensemble classifiers have better generalization power and are more robust than other machine learning classifiers. However, there are problems with transparency of the model, computational requirements, and adaptation of the model to attack paradigms.

MATERIALS AND METHODS

Research Design

The present study used an experimental research design to explore the effectiveness of machine learning algorithms in identifying malicious URLs. The experimental approach was considered appropriate due to the possibility of developing, training, testing and evaluating multiple models of machine learning in simulated conditions. The study focused on the analysis and comparison of

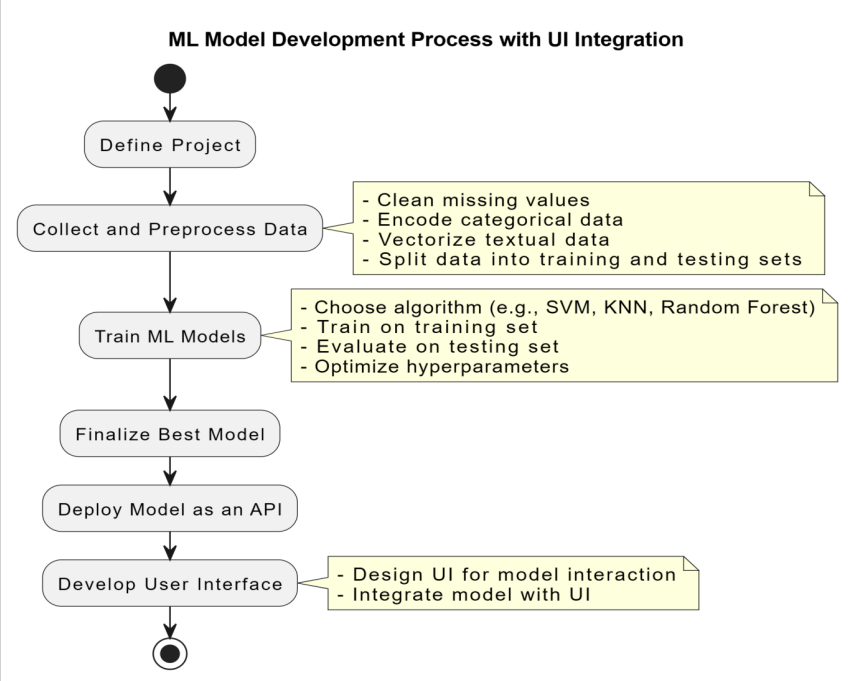

Figure 1: Model Flow

the classification accuracy of five machine learning algorithms: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Random Forest (RF), and Extreme Gradient Boosting (XGBoost) with a large-scale malicious URL dataset.

The study was conducted with the following overall framework: (i) dataset acquisition, (ii) data preprocessing, (iii) feature extraction, (iv) model training, (v) model evaluation and (vi) deployment of the best performing model. The objective was to identify which is the optimal algorithm to detect malicious and benign URLs with maximum accuracy, precision, recall and F1-score.

Dataset Description

The dataset utilized in this work has been acquired from the public Malicious URL DataSet (MUDS) from the Kaggle repository. Total amount of URL records in the dataset is 651,190, which are divided into four types: Benign, Phishing, Malware, Defacement. The data set has two primary attributes:

URL: Web address that is utilized as the predictor variable.

Type: Class label that matches to the benign/malicious URL type.

The dataset was chosen due of its size, the range of harmful URL categories and the appropriateness for machine learning based categorization investigations. Labeled samples were available, which allowed for supervised learning and for evaluating the performance of the models.

Data Preprocessing

The data underwent data pre-processing to enhance data quality and to prepare the data for machine learning research. A variety of pre-processing processes were used.

Missing Value Analysis

To check if there are any missing values in the data collection, Pandas in Python was used. No data means no bias and hence, no accuracy in prediction, which can have negative effects on the model performance. Analysis revealed that there were no missing data from the data set which means the data set was complete and can be used in the next study.

Stratified Sampling

There were more than 651,000 records in the original dataset, and computational costly to train the model. Stratified sampling was used to increase efficiency without affecting class distributions. Stratified sampling allows the percentage of each class to equal the percentage of the population. A percentage of 25% was used which resulted in a smaller data set with approximately 162,797 observations. This approach worked well in reducing the processing required and maintaining representativeness of the original sample of data.

Class Distribution Analysis

Exploratory data analysis was utilized to look at the distribution of URL types in the data set. The following class proportions were found for the analysis:

Benign: 65.74%

Defacement: 14.81%

Phishing: 14.45%

Malware: 4.99%

The results demonstrated considerable class imbalance, whereas the majority class were benign URLs. However, an adequate amount of examples for each category were available for training and testing the model's performance

Label Encoding

The dataset included the goal variable ("Type") with categorical labels that needed to be translated into numerical values for examination by machine learning models. Label Encoding was employed to encode the categories into numbers. All the URLs were assigned a unique numerical number during the encoding process, which allowed the machine learning algorithms to successfully operate with the target variable.

Feature Extraction

In the context of machine learning applications utilizing textual data, feature extraction is an important step. The URLs are strings of text and consequently cannot be directly utilized as input to machine learning techniques. For this reason, textual URLs were represented with integers by employing the approach of Term Frequency–Inverse Document Frequency (TF-IDF).

Dataset Partitioning

The data set was then preprocessed and then extracted features from the data set and then split into training data set and test data set using train, test and split technique. The data set was divided as follows: Training Set (80%) and Testing Set (20%). The training set was used to build the machine learning models and to train them; the testing set was saved for performance testing of the models with unseen data. This division enabled the un-biased analysis of the generalization capability of each model.

Machine Learning Algorithms

In this work, five machine learning methods were adopted and tested.

Support Vector Machine (SVM)

Supervised learning algorithm which identifies an optimum decision boundary between the classes is Support Vector Machine. SVM has proven to be a very effective technique in high-dimensional feature spaces and in text classification applications. The fundamental purpose of the algorithm is to maximize the margin between the classes, which will enhance the classification accuracy and avoid overfitting

K-Nearest Neighbors (KNN)

K-nearest Neighbors is a sort of instance-based learning method, in which observations are classified based on the labels of the nearby neighbors. The algorithm for each

URL instance discovers the K nearest observations in the training set and gives the most frequent class value among the neighbors. The KNN is straightforward to comprehend and interpret, but it can be time-consuming if employed with huge datasets.

Naïve Bayes

Naïve Bayes is a sort of a classification method based on the Bayesian Theorem. The classifier takes the assumption that all predictors are conditionally independent, and computes the posterior probability of a URL being member of given class. It is an excellent and efficient model in text classification, which is suited as a baseline model for malicious URL detection.

Random Forest

Random Forest is a form of ensemble learning algorithm, which creates numerous decision trees and then majority vote is used to determine the final outcome. The approach takes the aggregate of a large number of individual trees to reduce variance and enhance classification accuracy. Random Forest is especially effective for high-dimensional data and interactions between the features.

Extreme Gradient Boosting (XGBoost)

XGBoost is an improved ensemble learning model based on gradient boosting. The model generates decision trees one by one, and rectifies faults created by the preceding decision trees. XGBoost themselves contain regularization approaches that help with better model generalization and avoiding overfitting. The algorithm has been proved to outperform other machine learning algorithms in a range of applications.

Model Evaluation Metrics

The performance of the machine learning models was assessed using multiple evaluation metrics to ensure comprehensive analysis.

Accuracy

Accuracy measures the overall proportion of correctly classified instances out of all predictions made by the model. In other words, it is the ratio of corrected prediction over the total numbers of input samples as shown in equation 3.0

Formula is:

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

Where:

TP = True Positives

TN= True Negatives

FP = False Positives

FN = False Negatives

Precision

Precision quantifies the proportion of true positive predictions among all positive predictions made by the model. That is, the number of right positive result divided by number of classifiers predicted result as shown in equation 3.1

Formula is:

$$\textbf{Precision} = \frac{TP}{TP+FP} \qquad (2)$$

Recall

Recall measures the proportion of the real positive cases that the model accurately recognizes. That is, the number of true positive results divided by the number of all relevant samples as shown in equation 3.2

Formula:

$$\textbf{Recall} = \frac{TP}{TP+FN} \qquad (3)$$

F1-score

The F1-score is the harmonic mean of precision and recall, providing a balanced metric that takes into account both false positives and false negatives.

Formula:

$$\textbf{F1} = 2 \times \frac{Precision \times Recall}{Precision+Recall} \qquad (4)$$

System Implementation

All the machine learning models were coded in Python programming language in the Jupyter Notebook development environment. A number of libraries were used, such as:

Pandas for data manipulation, NumPy for numerical computation, Scikit-learn for machine learning implementation, Matplotlib and Seaborn for visualization and XGBoost library for gradient boosting classification. The best model was used in a web-based malicious URL detection system after model evaluation.

The front end was built with React and the trained machine learning model for real-time URL classification was incorporated into the back end. The final application was put online and made available to users as a safe platform to test URL safety.

RESULTS AND DISCUSSION

The chapter addresses the process of exploring the data, data preprocessing activities, algorithms training process, results of performance of the trained algorithms, comparison between the different machine learning algorithms and the implementation of the web-based malicious URL detection application. Classification metrics like accuracy, precision, recall, and F1-score were used to evaluate the performance of the Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest, Naïve Bayes, and XGBoost models.

Figure 2 shows the successful loading of the malicious URL dataset and the verification of missing values. The result confirms that both URL and Type attributes contained no null values.

Stratified sampling was used to reduce the computational complexity of the data set, while still maintaining the representation of the different classes in the original data set (which consisted of more than 651,000 records). All

Loading the Data

To begin, I will load the dataset from the csv file containing the malicious urls and their types. Python's `pandas` library makes it easy to work with structured data like this.

This step ensures that the data is accessible and correctly loaded for further analysis.

```
[2]: # Load data
data = pd.read_csv('malicious_phish.csv')
missing_values = data.isnull().sum()

print("Data loaded successfully!")

print(data)
```

```
Data loaded successfully!
                                                url        type
0                                    br-icloud.com.br    phishing
1                  mp3raid.com/music/krizz_kaliko.html      benign
2                      bopsecrets.org/rexroth/cr/1.htm      benign
3       http://www.garage-pirenne.be/index.php?option=...  defacement
4       http://adventure-nicaragua.net/index.php?optio...  defacement
...                                                   ...         ...
651186           xbox360.ign.com/objects/850/850402.html    phishing
651187      games.teamxbox.com/xbox-360/1860/Dead-Space/    phishing
651188        www.gamespot.com/xbox360/action/deadspace/    phishing
651189     en.wikipedia.org/wiki/Dead_Space_(video_game)    phishing
651190         www.angelfire.com/goth/devilmaycrytonite/    phishing

[651191 rows x 2 columns]
```

Figure 2: Dataset Loading

Step 2: Data Exploration and Visualization

Missing Values

In the first step, we check for missing values in the dataset to ensure that no critical data is absent. Missing data can lead to inaccuracies in our model predictions. The `isnull().sum()` method is used to identify the count of missing values in each column.

```
[3]: # Step 1: Data Exploration and Visualization
# Check for missing values
missing_values = data.isnull().sum()
print("Missing values:\n", missing_values)
```

```
Missing values:
 url     0
type    0
dtype: int64
```

Figure 3: Data Exploration Showing Missing Values

Stratification

In this section of the code, stratified sampling is performed to reduce the size of the dataset while maintaining the original ratio of classes.

Why Stratified Sampling?

- **Memory Efficiency**: Reducing the dataset size makes it easier to handle large datasets on systems with limited computational resources.
- **Preserving Ratios**: Stratification maintains the original class distribution, which is crucial for training balanced models and avoiding biases.

```
[4]: # Calculate the total number of samples to take (25% of the dataset)
sample_size = int(len(data) * 0.25)

# Perform stratified sampling to maintain the original ratio of types
stratified_sample = data.groupby('type', group_keys=False).apply(
    lambda x: x.sample(frac=sample_size / len(data), random_state=42)
)

# Reset index for the sampled data
data = stratified_sample.reset_index(drop=True)

# Display the sampled data
print(data)
```

```
                                              url      type
0       montreal.louer.com/ahuntsic-rentals/houses-rent/    benign
1           comunidade.sol.pt/blogs/hytigin/default.aspx    benign
2                      youtube.com/watch?v=PUDOlyCpHwo    benign
3                        uiowa.edu/~acadtech/phonetics/    benign
4                     flixster.com/actor/steve-mcqueen    benign
...                                                ...       ...
162792                  www.freewarelovers.com/android  phishing
162793                       tools.ietf.org/html/rfc2033  phishing
162794                www.science.uva.nl/events/HPCN99/  phishing
162795      en.wikipedia.org/wiki/D_programming_language  phishing
162796                www.ohiobusinesscollege.edu/CDL.asp  phishing
```

Figure 4: Stratified Sampling Output

URL categories were represented in the reduced data set in the same proportions as in the complete set, as the sampling process was stratified. This way, every category was adequately represented during the training and testing of the model as shown in Figure 4

Figure 5 shows the distribution chart that the majority of the data comes from benign URLs. Defacement and phishing URLs make up for approximately the same amount, with the least prevalent being the malware URLs. There were enough number of observations in all of the categories for good machine learning classification, although there was an imbalance of classes.

Figure 6 shows the classification reports of the five machine learning models used in this study. The reports include detailed performance statistics for each category of URL including benign, defacement, malware and phishing URLs. The Support Vector Machine (SVM) model was found to have an overall accuracy of approximately 94%. The model was efficient in classifying most of the URL categories, especially on malware URLs where it achieved a precision score of 1.00 and a recall score of 0.93. The model also performed remarkably well in the detection

Distribution of Malicious URL Types

We visualize the distribution of the target variable (`type`) using Seaborn's `countplot`. This gives an overview of the data balance, helping us understand whether the dataset is imbalanced or not.

```
[5]: # Calculate and print the statistics of the new sampled dataset
type_counts = data['type'].value_counts()
total_samples = len(data)

print("New Dataset Statistics:")
for t, count in type_counts.items():
    print(f"Type: {t}, Count: {count}, Percentage: {count / total_samples:.2%}")
```

```
New Dataset Statistics:
Type: benign, Count: 107025, Percentage: 65.74%
Type: defacement, Count: 24114, Percentage: 14.81%
Type: phishing, Count: 23528, Percentage: 14.45%
Type: malware, Count: 8130, Percentage: 4.99%
```

```
[6]: # Visualizing the distribution of the 'type' column
plt.figure(figsize=(8, 5))
sns.countplot(data['type'], palette='viridis')
plt.title('Distribution of Malicious URL Types')
plt.xlabel('Count')
plt.ylabel('Type')
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_2400\731615749.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data['type'], palette='viridis')
```

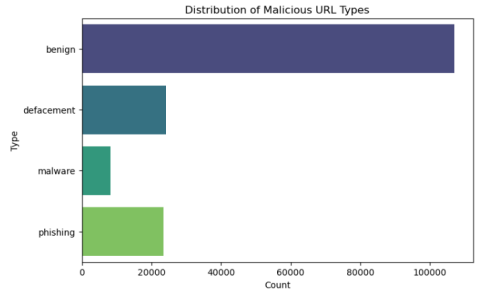

Figure 5: Distribution of Malicious URL Types

of benign and defacement URLs with F1-score of 0.96 and 0.98 respectively. However, the recall score of the phishing category was lower at 0.69, indicating that some phishing URLs were misclassified as other URL types. However, with this limitation, SVM appeared to be a very effective classifier for detecting malicious URLs.

The K-Nearest Neighbors (KNN) model was the worst performing model out of all the models tested, with

Classification Report for SVM:

	precision	recall	f1-score	support
benign	0.93	0.99	0.96	21211
defacement	0.97	0.99	0.98	4860
malware	1.00	0.93	0.96	1660
phishing	0.92	0.69	0.79	4829
accuracy			0.94	32560
macro avg	0.96	0.90	0.92	32560
weighted avg	0.94	0.94	0.94	32560

Classification Report for KNN:

	precision	recall	f1-score	support
benign	0.98	0.18	0.30	21211
defacement	1.00	0.72	0.83	4860
malware	1.00	0.62	0.76	1660
phishing	0.20	0.98	0.33	4829
accuracy			0.40	32560
macro avg	0.79	0.62	0.56	32560
weighted avg	0.87	0.40	0.41	32560

Classification Report for XGBoost:

	precision	recall	f1-score	support
benign	0.93	0.99	0.96	21211
defacement	0.96	0.98	0.97	4860
malware	0.97	0.93	0.95	1660
phishing	0.91	0.65	0.76	4829
accuracy			0.93	32560
macro avg	0.94	0.89	0.91	32560
weighted avg	0.93	0.93	0.93	32560

Classification Report for Naive Bayes:

	precision	recall	f1-score	support
benign	0.82	1.00	0.90	21211
defacement	0.96	0.90	0.93	4860
malware	0.98	0.75	0.85	1660
phishing	0.98	0.23	0.37	4829
accuracy			0.85	32560
macro avg	0.94	0.72	0.76	32560
weighted avg	0.88	0.85	0.82	32560

Classification Report for Random Forest:

	precision	recall	f1-score	support
benign	0.94	0.99	0.97	21211
defacement	0.98	0.99	0.98	4860
malware	0.99	0.93	0.96	1660
phishing	0.93	0.74	0.82	4829
accuracy			0.95	32560
macro avg	0.96	0.91	0.93	32560
weighted avg	0.95	0.95	0.95	32560

Figure 6: Classification Performance of Machine Learning Models

an overall accuracy of about 40%. However, the model could not classify benign, malware and defacement URLs accurately although it performed well in detecting phishing URLs with a recall score of 0.98. The low weighted average F1-score indicates that the model overall does not generalize well. This performance can be credited to the high dimensionality of the TF-IDF feature space which affects negatively distance-based classification algorithms like KNN.

The Random Forest model was the best performing classifier with an overall accuracy of around 95%. The model consistently generated high values for precision, recall, and F1-score for all URL categories. Benign URLs had an F1-score of 0.97, defacement URLs had an F1-score of 0.98 and malware URLs had an F1-score of 0.96. All the models performed better than all the other models, although phishing URLs achieved a slightly lower F1-score of 0.82. The better performance of Random Forest can be attributed to the ensemble learning architecture of combining the predictions of multiple decision trees to improve the classification accuracy and avoid over-fitting.

The Naïve Bayes classifier showed an overall accuracy of around 85%. The model showed good recall on benign URLs, correctly classifying almost all benign instances in the dataset. But the model performance dropped when it came to the classification of phishing URLs and the F1-score of that category was quite low. The assumption of feature independence is a simplification, but Naive Bayes delivered satisfactory classification results and showed the effectiveness of probabilistic learning techniques for text-based URL classification.

The XGBoost model also showed excellent predictive performance, with an overall accuracy of around 93%. The model has high precision and recall values for benign, malware and defacement URL categories. As SVM and Random Forest, phishing URLs were still the most difficult category to classify accurately. But, the iteratively learning process of XGBoost with gradient boosting created good overall results.

It is evident from the classification reports that classification of phishing URLs was the most difficult for all models. This challenge could be linked to the similarity in lexical between the phishing URL and the trusted URL, since the attackers aim to create their phishing URL similar to trusted URLs. Contrary to this, the URL of defection and malware was easier to be identified, the structure of the URL of defaction and malware had inherent features which can be easily learned by machine. Learning algorithms. Furthermore, the results show that the ensemble learning methods are effective in detecting malicious URLs. The values of accuracy, precision, recall, F1 score for both the Random Forest and the XGBoost were higher as compared to the traditional machine learning algorithms. They have been able to capture complex relationships in the URL dataset that have been key in their strong classification performance.

Figure 7 shows the interface that allows users to enter a URL for analysis. This is the key communication connection between the user and the detecting system. Users can choose any of the learned machine learning techniques to classify urls. This is a feature that enables comparing multiple machine learning algorithms inside of the deployed system. Lastly, in Figure 4.8, there is a detection button that triggers the classification process by passing the URL entered by the user to the selected machine learning model for prediction. It was able to correctly classify the test URL as a phishing URL with

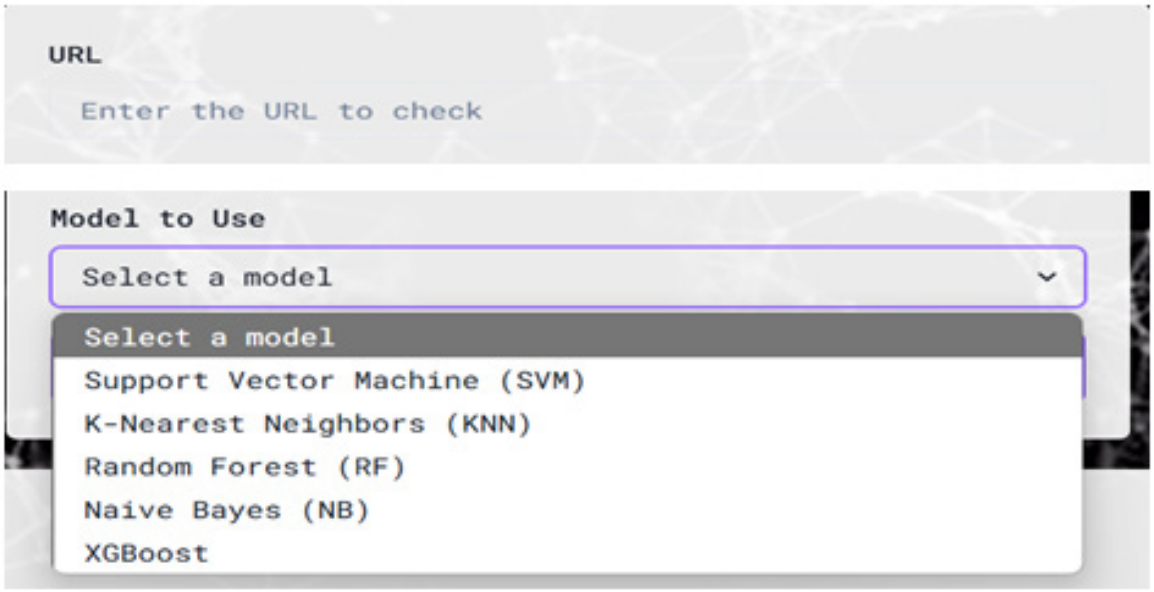

Figure 7: URL Input Interface.

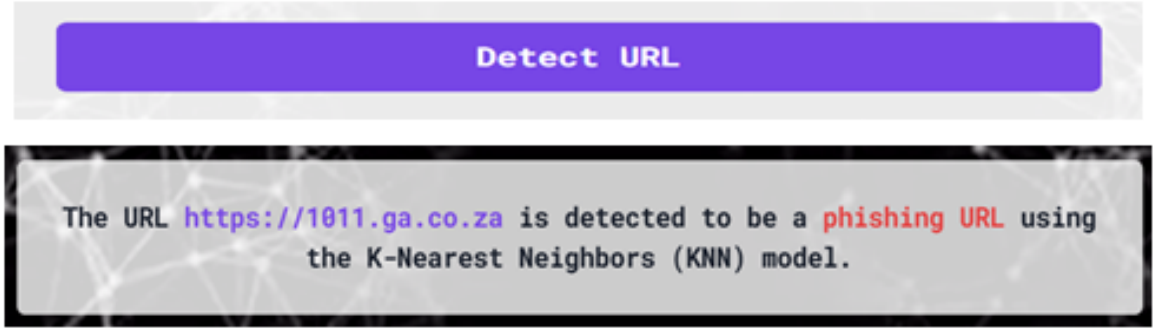

Figure 8: URL Classification Result

a K-Nearest Neighbors model. The result demonstrates the implementation of machine learning algorithms in real time harmful URL detection.

CONCLUSION

This work was able to build and evaluate a machine learning-based harmful URLs detection system. The work was based on a systematic machine learning approach including data preprocessing, stratified sampling, TF-IDF feature extraction, model training, and evaluation. The results suggested that machine learning algorithms are particularly effective in differentiating between benign and dangerous URLs. The best algorithm that was investigated was Random Forest with a classification accuracy of approximately 95%. Random Forest had the highest classification accuracy of all the algorithms investigated, around 95 percent. Both XGBoost and SVM produced wonderful performance and Naïve Bayes produced good performance. The KNN classification had the lowest accuracy and was the least applicable to the large-scale hazardous URL classification. The results also showed that ensemble learning methods achieve better performance than traditional machine learning methods, as they are able to capture complex patterns in the URL structure and minimize classification errors. While phishing URLs were the most difficult category to correctly identify, the results confirmed the effectiveness of machine learning in detecting a variety of harmful URLs, including phishing, malware and defacement types of attacks. The study eventually finds that machine learning, especially ensemble learning algorithm like Random Forest, XGBoost is a reliable, accurate and scalable solution for dangerous URL detection. The efficient implementation of the detection system signifies the importance of intelligent cybersecurity solutions in reducing online threats and enhancing the safety of internet users.

REFERENCES

Alsad, O., & Abu Al-Haija, Q. (2024). DNS cache poisoning attack detection: A systematic review. *IET Conference Proceedings, 2023*(44), 426–432. https://doi.org/10.1049/icp.2024.0962

Alzahrani, A., Alenazi, M., Alshammari, S., & Alharthi, R. (2020). Email spoofing attack detection through an end-to-end authorship attribution system. In *Proceedings of the 6th International Conference on Information Systems Security and Privacy* (ICISSP 2020) (pp. 64–74). SCITEPRESS. https://doi.org/10.5220/0008954600640074

Aryee, B. A., Umoren, J., & Agymang, K. A. (2025). Artificial intelligence for strengthening cybersecurity in U.S. healthcare systems. *American Journal of Medical Science and Innovation, 4*(2), 150–158. https://doi.org/10.54536/ajmsi.v4i2.6178

Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., & Akin, E. (2023). A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics, 12*(6), 1333. https://doi.org/10.3390/electronics12061333

Catak, F. O., Şahinbaş, K., & Dörtkardeş, V. (2021). Malicious URL detection using machine learning. *In A. K. Luhach & A. Elçi* (Eds.), Artificial intelligence paradigms for smart cyber-physical systems (pp. 160–180). IGI Global. https://doi.org/10.4018/978-1-7998-5101-1.ch008

Cisco. (2022). *Cisco event response: Corporate network security incident*. Cisco Security Center. https://sec.cloudapps.cisco.com/security/center/resources/corp_network_security_incident

Dawood, M., & Ibrahim, O. (2020). Exploration of hidden fraudulent website dataset with different perspective. *Journal of Computational and Theoretical Nanoscience, 17*(2–3), 980–984. https://doi.org/10.1166/jctn.2020.8738

Geldenhuys, K. (2024). Spoofing unmasked: Cheating criminals shatter trust. *Servamus Community-based Safety and Security Magazine, 117*(10). https://hdl.handle.net/10520/ejc-servamus_v117_n10_a6

Hasan, M. K. (2024). New heuristics method for malicious URLs detection using machine learning. *Wasit Journal of Computer and Mathematics Science, 3*(3), 60–67. https://doi.org/10.31185/wjcms.267

Ijiga, O. M., Idoko, I. P., Ebiega, G. I., Olajide, F. I., Olatunde, T. I., & Ukaegbu, C. (2024). Harnessing adversarial machine learning for advanced threat detection: AI-driven strategies in cybersecurity risk

assessment and fraud prevention. *Open Access Research Journal of Science and Technology, 11*(1), 1–24. https://doi.org/10.53022/oarjst.2024.11.1.0060

Imani, M., Beikmohammadi, A., & Arabnia, H. R. (2025). Comprehensive analysis of Random Forest and XGBoost performance with SMOTE, ADASYN, and GNUS under varying imbalance levels. *Technologies, 13*(3), 88. https://doi.org/10.3390/technologies13030088

Kailas, S., & Roopalakshmi, R. (2025). 'Think before you click' - Malicious URL detection in cybersecurity: A systematic review and research roadmap. *IEEE Access, 13*, 154305–154325. https://doi.org/10.1109/ACCESS.2025.3601387

Li, M., Jiang, Y., Zhang, Y., & Zhu, H. (2023). Medical image analysis using deep learning algorithms. *Frontiers in Public Health, 11*, 1273253. https://doi.org/10.3389/fpubh.2023.1273253

Li, W., Manickam, S., Chong, Y.-W., Leng, W., & Nanda, P. (2024). A state-of-the-art review on phishing website detection techniques. *IEEE Access, 12*, 187976–188012. https://doi.org/10.1109/ACCESS.2024.3514972

Mehndiratta, M., Jain, N., Malhotra, A., Gupta, I., & Narula, R. (2023). *Malicious URL: Analysis and detection using machine learning.*

Mosa, D. T., Shams, M. Y., Abohany, A. A., El-Kenawy, E. M., & Thabet, M. (2023). Machine learning techniques for detecting phishing URL attacks. *Computers, Materials & Continua, 75*(1), 1271–1290. https://doi.org/10.32604/cmc.2023.036422

Ogunbiyi, S. A., Adeleke, O., & Osunade, O. (2026). An ensemble machine learning framework for automated cybersecurity incident classification using structured metadata. *SADI International Journal of Science, Engineering and Technology, 13*(2), 1–11. https://doi.org/10.5281/zenodo.20025311https://doi.org/10.5281/zenodo.20025311

Olatunji, B. T., Adeleke, O., Osunade, O., & Asoro, B. (2026). Intelligent classification of healthcare incident reports using supervised learning. *SADI International Journal of Science, Engineering and Technology (SIJSET), 13*(2), 25–36. https://doi.org/10.5281/zenodo.20542825

Osmanoğlu, M., Gupta, D., & Sharma, V. (2026). *A comprehensive review of malicious URLs: Detection techniques, features and datasets. Computers & Electrical Engineering, 136*, 111186. https://doi.org/10.1016/j.compeleceng.2026.111186

Zhou, J., Ye, Z., Zhang, S., Geng, Z., Han, N., & Yang, T. (2024). Investigating response behavior through TF-IDF and Word2vec text analysis: A case study of PISA 2012 problem-solving process data. *Heliyon, 10*(16), e35945. https://doi.org/10.1016/j.heliyon.2024.e35945