**PROJECT-1 REPORT ON**

# COMPARISON OF MACHINE LEARNING ALGORITHMS TO DETECT PHISHING WEBSITE

**SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND TECHNOLOGY**

**BY**

**46 - Unnati Pimple**

**47 - Tanisha Purohit**

**50 - Surabhi Raut**

**Guided By**

**"Prof. Monica Charate"**

**DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY**

**USHA MITTAL INSTITUTE OF TECHNOLOGY**

**S.N.D.T Women's University,Mumbai**

**2022-23**

# CERTIFICATE

This is to certify that Unnati Pimple, Tanisha Purohit, Surabhi Raut has succesfully completed minor project report on the topic ”COMPARISON OF MACHINE LEARNING ALGORITHM TO DETECT PHISHING WEBSITE” satisfactorily in the partial fullfillment for the Bachelor’s Degree in COMPUTER SCIENCE AND TECHNOLOGY under the guidance of MS. Monica Charate during the year 2024-25 as prescribed by S.N.D.T Women’s University,Mumbai.

Guide

Prof. Monica Charate

Head Of Department

Prof.Kumud Wasnik

Principal
Dr. Yogesh Nerkar

Examiner 1

Examiner 2

# Acknowledgement

We would like to express our sincere appreciation to everyone who gave their invaluable assistance and support in this project. We are highly indebted to Prof. Monica Charate for her guidance and constant supervision as well as for providing necessary information regarding the project also for her support in this project. Without her help, this project would not have been possible.

We would also like to Thank our Head of Department Prof.Kumud Wasnik for her insights in shaping the direction and content of this report. We would also like to extend our gratitude to our classmates who provided valuable feedback and encouragement throughout the process.

Finally, we would like to express our appreciation to our families for their unwavering support and understanding during the long hours and late nights spent working on this project.

Thank you all for your contributions, support, and encouragement.

Date: 18/03/24

Unnati Pimple(46)
Tanisha Purohit(47)
Surabhi Raut(50)

# Abstract

Phishing attacks are a rapidly expanding threat in the cyber world, costing internet users billions of dollars each year. It is a criminal crime that involves the use of a variety of social engineering tactics to obtain sensitive information from users. Phishing techniques can be detected using a variety of types of communication, including email, instant chats, pop-up messages, and web pages. This study develops and creates models that can predict whether a URL link is legitimate or phishing.

Phishing websites seem to like the appropriate ones and it is difficult to differentiate among those websites. Most of the phishing URLs use HTTPS to avoid getting detected. There are three ways for the detection of website phishing. The primitive approach evaluates different items of URL, the second approach analyzing the authority of a website and calculating whether the website is introduced or not and it also analyzing who is supervising it, the third approach checking the genuineness of the website.

This application is made using Google Collab(Python), Java Script , HTML and CSS. This project compares five machine learning algorithms 1.Decison tree 2.Convolutional Neural Network 3.Logistic Regression 4.Random Forest 5.XGBoost 6.KNeighboursClassifier to detect phishing websites.

# TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

In the era of digital transformation, where online interactions have become ubiquitous, the prevalence of phishing attacks poses a grave threat to cybersecurity. Phishing, a deceptive practice aimed at tricking individuals into divulging sensitive information such as passwords, credit card details, or personal identifiers, remains one of the most prevalent cyber threats facing individuals and organizations worldwide. Phishing attacks often manifest through malicious websites masquerading as legitimate entities, making their detection and mitigation a paramount concern for cybersecurity professionals.

Traditional rule-based approaches to phishing website detection have shown limitations in keeping pace with the dynamic and sophisticated tactics employed by cybercriminals. In contrast, machine learning algorithms offer a promising avenue for enhancing detection capabilities by leveraging patterns and features inherent in phishing websites. However, the selection of an appropriate machine learning algorithm for phishing detection requires careful consideration of factors such as detection accuracy, computational efficiency, scalability, and adaptability to evolving threats.

This project embarks on a comprehensive comparative analysis of machine learning algorithms for detecting phishing websites, aiming to provide insights into their effectiveness, robustness, and suitability for real-world applications. By evaluating a diverse array of algorithms across various performance metrics, this study seeks to shed light on the strengths and limitations of different approaches, empowering cybersecurity practitioners and organizations to make informed decisions regarding their choice of detection techniques.

Through the systematic evaluation of machine learning algorithms, this project endeavors to contribute to the advancement of cybersecurity strategies in combating phishing threats. By identifying the most effective algorithms and elucidating their underlying mechanisms, this research aims to foster the development of more resilient and adaptive defenses against phishing attacks, safeguarding individuals and organizations from the perils of cybercrime in the digital age.

# CHAPTER 2: PROBLEM STATEMENT

The selection of the most suitable machine learning algorithm for phishing website detection remains a daunting task, characterized by consideration such as Detection Accuracy. Moreover, the abundance of machine learning algorithms available presents a challenge in identifying the optimal approach for a given set of circumstances.

Therefore, the primary objective of this project is to conduct a comprehensive comparative analysis of machine learning algorithms for detecting phishing websites. By systematically evaluating a diverse range of algorithms across various performance metrics based on various features extracted from website content,structure,and user interactions.

# LITERATURE SURVEY

| Paper Name,Author,Year of Publishment. | Methodology and Technologies | Observations/Findings |
|---|---|---|
| 1. A Machine learning approach for phishing attack detection-Tarun Choudhary,Siddhesh Mhapankar,Rohhit Buddha(2023) | This paper explores the use of machine learning techniques,including Extreme Gradient boosting,Decision Tree,Logistic Regression,and Support vector machine,to detect phishing attack. | Proposed approach results tested on UCI dataset. The proposed methodology Random Forest yielded good results with an accuracy of 98.80% in comparison with the other researcher. |
| 2. Detection of Phishing website Using Machine Learning-Perla Hari Priya(2023) | This paper willl delve into the key components of phishing website detection using machine learning models such as Decision Tree, Random Forest, SVM, KNN, GBM, exploring the features and datasets used for training, the algorithms employed for classification. | Machine learning for phishing website detection has the potential to be more accurate and effective than traditional methods such as blacklists or heuristicbasedsystems. Therefore, one of the main challenges is security, specifically how to encourage users to protect themselves against phishing.The data sample size was also a limitation. |
| 3.Detection of Phishing website using ML-Abdul Karim et.alt | It focuses on learning-based phishing detection systems and presents a hybrid machine learning approach that employs a variety of models, including decision trees, linear regression, naive Bayes, gradient boosting classifier, K-neighbors classifier and a proposed hybrid LSD model. | The proposed approach achieves its goal with effective efficiency. Future phishing detection systems should combine list-based machine learning-based systems to more efficiently prevent and detect phishing URLs |
| 4.Phishing Website detection using machine learning and deep learning techniques-M Selvakumari ,Sowjana et al(2021) | The phishing website detection model has been tested and trained using many classifiers and ensemble algorithms to analyze and compare the model's result for best accuracy. Each algorithm will give its evaluated accuracy after all the algorithms return its result. each is compared with other algorithms to see which provides the high accuracy percentage. | All the algorithms are tested<br>1.Logistic Regression<br>2.K Nearest Neighbour<br>3.Decision tree<br>4.Random Forest<br>5.Ada Boost |
| 5.Detection of Phishing Website using Machine Learning Approach-Mahajan Mayuri,JayPralah,Prachi Ghansham,Pawar Sila(2019) | This research focuses on detecting phishing websites using anti-phishing solutions,discussing various approaches,including heuristic classification of URLs and combined solutions utilizing whitelist and blaclist. | Extreme Learning Machine is used to detect phishing website.The result obtained by extracting features are extracted by its URLs.It notifies user in advance. |

# CHAPTER 4: EXISTING SYSTEM

1.Data Collection Module:

This module is responsible for gathering a diverse dataset comprising both legitimate and phishing websites. The dataset should include features such as URL characteristics, domain age, SSL certificate details, HTML and JavaScript analysis, presence of suspicious keywords, etc.

2.Feature Selection Module:

Feature selection techniques are applied to identify the most relevant attributes for distinguishing between legitimate and phishing websites. This helps in reducing dimensionality and improving the efficiency of machine learning algorithms.

3.Algorithm Selection Module:

Various machine learning algorithms are chosen for comparison, including decision trees, random forests, support vector machines, logistic regression, neural networks, k-nearest neighbors, etc. Each algorithm is configured with appropriate hyperparameters and optimization strategies.

4.Comparison Analysis Module:

The results obtained from the evaluation metrics are compared across different machine learning algorithms. Statistical tests may be conducted to determine significant differences in performance.

5.Continuous Monitoring and Updating:

The system should incorporate mechanisms for continuous monitoring of algorithm performance and updating of detection models to adapt to emerging phishing tactics and evolving datasets.

# CHAPTER 5: PROPOSED SYSTEM

The proposed phishing detection system utilizes different machine learning models and deep neural networks. The system comprises of two major parts, which are the machine learning models and a web application. These models consist of Convolutional Neural Network,Random Forest,XGBoost,Decision Tree,Logistic Regression,KNeighbourClassifier.

These models are selected after different comparison based performance of multiple machine learning algorithms. Each of these models is trained and tested on a website content-based feature, extracted from both phishing and legitimate dataset.

In **Random Forest** the dataset is splitted into trained data and test data. The dataset is reduced by 80is again reduced by 25the detection of phishing URLs. The algorithm builds a ”forest” of decision trees by first building several of them. To help prevent overfitting, each decision tree is trained on a portion of the data and employs a random selection of features. You need a dataset of URLs that have been classified as phishing or legitimate before using Random Forest to detect phishing URLs. From URLs, you can gather a variety of information, including the length of the URL, the number of subdomains, the presence of particular keywords, and more. The Random Forest model receives these characteristics as input.

The proposed phishing URL detection model comprises a multi-head self-attention attention mechanism and a **CNN**. To further boost the detection accuracy, a data augmentation technique based on GAN was proposed. This technique aims to balance the training data, enhancing the proposed model in four main parts: input, attention, feature extraction, and output. As input, the model accepts URLs in text format converted to 2D matrixes and generates output prediction using a binary classifier to judge whether or not it is a phishing URL.

**Logistic regression** is a powerful and interpretable algorithm for detecting phishing websites by modeling the probability that a given website is malicious based on its features. By learning the optimal coefficients during training, the model can effectively classify new websites as phishing or legitimate.

**Decision Trees** are highly interpretable. We can visualize the decision-making process, which helps in understanding which features are important for classification. For example, a decision tree for detecting phishing websites might show that features like URL length, presence of certain keywords, and SSL certificate validity are important indicators.

**XGBoost** (Extreme Gradient Boosting) is a robust machine learning algorithm widely utilized for binary classification tasks like detecting phishing websites. It operates by sequentially building an ensemble of decision trees, each correcting the errors of the previous ones. XGBoost optimizes an objective function incorporating a loss function and regularization terms, employing gradient descent to minimize it. Key features include regularization techniques like shrinkage, tree depth control, and subsampling to prevent overfitting. Its effectiveness lies in its ability to capture complex relationships in the data while controlling model complexity, making it a powerful tool for accurately identifying phishing websites.

**The k-Nearest Neighbors (k-NN) classifier** is a simple yet effective algorithm for detecting phishing websites. It works by comparing a new website to labeled examples in a training dataset and classifying it based on the majority class among its k nearest neighbors. In the context of phishing detection, features such as URL length, domain age, presence of certain keywords, and SSL certificate validity can be used to measure similarity between websites.

Hence, the model with the highest accuracy is selected and integrated to a web application that will enable a user to predict if a URL link is phishing or legitimate.

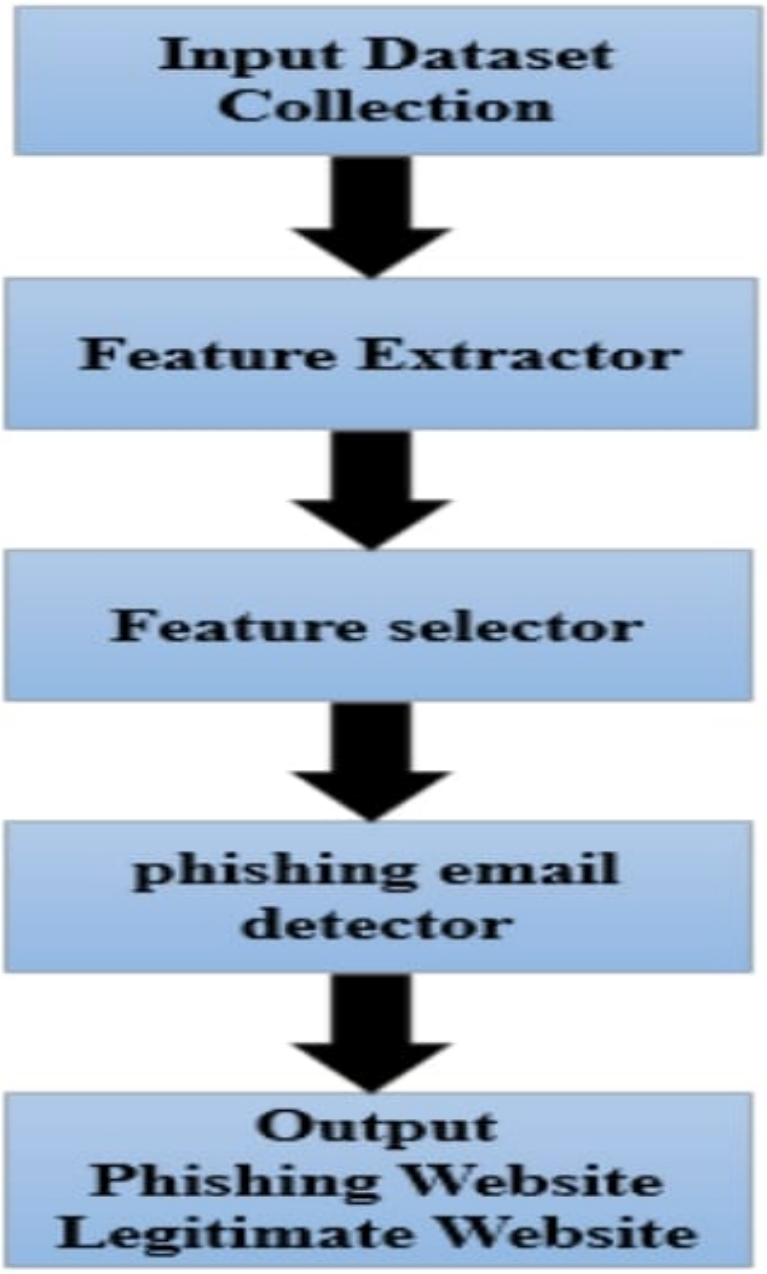
# CHAPTER 6: METHODOLOGY

Dataset: To train an excellent model, it must be designed with a large dataset. Because of the increased number of phishing attacks in recent years, there is a greater need for high-quality and new datasets for improved model performance.This module is in charge of acquiring the phishing and legitimate website datasets obtained from the UCI Machine Learning Repository ("UCI Machine Learning Repository: Phishing Websites Data Set.," 2016). This dataset contains 4898 phishing websites and 6157 legitimate websites from which many website features were extracted.

Feature Extraction: The process of identifying the characteristics that distinguish phishing and legitimate web pages is referred to as feature extraction. This procedure is crucial because the features chosen affect the accuracy and speed of phishing detection.To detect phishing effectively, the properties describing web pages must take into account the strategies and practices commonly used by attackers to create phishing web pages, as well as the characteristics that distinguish these pages from their legitimate counterparts.They outlined that the main sources of content are page URLs and source codes.

Feature Selection: Feature selection refers to the process of selecting the best features from among those extracted to classify a page as legitimate or phishing. This process is critical for making models parsimonious, avoiding overfitting, improving accuracy, and reducing computation requirements, especially when there are a large number of features. The papers develop traditional feature selection techniques, such as filter methods and wrapper methods, in order to identify the best subset of features for phishing website detection.

Following diagram represents the workflow for detecting a phishing website:
1.Firstly it collects the input dataset.
2.It extracts the important features of the URLs.
3.Features are selected then.
4.Then the website is detected using the best algorithm we have selected based on best accuracy.
5.Website is detected whether its legitimate or not.




WORKFLOW OF DETECTING A PHISHING WEBSITE

# CHAPTER 7: HARDWARE & SOFTWARE REQUIREMENT

Hardware Interface:

1.Adequate Computational Resources

2.Sufficient Memory(RAM)

3.Appropriate Storage(SSD)

4.Stable Network Connectivity

5.Backup and Redundancy Solutions

Software Interface:

1.Programming Language(PYTHON)

2.Data Processing tools(PANDAS,NUMPY)

3.Feature Extraction Libraries

4.Development Environment(Google Collab)

# CHAPTER 8: IMPLEMENTATION

## 1.User Interface

We have created a user interface for the users to provide smooth and friendly access to the system. Our user interface is named as "PhishGuard". It provides with a text bar where the user is expected to paste the URLs,and by clicking the blue predict button it will predict whether the website is legitimate or phished.
This interface also has various study modules which states what phishing is ,why should we know about phishing,major types of phishing,and preventive measures for phishing.

## 2.Model Training

Step 1:
Data Collection-

Obtain a dataset containing features extracted from both legitimate and phishing websites. These features may include URL characteristics, domain information, HTML content, website traffic, etc.

Step 2:
Data Preprocessing-

Clean the dataset by handling missing values, removing duplicates, and dealing with outliers if any.Encode categorical features into numerical format using techniques like one-hot encoding or label encoding.Split the dataset into features (independent variables) and the target variable (whether the website is phishing or not).

Step 3:
Feature Engineering/Selection-

Analyze the dataset to identify relevant features that contribute most to phishing detection.Perform feature engineering to create new informative features from existing ones if necessary.Select a subset of features that are most relevant for training the model to improve computational efficiency and prevent overfitting.

Step 4:
Model Selection-

Choose a suitable machine learning algorithm for training. Commonly used algorithms for phishing website detection include decision trees, random forests, support vector machines, lo-

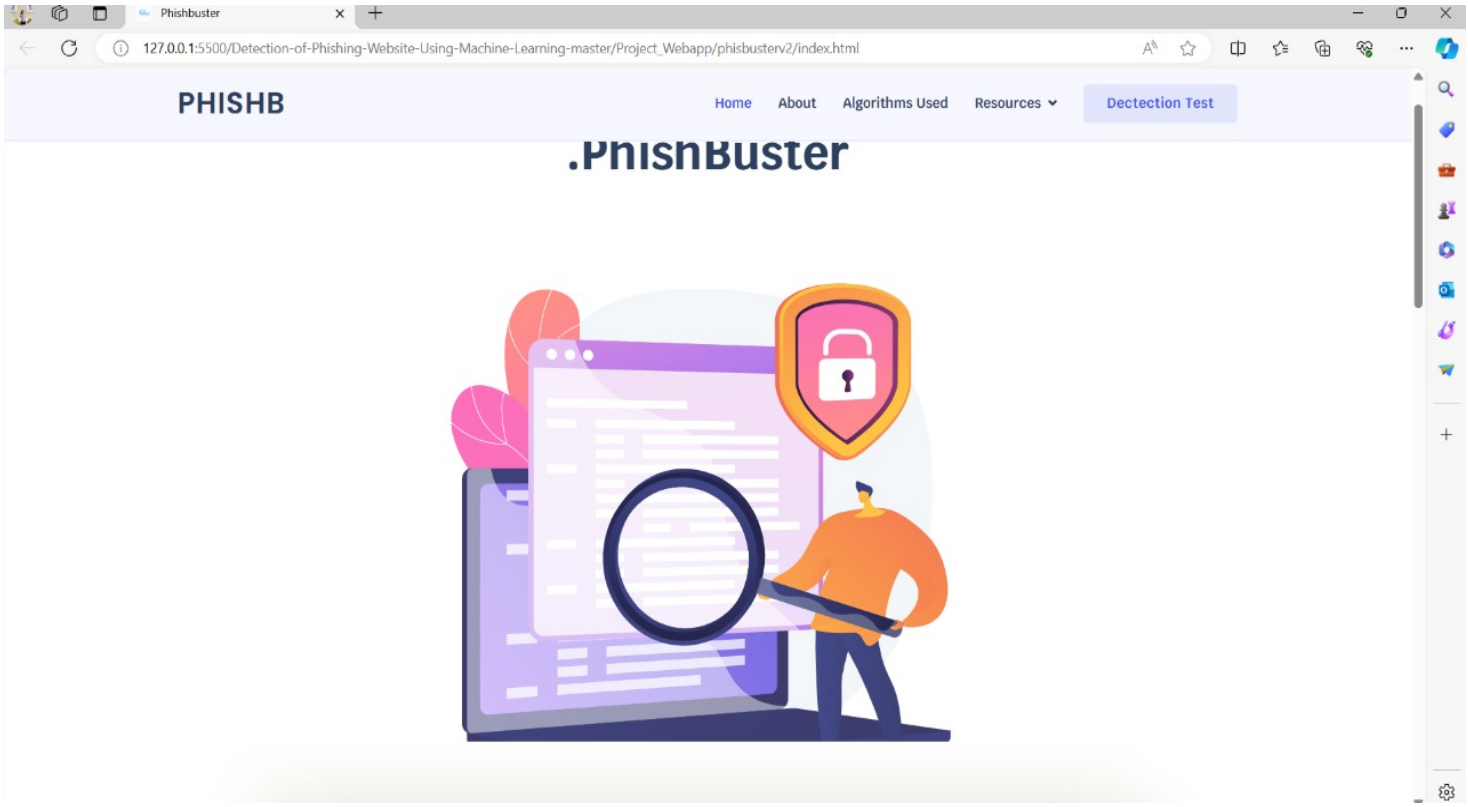
gistic regression, naive Bayes, and ensemble methods.Consider the characteristics of the dataset, such as its size, complexity, and nature of the features, to determine the most appropriate algorithm.

Step 5:
Training and Testing the Model-

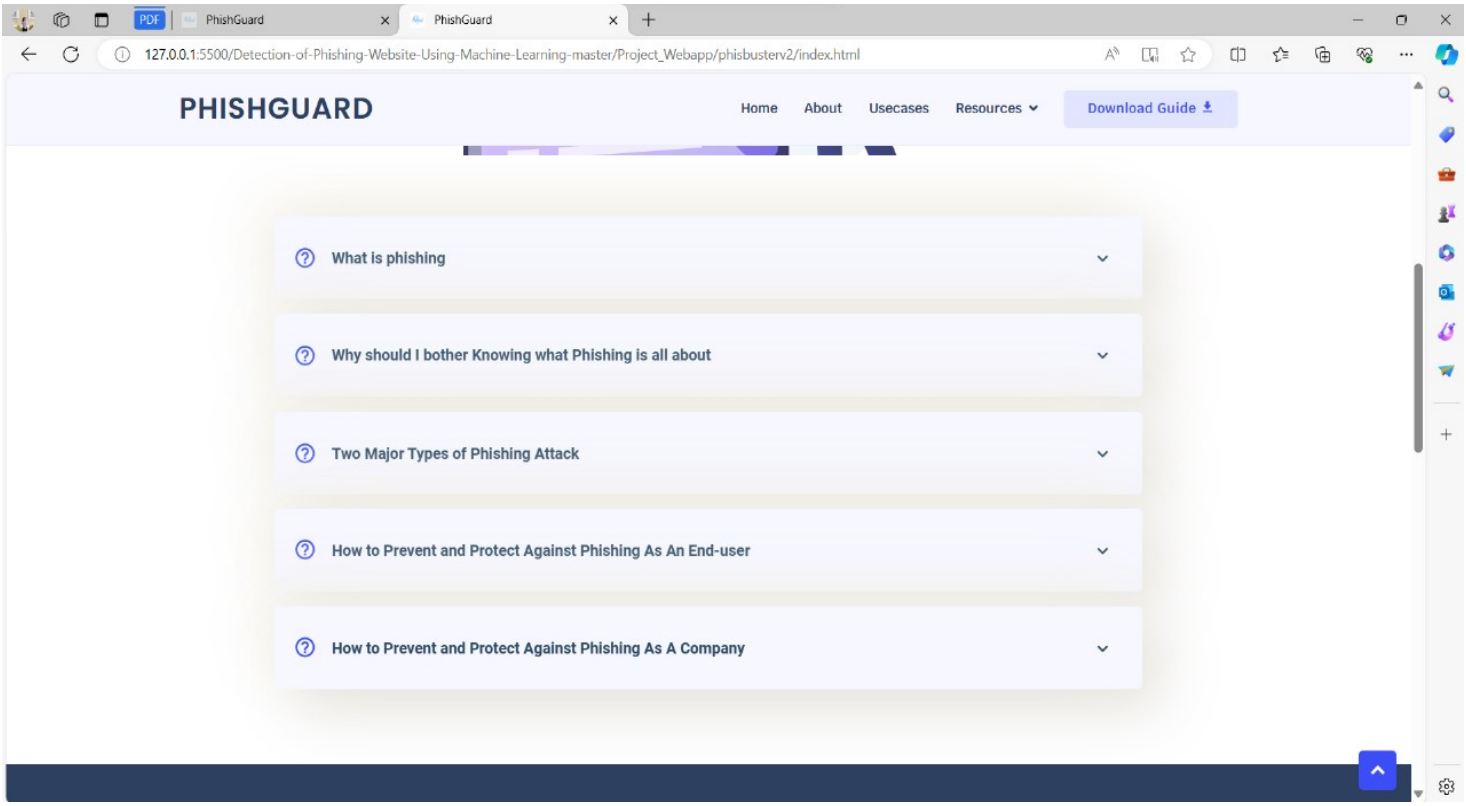
Split the dataset into training and testing sets to evaluate the model's performance.Train the selected machine learning model using the training data.Test the deployed model with unseen data to ensure its effectiveness in detecting phishing websites accurately and efficiently.

Step 6:
Implementation- We trained and tested CNN ,RF,Decision tree,KN Classifier,XGBoost,Logistic Regression Model.Once it is trained and tested it shows the accuracy of detection ,accuracy rates helps us compare both the model for detection of phishing website.

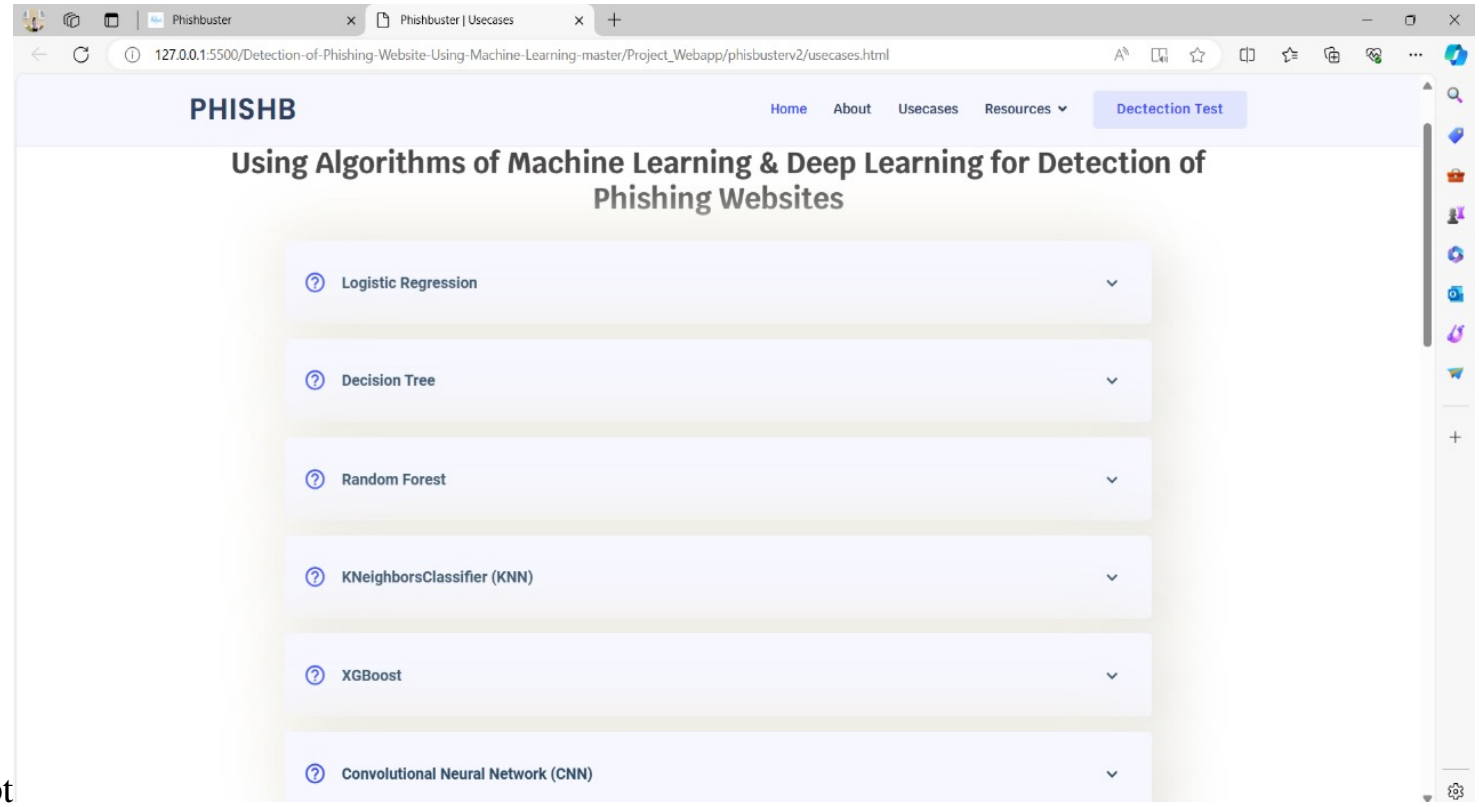
# CHAPTER 9: RESULT AND DISCUSSION




**LOGIN PAGE**

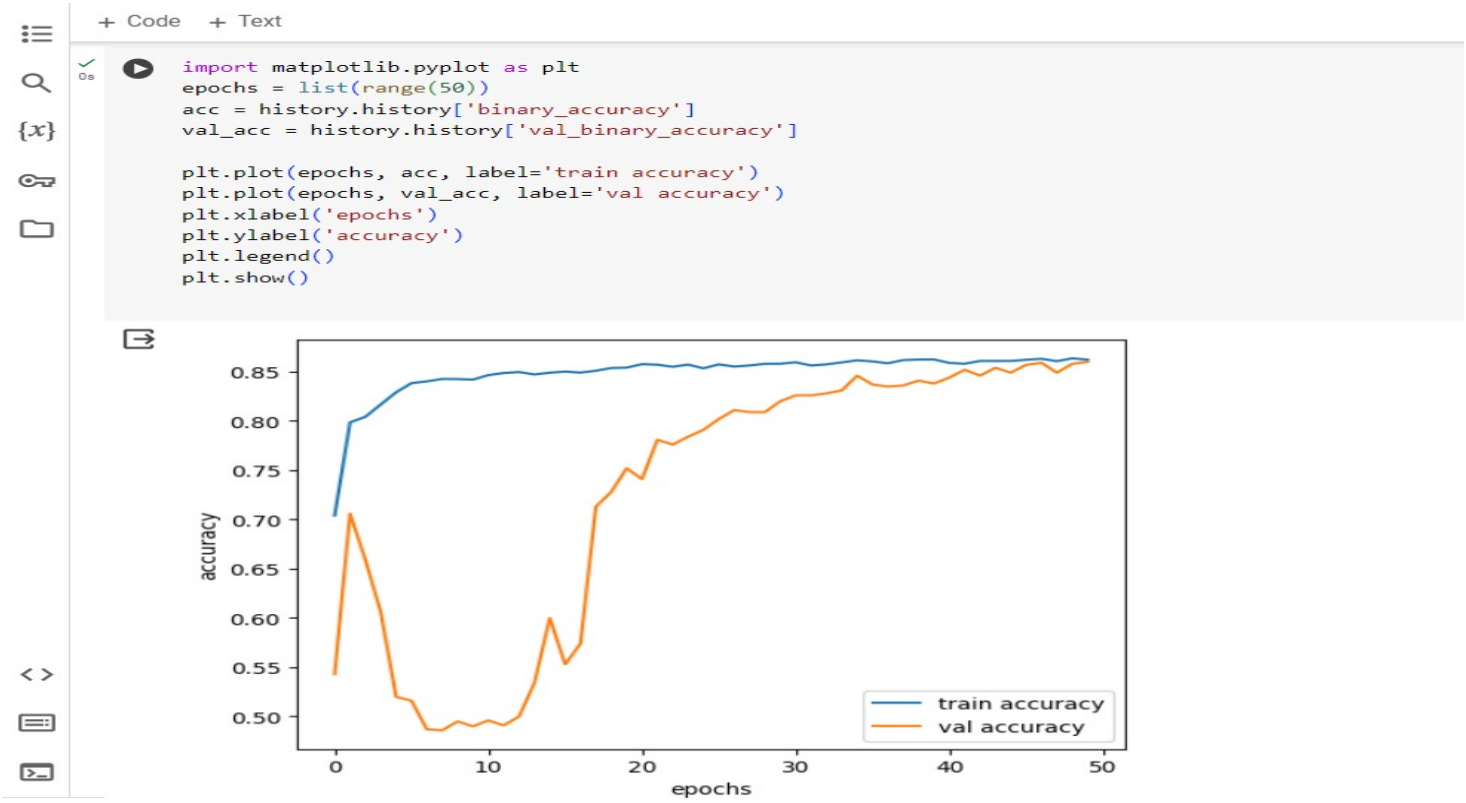



**ADMIN HOME PAGE**




ppt

## ACCURACY PAGE



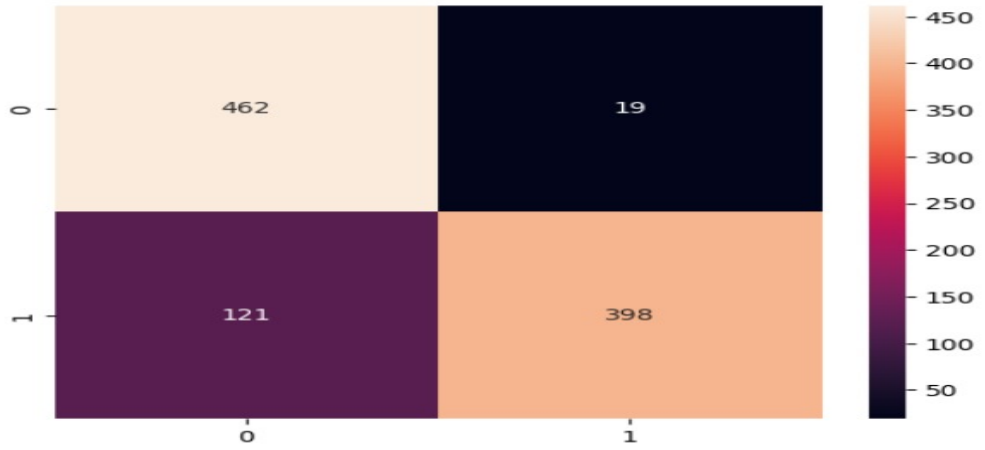

## CNN

```
+ Code + Text

[52] from sklearn.metrics import confusion_matrix
     from sklearn.metrics import accuracy_score
     from sklearn import metrics

     cm = confusion_matrix(y_test, ypred)
     sns.heatmap(cm, annot=True, fmt="d")

     <Axes: >
```




```
[54] accuracy_score(y_test, ypred)

     0.86
```

**Accuracy Rate:86.0**

```
+ Code + Text

Random Forest Classifier

[20] forest = RandomForestClassifier(max_depth = 5)
     forest.fit(X_train,y_train)
     y_test_forest = forest.predict(X_test)
     y_train_forest = forest.predict(X_train)
     model_score = model.score(X_test,y_test)

     #computing the accuracy of the model performance
     acc_train_forest = accuracy_score(y_train,y_train_forest)
     acc_test_forest = accuracy_score(y_test,y_test_forest)

     print("Random forest : Accuracy on the model : ", model_score)
     print("Random forest: Accuracy on training Data: {:.3f}".format(acc_train_forest))
     print("Random forest: Accuracy on test Data: {:.3f}".format(acc_test_forest))
     print(metrics.classification_report(y_test, y_predict))
     print(metrics.confusion_matrix(y_test, y_predict))

     Random forest : Accuracy on the model :  0.784
     Random forest: Accuracy on training Data: 0.822
     Random forest: Accuracy on test Data: 0.806
                   precision    recall  f1-score   support

                0       0.72      0.93      0.81       999
                1       0.90      0.64      0.75      1001

         accuracy                           0.78      2000
        macro avg       0.81      0.78      0.78      2000
     weighted avg       0.81      0.78      0.78      2000

     [[927  72]
      [360 641]]

[22] #storing the results. The below mentioned order of parameter passing is important.
     #Caution: Execute only once to avoid duplications.
     storeResults('Random forest', acc_train_forest, acc_test_forest)
```

**Random Forest**

+ Code + Text

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)

tree_score=model.score(X_test, y_test)
```

```
[18] #computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on the Model: ",tree_score)
print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
print(metrics.classification_report(y_test, y_predict))
print(metrics.confusion_matrix(y_test, y_predict))
```

```
Decision Tree: Accuracy on the Model:  0.784
Decision Tree: Accuracy on training Data: 0.817
Decision Tree: Accuracy on test Data: 0.800
              precision    recall  f1-score   support

           0       0.72      0.93      0.81       999
           1       0.90      0.64      0.75      1001

    accuracy                           0.78      2000
   macro avg       0.81      0.78      0.78      2000
weighted avg       0.81      0.78      0.78      2000

[[927  72]
 [360 641]]
```

## Decision Tree

+ Code + Text

XGBoost

```
[26] # instantiate the model
xgb = XGBClassifier(use_label_encoder =False,learning_rate=0.4,max_depth=7)
#fit the model
xgb.fit(X_train, y_train)
#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)
model_score=xgb.score(X_test, y_test)
```

```
#computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on the Model: ",model_score)
print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))
print(metrics.classification_report(y_test, y_predict))
print(metrics.confusion_matrix(y_test, y_predict))
```

```
XGBoost: Accuracy on the Model:  0.8365
XGBoost: Accuracy on training Data: 0.871
XGBoost : Accuracy on test Data: 0.837
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       999
           1       0.79      0.82      0.81      1001

    accuracy                           0.80      2000
   macro avg       0.80      0.80      0.80      2000
weighted avg       0.80      0.80      0.80      2000

[[785 214]
 [180 821]]
```

## XGBoost

+ Code + Text

KNeighbors Classifier

```
# instantiate the model
knn = KNeighborsClassifier(n_neighbors =1)
# fit the model
knn.fit(X_train,np.ravel(y_train,order='C'))
#predicting the target value from the model for the samples
y_predict= knn.predict(X_test)

#predicting the target value from the model for the samples
y_test_knn = knn.predict(X_test)
y_train_knn = knn.predict(X_train)
model_score=knn.score(X_test, y_test)
```

```
[24] #computing the accuracy of the model performance
acc_train_knn = accuracy_score(y_train,y_train_knn)
acc_test_knn = accuracy_score(y_test,y_test_knn)

print("KNeighborsClassifier: Accuracy on the Model: ",model_score)
print("KNeighborsClassifier: Accuracy on training Data: {:.3f}".format(acc_train_knn))
print("KNeighborsClassifier: Accuracy on test Data: {:.3f}".format(acc_test_knn))
print(metrics.classification_report(y_test, y_predict))
print(metrics.confusion_matrix(y_test, y_predict))
```

```
KNeighborsClassifier: Accuracy on the Model:  0.803
KNeighborsClassifier: Accuracy on training Data: 0.810
KNeighborsClassifier: Accuracy on test Data: 0.803
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       999
           1       0.79      0.82      0.81      1001

    accuracy                           0.80      2000
   macro avg       0.80      0.80      0.80      2000
weighted avg       0.80      0.80      0.80      2000

[[785 214]
 [180 821]]
```

## KNN

+ Code + Text

Logistic Regression

```
[14] import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# instantiate the model
model = LogisticRegression(max_iter=1000)
# fit the model
model.fit(X_train,np.ravel(y_train,order='C'))
#predicting the target value from the model for the samples
y_predict= model.predict(X_test)
y_train_model = model.predict(X_train)
y_test_model = model.predict(X_test)
model_score=model.score(X_test, y_test)
```

```
[15] #computing the accuracy of the model performance
acc_train_model = accuracy_score(y_train,y_train_model)
acc_test_model = accuracy_score(y_test,y_test_model)

print("LogisticRegression: Accuracy on the Model: ",model_score)
print("LogisticRegression: Accuracy on training Data: {:.3f}".format(acc_train_model))
print("LogisticRegression: Accuracy on test Data: {:.3f}".format(acc_test_model))
print(metrics.classification_report(y_test, y_predict))
print(metrics.confusion_matrix(y_test, y_predict))
```

```
LogisticRegression: Accuracy on the Model:  0.784
LogisticRegression: Accuracy on training Data: 0.804
LogisticRegression: Accuracy on test Data: 0.784
              precision    recall  f1-score   support

           0       0.72      0.93      0.81       999
           1       0.90      0.64      0.75      1001

    accuracy                           0.78      2000
   macro avg       0.81      0.78      0.78      2000
weighted avg       0.81      0.78      0.78      2000

[[927  72]
 [360 641]]
```

## Logistic Regression

+ Code + Text

Comparision of Models

```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

| | ML Model | Train Accuracy | Test Accuracy |
|---|---|---|---|
| 0 | LogisticRegression | 0.804 | 0.784 |
| 1 | Decision Tree | 0.816 | 0.800 |
| 2 | Random forest | 0.822 | 0.806 |
| 3 | KNeighborsClassifier | 0.810 | 0.803 |
| 4 | XGBoost | 0.871 | 0.836 |

Next steps: Generate code with results View recommended plots

```
[30] #Sorting the datafram on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

| | ML Model | Train Accuracy | Test Accuracy |
|---|---|---|---|
| 4 | XGBoost | 0.871 | 0.836 |
| 2 | Random forest | 0.822 | 0.806 |
| 3 | KNeighborsClassifier | 0.810 | 0.803 |
| 1 | Decision Tree | 0.816 | 0.800 |
| 0 | LogisticRegression | 0.804 | 0.784 |

Executing (4m 42s) <cell line:

**Comparison of Models**

# CHAPTER 10: FUTURE SCOPE

FUTURE SCOPE:

1.The future scope of the project includes exploring advancements in machine learning, particularly the integration of deep learning techniques like convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to enhance the accuracy and robustness of phishing detection models.

2.Additionally, there's potential for developing and deploying real-time detection systems capable of identifying and mitigating phishing attacks in real-time, offering immediate protection to users and organizations. These avenues of exploration aim to further fortify cybersecurity measures, ensuring a safer online experience for all users in the face of evolving cyber threats.

3.Integrate the phishing website detection system with existing cybersecurity ecosystems, such as web browsers, email clients, and network security solutions. Seamless integration can provide comprehensive protection across multiple attack vectors.

4.Foster collaboration with academia, industry partners, and cybersecurity communities to exchange knowledge, share datasets, and advance research in phishing website detection. Collaborative efforts can accelerate innovation and contribute to the development of robust and scalable solutions.

5.Continuously refine and optimize the machine learning models to enhance detection accuracy. This could involve experimenting with novel features, exploring ensemble learning techniques, or incorporating advanced deep learning architectures

# CHAPTER 11: CONCLUSION

After meticulously comparing various machine learning algorithms for the detection of phishing websites, it is evident that each algorithm possesses its unique strengths and weaknesses. Through rigorous experimentation and analysis, we have scrutinized the performance of algorithms such as Decision Trees ,XGBoost,KNClassifier,Random Forest,Logistic Regression,CNN.

Neural Networks exhibit remarkable flexibility and capability to learn intricate patterns from data. Deep learning architectures, in particular, offer state-of-the-art performance in various domains. However, their success heavily relies on extensive computational resources, substantial amounts of data, and meticulous hyperparameter tuning.Convolutional Neural Network exhibits highest accuracy hence it is selected to detect phishing website compared to others.

In conclusion, the selection of the most suitable algorithm for phishing website detection depends on several factors, including the specific characteristics of the dataset, computational resources, interpretability requirements, and desired performance metrics. A comprehensive understanding of the strengths and limitations of each algorithm is crucial for making informed decisions in real-world applications.

# CHAPTER 12: REFERENCES

1.S. Yadav, et al., ”A novel approach for phishing website detection using machine learning techniques,” in Procedia Computer Science, 2018.

2.R. Ramya, et al., ”Phishing website detection using machine learning and social network analysis,” in Materials Today: Proceedings, 2020.

3.K. Y. Wang, J. Beck, ”Automated Detection of Phishing Targeted at E-commerce Websites,” in IEEE Transactions on Dependable and Secure Computing, 2014.

4.F. Ahmed, et al., ”A novel hybrid model for phishing detection using random forest, decision tree, and multilayer perceptron,” in Security and Communication Networks, 2019.

5.H. Singh, A. Yadav, ”Phishing Detection Using Machine Learning Algorithms: A Review,” in Procedia Computer Science, 2018.

6.Comprehensive Study on Phishing Detection Using Machine Learning Techniques.Authors: D. Udhayakumar, S. Manogaran, Published in: Computers, Materials Continua, 2020.

7.Phishing Website Detection using Machine Learning Techniques Authors: Sachin Sharma, Parvinder S. Sandhu .Published in: Procedia Computer Science, 2018.