

PHISHING WEBSITE DETECTION USING MACHINE LEARNING

*This project report is submitted to
Rashtrasant Tukadoji Maharaj Nagpur University
in the partial fulfilment of the requirement for the award of the degree of*

Bachelor of Technology in Computer Technology

by

Ms. Vaishnavi Mehar (CT20057)

Ms. Yashu Patle (CT20083)

Mr. Ayaan Syed (CT20085)

Ms. Ketki Piprode (CTD20099)

Ms. Renuka Kotawar (CTD20103)

Under the guidance of

Dr. Vilas P. Mahatme

Professor



2023-2024

**DEPARTMENT OF COMPUTER TECHNOLOGY
KAVIKULGURU INSTITUTE OF TECHNOLOGY AND SCIENCE
RAMTEK, NAGPUR, MAHARASHTRA, INDIA-441106**

**KAVIKULGURU INSTITUTE OF TECHNOLOGY AND SCIENCE
RAMTEK, NAGPUR, MAHARASHTRA INDIA-441106**

DEPARTMENT OF COMUPTER TECHNOLOGY



CERTIFICATE

This is to certify that the project report entitled '**Phishing website detection using machine learning**' carried out by **Ms. Vaishnavi Mehar (CT20057)**, **Ms. Yashu Patle (CT20083)**, **Mr. Ayaan Syed (CT20085)**, **Ms. Ketki Piprode (CTD20099)**, **Ms. Renuka Kotawar (CTD20103)** of the B.Tech. Final year of computer technology, during the academic year 2023-2024, in the partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology (Computer Technology)** offered by the **Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur**.

Dr. Vilas P. Mahatme
Guide

Dr. Vilas P. Mahatme
Head of Department

Dr. Avinash N. Shrikhande
Principal

Date:

Place: Ramtek

ACKNOWLEDGEMENT

We are grateful to our respected guide **Dr. Vilas P. Mahatme**, Head of Department for his kind, disciplined and invaluable guidance which inspired us to solve all the difficulties that came across during completion of project.

Our sincere thanks are due to **Dr. Avinash N. Shrikhande**, Principal, for extending all the possible help and allowing us to use all resources that are available in the Institute.

We are also thankful to our **Parent** and **Friends** for their valuable corporation and standing with us in all difficult conditions.

Project-mates

ABSTRACT

Phishing websites are a serious threat to online security, as they attempt to steal sensitive information from unsuspecting users. To combat this threat, researchers have developed various techniques for detecting phishing websites, including machine learning algorithms. Machine learning algorithms can be trained on large datasets of phishing and legitimate websites to learn patterns and characteristics that distinguish between the two. These algorithms are used to identify and block phishing websites before users can be victimized. One approach to phishing website detection using machine learning involves feature extraction, where various features of a website such as URL structure, domain age, and content are analyzed to identify phishing websites. Another approach involves using deep learning algorithms to automatically extract features and learn complex patterns in website data. With the help of this algorithms and approaches it can detect a website as legitimate or phishing website. Overall, machine learning-based phishing website detection techniques have shown promising results, achieving high accuracy rates and outperforming traditional rule-based methods. With further research and development, these techniques have the potential to become an important tool in the fight against online phishing attacks. Phishing attacks remain a significant threat to cybersecurity, with attackers using deceptive websites to steal sensitive information from unsuspecting users. It consists a comprehensive dataset of legitimate and phishing websites, extract relevant features, and employ machine learning algorithms for classification. This model demonstrates high accuracy, precision, and recall in identifying phishing websites, making it a valuable tool for enhancing online security. By combining feature engineering and machine learning, this research contributes to the ongoing efforts to mitigate the risks associated with phishing attacks and protect users from online threats.

KEYWORDS: Machine learning techniques, Xgboost, Gradient boosting, Adaboost, SVM, Random Forest, evaluation

CONTENT

<i>Acknowledgement</i>	<i>i</i>
<i>Abstract</i>	<i>ii</i>
<i>Content</i>	<i>iii</i>
<i>Abbreviations</i>	<i>vi</i>
<i>List of Figures</i>	<i>vii</i>
CHAPTER 1 INTRODUCTION	1-3
1.1 Preamble	1
1.2 Motivation	2
1.3 Aim	2
1.4 Objectives	2
1.5 Organization of Report	3
CHAPTER 2 PRIOR ART	4-6
2.1 URL phishing Detection System Using Hybrid Approach	4
2.2 A System and A Method for Predicting and Preventing Phishing	5
2.3 Advanced Phishing Detection Mechanism	5
2.4 A System and Method for Training a User Against Phishing Attacks	5
2.5 Phishing Website Detection Method and System Based on Self-Adaptive Heterogenous Multi-Classification Model	6
CHAPTER 3 LITERATURE REVIEW	7-11
3.1 Detecting Phishing Attacks Using NLP and Machine Learning	7
3.2 Phishing web sites features classification based on extreme learning machine	8

3.3	Detecting Phishing Attacks Using NLP and ML	8
3.4	Performance comparison of classifiers on reduced phishing website database	8
3.5	Classification of URL bitstreams using bag of bytes	9
3.6	A Novel ML Approach to Detect Phishing Website	9
3.7	A New Method for Detection of Phishing Website: URL Detection	9
3.8	Phishing Site Detection Based on C4.5 Decision Tree Algorithm	10
3.9	Boosting the Phishing Detection Performance by Semantic Analysis	10
3.10	Phishing Website Prediction Using Classification Techniques	11
CHAPTER 4	TOOLS AND TECHNOLOGIES	12-40
4.1	Python	12
4.2	HTML	20
4.3	XAMPP	22
4.4	MySQL	23
4.6	Algorithms	25
4.6.1	Random Forest Classifier	25
4.6.2	Support Vector Machine	29
4.6.3	XGBoost Algorithm	31
4.6.4	AdaBoost Algorithm	36
4.6.5	Gradient Boosting	38
CHAPTER 5	PROPOSED APPROACH AND SYSTEM ARCHITECTURE	41-49
5.1	Block Diagram	42
5.2	System Architecture	42
5.3	UML Diagram	44
5.4	Class Diagram	46

5.5 Sequence Diagram	46
5.6 Activity Diagram	47
5.7 E-R Diagram	47
5.8 DFD Diagram	48
CHAPTER 6 IMPLEMENTATION	50-52
6.1 Data Collection	50
6.2 Data Gathering and Preprocessing	50
6.3 Model Training	51
6.4 Testing and Evaluating the Model	51
6.5 Working for Phishing Website Detection	52
CHAPTER 7 RESULTS AND DISCUSSION	53-58
7.1 Home Page	53
7.2 Load Page	53
7.3 View Page	54
7.4 URL Input	55
7.5 Prediction	55
7.6 Accuracy Graph	57
7.7 Recall Graph	58
CHAPTER 8 CONCLUSION	59-61
8.1 Limitation of the Study	59
8.2 Future Scope of Work	61
<i>References</i>	62-63

ABBREVIATIONS

Abbreviations	Description
URL	Uniform Resource Locator
SVM	Support Vector Machine
SSL	Secure Socket Layer
HTML	Hyper Text Markup Language
MYSQL	My Structured Query Language
NLP	Natural Language Processing
IP	Internet Protocol
CNN	Convolutional Neural Network
DM	Data Mining
NUMPY	Numerical Python
IDE	Integrated Development Environment
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
XML	Extensible Markup Language
DFD	Data Flow Diagram
E-R	Entity Relationship
UML	Unified Modelling Language
ML	Machine Learning
K-NN	K-Nearest Neighbor
SSD	Solid State Drive
AI	Artificial Intelligence
PWD	Public Work Department
XAMPP	X-operating system, Apache, MySQL, PHP, Perl
ID	Identity Document
XGBOOST	Extreme Gradient Boosting
ADABOOST	Adaptive Boosting
GDBT	Gradient-Boosted Decision Trees

LIST OF FIGURES

Figure	Title	Page
4.1	Client Server Model	23
4.2	Three types of Nodes in a Decision Tree	27
4.3	Decision Tree Diagram for Features of the Phone	29
4.4	Categories by Hyperplane	30
4.5	Hyper Planes in 2D and 3D Feature Space	30
4.6	Diagram for XGBoost	32
4.7	Decision Tree	33
4.8	Random Forest	35
4.9	Random Forest Algorithm	36
4.10	Diagram for AdaBoost Algorithm	37
4.11	Gradient Boosted Trees for Regression	40
5.1	Block Diagram	42
5.2	System Architecture	42
5.3	UML Diagram	45
5.4	Class Diagram	46
5.5	Sequence Diagram	46
5.6	Activity Diagram	47
5.7	E-R Diagram	48
5.8	DFD Diagram	48
7.1	Home page	53
7.2	Load Page	54
7.3	View Page	54
7.4	URL Input	55
7.5	Prediction	55
7.6	Accuracy Graph	57
7.7	Recall Graph	58

CHAPTER 1

INTRODUCTION

1.1 Preamble

Phishing website detection is the process of identifying and flagging websites that attempt to impersonate legitimate websites with the goal of stealing sensitive information such as login credentials, credit card numbers, and personal identification information. Phishing attacks have become increasingly sophisticated over the years, and attackers often use tactics such as social engineering and fake login screens to trick users into giving up their sensitive information. Phishing websites may also use URL spoofing to make it appear that the user is on a legitimate. Phishing attacks are a common type of cybercrime that involves the use of fraudulent emails, messages, or websites to trick users into revealing sensitive information. One of the most effective ways to combat phishing attacks is through the detection and blocking of phishing websites. Phishing website detection involves the use of various techniques and technologies to identify and flag websites that are designed to deceive users. One common technique used by attackers is to create websites that closely mimic the appearance of legitimate sites, such as banks or e-commerce sites. These phishing sites are often hosted on compromised servers or using domain names that are similar to the real sites.

One approach to detecting phishing websites is to use machine learning algorithms that can analyze website content, metadata, and other features to identify potential phishing sites. These algorithms can be trained on large datasets of known phishing websites to identify common patterns and characteristics. Some machine learning models may also incorporate real-time data feeds to identify and flag new phishing sites as they are created. Another approach to phishing website detection is to use reputation-based systems that maintain lists of known malicious websites. These systems can use various sources of information, such as blacklists, user reports, and threat intelligence feeds to identify and block phishing sites. Some web browsers also use reputation-based systems to warn users when they attempt to access a known phishing site. Phishing website detection may also involve the use of behavioral analysis techniques that can detect

unusual or suspicious activity on a website. For example, these techniques may look for patterns of user behavior that differ from normal usage, such as a sudden increase in requests for login credentials or an unusual number of redirects to other sites. Overall, the detection and blocking of phishing websites is an essential component of any effective cyber-security strategy. By using a combination of machine learning algorithms, reputation-based systems, and behavioral analysis techniques, organizations can protect their users and prevent sensitive information from falling into the hands of attackers. Therefore, it is crucial to remain vigilant and stay up-to-date with the latest phishing threat trends and detection methods.

1.2 Motivation

The motivation behind this project is to summarize Phishing attacks. As Phishing attacks are on the rise, with cybercriminals continuously refining their techniques to deceive users and compromise their sensitive information. This aims to enhance online security by harnessing the power of machine learning to identify and prevent phishing attacks. Phishing website detection involves analyzing various indicators, such as URL structure, SSL certificates, website content, and user behavior, to identify suspicious or fraudulent websites designed to steal sensitive information.

1.3 Aim

To develop a machine learning model for the detection of phishing websites and to efficiently identify websites that are designed to steal sensitive information from users by analyzing website content, user behavior and other relevant features.

1.4 Objectives

- To study Machine Learning Algorithm.
- To study Machine Learning Library.
- To study Python Library.
- To study Data Library.
- To gather a diverse and representative dataset of both legitimate and phishing websites for training and testing.

- To identify relevant features or attributes from website content, structure, and metadata. These features can include URLs, domain age, SSL certificates, content analysis, and more.
- To train the selected machine learning models on the labelled dataset, optimizing hyper parameters and model performance.
- To analyze and reduce false positives by fine-tuning the model or improving feature selection.
- To investigate a specific problem of whether it is valuable or not to use machine learning techniques to predict the type of website.

1.5 Organization of Report

Chapter one contains introduction of Phishing Website Detection Using Machine Learning which is used to detect whether a website is legitimate or phishing website.

Chapter two includes prior art which refers to all publicly available information and knowledge that existed before a particular invention, often used in patent examinations to determine novelty and patentability.

Chapter three contain literature review of project Phishing Website Detection Using Machine Learning. It is a recap of the important information of the source.

Chapter four explain tools and technologies which are used in project Phishing Website Detection Using Machine Learning. Tools and technologies used in this are: PYTHON, HTML, MYSQL, XAMPP server.

Chapter five refers proposed approach and system architecture. This system will provide an easy approach to detect phishing websites. Multiple users can access simultaneously on the web application. It also contains system architecture.

Chapter six explains implementation. It shows the plan for implementing project.

Chapter seven describes result and discussion. It will show the implemented work related to project Phishing website detection using machine learning.

Chapter eight contains conclusion and future scope of the project. This project has given the opportunity to work on a new platform and learn completely new technology.

CHAPTER 2

PRIOR ART

Prior art in patent law includes publicly available information like patents, literature, products and prior inventions. It is crucial for evaluating the novelty and non-obviousness of new inventions, acting as a vital reference for determining patent eligibility and safeguarding intellectual property.

2.1 Title: URL phishing detection system using hybrid approach

Patent No: 202341058195

Social Engineering threats pose a significant threat to individuals, organizations, and society as a whole. Detecting and mitigating such attacks is a challenging task due to their deceptive nature. In recent years, machine learning techniques have emerged as promising solutions for threat detection. This research focuses on leveraging linear regression and naive Bayes algorithms for detecting general social engineering attacks. A comparative analysis of these algorithms is performed to evaluate their effectiveness in identifying and mitigating social engineering threats. The study begins by outlining the characteristics and challenges associated with social engineering attacks. Various ML techniques have been proposed in the literature, but this research concentrates on linear regression and naive Bayes due to their simplicity, interpretability, and effectiveness in classification tasks. To evaluate the performance of linear regression and naive Bayes, a comprehensive dataset of real-world social engineering attacks is collected and labelled. The dataset consists of both legitimate and attack instances, and the features are extracted using NLP techniques, sentiment analysis, and psychological analysis. The performance metrics, such as accuracy, precision, recall, and F1-score, are calculated for both algorithms. The results of the comparative analysis indicate that both logistic regression and naive Bayes show promising results in detecting social engineering threats. However, they exhibit different strengths and weaknesses. Logistic regression performs well in scenarios where feature independence is not a significant factor, while naive Bayes excels when features are assumed to be independent.

2.2 Title: A System and A Method for Predicting and Preventing Phishing

Patent No: 202311055390

The present disclosure relates to security system for predicting and preventing phishing and method, that includes a processor and memory. The system collect data and analyze the collected data using machine learning algorithms to detect anomalous behavior and potential security threats. The system also monitors computing devices in real-time and detect security threats based on the analyzed data and respond to detected security threats by initiating appropriate action. Based on the analyzed data the system may block emails, warn a user, quarantine affected computing devices and alerting administrator.

2.3 Title: Advanced Phishing Detection Mechanism

Patent No: 202311054089

The invention is a system designed to detect and respond to phishing attacks in login sequences. It incorporates multiple components that work together to enhance security and protect users from potential threats. The system includes a Login Failure Detector on user devices that continuously monitors login failure sequences and identifies abnormal patterns indicative of phishing attacks. These sequences are further analysed using a Machine Learning algorithm trained on known phishing attack patterns. When a phishing attack is detected, the system's Response Module takes appropriate actions such as blocking suspicious IP addresses, alerting users, and reporting incidents to authorities. The system integrates with secure storage systems to store and encrypt user credentials, ensuring the protection of sensitive information. By leveraging real-time monitoring, advanced analysis techniques, and proactive measures, the invention provides an effective solution to address the rising challenge of phishing attacks, helping individuals and organizations safeguard their sensitive information and prevent unauthorized access.

2.4 Title: A System and Method for Training a User Against Phishing Attacks

Patent No: 202311053441

Embodiments of the present disclosure relates to a system and method for training a user against phishing attacks by providing a simulated phishing environment. The system comprises a processor coupled to a memory. The memory stores processor-executable instructions. The processor is configured to simulate a phishing attack environment.

Further, the processor is configured to train the user to respond to phishing attacks in the simulated environment. In the end, the processor is configured to generate a report for the user based on the training.

2.5 Title: Phishing website detection method and system based on self-adaptive heterogeneous multi-classification model

Patent No: CN108965245B

The invention provides a phishing website detection method and system based on a self-adaptive heterogeneous multi-classification model. The method comprises the steps of constructing a self-adaptive heterogeneous multi-classification model by linear addition on multiple base classification algorithms, training the multi-classification model, wherein the input of the model is the input of each base classification algorithm, the output of the model is a sample label, and each base classification algorithm extracts corresponding characteristics from a sample record as input; and solving the model parameters by adopting a machine learning algorithm, and testing and optimizing by using the test set to finally obtain the detection model of the phishing website. The system comprises a domain name morpheme feature classifier, a topic index feature classifier, a content similarity feature classifier, a structure style feature classifier, a visual rule feature classifier, a linear addition training module, an integrated classifier, a training data set management module and a detection and alarm module. The invention realizes real-time detection of the phishing website and improves the accuracy and stability of the phishing website detection.

CHAPTER 3

LITERATURE REVIEW

3.1 Detecting Phishing attacks using Natural Language Processing and Machine Learning

Padmanaban A (2023) proposed a model for detection of phishing website detection. Machine learning is a field of artificial intelligence that allows computers to improve their performance on specific tasks by learning from data instead of just following predefined rules. It encompasses supervised, unsupervised, and semi-supervised learning, which depend on the availability of labelled data. Phishing detection employs machine learning to extract features, classify URLs, adapt to new threats, and analyze data to detect clusters of phishing attack. It plays a crucial role in automating the detection of phishing attacks. The proliferation of malicious websites and internet criminal activities have raised concerns among web users and service providers. To address this issue, we propose a learning-based algorithms such as CAT boost, Adaboost, Random Forest and Support vector machine to classify websites based on the URLs into three categories: benign, spam, and malicious. Benign websites offer legitimate services and are safe to use, while spam websites inundate users with ads, fake surveys, or dating sites. Malicious websites are created by attackers with the intent of disrupting computer operations, stealing confidential data, or gaining unauthorized access to private systems. The proposed mechanism analyses only the URL of websites and does not access their content, reducing runtime latency and eliminating the possibility of exposing users to browser-based vulnerabilities. A large dataset of labelled URLs is used to train a classification model, which is then used to classify new URLs. After experimentally evaluating the proposed approach using a publicly available dataset, it is demonstrated that the approach achieves 98.3% accuracy, with the random forest model and SVM model, outperforming traditional blacklisting services in generality and coverage, and having the ability to adapt to new threats and enhance its performance over time. It presenting a promising solution for accurately detecting phishing attacks in URLs and of interest to researchers and practitioners working in cybersecurity.

3.2 Phishing web sites features classification based on extreme learning machine

Y. Sönmez, T. Tuncer (2018) presented a model for classification of phishing website features. Phishing is a common attack on credulous people by making them to disclose their unique information using counterfeit websites. The objective of phishing website URLs is to purloin the personal information like user name, passwords and online banking transactions. Phishers use the websites which are visually and semantically similar to those real websites. As technology continues to grow, phishing techniques started to progress rapidly and this needs to be prevented by using anti-phishing mechanisms to detect phishing. Machine learning is a powerful tool used to strive against phishing attacks. This paper surveys the features used for detection and detection techniques using machine learning.

3.3 Detecting Phishing Attacks Using NLP and Machine Learning

T. Peng, I. Harris (2018) proposed a model for detecting phishing attacks with the help of natural language processing. Phishing attacks are one of the most common and least defended security threats today. An approach is presented which uses natural language processing techniques to analyze text and detect inappropriate statements which are indicative of phishing attacks. Our approach is novel compared to previous work because it focuses on the natural language text contained in the attack, performing semantic analysis of the text to detect malicious intent. To demonstrate the effectiveness of our approach, it is evaluated it using a large benchmark set of phishing emails.

3.4 Performance comparison of classifiers on reduced phishing website dataset

M. Karabatak and T. Mustafa (2018) studied that numerous enemies of phishing frameworks are being created to recognize phishing substance in online correspondence frameworks. In spite of the accessibility of hordes hostile to phishing frameworks, phishing proceeds with unabated because of lacking recognition of a zero-day assault, pointless computational overhead and high bogus rates. In spite of the fact that Machine Learning approaches have accomplished promising exactness rate, the decision and the exhibition of the component vector limit their successful location. Phishing is a typical assault on guileless individuals by making them to unveil their one-of-a-kind data utilizing fake sites. In this work, an upgraded AI based prescient model is proposed to improve the effectiveness of against phishing plans. The prescient model comprises of Feature Selection Module which is utilized for the development of a successful element

vector. These highlights are removed from the URL, website page properties and site page conduct utilizing the gradual segment-based framework to introduce the resultant component vector to the prescient model. The proposed framework utilizes CNN, KNN and SVM which have been prepared on a 30-dimensional list of capabilities. AI is an incredible asset used to endeavor against phishing assaults.

3.5 Classification of URL bitstreams using bag of bytes

K. Shima (2018) studied that websites are main responsible for the rapid growth of criminal activities in the internet and corresponding activities which results in the many illegal things. So, there are many preventive steps to be taken to stop these kinds of activities. Here a model is proposed which will classify the given URL into any of the three possible classes, i.e., Benign, spam and malware. This model will detect the classification of the URL without using any websites content.

3.6 A Novel Machine Learning Approach to Detect Phishing Websites

J. Shad and S. Sharma (2018) proposed many detection techniques using URL, Hyperlinks features that can be used to differentiate between the defective and non-defective website. In the last few years, many fake websites have developed on the World Wide Web to harm users by stealing their confidential information such as account ID, user name, password, etc. Phishing is the social engineering attacks and currently attacks on mobile devices. That might result in the form of financial losses. There are six main approaches such as heuristic, blacklist, fuzzy rule, machine learning, image processing, and CANTINA based approach. It delivers a good consideration of the phishing issue, a present machine learning solution, and future study about Phishing threats by using machine learning Approach.

3.7 A New Method for Detection of Phishing Websites: URL Detection

S. Parekh (2018) proposed a model as a solution to detect phishing websites by using the URL detection method using Random Forest algorithm. Phishing is an unlawful activity wherein people are misled into the wrong sites by using various fraudulent methods. The aim of these phishing websites is to confiscate personal information or other financial details for personal benefits or misuse. As technology advances, the phishing approaches used need to get progressed and there is a dire need for better security and better mechanisms to prevent as well as detect these phishing approaches. There are three major phases such as Parsing, Heuristic Classification of data, and

Performance Analysis in this model and each phase makes use of a different technique or algorithm for processing of data to give better results.

3.8 Phishing Sites Detection Based on C4.5 Decision Tree Algorithm

L. MacHado and J. Gadge (2017) presented a model based on C4.5 decision tree algorithm to detect phishing websites. Among various anti-phishing solutions, Machine Learning techniques are considered to be promising. The purpose of this study was to evaluate the effect of C4.5 decision tree algorithm on phishing website detection. In the experiment, C4.5 had learned two models using two phishing website datasets respectively, which were PWD and PWD2, the latter was obtained through dimensionality reduction on the former. Under 10-fold cross-validation, various metrics indicated that the two models all done very well. The results of the corrected paired t- test under the significance of 0.05 (two tailed), shown that accuracy and recall metrics of the model based on PWD is not statistically significantly different to those of PWD2. According to the better model, the one based on PWD2, which had the lower complexity and the similar performance compared with the one based on PWD, the top 5 key features for classification were obtained.

3.9 Boosting the phishing Detection Performance by Semantic Analysis

X. Zhang, Y. Zeng (2017) proposed a method provides a promising way for phishing detection in actual Internet environment, which boosts the phishing detection performance effectively. Phishing is increasingly severe in recent years, which seriously threatens the privacy and property security of netizens. Phishing is essentially a counterfeiting of brands. In order to effectively cheat the victim, phishing sites are visually and semantically highly similar to real sites. In recent years, anti-phishing methods based on machine learning are mainstream anti-phishing methods. The effectiveness of the machine learning models hinges on the extracted statistical features. However, the extracted statistical features mainly focus on visual similarity, stealing information and third-party services, which ignore the semantic information of web pages. Therefore, extracted a series of semantic features through word2vec to better describe the features of phishing sites, and further fuse them with other multi-scale statistical features to construct a more robust phishing detection model. The experimental results on the actual data sets show that the majority of phishing websites are effectively identified by only mining the semantic features of word embedding's.

The phishing detection models based on fusion features obtained the best detection results, which shows that semantic features and other statistical features have good complementarity.

3.10 Phishing Websites prediction using classification techniques

Dyana Rashid Ibrahim (2017) presented a model for prediction of phishing website with the help of classification techniques. Phishing is an important issue that faces the cyber security. This paper exploits the capabilities of classification techniques on PWP, and introduces a methodology to protect users from the attackers. The blacklist procedure isn't a strong enough way to stay safe from the cybercriminals. Therefore, phishing website indicators have to be considered for this purpose, with the existence and usage of machine learning algorithms. Five different classification techniques have been used to evaluate their efficiency on PWP in terms of accuracy and the relative absolute error value for each one of them, with and without the feature selection process. WEKA tool was used for the implementation of these classifiers on a public dataset from NASA repository. The motivation behind this investigation is to employ a number of DM algorithms for the prediction purpose of phishing websites and compare their effectiveness in terms of accuracy and RAE. Where DM classifiers have proved their goodness in this kind of problems.

CHAPTER 4

TOOLS AND TECHNOLOGIES

This chapter describes various technologies those are being used in the development of the proposed system. The function and modules are explained along with features and components.

4.1 PYTHON

Python is a high-level, general-purpose, and interpreted programming language used in various sectors including machine learning, artificial intelligence, data analysis, web development, and many more. Python is known for its ease of use, powerful standard library, and dynamic semantics. It also has a large community of developers who keep on contributing towards its growth. The major focus behind creating it is making it easier for developers to read and understand, also reducing the lines of code.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Libraries

NumPy

NumPy is a general-purpose array-processing library that is one of Python's most essential libraries. It provides high-performance multidimensional array objects as well as array-related utilities. NumPy is a useful library with a wealth of functionality for dealing with arrays of numerical data. In addition, the implementation offers memory and execution efficiency that often comes close to compiled code, as well as serving as an interchange format for many existing libraries. NumPy is an efficient multidimensional data container. The basic object of NumPy is the homogeneous multidimensional array. It is a table of the same datatype elements or numbers, indexed by a tuple of positive integers. Dimensions are referred to as axes in NumPy and the number of axes is referred to as rank. The array class in NumPy is named ND array aka array. NumPy is used to manipulate arrays that contain values of the same datatype. NumPy simplifies array math operations and vectorization. This considerably improves performance and shortens the execution time. A primitive data type just means that the data is stored directly as bytes. So for example, an image is usually composed of pixels, each with red, green and blue values between 0 and 255. In NumPy, each pixel is stored as three bytes, one per color. The bytes of all the pixels are stored sequentially in memory. This is very compact and quick to access.

The NumPy library also includes lots of functions for manipulating ndarray objects. These make it very convenient to process arrays, but they are also very fast. They are written in C, a language that is very fast at processing arrays of primitive data types. The fact that the data exists in memory as an array of primitive data types also means that the data can be easily be exchanged with other libraries that might be written in Python, C, or just about any other language there is.

NumPy's versatility is evident in its support for random number generation, critical for simulations and statistical analyses. The library's capacity for reshaping and restructuring data, coupled with broadcasting capabilities, simplifies complex operations on arrays with varying dimensions. Noteworthy is NumPy's commitment to memory efficiency, enabling the handling of extensive datasets without compromising performance.

Facilitating seamless interaction with external data sources, NumPy provides functions for file input/output, supporting both binary and text formats. In the realm of signal processing, NumPy proves invaluable, offering tools for fundamental operations like filtering, convolution, and Fourier analysis.

Uses of NumPy

- Basic array operations include adding, multiplying, slicing, flattening, reshaping and indexing array.
- Advanced array operations: stack arrays, sectioned arrays and broadcast arrays.
- NumPy arrays are more memory-efficient than Python lists, especially for large datasets.
- Use Date Time or Linear Algebra.
- NumPy has functions for generating random numbers, essential for simulations and statistical sampling.
- NumPy Python Basic Slicing and Advanced Indexing.
- Useful for basic signal processing tasks like filtering, convolution, and Fourier analysis.
- It a powerful tool for numerical computing.
- NumPy supports essential linear algebra operations like matrix multiplication and solving linear system.

Advantages of NumPy

- NumPy is very useful for performing logical and mathematical calculations on arrays and matrices. This tool performs these operations much faster and more efficiently than Python lists.
- NumPy uses less memory and storage space, which is the main advantage. In addition, NumPy offers better performance in terms of execution speed. However, it is easier and more convenient to use.
- It is an open-source tool that can be used completely free of charge. It is based on Python which is an extremely popular programming language with many high- quality libraries for any task. Finally, it is very easy to connect existing C code to the Python interpreter.

- NumPy supports slicing, similar to Python lists. For example: `data5 = data1[1::2]`
[2, 6, 10]
- NumPy supports many array level operations. These include: Joining and splitting arrays. For example, a 3D array representing a color image (width by height by 3 channels for RGB) can be split into 3 separate greyscale images (width by height by 1). Or the 3 images can be joined. Arrays can be sorted and filtered. Reducing operators can be applied - for example finding the smallest value, or the sum of all the values.

Pandas

Pandas, a robust open-source library for Python, is an essential tool for data manipulation and analysis, serving data scientists, analysts and engineers. At its core are two fundamental data structures: Series, a labelled one-dimensional array and Data Frame, a versatile two-dimensional tabular structure. Pandas excels in data input/output, efficiently handling various file formats and enabling smooth data interchange. Its features simplify data cleaning and preprocessing, including operations for managing missing data, filtering, sorting, merging, reshaping. Intuitive indexing and selection methods streamline data access. The library facilitates aggregation and grouping, supports time series data management, includes statistical analysis functions. Although Pandas does not possess native data visualization capabilities, it integrates seamlessly with visualization libraries. Built on NumPy, it is highly interoperable and adapts to both structured and unstructured data, offering flexibility. With the ability to handle large datasets in chunks, Pandas empowers efficient data analysis, making it a pivotal component in data analysis, data science and data engineering endeavors.

Uses of Pandas

- Pandas streamlines data cleaning, handling missing values and data transformation tasks.
- It simplifies data exploration and analysis through aggregation and statistical operations.
- Pandas is vital for data preprocessing tasks in machine learning, such as feature engineering and data scaling.
- It efficiently imports data from various sources and exports it in different formats, aiding data ingestion and extraction.

- Pandas is instrumental in time series data analysis, offering powerful tools for handling and analyzing temporal data, including resampling and date/time manipulation.

Advantages of Pandas

- **Data Structures:** Pandas provides two main data structures: Series and Data Frame. These structures are highly flexible and allow for easy manipulation and analysis of labeled and structured data.
- **Handling Missing Data:** Pandas provides robust tools for handling missing data, making it easier to work with datasets that may have incomplete or inconsistent information. Methods like ``dropna()``, ``fillna()``, and ``interpolate()`` help manage missing values effectively.
- **Data Alignment:** Pandas automatically aligns data based on label indexes, enabling seamless operations on datasets even when they are differently indexed or partially overlapping.
- **Powerful Operations:** Pandas supports a wide range of operations for data manipulation, including slicing, indexing, merging, concatenating, reshaping, and pivoting. It allows for efficient data transformations and calculations.
- **Integration with Other Libraries:** Pandas integrates well with other Python libraries commonly used in data analysis, such as NumPy, Matplotlib, SciPy, and scikit-learn. This interoperability makes it easy to incorporate pandas into existing workflows.
- **Data I/O:** Pandas provides convenient methods for reading and writing data from and to various file formats, including CSV, Excel, SQL databases, JSON, HTML, and more. This simplifies the process of loading and saving data.
- **Time Series Analysis:** Pandas offers robust support for time series data, including date/time indexing, resampling, shifting, and rolling window operations. It makes analyzing time series data straightforward and efficient.
- **Flexible Indexing:** Pandas allows for flexible indexing of data, including hierarchical indexing, multi-level indexing, and custom indexing. This enables complex data organization and retrieval.
- **Performance:** Pandas is built on top of NumPy, a high-performance numerical

computing library. It leverages the efficiency of NumPy arrays for data storage and computation, resulting in fast and efficient data processing.

- **Documentation and Community:** Pandas has extensive documentation and a large community of users and developers. This makes it easy to find resources, tutorials, and solutions to common problems, as well as to seek help and contribute to the development of the library.
- Overall, pandas is a powerful and versatile library that simplifies data manipulation and analysis tasks in Python, making it an essential tool for data scientists, analysts, and researchers.

PyCharm

PyCharm, developed by JetBrains, stands as a comprehensive and highly acclaimed integrated development environment (IDE) tailored explicitly for Python development. This IDE has garnered immense popularity within the Python programming community due to its wide-ranging features and tools. It offers a powerful code editor with real-time error highlighting, code completion and inspection, facilitating error-free and efficient coding. The intelligent code completion feature suggests code based on context, enhancing coding efficiency. PyCharm includes a robust integrated debugger, unit testing support with compatibility for frameworks like Py test and unit test and seamless integration with version control systems like Git. Its project navigation tools, file structure views and a navigation bar make code exploration and organization straightforward. Developers can perform code refactoring with ease, maintain code quality through inspections and work within isolated virtual environments.

PyCharm, developed by JetBrains, stands out as a powerful integrated development environment (IDE) tailored specifically for Python developers. Offering a rich array of features, PyCharm aims to streamline the coding process and enhance productivity. The IDE excels in providing intelligent code assistance, including features like code completion, error checking, and suggestions, contributing to improved code quality. It supports various Python versions and seamlessly integrates with virtual environments. Navigating through code and refactoring are made efficient with PyCharm's intuitive tools, allowing developers to easily traverse files, classes, and methods. The integrated debugger facilitates smooth debugging experiences, enabling developers to set

breakpoints, inspect variables, and step through code to identify and resolve issues effectively.

PyCharm offers robust testing support, accommodating popular testing frameworks such as py test, unit test, and doc test. Its integrated testing tools empower developers to create, run, and analyze tests, ensuring code reliability. Furthermore, PyCharm seamlessly integrates with version control systems like Git, Mercurial, and SVN, simplifying collaboration and code tracking.

Beyond Python-specific features, PyCharm extends support to web development with frameworks like Django and Flask. This includes features such as templates, code completion, and project management, making it versatile for a range of development needs.

Database tools are also part of PyCharm's toolkit, providing SQL code completion, database exploration, and data manipulation directly within the IDE. The IDE's cross-platform compatibility ensures a consistent user experience across Windows, macOS, and Linux operating systems.

PyCharm offers customization options, allowing users to tailor the interface and choose from various themes to suit individual preferences. It is available in two editions a free and open- source Community edition and a Professional edition with additional features designed for larger projects and web development, available through a subscription.

PyCharm supports popular web frameworks like Django and Flask, making it ideal for web development. It also incorporates database tools for SQL databases and offers web development support, including HTML, CSS, and Java-Script features. PyCharm integrates well with scientific tools such as Jupiter notebooks, NumPy and pandas, making it a preferred choice for data analysis tasks. It caters to cloud and container technologies, simplifying cloud-native development. Extensibility is a key feature, with a wealth of plugins available via the JetBrains Marketplace, enabling developers to customize the IDE. With both Professional and Community editions, PyCharm caters to various user needs, further solidifying its position as a top choice for Python developers. PyCharm offers intelligent code completion, syntax highlighting, error highlighting, and code inspection. It helps developers write clean, error-free code by providing suggestions, warnings, and hints as they type. PyCharm allows for easy navigation through codebases with features like go to Definition, Find Usages, Navigate to Symbol, and Navigate to Class. This makes it convenient to explore and

understand large projects. PyCharm includes a powerful debugger that allows developers to step through code, set breakpoints, inspect variables, and evaluate expressions. It helps in identifying and fixing bugs more efficiently. PyCharm offers a variety of refactoring tools to help improve code quality and maintainability. These tools allow developers to rename variables, extract methods, inline variables, and more, with confidence that the changes are applied consistently. PyCharm can be customized and extended according to individual preferences and workflow requirements. It supports plugins and allows developers to configure key bindings, themes, and UI settings. PyCharm simplifies the management of Python virtual environments, allowing developers to create, activate, and manage project-specific environments effortlessly. It contains modules and packages that help programmers develop software using Python in less time and with minimal effort. This helps in isolating project dependencies and ensuring reproducibility.

Advantages of PyCharm:

- **Intelligent Code Assistance:** PyCharm provides intelligent code completion, code analysis, and error detection features that help developers write code faster and with fewer errors. It offers context-aware suggestions, auto-imports, and quick-fixes to streamline coding workflows.
- **Powerful Debugger:** PyCharm includes a robust debugger with features like breakpoints, watch expressions, step-by-step execution, and variable inspection. It allows developers to debug Python code effectively, identify and fix issues, and understand code behavior during runtime.
- **Advanced Refactoring:** PyCharm offers a wide range of code refactoring tools that enable developers to improve code structure, readability, and maintainability. It supports refactoring's such as renaming, extracting Variables/methods/functions, moving code blocks, and more, ensuring code changes are safe and efficient.
- **Comprehensive Testing Support:** PyCharm integrates seamlessly with popular Python testing frameworks like py test, unit test, and doc test. It provides features for test creation, execution, result visualization, and code coverage analysis, enabling developers to write high-quality tests and ensure code reliability.

- **Version Control Integration:** PyCharm offers built-in support for version control systems like Git, Mercurial, and Subversion. It provides features for commit management, branch visualization, conflict resolution, and code collaboration, facilitating efficient collaboration and version control workflows.
- **Productive Coding Environment:** PyCharm includes features like code templates, code snippets, and customizable keyboard shortcuts that enhance developer productivity. It supports multiple project types, virtual environments, and package managers, allowing developers to tailor their coding environment to their specific needs.
- **Web Development Tools:** PyCharm provides comprehensive support for web development with Python, including popular web frameworks like Django, Flask, and Pyramid. It offers features for web project creation, template editing, JavaScript support, and integration with front-end development tools, enabling full-stack development within the IDE.
- **Database Tools:** PyCharm includes database tools that allow developers to work with databases directly from the IDE. It supports various database systems like MySQL, PostgreSQL, SQLite, and others, providing features for database exploration, query execution, and schema management.
- **Extensibility:** PyCharm has a rich ecosystem of plugins and extensions that extend its functionality with additional features and integrations. Developers can install plugins from the JetBrains Marketplace or create custom plugins to enhance the IDE's capabilities and tailor it to their specific requirements.
- **Cross-Platform Support:** PyCharm is available on multiple platforms, including Windows, macOS, and Linux, ensuring a consistent development experience across different operating systems. It allows developers to work seamlessly on their preferred platform without sacrificing productivity or features.

4.2 HTML

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: Hypertext simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you

have clicked on a hypertext. Hypertext is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages. HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content. Developed to organize and structure documents on the internet, HTML uses a system of tags to define elements within a document. Each tag, consisting of an opening and closing pair, surrounds content such as text, images, or links.

HTML's collaboration with CSS for styling and JavaScript for dynamic behavior makes it an essential tool for crafting modern, interactive websites and applications. HTML documents can be created and edited using plain text editors, integrated development environments (IDEs), or specialized HTML editors. Its simplicity and versatility make HTML a fundamental language for anyone entering the world of web development.

Applications of HTML

- As mentioned before, HTML is one of the most widely used language over the web. I'm going to list few of them here:
- Web pages development - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- Internet Navigation - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- Responsive UI - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- Offline support HTML pages once loaded can be made available offline on the machine without any need of internet.

- Game development- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

4.3 XAMPP

XAMPP is an abbreviation where X stands for Cross-Platform, A stands for Apache, M stands for MYSQL, and the Ps stand for PHP and Perl, respectively. It is an open-source package of web solutions that includes Apache distribution for many servers and command-line executables along with modules such as Apache server, MariaDB, PHP, and Perl. XAMPP helps a local host or server to test its website and clients via computers and laptops before releasing it to the main server. It is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl, MySQL database, and PHP through the system of the host itself. Among these technologies, Perl is a programming language used for web development, PHP is a backend scripting language, and MariaDB is the most vividly used database developed by MySQL. XAMPP is one of the widely used cross-platform web servers, which helps developers to create and test their programs on a local webserver. It was developed by the Apache Friends, and its native source code can be revised or modified by the audience. It consists of Apache HTTP Server, MariaDB, and interpreter for the different programming languages like PHP and Perl. It is available in 11 languages and supported by different platforms such as the IA-32 package of Windows & x64 package of macOS and Linux.

It is an open- source package of web solutions that includes Apache distribution for many servers and command-line executables along with modules such as Apache server, MariaDB, PHP, and Perl.

XAMPP helps a local host or server to test its website and clients via computers and laptops before releasing it to the main server. It is a platform that furnishes a suitable environment to test and verify the working of projects based on Apache, Perl, MySQL database, and PHP through the system of the host itself. Among these technologies, Perl is a programming language used for web development, PHP is a backend scripting language, and MariaDB is the most vividly used database developed by MySQL.

XAMPP is a popular cross- platform web server that allows programmers to write and test their code on a local webserver. It was created by Apache Friends, and the public can revise or modify its native source code. It includes MariaDB, Apache

HTTP Server, and interpreters for PHP and Perl, among other computer languages. Because of XAMPP's simplicity of deployment, a developer can quickly and easily install a WAMP or LAMP stack on an operating system, with the added benefit that common add-in apps like WordPress and Joomla can also be loaded.

Applications of XAMPP

- The creators of XAMPP intended it to be used as a development tool, allowing web designers and programmers to test their work on their personal computers without the need for Internet connections. Many key security elements are disabled by default to make this as simple as feasible. XAMPP is used to serve the web pages on the Internet.
- It can also use to create and manipulate databases in MariaDB and SQLite, among other databases.
- Once XAMPP is installed, an FTP client can connect to a local host and treat it as if it were a remote host. When installing a content management system like Joomla or WordPress, using a tool like FileZilla. You can also use an HTML editor to connect to a local host through FTP.

4.4 MYSQL

MySQL is an open-source relational database management system. As with other relational databases, MySQL stores data in tables made up of rows and columns. Users can define, manipulate, control, and query data using Structured Query Language, more commonly known as SQL. MySQL's name is a combination of "My," the name of MySQL creator Michael Widenius's daughter, and "SQL". A flexible and powerful program, MySQL is the most popular open-source database system in the world. As part of the widely-used LAMP technology stack which consists of a Linux-based operating system, the Apache web server, a MySQL database, and PHP for processing, it's used to store and retrieve data in a wide variety of popular applications, websites, and services.

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the

instructions are matched. The process of MySQL environment is the same as the client-server model.

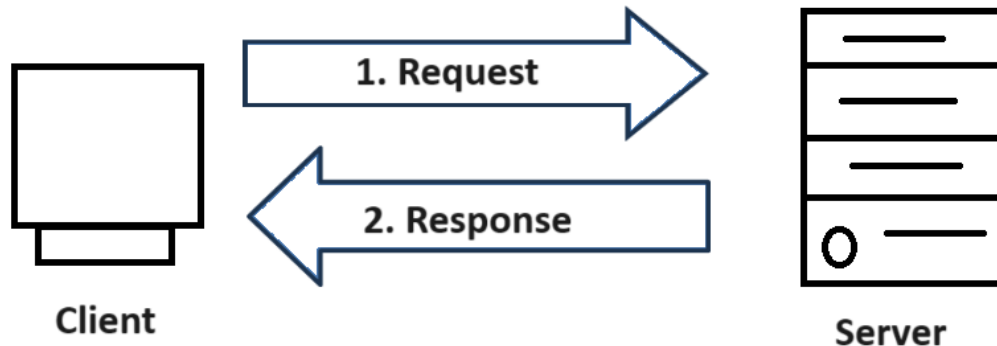


Fig 4.1: Client Server Model

The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

MySQL is the world's most popular open-source database. According to DB-Engines, MySQL ranks as the second-most-popular database, behind Oracle Database. MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com. MySQL is used with other programs to implement applications that need relational database capability. The MySQL server software itself and the client libraries use dual-licensing distribution.

Since MySQL is open source, it includes numerous features developed in close cooperation with users over more than 25 years. So, it's very likely that your favorite application or programming language is supported by MySQL Database.

1. **Open Source:** MySQL is open-source software, which means it's free to use and has a large community of developers contributing to its improvement.
 2. **Relational:** MySQL follows the relational database model, allowing users to organize data into tables with rows and columns, facilitating efficient data storage and retrieval.
 3. **Reliability:** MySQL has been around for a long time and is known for its stability and reliability.
 4. **Performance:** MySQL is optimized for performance, making it capable of handling high-volume transactions and large datasets efficiently.
 5. **Scalability:** MySQL can scale both vertically and horizontally to accommodate growing data and user loads. You can add more resources to a single server or distribute the workload across multiple servers using techniques like replication.
 6. **Compatibility:** MySQL is widely supported by many programming languages, frameworks, and tools. It offers connectors and APIs for popular languages like PHP, Python, Java, and more, making it easy to integrate with your existing software stack.
 7. **Security:** MySQL provides robust security features to protect your data, including access controls, encryption, and auditing capabilities. With proper configuration, you can ensure that only authorized users have access to sensitive information.
- MySQL stands out as a powerful and versatile open-source Relational Database Management System. Its reliability, scalability, performance, and compatibility make it a preferred choice across various industries, from e-commerce to healthcare and social media. With its robust security features and widespread support, MySQL remains an essential tool for efficiently managing and manipulating data in the modern world.

4.6 ALGORITHMS

4.6.1 RANDOM FOREST CLASSIFIER

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble

meta- algorithm that improves the accuracy of machine learning algorithms. The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome. A random forest eradicates the limitations of a decision tree algorithm. It reduces the over fitting of datasets and increases precision. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. It generates predictions without requiring many configurations in packages like Scikit-learn.

Features of a Random Forest Algorithm:

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of over fitting in decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.
- Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.
- A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.
- The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.

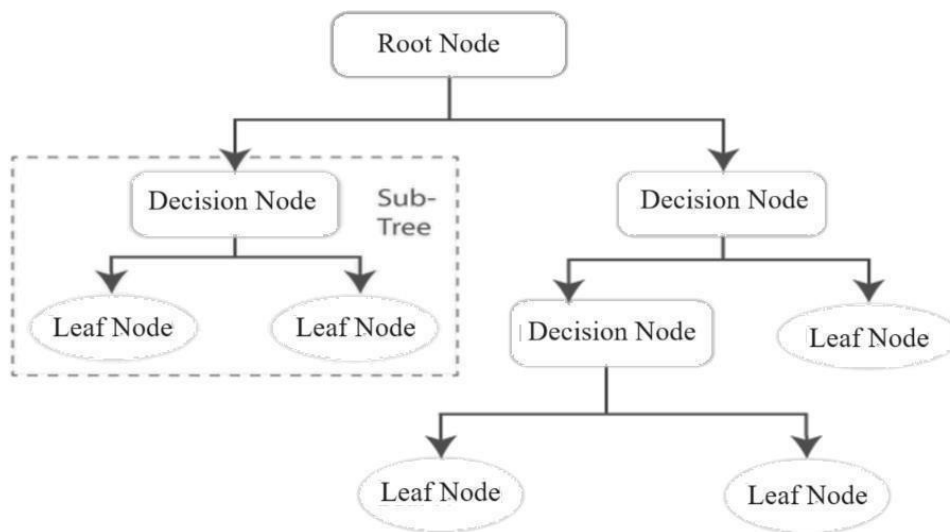


Fig 4.2: Three types of nodes in a decision tree

The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview of these fundamental concepts will improve our understanding of how decision trees are built. Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let's take a simple example of how a decision tree works. Suppose want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either buying or not buying.

The main features that determine the choice include the price, internal storage, and Random Access Memory.

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction. Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let's assume we have only four decision trees. In this case, the training data comprising the phone's observations and features will be divided into four root nodes.

The root nodes could represent four features that could influence the customer's choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees. The outcome chosen by most decision trees will be the final choice. If three trees predict buying, and one tree predicts not buying, then the final prediction will be buying. In this case, it's predicted that the customer will buy the phone. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

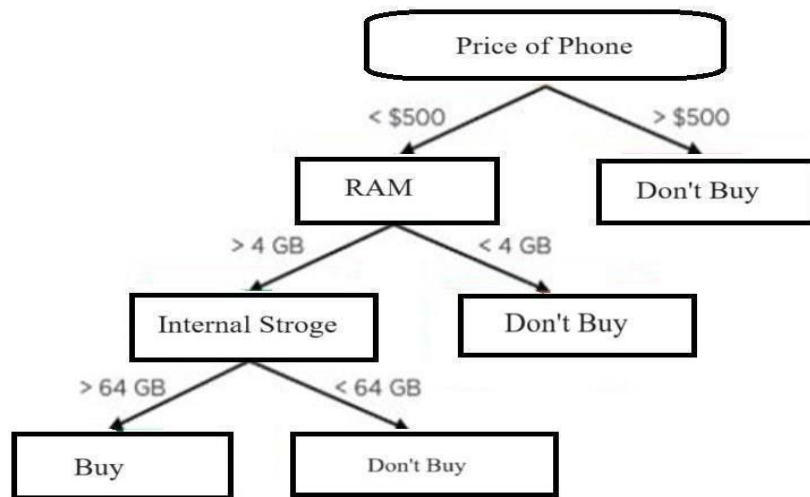


Fig 4.3: Decision tree diagram for features of the phone

4.6.2 Support Vector Machine

SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that the new data point can be easily put in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

SVM can be of two types

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

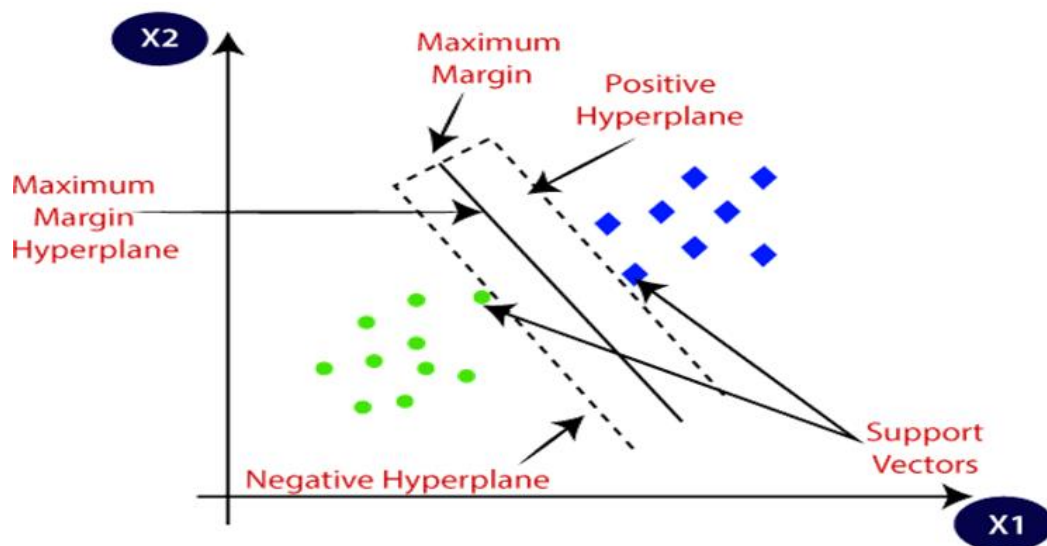


Fig: 4.4: Categories by Hyper plane

Hyperplane and Support Vectors in the SVM algorithm

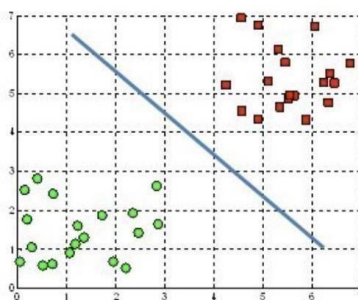
There can be multiple lines/decision boundaries to segregate the classes in n -dimensional space, but it is necessary to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyper plane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyper plane will be a straight line. And if there are 3 features, then hyper plane will be a 2-dimension plane.

The dimensions of the hyper plane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyper plane will be a straight line. And if there are 3 features, then hyper plane will be a 2-dimension plane.

Support Vectors

A hyperplane in \mathbb{R}^2 is a line



A hyperplane in \mathbb{R}^3 is a plane

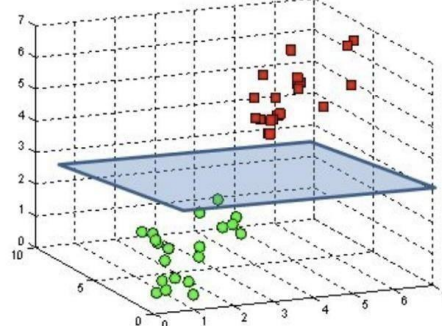


Fig 4.5: Hyper planes in 2D and 3D feature space

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane is termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

4.6.3 XGBoost

Extreme Gradient Boosting is a scalable, distributed GBDT machine learning framework. It is the top machine learning package for regression, classification, and ranking tasks, and it supports parallel tree boosting. To understand XGBoost, first understand the machine learning ideas and methods on which it is based: supervised machine learning, decision trees, ensemble learning, and gradient boosting. Supervised machine learning use algorithms to train a model to detect patterns in a dataset containing labels and features, and then employs the trained model to predict the labels on the features of a new dataset. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time. XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

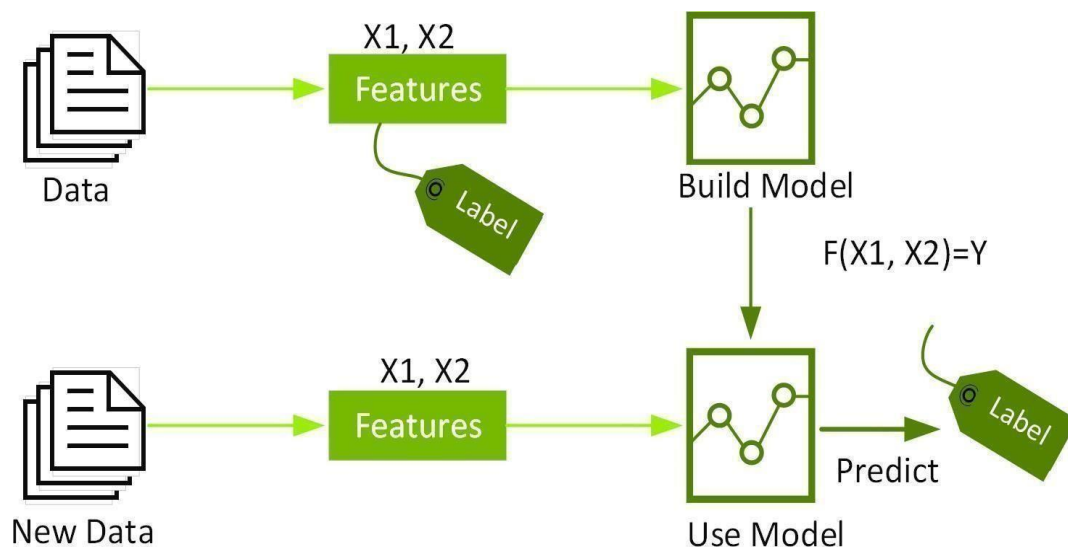


Fig 4.6: Diagram for XG Boost

XGBoost Benefits and Attributes

1. High accuracy: XGBoost is known for its accuracy and has been shown to outperform other machine learning algorithms in many predictive tasks.
2. Scalability: It is highly scalable and can handle large datasets with millions of rows and columns.
3. Efficiency: It is designed to be computationally efficient and can quickly train models on large datasets.
4. Flexibility: It supports a variety of data types and objectives, including regression, classification, and ranking problems.
5. Regularization: It incorporates regularization techniques to avoid overfitting and improve generalization performance.
6. Interpretability: It provides feature importance scores that can help users understand which features are most important for making predictions.
7. Open-source: XGBoost is an open-source library that is widely used and supported by the data science community.

Key Takeaways on XGBoost Algorithm

- XGBoost is an efficient and powerful algorithm for machine learning.
- It is a gradient boosting algorithm that can be used for classification and regression problems.

- It is an optimized version of decision tree algorithms and uses a weighted sum of decision trees to make predictions.
- XGBoost can be used for hyperparameter optimization to find the best model for a given dataset.
- XGBoost has been proven to be more accurate than traditional machine learning algorithms.
- XGBoost algorithm is highly scalable and can be used for large datasets.

XGBoost, with its efficient regularization, parallel processing, and handling of sparse data, shines in diverse applications. Random Forest, a robust ensemble learner, excels in reducing overfitting and offers flexibility in various domains. XGBoost has 3 builtin tree methods, namely exact, approx. and hist. Along with these tree methods, there are also some free-standing updaters including refresh, prune and sync. By evaluating a tree of if-then-else true/false feature questions and estimating the minimum number of questions required to assess the probability of making a correct decision, decision trees generate a model that predicts the label. Decision trees can be used to predict a category or a continuous numeric value using classification or regression. A decision tree is used in the basic example below to predict a property price (the label) based on the size and number of bedrooms (the features).

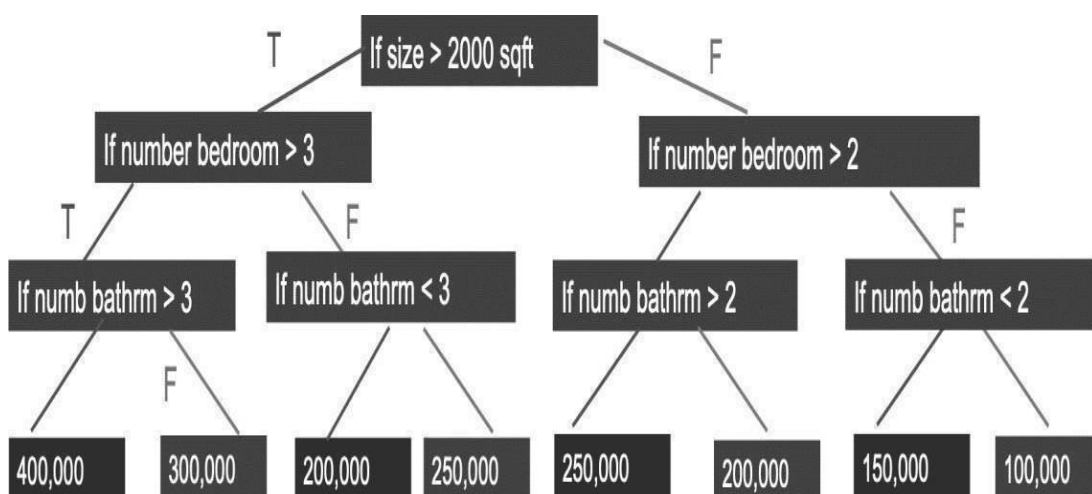


Fig 4.7: Decision Tree

GBDT is a decision tree ensemble learning algorithm for classification and regression that is similar to random forest. To create a better model, ensemble learning techniques mix different machine learning methods. Both random forest and GBDT construct a model from multiple decision trees. The distinction is in how the trees are constructed and joined.

A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test and each leaf node (terminal node) holds a class label.

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

Types of Decision Trees

Hunt’s algorithm, which was developed in the 1960s to model human learning in Psychology, forms the foundation of many popular decision tree algorithms, such as the following:

- **ID3:** Ross Quinlan is credited within the development of ID3, which is shorthand for “Iterative Dichotomiser 3.” This algorithm leverages entropy and information gain as metrics to evaluate candidate splits. Some of Quinlan’s research on this algorithm from 1986 can be found in web.
- **C4.5:** This algorithm is considered a later iteration of ID3, which was also developed by Quinlan. It can use information gain or gain ratios to evaluate split points within the decision trees.
- **CART:** The term, CART, is an abbreviation for “classification and regression trees” and was introduced by Leo Breiman. This algorithm typically utilizes Gini impurity to identify the ideal attribute to split on. Gini impurity measures how often a randomly chosen attribute is misclassified. When evaluating using Gini impurity, a lower value is more ideal.

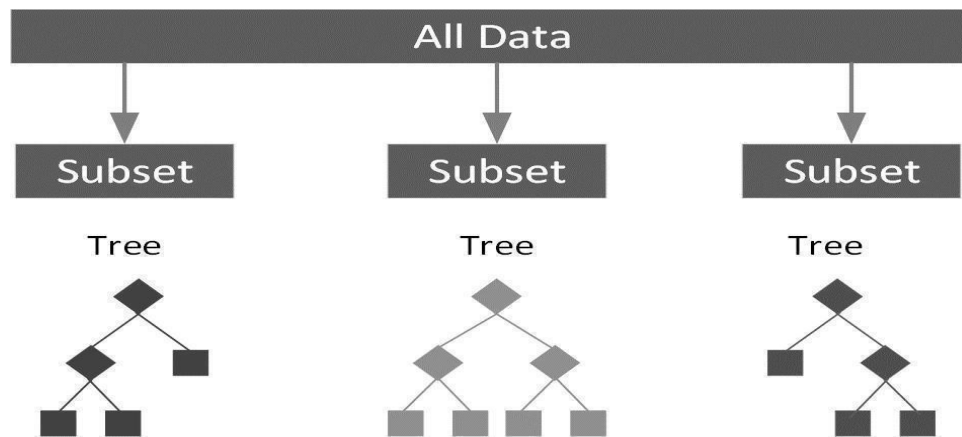


Fig 4.8: Random Forest Tree Structure

Random forest uses a technique called bagging to build full decision trees in parallel from random bootstrap samples of the data set. The final prediction is an average of all of the decision tree predictions.

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

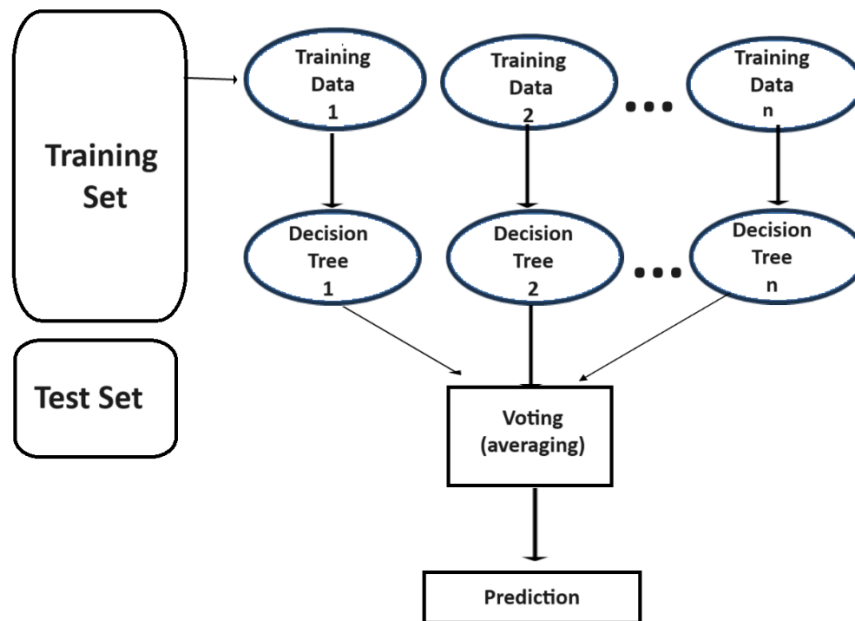


Fig 4.9: Random Forest Algorithm

Applications of Random Forest

There are mainly four sectors where Random Forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

The basic idea behind this is to combine multiple decision trees determining the final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

4.6.4 AdaBoost Algorithm

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are

also called Decision Stumps. Adaboost is less prone to overfitting as the input parameters are not jointly optimized. The accuracy of weak classifiers can be improved by using Adaboost. Nowadays, Adaboost is being used to classify text and images rather than binary classification problems.

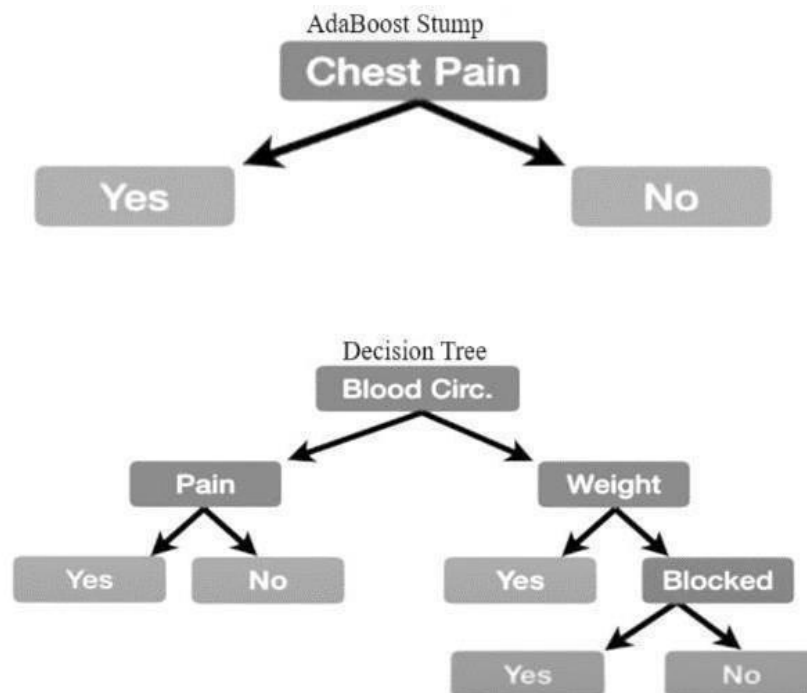


Fig 4.10: Diagram for AdaBoost Algorithm

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

This figure shows how the first model is made and errors from the first model are noted by the algorithm. The record which is incorrectly classified is used as input for the next model. This process is repeated until the specified condition is met. As you can see in the figure, there are 'n' number of models made by taking the errors from the previous

model. This is how boosting works. The models 1,2, 3,...,N are individual models that can be known as decision trees. All types of boosting models work on the same principle. It makes 'n' number of decision trees during the data training period. As the first decision tree/model is made, the incorrectly classified record in the first model is given priority. Only these records are sent as input for the second model. The process goes on until we specify a number of base learners we want to create. repetition of records is allowed with all boosting techniques. Since we now know the boosting principle, it will be easy to understand the AdaBoost algorithm. When the random forest is used, the algorithm makes an 'n' number of trees. It makes proper trees that consist of a start node with several leaf nodes. Some trees might be bigger than others, but there is no fixed depth in a random forest. With AdaBoost, however, the algorithm only makes a node with two leaves, known as Stump. Ada Boosting is best used to boost the performance of decision trees and this is based on binary classification problems. AdaBoost was originally called AdaBoost.M1 by the author. More recently it may be referred to as discrete Ada Boost. As because it is used for classification rather than regression. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners.

4.6.5 Gradient boosting

Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As it is known that the errors in machine learning algorithms are broadly classified into two categories i.e., Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Unlike, Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed and i.e., Decision Stump. Like, AdaBoost, it can tune the n estimator of the gradient boosting algorithm. However, if the value of n estimator is not mentioned, the default value of an estimator for this algorithm is 100.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error and when it is used as a classifier then the cost function is Log loss.

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART. Gradient Boosting tends to train many models in a gradual, additive, and sequential manner. The primary difference between AdaBoost and Gradient Boosting Algorithm is the way in which the two algorithms identify the shortcomings of weak learners (in this case decision trees). While the AdaBoost model explores the shortcomings by making use of high weight data points, gradient boosting tends to perform the same by using gradients in the loss function ($y=ax+b+e$, e needs a special mention as it is the error term). The loss function accounts to be a measure indicating how good model's coefficients are at performing fitting the underlying data.

A logical understanding of loss function would be directly interlinked with what we are trying to optimize. Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function ($y=ax+b+e$, e needs a special mention as it is the error term). The loss function is a measure indicating how good model's coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimize.

For example, if we are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices. Similarly, if our goal is to classify credit defaults, then the loss function would be a measure of how good our predictive model is at classifying bad loans. One of the biggest motivations of using gradient boosting is that it allows one to

optimize a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.

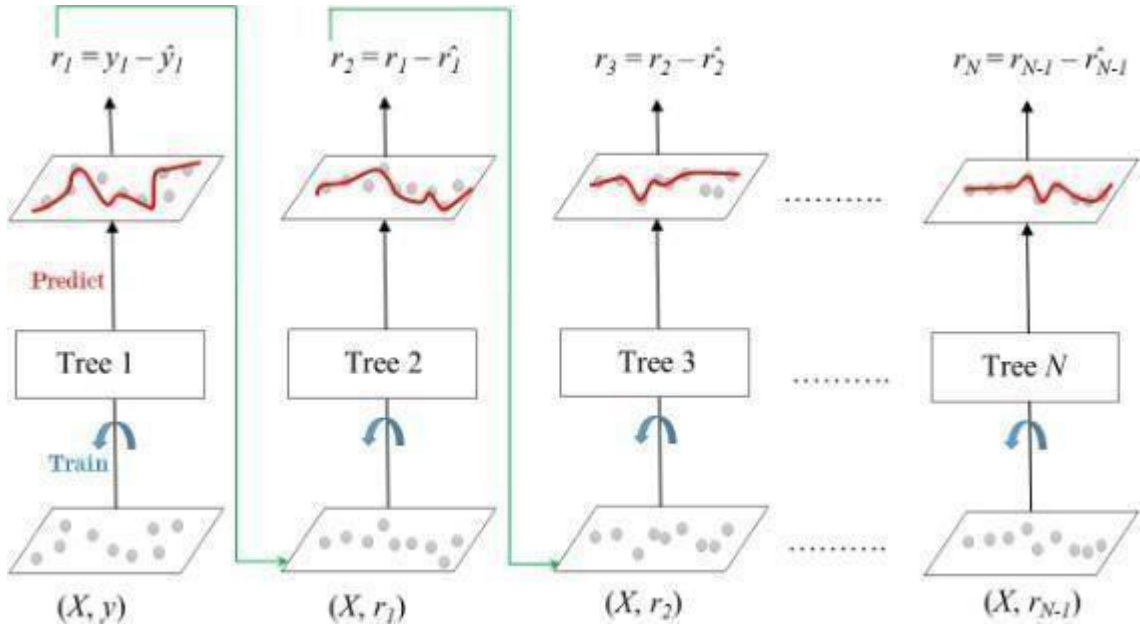


Fig 4.11: Gradient Boosted Trees for Regression

The ensemble consists of M trees. Tree1 is trained using the feature matrix X and the labels y . The predictions labeled \hat{y}_1 are used to determine the training set residual errors r_1 . Tree2 is then trained using the feature matrix X and the residual errors r_1 of Tree1 as labels. The predicted results \hat{r}_1 are then used to determine the residual r_2 . The process is repeated until all the M trees forming the ensemble are trained. There is an important parameter used in this technique known as Shrinkage. Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (η) which ranges between 0 to 1. There is a trade-off between η and the number of estimators, decreasing learning rate needs to be compensated with increasing estimators in order to reach certain model performance. Since all trees are trained now, predictions can be made. Each tree predicts a label and the final prediction is given by the formula.

CHAPTER 5

PROPOSED APPROACH AND SYSTEM ARCHITECTURE

The proposed system for phishing website detection using machine learning algorithms aims to overcome the limitations of existing systems. One approach is to use supervised learning techniques that do not require labelled data for training. This can reduce the time and cost of data labelling and improve scalability.

- Another approach is to incorporate multiple machine learning algorithms to enhance the accuracy and robustness of the system. The system can also be augmented with additional features such as website behavior analysis and user behavior monitoring to improve its ability to detect phishing websites.
- Overall, the proposed system aims to improve the accuracy, efficiency, and effectiveness of phishing website detection using machine learning algorithms.

SOFTWARE AND HARDWARE REQUIREMENTS

- **Hardware**

- Operating system : Windows 7 or 7+
- RAM : 8 GB
- Hard disc or SSD : More than 500 GB
- Processor : Intel 3rd generation or high or Ryzen with 8 GB RAM

- **Software**

- Software's : Python 3.6 or high version
- IDE :PyCharm
- Framework : Flask

5.1 Block Diagram

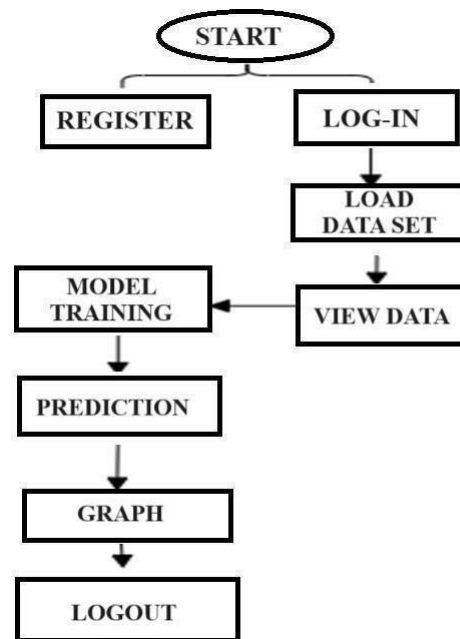


Fig 5.1: Block Diagram of Phishing Website Detection Using Machine Learning

5.2 System Architecture

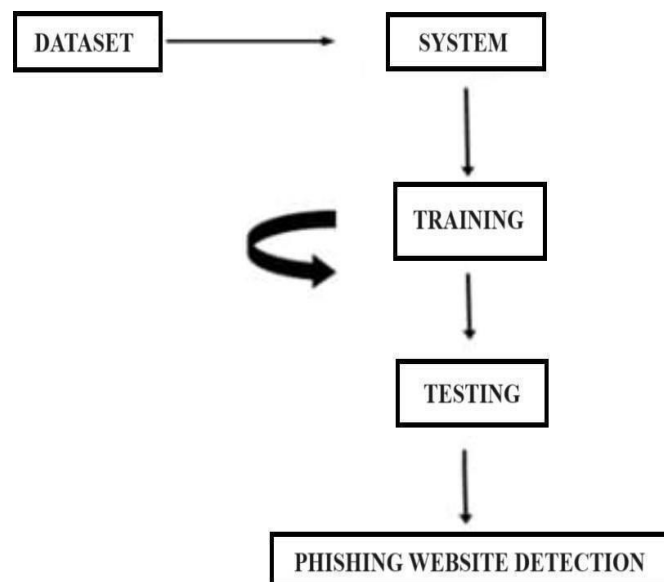


Fig 5.2: System Architecture for Phishing Website Detection Using ML

SYSTEM DESIGN

Input Design

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.

Objectives for Input Design

The objectives of input design are

- To design data entry and input procedures.
- To reduce input volume.
- To design source documents for data capture or devise other data capture methods.
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Output Design

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design

The objectives of input design are

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.

- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

MODULES

1. User

1.1 View Home page

Here user view the home page of the phishing website prediction web application.

1.2 View Upload page

In the about page, users can learn more about the phishing prediction.

View Page

In view page, user can see the dataset.

1.3 Input Model

The user must provide input values for the certain fields in order to get results.

1.4 View Results

User view's the generated results from the model.

1.5 View score

Here user have ability to view the score in %.

5.3 UML DIAGRAM

UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of Object-Oriented tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

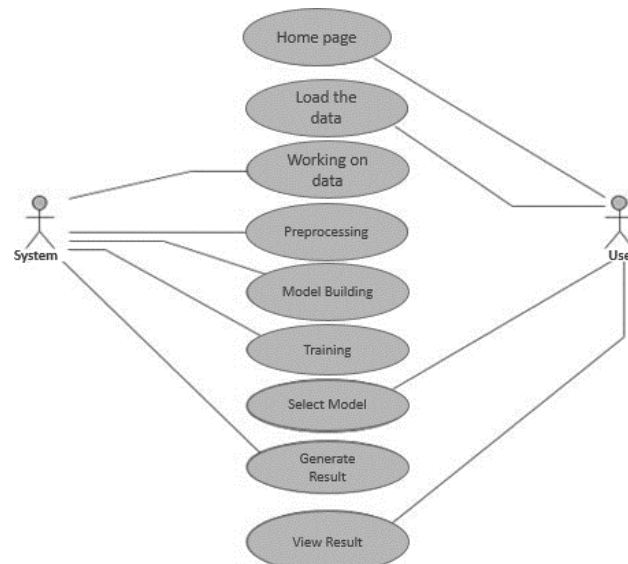


Fig 5.3 UML Diagram for Phishing website detection using machine learning

5.4 CLASS DIAGRAM

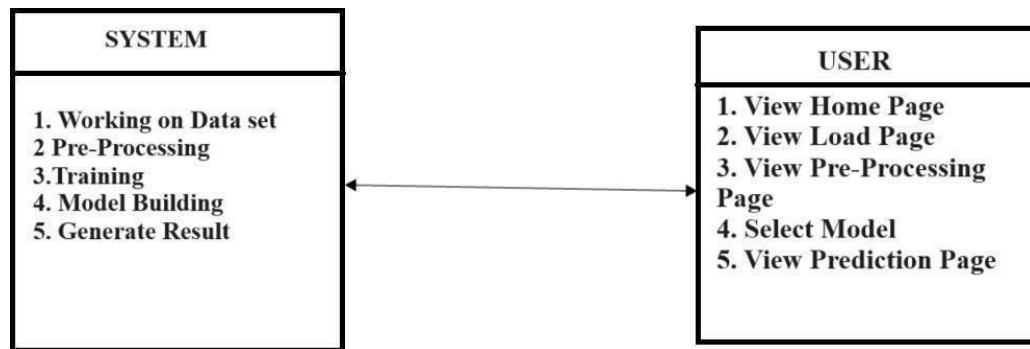


Fig 5.4 Class Diagram for Phishing website detection using ML

5.5 SEQUENCE DIAGRAM

- A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order.
- It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

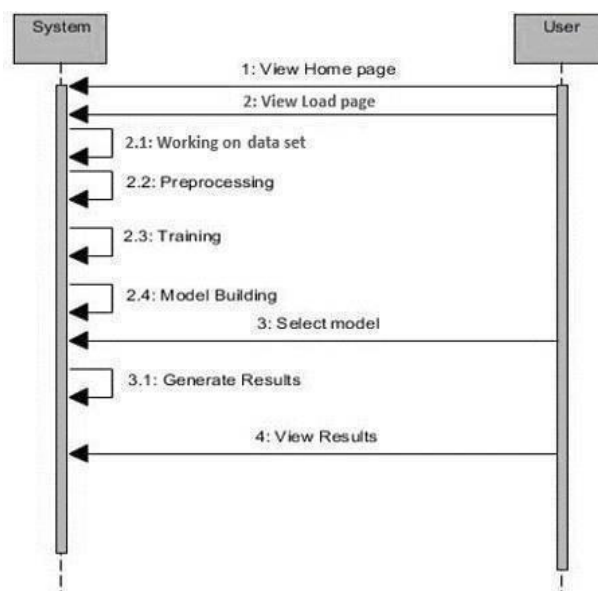


Fig 5.5 Sequence Diagram for Phishing website detection using machine learning

5.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. An activity diagram is a type of UML diagram that is used to represent the flow of activities or processes in a system. It typically consists of various elements such as actions, control flow, decision points, and start and end points to illustrate how different activities are related and how they are executed. Activity diagrams are often used to model business processes, software workflows, and various other processes in system development. If you have a specific question or need assistance with creating or understanding an activity diagram.

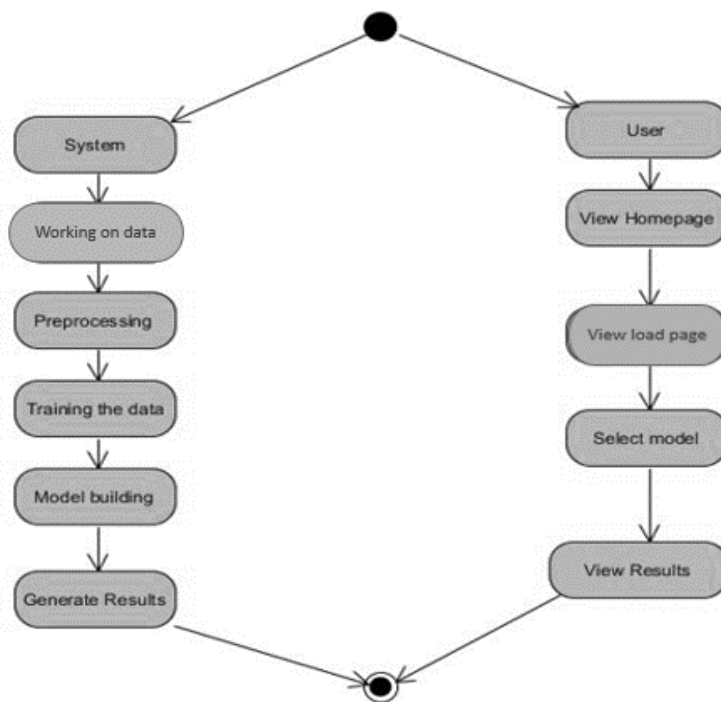


Fig 5.6 Activity Diagram for Phishing website detection using machine learning

5.7 ER DIAGRAM

An ER model describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

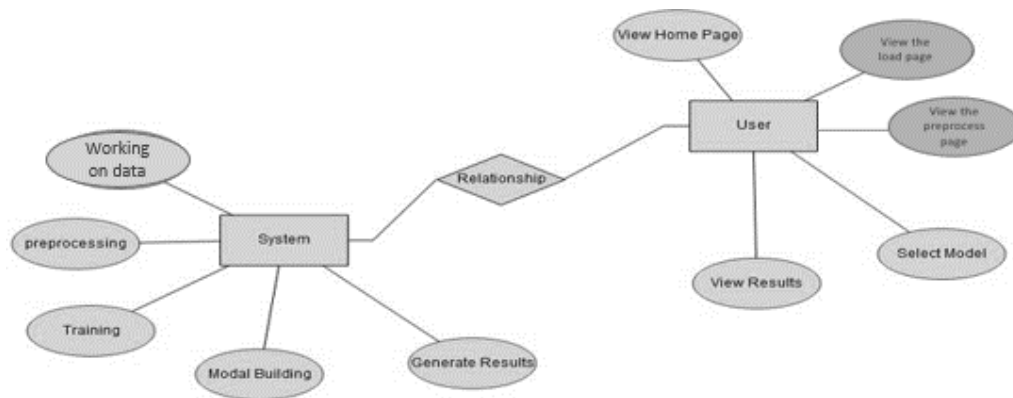
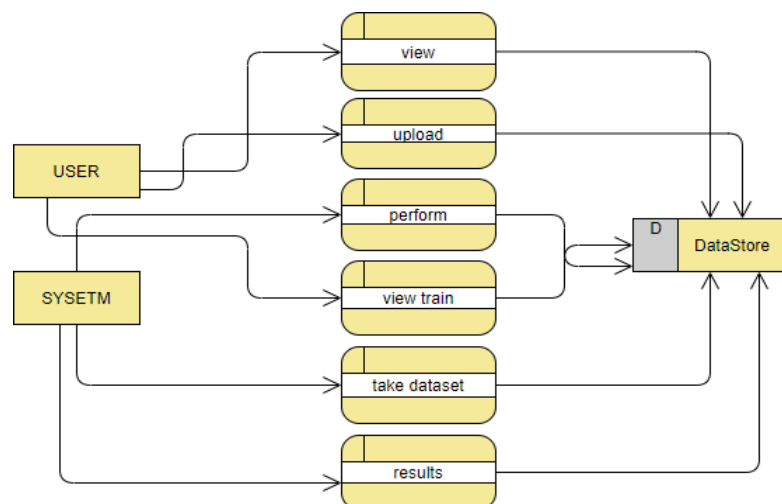


Fig 5.7 ER Diagram for Phishing website detection using machine learning

5.8 DFD DIAGRAM

A DFD is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.



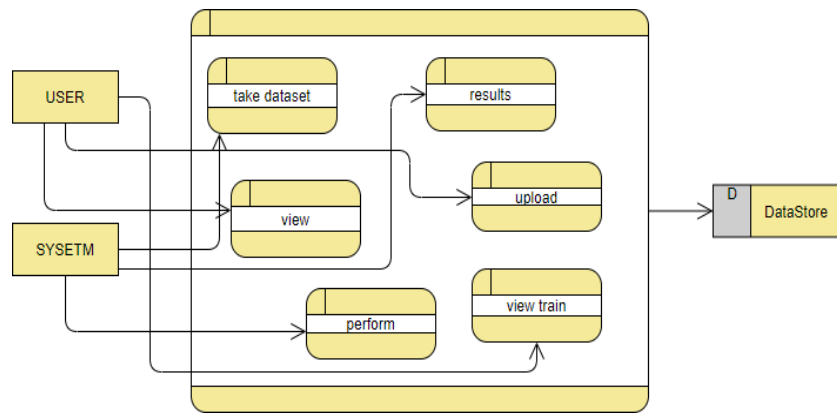


Fig 5.8 DFD Diagram for Phishing website detection using machine learning

CHAPTER 6

IMPLEMENTATION

Implementation represents the development stages of the proposed application. This chapter also describes the functionalities and testing methods for proposed applications.

6.1 Data Collection

- Utilize various sources to compile a comprehensive dataset. This could include publicly available datasets like the Phish Tank database, security research repositories, or data provided by cybersecurity companies.
- Consider using web scraping techniques to gather real-time data from online sources. Tools like BeautifulSoup or Scrapy can be employed to extract HTML content, URL features, and other relevant information from websites.
- Collaborate with security professionals and researchers to access proprietary datasets or obtain labeled data for training purposes.
- Ensure the dataset includes a diverse range of phishing and legitimate websites, covering different types of phishing attacks and legitimate web domains.

6.2 Data Gathering and Preprocessing

- Develop robust data gathering pipelines to collect data from multiple sources efficiently. This may involve creating custom scripts or leveraging APIs provided by data sources.
- Implement data preprocessing steps to clean and prepare the collected data for analysis. This includes handling missing values, removing duplicates, and standardizing data formats.
- Use feature extraction techniques to derive meaningful features from raw data. For example, extract features from URLs such as domain length, presence of subdomains, and use of HTTPS.
- Employ techniques like tokenization and vectorization to process textual data extracted from HTML content or WHOIS records.
- Normalize numerical features to ensure consistency and mitigate the impact of scale differences between different features.

6.3 Model Training

- Select a range of machine learning algorithms suitable for binary classification tasks. Experiment with algorithms like Random Forest, XGBoost, Gradient Boosting, Support Vector Machine (SVM), and Logistic Regression to find the best-performing model.
- Split the preprocessed dataset into training, validation, and testing sets. Use techniques like k-fold cross-validation to assess model performance and generalize better.
- Optimize model hyperparameters using techniques like grid search or randomized search to maximize performance on the validation set.
- Train multiple models and compare their performance metrics to select the most effective one for phishing website detection.

6.4 Testing and Evaluating the Model

- Evaluate trained models using a variety of evaluation metrics to gain a comprehensive understanding of their performance. Consider metrics like accuracy, precision, recall, F1-score, and area under the ROC curve (ROC-AUC).
- Conduct in-depth analysis of model predictions using techniques like confusion matrices to identify areas of improvement and potential sources of misclassification.
- Perform sensitivity analysis to assess the impact of different features on model predictions and identify influential factors in phishing website detection.
- Validate the robustness of the model by testing it on unseen data, including websites not present in the training dataset, to ensure generalization to real-world scenarios.
- Building the Frontend:
- Design an intuitive and user-friendly frontend interface that enables users to interact with the phishing website detection system easily.
- Implement input forms or text fields where users can enter URLs for analysis.
- Develop backend functionality to process user inputs, make predictions using the trained machine learning model, and return results to the frontend.
- Utilize web development frameworks like Flask, Django, or React to create responsive and interactive frontend interfaces.

- Incorporate visualizations or feedback mechanisms to enhance user experience and provide insights into model predictions.

6.5 Working for Phishing Website Detection

- Users access the frontend interface through web browsers or standalone applications.
- Input URLs of interest into the frontend interface.
- The frontend sends the URL data to the backend server for analysis.
- The backend server applies the trained machine learning model to predict whether each URL is phishing or legitimate.
- Prediction results are returned to the frontend and displayed to the user, along with relevant information such as confidence scores or probabilities.
- Users can review the prediction results and take appropriate actions based on the detected risk levels of the analyzed websites.

CHAPTER 7

RESULTS AND DISCUSSION

This chapter shows the result of the project and the changes that occur, as result of its action this typically involve improvement for a model. When designing a project, it is important to know what a project outcome are so it has a way or measuring success.

7.1 Home Page

The homepage of a phishing website detection system using machine learning serves as the entry point for users, providing essential information about the system's functionality, features, and how to utilize it effectively.

Here user view the home page of phishing website prediction web application.

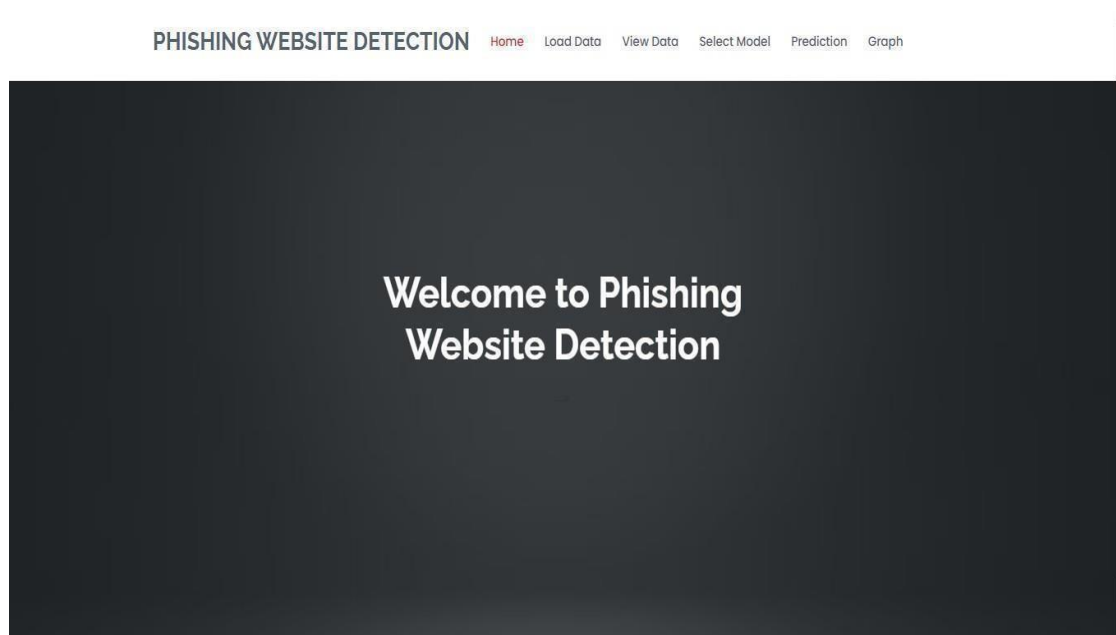


Fig 7.1: Home Page

7.2 Load Page

Users can load a dataset for phishing website detection involves creating a user interface that allows users to upload their dataset files. It includes a file upload component, such as a csv file input field, to allow users to select dataset files from their local storage. Clear instructions and guidelines are provided on supported file formats and any specific requirements for the dataset files.

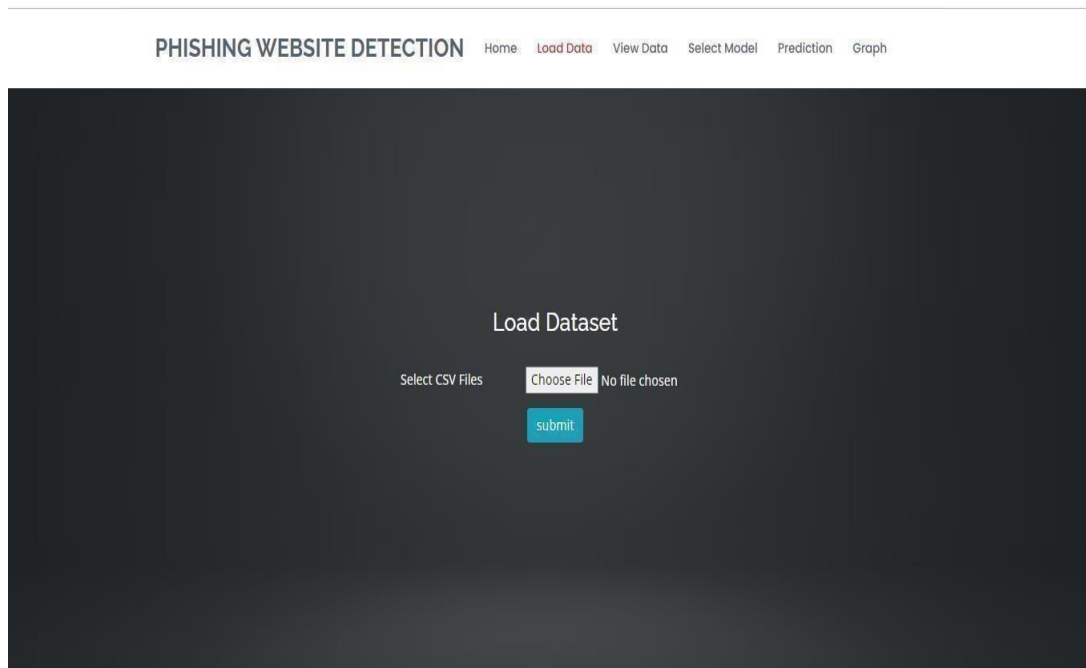


Fig 7.2: Load Page

7.3 View Page

Here the uploaded data set can be seen.

PHISHING WEBSITE DETECTION					
Home Load Data View Data Select Model Prediction Graph					
S/N	Domain	Have_IP	Have_At	URL	Le
1	graphicriver.net	0	0	1	
2	ecnavi.jp	0	0	1	
3	hubpages.com	0	0	1	
4	extratorrent.cc	0	0	1	
5	icicibank.com	0	0	1	
6	nypost.com	0	0	1	
7	kienthuc.net.vn	0	0	1	
8	thenextweb.com	0	0	1	
9	tobogo.net	0	0	1	
10	akhbarelyom.com	0	0	1	
11	tunein.com	0	0	1	
12	tune.pk	0	0	1	
13	sfglobe.com	0	0	1	
14	mic.com	0	0	1	
15	thenextweb.com	0	0	1	

Fig 7.3: View Page

7.4 URL Input

The prediction page features an input form where users can enter one or multiple URLs for analysis. Providing placeholder text or instructions to guide users on the type of input expected (e.g., "Enter URL here").

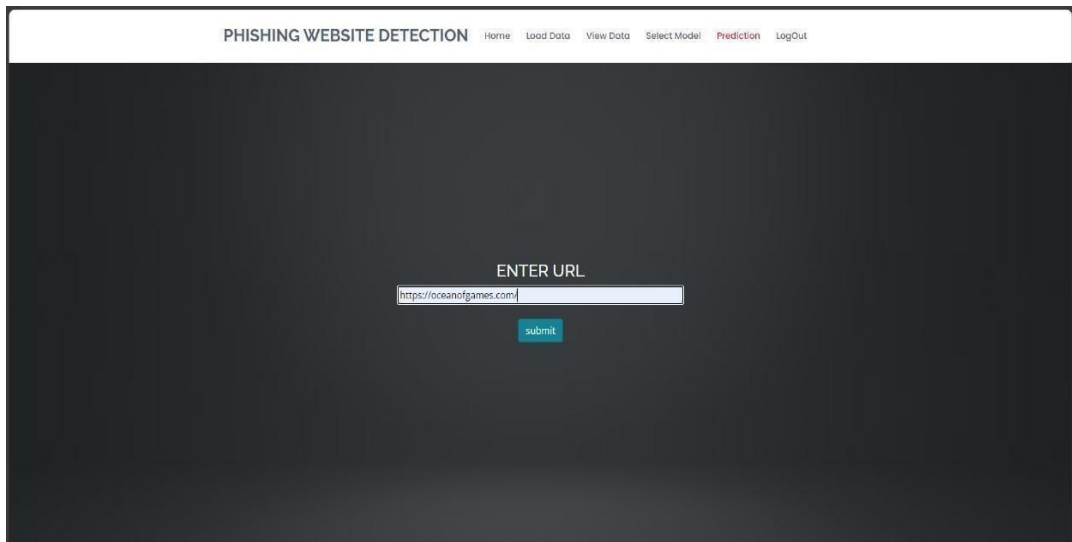


Fig 7.4:URL Page

7.5 Prediction

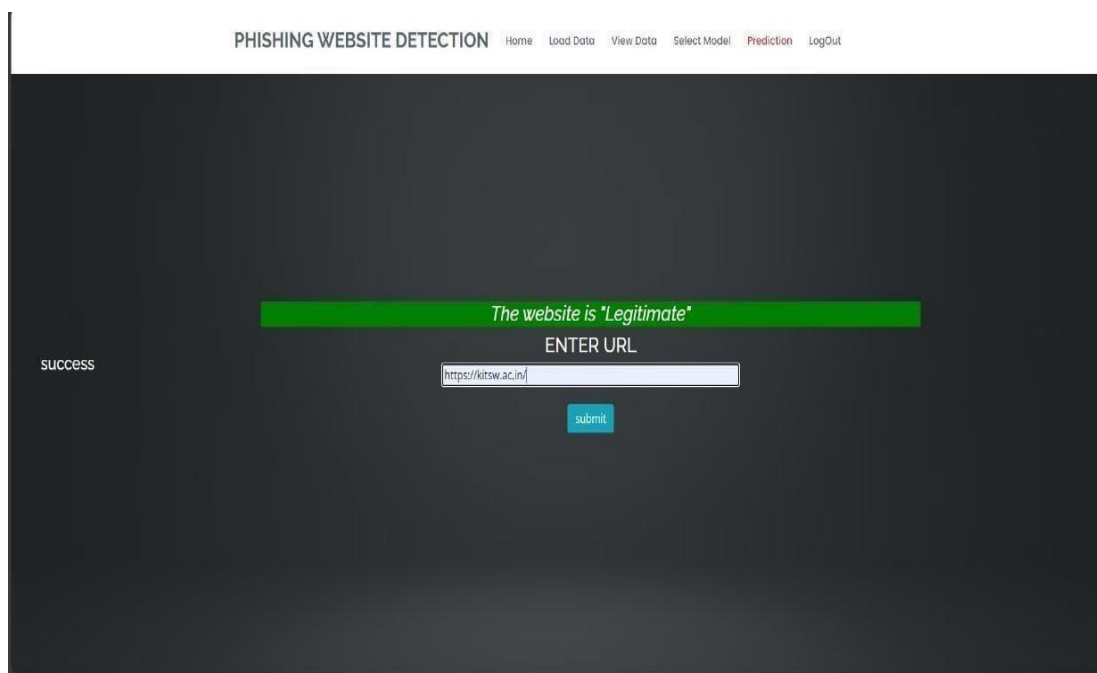


Fig 7.5: Prediction 1

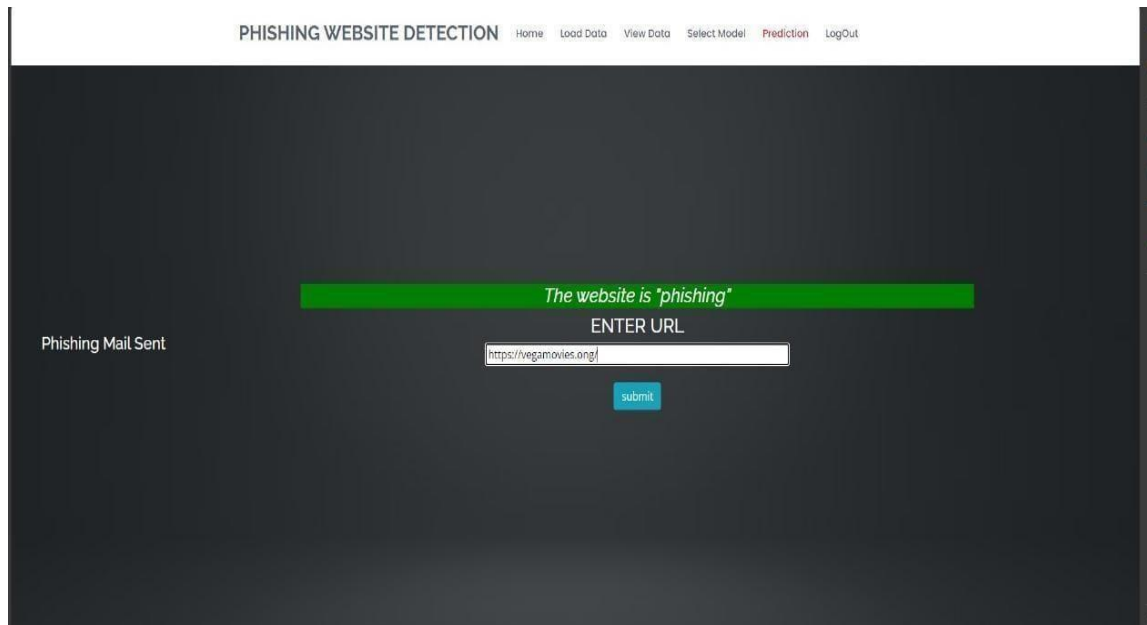


Fig 7.5.1: Prediction 2

The prediction page for phishing website detection using machine learning provides users with real-time insights into the legitimacy of a given URL. Upon entering a URL into the interface, the system utilizes a trained machine learning model to analyze various features and patterns associated with the input URL. Leveraging techniques such as URL analysis, HTML content examination, SSL certificate verification, and data inspection, the model generates predictions regarding the likelihood of the provided URL being either legitimate or phishing. The prediction page presents users with a clear indication of the predicted category (legitimate or phishing) alongside relevant confidence scores or probabilities, providing transparency into the model's certainty in its prediction. Additionally, users may receive explanations or visualizations highlighting key features contributing to the model's decision, empowering them to understand and trust the system's output. This intuitive interface not only assists users in identifying potentially malicious URLs but also fosters awareness and education regarding common indicators of phishing activity, ultimately contributing to enhanced online security practices.

7.6 Accuracy Graph

In the accuracy graph for phishing website detection using machine learning, each point represents the performance of the model in distinguishing between legitimate and phishing URLs across different thresholds or decision boundaries. The x-axis typically represents the threshold value, which determines the classification boundary: URLs with predicted probabilities above the threshold are classified as phishing, while those below are classified as legitimate. The y-axis represents the accuracy of the model at each threshold, indicating the proportion of correctly classified URLs out of the total. As the threshold varies, the accuracy of the model may fluctuate, reflecting the trade-off between correctly identifying phishing URLs (true positives) and incorrectly labeling legitimate URLs as phishing (false positives).

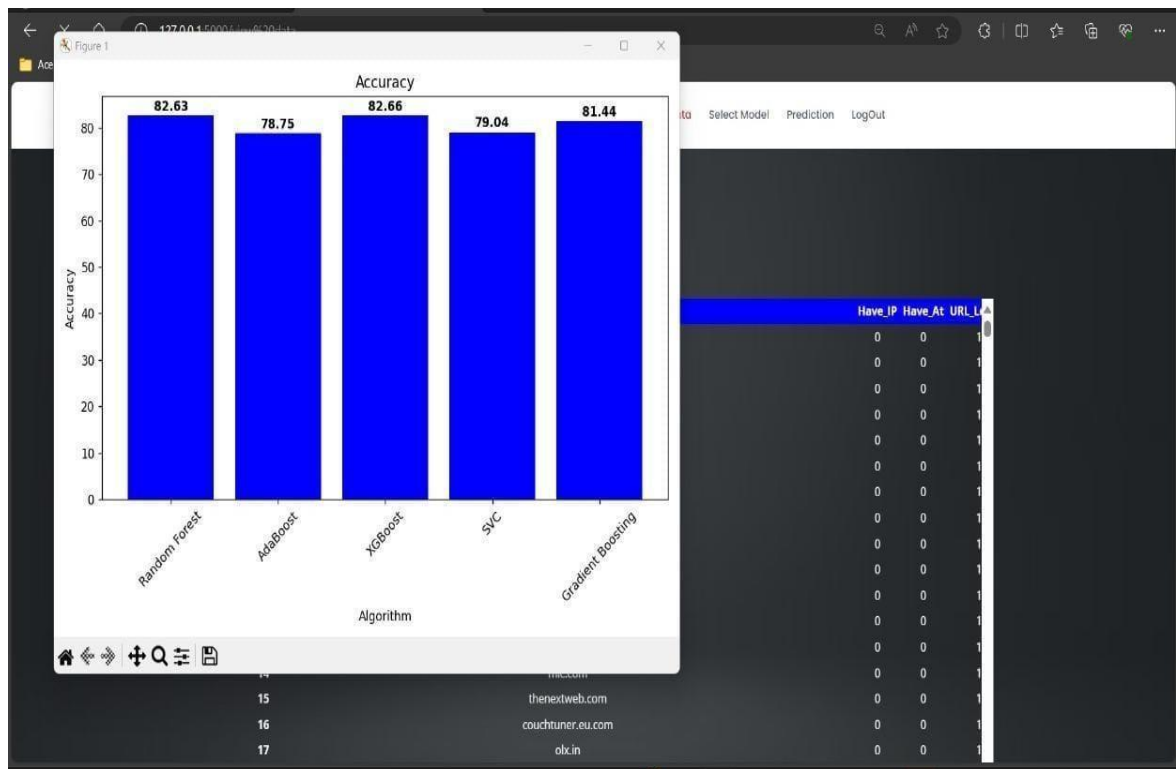


Fig 7.6: Accuracy Graph

7.7 Recall Graph

Recall is a metric that measures how often a machine learning model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset.

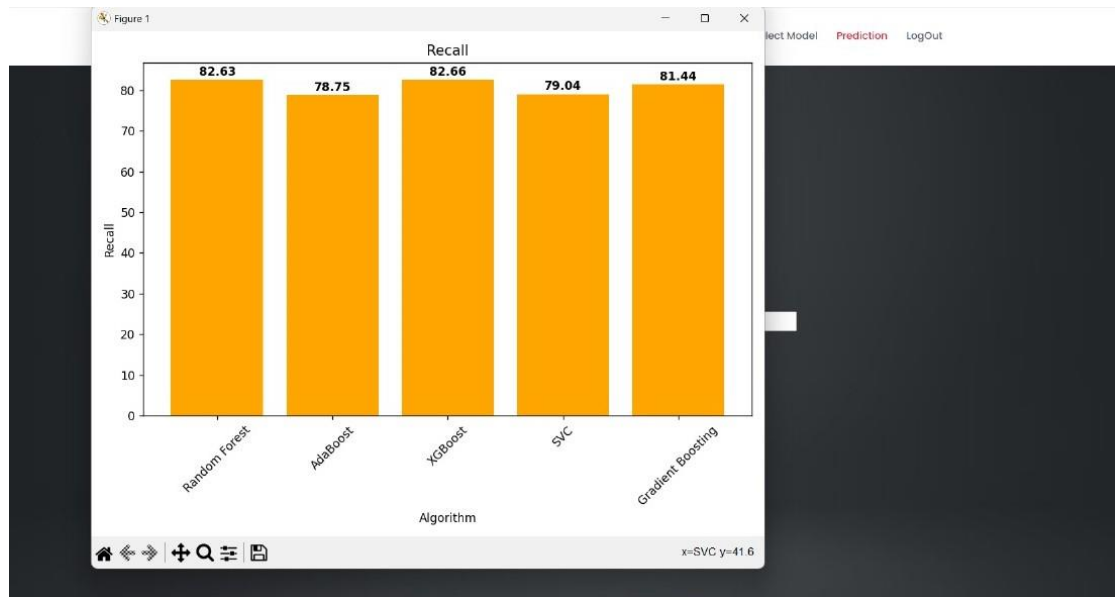


Fig 7.7: Recall Graph

CHAPTER 8

CONCLUSION

Phishing website detection using machine learning is a promising approach to combat the growing threat of online fraud. Machine learning algorithms can be trained to detect patterns in the behavior and characteristics of phishing websites, allowing them to identify and block suspicious sites before they can do harm. Recent studies have shown that machine learning algorithms can achieve high levels of accuracy in detecting phishing websites. These algorithms can analyze various features of a website, such as its URL structure, content, and user interface, to determine whether it is likely to be a phishing site. However, it is important to note that machine learning algorithms are not perfect and can sometimes produce false positives or false negatives. Additionally, phishing attackers are constantly evolving their tactics, so machine learning models must be continuously updated and refined to stay effective. Overall, phishing website detection using machine learning is a valuable tool in the fight against online fraud, but it should be used in conjunction with other security measures to provide the most comprehensive protection for users.

8.1 Limitations of the Study

While utilizing machine learning for phishing website detection offers numerous advantages, there are several limitations and challenges associated with this approach. Here are some of the key limitations of a study focused on Phishing Website Detection Using Machine Learning

Data Imbalance: The datasets used for training machine learning models may suffer from data imbalance, where legitimate websites significantly outnumber phishing websites. This can lead to biased models that perform well on normal websites but struggle with rare phishing cases.

Evolving Techniques: Phishing attacks constantly evolve, and machine learning models may struggle to keep up with new and sophisticated tactics employed by cybercriminals. They require regular updates and retraining to remain effective.

Generalization Issues: Machine learning models developed for phishing detection might not generalize well across different domains or industries. A model trained on

one dataset may not perform effectively on websites from different sectors, leading to a lack of versatility.

False Positives and Negatives: Even the most advanced machine learning models are not infallible. They may produce false positives (misclassifying legitimate websites as phishing sites) and false negatives (failing to identify actual phishing sites), which can be problematic for user trust and security.

Adversarial Attacks: Malicious attackers can deliberately craft phishing websites to bypass machine learning models by employing evasion techniques. Adversarial attacks can undermine the reliability of these models.

Resource Requirements: Training and deploying machine learning models for real-time phishing detection can be resource-intensive. It requires computational power and expertise, which may not be available to all organizations.

Interpretability: Some machine learning models, such as deep neural networks, can be challenging to interpret, making it difficult to understand why a particular decision was made. This lack of transparency can hinder trust in the system.

Privacy Concerns: Some machine learning approaches for phishing detection involve the analysis of user data, which can raise privacy concerns. Users may be uncomfortable with the collection and processing of their online behavior for security purposes.

Legitimate Website Changes: Legitimate websites often undergo updates and changes in their design and structure. Machine learning models may struggle to adapt to these changes, leading to false alarms or missed phishing attempts.

Limited Feature Sets: The effectiveness of machine learning models can be constrained by the feature sets used for training. If important features are missing or not adequately represented, the model's performance may suffer.

Lack of Ground Truth: Creating a reliable ground truth dataset for phishing websites can be challenging. It's often based on reported incidents or known phishing URLs, which may not cover all potential threats.

Human Verification: Machine learning models should ideally be complemented with human verification to handle edge cases and ensure the accuracy of results. This requires additional human resources.

8.2 Future Scope of Work

In future Hybrid Technology can be implemented to detect phishing websites more accurately, for which random forest algorithm of machine learning technology and blacklist method will be used. Website Blacklisting method will be added. Developing more sophisticated features for machine learning models, such as analyzing website content, behavior patterns, and user interactions to improve detection accuracy. Exploring the use of deep learning models like convolutional neural networks and RNNs for better pattern recognition in phishing websites. Enhancing the speed and efficiency of detection algorithms to identify phishing websites in real-time.

References

1. J. Shad and S. Sharma, (2018). “A Novel Machine Learning Approach to Detect Phishing Websites Jaypee Institute of Information Technology”, *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*.425–430.
2. Y. Sönmez, T. Tuncer, H. Gökal, and E. Avci, (2018). “Phishing web sites features classification based on extreme learning machine”, *Information Security and Digital Forensics and Security (ISDFS)*. 1–5.
3. T. Peng, I. Harris, and Y. Sawa, (2018). “Detecting Phishing Attacks Using Natural Language Processing and Machine Learning”, *Institute of Electrical and Electronics Engineers (IEEE)*. 300–301.
4. M. Karabatak and T. Mustafa, (2018). “Performance comparison of classifiers on reduced phishing website dataset”, *Information Security and Digital Forensics and Security (ISDFS)*. 1–5.
5. S. Parekh, D. Parikh, S. Kotak, and P. S. Sankhe, (2018). “A New Method for Detection of Phishing Websites: URL Detection”, *International Conference on Inventive Communication and Computational Technologies (ICICCT)*. 949–952.
6. K. Shima et al., (2018). “Classification of URL bitstreams using bag of bytes”, *Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 1–5.
7. A. Vazhayil, R. Vinayakumar, and K. Soman, (2018). “Comparative Study of the Detection of Malicious URLs Using Shallow and Deep Networks”, *International Conference on Computing, Communication and Networking Technologies, (ICCCNT)*. 1– 6.
8. W. Fadheel, M. Abusharkh, and I. Abdel-Qader, (2017). “On Feature Selection for the Prediction of Phishing Websites”, *Institute of Electrical and Electronics Engineers (IEEE)*. 871–876.
9. X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng, (2017). “Boosting the Phishing Detection Performance by Semantic Analysis”, *International Conference on New Trends in Computing Sciences (ICTCS)*.300-308.
10. L. MacHado and J. Gadge, (2018). “Phishing Sites Detection Based on C4.5 Decision Tree Algorithm”, *International Conference on Computing, Communication, Control and Automation, (ICCUBEA)*. 1–5.

11. Padmanaban A, (2023). “Detecting Phishing attacks using Natural Language Processing and Machine Learning”, *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*.32-38.
12. K. Shima, (2021). “Classification of URL bitstreams using Bag of Bytes”, *Institute of Electrical and Electronics Engineers (IEEE)*. 871–876.
13. Dyana R. Ibrahim, (2017). “Phishing Websites prediction using classification techniques”, *International Conference on New Trends in Computing Sciences (ICTCS)*.300-308.