

A  
Major Project Report  
On  
**Malicious URL Detection using Machine Learning**

*Submitted to CMREC, HYDERABAD*

*In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY IN  
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**

Submitted

By

**K.Sai Shivaji (218R1A6733)**

**B.VenuMadhav (218R1A6717)**

**G.VaibhavKumar (218R1A6724)**

**L.Nikitha (218R1A6737)**

Under the Esteemed guidance of

**Mrs. A.Shravani**

Assistant Professor, Department of CSE (Data Science)



**Department of Computer Science & Engineering (Data Science)**

**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)  
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

**2024-2025**

# CMR ENGINEERING COLLEGE

## UGC AUTONOMOUS

*(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501 401*

### **Department of Computer Science & Engineering (Data Science)**



### **CERTIFICATE**

This is to certify that the project entitled “**Malicious URL Detection using Machine Learning**” is a Bonafide work carried out by

<b>K.Sai Shivaji</b>	<b>(218R1A6733)</b>
<b>B.VenuMadhav</b>	<b>(218R1A6717)</b>
<b>G.VaibhavKumar</b>	<b>(218R1A6724)</b>
<b>L.Nikitha</b>	<b>(218R1A6737)</b>

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

#### **Internal Guide**

**Mrs. A.Shravani**

Assistant Professor  
CSE (Data Science), CMREC

#### **Major Project Coordinator**

**Mrs. G. Shruthi**

Assistant Professor  
CSE (Data Science),  
CMREC

#### **Head of the Department**

**Dr. M. Laxmaiah**

Professor & HOD  
CSE (Data Science),  
CMREC

#### **External Examiner**

## **DECLARATION**

This is to certify that the work reported in the present Major project entitled "**Malicious URL detection using Machine Learning**" is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

<b>K.SaiShivaji</b>	<b>218R1A6733</b>
<b>B.Venu Madhav</b>	<b>218R1A6717</b>
<b>G.VaibhavKumar</b>	<b>218R1A6724</b>
<b>L.Nikitha</b>	<b>218R1A6737</b>

## **ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, HOD, Professor **Department of CSE (Data Science), CMR Engineering College** for their constant support.

We are extremely thankful to **Mrs. A.Shravani**, Assistant Professor, Internal Guide, Department of CSE(DS), for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G. Shruthi**, Assistant Professor ,CSE(DS) Department ,Major Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

<b>K.SaiShivaji</b>	<b>218R1A6733</b>
<b>B.Venu Madhav</b>	<b>218R1A6717</b>
<b>G.VaibhavKumar</b>	<b>218R1A6724</b>
<b>L.Nikitha</b>	<b>218R1A6737</b>

## **ABSTRACT**

With the escalating risk of cyber fraud driven by sophisticated techniques like social engineering and phishing, the detection of malicious Uniform Resource Locators (URLs) has become imperative. This paper introduces a novel approach to detecting cyber fraud, specifically focusing on malicious URLs and text by leveraging machine learning techniques and big data analytics. Traditional methods such as signature-based detection suffer limitations in identifying new threats, prompting the need for more advanced methodologies. The proposed system utilizes a combination of machine learning algorithms, namely Support Vector Machine (SVM), Naive Bayes, Logistic Regression, Decision Tree Classifier, SGD Classifier, to classify URLs and text messages based on their behaviors and attributes. Unlike existing systems, this approach incorporates both static and dynamic characteristics of URLs, leading to a more comprehensive detection mechanism. The novelty lies in the introduction of new URL features specifically tailored for malicious URL and text detection. Advantages of the proposed system include its ability to harness the potential of these new features, enhancing the accuracy and efficiency of malicious URL and text detection. While These are showcased in this research, the framework remains flexible, allowing for the incorporation of other algorithms. Experimental results prove the efficiency of the proposed method in significantly improving malicious URL detection and Text detection making it a promising solution in combating cyber fraud.

# CONTENTS

<b>TITLE</b>	<b>PAGE NO</b>
<b>ABSTRACT</b>	<b>i</b>
<b>LIST OF FIGURES</b>	<b>ii</b>
<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Overview	2
1.2 Purpose of the project	2
1.3 Motivation	3
<b>2. LITERATURE SURVEY</b>	<b>4</b>
<b>3. METHODOLOGY</b>	<b>6</b>
3.1 System Architecture	7
3.2 Client Server	8
<b>4. TECHNOLOGY DESCRIPTION</b>	<b>9</b>
4.1 Hardware System Configuration	9
4.2 Software Requirements	9
<b>5. SYSTEM ANALYSIS</b>	<b>10</b>
5.1 Existing System	13
5.2 Proposed System	14
<b>6. SYSTEM DESIGN</b>	<b>15</b>
6.1 Data Flow Diagram	16
6.2 Uml Diagram	17
6.2.1 Use case Diagram	18
6.2.2 Class Diagram	19
6.3 Input and Output Design	20
6.3.1 Input Design	21
6.3.2 Output design	22
<b>7. SOFTWARE REQUIREMENTS</b>	<b>23</b>
7.1 Python	23
7.2 Machine Learning	34

<b>8. IMPLEMENTATION</b>	<b>42</b>
8.1 Source Code	42
8.2 Results	47
<b>9. TESTING</b>	<b>53</b>
9.1 System Testing	53
9.2 Module testing	53
9.3 Intergration Testing	53
9.4 Acceptance Testing	53
9.5 Regression Testing	53
<b>10. CONCLUSION AND FUTURE SCOPE</b>	<b>54</b>
10.1 Conclusion	56
10.2 Future scope	56
<b>11. REFERENCES</b>	<b>58</b>

## **LIST OF FIGURES**

<b>FIG.NO</b>	<b>DESCRIPTION</b>	<b>PAGE NO</b>
3.1	System Architecture for Malicious URL detection	7
6.1	Flow chart of end user	17
6.2	Flow chart of end server	18
6.3	Class Diagram for Malicious URL detection	19
6.2.1	Use case diagram for Malicious URL detection	2
8.2.1	Home Page	53
8.2.2	View All remote Users Page	53
8.2.3	View URL Datasets Trained and Tested results Page	54
8.2.4	Pie Chart and Line Chart Page	54
8.2.5	View URL Datasets Trained and Tested results Page	55



## **LIST OF SCREENSHOTS**

<b>FIG.NO</b>	<b>DESCRIPTION</b>	<b>PAGE NO</b>
8.2.1	Home Page	53
8.2.2	View All remote Users Page	53
8.2.3	View URL Datasets Trained and Tested results Page	54
8.2.4	Pie Chart and Line Chart Page	54
8.2.5	View URL Datasets Trained and Tested results Page	55

# **1. INTRODUCTION**

Uniform Resource Locator (URL) is used to refer to resources on the Internet. In [1], Sahoo et al. presented about the characteristics and two basic components of the URL as: protocol identifier, which indicates what protocol to use, and resource name, which specifies the IP address or the domain name where the resource is located. It can be seen that each URL has a specific structure and format. Attackers often try to change one or more components of the URL's structure to deceive users for spreading their malicious URL. Malicious URLs are known as links that adversely affect users. These URLs will redirect users to resources or pages on which attackers can execute codes on users' computers, redirect users to unwanted sites, malicious website, or other phishing site, or malware download. Malicious URLs can also be hidden in download links that are deemed safe and can spread quickly through file and message sharing in shared networks. Some attack techniques that use malicious URLs include [2, 3, 4]: Drive-by Download, Phishing and Social Engineering, and Spam.

According to statistics presented in [5], in 2019, the attacks using spreading malicious URL technique are ranked first among the 10 most common attack techniques. Especially, according to this statistic, the three main URL spreading techniques, which are malicious URLs, botnet URLs, and phishing URLs, increase in number of attacks as well as danger level.

From the statistics of the increase in the number of malicious URL distributions over the consecutive years, it is clear that there is a need to study and apply techniques or methods to detect and prevent these malicious URLs. In our research, machine learning algorithms are used to classify URLs based on the features and behaviors of URLs. The features are extracted from static and dynamic behaviors of URLs and are new to the literature. Those newly proposed features are the main contribution of the research. Machine learning algorithms are a part of the whole malicious URL detection system. Two supervised machine learning algorithms are used, Support vector machine (SVM) and Random forest (RF).

## **1.1 OVERVIEW**

The project delves into the intricate world of malicious URLs, which serve as a primary vector for cyber attacks. It begins by elucidating the fundamental structure of Uniform Resource Locators (URLs), highlighting their two basic components: the protocol identifier and the resource name. Emphasis is placed on how attackers exploit this structure by tampering with various components to deceive users. By altering elements such as the domain name or path, attackers can redirect users to nefarious destinations, including websites harboring malware or phishing pages. The project draws attention to the alarming statistics provided in various sources, particularly noting the prominence of malicious URL attacks in 2019. It underscores the escalation in both the frequency and severity of these attacks, with specific emphasis on three primary types of malicious URLs: malicious URLs, botnet URLs, and phishing URLs. This statistical analysis serves to underscore the pressing need for robust detection and prevention measures to counter the escalating threat posed by malicious URLs.

## **1.2 PURPOSE OF THE PROJECT**

The primary purpose of the project is to develop an effective system for detecting malicious URLs using machine learning algorithms. It aims to address the limitations of existing detection methods by proposing new features extracted from both static and dynamic behaviors of URLs. By leveraging machine learning, specifically Support Vector Machine (SVM) and Random Forest (RF) algorithms, the project seeks to classify URLs based on their attributes and behaviors accurately. Ultimately, the goal is to enhance cybersecurity measures and protect users from the detrimental effects of malicious URL attacks.

The motivation driving the project is rooted in the and user safety. With the proliferation of internet- connected devices and the increasing reliance on online platforms for various activities, the threat posed by malicious URLs has become more pronounced than ever before. The exponential growth in the number of attacks utilizing malicious URLs underscores the urgency of developing effective detection and prevention mechanisms.

### 1.3 MOTIVATION

The motivation driving the project is rooted in the profound implications of malicious URL attacks on cybersecurity and user safety. With the proliferation of internet-connected devices and the increasing reliance on online platforms for various activities, the threat posed by malicious URLs has become more pronounced than ever before. The exponential growth in the number of attacks utilizing malicious URLs underscores the urgency of developing effective detection and prevention mechanisms. Ultimately, the project's motivation is deeply rooted in a dedication to enhancing cybersecurity resilience and empowering users to navigate the online landscape with confidence. By addressing the pervasive threat of malicious URLs, the project endeavors to foster a safer and more secure digital environment for all stakeholders.

The primary purpose of the project is to develop an effective system for detecting malicious URLs using machine learning algorithms. It aims to address the limitations of existing detection methods by proposing new features extracted from both static and dynamic behaviors of URLs. By leveraging machine learning, specifically Support Vector Machine (SVM) and Random Forest (RF) algorithms, the project seeks to classify URLs based on their attributes and behaviors accurately. Ultimately, the goal is to enhance cybersecurity measures and protect users from the detrimental effects of malicious URL attacks.

## **2. LITERATURE SURVEY**

In this paper, we suggest a novel approach to malicious URL and text detection using ML techniques. Our method involves classifying URLs based on a comprehensive set of features and behaviors extracted from both static and dynamic analyses. These newly proposed features contribute significantly to the field of malicious URL detection.

### **Signature-based Malicious URL Detection**

Signature-based detection methods have been extensively studied in the realm of malicious URL detection. These methods rely on predefined sets of signatures or patterns associated with known malicious URLs. Whenever a user accesses a URL, the system queries a database to check if the URL matches any of the known signatures. If a match is found, the URL is flagged as malicious. However, the main drawback of this approach is its inability to detect new and evolving threats that do not match any of the predefined signatures [6, 7, 8].

### **Machine Learning-based Malicious URL Detection**

Machine learning techniques offer a promising avenue for detecting malicious URLs by analyzing URL behaviors. There are three main types of machine learning algorithms employed in this context supervised learning, unsupervised learning, and semi-supervised learning. Various studies have investigated the efficacy of these algorithms [1].

Dynamic URL analysis involves examining the actions performed by URLs in real-time, whereas static analysis focuses on characteristics such as lexical, content, host, and popularitybased features. These studies have utilized online learning algorithms to extract URL attributes based on static and dynamic behaviors [9, 10, 11].

Additionally, [12, 13] presented methods for detecting malicious URLs based on dynamic actions, extracting attributes such as character and semantic groups, abnormal behaviors in websites, host-based groups, and correlated groups. Despite the advancements in machine learning-based detection methods, there remain challenges related to algorithm selection and URL attribute extraction [12, 13].

In this paper, we suggest a novel approach to malicious URL and text detection using ML techniques. Our method involves classifying URLs based on a comprehensive set of features and behaviors extracted from both static and dynamic analyses. These newly proposed features contribute significantly to the field of malicious URL detection

Furthermore, our proposed system integrates big data technology to enhance detection capabilities by identifying abnormal URL and Text behaviors. The system comprises supervised machine learning algorithms, which have been chosen to demonstrate the effectiveness of the overall detection system.

The proposed algorithms leverage the usefulness of our newly introduced features, enhancing the accuracy of malicious URL detection. While these are showcased in our work, the system remains flexible, allowing for the implementation of alternative algorithms [1].

In conclusion, our proposed system offers an optimized and user-friendly solution for detecting malicious URLs, addressing limitations present in existing approaches. The experimental results demonstrate the efficacy of the proposed URL attributes and behaviors in significantly improving the detection of malicious URLs.

Author(s) & Year	Title	Methodology	Key Findings	Limitations
Ma et al. (2023)	"Beyond Blacklists: Learning to Detect Malicious Web Sites"	Used lexical and host-based features with supervised learning algorithms (SVM, Logistic Regression)	Machine learning significantly improved detection accuracy over blacklist-based approaches	Limited dataset size, no deep learning comparison
Zhang et al. (2023)	"Detecting Malicious Websites by Integrating Online Sources"	Integrated multiple online sources like Google Safe Browsing with machine learning techniques	Improved precision in detecting malicious URLs	Dependency on third-party sources may cause latency
Sahoo et al. (2022)	"Malicious URL Detection using Machine Learning: A Survey"	Compared feature extraction methods (lexical, host-based, content-based) and ML models (Random Forest, SVM)	Identified the most effective features for malicious URL detection	High computational cost for real-time analysis
Marchal et al. (2021)	"Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Solution"	Combined heuristic and ML-based techniques for phishing detection	Improved real-time phishing detection	High false-positive rates in some cases
Verma & Das (2020)	"Malicious URL Detection Using Deep Learning"	Used LSTMs and CNNs to analyze URL text	Achieved high accuracy in URL classification	Computationally expensive, requires large datasets
Aghila et al. (2021)	"Hybrid Approach for Malicious URL Detection Using Ensemble Learning"	Combined Random Forest, XGBoost, and CNN for classification	Improved overall accuracy and robustness	Increased training time and resource consumption
Xu et al. (2022)	"Real-time Malicious URL Detection with Lightweight Machine Learning Models"	Used lightweight ML models optimized for real-time detection	Achieved faster detection while maintaining good accuracy	Slight trade-off in accuracy for speed

Table 1 Literature Survey

### 3. METHODOLOGY

#### 3.1 System Architecture

Figure 3.1 illustrates the proposed system architecture. We evaluated our algorithms using synthetic, measured and real topology. The framework comprises of modules and risk modules.

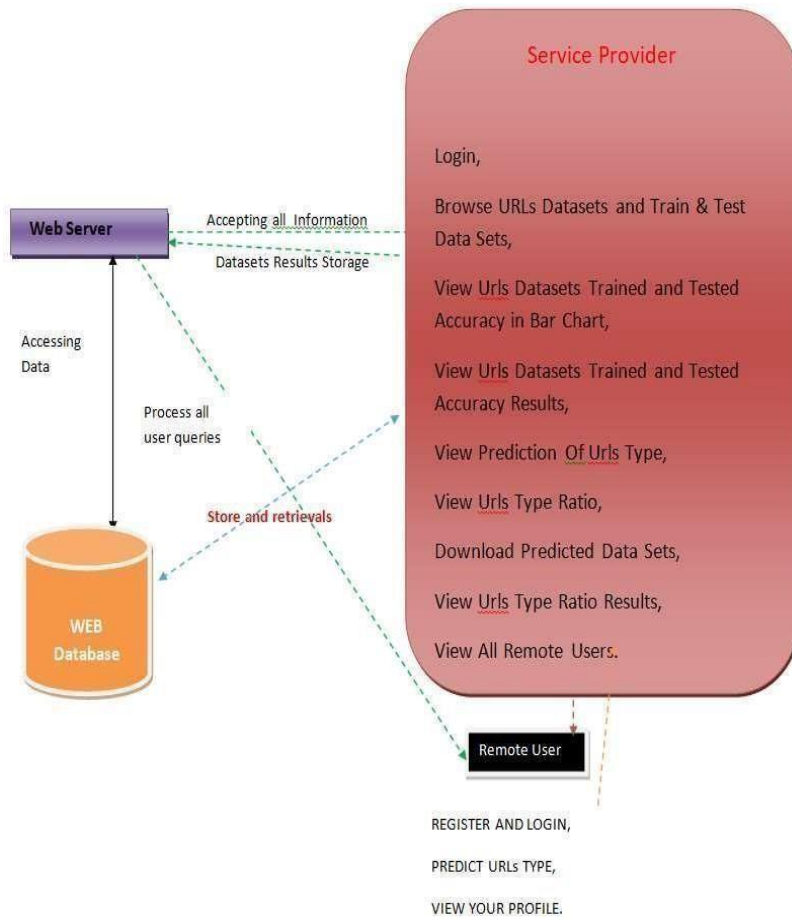


Fig.3.1: System Architecture



## **3.2 Client Server**

### **3.2.1 Overview**

With the varied topic in existence in the fields of computers, Client Server is one, which has generated more heat than light, and also more hype than reality. This technology has acquired a certain critical mass attention with its dedication conferences and magazines. Major computer vendors such as IBM and DEC, have declared that Client Servers is their main future market. A survey of DBMS magazine revealed that 76% of its readers were actively looking at the client server solution. The growth in the client server development tools from \$200 million in 1992 to more than \$1.2 billion in 1996.

Client server implementations are complex but the underlying concept is simple and powerful. A client is an application running with local resources but able to request the database and relate the services from separate remote server. The software mediating this client server interaction is often referred to as middleware.

The typical client either a PC or a Work Station connected through a network to a more powerful PC, Workstation, Midrange or Main Frames server usually capable of handling request from more than one client. However, with some configuration server may also act as client. A server may need to access other server in order to process the original client request.

The key client server idea is that client as user is essentially insulated from the physical location and formats of the data needs for their application. With the proper middleware, a client input from or report can transparently access and manipulate both local database on the client machine and remote databases on one or more servers. An added bonus is the client server opens the door to multi-vendor database access indulging heterogeneous table joins.

### **3.3.2. What is a Client Server?**

Two prominent systems in existence are client server and file server systems. It is essential to distinguish between client servers and file server systems. Both provide shared network access to data but the comparison dens there! The file server simply provides a remote disk drive that can be accessed by LAN applications on a file by file basis. The client server offers full relational database services such as SQL-Access, Record modifying, Insert, Delete with full relational integrity backup/ restore performance for high volume of transactions, etc. the client server middleware provides a flexible interface between client and server, who does what, when and to whom.

### **3.3.3. Why Client Server?**

Client server has evolved to solve a problem that has been around since the earliest days of computing: how best to distribute your computing, data generation and data storage resources in order to obtain efficient, cost effective departmental and enterprise wide data processing. During mainframe era choices were quite limited. A central machine housed both the CPU and DATA (cards, tapes, drums and later disks). Access to these resources was initially confined to batched runs that produced departmental reports at the appropriate intervals. A strong central information service department ruled the corporation. The role of the rest of the corporation limited to requesting new or more frequent reports and to provide hand written forms from which the central data banks were created and updated. The earliest client server solutions therefore could best be characterized as “Slave-Master”.

Time-sharing changed the picture. Remote terminal could view and even change the central data, subject to access permissions. And, as the central data banks evolved in to sophisticated relational database with non-programmer query languages, online users could formulate ad-hoc queries and produce local reports without adding to the MIS applications software backlog. However remote access was through dumb terminals, and the client server remained subordinate to the Slave\Master.

## **4. TECHNOLOGY DESCRIPTION**

### **4.1 Hardware System Configuration**

Processor	-	Intel i3 core
RAM	-	4 GB (min)
Hard Disk	-	500 GB

### **4.2 Software Requirements**

Operating System	-	Windows 7 or above Coding
Language	-	Java/J2EE (JSP, Servlet)
Front End	-	HTML, CSS.
Back End	-	J2EE.
Database	-	My SQL
Tool	-	NetBeans

## **5. SYSTEM ANALYSIS**

### **5.1 Existing System**

The comparison between signature-based and machine learning-based approaches to malicious URL and text detection underscores the nuanced considerations involved in developing effective cybersecurity measures. Signature-based detection, while efficient in identifying known threats by matching against predefined lists of malicious entities, faces significant limitations in adapting to emerging or evolving threats. This method's reliance on static signatures renders it ineffective against novel attack vectors that deviate from established patterns. Furthermore, the maintenance and upkeep of signature databases incur substantial resource costs and may still result in false negatives if malicious URLs or text do not align with existing signatures.

In contrast, machine learning-based detection offers greater adaptability and flexibility by leveraging various algorithms to analyze URL attributes and behaviors. This approach has the potential to learn from new data and adapt its detection capabilities to emerging threats. However, without a structured approach to algorithm selection and attribute extraction, the effectiveness of machine learning-based detection may be compromised. Careful consideration must be given to selecting algorithms that are well-suited to the task and extracting relevant features that capture the diverse aspects of malicious intent. Moreover, the challenge of addressing new threats not present in the training data remains a persistent concern.

Without adequate representation of emerging threats in the training data, machine learning models may struggle to generalize and detect them effectively. Therefore, ongoing monitoring and updating of the detection system are imperative to ensure its continued effectiveness in the face of evolving cyber threats. To mitigate these limitations, a structured approach to machine learning-based detection is essential. This includes comprehensive training datasets that encompass a wide range of potential threats, thoughtful algorithm selection based on performance metrics and task suitability.

This method relies on predefined lists of known malicious URLs and text patterns. It benefits from being able to quickly and accurately detect known threats, as it compares URLs and text against a database of known malicious entities. However, its reliance on predefined lists poses significant limitations. It struggles to detect new threats that do not match any signatures in the database, rendering it ineffective against emerging or evolving threats. Additionally, maintaining and updating the signature database can be resource-intensive, and there is a risk of false negatives if the malicious URLs or text do not match existing signatures.

## **Disadvantages of Existing System**

Concisely summarizing the disadvantages of the above implementations:

- Relies solely on predefined lists of known malicious URLs and text patterns, making it ineffective against new or emerging threats not present in the signature database.
- Signature-based systems lack adaptability to evolving attack techniques, as they cannot dynamically adjust to detect variations or mutations of known threats.
- Maintaining and updating the signature database requires significant resources and effort, as it must continuously incorporate new threat signatures and remove outdated ones.
- There is a risk of false negatives if a malicious URL or text does not match any existing signatures in the database, potentially leading to undetected security breaches.

Without a structured approach, selecting appropriate machine learning algorithms can be challenging, leading to suboptimal performance or inefficiency in detecting malicious URLs and text..

## 5.2 Proposed System

Malicious URL and text detection using machine learning represents a cutting-edge approach to cybersecurity, leveraging advanced algorithms to identify and mitigate online threats. One of the key strengths of this methodology lies in its ability to incorporate novel URL and text features extracted from both static and dynamic analyses, enhancing its detection capabilities. Static analysis involves examining the structural attributes of URLs and the content of text strings without executing them.

### Support Vector Machine (SVM)

SVM is employed in the project for its robustness in classifying URLs and text messages. It works by finding the optimal hyperplane that best separates different classes in the feature space. In this context, SVM is utilized to discern patterns and characteristics indicative of malicious URLs and text, aiding in the classification process. By leveraging SVM, the system can effectively identify complex relationships between features and classify URLs and text messages accurately.

### Naive Bayes

Naive Bayes is utilized due to its simplicity and effectiveness, especially in text classification tasks. Despite its assumption of feature independence, Naive Bayes performs well in practice and is particularly suitable for processing large volumes of text data. In this project, Naive Bayes is applied to classify URLs and text based on their respective attributes and behaviors. By calculating the probability of a URL or text message belonging to a certain class, Naive Bayes contributes to the overall classification accuracy of the system.

### Logistic Regression

Logistic Regression is employed as a linear classification algorithm capable of estimating the probability of a URL or text message belonging to a particular class. Its simplicity and interpretability make it a valuable component in the classification pipeline.

## **Decision Tree Classifier**

Decision Tree Classifier is utilized to create a tree-like structure that partitions the feature space based on the characteristics of URLs and text messages. By recursively splitting the data into subsets, decision trees effectively capture complex decision boundaries and patterns within the dataset. In this project, Decision Tree Classifier is employed to classify URLs and text messages by evaluating various features and their importance in distinguishing between malicious and benign entities.

## **SGD Classifier**

Stochastic Gradient Descent (SGD) Classifier is chosen for its efficiency and scalability, particularly in large-scale classification tasks. By iteratively optimizing the classification model using gradient descent, SGD Classifier can effectively handle high-dimensional feature spaces and large datasets. In the context of this project, SGD Classifier is utilized to classify URLs and text messages based on their behaviors and attributes, contributing to the overall accuracy and efficiency of the detection system.

## **Support Vector Machine (SVM)**

SVM is employed in the project for its robustness in classifying URLs and text messages. It works by finding the optimal hyperplane that best separates different classes in the feature space. In this context, SVM is utilized to discern patterns and characteristics indicative of malicious URLs and text, aiding in the classification process. By leveraging SVM, the system can effectively identify complex relationships between features and classify URLs and text messages accurately.

## **Advantages of Proposed System**

The proposed system has the following advantages:

1. By incorporating novel URL and text features extracted from both static and dynamic analyses, the system can achieve higher detection accuracy compared to signature-based or rule-based methods.
2. Machine learning algorithms can analyze large volumes of data and identify subtle patterns indicative of malicious intent, leading to fewer false positives compared to rule-based systems.
3. This scalability ensures that the system can effectively protect against threats across a wide range of use cases and network environments.

## **System Study**

### **Feasibility Study**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

### **Economic Feasibility**

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning and for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at any time. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.



Technical Feasibility is the assessment of the technical resources of the organization. The organization needs compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, Hypertext Markup Language (HTML), Structured Query Language (SQL) server and Web Logic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

### **Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

## 6. SYSTEM DESIGN

### 6.1 DATA FLOW DIAGRAM (DFD)

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The Data Flow Diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

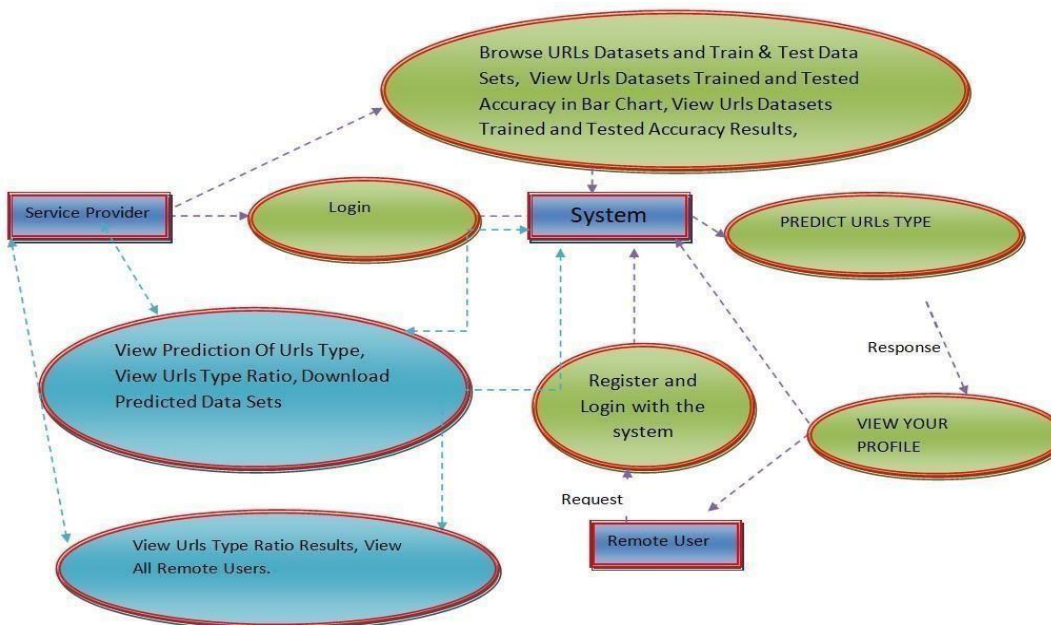


Fig 6.1 Data Flow Diagram

### Flow Chart1: Remote User

The provided flowchart illustrates the process for a Remote User accessing a malicious URL detection system. The process begins with the Start node, followed by a Login step. The system then verifies the login credentials through the Status decision node. If the credentials are incorrect, the user is prompted with a Username & Password Wrong message. If correct, the user is granted access to the system, allowing them to Register and Login, Predict URL Type, and View Profile. The flow concludes with the Logout option, ensuring a structured navigation path for the user.

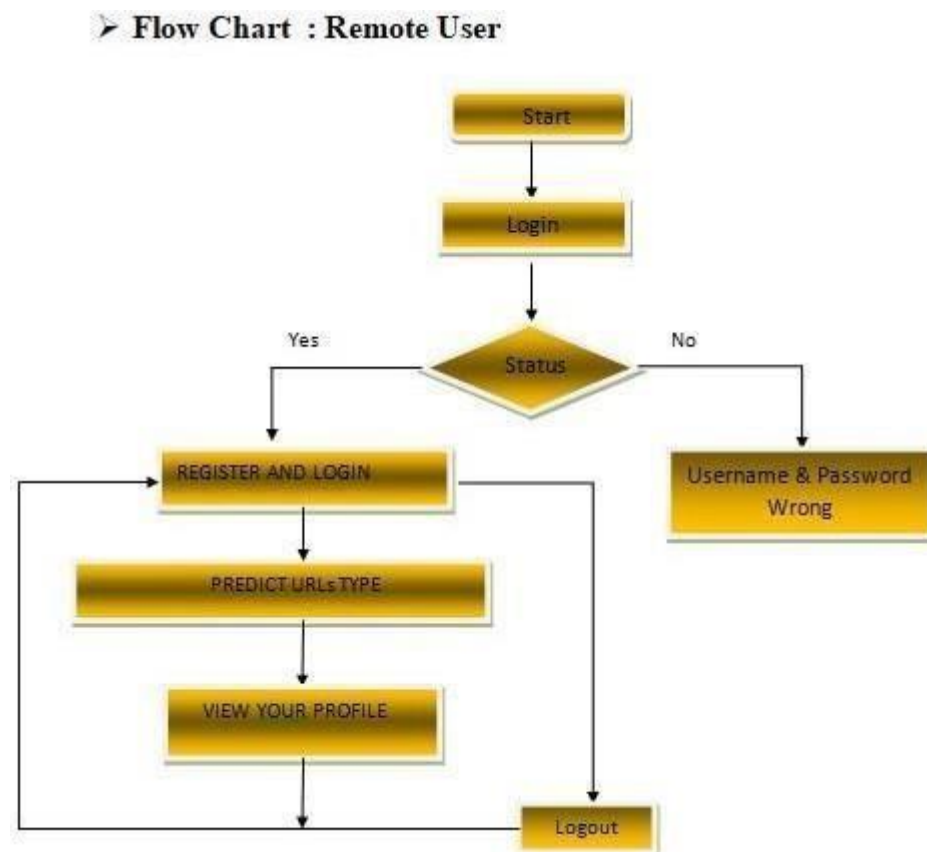


Fig. 6.2: Flow Chart of Remote User

## Flow Chart2: Service provider

The flowchart represents the Service Provider process in a malicious URL detection system. The process starts with the Login, and based on the Status decision, access is either granted or denied. If the credentials are incorrect, the user is shown a Username & Password Wrong message. If authenticated, the service provider can Browse URL Datasets, Train & Test Data Sets, and View Accuracy Results in bar charts. Additionally, they can View URL Predictions, Check URL Type Ratios, Download Predicted Data Sets, and View All Remote Users. The session ends with a Logout option, ensuring a structured workflow for managing URL data and predictions.

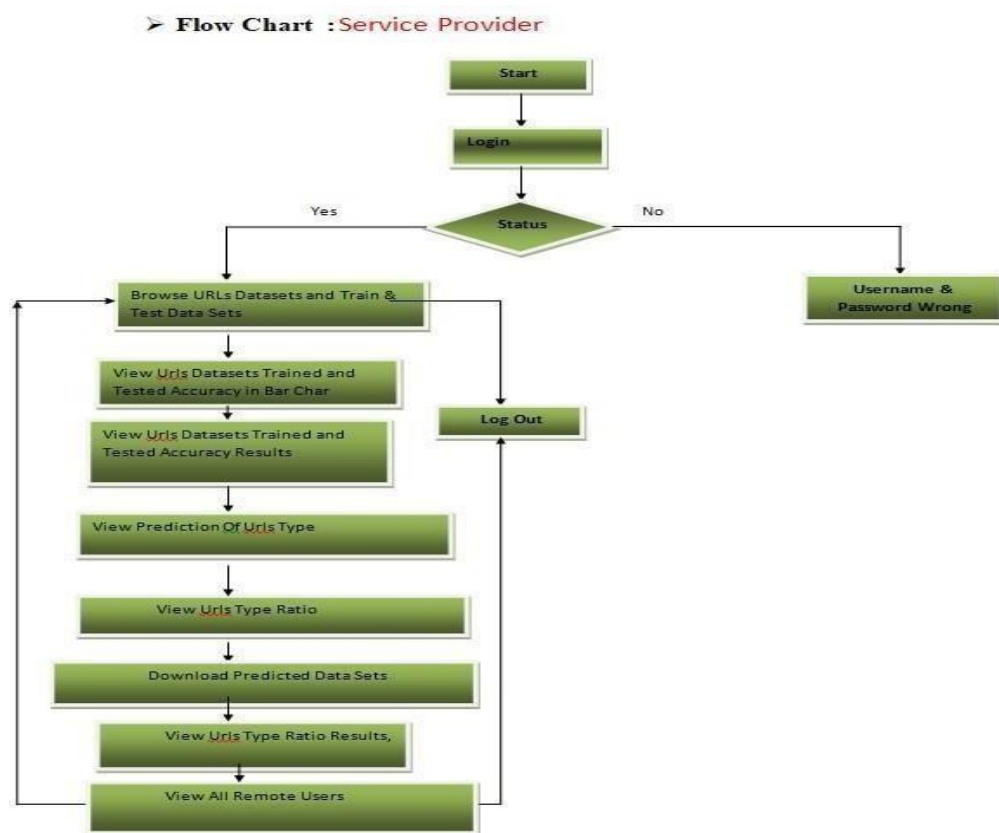


Fig. 6.3: Flow Chart of Service Provider

## 6.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling 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.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling 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:

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

## 6.2.1 Class diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

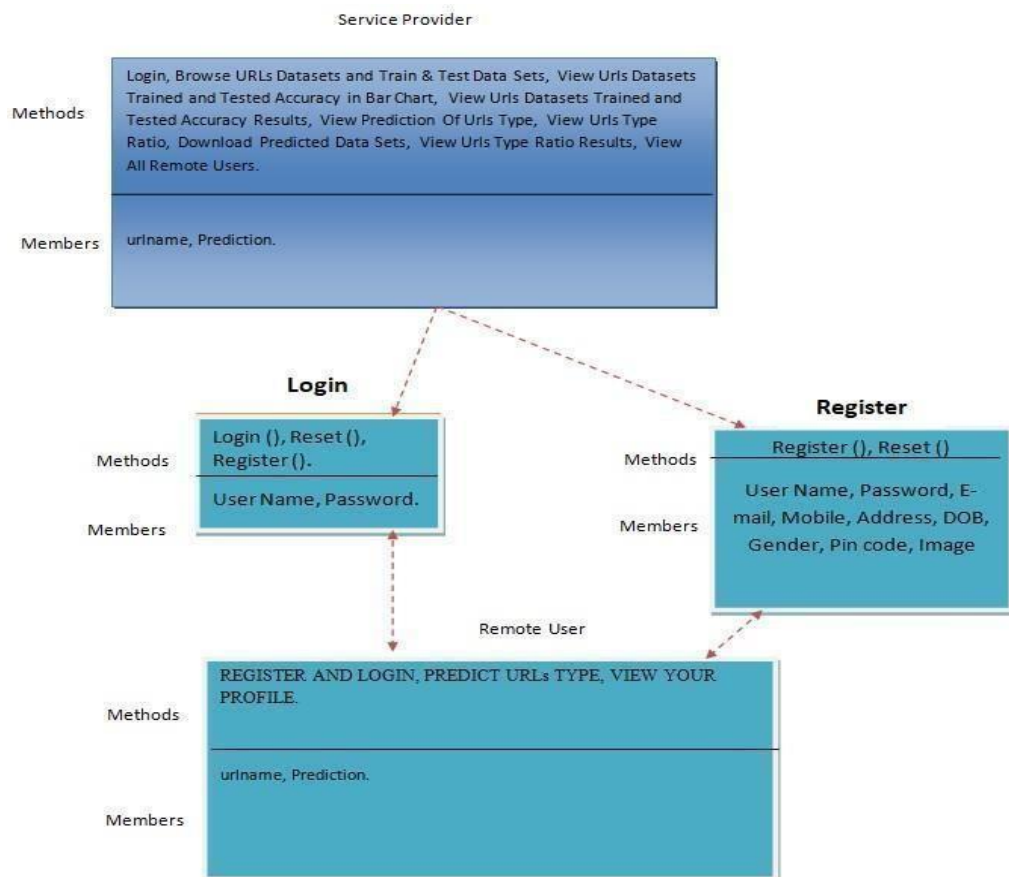


Fig 6.2.1 Class Diagram

## 6.2.2 Usecase Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

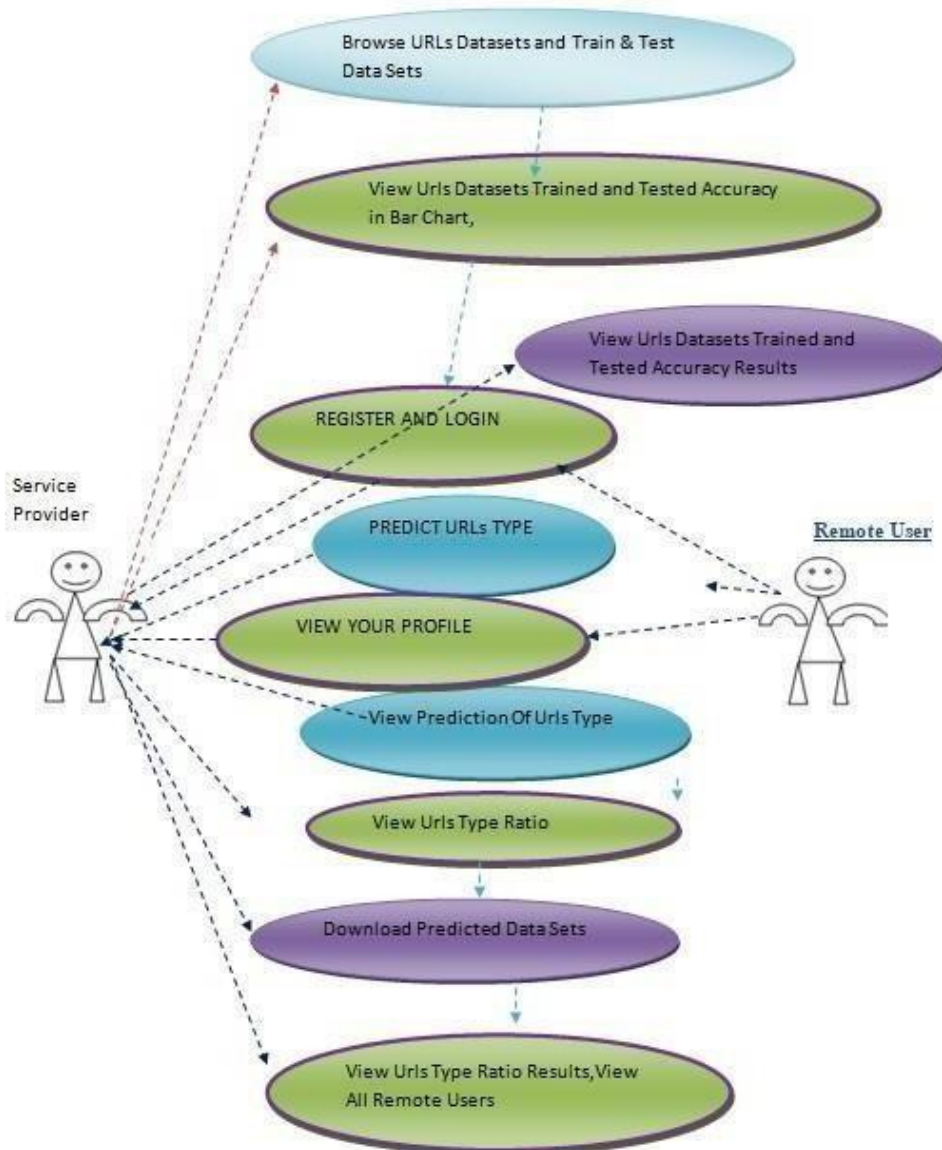


Fig. 6.2.2 : Use Case Diagram

## **6.3 INPUT AND OUTPUT DESIGN**

### **6.3.1 Input Design**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

### **OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.



### **6.3.2 Output Design**

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only. The application starts running when it is executed for the first time. The server has to be started and then the internet explorer is used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

## **7. SOFTWARE REQUIREMENTS**

### **7.1 What is Python?**

Python is currently the most widely used multi-purpose, high-level programming language. It supports both Object-Oriented and Procedural programming paradigms, making it versatile and easy to use. Python programs are generally smaller compared to those written in other languages like Java, as it requires less typing and enforces indentation, which enhances code readability. Due to its simplicity and efficiency, Python is widely adopted by major tech giants such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. It is also extensively used in fields like data science, artificial intelligence, and web development.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

### **Advantages of Python**

Let's see how Python dominates over other languages.

#### **Extensive Libraries**

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

### Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

### Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

### Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

### IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

### Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

### Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory.

### Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

### Portable

When you code your project in a language like C++, you may need to make some

changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

#### Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

## Advantages of Python Over Other Languages

#### Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

#### Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

#### Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

## Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution.

In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

## 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client- side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

## 3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

## 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

## 5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

# History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was

conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners<sup>1</sup>, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

## **Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e.int. long is int as well.

## Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers— even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

## Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## Modules Used in Project

### TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

### NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

### Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

### Matplotlib



Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots,

histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

## **Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious

repetition of code. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background.

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high- level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system. Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	Download	<a href="#">Release Notes</a>

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

### Files

Version	Operating System	Description	MD5 Sum	File Size	GPU
Clipped source tarball	Source release		68111673a5b2db4ae77b9ab01b7079be	23017663	3x5
XZ compressed source tarball	Source release		d33e4a8e66097051c3eca45ee3604803	17131432	3x5
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7563da71a4c3ba0ce08e6	34898416	3x5
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a457738f5ea930b2a1f	28882845	3x5
Windows help file	Windows		d83999573a2e9882ac58cade0b4f7rd2	8131761	3x5
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	980b3b3f628e3b0a0e32184a61728a2	7504391	3x5
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	a702b4b0a7f6d9d630c3a563e563e00	26885368	3x5
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	28c31c9088b6f73ae0e51a3b035184bd2	1362904	3x5
Windows x86 embeddable zip file	Windows		9fab38d198a1379fda94132574139d8	6741626	3x5
Windows x86 executable installer	Windows		33c862942a54446a3d8451476394789	25665848	3x5
Windows x86 web-based installer	Windows		1b670cfa5d117d82c30933ea371d87c	1324608	3x5

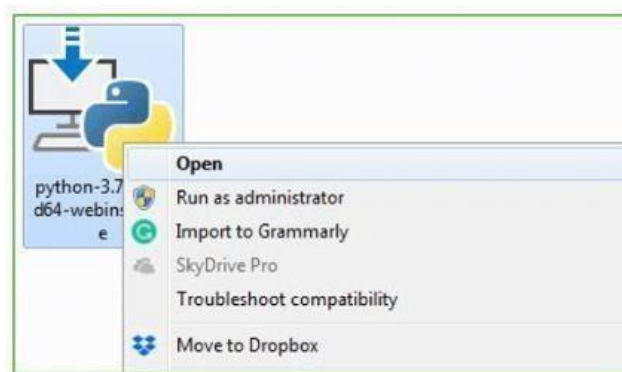
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

### Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



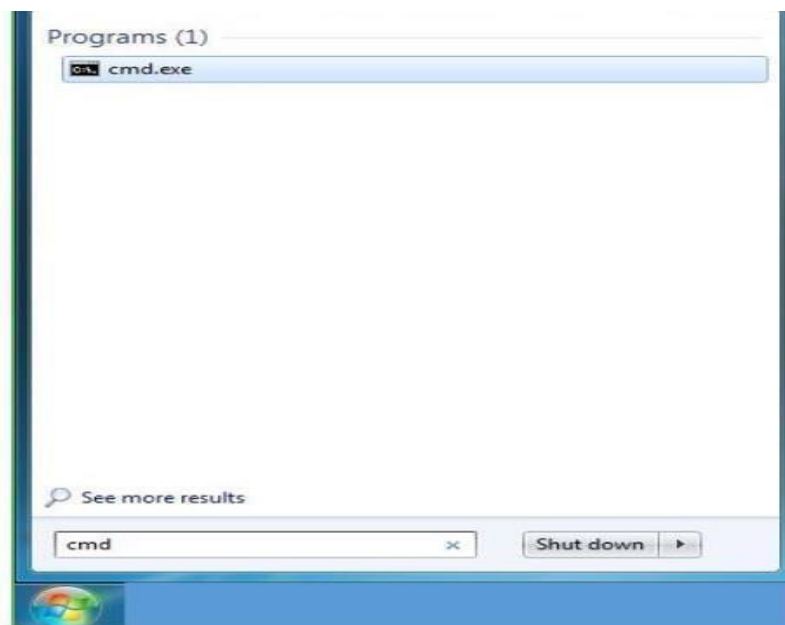
Step 3: Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

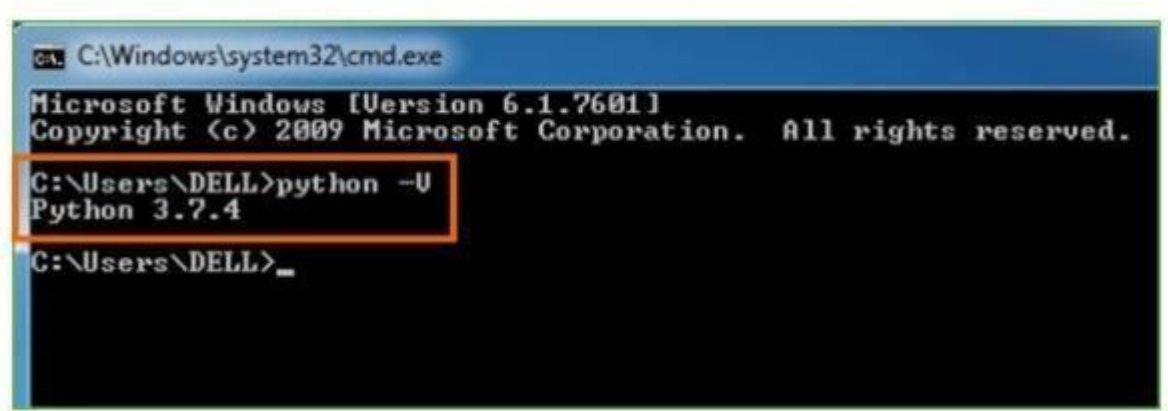
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

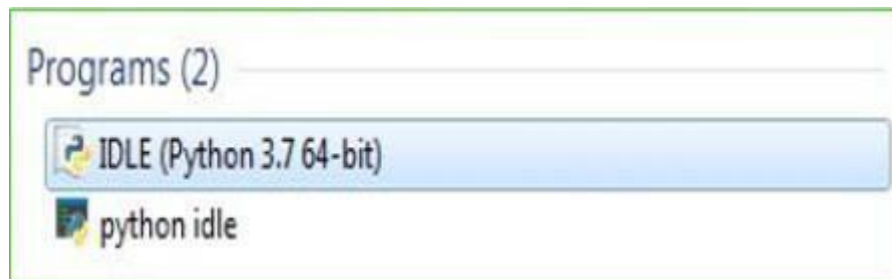
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE

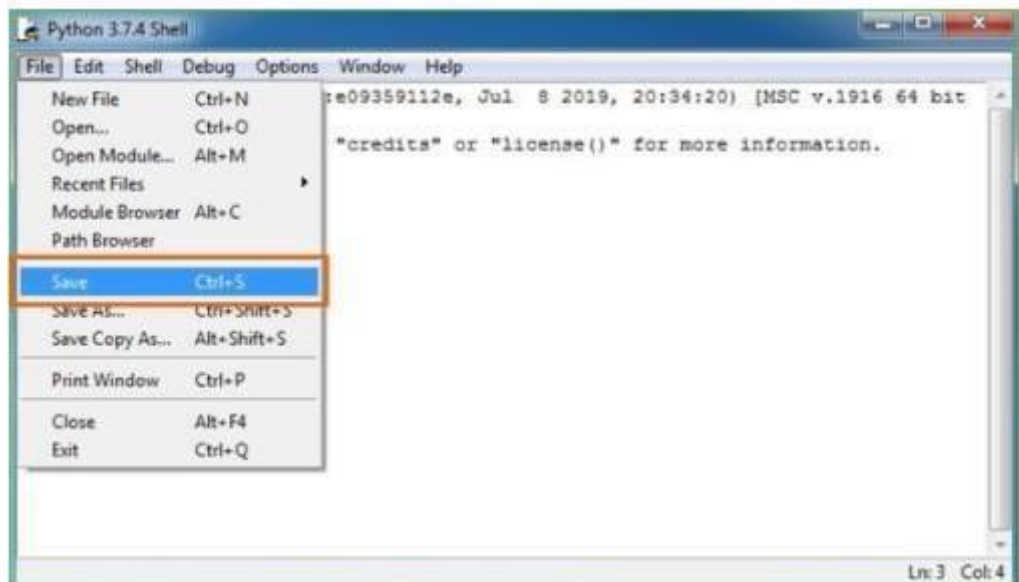
works Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



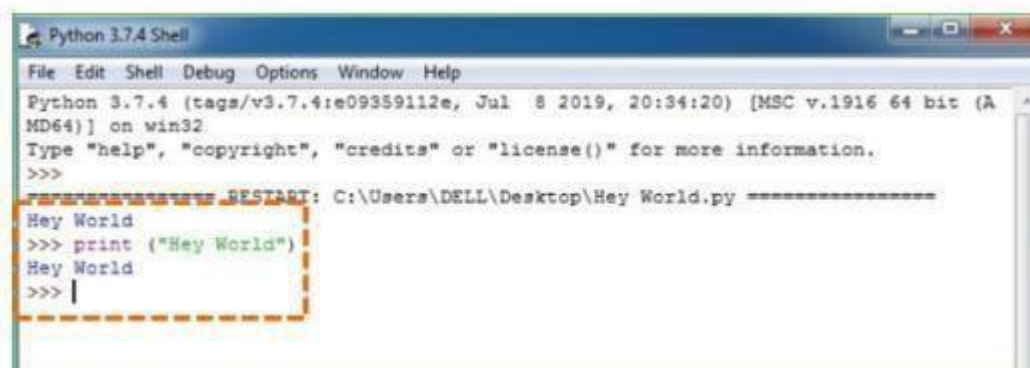
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

## 7.2 Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

### Categories of Machine Learning:

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.



## **Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

## **Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML

because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

## **Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

## **Advantages & Disadvantages of Machine learning**

### **Advantages:**

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it

means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own.

### 3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

### 4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

### 5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

## **Disadvantages:**

### 1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

### 2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

### 3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

## Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision-making knowledge from the supplied data. Decision tree can be generated from training sets.

The procedure for such generation based on the set of objects ( $S$ ), each belonging to one of the classes  $C_1, C_2, \dots, C_k$  is as follows:

Step 1. If all the objects in  $S$  belong to the same class, for example  $C_i$ , the decision tree for  $S$  consists of a leaf labeled with this class

Step 2. Otherwise, let  $T$  be some test with possible outcomes  $O_1, O_2, \dots, O_n$ . Each object in  $S$  has one outcome for  $T$  so the test partitions  $S$  into subsets  $S_1, S_2, \dots, S_n$  where each object in  $S_i$  has outcome  $O_i$  for  $T$ .  $T$  becomes the root of the decision tree and for each outcome  $O_i$  we build a subsidiary decision tree by invoking the same procedure recursively on the set  $S_i$ .

## Gradient boosting

It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision tree. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differential loss function.

## Logistic regression Classifiers

Logistic regression analysis examines the relationship between a categorical dependent variable and a set of independent variables. The term "logistic regression" is used when the dependent variable has only two possible values, such as 0 and 1 or Yes and No. Logistic regression is often preferred over discriminant analysis for analyzing categorical-response variables, as it does not assume a normal distribution of independent variables. This makes it a more versatile and widely applicable method. The logistic regression model computes both binary and multinomial logistic regression on numerical and categorical independent variables. It provides insights into the regression equation, goodness of fit, odds ratios, confidence limits, likelihood, and deviance. Additionally, it includes comprehensive residual analysis

with diagnostic reports and plots. The model can perform an independent variable subset selection search to identify the best regression model with the fewest independent variables. It also offers confidence intervals for predicted values and generates ROC curves to determine the optimal cutoff point for classification. Furthermore, it supports validation by automatically classifying rows that were not included in the analysis, ensuring robust and reliable results.

## **Naïve Bayes**

It is a supervised learning method based on a simplistic yet effective assumption: it considers that the presence or absence of a particular feature in a class is independent of other features. Despite this strong assumption, the Naïve Bayes classifier remains robust and efficient, often performing competitively with other supervised learning techniques. One explanation for its success is its representation bias, as it functions as a linear classifier similar to logistic regression, linear discriminant analysis, and support vector machines (SVM). However, it differs in its method of estimating classifier parameters, known as learning bias. Although widely used in research due to its simplicity, fast learning speed, and reasonable accuracy. To address this limitation, a new way of presenting the learning process results is introduced, making the classifier more understandable and easier to deploy. The theoretical aspects of Naïve Bayes are first explored before implementing the approach using the Tanagra dataset. The results are then compared with those from other linear classifiers such as logistic regression, linear discriminant analysis, and SVM, demonstrating consistency in performance.

## **Random Forest**

It is, also known as random decision forests, are ensemble learning methods used for classification, regression, and other tasks. They operate by constructing multiple decision trees during training, where for classification tasks, the class selected by the majority of trees is chosen, and for regression tasks, the average prediction of the trees is returned. While they generally outperform individual decision trees, their accuracy is often lower than gradient-boosted trees, although this can vary depending on data characteristics. The first random forest algorithm was introduced by Tin Kam Ho in 1995, utilizing the random subspace method, which was a way to implement the stochastic discrimination approach proposed by Eugene Kleinberg. Further

developments were made by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006. Their extension combined Breiman's "bagging" technique with random feature selection, first introduced by Ho and later independently by Amit and Geman. Due to their ability to provide reliable predictions with minimal configuration, random forests are widely used as "black-box" models in businesses across various industries.

## **Support Vector Machine (SVM)**

Classification tasks, discriminant machine learning techniques aim to find a function that accurately predicts labels for new instances based on an independent and identically distributed (iid) training dataset. Unlike generative machine learning approaches, which compute conditional probability distributions, discriminant classifiers assign data points to predefined classes without estimating underlying distributions. Although generative approaches are more powerful for outlier detection, discriminant methods require fewer computational resources and less training data, especially when dealing with high-dimensional feature spaces. SVM is a discriminant classification technique that solves convex optimization problems analytically, ensuring it always returns the same optimal hyperplane parameters. This characteristic differentiates SVM from genetic algorithms (GAs) and perceptrons, both of which are commonly used for classification but depend heavily on initialization and termination criteria. Unlike perceptrons and GA-based classifiers, which yield different models with each training session, SVM produces uniquely defined model parameters for a given training dataset when using a specific kernel transformation from input space to feature space.

## 8. IMPLEMENTATION

### 8.1 SOURCE CODE

```
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'malicious_url_detection.settings') application = get_asgi_application()

from django.contrib
import admin from
django.apps import
AppConfig class
ResearchSiteConfig(App
Config):
name = 'Service_Provider'
from django.db.models import
Count, Avg from
django.shortcuts import render,
redirect from django.db.models
import Count
from django.db.models
import Q import
datetime
import xlwt
from django.http import HttpResponseRedirect

import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

# Create your views here.
from Remote_User.models import
ClientRegister_Model,malicious_url_detection,detection_ratio,detection_accuracy
```

```
return render(request,'SProvider/serviceproviderlogin.html')
```

```
def View_Drug_Response_Ratio(request):
```

```
    detection_ratio.objects.all().delete()
```

```
    ratio = ""
```

```
    kword = 'Malicious URL'
```

```
    print(kword)
```

```
    obj = malicious_url_detection.objects.all().filter(Q(Prediction=kword))
```

```
    obj1 = malicious_url_detection.objects.all()
```

```
    count = obj.count();
```

```
    count1 = obj1.count();
```

```
    ratio = (count / count1) * 100
```

```
    if ratio != 0:
```

```
        detection_ratio.objects.create(names=kword, ratio=ratio)
```

```
    ratio12 = ""
```

```
    kword12 = 'Good URL'
```

```
    print(kword12)
```

```
    obj12 = malicious_url_detection.objects.all().filter(Q(Prediction=kword12))
```

```
    obj112 = malicious_url_detection.objects.all()
```

```
    count12 = obj12.count();
```

```
    count112 = obj112.count();
```

```
    ratio12 = (count12 / count112) * 100
```

```
    if ratio12 != 0:
```

```
        detection_ratio.objects.create(names=kword12, ratio=ratio12)
```

```
    obj = detection_ratio.objects.all()
```

```
    return render(request, 'SProvider/View_Drug_Response_Ratio.html', {'objs': obj})
```

```
def View_Remote_Users(request):
```

```
    obj=ClientRegister_Model.objects.all()
```

```
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})
```

```
def charts(request,chart_type):
```

```
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
```

```
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})
```



```

def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Prediction_Of_Drug_Response(request):
    obj =malicious_url_detection.objects.all()
    return render(request, 'SProvider/View_Prediction_Of_Drug_Response.html', {'list_objects':
obj})

def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})

def Download_Trained_DataSets(request):

    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="Predicted_Datasets.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True
    # writer = csv.writer(response)
    obj = malicious_url_detection.objects.all()
    data = obj # dummy method to fetch data.
    for my_row in data:
        row_num = row_num + 1

        ws.write(row_num, 0, my_row.urlname, font_style)
        ws.write(row_num, 1, my_row.Prediction, font_style)

```

```

wb.save(response)
return response

def train_model(request):
    detection_accuracy.objects.all().delete()

    df = pd.read_csv('Malicious_URLs.csv')

    mapping = {'good': 0, 'bad': 1}
    df['Results'] = df['Class'].map(mapping)

    cv = CountVectorizer()
    X = df['URLs']
    y = df['Results']

    print("URLs")
    print(X)
    print("Results")
    print(y)

    X = cv.fit_transform(X)

    models = []
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
    X_train.shape, X_test.shape, y_train.shape

    print(X_test)

    print("Naive Bayes")

    from sklearn.naive_bayes import MultinomialNB
    NB = MultinomialNB()
    NB.fit(X_train, y_train)
    predict_nb = NB.predict(X_test)
    naivebayes = accuracy_score(y_test, predict_nb) * 100

```

```

print(naivebayes)
print(confusion_matrix(y_test, predict_nb))
print(classification_report(y_test, predict_nb))
models.append(('naive_bayes', NB))
detection_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)

# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))
detection_accuracy.objects.create(names="SVM", ratio=svm_acc)

print("Logistic Regression")

from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
y_pred = reg.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('logistic', reg))
detection_accuracy.objects.create(names="Logistic Regression", ratio=accuracy_score(y_test,
y_pred) * 100)

print("Decision Tree Classifier")

```

```

Dtc=DecisionTreeC
lassifier()
dtc.fit(X_train,
y_train) dtcpredict
=
dtc.predict(X_test)
print("ACCURAC
Y")

    print(accuracy_score(y_test, dtcpredict) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, dtcpredict))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test,
dtcpredict))
    models.append(('DecisionTree
Classifier', dtc))
detection_accuracy.objects.create(names="DecisionTreeClassifier",
ratio=accuracy_score(y_test, dtcpredict) * 100)

print("SGD Classifier")
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(loss='hinge', penalty='l2',
random_state=0) sgd_clf.fit(X_train, y_train)
sgdpredict =
sgd_clf.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, sgdpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, sgdpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, sgdpredict))
models.append(('SGDClassifier', sgd_clf))
csv_format = 'Results.csv'
df.to_csv(csv_format,
index=False) df.to_markdown

```

## 8.2 Results



Fig.8.2.1: Home Page

**Description:** This is the Home page, in the Home page we have the faculty (data owner), student (data user), trusted authority and cloud. The Home page even consists of information related to project.

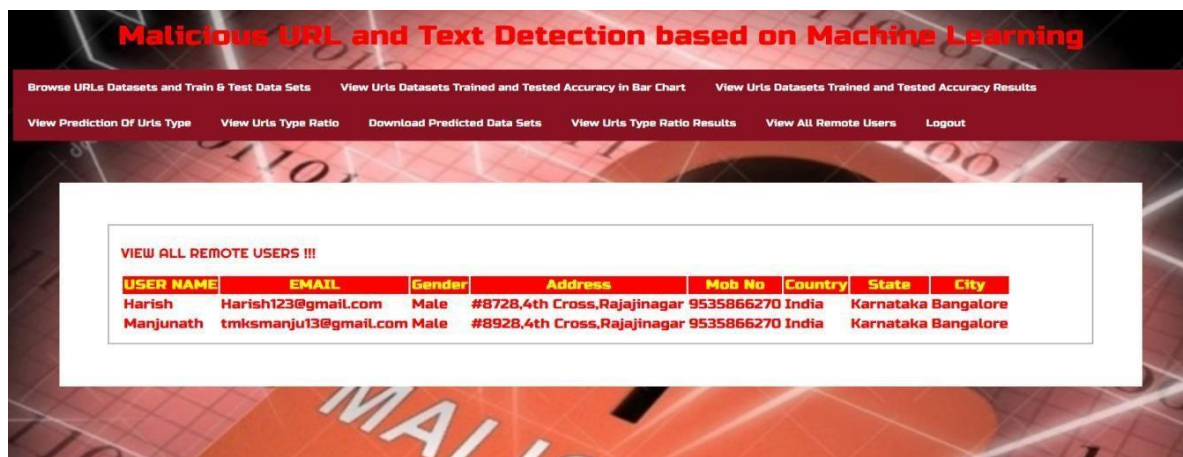


Fig.8.2.2 : View All remote Users Page

**Description:** In this the faculty (data owner) should register first to send the data to the data user and then the faculty can login to share the data.

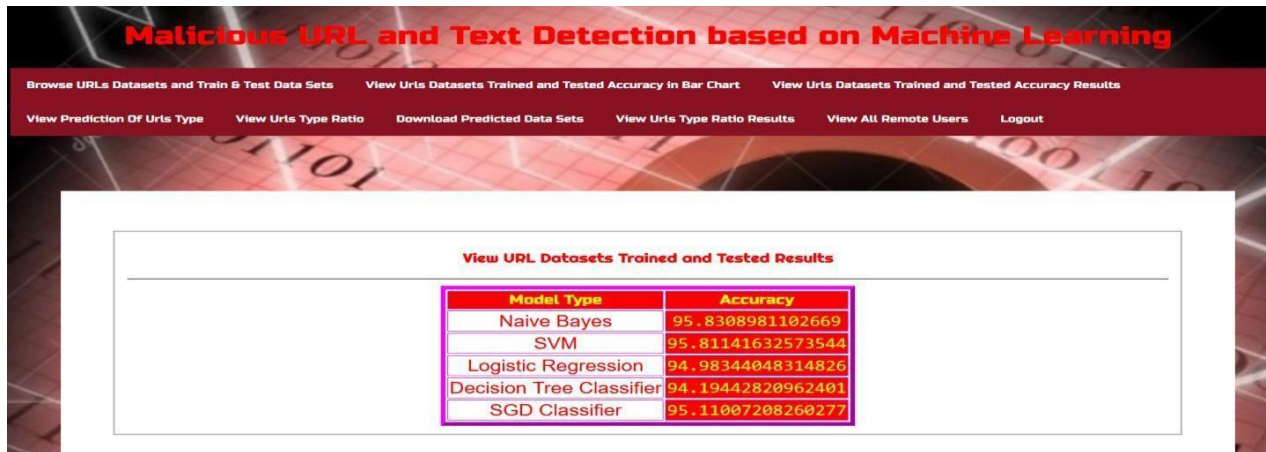


Fig.8.2.3 View URL Datasets Trained and Tested results Page

**Description:** In this the student (data user) should register first to receive the data from the data user and then the student can login to receive the data.

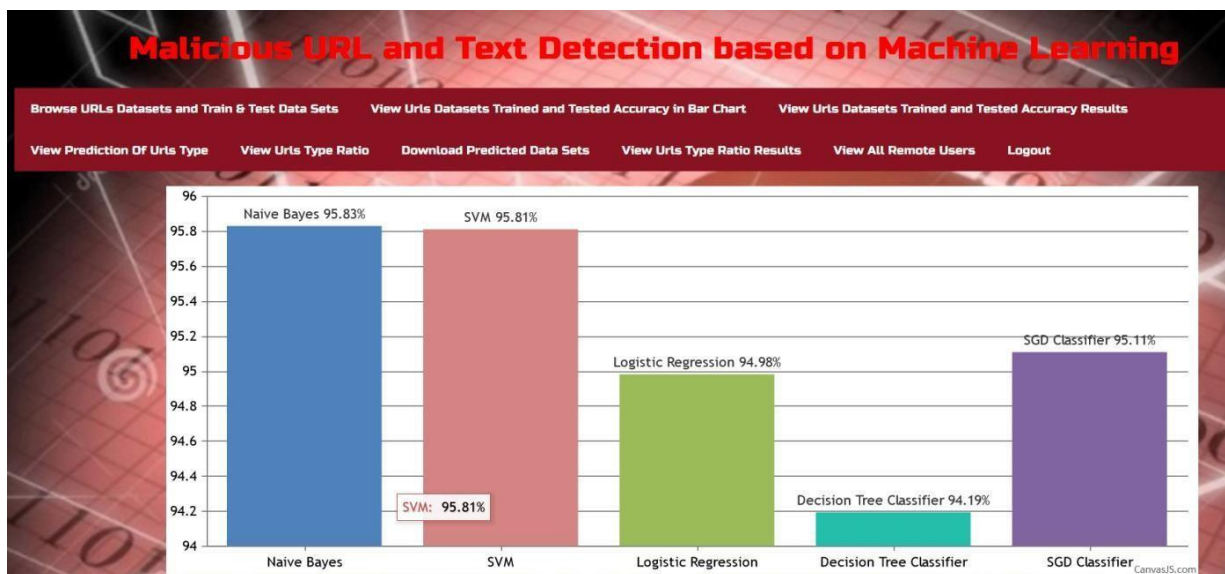


Fig.8.2.4 Algorithm wise Graphs Page

**Description:** We also create a login for the trusted authority and once we logged in, we can have various privileges like viewing the details.

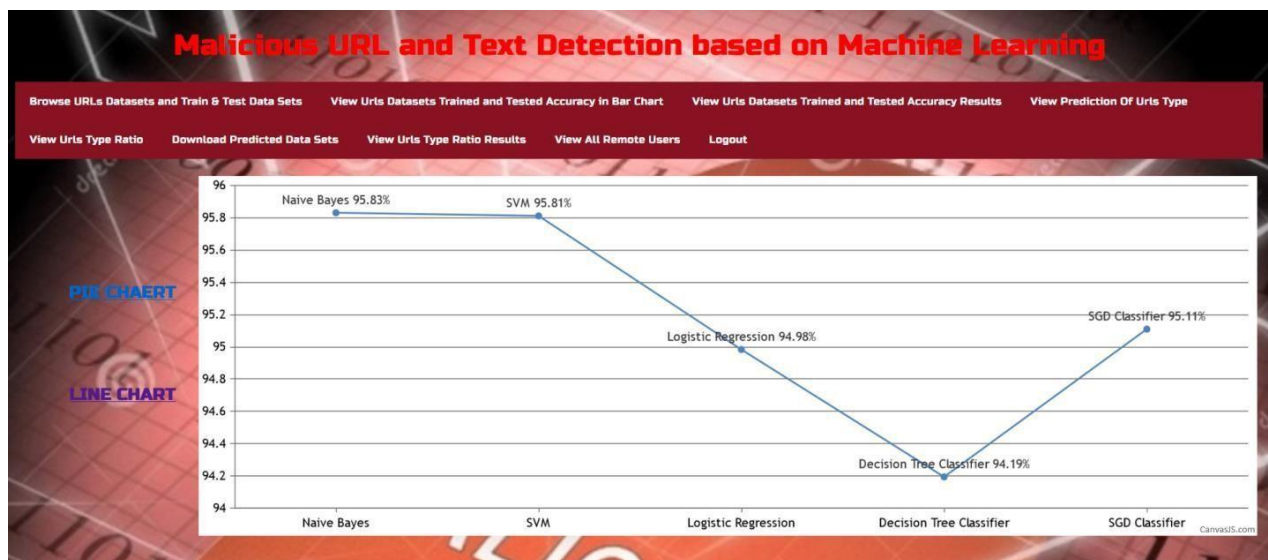


Fig.8.2.5 : Pie Chart and Line Chart Page

**Description:** We do create one more login for cloud and once we logged in we do have certain privileges of accessing files.

## **9. SYSTEM TESTING**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system properly function as a unit. The test data should be chosen such that it passed through all possible condition.

### **9.1 System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. The program was tested logically and pattern of execution of the program for a set of data are repeated.

### **9.2 Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

### **9.3 Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system



## **9.4 Accepting Testing**

When that user find no major problems with its accuracy, the system passes through a final acceptance test. This test confirms that the system meets the original goals, objectives and requirements established during analysis without actual execution which eliminates wastage.

## **9.5 Regression Testing**

Regression testing is performed to ensure that any modifications, enhancements, or bug fixes in the system do not negatively impact the existing functionality. It involves re-running previous test cases to confirm that the new changes do not introduce new errors. This type of testing is crucial when updating or expanding a system to maintain overall stability and performance.

## **10. CONCLUSION AND FUTURE SCOPE**

### **10.1 CONCLUSION**

In conclusion, the proposed malicious URL detection system, leveraging machine learning algorithms and big data technology, offers a promising solution to combat the increasing risk of cyber fraud. By analyzing both static and dynamic behaviors of URLs, our system introduces novel features for more accurate detection of malicious URLs. The experimental results demonstrate significant improvements in detection capability, indicating the effectiveness of the proposed approach. Moreover, the flexibility of incorporating various machine learning algorithms underscores the adaptability and scalability of the system. With ongoing advancements in cyber threats, the proposed system provides a robust and user-friendly solution for enhancing network security and protecting users from malicious online activities. Further research and development in this area hold the potential to refine and extend the capabilities of the system, making it even more resilient against evolving cyber threats.

### **10.2 FUTURE WORK**

Future work will focus on several key areas to enhance the proposed malicious URL detection system. Firstly, exploring additional machine learning algorithms beyond could further improve detection accuracy and robustness. Additionally, refining feature selection techniques and incorporating advanced deep learning models may enhance the system's ability to detect emerging threats effectively. Furthermore, integrating real-time data streams and leveraging cutting-edge big data analytics technologies could enhance the system's scalability and responsiveness. Finally, conducting extensive testing and validation in diverse network environments and continuously updating the system with new threat intelligence will be essential to ensure its effectiveness against evolving cyber

## **11. REFERENCES**

- [1] Thomas G. Dietterich. Ensemble Methods in Machine Learning. International Workshop on Multiple Classifier Systems, pp 1-15, Cagliari, Italy, 2000.
- [2] Leo Breiman.: Random Forests. Machine Learning 45 (1), pp. 5- 32,(2001).
- [3] C. Seifert, I. Welch, and P. Komisarczuk, “Identification of malicious web pages with static heuristics,” in Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian. IEEE, 2008, pp. 91–96.
- [4] S. Sinha, M. Bailey, and F. Jahanian, “Shades of grey: On the effectiveness of reputation-based “blacklists”,” in Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on. IEEE, 2008, pp. 57–64.
- [5] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” in Proceedings of Sixth Conference on Email and Anti-Spam (CEAS), 2009.
- [6] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying suspicious urls: an application of large-scale online learning,” in Proceedings of the 26th Annual International Conference on Machine Learning. ACM,2009, pp. 81–688.
- [7] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of drivebydownload attacks and malicious javascript code,” in Proceedings of the19th international conference on World wide web. ACM, 2010, pp. 281– 290.
- [8] S. Purkait, “Phishing counter measures and their effectiveness– literature review,” Information Management & Computer Security, vol. 20, no. 5,pp. 382–420, 2012.
- [9] M. Khonji, Y. Iraqi, and A. Jones, “Phishing detection: a literature survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.
- [10] B. Eshete, A. Villafiorita, and K. Weldemariam, “Binspect: Holistic analysis and detection of malicious web pages,” in Security and Privacyin Communication Networks. Springer, 2013, pp. 149–166.
- [11] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, “Detection of malicious web pages using system calls sequences,” in Availability, Reliability, and Security in Information Systems. Springer, 2014, pp.226-23.
- [12] R. Heartfield and G.

Loukas, “A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks,” ACM Computing Surveys (CSUR), vol. 48, no. 3, p. 37, 2015.

[13] D. Sahoo, C. Liu, S.C.H. Hoi, “Malicious URL Detection using Machine Learning: A Survey”. CoRR, abs/1701.07179, 2017.

[14] Internet Security Threat Report (ISTR) 2019–Symantec. <https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf> [Last accessed 10/2019].

[15] Developer Information. [https://www.phishtank.com/developer\\_info.php](https://www.phishtank.com/developer_info.php). [Last accessed 11/2019].

[16] URL haust Database Dump. <https://urlhaus.abuse.ch/downloads/csv/>. [Ngày truy nhập 11/2019].

[17] URLhaus Database Dump. <https://urlhaus.abuse.ch/downloads/csv/>. [Ngày truy nhập 11/2019].

[18] Dataset URL. [http://downloads.majestic.com/majestic\\_million.csv](http://downloads.majestic.com/majestic_million.csv). [Last accessed 10/2019].

[19] Malicious\_n\_Non-MaliciousURL. <https://www.kaggle.com/antonyj453/urldataset#data.csv>. [Last accessed 11/2019].