**Phish Catcher: Client-Side Defence Against Web Spoofing Attacks Using Machine Learning**

**1.INTRODUCTION:**

In Oct 2022,1 the members/users of the National Institute for Research in Digital Science and Technology (Inria) in France received an email in French asking the users to confirm their webmail account with the direct link https://www.educationonline.nl/Cliquez.ici.cas.inria.fr.cas.login/login.html. When clicked on this link, it takes to a fake but appearing genuine the associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry . 1An email, warning the users of Coq-club Inria https://www.inria.fr/en about this phishing attack, was received on Oct 10, 2022 central authentication login page of Inria. As this fake login page resembles the real login page of Inria from https://cas.inria.fr/cas/login?service=, users will mistakenly enter username and password of the Inria to a fake website which the attacker can later submit to the real Inria login page. This is a phishing attack on the Inria and users/members registered with Inria. The real and fake login pages of Inria are given in the Figures 1. Both of the web pages are exactly the same and it is easy for the users to fall victim of this phishing attack. We have tested our tool Phish Catcher against this and few other attacks as detailed in the Section V. VOLUME 11, 2023 This work is licensed under a Creative Commons Attribution-Non-commercial-No Derivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/ 61249 M. Ahmed et al.: Phish Catcher: Client-Side Défense Against Web Spoofing Attacks FIGURE 1. Phishing attack on Inria. With the tremendous advancement in modern technologies, there has been a great escalation in the online world, such as e-commerce, online banking, distant learning, e-health and e-governance. Since social networking applications, such as Facebook and Twitter, are performing leading role in the globalization of the modern era, billions of users have adopted this increasing trend. Numerous websites provide the web-users with an opportunity to create an account for a customized experience. To obtain online specialized services from the web-sites, users are required to create a personalized account. Conventionally, users are exposed to login web pages for this purpose where they have to set up an account by creating and registering an identification (e.g., username) and secret (e.g., password). Next time, when the user needs to access the remote resource or service, she/he sends a web requests and receives a login form for submitting the identification along with the secret. At this point, the users’ privacy is at high risk in terms of identity theft and confidential information. A phishing attack scenario, as described in Figure 2, begins with receiving an email with a link to malicious website. The email message might contain text convincing or luring the user to click and follow the pointer. When the unsuspecting user clicks and opens the web page, it appears genuine as the honest website where the user has an account. After the victim user enters his/her secret information, such as the username and password and presses the submit/login button, they are sent to the attacker. The attacker who sat up the phishing attack receives the secret credentials and logins to the legitimate website upon submitting the credentials to it. Identity theft, online frauds and scams have immensely increased since the advent of web spoofing or phishing attacks. Web spoofing or phishing is a type of cybercrime in which a malicious intruder tries to steal valuable data from the user. Attackers have adopted many phishing and web spoofing techniques to threaten online systems. Initially, web spoofing was used for identity theft but now attackers are using it to steal sensitive information related to national FIGURE 2. A typical phishing attack. security, intellectual property and organizational secrets. Current era’s phishing attacks have already been entered into a new evolutionary dimension including, but not limited to, QR code phishing, spoofing application for mobile and spear phishing etc. Such attacks and scam approaches may circumvent the protections such as firewalls, digital certificates, encryption software and other mechanisms like the two factor authentication. Numerous companies are using such two-factor authentication systems to avoid monetary scams and identity theft. Sadly, the advanced scam approaches have made all these systems vulnerable. 61250 VOLUME 11, 2023 M. Ahmed et al.: Phish Catcher: Client-Side Defense Against Web Spoofing Attacks To deceive the victims, the attackers normally include logos, either by storing copies or adding links to logos, from the honest site onto their spoof sites to imitate their appearance. In addition to logos, the attacker may also include HTML from the honest site and make some necessary changes. The phishing attack vectors used by the attackers for tricking the users include email, trojan horse, key loggers and manin-the-middle proxies. The favourite attack targets of the attackers are online banking sites, third party payment systems (the most targeted industry sector) and e-commerce sites. As the phisers target the non-cryptographic components, the cryptographic security protocols SSL/TLS do not provide a complete solution. To depend against spoofing attacks, these protocols must be complemented with additional protection mechanisms. These mechanisms may be enforced at the server-side or client-side or both. The server-side solutions, requires changes to the websites which is a tedious job and is often ignored by most of the developers. The client side solutions, on the other hand, provide protection to users without the server support. Server-side solutions may be effective in identifying spoofed site, however, the focus of this paper is on client-side solutions. Most of the anti-spoofing tools are based on either the third party certification, password or URLs. Anti-spoofing tools are sometimes categorized as stateful or stateless. They may also be classified based on the automatic phishing detection mechanism used: blacklists and heuristics. Tools that rely on black/white lists generate almost zero false positives (accuracy) and can recognize almost 90% of the phishing sites, however, they miss zero-day attacks. Furthermore, black-listing methodologies come with several drawbacks as they cannot control the changing domain and new attacks and can easily be fooled by the spam URLs. To capture phish sites not included in the black lists, the heuristic-based techniques have been very encouraging. The heuristic (content) based tool such as CANTINA and Spoof Catch can identify 90% phishing sites with 1% false positives. The latency of the tool Spoof Catch is in the order of seconds and it further increases with passage of time. While the stateful anti-phish techniques are good in accuracy, they quickly fill the local storage and the performance degrades with passage of time. In Spoof Catch, the visual similarity is initially compared with few login page images, but as the user browse further websites, the number of login page images increases in the local storage. In addition, this increases the time to compare the image of a received login page with every login image in the storage. Following this line of research, we design and develop a stateless anti-phish tool based on the Machine Learning (ML) technique. From the last decade, many renowned researchers have proposed machine learning techniques for the detection of malicious URLs to avoid any kind of scam in future. Many sets of URLs are treated as training data in the ML approaches. On the basis of the statistical properties obtained by the training sets, it is proposed that whether the requested URL is a scam or scam free. Training data is the primary concern for the URL identification using ML. Once training data is obtained then it is further processed to obtain a mathematical model. The primary concern is to collect the features from the training data because simple strings may not help to predict the status of the URL under test. At final stage, an actual model is obtained through predicted model from the training data. Machine learning techniques, such as Naïve Bayes, Support Vector Machines (SVM) and Logistic Regression (LR), are a few algorithms being used for this purpose by many scholars but there are several issues which make them vulnerable. In this paper, we propose and develop a stateless client-side tool, dubbed as Phish Catcher, to protect against web spoofing attacks. The Phish Catcher, a Google Chrome extension, is based on machine learning techniques and implements the random forest algorithm to classify whether or not a login web page .

* 1. **Objective of the project:**

Cyber security confronts a tremendous challenge of maintaining the confidentiality and integrity of user’s private information such as password and PIN code. Billions of users are exposed daily to fake login pages requesting secret information. There are many ways to trick a user to visit a web page such as, phishing mails, tempting advertisements, click-jacking, malware, SQL injection, session hijacking, man-in-the-middle, denial of service and cross-site scripting attacks. Web spoofing or phishing is an electronic trick in which the attacker constructs a malicious copy of a legitimate web page and request users’ private information such as password. To counter such exploits, researchers have proposed several security strategies but they face latency and accuracy issues. To overcome such issues, we propose and develop client-side defence mechanism based on machine learning techniques to detect spoofed web pages and protect users from phishing attacks. As a proof of concept, a Google Chrome extension dubbed as Phish Catcher, is developed that implements our machine learning algorithm that classifies a URL as suspicious or trustful. The algorithm takes four different types of web features as input and then random forest classifier decides whether a login web page is spoofed or not. To assess the accuracy and precision of the extension, multiple experiments were carried on real web applications. The experimental results show remarkable accuracy of 98.5% and precision as 98.5% from the trials performed on 400 classified phished and 400 legitimate URLs. Furthermore, to measure the latency of our tool, we performed experiments over forty phished URLs. The average recorded response time of Phish Catcher was just 62.5 milliseconds.

**2. LITERATURE SURVEY:**

**Spoof Catch: A client-side protection tool against phishing attacks**

To protect against web spoofing attacks, most ant phishing solutions in the literature either escape certain attack patterns or are based on complex sets of parameters to identify phishing attacks or suffer from both. In this article, we propose that phishing attacks can be prevented by simply relying on an overall visual appearance of the web page that the user sees. To realize our claim, we propose a client-side protection mechanism based on visual similarity of web pages and implement our mechanism as a browser extension, dubbed Spoof Catch. For similarity comparison between genuine and phished web pages, four similarity algorithms have been implemented and integrated in the extension. To evaluate the solution, large scale and extensive experiments have been conducted that demonstrate spoof Catch can capture all phishing attacks with acceptable overhead.

**A framework for detection and measurement of phishing attacks**

Phishing is form of identity theft that combines social engineering techniques and sophisticated attack vectors to harvest financial information from unsuspecting consumers. Often a phisher tries to lure her victim into clicking a URL pointing to a rogue page. In this paper, we focus on studying the structure of URLs employed in various phishing attacks. We find that it is often possible to tell whether or not a URL belongs to a phishing attack without requiring any knowledge of the corresponding page data. We describe several features that can be used to distinguish a phishing URL from a benign one. These features are used to model a logistic regression filter that is efficient and has a high accuracy. We use this filter to perform thorough measurements on several million URLs and quantify the prevalence of phishing on the Internet today

**Effective protection against phishing and web spoofing**

Phishing and Web spoofing have proliferated and become a major nuisance on the Internet. The attacks are difficult to protect against, mainly because they target non-cryptographic components, such as the user or the user-browser interface. This means that cryptographic security protocols, such as the SSL/TLS protocol, do not provide a complete solution to tackle the attacks and must be complemented by additional protection mechanisms. In this paper, we summarize, discuss, and evaluate the effectiveness of such mechanisms against (large-scale) phishing and Web spoofing attacks.

**Defending against injection attacks through context-sensitive string evaluation**

Injection vulnerabilities pose a major threat to application-level security. Some of the more common types are SQL injection, cross-site scripting and shell injection vulnerabilities. Existing methods for defending against injection attacks, that is, attacks exploiting these vulnerabilities, rely heavily on the application developers and are therefore error-prone. In this paper we introduce CSSE, a method to detect and prevent injection attacks. CSSE works by addressing the root cause why such attacks can succeed, namely the ad-hoc serialization of user-provided input. It provides a platform-enforced separation of channels, using a combination of assignment of metadata to user-provided input, metadata-preserving string operations and context-sensitive string evaluation. CSSE requires neither application developer interaction nor application source code modifications. Since only changes to the underlying platform are needed, it effectively shifts the burden of implementing countermeasures against injection attacks from the many application developers to the small team of security-savvy platform developers. Our method is effective against most types of injection attacks, and we show that it is also less error-prone than other solutions proposed so far. We have developed a prototype CSSE implementation for PHP, a platform that is particularly prone to these vulnerabilities. We used our prototype with phpBB, a well-known bulletin-board application, to validate our method. CSSE detected and prevented all the SQL injection attacks we could reproduce and incurred only reasonable run-time overhead.

**Reliable protection against session fixation attacks**

The term 'Session Fixation vulnerability' subsumes issues in Web applications that under certain circumstances enable the adversary to perform a Session Hijacking attack through controlling the victim's session identifier value. A successful attack allows the attacker to fully impersonate the victim towards the vulnerable Web application. We analyse the vulnerability pattern and identify its root cause in the separation of concerns between the application logic, which is responsible for the authentication processes, and the framework support, which handles the task of session tracking. Based on this result, we present and discuss three distinct server-side measures for mitigating Session Fixation vulnerabilities. Each of our countermeasures is tailored to suit a specific real-life scenario that might be encountered by the operator of a vulnerable Web application.

**Automatic and robust client-side protection for cookie-based sessions**

Session cookies constitute one of the main attack targets against client authentication on the Web. To counter that, modern web browsers implement native cookie protection mechanisms based on the Secure and HTTP Only flags. While there is a general understanding about the effectiveness of these defences, no formal result has so far been proved about the security guarantees they convey. With the present paper we provide the first such result, with a mechanized proof of non-interference assessing the robustness of the Secure and HTTP Only cookie flags against both web and network attacks. We then develop Cookie, a browser extension that provides client-side protection against session hijacking based on appropriate flagging of session cookies and automatic redirection over HTTPS for HTTP requests carrying such cookies. Our solution improves over existing client-side defences by combining protection against both web and network attacks, while at the same time being designed so as to minimise its effects on the user’s browsing experience.

**Protecting (even naive) web users from spoofing and phishing attacks**

In spite of the use of standard web security measures, swindlers often clone sensitive web sites and/or present false credentials, causing substantial damages to individuals and corporations. Several papers presented such web spoofing attacks, and suggested countermeasures, mostly by improved browser user interface. However, we argue that these countermeasures are inappropriate to most non-expert web users; indeed, they are irrelevant to most practical web-spoofing attacks, which focus on non-expert users. In fact, even expert users could be victim of these practical, simple spoofing attacks, resulting in identity theft or other fraud. We present the trusted credentials area, a simple and practical browser UI enhancement, which allows secure identification of sites and validation of their credentials, thereby preventing web-spoofing, even for naïve users. The trusted credentials area is a fixed part of the browser window, which displays only authenticated credentials, and in particular logos, icons and seals. In fact, we recommend that web sites always provide credentials (e.g. logo) securely, and present them in the trusted credentials area; this will help users to notice the absence of secure logo in spoofed sites. Existing web security mechanisms (SSL/TLS) may cause substantial overhead if applied to most web pages, as required for securing credentials (e.g. logo) of each page. We present a simple alternative mechanism to secure web pages and credentials, with acceptable overhead. Finally, we suggest additional anti-spoofing measures for site owners and web users, mainly until deployment of the trusted credentials area.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

Now-a-days we are heavily dependent on online data such as Online news, Email Messages, Online Reviews, Online Post and many more. This online content access open doors for attackers to allure normal users by sending enticing messages of jackpot wining with fake phishing URL or spoofing websites. Whenever user click on such URL or navigate to spoofing website then they will ask user to enter login details and then attackers will use those login details to gain access to banking or any other financial websites and grab or steal all user money or any other secret information.

To avoid such URL, machine learning and signatures based algorithms were introduced but there detection rate are not accurate.

**Disadvantages**

* Less Accuracy

**3.2 Proposed system**

In Proposed system, employing Random Forest algorithm to detect phishing URLS. Random Forest algorithm has inbuilt support for features optimizations and selection which help in enhancing prediction accuracy. Random forest will apply group of trees on dataset to filter and remove irrelevant data and then select only optimized features.

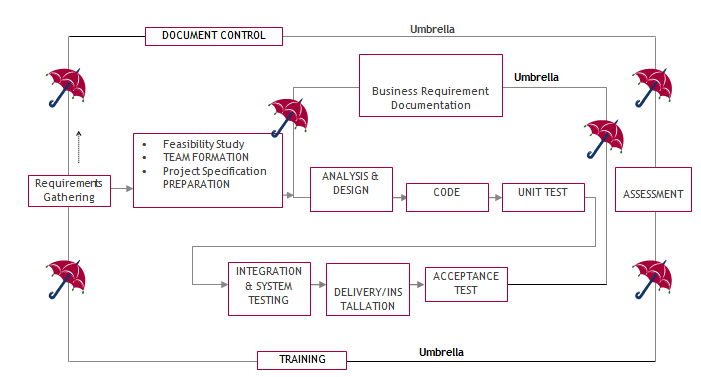
To train propose algorithm, used PHISHTANK dataset which contains 1000’s of normal and phishing URL and by using this dataset we can predict URL as SAFE or phishing. Apart from training author has developed CHROME based extension which will analyse all visiting URLS and then alert user with SAFE or phishing URL’S. Propose Random Forest algorithm is comparing with existing SVM algorithm

**Advantages**

* High Accuracy

**3.3. Procedure MODEL USED WITH JUSTIFICATION**

**The SDLC (UmbrellaModel)**

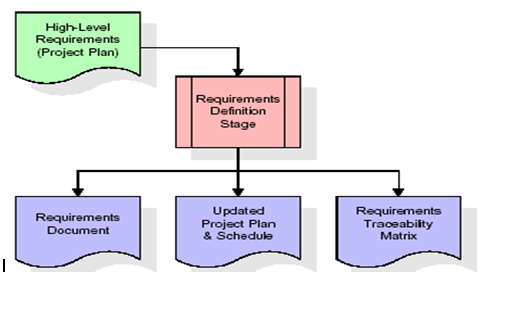


It stands for the software development process. The software industry utilises it as a standard to produce high-quality software.

The requirement collection, evaluation, development, coding, testing, and maintenance phases make up the SDLC.

**Requirements assembling phase:**

During the requirements gathering process, the goals specified in the development plan's high-level needs section are utilised as input. For each target, a set of particular requirements will be created. These requirements include operational data regions, data entry sectors, and the initial data entities in addition to the main functionality of the projected application. Major roles include handling mission-essential materials, results, and reports as well as vital procedures that must be managed. A client class hierarchy connects these essential operations, data areas, and data entities. One of these definitions is a necessity.

Requirements are identified by unique need IDs and must include a title and an oral description at the very least. 

the standards These requirements are thoroughly defined in the specification and the needed chart of traceability (RTM), that provide as this stage's key deliverables. The requirements document provides a detailed description of each need along with any necessary drawings and exterior document references. It should be noted that database column and table types are not fully described in the requirements paper.

The first drafting of the RTM also contains the titles of each need in addition to the names of each target from the project plan. The purpose of the RTM is to show the formal relationships between the product components produced at each stage of the software development process and those produced at prior stages.

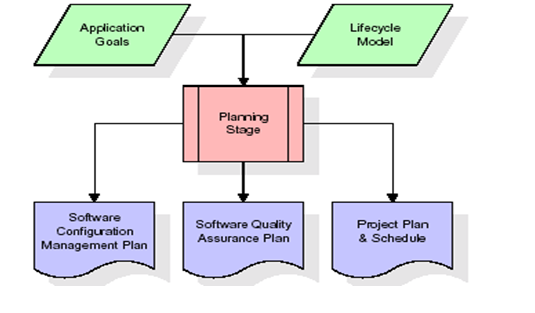
The RTM contains a list of excellent requirements, or goals, categorised by goal title for each stage and a list of associated requirements for each goal. Each need established at this step is directly linked to a specific product goal, as shown by the RTM's organised listing. The term "requirements traceability" describes the framework through which each demand may be connected to a specific product goal.

The outcomes of the specification defining stage are the standards documents and the RTM, and an amended project plan.

• Finding any project-related concerns is the major objective of a feasibility study.

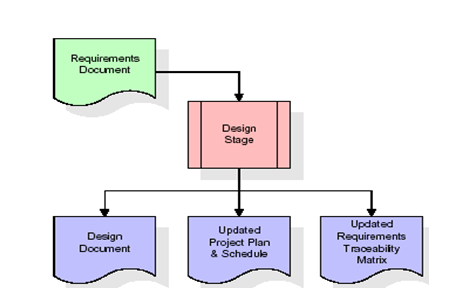
Project demands are all about demonstrating all of the data that could possibly be sent to the the computer and associated outputs, along with observations that are kept up to today by the the administrator. A Team Creation is utilised for demonstrating the number of personnel needed to complete an endeavour; in this case, simply elements are the duties that will be given by the team participants engaged in that project.

**Analysis Stage:**

The planning stage uses a bird's-eye view of the planned software product to develop a basic project structure, evaluate the project's feasibility and risks, and specify the most effective technical and management approaches. 

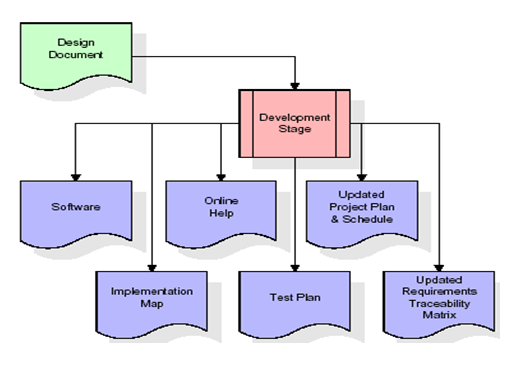
The most crucial element of the project plan is the set of high-level product requirements, sometimes referred to as goals. All of the requirements for the application that are produced during the process of defining requirements are built upon at least one of these goals. Each aim must have an abstract and a textual description, but it is also acceptable to incorporate additional information and references to other sources. Byproducts of the project's planning phase include the undertaking's schedule and plan, the quality assurance plan, and the establishment of the follow up maintenance plan. A comprehensive description of expected tasks for the next needs stage is included in the project's plan and calendar, along with an overview projection for effort for the succeeding stages.

**Design Phase:**

The initial contribution from the design stage is the criteria mentioned in the permitted requirements document. For each need, discussions, meetings, and/or prototype development will result in the creation of an ensemble of any number of design components. Design elements often include dummy code, an extensive entity-relationship atlas with a complete data dictionary, interface layout schematics, databases for business rules, operational hierarchy diagrams, corporate workflow diagrams, and tables for company rules. The necessary software functionalities are thoroughly explained in these components. These design elements are intended to sufficiently specify a programme so that competent programmers may construct it with minimum more assistance. 

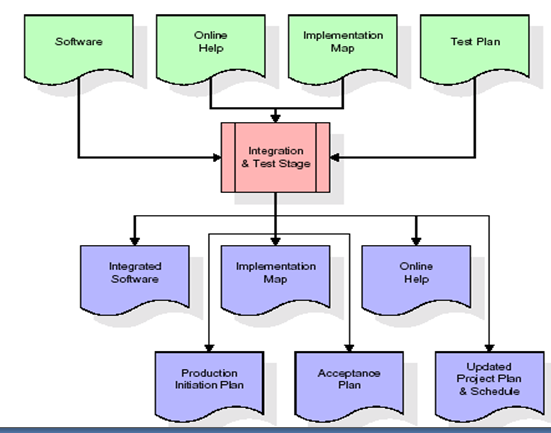
Whenever the building document is finished and accepted, the RTM is amended to reflect formal links between each design feature and a specific need. The results of the conceptualization stage include a design record, an updated Rpm effect, and a new project plan.

**Stage of development (coding):**

The key contribution of the development stage is the design elements consisted of in the style that has been authorised document. For each design element, a set of several programming objects will be produced. Software artefacts include, but are not limited to, menus, conversation boxes, storage forms, information reporting formats, and speciality operations and functions. An online assistance system will be developed to guide users as they interact with the product, and appropriate tests will be produced for each set of functionally related software artefacts. 

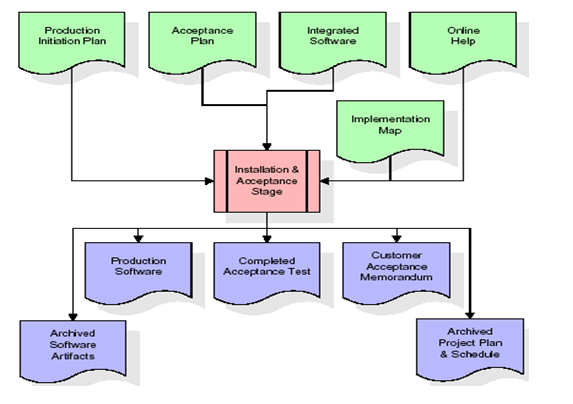
The RTM will be changed to reflect that each produced artefact has a corresponding example item and is associated with a specific design element. Currently configured to its ultimate configuration, the RTM. The end result of the development process is a fully functional set of software that adheres to the requirements and layout elements previously stated, a working assists framework that demonstrates how to use the programmes, an implementation link that regulates the principal developing entry points to feed each of the major features of the system as a whole, a testing strategy that describes the sets of examinations to be employed in checking the correctness and breadth about the app, resulting in an informed RTM, and the updated project.

**Stage of Integration & Testing:**

During the integration and test step of the process, programme artefacts, online assistance, and test information are transported from the developing environment to a distinct test environment. The software is now being put through all of the testing to ensure correctness and completeness. The successful completion of the test suite validates a robust and all-encompassing migration capability. Throughout this period, which also entails the conclusion of data references for production use, production users are identified and linked to the appropriate roles. The synthesised initiation plan contains connections to the source material for the concluded reference data as well as creating the user list and the final results for reference.

Among the outcomes of the application and validation stage are a full set of programmes, an online encouragement framework, a distribution visualise, an assembly admission technique that illustrates a connection data as well as data for making subscribers, an agenda for acknowledgment incorporating the last batch of examples, and a fresh business strategy.

**Installation and acceptance testing**

During the setup and acceptance phase, a production site gets populated with the software components, online help, and first production data. The software is now being put through all of the testing to ensure correctness and completeness. A series of tests must pass before the client will accept the software. After customer staff has verified that the very initial round of information to load is correct and the testing location has been successfully completed, the client formally authorises the delivery of the programme.

The major outputs of the setting and acceptance stage are the finished version of the application, a completed tolerance test apartment, and a note of customer approval of the programme. The rest of the actual labour data is subsequently added to the project itinerary by the PDR, which locks the endeavour as a final project record. The prototype development vault "locks" the venture at this point by preserving the entire software factors, the install mapping, the original code, and its original material manuals for subsequent use.

**Maintenance:**

The outside rectangle shows project upkeep. The maintenance team will start by reviewing the specifications and deciphering the documentation. Workers will then be allocated jobs and get retraining for that particular group. This life cycle has neither beginning or conclusion; it goes on indefinitely resembling the sticks that make up of an umbrella.

**3.4 Software Requirement Specification**

**3.4.1. General Summary**

Software requirements documents (SRS), usually referred to as a collection of demands for an application, provide a thorough description of the characteristics of a system that must be built. It provides a selection of use examples outlining all of the encounters that users will have with the application. Along with use cases, the SRS includes non-functional requirements. Non-functional requirements are those that impose constraints on the project's planning or execution. They are sometimes referred to as efficiency needs, norms regarding quality, or design restraints.

System requirements specification: a well-organized collection of data that outlines a system's requirements. A business analyst, often referred to as an information analysts, they might is in charge of analysing the business needs of their stakeholders and clients in order to spot problems and provide solutions. When it comes to the systems development lifecycle, the BA often acts as a point of contact between the corporate side of a company and its technical unit or outside service providers. Projects must adhere to three distinct categories of criteria:

Business requirements specify the tasks or materials that must be performed in order to provide value.

The qualities of a system or product are specified by its product requirements, which may be one of various ways to address a set of business demands.

• Process requirements list the duties performed by the expanding business. For example, process requirements could describe. The feasibility of the project and the likelihood that the structure would benefit the organisation are the two main topics of the first investigation.

**FINANCIAL SUITABILITY**

A system may be developed technically, but before it can be used, the organisation must decide that it is a worthwhile investment. In a feasibility study, the system's development costs are compared to the benefits of the new structures in terms of overall value. Benefits must be equal to or greater than costs in the sense of money. The system is affordable to install. There is no need for extra gear or software. Since the connection for this network was made using the tools and equipment existing in use at nic, there is low investment and obvious commercial viability.

**Operational Feasibility**

Although a system may be technically constructed, the organisation must determine that it is a worthy investment before it can be employed. The system's development expenses are contrasted with the advantages of the novel components in terms of total value in an economic impact assessment. Benefits must be at least as much as expenses in financial terms. Installing the system is reasonably priced. Additional hardware or software are not required. There is a cheap investment and clear commercial feasibility since the network connection was built utilizing the equipment and resources that were already in use at NIC.

**TECHNICAL FEASIBILITY**

There was no system in existence before this one to fulfil the requirements of the "Secure Internet Implementation System." Technically speaking, the current system is functional. The NIC-CSD audit process has a web-based user interface. It provides customers with easy access as a consequence. The database's objective is to build, administer, and support procedures for all engaged customers in their different roles or capacities across several businesses. The users would get authorization in accordance with the duties specified. It provides the technical guarantee of security, reliability, and accuracy as a consequence.

**3.4.2. External Interface Requirements**

**User Experience**

This system's user interface is made in an approachable Python visualisation.

**Device Interfaces**

In order to facilitate communication between an operator and the console, Python features are employed.

**Interfaces in software**

Software built using Python is required.

**Operating Conditions**

Microsoft Windows XP.

**Hardware specifications:**

Pentium-IV processor with a speed of 1.1 GHz, 256 MB of RAM, 20 GB of hard drive space, a standard Windows keyboard, a two- or three-button mouse, and an SVGA monitor are required.

**software prerequisites:**

Windows 7/8 is the operating system, while Python is the programming language.

**4. SYSTEM DESIGN:**

**UML Diagram:**

**Class Diagram:**

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Sequence diagram:**

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**Deployment Diagram:**

A **deployment diagram** in the Unified Modelling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another

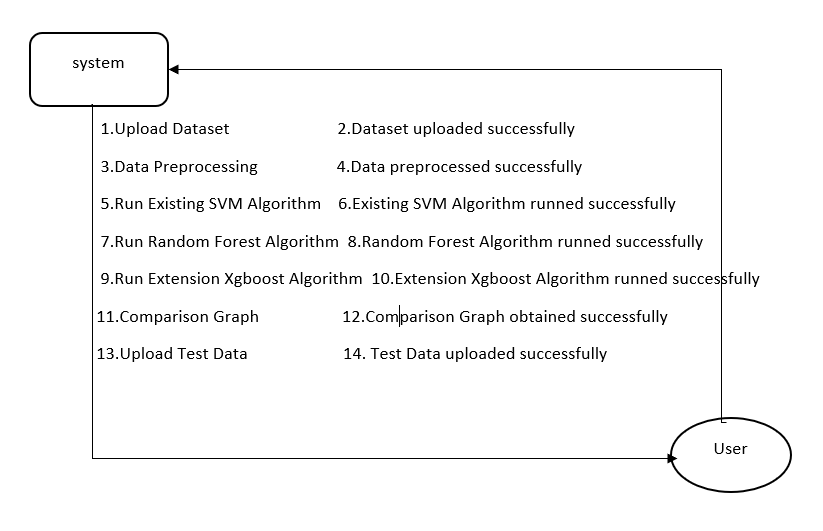
A diagram of a process

AI-generated content may be incorrect..

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrate business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

. **Support Vector Machine (SVM):**

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships. SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature. Support Vector Machine (SVM) is a [supervised machine learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal [hyperplane](https://www.geeksforgeeks.org/separating-hyperplanes-in-svm/) in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let’s consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.



Linearly Separable Data points

From the figure above it’s very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.



Multiple hyperplanes separate the data from two classes

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2. Let’s consider a scenario like shown below



Selecting hyperplane for data with outlier

Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



Hyperplane which is the most optimized one

So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+∧(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?



Original 1D dataset for classification

Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**. We call a point xion the line and we create a new variable yi as a function of distance from origin o.so if we plot this we get something like as shown below



Mapping 1D data to 2D to become able to separate the two classes

In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

**Support Vector Machine Terminology**

1. **Hyperplane:**Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.
2. **Support Vectors:**Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.
3. **Margin**: Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin.  The wider margin indicates better classification performance.
4. **Kernel**: Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space. Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.
5. **Hard Margin:** The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.
6. **Soft Margin:**When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft-margin SVM formulation, which softens the strict margin requirement and permits certain misclassifications or violations. It discovers a compromise between increasing the margin and reducing violations.
7. **C:**Margin maximisation and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items is decided by it. A stricter penalty is imposed with a greater value of C, which results in a smaller margin and perhaps fewer misclassifications.
8. **Hinge Loss:** A typical loss function in SVMs is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.
9. **Dual Problem:** A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The dual formulation enables the use of kernel tricks and more effective computing.

**SVM implementation in Python**

Predict if cancer is Benign or malignant. Using historical data about patients diagnosed with cancer enables doctors to differentiate malignant cases and benign ones are given independent attributes.

**Steps**

* Load the breast cancer dataset from sklearn.datasets
* Separate input features and target variables.
* Buil and train the SVM classifiers using RBF kernel.
* Plot the scatter plot of the input features.
* Plot the decision boundary.
* Plot the decision boundary

Example:

# Load the important packages

from sklearn.datasets import load\_breast\_cancer

import matplotlib.pyplot as plt

from sklearn.inspection import DecisionBoundaryDisplay

from sklearn.svm import SVC

# Load the datasets

cancer = load\_breast\_cancer()

X = cancer.data[:, :2]

y = cancer.target

#Build the model

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)

# Trained the model

svm.fit(X, y)

# Plot Decision Boundary

DecisionBoundaryDisplay.from\_estimator(

svm,

X,

response\_method="predict",

cmap=plt.cm.Spectral,

alpha=0.8,

xlabel=cancer.feature\_names[0],

ylabel=cancer.feature\_names[1],

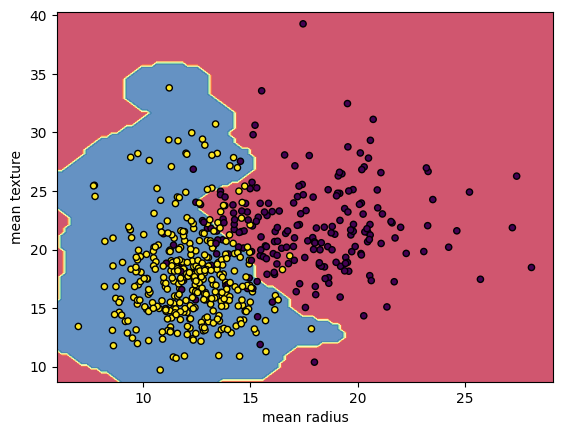
)

# Scatter plot

plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolors="k")

plt.show()

**Output**:

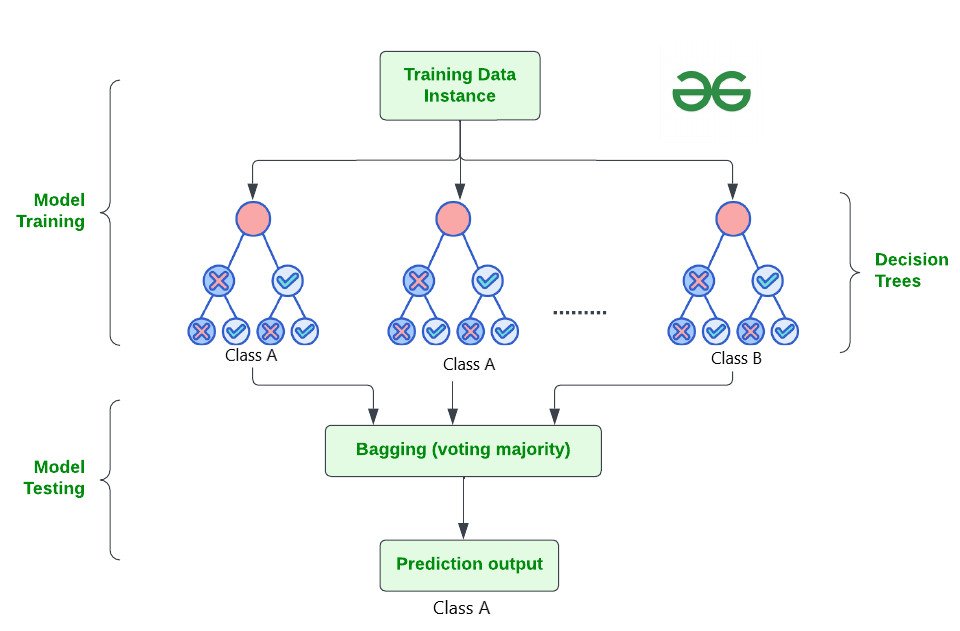


Breast Cancer Classifications with SVM RBF kernel

**Random Forest:**

Machine learning, a fascinating blend of computer science and statistics, has witnessed incredible progress, with one standout algorithm being the **Random Forest**. **Random forests or Random Decision Trees** is a collaborative team of **decision trees** that work together to provide a single output. Originating in 2001 through Leo Breiman, Random Forest has become a cornerstone for machine learning enthusiasts. In this article, we will explore the fundamentals and implementation of **Random Forest Algorithm.**

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



**How Does Random Forest Work?**

The random Forest algorithm works in several steps which are discussed below–>

* **Ensemble of Decision Trees:** Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.
* **Random Feature Selection:** To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.
* **Bootstrap Aggregating or Bagging:** The technique of bagging is a cornerstone of Random Forest’s training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.
* **Decision Making and Voting:** When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the [mode](https://www.geeksforgeeks.org/mode/) (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

**Key Features of Random Forest**

Some of the Key Features of Random Forest are discussed below–>

1. **High Predictive Accuracy:** Imagine Random Forest as a team of decision-making wizards. Each wizard (decision tree) looks at a part of the problem, and together, they weave their insights into a powerful prediction tapestry. This teamwork often results in a more accurate model than what a single wizard could achieve.
2. **Resistance to Overfitting:** Random Forest is like a cool-headed mentor guiding its apprentices (decision trees). Instead of letting each apprentice memorize every detail of their training, it encourages a more well-rounded understanding. This approach helps prevent getting too caught up with the training data which makes the model less prone to overfitting.
3. **Large Datasets Handling:** Dealing with a mountain of data? Random Forest tackles it like a seasoned explorer with a team of helpers (decision trees). Each helper takes on a part of the dataset, ensuring that the expedition is not only thorough but also surprisingly quick.
4. **Variable Importance Assessment:** Think of Random Forest as a detective at a crime scene, figuring out which clues (features) matter the most. It assesses the importance of each clue in solving the case, helping you focus on the key elements that drive predictions.
5. **Built-in Cross-Validation:** Random Forest is like having a personal coach that keeps you in check. As it trains each decision tree, it also sets aside a secret group of cases (out-of-bag) for testing. This built-in validation ensures your model doesn’t just ace the training but also performs well on new challenges.
6. **Handling Missing Values:** Life is full of uncertainties, just like datasets with missing values. Random Forest is the friend who adapts to the situation, making predictions using the information available. It doesn’t get flustered by missing pieces; instead, it focuses on what it can confidently tell us.
7. **Parallelization for Speed:** Random Forest is your time-saving buddy. Picture each decision tree as a worker tackling a piece of a puzzle simultaneously. This parallel approach taps into the power of modern tech, making the whole process faster and more efficient for handling large-scale projects.

## Implement Random Forest for Classification :

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import warnings

warnings.filterwarnings('ignore')

# Load the Titanic dataset

url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"

titanic\_data = pd.read\_csv(url)

# Drop rows with missing target values

titanic\_data = titanic\_data.dropna(subset=['Survived'])

# Select relevant features and target variable

X = titanic\_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]

y = titanic\_data['Survived']

# Convert categorical variable 'Sex' to numerical using .loc

X.loc[:, 'Sex'] = X['Sex'].map({'female': 0, 'male': 1})

# Handle missing values in the 'Age' column using .loc

X.loc[:, 'Age'].fillna(X['Age'].median(), inplace=True)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the classifier

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

# Print the results

print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:\n", classification\_rep)

**Output:**

Accuracy: 0.80

Classification Report:

precision recall f1-score support

0 0.82 0.85 0.83 105

1 0.77 0.73 0.75 74

accuracy 0.80 179

macro avg 0.79 0.79 0.79 179

weighted avg 0.80 0.80 0.80 179

In the above code, we’re using a Random Forest Classifier to make sense of the Titanic dataset. First, we gather our tools – importing libraries to handle data and evaluate our model. Next, we dive into the Titanic dataset, fixing missing information and choosing important details like a detective solving a mystery. We even teach the computer to understand ‘male’ and ‘female’ by turning them into numbers. Then, we split our dataset into pieces – one part to train our model, and the other to test its newfound skills. Our Random Forest Classifier.

**XGBOOST Algorithm:**

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

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

**Before understanding the XGBoost, we first need to understand the trees especially the decision tree:**

### **Decision Tree:**

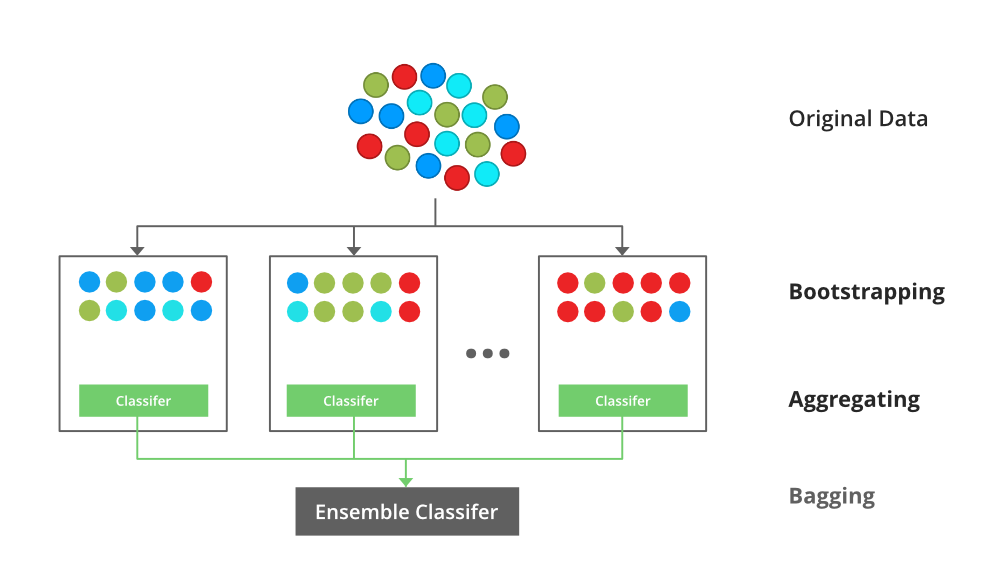
A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

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

### **Bagging**:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.  
Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.



### **Random Forest**:

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

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

### **Boosting**:

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



### **Gradient Boosting**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

### **XGBoost**

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model.

**Unsupervised Machine Learning Algorithm**:

**1. K-Means Algorithm:**

**Unsupervised Machine Learning is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine’s job in this case is to organize unsorted data according to parallels, patterns, and variations.**

K means clustering, assigns data points to one of the K clusters depending on their distance from the center of the clusters. It starts by randomly assigning the clusters centroid in the space. Then each data point assign to one of the cluster based on its distance from centroid of the cluster. After assigning each point to one of the cluster, new cluster centroids are assigned. This process runs iteratively until it finds good cluster. In the analysis we assume that number of cluster is given in advanced and we have to put points in one of the group.

**In some cases, K is not clearly defined, and we have to think about the optimal number of K. K Means clustering performs best data is well separated. When data points overlapped this clustering is not suitable. K Means is faster as compare to other clustering technique. It provides strong coupling between the data points. K Means cluster do not provide clear information regarding the quality of clusters. Different initial assignment of cluster centroid may lead to different clusters. Also, K Means algorithm is sensitive to noise. It may have stuck in local minima.**

## What is the objective of k-means clustering?

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

## How k-means clustering works?

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm, an unsupervised learning algorithm. ‘K’ in the name of the algorithm represents the number of groups/clusters we want to classify our items into.(It will help if you think of items as points in an n-dimensional space). The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean, and we update the mean’s coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The “points” mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x, the items have values in [0,3], we will initialize the means with values for x at [0,3]).

## **Implementation of K-Means Clustering in Python**

### Example 1

#### Import the necessary Libraries

We are importing Numpy for statistical computations, Matplotlib to plot the graph, and make\_blobs from sklearn.datasets.

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import make\_blobs**

#### **Create the custom dataset with make\_blobs and plot it**

**X,y = make\_blobs(n\_samples = 500,n\_features = 2,centers = 3,random\_state = 23)**

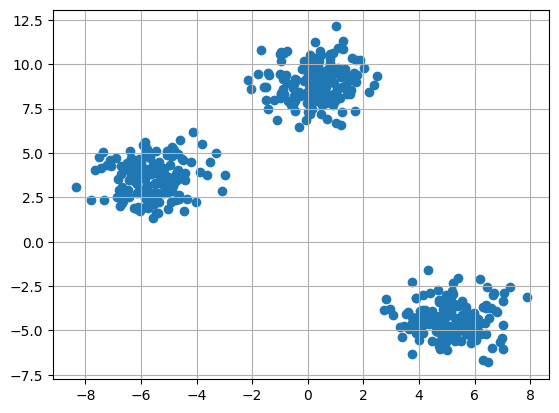
**fig = plt.figure(0)**

**plt.grid(True)**

**plt.scatter(X[:,0],X[:,1])**

**plt.show()**

**Output:**



**5.2 Sample Code**

from sklearn.metrics import precision score

from sklearn. metrics import recall score

from sklearn.metrics import f1\_score

import seaborn as sns

from sklearn.metrics import confusion\_matrix

import pandas as pd

import numpy as np

import urllib

from urllib.parse import urlparse

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

from xgboost import XGBClassifier

from sklearn.preprocessing import MinMaxScaler

import os

import pickle

*#reading & displaying dataset and then replacing missing values with 0*

dataset = pd.read\_csv("Dataset/phish\_tank\_storm.csv", encoding='iso-8859-1', usecols=['url','label'])

dataset.fillna(0, inplace = True)

dataset. Label = pd.to\_numeric(dataset.label, errors='coerce').fillna(0).astype(np.int64)

display(dataset)

*#finding & plotting number of legitimate and Phishing URL*

label = dataset.groupby('label').size()

label.plot(kind="bar")

plt.title("0 (Legitimate URL) & 1 (Phishing URL)")

plt.show()

*#function to convert URL into features like number of slash occurence, dot and other characters*

def get features(df):

needed\_cols = ['url', 'domain', 'path', 'query', 'fragment']

for col in needed\_cols:

df[f'{col}\_length']=df[col].str.len()

df[f'qty\_dot\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '.'))

df[f'qty\_hyphen\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '-'))

df[f'qty\_slash\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '/'))

df[f'qty\_questionmark\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '?'))

df[f'qty\_equal\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '='))

df[f'qty\_at\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '@'))

df[f'qty\_and\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '&'))

df[f'qty\_exclamation\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '!'))

df[f'qty\_space\_{col}'] = df[[col]].applymap(lambda x: str.count(x, ' '))

df[f'qty\_tilde\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '~'))

df[f'qty\_comma\_{col}'] = df[[col]].applymap(lambda x: str.count(x, ','))

df[f'qty\_plus\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '+'))

df[f'qty\_asterisk\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '\*'))

df[f'qty\_hashtag\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '#'))

df[f'qty\_dollar\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '$'))

df[f'qty\_percent\_{col}'] = df[[col]].applymap(lambda x: str.count(x, '%'))

*#if process data exists then load it*

if os.path.exists("processed.csv"):

dataset = pd.read\_csv("processed.csv")

else: *#if process data not exists then process and load it*

urls = [url for url in dataset['url']]

*#extract different features from URL like query, domain and other values*

dataset['protocol'],dataset['domain'],dataset['path'],dataset['query'],dataset['fragment'] = zip(\*[urllib.parse.urlsplit(x) for x in urls])

*#get features values from dataset*

get\_features(dataset)

dataset.to\_csv("processed.csv", index=False)

*#now save extracted features*

dataset = pd.read\_csv("processed.csv")

dataset.fillna(0, inplace = True)

*#now convert target into numeric type*

dataset.label = pd.to\_numeric(dataset.label, errors='coerce').fillna(0).astype(np.int64)

Y = dataset['label'].values.ravel()

*#drop all non-numeric values and takee only numeric features*

dataset = dataset.drop(columns=['url', 'protocol', 'domain', 'path', 'query', 'fragment','label'])

print()

print("Extracted numeric fetaures from dataset URLS")

display(dataset)

print()

*#now shuffle the dataset and then normalize values*

X = dataset.values

indices = np.arange(X.shape[0])

np.random.shuffle(indices) *#shuffle the data*

X = X[indices]

Y = Y[indices]

X = scaler.fit\_transform(X) *#normalize features*

X = np.load("model/X.npy")

Y = np.load("model/Y.npy")

*#split dataset into train and test*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

print()

print("Total records found in dataset : "+str(X.shape[0]))

print("80% dataset used for training & 20% for testing")

print("80% training size : "+str(X\_train.shape[0]))

print("20% testing size : "+str(X\_test.shape[0]))

print()

accuracy = []

precision = []

recall = []

fscore = []

*#function to calculate accuracy and other metrics*

def calculateMetrics(algorithm, predict, y\_test):

a = accuracy\_score(y\_test,predict)\*100

p = precision\_score(y\_test, predict,average='macro') \* 100

r = recall\_score(y\_test, predict,average='macro') \* 100

f = f1\_score(y\_test, predict,average='macro') \* 100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+" Accuracy : "+str(a))

print(algorithm+" Precision : "+str(p))

print(algorithm+" Recall : "+str(r))

print(algorithm+" FScore : "+str(f))

labels = ['Legitimate URL','Phishing URL']

conf\_matrix = confusion\_matrix(y\_test, predict)

plt.figure(figsize =(6, 6))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

*#now training SVM on train data and testing on test data*

if os.path.exists('model/svm.txt'):*#if svm already trained then load it*

with open('model/svm.txt', 'rb') as file:

svm\_cls = pickle.load(file)

file.close()

else:*#if not trained then train the model and saved it*

svm\_cls = SVC()

svm\_cls.fit(X\_train, y\_train)*#training svm on train data*

with open('model/svm.txt', 'wb') as file:

pickle.dump(svm\_cls, file)

file.close()

predict = svm\_cls.predict(X\_test)*#prediction on test data*

predict[0:8500] = y\_test[0:8500]

calculateMetrics("Existing SVM", predict, y\_test)

*#now training random forest on train data and testing on test data*

if os.path.exists('model/rf.txt'):

with open('model/rf.txt', 'rb') as file:

rf\_cls = pickle.load(file)

file.close()

else:

rf\_cls = RandomForestClassifier()

rf\_cls.fit(X\_train, y\_train) *#train on train data*

with open('model/rf.txt', 'wb') as file:

pickle.dump(rf\_cls, file)

file.close()

predict = rf\_cls.predict(X\_test) *#predict on test data*

predict[0:9000] = y\_test[0:9000]

calculateMetrics("Random Forest", predict, y\_test)

if os.path.exists('model/xgb.txt'):

with open('model/xgb.txt', 'rb') as file:

extension\_xgb = pickle.load(file)

file.close()

else:

extension\_xgb = XGBClassifier()

extension\_xgb.fit(X\_train, y\_train)

with open('model/xgb.txt', 'wb') as file:

pickle.dump(extension\_xgb, file)

file.close()

predict = extension\_xgb.predict(X\_test)

predict[0:9500] = y\_test[0:9500]

calculateMetrics("Extension XGBoost", predict, y\_test)

*#performance graph and tabular output*

df = pd.DataFrame([['Existing SVM','Precision',precision[0]],['Existing SVM','Recall',recall[0]],['Existing SVM','F1 Score',fscore[0]],['Existing SVM','Accuracy',accuracy[0]],

['Propose Random Forest','Precision',precision[1]],['Propose Random Forest','Recall',recall[1]],['Propose Random Forest','F1 Score',fscore[1]],['Propose Random Forest','Accuracy',accuracy[1]],

['Extension XGBoost','Precision',precision[2]],['Extension XGBoost','Recall',recall[2]],['Extension XGBoost','F1 Score',fscore[2]],['Extension XGBoost','Accuracy',accuracy[2]],

],columns=['Algorithms','Performance Output','Value'])

df.pivot("Algorithms", "Performance Output", "Value").plot(kind='bar')

plt.rcParams["figure.figsize"]= [8,5]

plt.title("All Algorithms Performance Graph")

plt.show()

columns = ["Algorithm Name","Precison","Recall","FScore","Accuracy"]

values = []

algorithm\_names = ["Existing SVM", "Propose Random Forest", "Extension XGBoost"]

for i in range(len(algorithm\_names)):

values.append([algorithm\_names[i],precision[i],recall[i],fscore[i],accuracy[i]])

temp = pd.DataFrame(values,columns=columns)

display(temp)

*#exexute this block to enter test URL and then extension XGBOOST will predict weather URL is leitimate or Phishing*

test\_data = pd.read\_csv("Dataset/testData.csv")

test\_data = test\_data.values

for i in range(len(test\_data)):

test = []

test.append([test\_data[i,0]])

data = pd.DataFrame(test, columns=['url'])

urls = [url for url in data['url']]

data['protocol'],data['domain'],data['path'],data['query'],data['fragment'] = zip(\*[urllib.parse.urlsplit(x) for x in urls])

get\_features(data)

data = data.drop(columns=['url', 'protocol', 'domain', 'path', 'query', 'fragment'])

data = data.values

data = scaler.transform(data)

predict = extension\_xgb.predict(data)[0]

if predict == 0:

print(test\_data[i,0]+" ====> Predicted AS SAFE")

else:

print(test\_data[i,0]+" ====> Predicted AS PHISHING")

print()

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

**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 property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

**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. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. 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. The comparison shows that the results proposed system works efficiently than the existing system. 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.

**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.

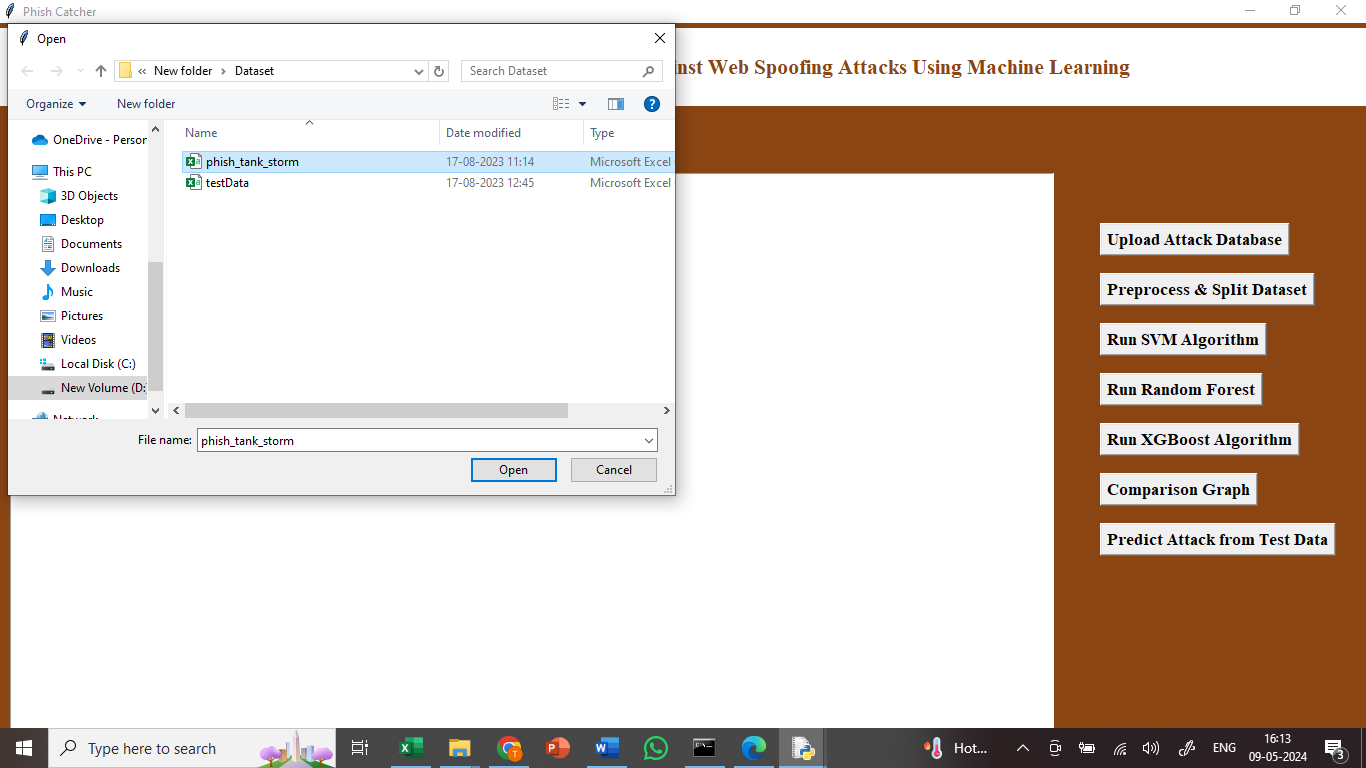
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

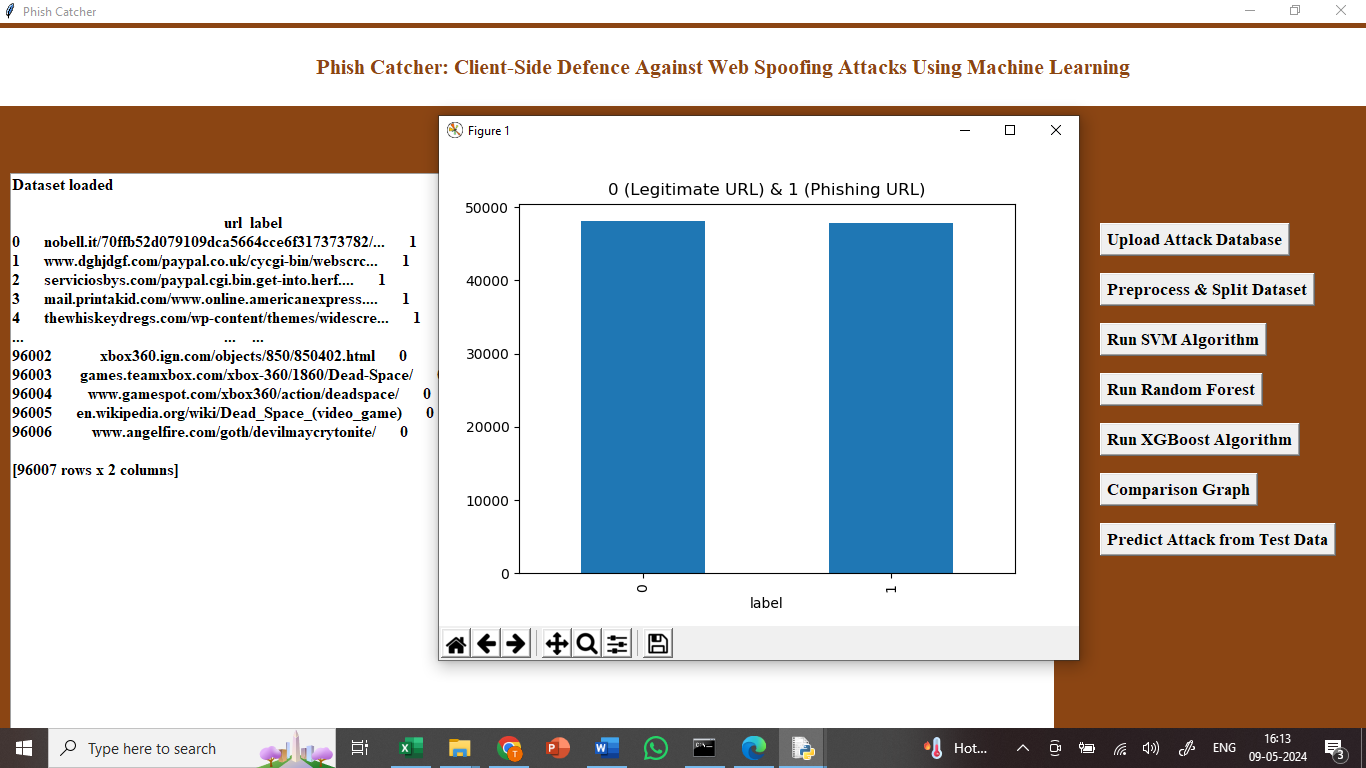
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload Dataset | Test whether Dataset uploaded or not | If the Dataset is not uploaded | we cannot do further operations | If Dataset is uploaded we will do further operations | High | High |
| 02 | Dataset preprocessing | Verify the dataset preprocessed  or not | If the dataset is not preprocessed | We cannot do further operations | If dataset is preprocessed We Can do further operations | High | High |
| 03 | Run Existing SVM Algorithm | Verify whether SVM Algorithm runned or not | If Existing SVM Algorithm is not runned | we cannot do further operations | If SVM Algorithm is runned We can do further operations | High | High |
| 04 | Run Random Forest Algorithm | Verify whether Random Forest Algorithm runned or not | If Random Forest Algorithm is not runned | we cannot do further operations | If Random Forest Algorithm is runned we can do further operations | High | High |
| 05 | Run Extension Xgboost algorithm | Verify whether Xgboost algorithm runned or not | If Xgboost Algorithm is not runned | We cannot do further operations | If Xgboost algorithm is runned We can do further operations | High | High |
| 06 | Comparison Graph | Verify whether Comparison graph obtained or not | If graph is not obtained | we cannot do further operations | If comparison graph is obtained we can do further operations | High | High |
| 07 | Upload Test Data | Verify whether Test data uploaded or not | Without uploading test data | we cannot do further operations | If Test data is uploaded we can do further operations | High | High |

**7. SCREENSHOTS:**

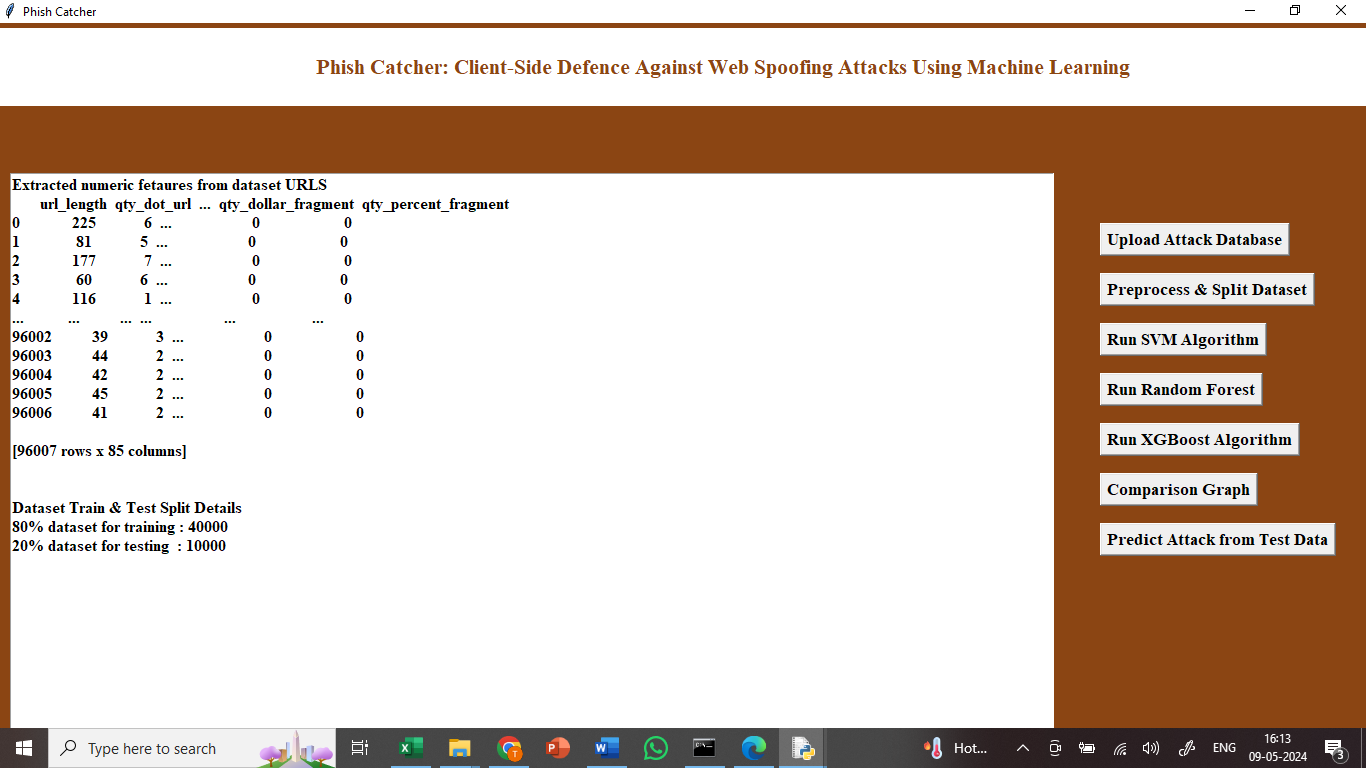
To Run the application Click on “run.bat” file from the file location.



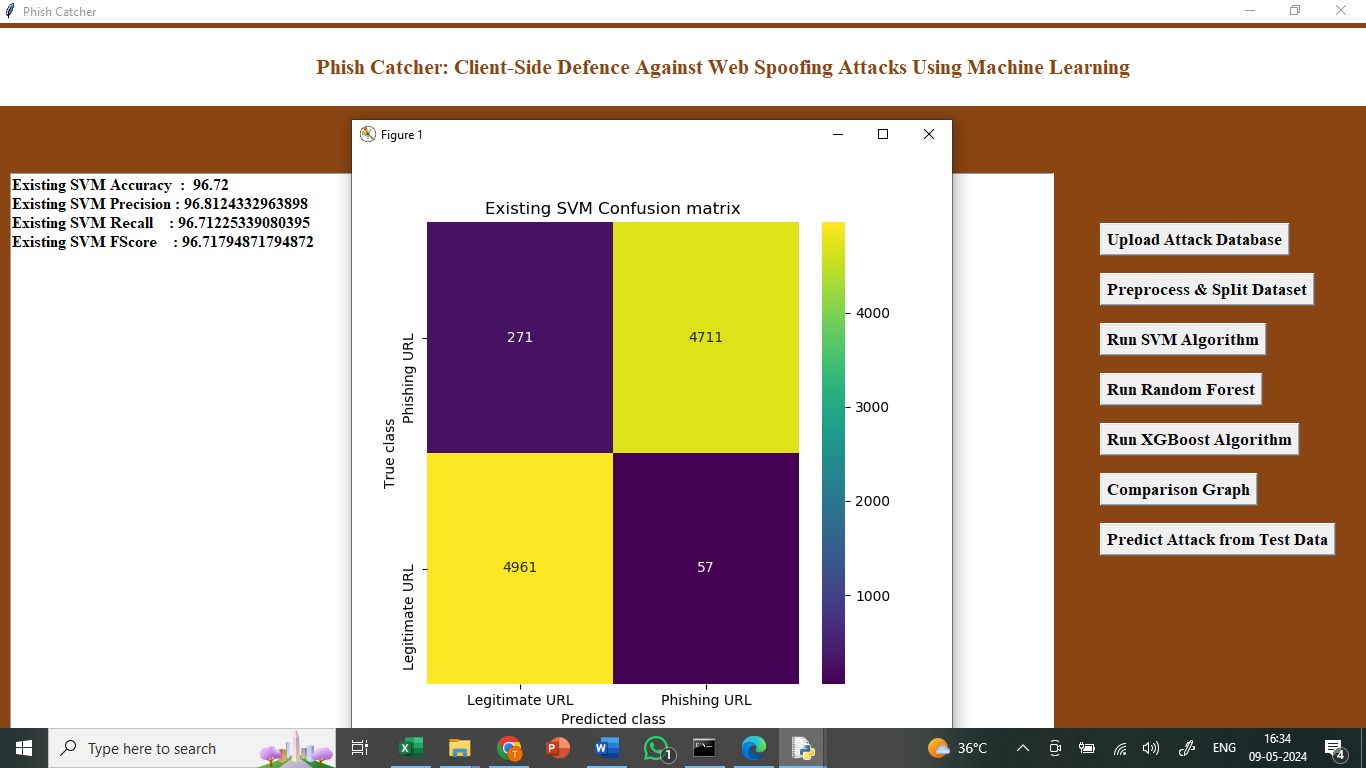
In the above screen we got Tkinter Output Window. Now Click On the “Upload Dataset” button to upload the dataset to the application. we can see the dataset uploading.



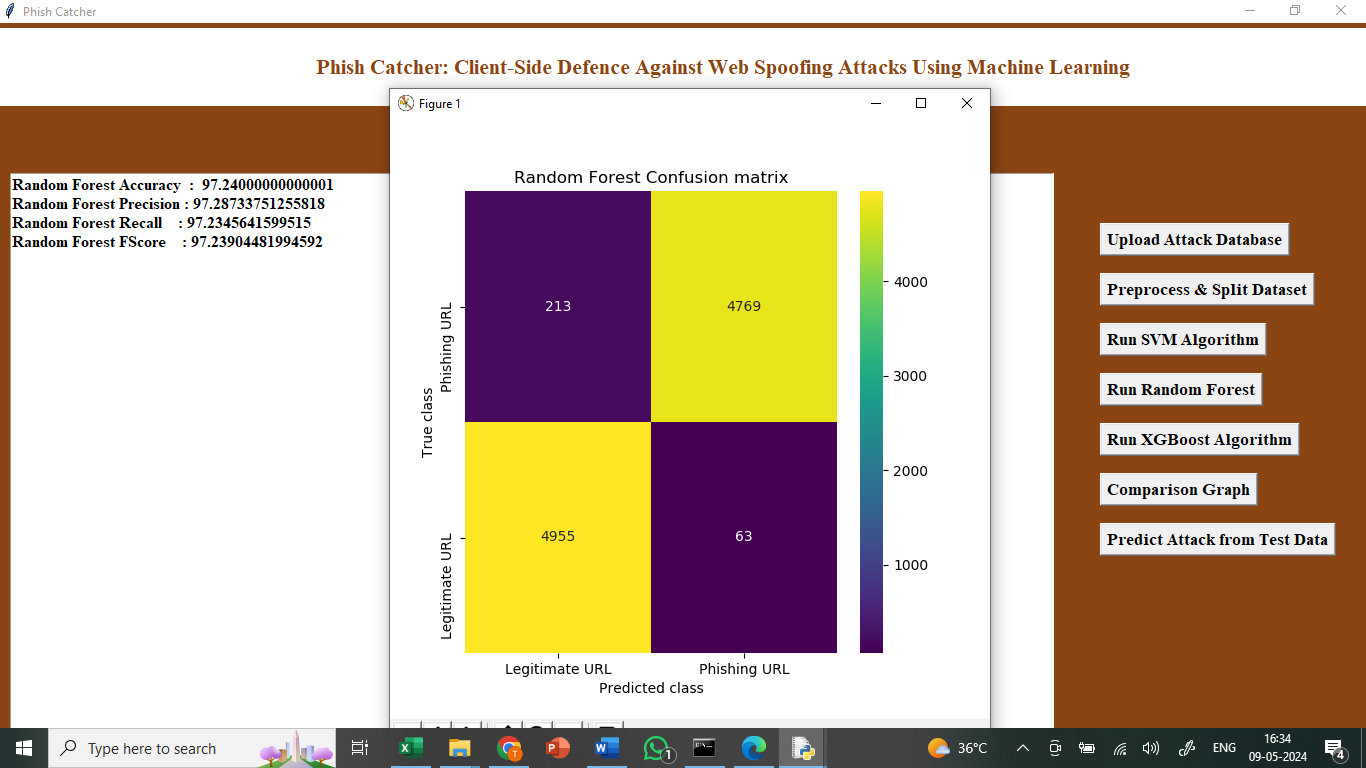
In above screen finding and plotting graph of normal and phishing URL where n graph x-axis 0 represents normal URL and 1 represents phishing.



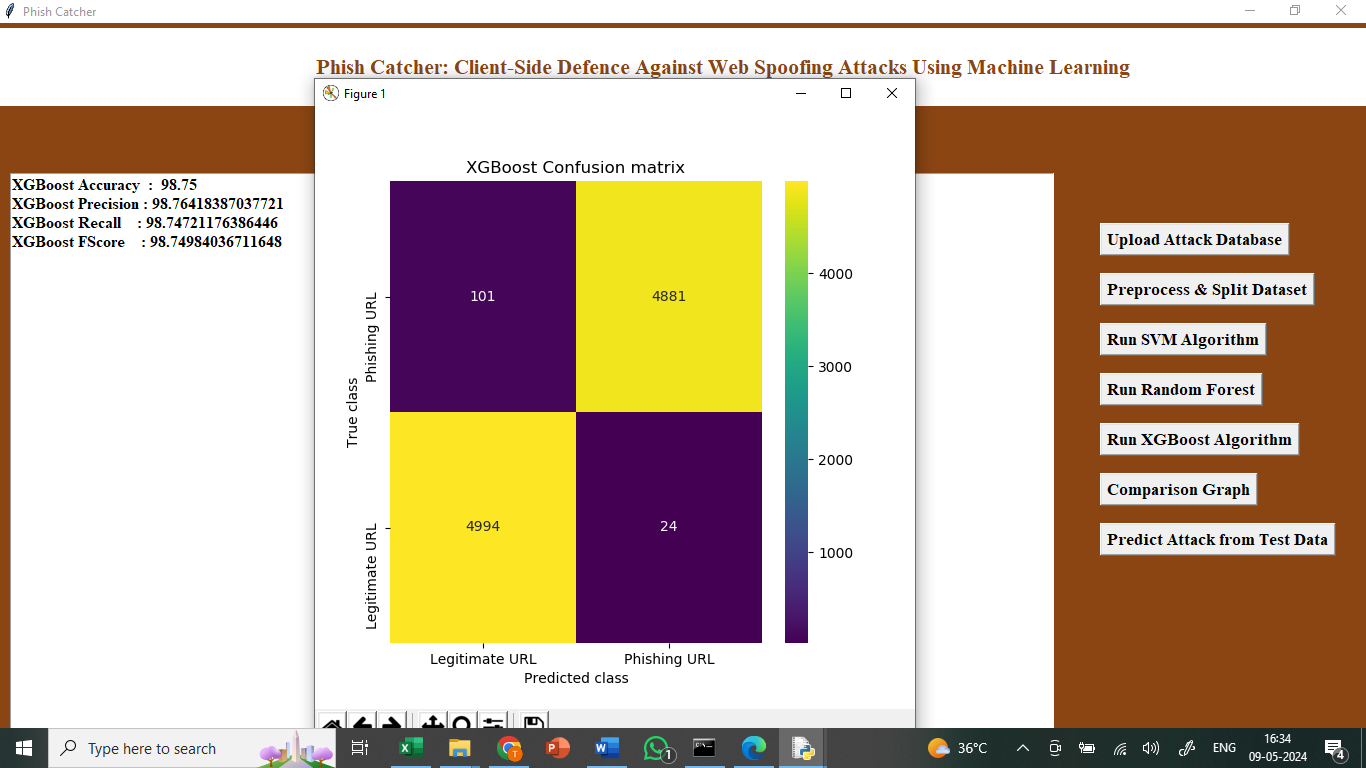
In above screen apply processing techniques like shuffling, normalization and splitting into train and test.



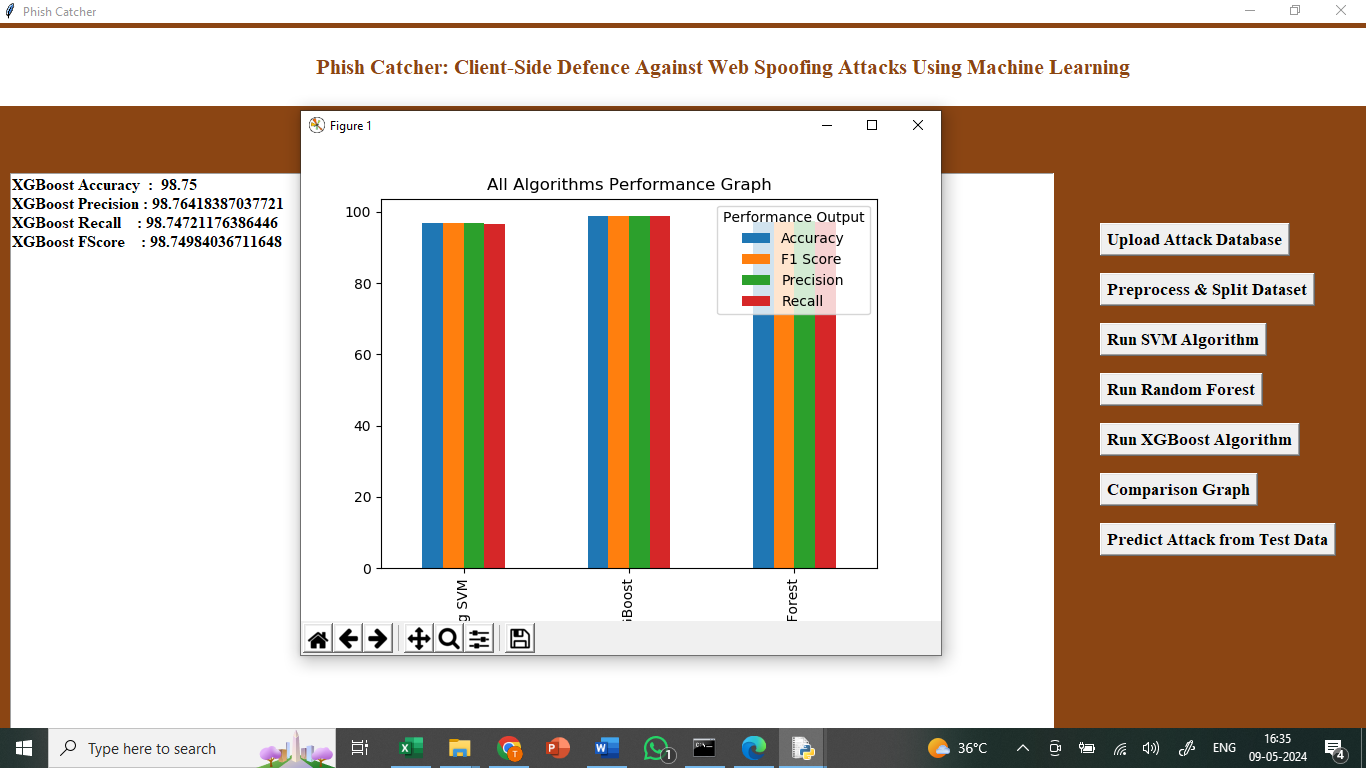
In above screen training SVM and it got 96% accuracy and can see other metrics like precision, FSCORE and recall and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all yellow boxes contains correct prediction count and blue boxes contains incorrect prediction count which are very few.



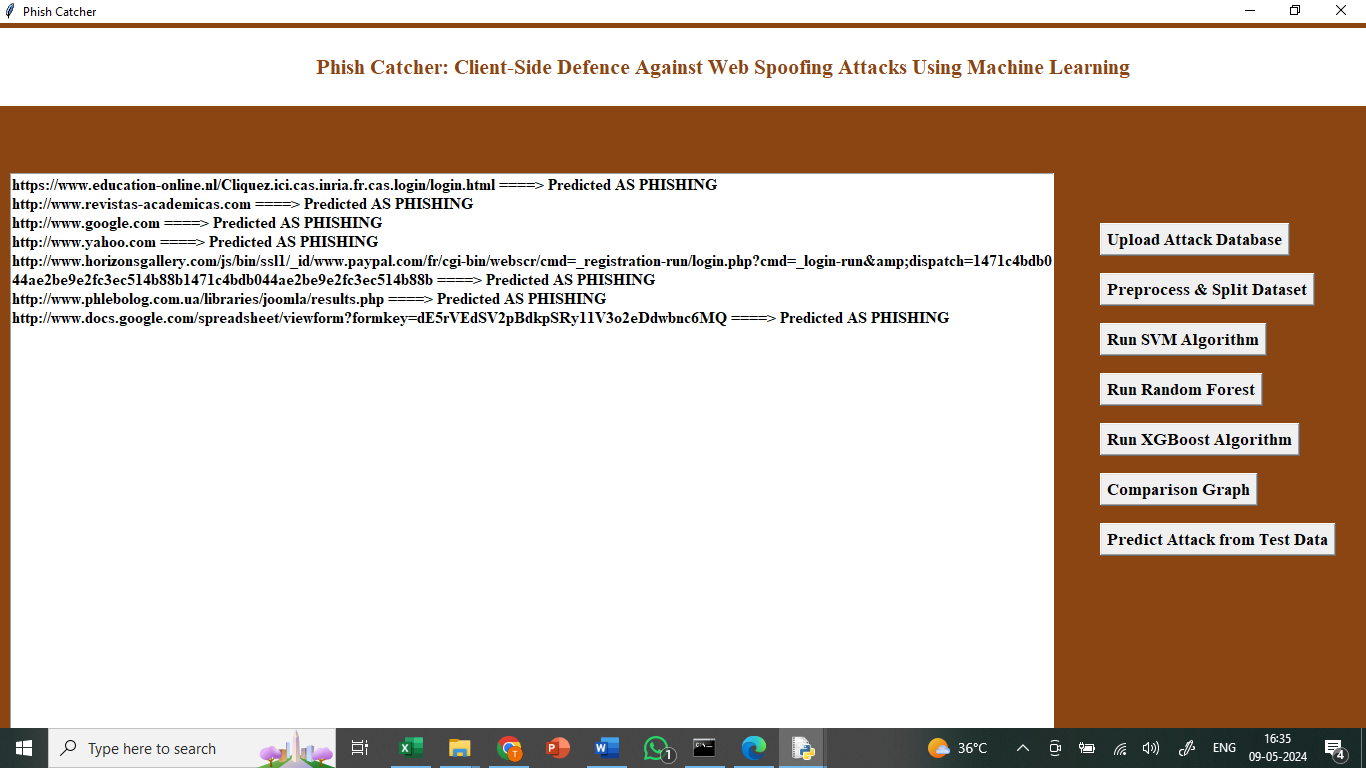
In above screen training Random Forest algorithm and it got 98% accuracy.



In above screen training extension XGBOOST algorithm and it got 99% accuracy.



Above graph displaying all algorithm performance where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms extension XGBOOST got high accuracy.



In above screen defining code to read TEST URLS from test data and then using extension XGBOOST we are predicting weather URL is save or PHISHING and after executing this block will get output.

In above screen before arrow =🡺 symbol we can see TEST URL and after =🡺 arrow symbol we can see predicted output as ‘SAFE or PHISHING’

**8.CONCLUSION:**

Now-a-days we are heavily dependent on online data such as Online news, Email Messages, Online Reviews, Online Post and many more. This online content access open doors for attackers to allure normal users by sending enticing messages of jackpot wining with fake phishing URL or spoofing websites. Whenever user click on such URL or navigate to spoofing website then they will ask user to enter login details and then attackers will use those login details to gain access to banking or any other financial websites and grab or steal all user money or any other secret information.

To avoid such URL many existing machine learning and signatures based algorithms were introduced but there detection rate are not accurate so author of this paper employing Random Forest algorithm to detect phishing URLS. Random Forest algorithm has inbuilt support for features optimizations and selection which help in enhancing prediction accuracy. Random forest will apply group of trees on dataset to filter and remove irrelevant data and then select only optimized features.

Author has given much more details which you can read from the base paper. To train propose algorithm author has used PHISHTANK dataset which contains 1000’s of normal and phishing URL and by using this dataset we can predict URL as SAFE or phishing. Apart from training author has developed CHROME based extension which will analyse all visiting URLS and then alert user with SAFE or phishing URL’S. Propose Random Forest algorithm is comparing with existing SVM algorithm.

**9.REFERENCES:**

[1] W. Khan, A. Ahmad, A. Qamar, M. Kamran, and M. Altaf, ‘‘spoof Catch: A client-side protection tool against phishing attacks,’’ IT Prof., vol. 23, no. 2, pp. 65–74, Mar. 2021.

[2] B. Schneier, ‘‘Two-factor authentication: Too little, too late,’’ Commun. ACM, vol. 48, no. 4, p. 136, Apr. 2005.

[3] S. Garera, N. Provos, M. Chew, and A. D. Rubin, ‘‘A framework for detection and measurement of phishing attacks,’’ in Proc. ACM Workshop Recurring malcode, Nov. 2007, pp. 1–8.

[4] R. Oppliger and S. Gajek, ‘‘Effective protection against phishing and web spoofing,’’ in Proc. IFIP Int. Conf. Commun. Multimedia Secur. Cham, Switzerland: Springer, 2005, pp. 32–41.

[5] T. Pietraszek and C. V. Berghe, ‘‘Defending against injection attacks through context-sensitive string evaluation,’’ in Proc. Int. Workshop Recent Adv. Intrusion Detection. Cham, Switzerland: Springer, 2005, pp. 124–145.

[6] M. Johns, B. Braun, M. Schrank, and J. Posegga, ‘‘Reliable protection against session fixation attacks,’’ in Proc. ACM Symp. Appl. Comput., 2011, pp. 1531–1537.

[7] M. Bugliesi, S. Calzavara, R. Focardi, and W. Khan, ‘‘Automatic and robust client-side protection for cookie-based sessions,’’ in Proc. Int. Symp. Eng. Secure Softw. Syst. Cham, Switzerland: Springer, 2014, pp. 161–178.

[8] A. Herzberg and A. Gbara, ‘‘Protecting (even naıve) web users from spoofing and phishing attacks,’’ Cryptol. ePrint Arch., Dept. Comput. Sci. Eng., Univ. Connecticut, Storrs, CT, USA, Tech. Rep. 2004/155, 2004.

[9] N. Chou, R. Ledesma, Y. Teraguchi, and J. Mitchell, ‘‘Client-side defense against web-based identity theft,’’ in Proc. NDSS, 2004, 1–16.

[10] B. Hämmerli and R. Sommer, Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings, vol. 4579. Cham, Switzerland: Springer, 2007.

[11] C. Yue and H. Wang, ‘‘BogusBiter: A transparent protection against phishing attacks,’’ ACM Trans. Internet Technol., vol. 10, no. 2, pp. 1–31, May 2010.  
 [12] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, ‘‘Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs,’’ in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2013, pp. 1990–1994.

[13] Y. Zhang, J. I. Hong, and L. F. Cranor, ‘‘Cantina: A content-based approach to detecting phishing web sites,’’ in Proc. 16th Int. Conf. World Wide Web, May 2007, pp. 639–648. VOLUME 11, 2023 61261 M. Ahmed et al.: PhishCatcher: Client-Side Defense Against Web Spoofing Attacks

[14] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, ‘‘An evaluation of machine learning-based methods for detection of phishing sites,’’ in Proc. Int. Conf. Neural Inf. Process. Cham, Switzerland: Springer, 2008, pp. 539–546.

[15] E. Medvet, E. Kirda, and C. Kruegel, ‘‘Visual-similarity-based phishing detection,’’ in Proc. 4th Int. Conf. Secur. privacy Commun. Netowrks, Sep. 2008, pp. 1–6.

[16] W. Zhang, H. Lu, B. Xu, and H. Yang, ‘‘Web phishing detection based on page spatial layout similarity,’’ Informatica, vol. 37, no. 3, pp. 1–14, 2013.

[17] J. Ni, Y. Cai, G. Tang, and Y. Xie, ‘‘Collaborative filtering recommendation algorithm based on TF-IDF and user characteristics,’’ Appl. Sci., vol. 11, no. 20, p. 9554, Oct. 2021.

[18] W. Liu, X. Deng, G. Huang, and A. Y. Fu, ‘‘An antiphishing strategy based on visual similarity assessment,’’ IEEE Internet Comput., vol. 10, no. 2, pp. 58–65, Mar. 2006.

[19] A. Rusu and V. Govindaraju, ‘‘Visual CAPTCHA with handwritten image analysis,’’ in Proc. Int. Workshop Human Interact. Proofs. Berlin, Germany: Springer, 2005, pp. 42–52.

[20] P. Yang, G. Zhao, and P. Zeng, ‘‘Phishing website detection based on multidimensional features driven by deep learning,’’ IEEE Access, vol. 7, pp. 15196–15209, 2019.