EVASION ATTACKS AND DEFENSE MECHANISMS FOR MACHINE LEARNING – BASED WEB PHISING CLASSIFIERS

A PROJECT REPORT

submitted to

Rayalaseema University, Kurnool

in partial fulfillment of the requirements

for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

AMPOLU SADVIK	**21RU1A0504**
ANNAPUREDDY HAMSA VARDHAN REDDY	**21RU1A0505**
GANDLA SIREESHA	**21RU1A0513**
PARVATHANENI LAKSHMAN RAO	**21RU1A0540**
NAGELLA JARUSHA ROSE	**22RU1A0505**

Under the esteemed guidance of

V.SATISH KUMAR **M. Tech (Ph. D)**

Assistant Professor (Ad hoc)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAYALASEEMA UNIVERSITY COLLEGE OF ENGINEERING

[RAYALASEEMA UNIVERSITY, KURNOOL – 518 007, A.P., INDIA]

MAY - 2025

RAYALASEEMA UNIVERSITY
COLLEGE OF ENGINEERING

KURNOOL – 518007, Andhra Pradesh (INDIA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"EVASION ATTACKS AND DEFENSE MECHANISMS LEARNING – BASED WEB PHISING"** is the record of bonafide work done by

ANNAPUREDDY HAMSA VARDHAN REDDY	**21RU1A0504**
AMPOLU SADVIK	**21RU1A0505**
GANDLA SIREESHA	**21RUA1A0513**
PARVATHANENI LAKSHMAN RAO	**21RU1A0540**
NAGELLA JARUSHA ROSE	**22RU5A0505**

and is being submitted to Rayalaseema University, Kurnool in partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** during the academic year 2024-25.

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

Signature of Guide | **Signature of HOD**

V. SATISH KUMAR
Assistant Professor (Ad hoc)
Dept. of CSE
Rayalaseema University
College of Engineering,
Kurnool – 518 007, A.P., India

V.SATISH KUMAR
Assistant Professor (Ad hoc) &
Coordinator
Dept. of Computer Science and Engineering
Rayalaseema University
College of Engineering,
Kurnool – 518 007, A.P., India

Submitted for university examination held on date …….

INTERNAL EXAMINER | **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to my guide, **Mr. V. Satish Kumar**, **M.Tech. (Ph. D),** Assistant Professor (Ad hoc), Department of Computer Science and Engineering, **Rayalaseema University College of Engineering**, Kurnool., for his indefatigable inspiration, guidance, cogent discussion and encouragement throughout this dissertation work.

I am grateful to **Mr.V. Satish Kumar, M. Tech, (Ph. D)** Coordinator of Computer Scienceand Engineering for his support and uninterrupted cooperation during my seminar work.

I am deeply indebted to my Incharge Principal & Registrar of RU **Dr. B. Vijaya Kumar Naidu Garu** for his constant support and valuable guidance was a source of inspiration for me.

This Acknowledgement will be incomplete without mentioning my sincere gratefulness to our Honorable Vice Chancellor **Prof. V. Venkata Basava Rao Garu**, who has been observed posing valiance in abundance, forwarding my individuality to acknowledge my project work tendentiously.

Last but not least, I wish to acknowledge my friends, family members and colleagues for giving moral strength and helping me to complete this dissertation.

NAME	(ROLL NO.)
AMPOLU SADVIK	**21RU1A0504**
ANNAPUREDDY HAMSA VARDHAN REDDY	**21RU1A0505**
GANDLA SIREESHA	**21RU1A0513**
PARVATHANENI LAKSHMAN RAO	**21RU1A0540**
NAGELLA JARUSHA ROSE	**22RU5A0505**

ABSTRACT

ABSTRACT

Phishing is an electronic fraud through which an attacker can access user credentials. Phishing websites are the ones that mimic legitimate websites. Fraudsters can replace them within hours to evade their detection. The effects of phishing attacks exhibit the need for anti-phishing mechanisms.

Several approaches were there to recognize the phishing websites, the white list approach, blacklist approach, machine learning, and heuristic-based approach. Earlier studies have shown that classifiers may be subject to evasion attacks although this point has only been explored on a small scale.

As a result, the study covers evasion attacks and their detection within the context of website classifiers, which is rarely explored. In response to the inadequacies, the proposed technique includes extracting information from URLs and classifying webpages using various machine learning methods. The methodology involves crafting adversarial samples targeting classification features, with a focus on maintaining the functionality and appearance of phishing websites.

TABLE OF CONTENTS

INTRODUCTION

1. INTRODUCTION

Phishing is now more dangerous to online security and in today's world, 50% of all emails include a bogus link that leads to fake sites. These websites may be fraudulent, in which attackers transmit phony emails or utilize fake websites which mimic the originals to get user credentials. Scammers employ various tactics to defraud people, including VOIP (Voice Over IP), fake websites, and covert redirection. The attacker develops counterfeit websites that are well-built and have a design that is comparable to real websites. The lack of awareness of end-users of security policies and the demerits of weaker web servers are also exploited by phishers. To mislead end-users that the fraudulent website is authentic, attackers acquire web security certifications. According to government data, phishing attempts have been on the rise for the past half-century.

Severe regulations are prohibiting such acts, and those who break them face penalties such as fines or jail. The security forces should undertake instant action toward these types of breaches because the phony websites may just exist a few times, as well as the adversary, may vanish into cyberspace after engaging in unlawful acts. The anti-phishing legislation wasenactedbytheIndiangovernmentunderSection43ofthe Computer Crime Act in 2000 [1] to prohibit such violations. Even if the legislation can help to avoid phishing attempts to some level, it cannot eliminate them. Anti-phishing tech nologies and heuristic- based approaches thus have evolved to minimize phishing assaults. There are several methods for detecting and avoiding phishing scams, and it is necessary to accurately forecast the assault to protect the victims. The most difficult problem is that the counterfeit website appears to be real, making it difficult to tell if it is a phishing site or not.

Government legislation was enacted to combat phishing assaults, while anti-phishing technologies like blacklists, and machine learning-based systems were developed to identify phishing websites. The classifiers based on machine learning

are excellent at finding those types of phishing sites still some of the classifiers are a risk of evasion attacks when set on the client-side. The evasion attack happens when the network is fed with adversarial examples and will generate as shown in Fig1. To a human, it will seem just like its uncompromised copy means the adversary creates an adversarial example for assault, which manipulated copy. Because existing assaults in this area destroy the functionality and look of websites, this sort of danger receives minimal attention.

1.1 OBJECTIVE

The primary objective of this study is to analyze and mitigate evasion attacks targeting machine learning-based web phishing classifiers. This includes examining adversarial tactics used to bypass detection mechanisms, exploring various machine learning approaches for phishing detection, and developing robust defense mechanisms. The study aims to enhance classifier resilience against adversarial examples while maintaining the accuracy and efficiency of phishing detection systems.

1.2 PROBLEM STATEMENT

Phishing attacks pose a significant threat to online security by deceiving users into providing sensitive credentials through fraudulent websites. While machine learning-based classifiers have proven effective in detecting phishing websites, they remain vulnerable to evasion attacks where adversaries manipulate website features to bypass detection. Existing anti-phishing techniques, such as blacklists and heuristic-based approaches, struggle to keep up with rapidly evolving attack strategies. This study addresses the challenge of enhancing phishing detection systems by investigating evasion attack techniques, assessing their impact on classifiers, and developing robust defense mechanisms to improve resilience against adversarial manipulation.

1.3 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Platform – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1) **Software : Anaconda**

2) **Primary Language : Python**
3) **Frontend Framework : Flask**
4) **Back-end Framework : Jupyter Notebook**
5) **Database : Sqlite3**
6) **Front-End Technologies : HTML,CSS,JavaScript and Bootstrap4**

1.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture – All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different

throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory – All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage – Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter – Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1) Operating System: Windows Only

2) Processor: i5 and above

3) Ram: 8gb and above

4) Hard Disk: 25 GB in local drive

FEASIBILITY STUDY

2. FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 Advanced Evasion Attacks and Mitigations on Practical ML-Based Phishing Website Classifiers:

2004.06954

ABSTRACT: —Machine learning (ML) based approaches have been the mainstream solution for anti-phishing detection. When they are deployed on the client-side, ML-based classifiers are vulnerable to evasion attacks, as shown by recent attacks. However, such potential threats have received relatively little attention because existing attacks destruct the functionalities or appearance of webpages and are conducted in the white-box scenario, making it less practical. Consequently, it becomes imperative to understand whether it is possible to launch evasion attacks with limited knowledge of the classifier, while preserving the functionalities and appearance. In this work, we show that even in the grey-, and black-box scenarios, evasion attacks are not only effective on practical ML-based classifiers, but can also be efficiently launched without destructing the functionalities and appearance. For this purpose, we propose three mutation-based attacks, differing in the knowledge of the target classifier, addressing a key technical challenge: automatically crafting an adversarial sample from a known phishing website in a way that can mislead classifiers. To launch attacks in the white- and grey-box scenarios, we also propose a sample-based collision attack to gain the knowledge of the target classifier. We demonstrate the effectiveness and efficiency of our evasion attacks on the state-of-the-art, Google's phishing page filter, achieved 100% attack success rate in less than one second per website. Moreover, the transferability attack on BitDefender's industrial phishing page classifier, TrafficLight, achieved up to 81.25% attack success rate. We further propose a similarity-based method to mitigate such evasion attacks, Pelican, which compares the similarity of an unknown website with recently detected phishing websites. We demonstrate that Pelican can effectively detect evasion attacks, hence could be integrated into ML-based classifiers. We also highlight two strategies of

classification rule selection to enhance the robustness of classifiers. Our findings contribute to design more robust phishing website classifiers in practice.

3.2 An Evasion Attack against ML-based Phishing URL Detectors

[2005.08454v1] An Evasion Attack against ML-based Phishing URL Detectors

ABSTRACT: Background: Over the year, Machine Learning Phishing URL classification (MLPU) systems have gained tremendous popularity to detect phishing URLs proactively. Despite this vogue, the security vulnerabilities of MLPUs remain mostly unknown. Aim: To address this concern, we conduct a study to understand the test time security vulnerabilities of the state-of-the-art MLPU systems, aiming at providing guidelines for the future development of these systems. Method: In this paper, we propose an evasion attack framework against MLPU systems. To achieve this, we first develop an algorithm to generate adversarial phishing URLs. We then reproduce 41 MLPU systems and record their baseline performance. Finally, we simulate an evasion attack to evaluate these MLPU systems against our generated adversarial URLs. Results: In comparison to previous works, our attack is: (i) effective as it evades all the models with an average success rate of 66% and 85% for famous (such as Netflix, Google) and less popular phishing targets (e.g., Wish, JBHIFI, Officeworks) respectively; (ii) realistic as it requires only 23ms to produce a new adversarial URL variant that is available for registration with a median cost of only $11.99/year. We also found that popular online services such as Google SafeBrowsing and VirusTotal are unable to detect these URLs. (iii) We find that Adversarial training (successful defence against evasion attack) does not significantly improve the robustness of these systems as it decreases the success rate of our attack by only 6% on average for all the models. (iv) Further, we identify the security vulnerabilities of the considered MLPU systems. Our findings lead to promising directions for future research. Conclusion: Our study not only illustrate vulnerabilities in MLPU systems but also highlights implications for future study towards assessing and improving these systems.

3.3 **Adversarial Sampling Attacks Against Phishing Detection**

:

(PDF) Adversarial Sampling Attacks Against Phishing Detection

ABSTRACT: Phishing websites trick users into believing that they are interacting with a legitimate website, and thereby, capture sensitive information, such as user names, passwords, credit card numbers and other personal information. Machine learning appears to be a promising technique for distinguishing between phishing websites and legitimate ones. However, machine learning approaches are susceptible to adversarial learning techniques, which attempt to degrade the accuracy of a trained classifier model. In this work, we investigate the robustness of machine learning based phishing detection in the face of adversarial learning techniques. We propose a simple but effective approach to simulate attacks by generating adversarial samples through direct feature manipulation. We assume that the attacker has limited knowledge of the features, the learning models, and the datasets used for training. We conducted experiments on four publicly available datasets on the Internet. Our experiments reveal that the phishing detection mechanisms are vulnerable to adversarial learning techniques. Specifically, the identification rate for phishing websites dropped to 70% by manipulating a single feature. When four features were manipulated, the identification rate dropped to zero percent. This result means that, any phishing sample, which would have been detected correctly by a classifier model, can bypass the classifier by changing at most four feature values; a simple effort for an attacker for such a big reward. We define the concept of vulnerability level for each dataset that measures the number of features that can be manipulated and the cost for each manipulation. Such a metric will allow us to compare between multiple defense models.

3.4 Phishing website detection using support vector machines and nature-inspired optimization algorithms

Phishing website detection using support vector machines and nature-inspired optimization algorithms | Telecommunication Systems

ABSTRACT: Phishing websites are amongst the biggest threats Internet users face today, and existing methods like blacklisting, using SSL certificates, etc. often fail to keep up with the increasing number of threats. This paper aims to utilise different properties of a website URL, and use a machine learning model to classify websites as phishing and non-phishing. These properties include the IP address length, the authenticity of the HTTPs request being sent by the website, usage of pop-up windows to enter data, Server Form Handler status, etc. A Support Vector Machine binary classifier trained on an existing dataset has been used to predict if a website was a legitimate website or not, by finding an optimum hyperplane to separate the two categories. This optimum hyperplane is found with the help of four optimization algorithms, the Bat Algorithm, the Firefly Algorithm, the Grey Wolf Optimiser algorithm and the Whale Optimization Algorithm, which are inspired by various natural phenomena. Amongst the four nature-inspired optimization algorithms, it has been determined that the Grey Wolf Optimiser algorithm's performance is significantly better than that of the Firefly Algorithm, but there is no significant difference while comparing the performance of any other pair of algorithms. However, all four nature-inspired optimization algorithms perform significantly better than the grid-search optimized Random Forest classifier model described in earlier research.

3.5 Automated Poisoning Attacks and Defenses in Malware Detection Systems: An Adversarial Machine Learning Approach

[1706.04146] Automated Poisoning Attacks and Defenses in Malware Detection Systems: An Adversarial Machine Learning Approach

ABSTRACT: The evolution of mobile malware poses a serious threat to smartphone security. Today, sophisticated attackers can adapt by maximally sabotaging machine-learning classifiers via polluting training data, rendering most recent machine learning-based malware detection tools (such as Drebin, DroidAPIMiner, and MaMaDroid) ineffective. In this paper, we explore the feasibility of constructing crafted malware samples; examine how machine-learning

classifiers can be misled under three different threat models; then conclude that injecting carefully crafted data into training data can significantly reduce detection accuracy. To tackle the problem, we propose KuafuDet, a two-phase learning enhancing approach that learns mobile malware by adversarial detection. KuafuDet includes an offline training phase that selects and extracts features from the training set, and an online detection phase that utilizes the classifier trained by the first phase. To further address the adversarial environment, these two phases are intertwined through a self-adaptive learning scheme, wherein an automated camouflage detector is introduced to filter the suspicious false negatives and feed them back into the training phase. We finally show that KuafuDet can significantly reduce false negatives and boost the detection accuracy by at least 15%. Experiments on more than 250,000 mobile applications demonstrate that KuafuDet is scalable and can be highly effective as a standalone system.

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

Several solutions come into effect for finding phishing websites in which most of them couldn't make decisions perfectly and accurately. This includes the anti-phishing approaches and methods adopted in establishing solutions to decrease the false positive rate. Mainly there are white-list approaches, black-list approaches, and machine learning-based approaches in the classification method of phishing websites. Blacklist techniques and white-list are ineffective in detecting new phishing attacks and have a high percentage of false positives. Furthermore, machine learning-based approaches collect characteristics from third-party sources, such as a search engine. As a result, they are complex, sluggish, and unsuitable for real-time environments.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. Low Accuracy.
2. More time taking process.

4.2 Proposed System:

In this study, we employed a dataset collected from two primary sources, namely Phishtank and the Alexa dataset, to investigate evasion attacks on machine learning-based web phishing classifiers. We extracted a set of features from these URLs. These features were carefully chosen based on their relevance to phishing detection and have been established as effective indicators for classifying phishing websites. Using the extracted features built some classifiers to classify phishing and legitimate sites. The new perturbed samples are produced depending on the current phishing cases recognized by the classifier,

replicating the attacker's method. The test pages are checked using the target classifier. That is give the URL as an input and the classifier will classify the adversarial site as legitimate. While implementing the classifiers such as k-nearest neighbor, logistic regression, SVM, decision tree and random forest

4.2.1 Advantages of proposed system:

1. High Accuracy.
2. Less time taking process.
3. Better prediction.

4.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modiling
5. Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non-functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement

- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

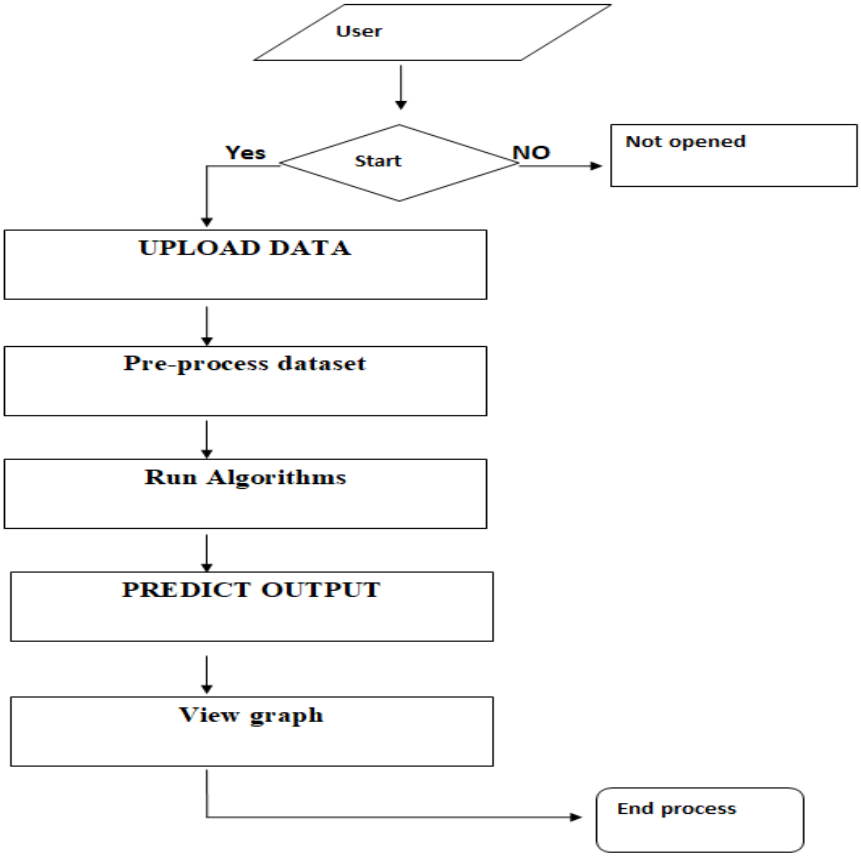
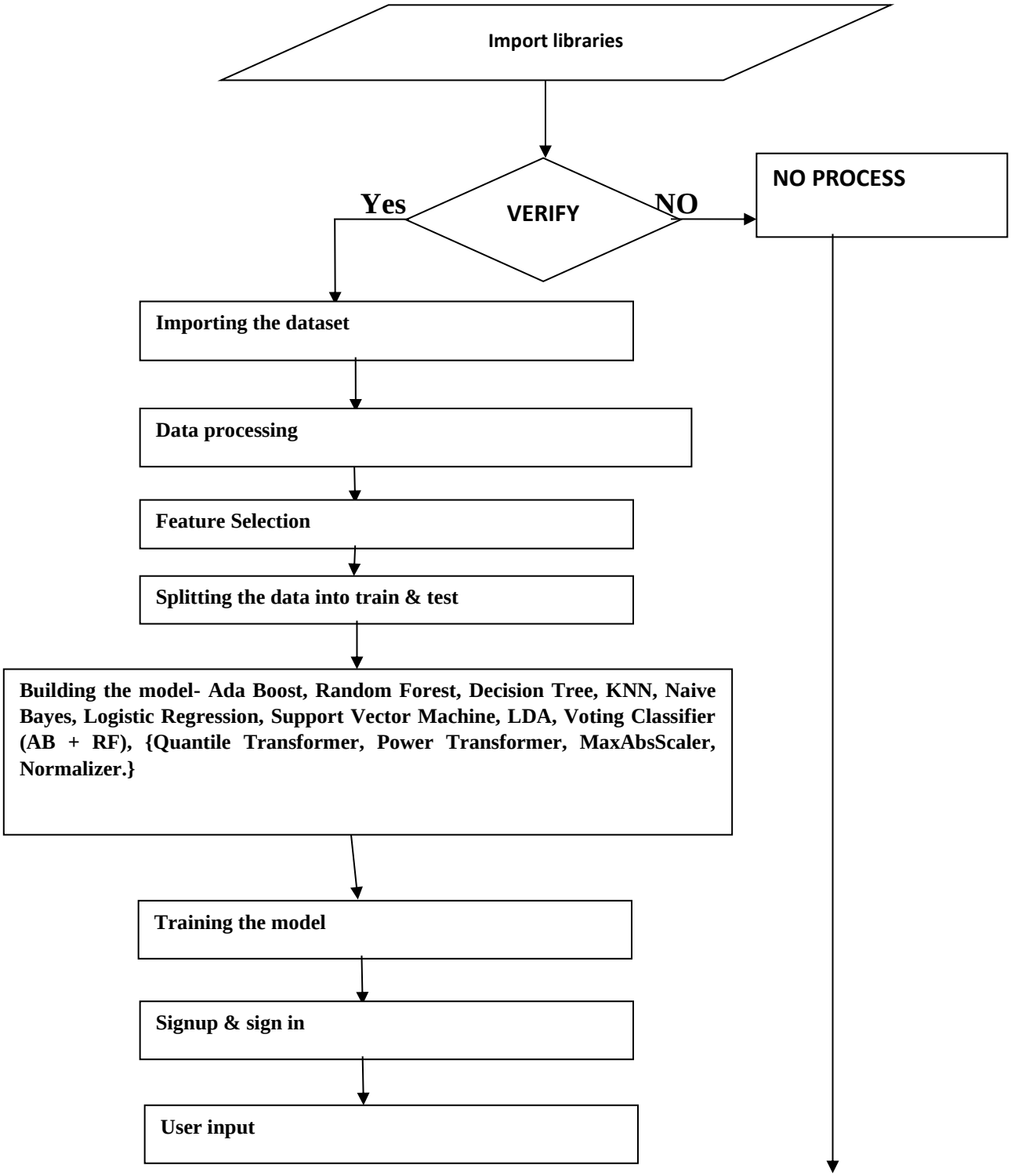


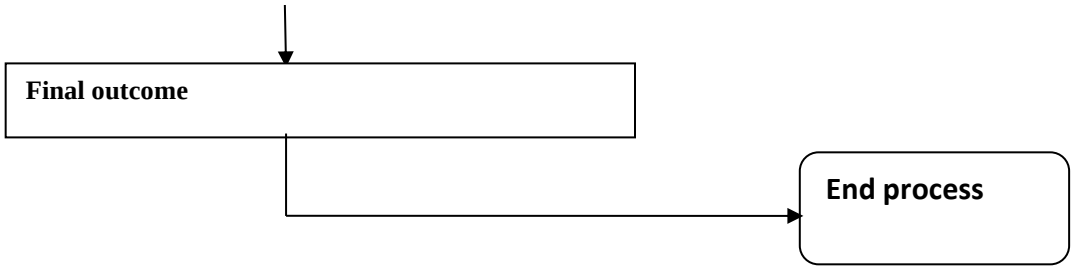
Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

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

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.





5.2 UML DIAGRAMS

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

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

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

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

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

GOALS:

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

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.

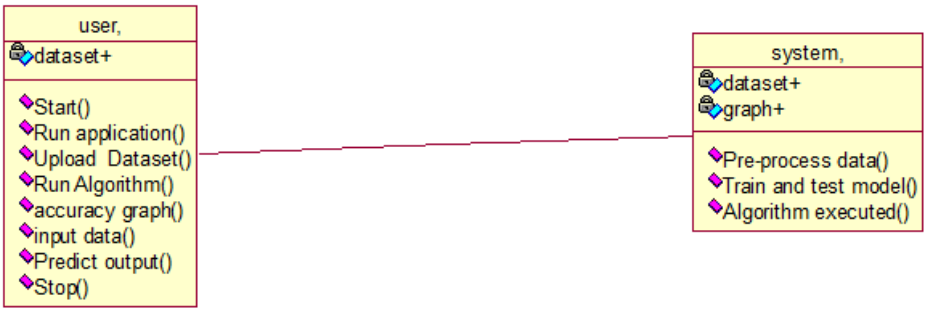
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Use case diagram:

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

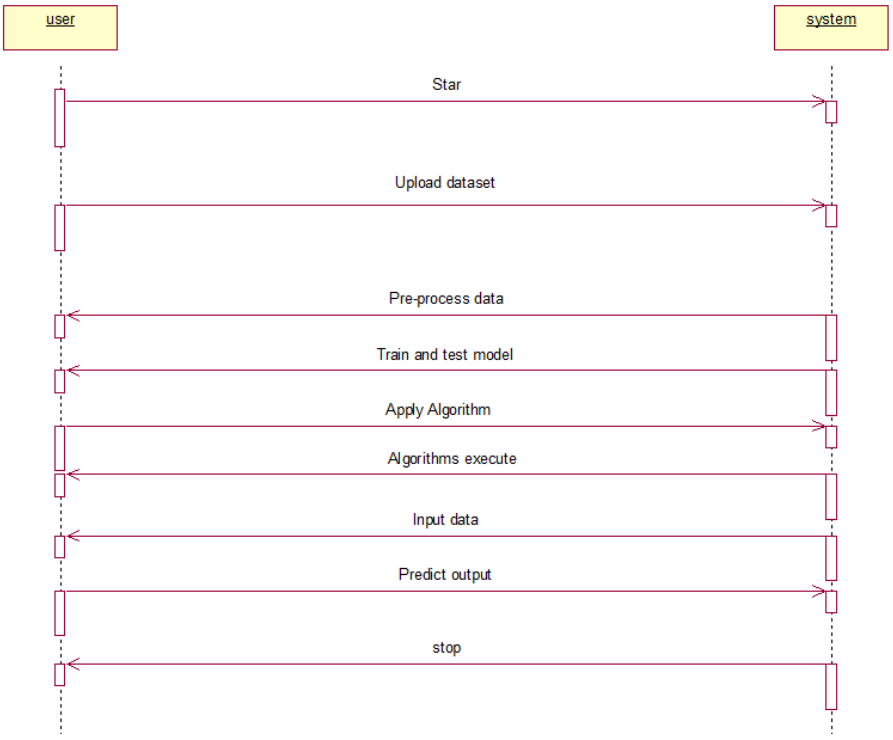
Class diagram:

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



Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

IMPLEMENTATION

6. IMPLEMENTATION

MODULES:

Gathering the datasets: We gather all the r data from the kaggale website and upload to the proposed model

Generate Train & Test Model: We have to preprocess the gathered data and then we have to split the data into two parts training data with 80% and test data with 20%

Run Algorithms: For prediction apply the machine learning models on the dataset by splitting the datasets in to 70 to 80 % of training with these models and 30 t0 20 % of testing for predicting

Obtain the accuracy: In this module we will get accuracies

Predict output: in this module we will predict output based on input data

Algorithms:

Voting Classifier: An ensemble method that combines the predictions of multiple models (like Logistic Regression, Decision Trees, etc.) by majority voting (for classification) or averaging (for regression) to improve overall performance.

Random Forest: An ensemble of decision trees, where each tree is trained on a random subset of data and features. It reduces overfitting and improves accuracy by averaging the results of individual trees.

Decision Tree: A model that splits data into branches based on feature conditions, creating a tree-like structure. Each node represents a decision rule, and leaves represent outcomes.

K-Nearest Neighbors (KNN): A non-parametric method that classifies data points based on the majority class among their

k

k nearest neighbors in the feature space.

Support Vector Machine (SVM): A supervised learning algorithm that finds the optimal hyperplane that maximizes the margin between different classes. It can use kernels to handle non-linear data.

Logistic Regression: A statistical model that uses a logistic function to model the probability of a binary outcome. It's widely used for classification tasks.

6.2 SAMPLE CODE:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load dataset
data = load_iris()
X = data.data
y = data.target

# Preprocessing: Standardize the data
scaler = StandardScaler ()
X_scaled = scaler.fit_transform(X)

# Split into train and test sets (80% train, 20% test)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Define individual models
model_rf = RandomForestClassifier (n_estimators=100, random_state=42)
model_dt = DecisionTreeClassifier(random_state=42)
model_knn = KNeighborsClassifier(n_neighbors=5)
model_svm = SVC (probability=True, kernel='rbf')
model_lr = LogisticRegression()

# Voting Classifier (soft voting for probabilistic averaging)
voting_clf = VotingClassifier(estimators=[
   ('rf', model_rf),
   ('dt', model_dt),
   ('knn', model_knn),
   ('svm', model_svm),
   ('lr', model_lr)
], voting='soft')

# Train models
voting_clf.fit(X_train, y_train)

# Predict and evaluate
y_pred = voting_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Voting Classifier Accuracy:", accuracy)
```

SOFTWARE ENVIRONMENT

7. SOFTWARE ENVIRONMENT

MACHINE LEARNING:

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

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

Challenges in Machines Learning:-

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

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

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

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

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of over fitting & under fitting − If the model is over fitting or under fitting, it cannot be represented well for the problem.

Curse of dimensionality − another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

DEEP LEARNING

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company

that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

What is Anaconda for Python?

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

How to install Anaconda for Python?

To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

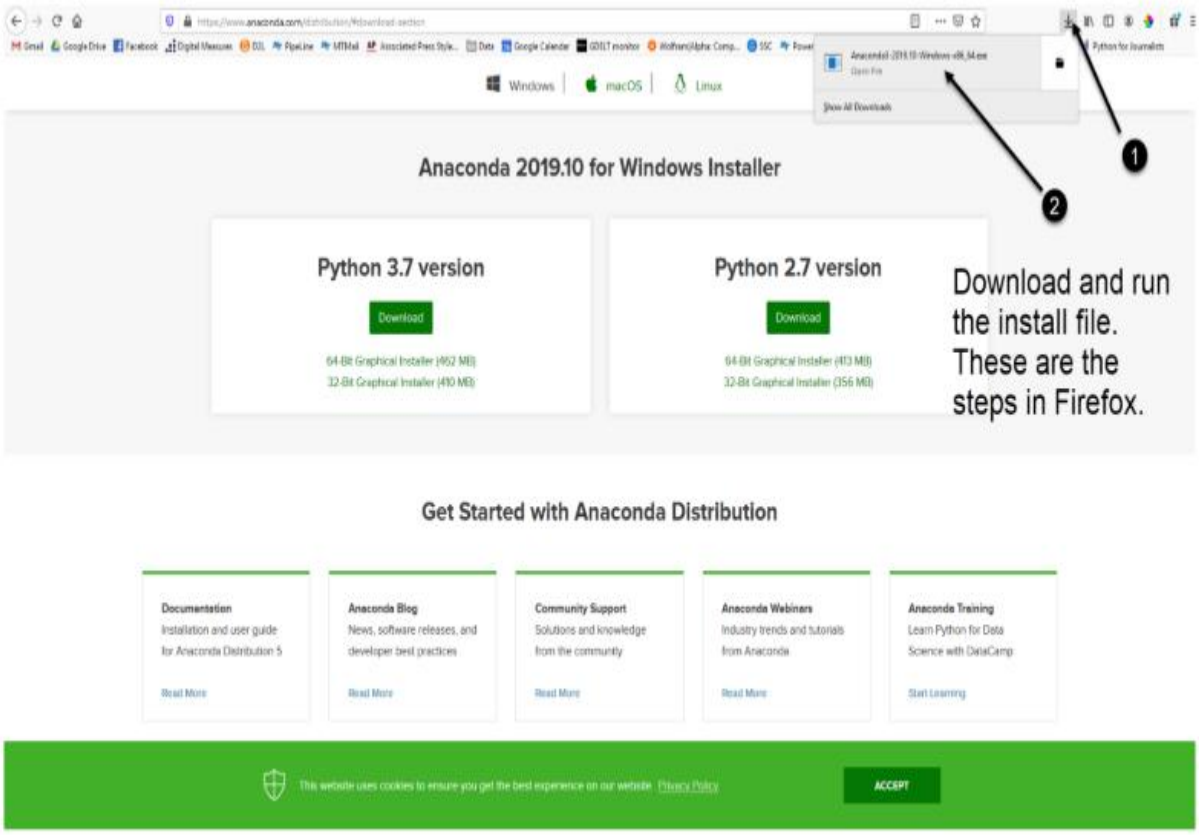
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

Python Anaconda Installation

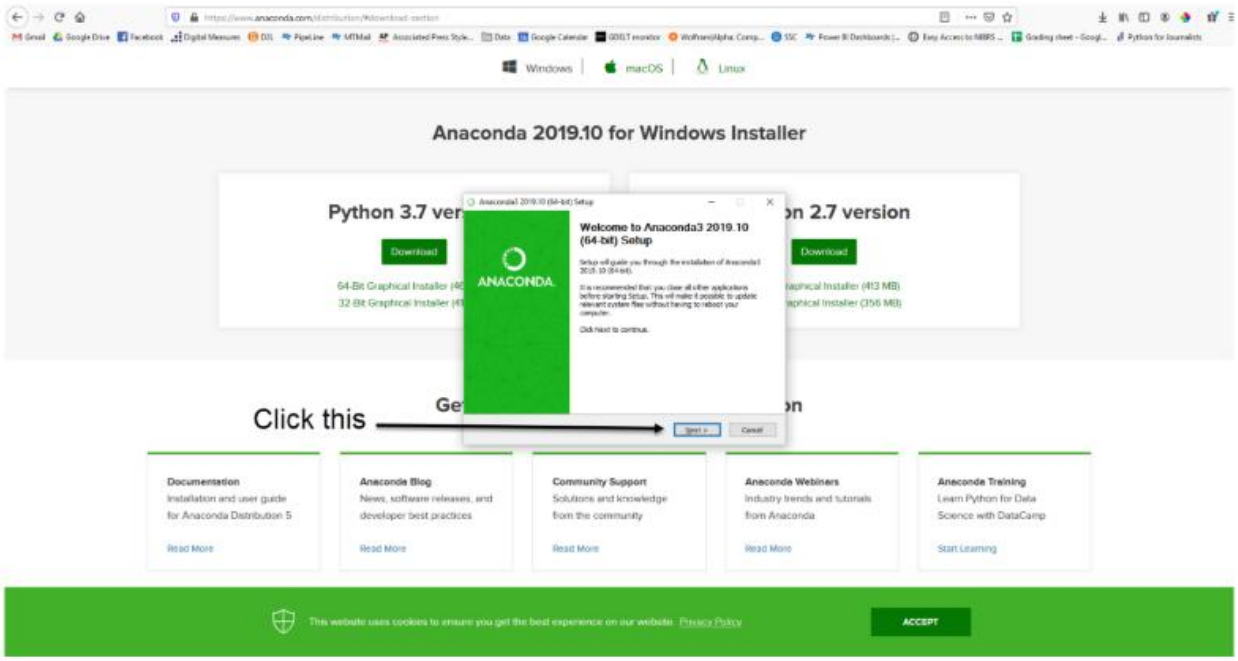
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – Download anaconda

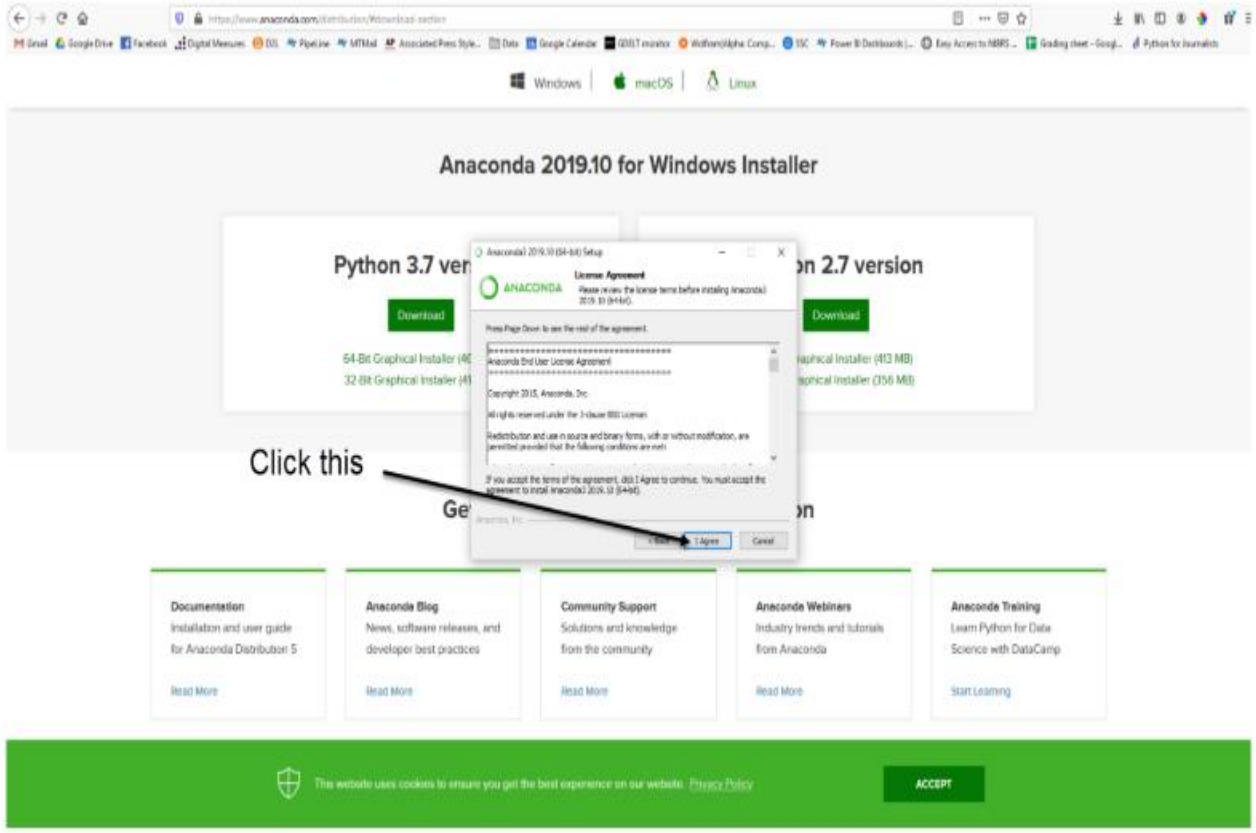


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

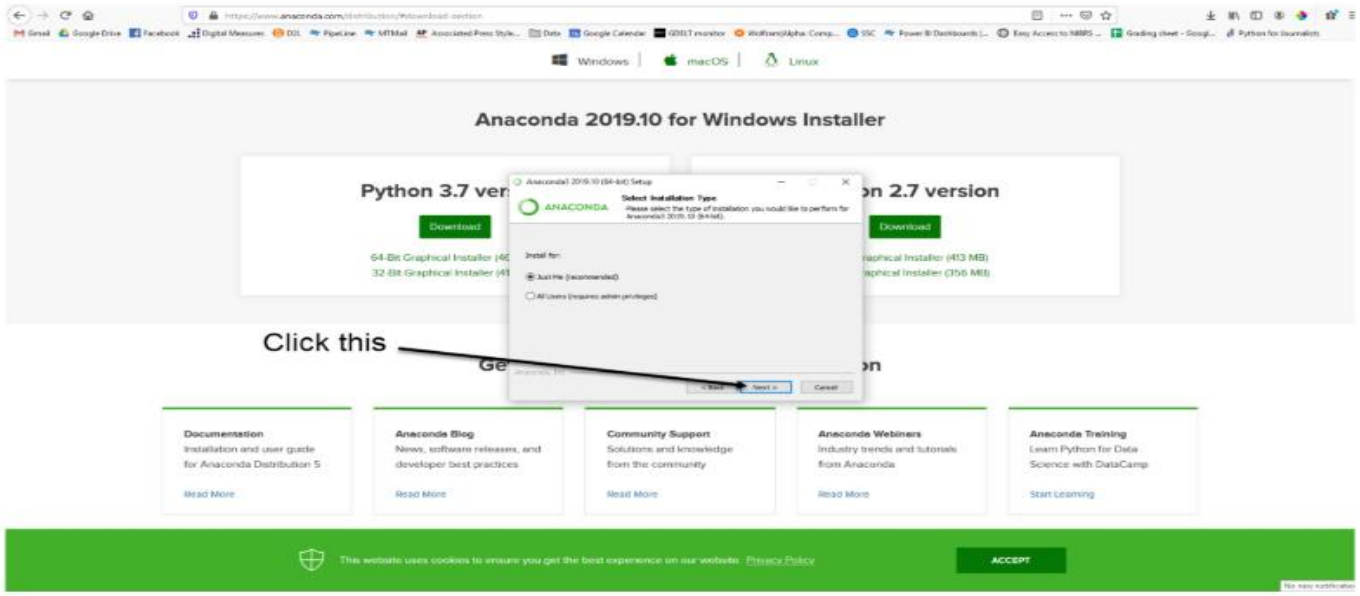
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



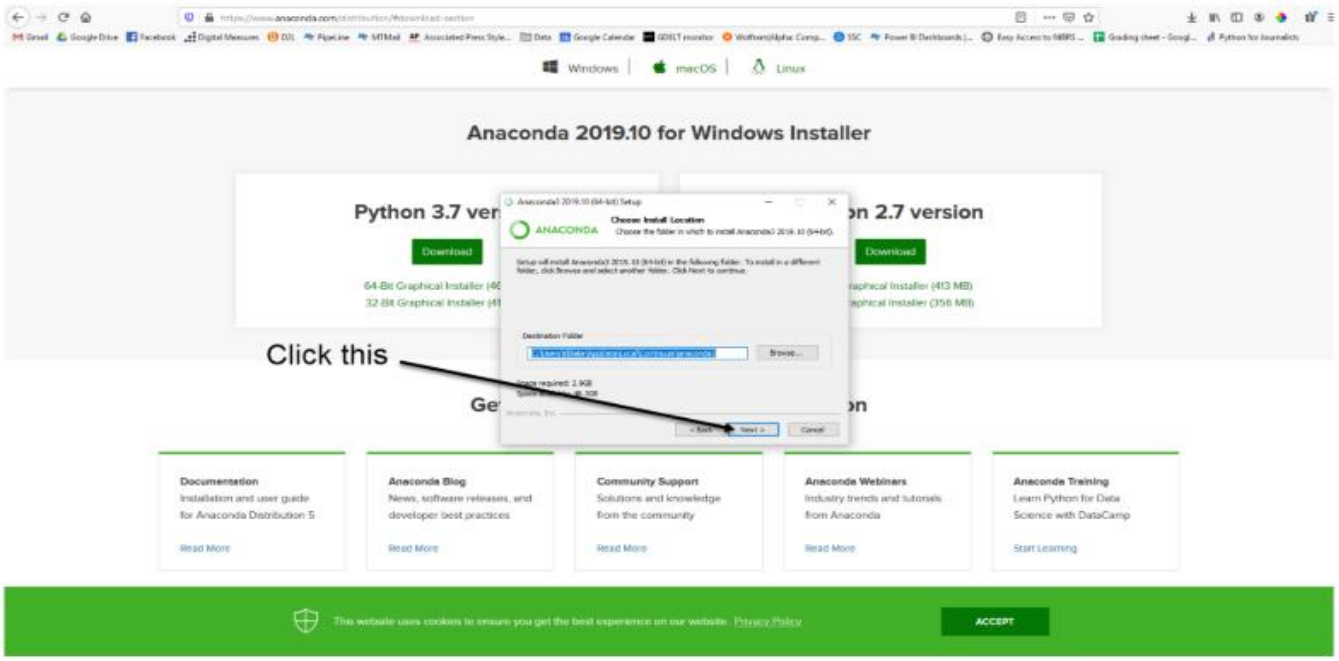
3. Now, you'll see the license agreement. Click on 'I Agree'.



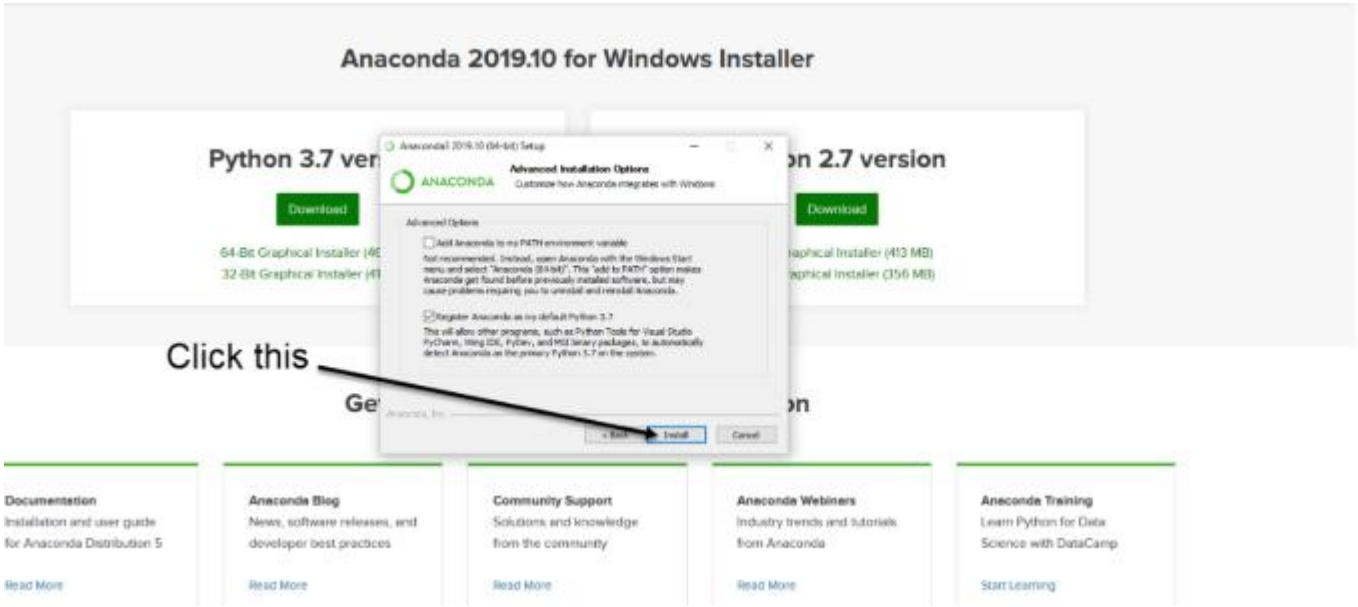
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



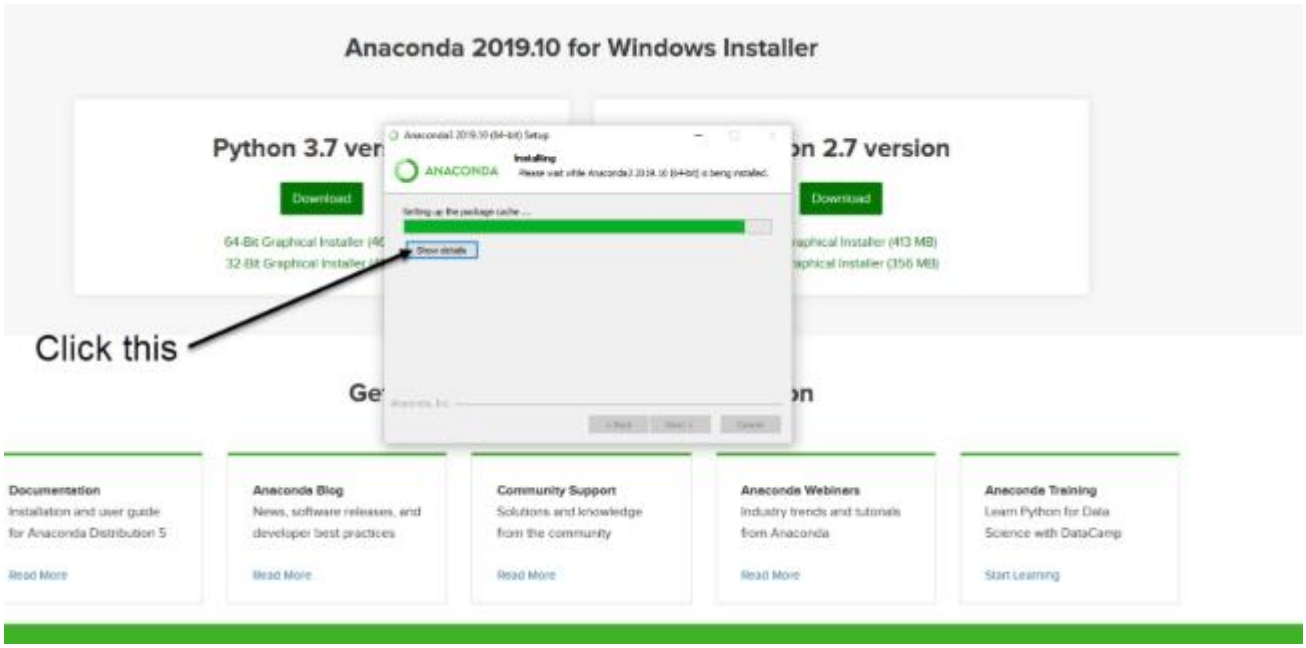
5. Choose where you want to install it. Here, you can see the available space and how much you need.



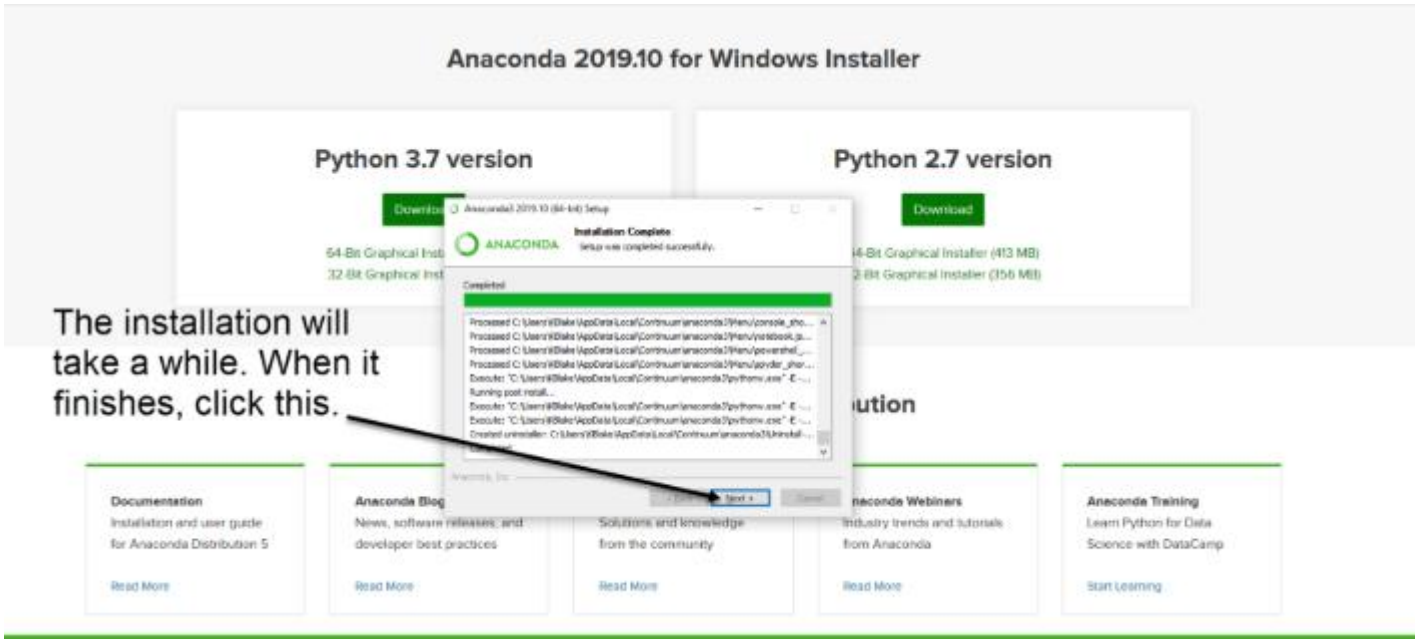
6. Now, you'll get some advanced options. You can add Anaconda to your system's PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on 'Install'.



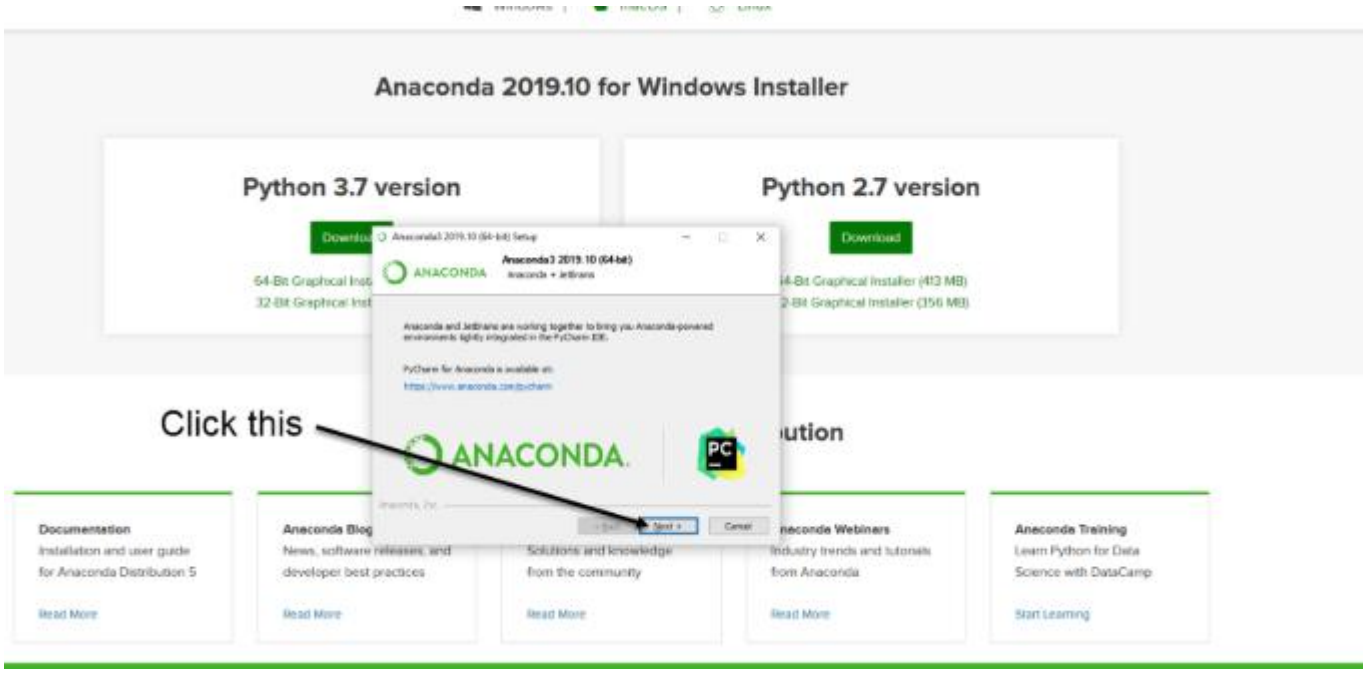
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



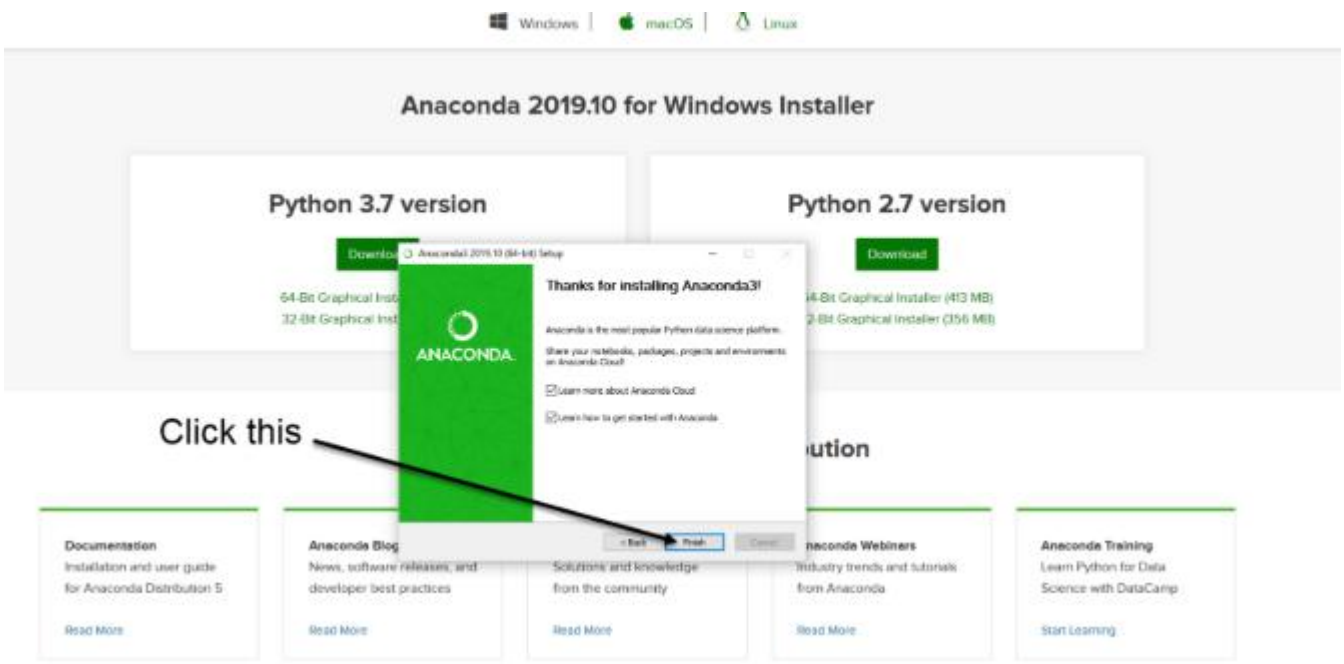
8. The installation is complete. Click Next.



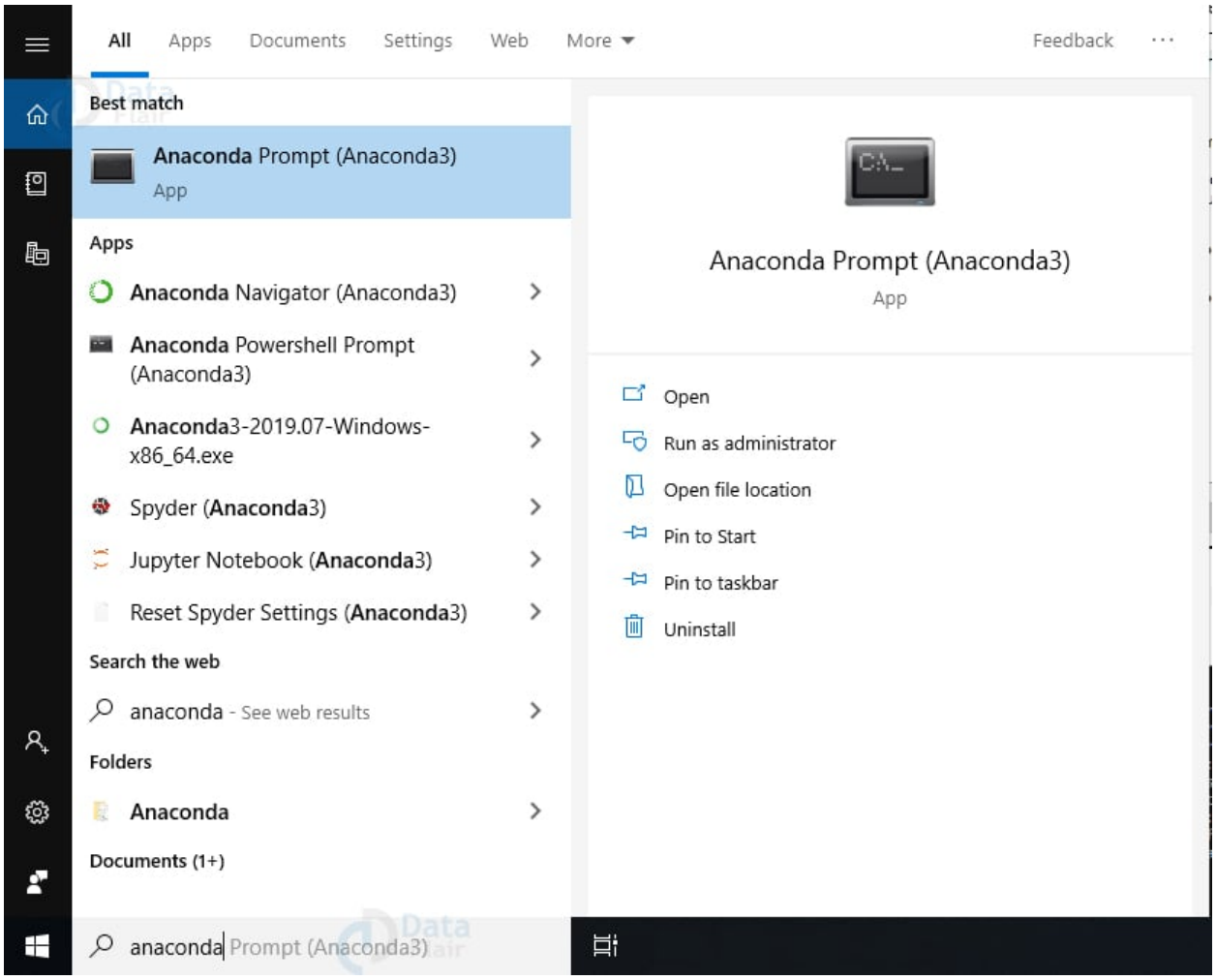
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:



PYTHON LANGUAGE:

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

Features in Python:

There are many features in Python, some of which are discussed below as follows:

1. Free and Open Source

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

2. Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-Oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Extensible feature

Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an integrated language

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called byte code.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

14. Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

LIBRARIES/PACKGES:-

Tensor flow

Tensor Flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library,

and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

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

Numpy

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

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

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

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

Pandas

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

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

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

Scikit – learn

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

CODE

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import joblib

import sqlite3


import numpy as np

import pandas as pd
```

```
from sklearn import metrics

import warnings

import pickle

warnings.filterwarnings('ignore')

from feature import FeatureExtraction


import pandas as pd

import numpy as np

import pickle

import sqlite3

import random


import smtplib

from email.message import EmailMessage

from datetime import datetime


app = Flask(__name__)


file = open("model.pkl","rb")

gbc = pickle.load(file)

file.close()
```

```
@app.route('/index')
def index():
    return render_template('index.html')



@app.route("/url", methods=["GET", "POST"])
def url():
    if request.method == "POST":

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)


        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
```

```
        return render_template('result.html',xx
=round(y_pro_non_phishing,2),url=url )

    return render_template("index.html", xx =-1)


@app.route("/about")

def about():

    return render_template("about.html")



@app.route('/')

@app.route('/home')

def home():

        return render_template('home.html')


@app.route('/logon')

def logon():

        return render_template('signup.html')


@app.route('/login')

def login():

        return render_template('signin.html')
```

```
@app.route("/signup")
def signup():
    global otp, username, name, email, number, password
    username = request.args.get('user','')
    name = request.args.get('name','')
    email = request.args.get('email','')
    number = request.args.get('mobile','')
    password = request.args.get('password','')
    otp = random.randint(1000,5000)
    print(otp)
    msg = EmailMessage()
    msg.set_content("Your OTP is : "+str(otp))
    msg['Subject'] = 'OTP'
    msg['From'] = "evotingotp4@gmail.com"
    msg['To'] = email



    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login("evotingotp4@gmail.com", "xowpojqyiygprhgr")
```

```
    s.send_message(msg)
    s.quit()
    return render_template("val.html")


@app.route('/predict1', methods=['POST'])
def predict1():
    global otp, username, name, email, number, password
    if request.method == 'POST':
        message = request.form['message']
        print(message)
        if int(message) == otp:
            print("TRUE")
            con = sqlite3.connect('signup.db')
            cur = con.cursor()
            cur.execute("insert into `info` (`user`,`email`,
`password`,`mobile`,`name`) VALUES (?, ?, ?, ?,
?)",(username,email,password,number,name))
            con.commit()
            con.close()
            return render_template("signin.html")
    return render_template("signup.html")
```

```
@app.route("/signin")
def signin():

    mail1 = request.args.get('user','')
    password1 = request.args.get('password','')
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("select `user`, `password` from info where `user` = ? AND `password` = ?",(mail1,password1,))
    data = cur.fetchone()

    if data == None:
        return render_template("signin.html")

    elif mail1 == str(data[0]) and password1 == str(data[1]):
        return render_template("index.html")
    else:
        return render_template("signin.html")

@app.route("/notebook1")
def notebook1():
    return render_template("AlexaData.html")
```

```
@app.route("/notebook2")
def notebook2():
    return render_template("PhishTank.html")



if __name__ == "__main__":
    app.run()
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Register V1</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!--
===============================================================
=====================================-->
```

<link rel="icon" type="image/png"
href="static/assets1/images/icons/favicon.ico"/>

<!--
===
===-->

<link rel="stylesheet" type="text/css"
href="static/assets1/vendor/bootstrap/css/bootstrap.min.css">

<!--
===
===-->

<link rel="stylesheet" type="text/css" href="static/assets1/fonts/font-awesome-4.7.0/css/font-awesome.min.css">

<!--
===
===-->

<link rel="stylesheet" type="text/css"
href="static/assets1/vendor/animate/animate.css">

<!--
===
===-->

<link rel="stylesheet" type="text/css" href="static/assets1/vendor/css-hamburgers/hamburgers.min.css">

<!--
===
===-->

<link rel="stylesheet" type="text/css"
href="static/assets1/vendor/select2/select2.min.css">

<!--
===
===-->

<link rel="stylesheet" type="text/css" href="static/assets1/css/util.css">

<link rel="stylesheet" type="text/css"
href="static/assets1/css/main.css">

<!--
===
===-->

</head>

<body>

<div class="login100-pic js-tilt" data-tilt>

<img src="static/assets1/images/img-01.png"
alt="IMG">

<form class="login100-form validate-form"
action="/signup" method="GET">

```
<span class="login100-form-title">
Member Register
</span>

<div class="wrap-input100 validate-input"
data-validate = "Valid email is required: ex@abc.xyz">
<input class="input100" type="text"
name="user" placeholder="USERNAME" required>
<span class="focus-input100"></span>
<span class="symbol-input100">
<i class="fa fa-user" aria-
hidden="true"></i>
</span>
</div>

<div class="wrap-input100 validate-input"
data-validate = "Valid email is required: ex@abc.xyz">
<input class="input100" type="text"
name="name" placeholder="NAME" required>
<span class="focus-input100"></span>
<span class="symbol-input100">
<i class="fa fa-user" aria-
hidden="true"></i>
```

```
                    </span>

                </div>


                <div class="wrap-input100 validate-input"
data-validate = "Valid email is required: ex@abc.xyz">

                    <input class="input100" type="text"
name="email" placeholder="EMAIL" required>

                    <span class="focus-input100"></span>

                    <span class="symbol-input100">

                        <i class="fa fa-envelope" aria-
hidden="true"></i>

                    </span>

                </div>


                <div class="wrap-input100 validate-input"
data-validate = "Valid email is required: ex@abc.xyz">

                    <input class="input100" type="text"
name="mobile" placeholder="MOBILE" required>

                    <span class="focus-input100"></span>

                    <span class="symbol-input100">

                        <i class="fa fa-mobile" aria-
hidden="true"></i>

                    </span>
```

```
                </div>

                <div class="wrap-input100 validate-input"
            data-validate = "Password is required">
                    <input class="input100"
        type="password" name="password" placeholder="PASSWORD" required>
                    <span class="focus-input100"></span>
                    <span class="symbol-input100">
                        <i class="fa fa-lock" aria-
                    hidden="true"></i>
                    </span>
                </div>

                <div class="container-login100-form-btn">
                    <button class="login100-form-btn">
                        Register
                    </button>
                </div>

                <div class="text-center p-t-136">
```


Did you an Account

</form>

<!--
==========
==========-->

<script src="static/assets1/vendor/jquery/jquery-3.2.1.min.js"></script>

<!--
==========
==========-->

<script src="static/assets1/vendor/bootstrap/js/popper.js"></script>

<script

src="static/assets1/vendor/bootstrap/js/bootstrap.min.js"></script>

<!--

==========

==========-->

<script src="static/assets1/vendor/select2/select2.min.js"></script>

<!--

==========

==========-->

<script src="static/assets1/vendor/tilt/tilt.jquery.min.js"></script>

<script >

$('.js-tilt').tilt({

scale: 1.1

})

</script>

<!--

==========

==========-->

<script src="static/assets1/js/main.js"></script>

</body>

</html>

SYSTEM TESTING

8. SYSTEM TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

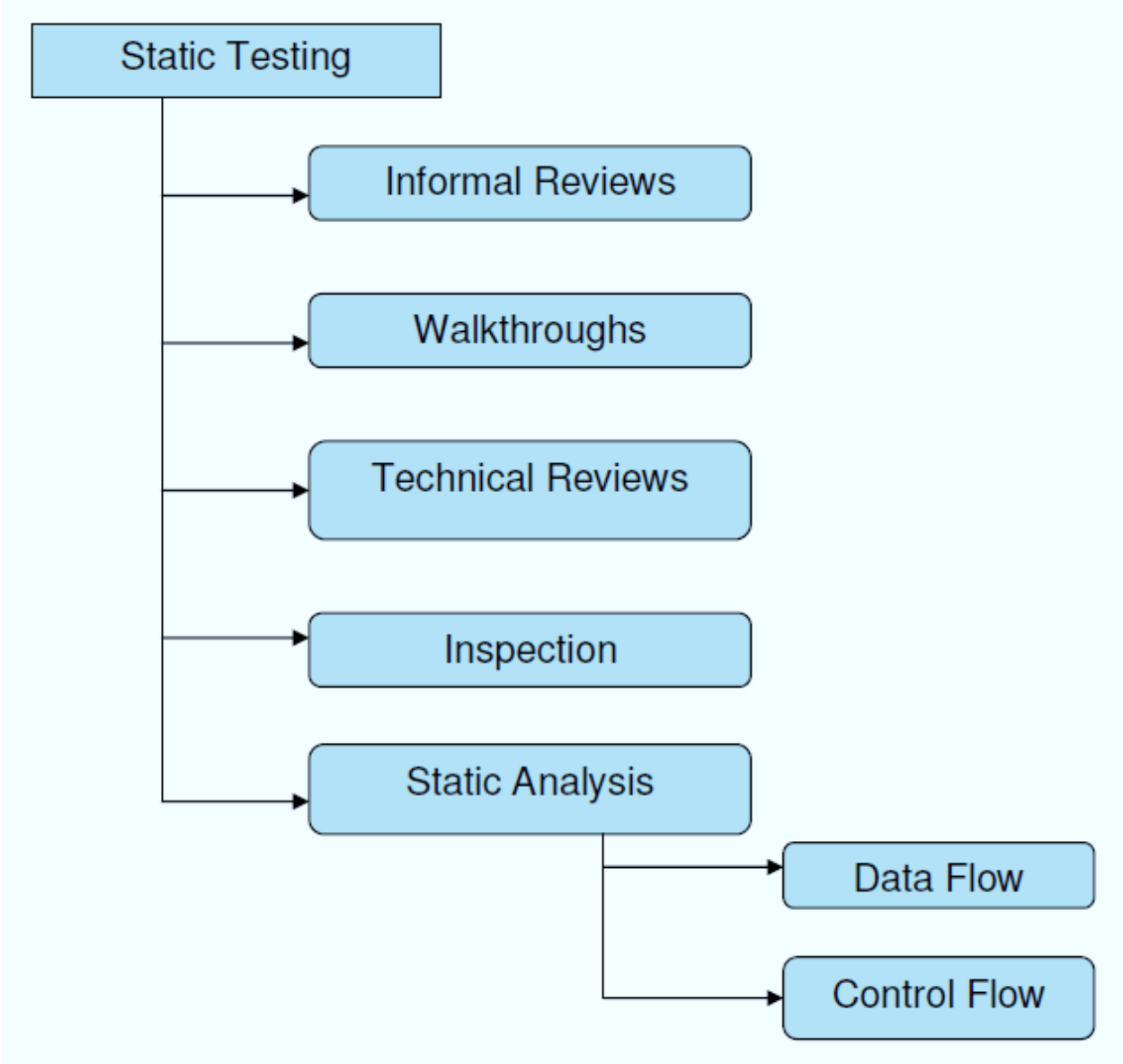
8.1Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

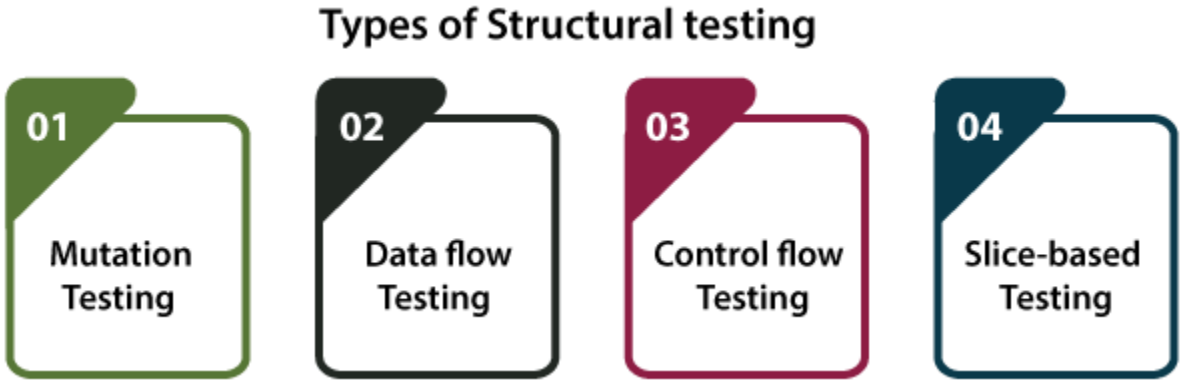
The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at

the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



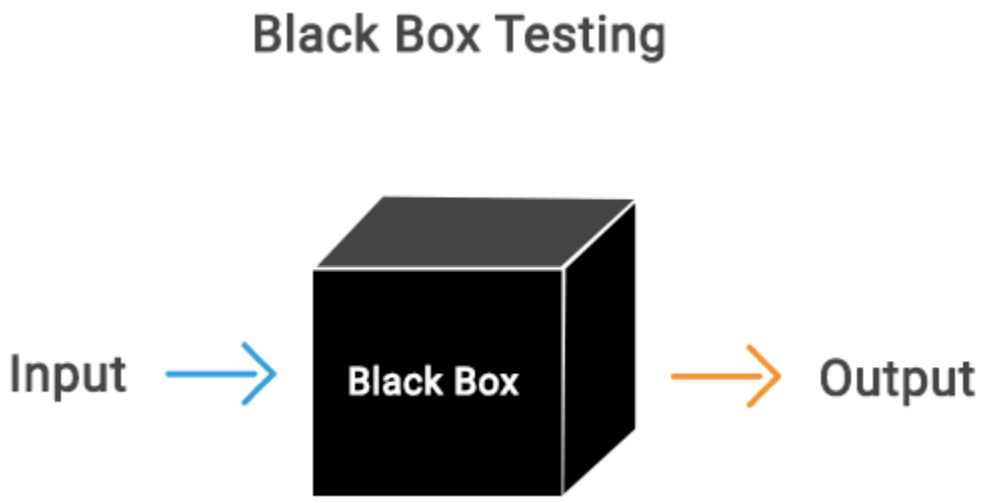
Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).



Behavioral Testing:

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

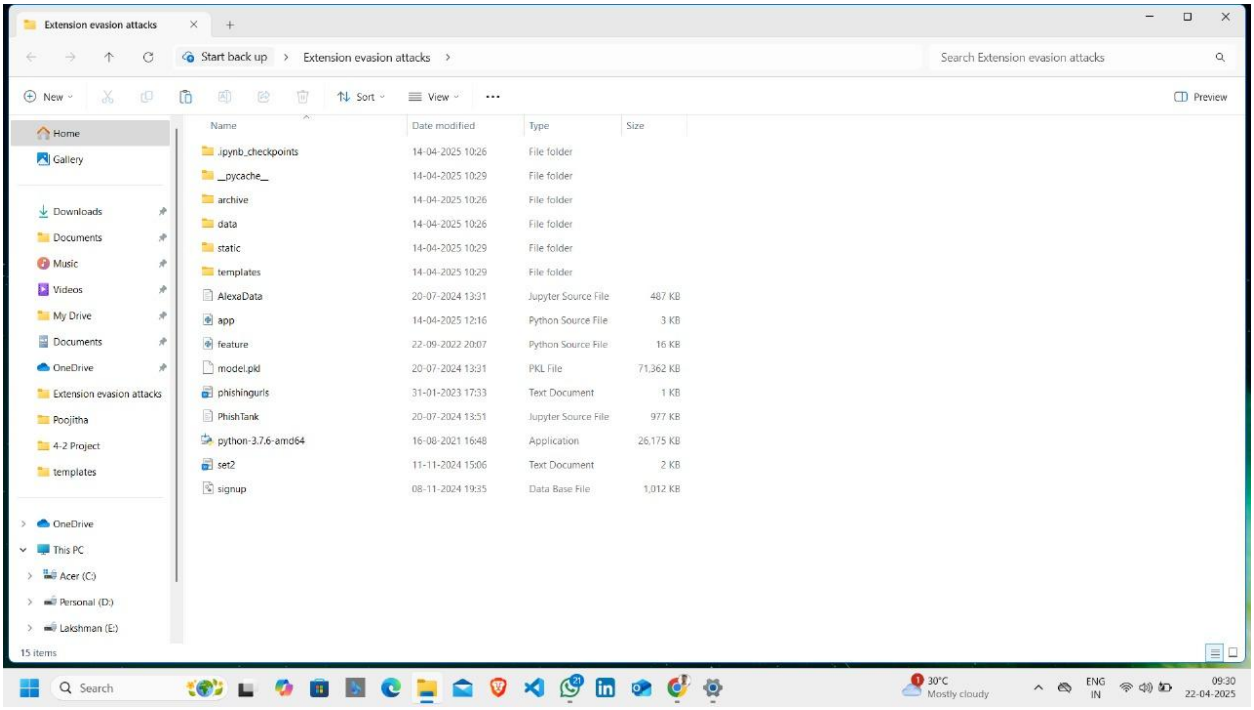


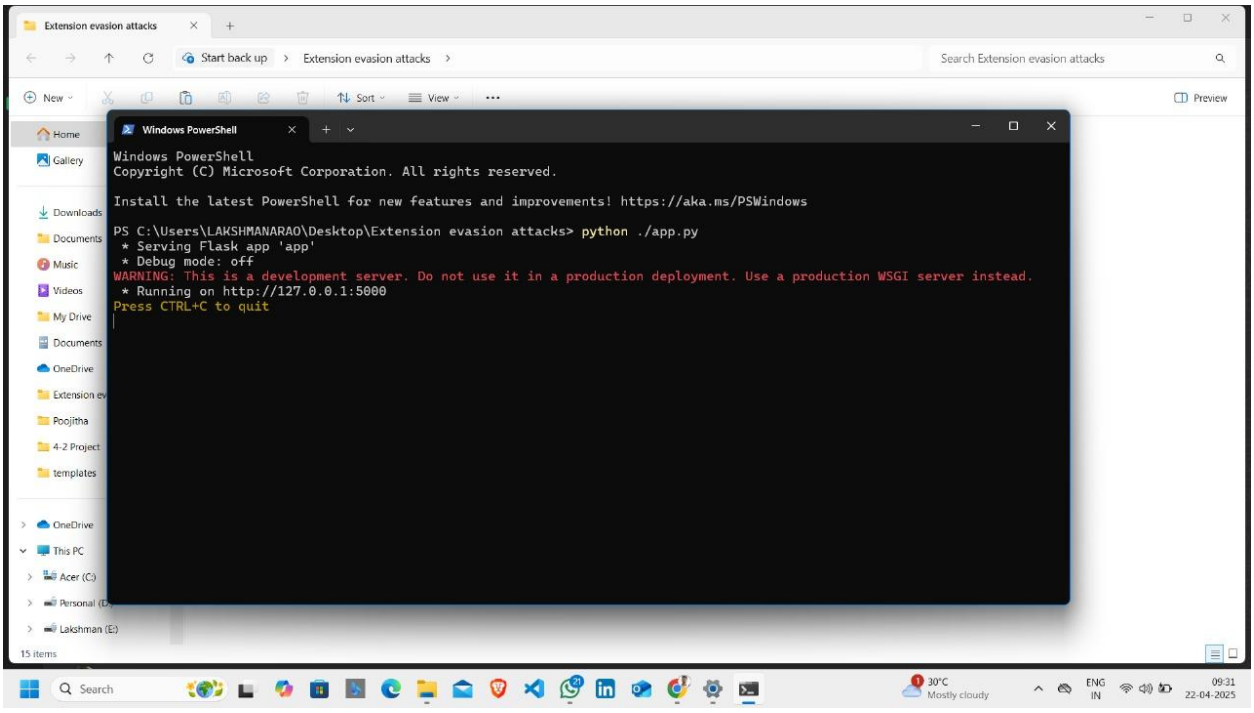
8.2 TEST CASES:

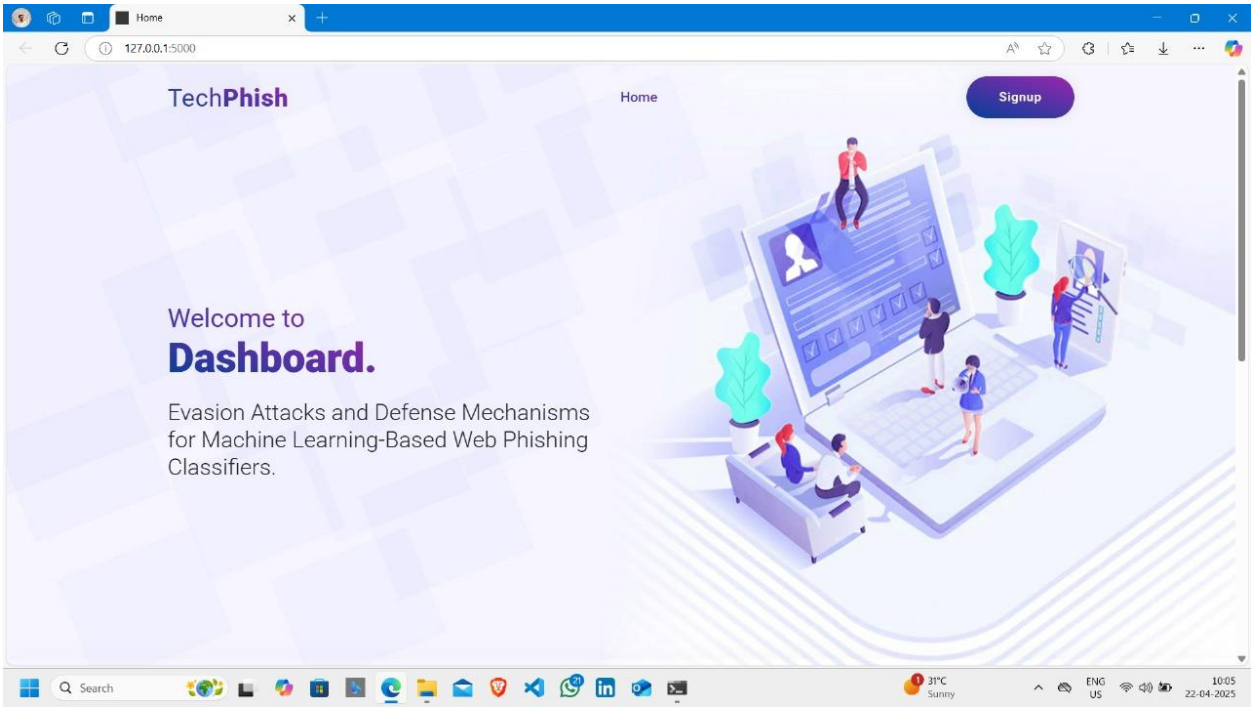
S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User sign in	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

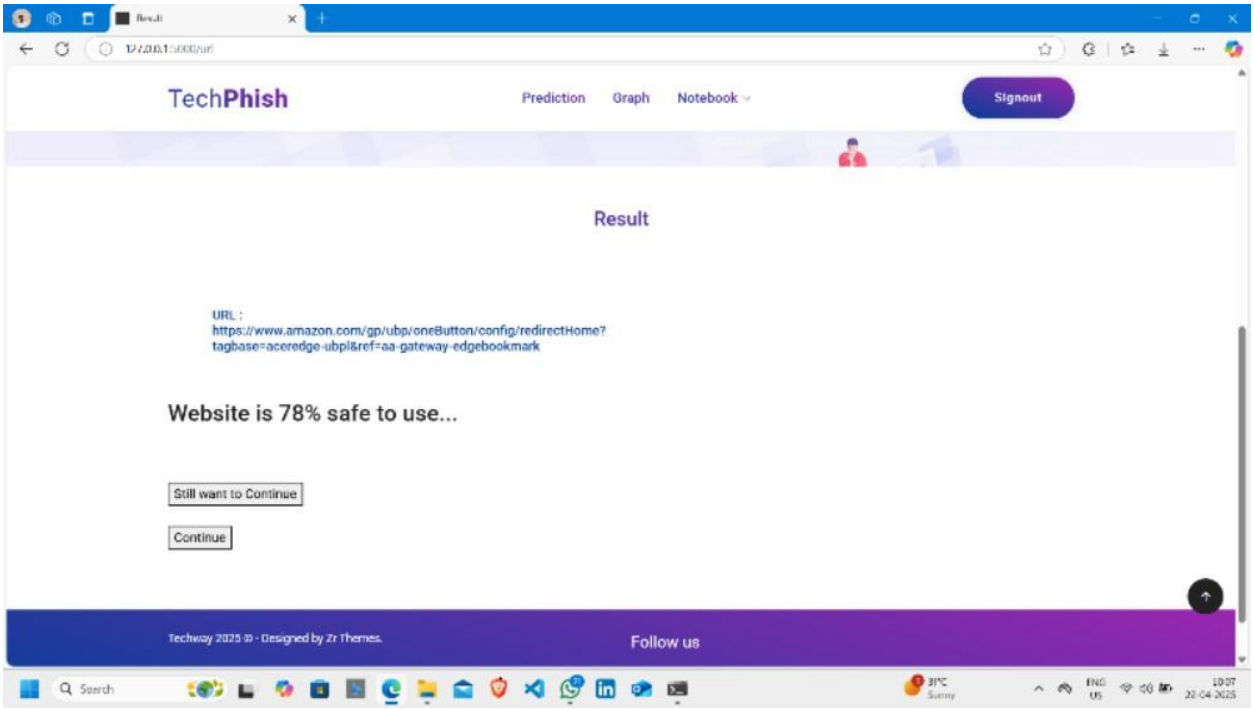
SCREENS

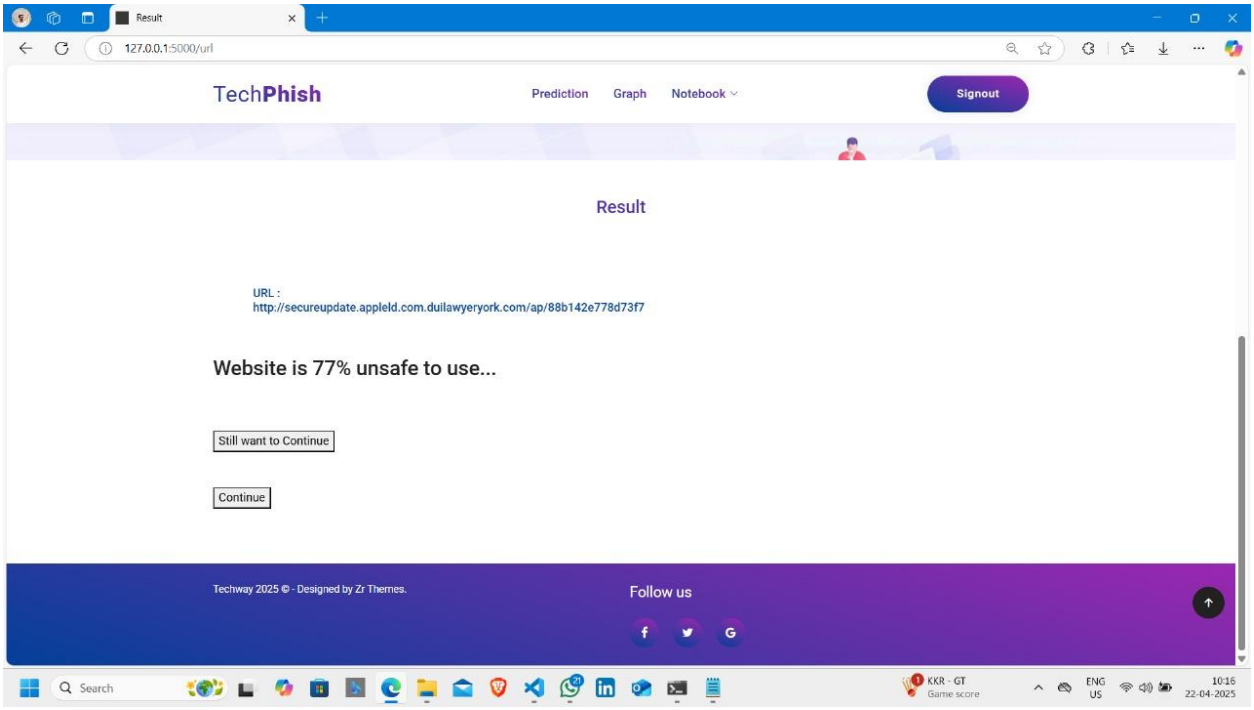
7. SCREENSHOTS

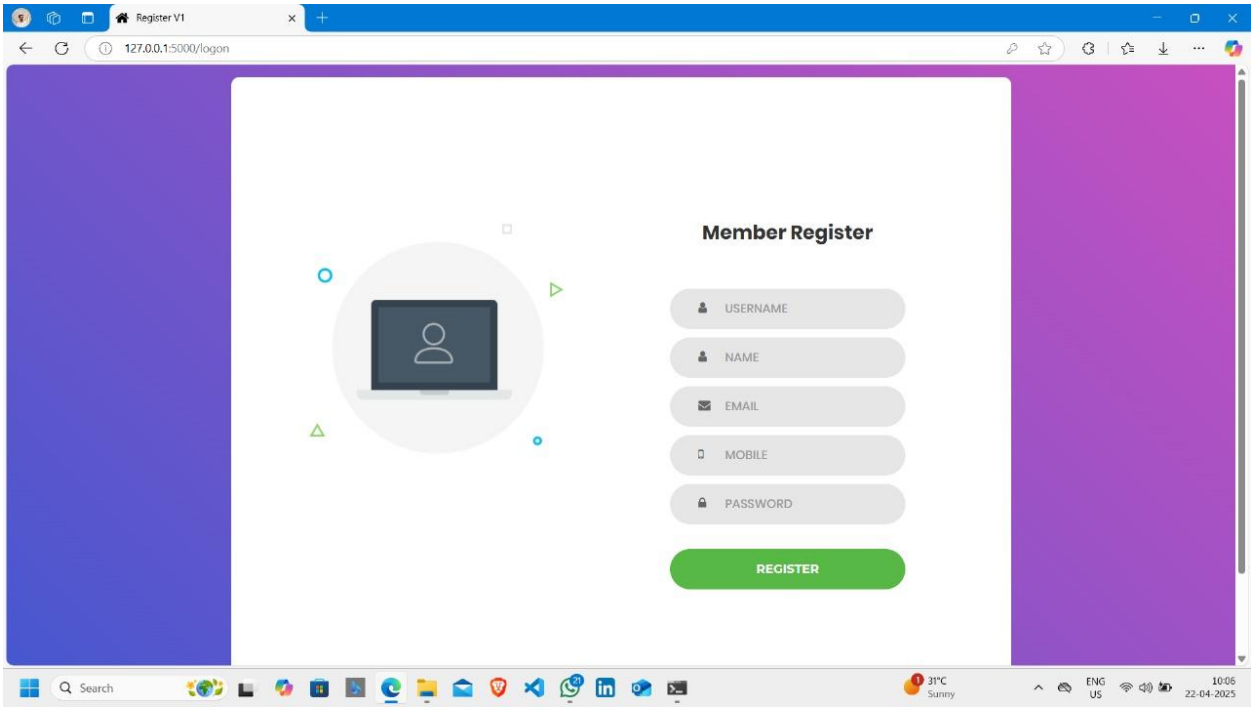





Home
127.0.0.1:5000
TechPhish
Home
Signup
Welcome to
Dashboard.
Evasion Attacks and Defense Mechanisms for Machine Learning-Based Web Phishing Classifiers.
Search
31°C
Sunny
ENG
US
10:05
22-04-2025


TechPhish
Prediction
Graph
Notebook
Signout
Result
URL :
https://www.amazon.com/gp/ubp/oneButton/config/redirectHome?tagbase=aceredge-ubpl&ref=aa-gateway-edgebookmark
Website is 78% safe to use...
Still want to Continue
Continue
Techway 2025 © - Designed by Zr Themes.
Follow us
Search
31°C
Sunny
ENG
US
10:07
22-04-2025


Result
127.0.0.1:5000/url
TechPhish
Prediction
Graph
Notebook
Signout
Result
URL :
http://secureupdate.appleId.com.duilawyeryork.com/ap/88b142e778d73f7
Website is 77% unsafe to use...
Still want to Continue
Continue
Techway 2025 © - Designed by Zr Themes.
Follow us


Register V1
127.0.0.1:5000/logon
Member Register
USERNAME
NAME
EMAIL
MOBILE
PASSWORD
REGISTER

Graph

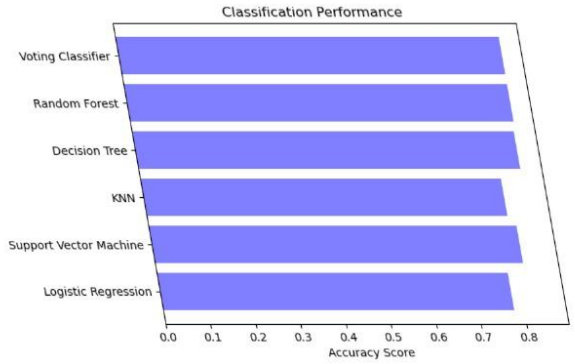
127.0.0.1:5000/about

TechPhish

Prediction Graph Notebook Signout

Have_ Have_ URL_Leng URL_Dep Redirecti Https_Doma TinyU Prefix/Suffi DNS_Reco Domain_A Domain_El iFran Mouse_Ov Right_Cli Web_Forwar

Accuracy:



Precision:

Search 31°C Sunny ENG US 10:08 22-04-2025

AlexaData

127.0.0.1:5000/notebook1

3	6356131	https://docs.google.com/forms/d/e/1FAIpQLScL6l...	http://www.phishtank.com/phish_detail.php?phis...	2020-01-13T20:13:37+00:00	yes	2020-01-17T01:55:38+00:00	yes	Other
4	6535965	https://oportunidadedasemana.com/americanas//7...	http://www.phishtank.com/phish_detail.php?phis...	2020-04-29T00:01:03+00:00	yes	2020-05-01T10:55:35+00:00	yes	Other

In [59]:
```
phishurl.to_csv('phish.csv')
```

In [55]:
```
phishurl.shape
```

Out[55]:
```
(5000, 8)
```

In [56]:
```
#Loading legitimate files
data1 = pd.read_csv("data/Benign_list_big_final.csv")
data1.columns = ['URLs']
data1.head()
```

Out[56]:

	URLs
0	http://1337x.to/torrent/1110018/Blackhat-2015-...
1	http://1337x.to/torrent/1122940/Blackhat-2015-...
2	http://1337x.to/torrent/1124395/Fast-and-Furio...
3	http://1337x.to/torrent/1145504/Avengers-Age-o...
4	http://1337x.to/torrent/1160078/Avengers-age-o...

In [57]:
```
#Collecting 5,000 Legitimate URLs randomly
legiurl = data1.sample(n = 5000, random_state = 12).copy()
legiurl = legiurl.reset_index(drop=True)
legiurl.head()
```

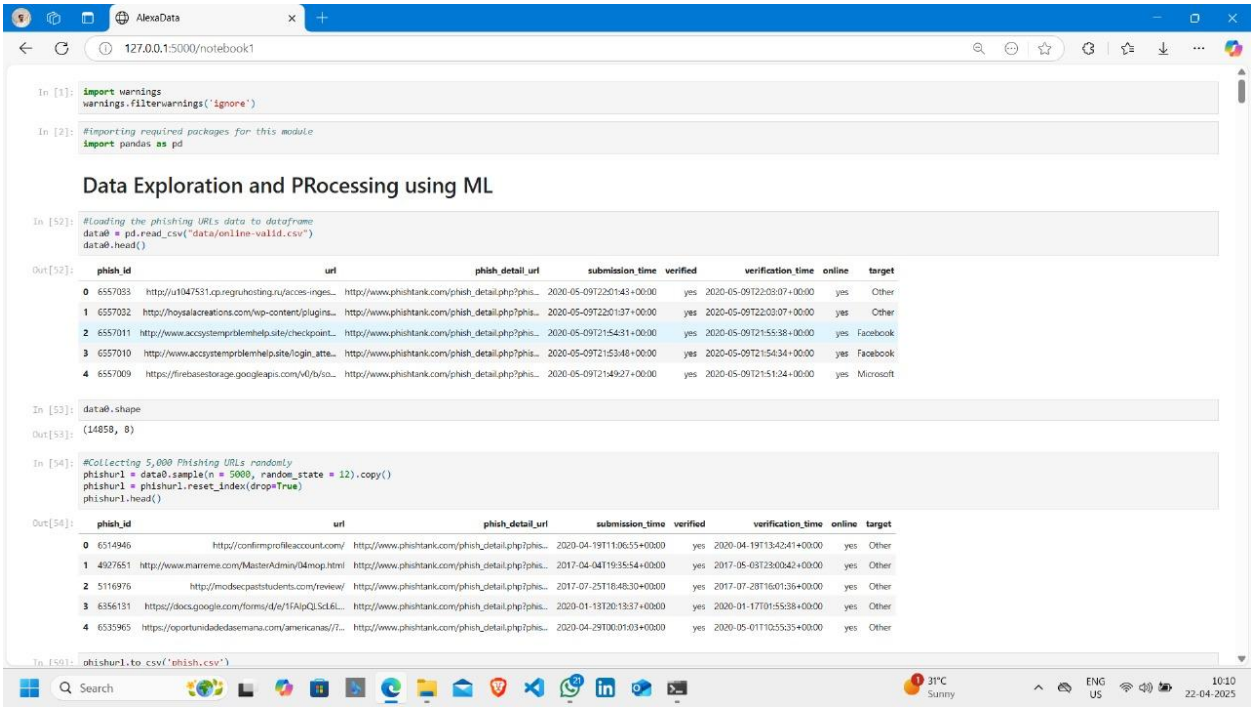
Out[57]:

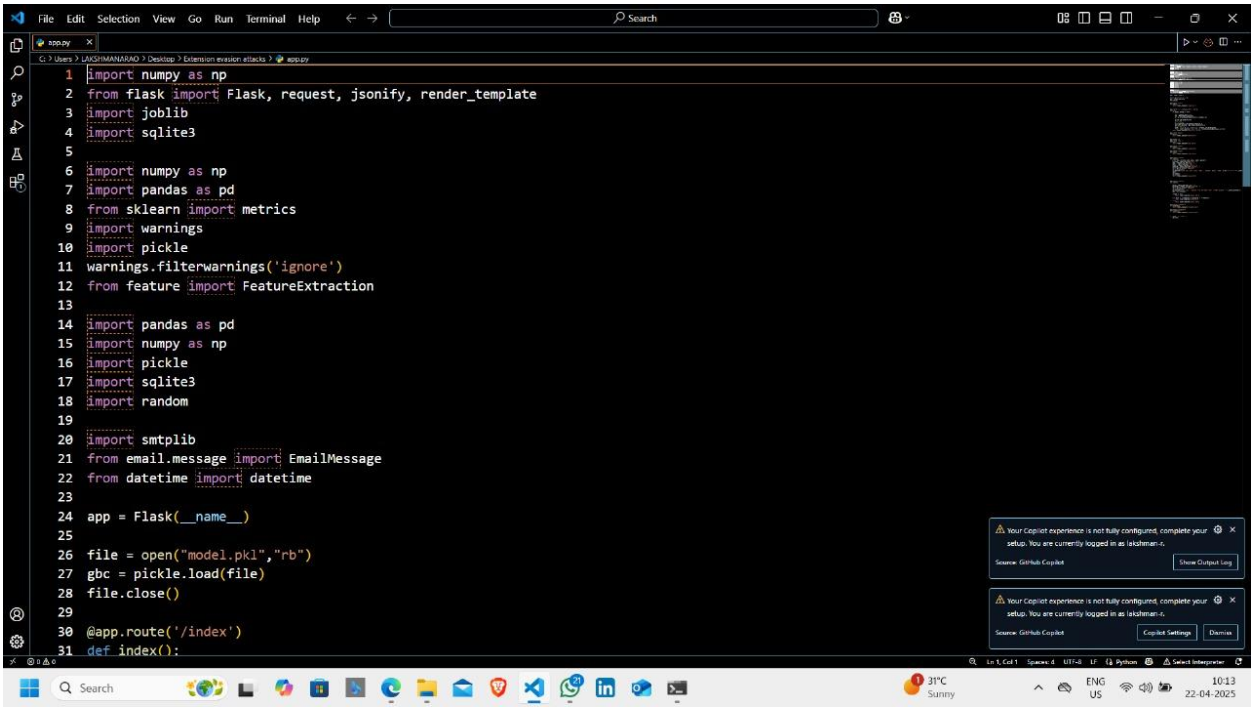
	URLs
0	http://graphicriver.net/search?date=this-month...
1	http://ecnavi.jp/redirect/?url=http://www.cros...
2	https://hubpages.com/signin?explain=follow+Hub...
3	http://extratorrent.cc/torrent/4190536/AOMEI+B...
4	http://icicibank.com/Personal-Banking/offers/o...

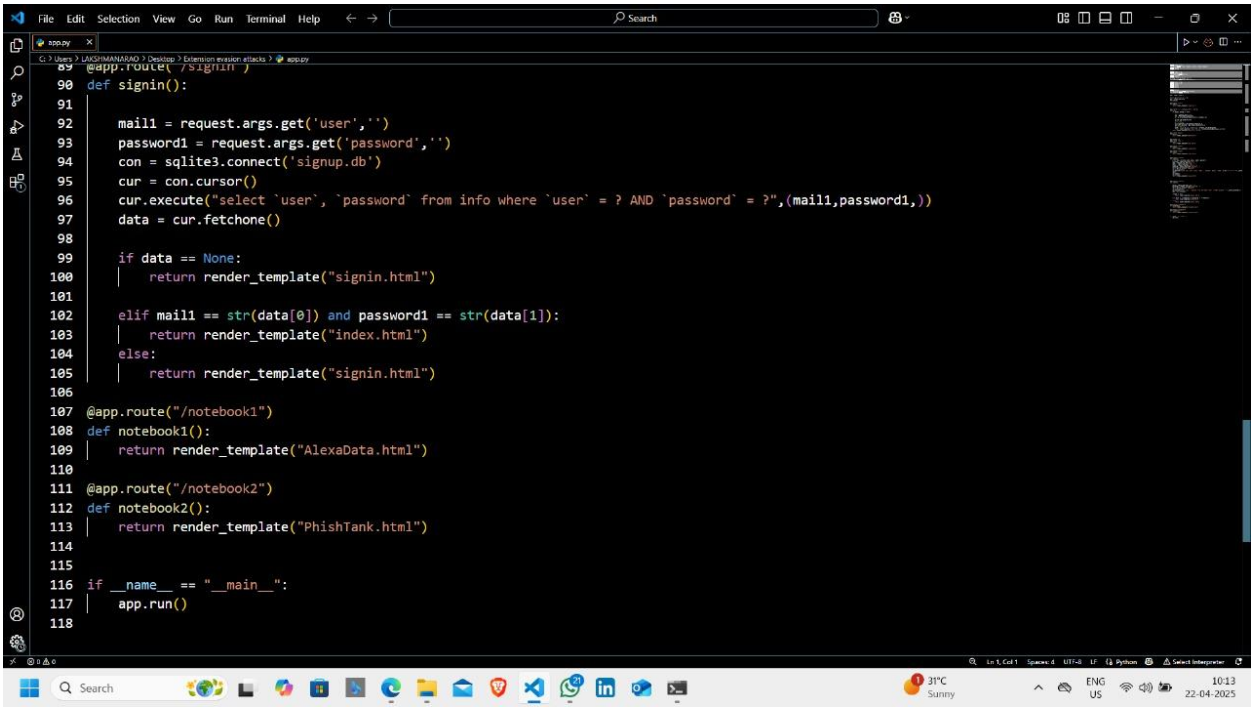
In [58]:
```
legiurl.to_csv('leg.csv')
```

In [0]:
```
legiurl.shape
```

Search 31°C Sunny ENG US 10:10 22-04-2025


AlexaData
127.0.0.1:5000/notebook1
In [1]:
import warnings
warnings.filterwarnings('ignore')
In [2]:
#importing required packages for this module
import pandas as pd
Data Exploration and PRocessing using ML
In [52]:
#Loading the phishing URLs data to dataframe
data0 = pd.read_csv("data/online-valid.csv")
data0.head()
Out[52]:
phish_id url phish_detail_url submission_time verified verification_time online target
0 6557033 http://u1047531.cp.regruhosting.ru/acces-inges... http://www.phishtank.com/phish_detail.php?phis... 2020-05-09T22:01:43+00:00 yes 2020-05-09T22:03:07+00:00 yes Other
1 6557032 http://hoysalacreations.com/wp-content/plugins... http://www.phishtank.com/phish_detail.php?phis... 2020-05-09T22:01:37+00:00 yes 2020-05-09T22:03:07+00:00 yes Other
2 6557011 http://www.accsystemprblemhelp.site/checkpoint... http://www.phishtank.com/phish_detail.php?phis... 2020-05-09T21:54:31+00:00 yes 2020-05-09T21:55:38+00:00 yes Facebook
3 6557010 http://www.accsystemprblemhelp.site/login_atte... http://www.phishtank.com/phish_detail.php?phis... 2020-05-09T21:53:48+00:00 yes 2020-05-09T21:54:34+00:00 yes Facebook
4 6557009 https://firebasestorage.googleapis.com/v0/b/so... http://www.phishtank.com/phish_detail.php?phis... 2020-05-09T21:49:27+00:00 yes 2020-05-09T21:51:24+00:00 yes Microsoft
In [53]:
data0.shape
Out[53]:
(14858, 8)
In [54]:
#Collecting 5,000 Phishing URLs randomly
phishurl = data0.sample(n = 5000, random_state = 12).copy()
phishurl = phishurl.reset_index(drop=True)
phishurl.head()
Out[54]:
phish_id url phish_detail_url submission_time verified verification_time online target
0 6514946 http://confirmprofileaccount.com/ http://www.phishtank.com/phish_detail.php?phis... 2020-04-19T11:06:55+00:00 yes 2020-04-19T13:42:41+00:00 yes Other
1 4927651 http://www.marreme.com/MasterAdmin/04mop.html http://www.phishtank.com/phish_detail.php?phis... 2017-04-04T19:39:54+00:00 yes 2017-05-03T23:00:42+00:00 yes Other
2 5116976 http://modsecpaststudents.com/review/ http://www.phishtank.com/phish_detail.php?phis... 2017-07-25T18:48:30+00:00 yes 2017-07-28T16:01:36+00:00 yes Other
3 6356131 https://docs.google.com/forms/d/e/1FAIpQLScL6L... http://www.phishtank.com/phish_detail.php?phis... 2020-01-13T20:13:37+00:00 yes 2020-01-17T01:55:38+00:00 yes Other
4 6535965 https://oportunidadedasemana.com/americanas/?... http://www.phishtank.com/phish_detail.php?phis... 2020-04-29T00:01:03+00:00 yes 2020-05-01T10:55:35+00:00 yes Other
In [59]:
phishurl.to_csv('phish.csv')
Search
31°C
Sunny
ENG
US
10:10
22-04-2025


File Edit Selection View Go Run Terminal Help
Search
app.py
C: > Users > LAKSHMANARAO > Desktop > Extension evasion attacks > app.py
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import joblib
4 import sqlite3
5
6 import numpy as np
7 import pandas as pd
8 from sklearn import metrics
9 import warnings
10 import pickle
11 warnings.filterwarnings('ignore')
12 from feature import FeatureExtraction
13
14 import pandas as pd
15 import numpy as np
16 import pickle
17 import sqlite3
18 import random
19
20 import smtplib
21 from email.message import EmailMessage
22 from datetime import datetime
23
24 app = Flask(__name__)
25
26 file = open("model.pkl","rb")
27 gbc = pickle.load(file)
28 file.close()
29
30 @app.route('/index')
31 def index():
Your Copilot experience is not fully configured, complete your setup. You are currently logged in as lakshman-s.
Source: GitHub Copilot
Show Output Log
Copilot Settings
Dismiss
Search
31°C
Sunny
ENG
US
10:13
22-04-2025



```python
@app.route('/signin')
def signin():

    mail1 = request.args.get('user','')
    password1 = request.args.get('password','')
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("select `user`, `password` from info where `user` = ? AND `password` = ?",(mail1,password1,))
    data = cur.fetchone()

    if data == None:
        return render_template("signin.html")

    elif mail1 == str(data[0]) and password1 == str(data[1]):
        return render_template("index.html")
    else:
        return render_template("signin.html")

@app.route("/notebook1")
def notebook1():
    return render_template("AlexaData.html")

@app.route("/notebook2")
def notebook2():
    return render_template("PhishTank.html")


if __name__ == "__main__":
    app.run()

```

CONCLUSION

10. CONCLUSION

This work mainly discusses about the evasion attacks and its detection in the case of websites. The analysis of URL dataset is carried out and the features that are important is extracted from the dataset.

Using these features the classification models such as K-Nearest Neighbour, Logistic regression, support vector machine, decision tree and random forest classifier was built.

The sample is created by adding the legitimate nodes to the phishing site and it successfully evaded the classifier. The evaluation of the attack is done by using mean squared error and finally the detection is done using the Decetor.

BIBILOGRAPHY

11. REFERENCES

[1] Song F, Lei Y, Chen S, Fan L, Liu Y, Advanced evasion attacks and miti gations on practical ML-based phishing website classifiers. International Journal of Intelligent Systems. 2021 Sep;36(9):5210-40. [2] Sabir B,BabarMA,GaireR,Anevasionattackagainst ml-based phishing url detectors,2020 May 18. [3] Shirazi H, Bezawada B, RayI,Anderson C,Adversarial sampling attacks against phishing detection. InIFIP Annual Conference on Data and Ap plications Security and Privacy 2019Jul15(pp.83-101).Springer,Cham. [4] Anupam S, Kar AK,Phishing website detection using support vector ma chines and nature-inspired optimization algorithms. Telecommunication Systems. 2021 Jan;76(1):17-32. [5] JainAK,GuptaBB,Towardsdetectionofphishingwebsitesonclient-side using machine learning based approach. Telecommunication Systems. 2018 Aug;68(4):687-700. [6] Chen S, Xue M, Fan L, Hao S, Xu L, Zhu H, Li B. Automated poisoning attacks and defenses in malware detection systems: An adversarial ma chine learning approach. computers & security. 2018 Mar 1;73:326-44. [7] Vayansky I, Kumar S, Phishing–challenges and solutions. Computer Fraud & Security. 2018 Jan 1;2018(1):15-20. [8] AbaidZ,KaafarMA,JhaS,Quantifyingtheimpactofadversarialevasion attacks on machine learning based android malware classifiers. In2017 IEEE 16th international symposium on network computing and applica tions (NCA) 2017 Oct 1 (pp. 1-10). IEEE. [9] Corona I, Biggio B, Contini M, Piras L, Corda R, Mereu M, Mureddu G, Ariu D, Roli F, Deltaphish: Detecting phishing webpages in compromised websites. InEuropeanSymposiumonResearchinComputerSecurity2017 Sep 11 (pp. 370-388). Springer, Cham [10] Xu W, Qi Y, Evans D, Automatically evading classifiers. InProceedings of the 2016 network and distributed systems symposium 2016 Feb (Vol. 10) [11] LiangB,SuM,YouW,ShiW,YangG,Crackingclassifiersforevasion:A case study on the google's phishing pages filter. InProceedings of the 25th International Conference on World Wide Web 2016 Apr 11 (pp. 345-356) [12] Harinahalli Lokesh G, BoreGowda G, Phishing website detection based on effective

machine learning approach. Journal of Cyber Security Tech nology. 2021 Jan 2;5(1):1-4. [13] Odeh A, Keshta I, Abdelfattah E,Machine learningtechniquesfor detec tion of website phishing: a review for promises and challenges. In2021 IEEE 11th Annual Computing and Communication Workshop and Con ference (CCWC) 2021 Jan 27 (pp. 0813-0818). [14] Zhuang W, Jiang Q, Xiong T, An intelligent anti-phishing strategy model for phishing website detection. In2012 32nd International Conference on Distributed Computing Systems Workshops 2012 Jun 18 (pp. 51-56). IEEE