**Phishing website prediction using base and ensemble classifier techniques with cross-validation**

**ABSTRACT**

**ABSTRACT**

This research addresses the escalating threat of phishing in the realm of public internetworks, aiming to enhance the security of internet users. Despite ongoing efforts to fortify IT infrastructure, the paper emphasizes the persistent risk of phishing attacks and focuses on the detection and prediction of phishing website URLs. Employing machine learning, the study employs primary classifiers such as Logistic Regression, Decision Tree, SVM, KNN, Naive Bayes, LDA, and ensemble-based techniques including CNN, RNN, and a Voting Classifier. The investigation unfolds in three phases: initial classification with base classifiers, subsequent deployment of ensemble classifiers, and evaluation with and without cross-validation on distinct datasets. Notably, CNN and RNN achieve remarkable 100% accuracy. The findings underscore the significance of ensemble techniques in phishing website prediction, providing valuable insights for future research and contributing to the ongoing efforts to secure online environments.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**INTRODUCTION**

**1. INTRODUCTION**

In an era where our reliance on the internet is pervasive, encompassing everything from online shopping and banking to sophisticated intelligent home solutions, our daily lives and work culture have undergone a significant transformation. However, this increased connectivity has brought about a proportional rise in the number of threats that jeopardize our digital existence. The globally operated network platforms, while offering unprecedented convenience, are also fertile grounds for various forms of cyber threats.

Beyond the well-known menaces like hacking, cracking, and the activities of online terrorist organizations, one particularly insidious threat looms large – phishing. This online criminal activity often goes unnoticed or is insufficiently acknowledged by its victims. Phishing poses a significant risk to users, with attackers targeting two primary groups: the uninformed newbies who lack awareness of the technical intricacies of the internet, and the careless individuals who, despite understanding the associated risks, remain inattentive to online security.

This paper delves into the escalating concern of phishing attacks, exploring the challenges posed by this form of cybercrime. As users increasingly navigate the digital landscape, it becomes imperative to comprehensively address the multifaceted nature of phishing threats and devise robust strategies to safeguard against them.

**1.1 Objective:**

This study aims to enhance phishing website detection on public internetworks by employing machine learning classifiers, including CNN, RNN, Logistic Regression, Decision Tree, SVM, KNN, Naive Bayes, LDA, and a Voting Classifier. The research evaluates their performance, emphasizing the importance of ensemble techniques in bolstering internet security.

**1.2 Problem Statement:**

The pervasive threat of phishing in public internetworks poses a persistent risk to users' data security. Despite continuous efforts, the challenge lies in effectively detecting and predicting phishing websites. This study addresses this concern, employing diverse machine learning classifiers to enhance the identification and mitigation of phishing threats.

**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**

**1.4 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**

**Feasibility Study**

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

**Types Of Feasibility Study**

A feasibility analysis evaluates the project’s potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

**1. Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn’t want to try to put Star Trek’s transporters in their building—currently, this project is not technically feasible.

**2. Economic Feasibility**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent [project assessment](https://www.simplilearn.com/risk-assessment-project-management-article) and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

### **3. Legal Feasibility**

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws,[data protection](https://www.simplilearn.com/understanding-data-security-rar30-article) acts or social media laws. Let’s say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization’s ideal location isn’t zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

### **4. Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization’s needs can be met by completing the project. Operational feasibility studies also examine how a [project plan](https://www.simplilearn.com/project-management-plans-in-project-environment-rar79-article) satisfies the requirements identified in the requirements analysis phase of system development.

### **5. Scheduling Feasibility**

This assessment is the most important for [project success](https://www.simplilearn.com/how-to-make-a-project-successful-article); after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

* Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
* Internal Corporate Constraints: Financial, Marketing, Export, etc.
* External Constraints: Logistics, Environment, Laws, and Regulations, etc.

**LITERATURE SURVEY**

**3.LITERATURE SURVEY**

**3.1 Phishing URL Detection with Lexical Features and Blacklisted Domains:**

[Phishing URL Detection with Lexical Features and Blacklisted Domains | SpringerLink](https://link.springer.com/chapter/10.1007/978-3-030-33432-1_12)

**ABSTRACT:** Many cyberattacks start with phishing to lure victims into malicious web pages where malware codes are hidden. Victim machines are infected by malware and the attacker can intrude the enterprise network, evading firewalls. Therefore, it is of fundamental importance to detect phishing URLs and prevent employees from visiting them. Many machine learning methods were proposed so far. In this work, we collect many lexical features after literature survey and combine them with blacklisted domains to improve the detection performance. We collect many recent phishing URLs because most of open datasets are outdated. Our method shows the F-1 of 0.84.

**3.2 A predictive model for phishing detection:**

[A predictive model for phishing detection - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1319157819304902)

**ABSTRACT:** Nowadays, many anti-phishing systems are being developed to identify phishing contents in online communication systems. Despite the availability of myriads anti-phishing systems, phishing continues unabated due to inadequate detection of a zero-day attack, superfluous computational overhead and high false rates. Although Machine Learning approaches have achieved promising accuracy rate, the choice and the performance of the feature vector limit their effective detection. In this work, an enhanced machine learning-based predictive model is proposed to improve the efficiency of anti-phishing schemes. The predictive model consists of Feature Selection Module which is used for the construction of an effective feature vector. These features are extracted from the URL, webpage properties and webpage behaviour using the incremental component-based system to present the resultant feature vector to the predictive model. The proposed system uses Support Vector Machine and Naïve Bayes which have been trained on a 15-dimensional feature set. The experiments were based on datasets consisting of 2541 phishing instances and 2500 benign instances. Using 10-fold cross-validation, the experimental results indicate a remarkable performance with 0.04% False Positive and 99.96% accuracy for both SVM and NB predictive models.

**3.3 PhiDMA – A phishing detection model with multi-filter approach:**

[PhiDMA – A phishing detection model with multi-filter approach - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1319157817301210)

**ABSTRACT:** Phishing remains a basic security issue in the cyberspace. In phishing, assailants steal sensitive information from victims by providing a fake site which looks like the visual clone of a legitimate site. Phishing shall be handled using various approaches. It is established that single filter methods would be insufficient to detect different categories of phishing attempts. This paper provides a multilayer model to detect phishing, titled as PhiDMA(Phishing Detection using Multi-filter Approach). The PhiDMA model incorporates five layers: Auto upgrade whitelist layer, URL features layer, Lexical signature layer, String matching layer and Accessibility Score comparison layer. A prototype implementation of the proposed PhiDMA model is built with an accessible interface so that persons with visual impairments shall access it without any barrier. The result from the experiment shows that the model is capable to detect phishing sites with an accuracy of 92.72%.

**3.4 CBR-PDS: a case-based reasoning phishing detection system:**

[CBR-PDS: a case-based reasoning phishing detection system | Request PDF (researchgate.net)](https://www.researchgate.net/publication/323446879_CBR-PDS_a_case-based_reasoning_phishing_detection_system)

**ABSTRACT:** Phishing attacks have become the preferred vehicle to gather sensitive information as well as to deliver dangerous malware. So far, there is still no phishing detection system that can perfectly detect and progressively self adapt to differentiate between phishing and legitimate websites. This paper proposes the case-based reasoning Phishing detection system (CBR-PDS) that relies on previous cases to detect phishing attacks. CBR-PDS is highly adaptive and dynamic as it can adapt to detect new phishing attacks using rather a small dataset size in contrast to other machine learning techniques. CBR-PDS aims to improve the detection accuracy and the reliability of the results by identifying a set of discriminative features and discarding irrelevant features. CBR-PDS relies on a two stage hybrid procedure using Information gain and Genetic algorithms. The reduction of the data dimensionality amounts to an improved accuracy rate, yet it necessitates a reduced processing time. The CBR-PDS is tested using different scenarios on a various balanced datasets. The obtained performances clearly show the suitability of our proposed hybrid feature selection procedure as well as the efficiency of the proposed CBR-PDS system. The obtained accuracy rates exceed 95%. We also show that the integration of an Online Phishing Threats component into the CBR-PDS system improves further the accuracy rate. Finally, CRB-PDS performances are compared to those of several known competitive classifiers.

**3.5 Machine learning based phishing detection from URLs:**

[Machine learning based phishing detection from URLs - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0957417418306067)

**ABSTRACT:** Due to the rapid growth of the Internet, users change their preference from traditional shopping to the electronic commerce. Instead of bank/shop robbery, nowadays, criminals try to find their victims in the cyberspace with some specific tricks. By using the anonymous structure of the Internet, attackers set out new techniques, such as phishing, to deceive victims with the use of false websites to collect their sensitive information such as account IDs, usernames, passwords, etc. Understanding whether a web page is legitimate or phishing is a very challenging problem, due to its semantics-based attack structure, which mainly exploits the computer users’ vulnerabilities. Although software companies launch new anti-phishing products, which use blacklists, heuristics, visual and machine learning-based approaches, these products cannot prevent all of the phishing attacks. In this paper, a real-time anti-phishing system, which uses seven different classification algorithms and natural language processing (NLP) based features, is proposed. The system has the following distinguishing properties from other studies in the literature: language independence, use of a huge size of phishing and legitimate data, real-time execution, detection of new websites, independence from third-party services and use of feature-rich classifiers. For measuring the performance of the system, a new dataset is constructed, and the experimental results are tested on it. According to the experimental and comparative results from the implemented classification algorithms, Random Forest algorithm with only NLP based features gives the best performance with the 97.98% accuracy rate for detection of phishing URLs.

**SYSTEM ANALYSIS**

**4. SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

The existing system employs a cutting-edge approach to combat the pervasive threat of phishing attacks by leveraging machine learning, specifically utilizing the Gradient Boosting Classifier model. The system focuses on scrutinizing various aspects of URL significance to distinguish between legitimate and phishing websites. By extracting and comparing distinctive features, such as linguistic and aesthetic elements, the Gradient Boosting Classifier is adept at identifying phishing URLs. This innovative method offers real-time detection capabilities, providing a robust defense against evolving phishing techniques facilitated by advancing technology. Recognizing the ease with which attackers exploit human susceptibility, the system serves as a proactive safeguard against unauthorized access to personal information. Through rigorous testing, the findings affirm the efficacy of the proposed approach, demonstrating its ability to successfully discern between authentic and malicious websites, thereby fortifying cybersecurity measures in the face of escalating phishing threats.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

* The existing system relies solely on the Gradient Boosting Classifier, potentially overlooking nuanced phishing patterns that other algorithms may better detect.
* The system's emphasis on linguistic and aesthetic features may make it susceptible to evolving phishing techniques that exploit novel characteristics not considered in the current feature set.
* Focusing primarily on URL features might result in reduced accuracy when dealing with phishing attempts that employ sophisticated tactics beyond URL manipulation.
* The system may face challenges in adapting to the rapidly changing phishing landscape, as it might struggle to incorporate new features or adapt its model to emerging attack vectors.

# 4.2 Proposed System:

The proposed system introduces a robust cybersecurity solution employing an ensemble of machine learning algorithms, including CNN, RNN, Logistic Regression, Decision Tree, SVM, KNN, Naive Bayes, LDA, and a Voting Classifier. Notably, the system prioritizes the superior accuracy achieved by Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in comparison to other algorithms. This emphasis on CNN and RNN, acknowledged for their proficiency in detecting phishing websites, aims to elevate the system's overall performance. By leveraging the strengths of these advanced algorithms, the proposed system seeks to enhance the accuracy and efficiency of phishing detection and prediction. This strategic amalgamation of diverse classifiers underscores the system's commitment to providing a comprehensive defense mechanism against the dynamic landscape of online threats, ensuring heightened security for users engaged in various internet activities.

# 4.2.1 Advantages of proposed system:

1. The proposed system incorporates a wide array of machine learning algorithms, ensuring a more comprehensive and adaptive approach to phishing detection.
2. Prioritizing Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) acknowledges their proven efficacy, potentially enhancing the system's accuracy in detecting complex phishing patterns.
3. Leveraging multiple advanced algorithms aims to improve overall system performance, providing a more robust defense against a diverse range of phishing tactics.
4. The strategic amalgamation of diverse classifiers demonstrates the proposed system's commitment to offering a holistic cybersecurity solution, capable of addressing the dynamic nature of online threats.

### **4.3 FUNCTIONAL REQUIREMENTS**

1. Data Collection

2. Data Pre-processing

3. Training and Testing

4. Modelling

5. Predicting

### **4.4 NON FUNCTIONAL REQUIREMENTS**

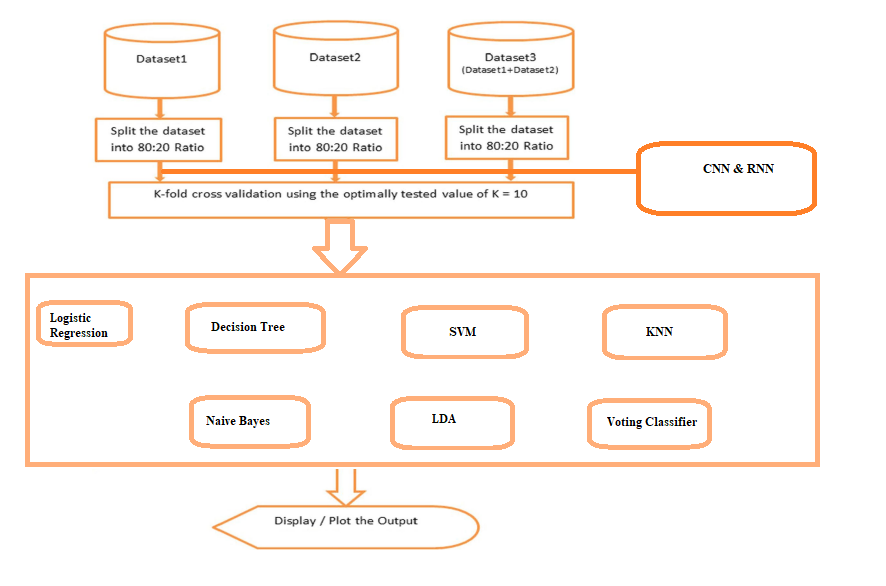
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“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 allows 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 are > 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.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Exploring the dataset**

**Data processing**

**Visualization**

**Splitting the data into train & test**

**Building the model - CNN - RNN - Logistic Regression - Decision Tree - SVM - KNN - Naive Bayes - LDA - Voting Classifier**

**Training the model**

**Signup & Signin**

**User input**

**Final outcome**

**End process**

**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.

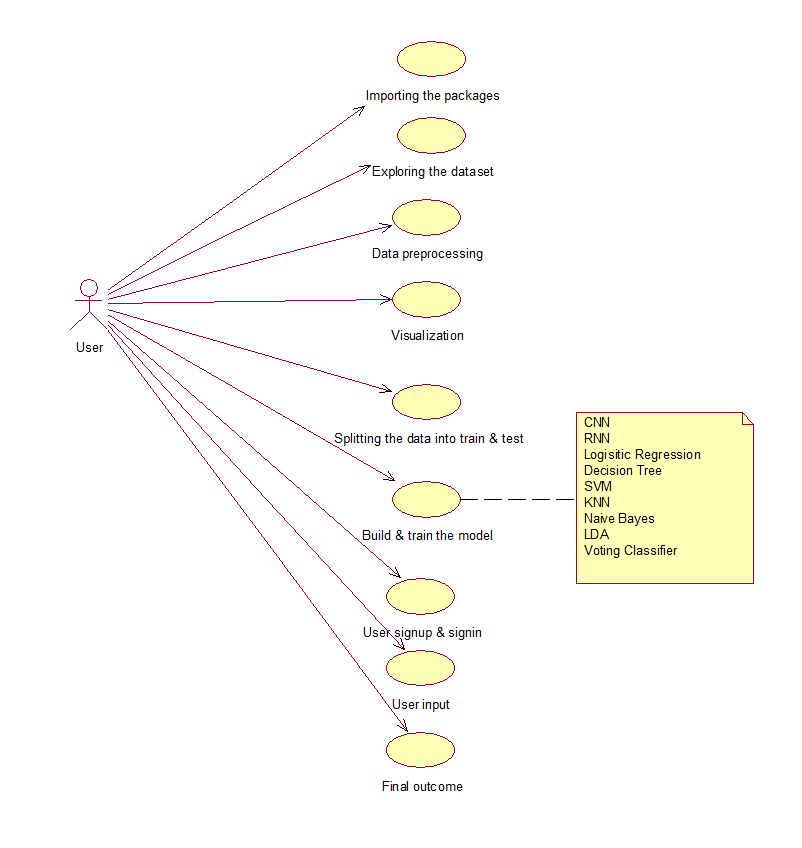
**GOALS:**

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.

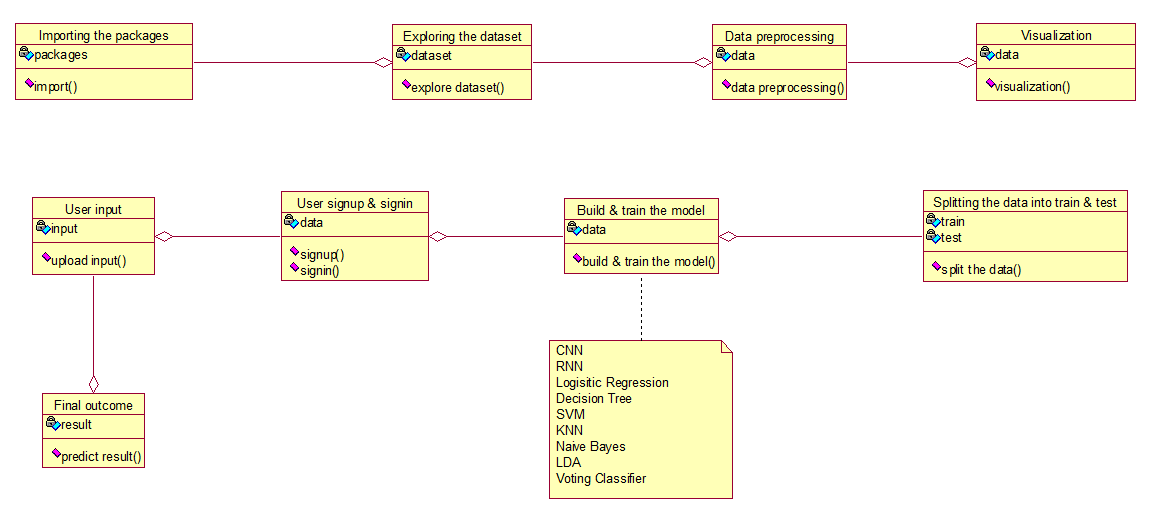
**Use case diagram:**

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.



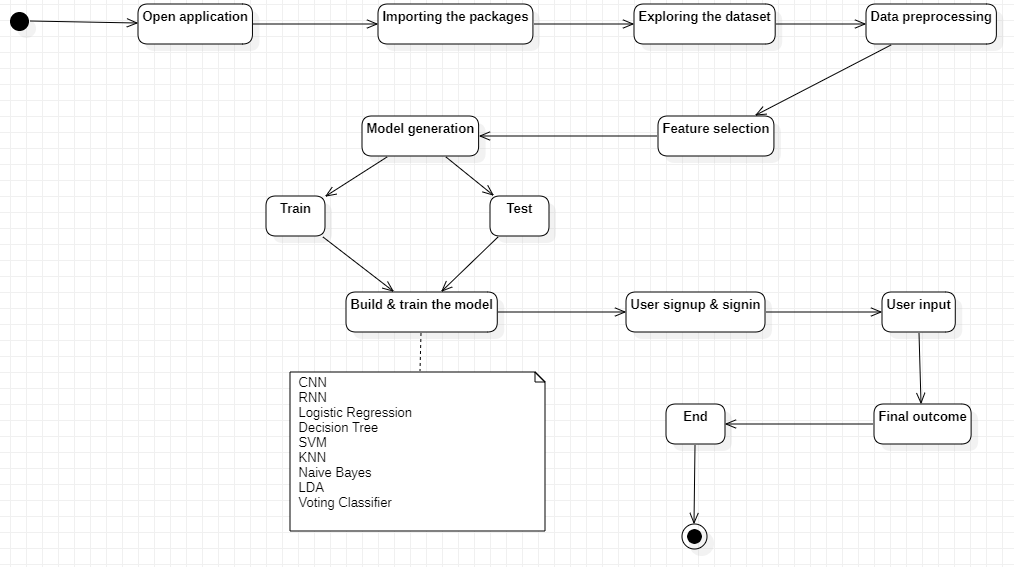
**Class diagram:**

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.



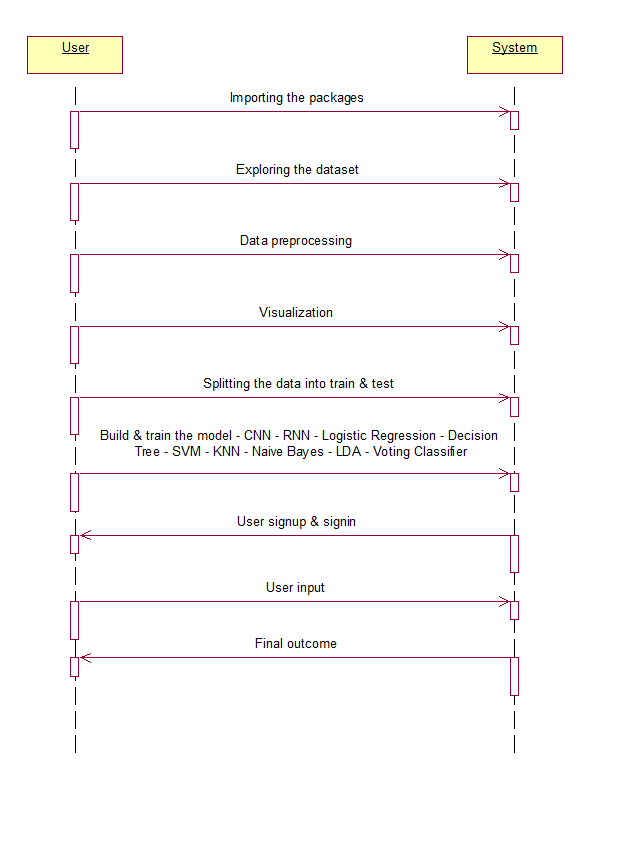
**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

****

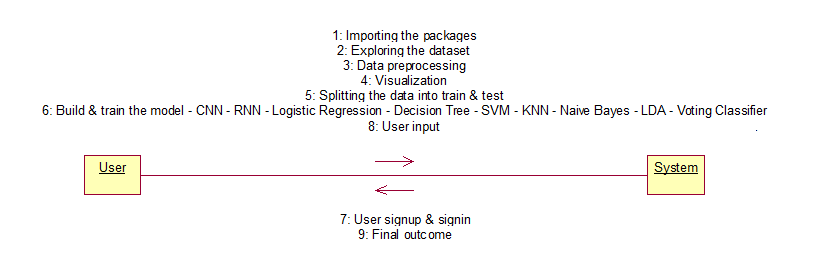
**Sequence diagram:**

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".



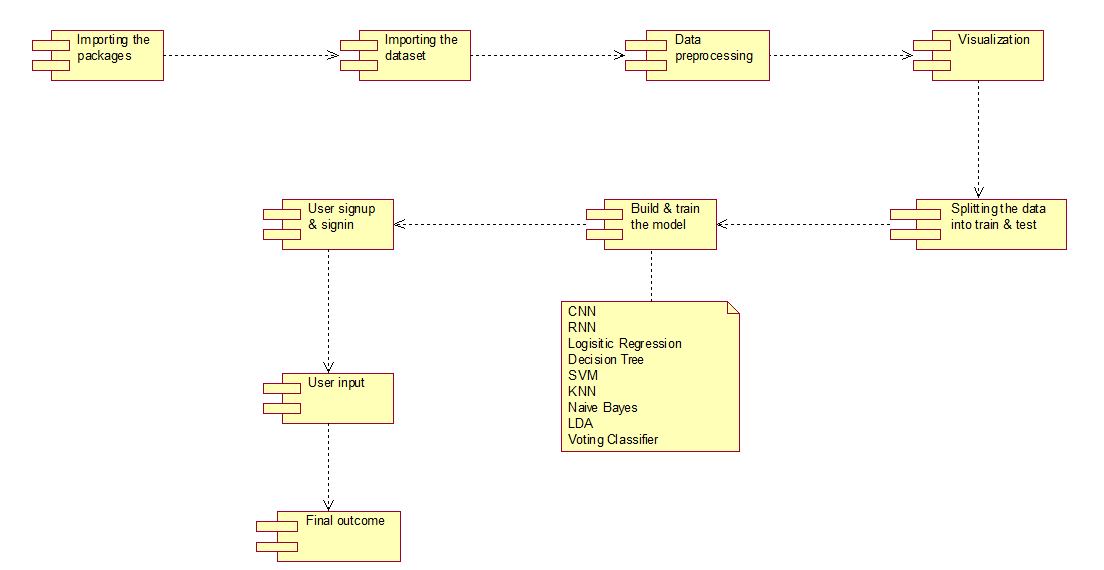
**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

****

**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**

1. **IMPLEMENTATION**

**MODULES:**

* **Data loading:** using this module we are going to import the dataset.
* **Data Preprocessing:** using this module we will explore the data.
* **Splitting data into train & test:** using this module data will be divided into train & test
* **Model generation:** Model building - CNN - RNN - Logistic Regression - Decision Tree - SVM - KNN - Naive Bayes - LDA - Voting Classifier. Algorithms accuracy calculated
* **User signup & login:** Using this module will get registration and login
* **User input:** Using this module will give input for prediction
* **Prediction:** final predicted displayed

**Algorithms:**

CNN: A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks.

RNN: Recurrent neural networks (RNNs) are the state of the art algorithm for sequential data and are used by Apple's Siri and Google's voice search. It is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data.

Logistic Regression: Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

Decision Tree: A decision tree algorithm is a machine learning algorithm that uses a decision tree to make predictions. It follows a tree-like model of decisions and their possible consequences. The algorithm works by recursively splitting the data into subsets based on the most significant feature at each node of the tree.

SVM: A support vector machine (SVM) is a type of supervised learning algorithm used in machine learning to solve classification and regression tasks; SVMs are particularly good at solving binary classification problems, which require classifying the elements of a data set into two groups.

KNN: The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Naïve Bayes: The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification. It is also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category.

LDA: Linear Discriminant Analysis (LDA) is a supervised learning algorithm used for classification tasks in machine learning. It is a technique used to find a linear combination of features that best separates the classes in a dataset.

Voting Classifier: A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

**6.2 SAMPLE CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

pd.set\_option('display.max\_columns', None)

plt.rcParams['figure.figsize'] = (12,6)

data = pd.read\_csv("/kaggle/input/phishing-dataset-for-machine-learning/Phishing\_Legitimate\_full.csv")

float\_cols = data.select\_dtypes('float64').columns

for c **in** float\_cols:

data[c] = data[c].astype('float32')

int\_cols = data.select\_dtypes('int64').columns

for c **in** int\_cols:

data[c] = data[c].astype('int32')

data.info()

data.rename(columns={'CLASS\_LABEL': 'labels'}, inplace=True)

data.sample(5)

data.describe()

data['labels'].value\_counts().plot(kind='bar')

def corr\_heatmap(data, idx\_s, idx\_e):

y = data['labels']

temp = data.iloc[:, idx\_s:idx\_e]

if 'id' **in** temp.columns:

del temp['id']

temp['labels'] = y

sns.heatmap(temp.corr(), annot=True, fmt='.2f')

plt.show()

corr\_heatmap(data, 0, 10)

corr\_heatmap(data, 10, 20)

corr\_heatmap(data, 20, 30)

corr\_heatmap(data, 30, 40)

corr\_heatmap(data, 40, 50)

from sklearn.feature\_selection import mutual\_info\_classif

X = data.drop(['id', 'labels'], axis=1)

y = data['labels']

discrete\_features = X.dtypes == int

mi\_scores = mutual\_info\_classif(X, y, discrete\_features=discrete\_features)

mi\_scores = pd.Series(mi\_scores, name='MI Scores', index=X.columns)

mi\_scores = mi\_scores.sort\_values(ascending=False)

mi\_scores

def plot\_mi\_scores(scores):

scores = scores.sort\_values(ascending=True)

width = np.arange(len(scores))

ticks = list(scores.index)

plt.barh(width, scores)

plt.yticks(width, ticks)

plt.title("MI Scores")

plt.figure(dpi=100, figsize=(12,12))

plot\_mi\_scores(mi\_scores)

from sklearn.linear\_model import LogisticRegression

from cuml.ensemble import RandomForestClassifier as cuRfc

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

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

def train\_logistic(data, top\_n):

top\_n\_features = mi\_scores.sort\_values(ascending=False).head(top\_n).index.tolist()

X = data[top\_n\_features]

y = data['labels']

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

lr = LogisticRegression(max\_iter=10000)

lr.fit(X\_train, y\_train)

y\_pred = lr.predict(X\_test)

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

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

f1 = f1\_score(y\_test, y\_pred)

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

return precision, recall, f1, accuracy

arr = []

for i **in** range(20,51,1):

precision, recall, f1, accuracy = train\_logistic(data, i)

print("Performance for Logistic Model with Top **{}** features is precision : **{}**, recall : **{}**, f1 score : **{}**, accuracy : **{}**".format(i, precision, recall, f1, accuracy))

arr.append([i, precision, recall, f1, accuracy])

df = pd.DataFrame(arr, columns=['num\_of\_features', 'precision', 'recall', 'f1\_score', 'accuracy'])

df

sns.lineplot(x='num\_of\_features', y='precision', data=df, label='Precision Score')

sns.lineplot(x='num\_of\_features', y='recall', data=df, label='Recall Score')

sns.lineplot(x='num\_of\_features', y='f1\_score', data=df, label='F1 Score')

sns.lineplot(x='num\_of\_features', y='accuracy', data=df, label='Acc Score')

def train\_rfc(data, top\_n):

top\_n\_features = mi\_scores.sort\_values(ascending=False).head(top\_n).index.tolist()

X = data[top\_n\_features]

y = data['labels']

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

rfc = cuRfc(n\_estimators=500,

split\_criterion=1,

max\_depth=32,

max\_leaves=-1,

max\_features=1.0,

n\_bins=128)

rfc.fit(X\_train, y\_train)

y\_pred = rfc.predict(X\_test, predict\_model='CPU')

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

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

f1 = f1\_score(y\_test, y\_pred)

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

return precision, recall, f1, accuracy

arr = []

for i **in** range(20,51,1):

precision, recall, f1, accuracy = train\_rfc(data, i)

print("Performance for RFC Model with Top **{}** features is precision : **{}**, recall : **{}**, f1 score : **{}**, accuracy : **{}**".format(i, precision, recall, f1, accuracy))

arr.append([i, precision, recall, f1, accuracy])

df = pd.DataFrame(arr, columns=['num\_of\_features', 'precision', 'recall', 'f1\_score', 'accuracy'])

df.head()

sns.lineplot(x='num\_of\_features', y='precision', data=df, label='Precision Score')

sns.lineplot(x='num\_of\_features', y='recall', data=df, label='Recall Score')

sns.lineplot(x='num\_of\_features', y='f1\_score', data=df, label='F1 Score')

sns.lineplot(x='num\_of\_features', y='accuracy', data=df, label='Acc Score')

top\_n\_features = mi\_scores.sort\_values(ascending=False).head(32).index.tolist()

X = data[top\_n\_features]

y = data['labels']

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

rfc = cuRfc(n\_estimators=500,

split\_criterion=1,

max\_depth=32,

max\_leaves=-1,

max\_features=1.0,

n\_bins=128)

rfc.fit(X\_train, y\_train)

y\_pred = rfc.predict(X\_test, predict\_model='CPU')

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

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

f1 = f1\_score(y\_test, y\_pred)

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

print("Performance for RFC Model with Top **{}** features is precision : **{}**, recall : **{}**, f1 score : **{}**, accuracy : **{}**".format(27, precision, recall, f1, accuracy))

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

**SOFTWARE ENVIRONMENT**

**7. SOFTWARE ENVIRONMENT**

**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.

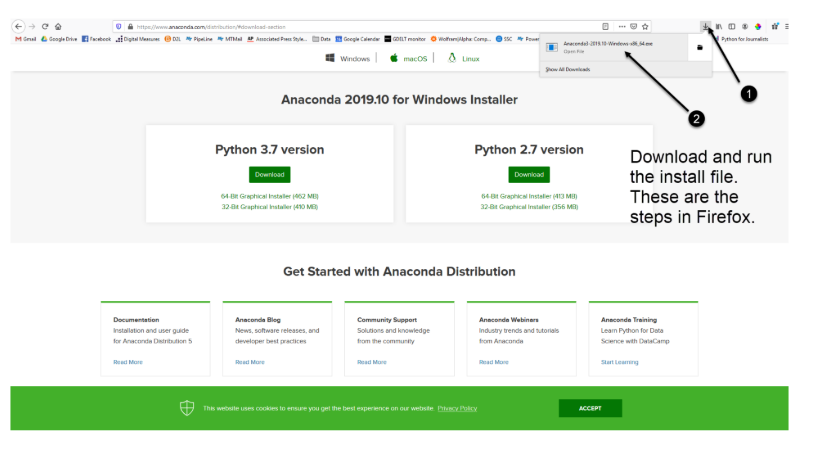
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to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

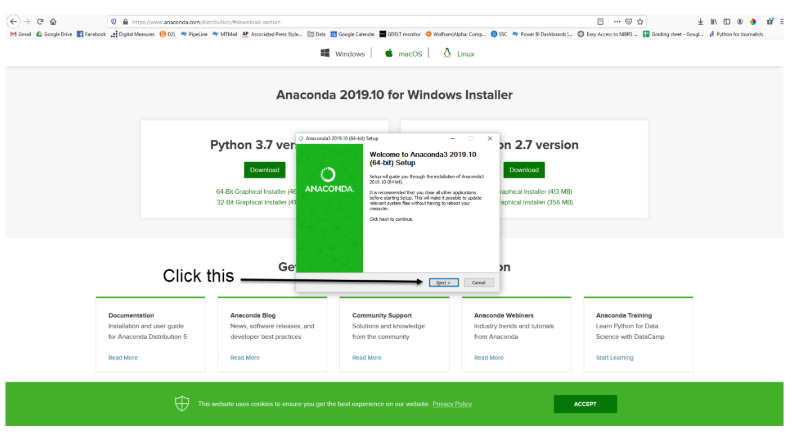
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](https://www.anaconda.com/distribution/)

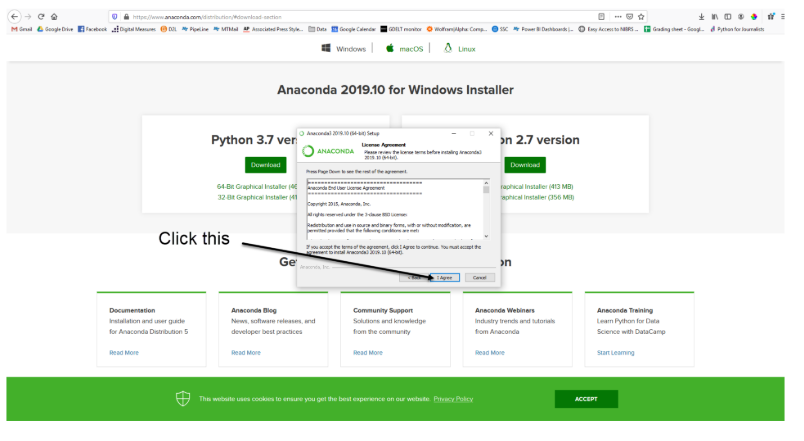


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

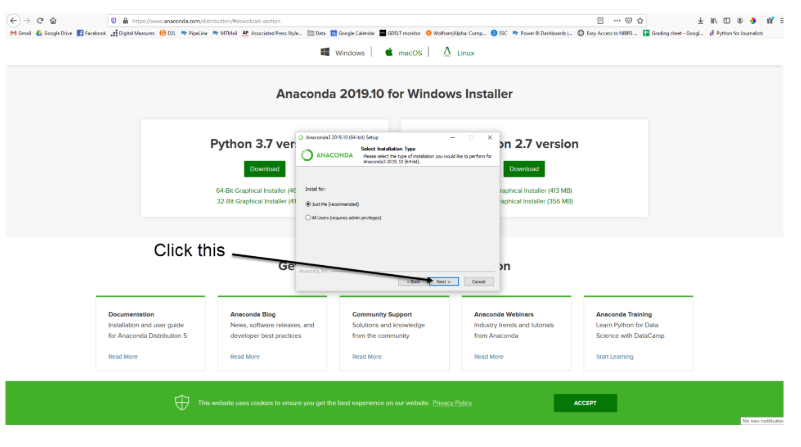
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



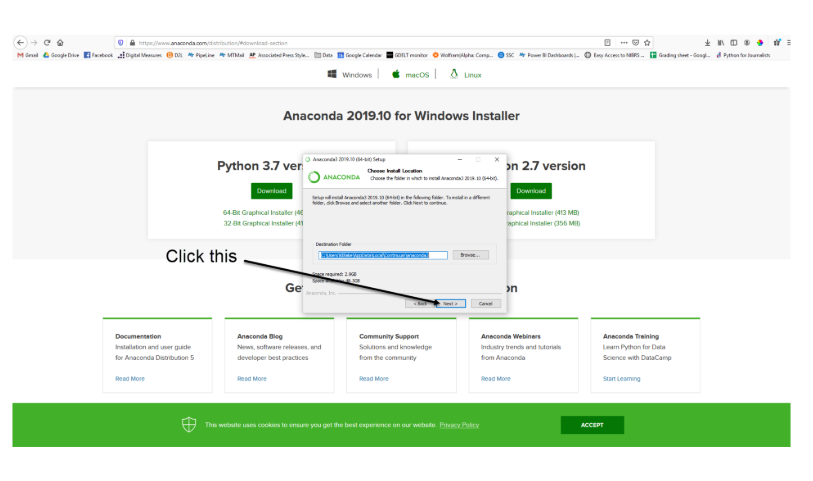
3. Now, you’ll see the license agreement. Click on ‘I Agree’.



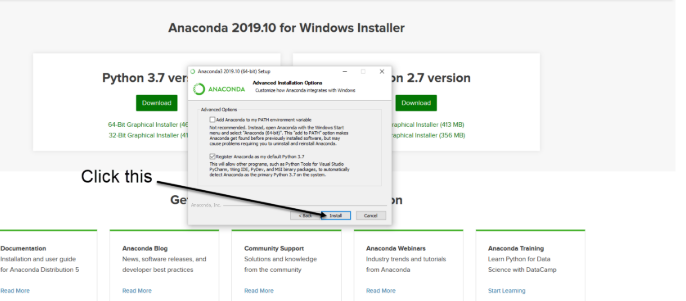
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



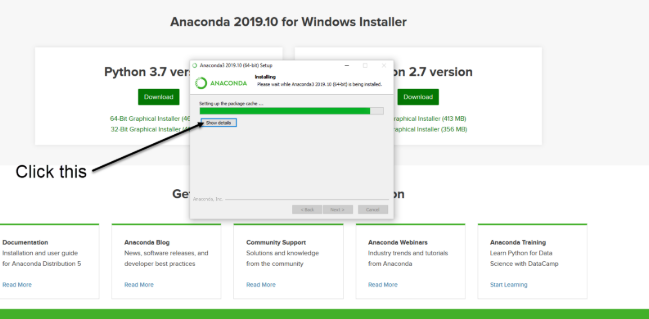
5. Choose where you want to install it. Here, you can see the available space and how much you need.



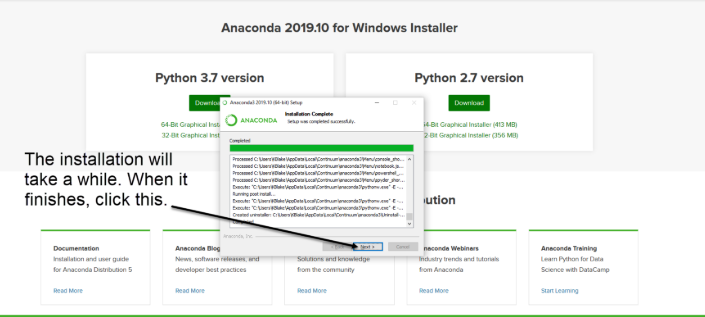
6. Now, you’ll get some advanced options. You can add Anaconda to your system’s PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on ‘Install’.



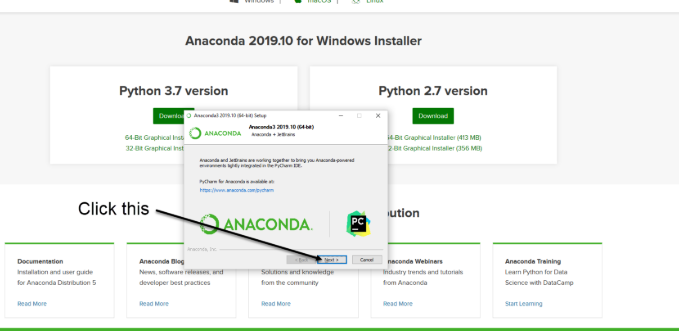
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.



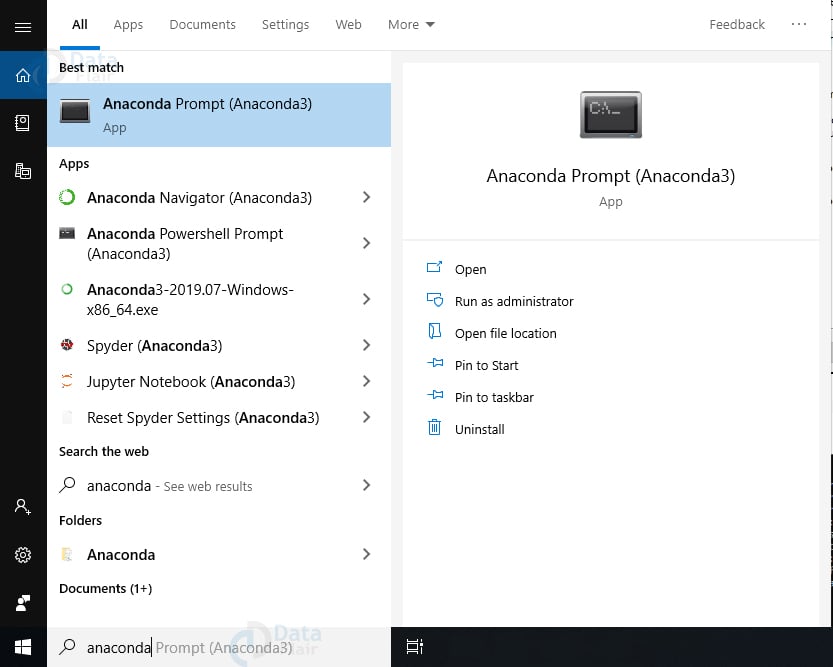
9. This screen will inform you about PyCharm. Click Next.



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 interpreted, 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](https://www.geeksforgeeks.org/python-programming-language/)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](https://www.python.org/downloads/) 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](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). 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](https://www.geeksforgeeks.org/python-oops-concepts/). 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](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). 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](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)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](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), 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++](http://www.geeksforgeeks.org/c-plus-plus/), 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](https://www.geeksforgeeks.org/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 **bytecode**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) 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 expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-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](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### **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.

**What is Machine Learning : -**

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

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

**Categories Of Machine Leaning :-**

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

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

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

## Need for Machine Learning

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

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

## Challenges in Machines Learning :-

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

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

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

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

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

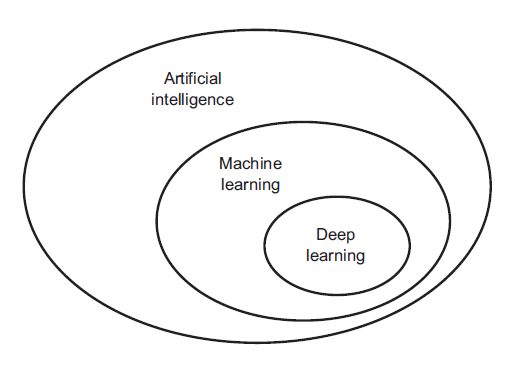
**Issue of overfitting & underfitting** − If the model is overfitting or underfitting, it cannot be represented well for the problem.

**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.

**What is deep learning?**

Deep learning is a subset of [machine learning](https://www.ibm.com/topics/machine-learning), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

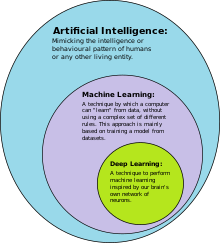


Deep learning drives many [artificial intelligence (AI)](https://www.ibm.com/topics/artificial-intelligence) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

**How deep learning works**

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

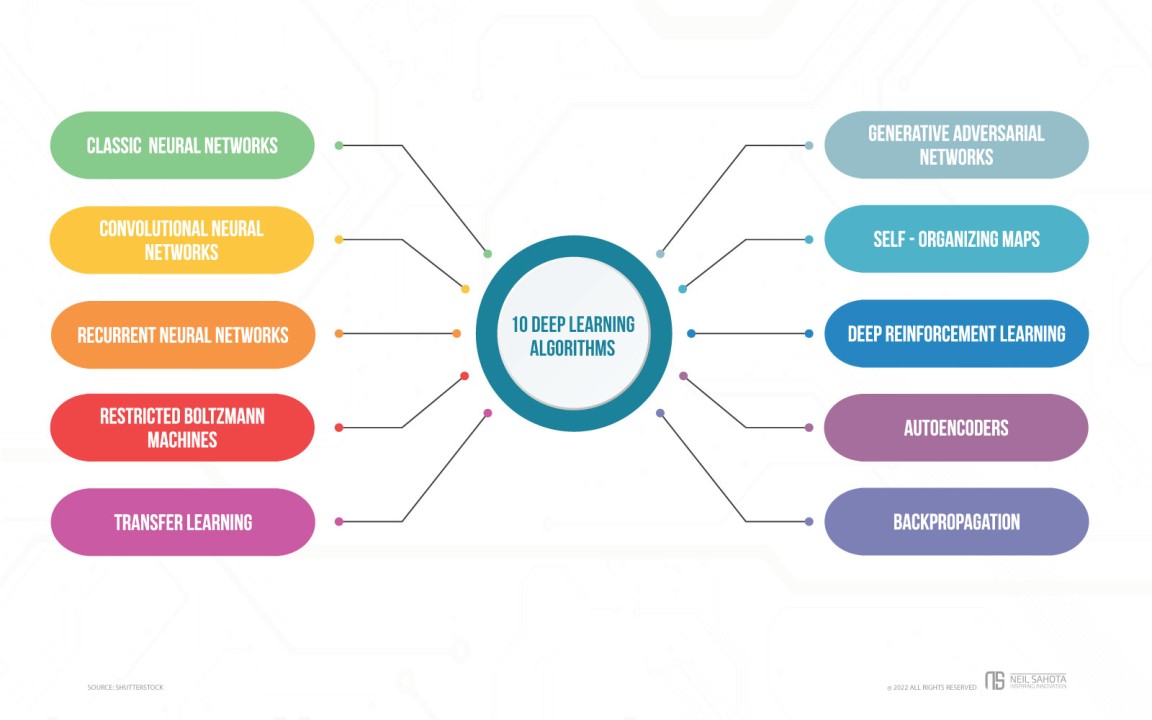
Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible*layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.



Another process called backpropagationuses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example,

* [*Convolutional neural networks (CNNs),*](https://www.ibm.com/topics/convolutional-neural-networks)used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.
* [*Recurrent neural network (RNNs)*](https://www.ibm.com/topics/recurrent-neural-networks)are typically used in natural language and speech recognition applications as it leverages sequential or times series data.



**LIBRARIES/PACKGES :-**

**Tensorflow**

TensorFlow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/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](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) 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](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

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

**Scikit – learn**

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

**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.

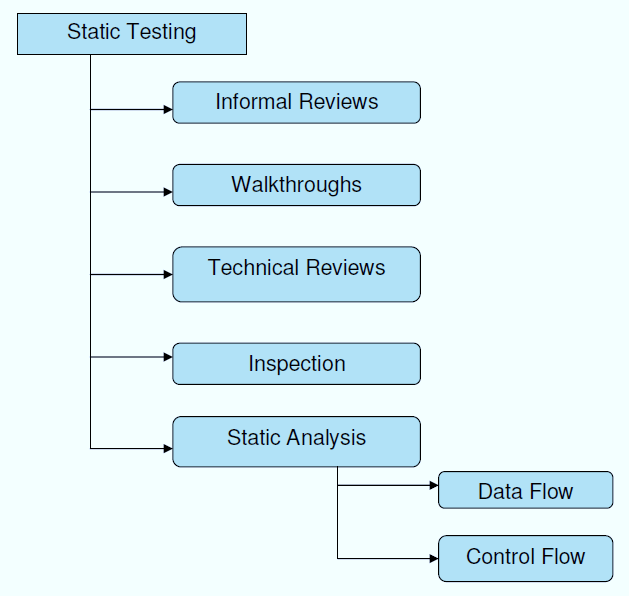
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**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:

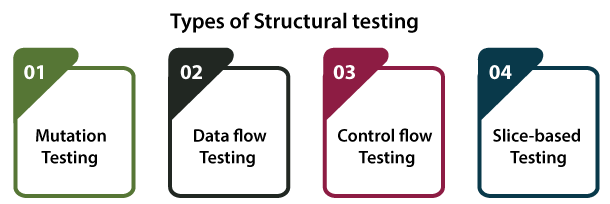
Static Testing:

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.



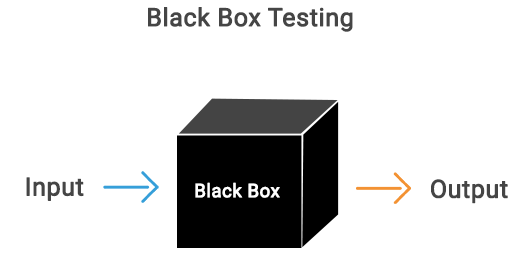
**Structural Testing:**

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.



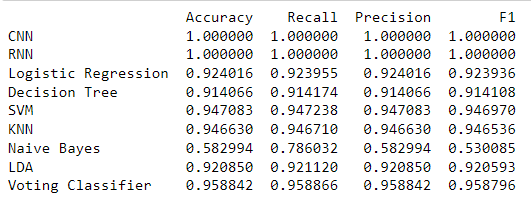
**8.2 TEST CASES:**

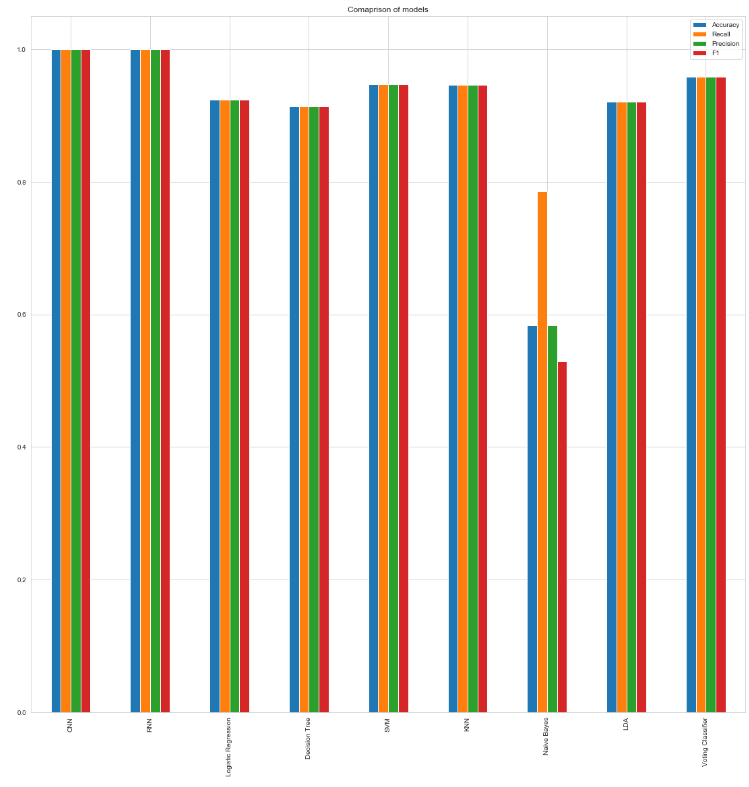
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User signin | User get login into the application | There is no process |
| 3 | Enter input for prediction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

**SCREENS**

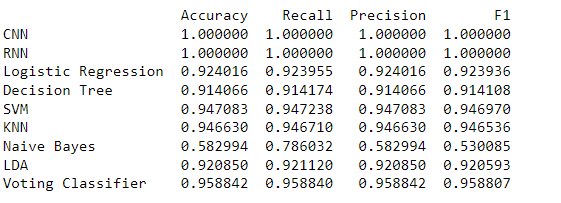
1. **SCREENSHOTS**

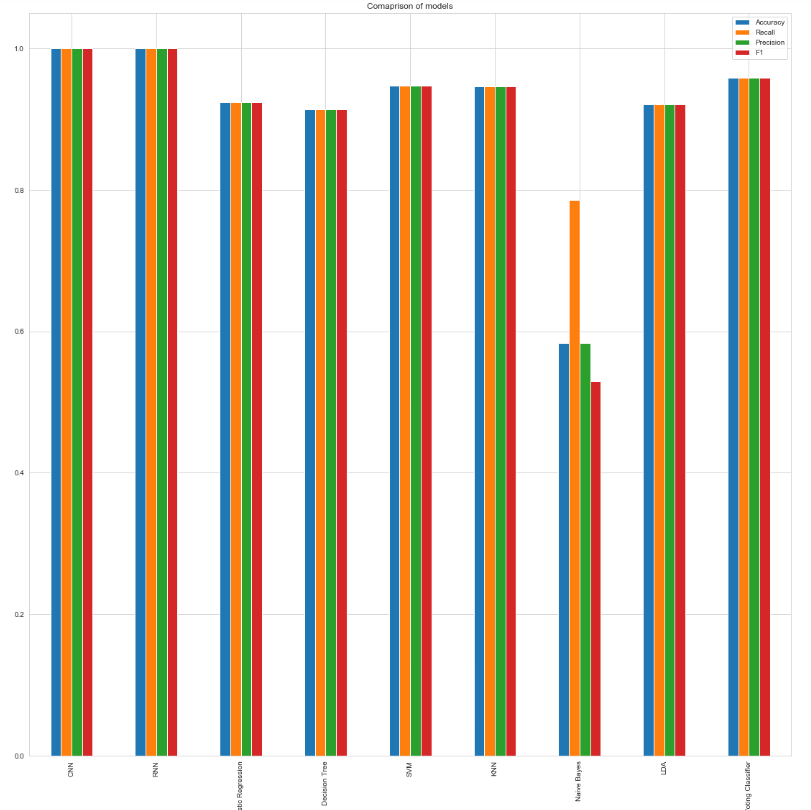
Dataset - 1



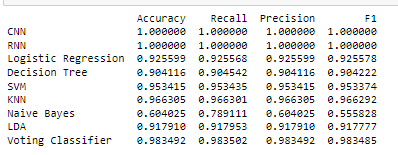


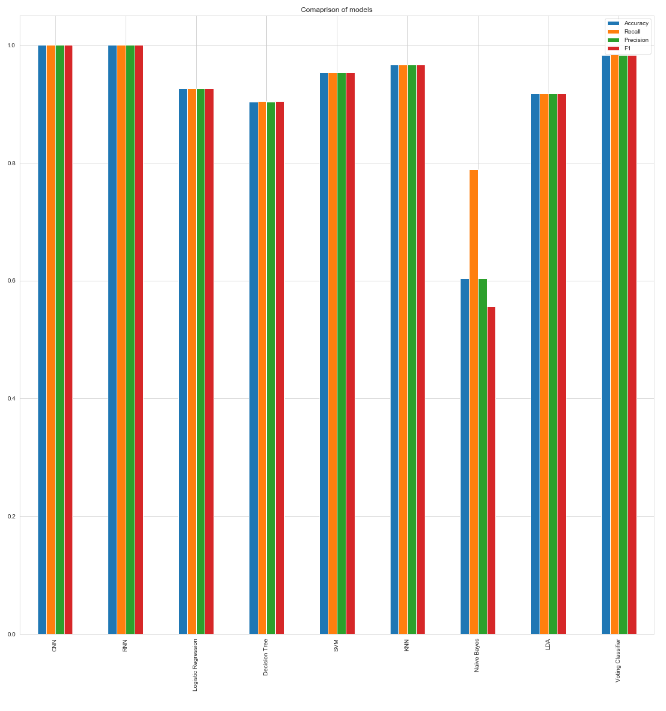
Dataset -2

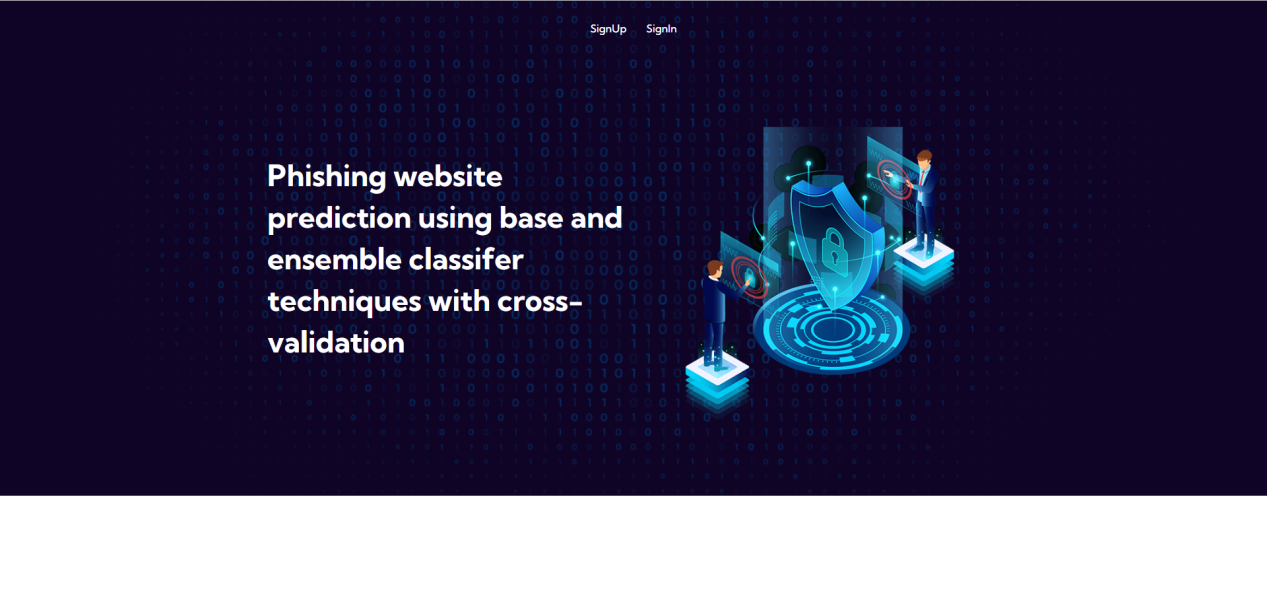


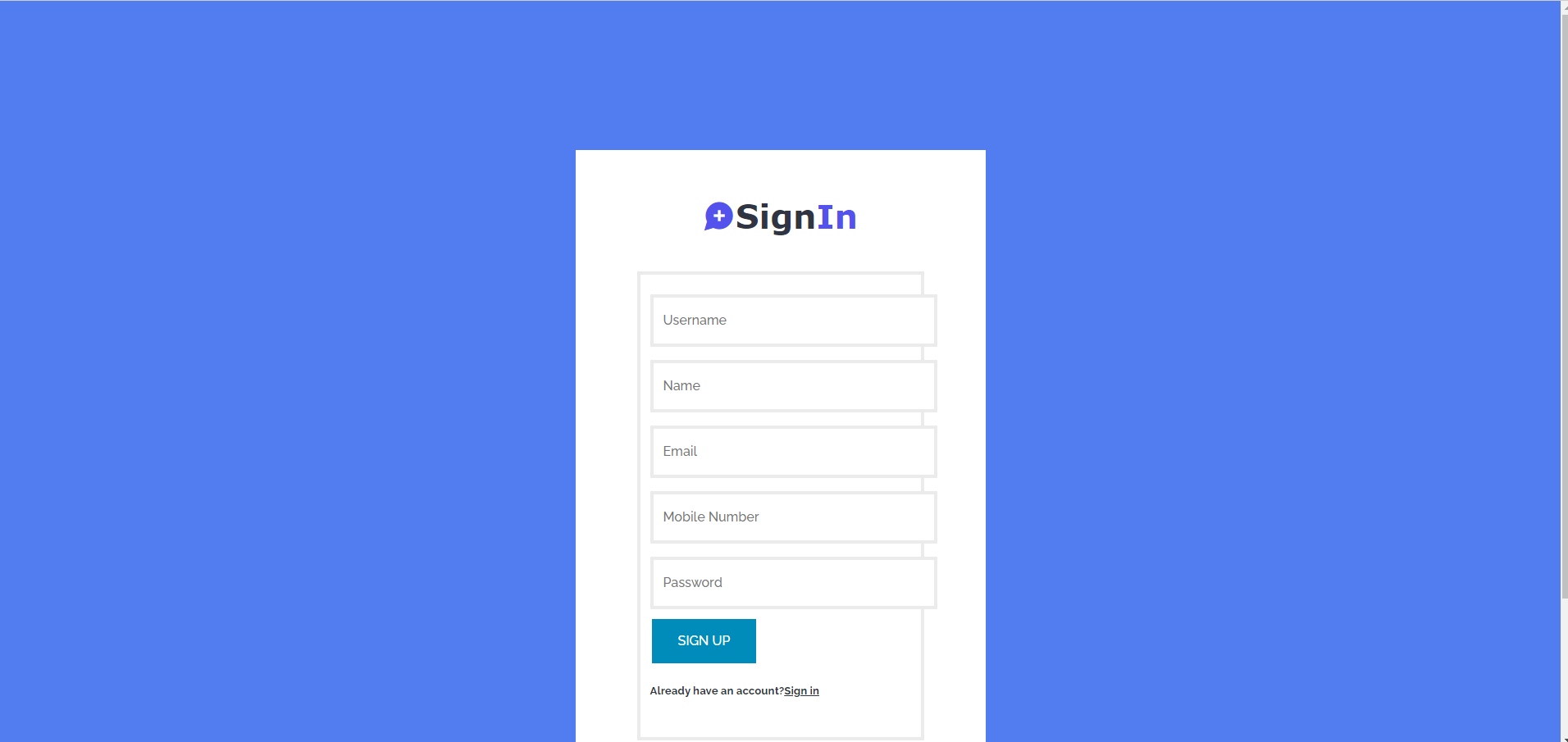


Dataset -3

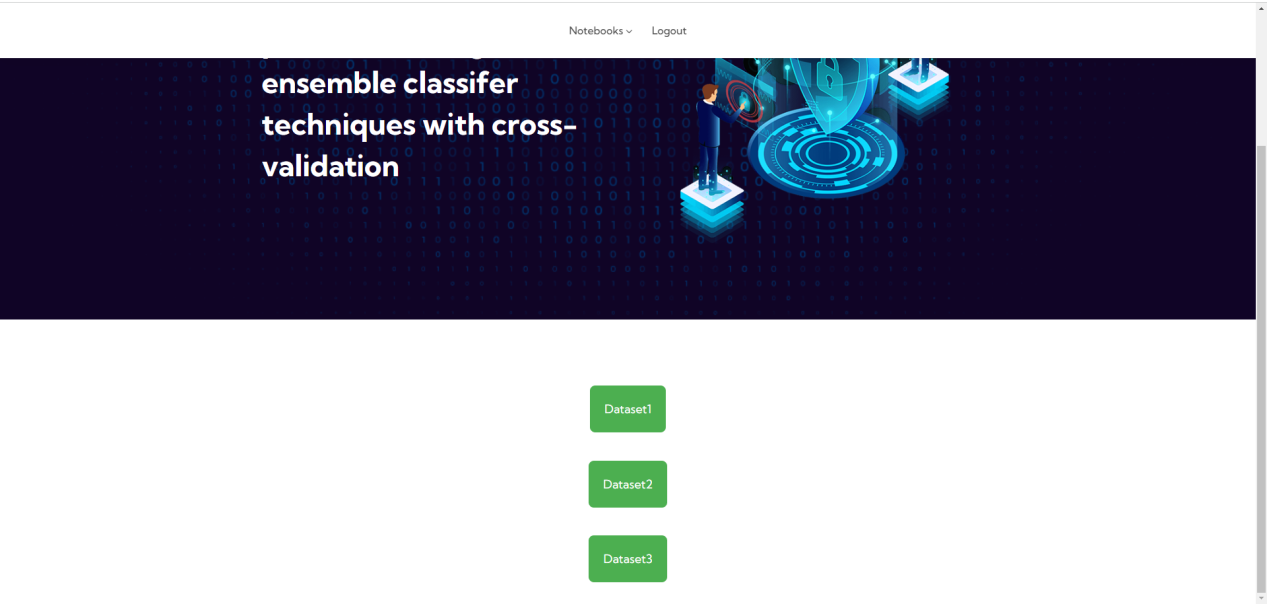


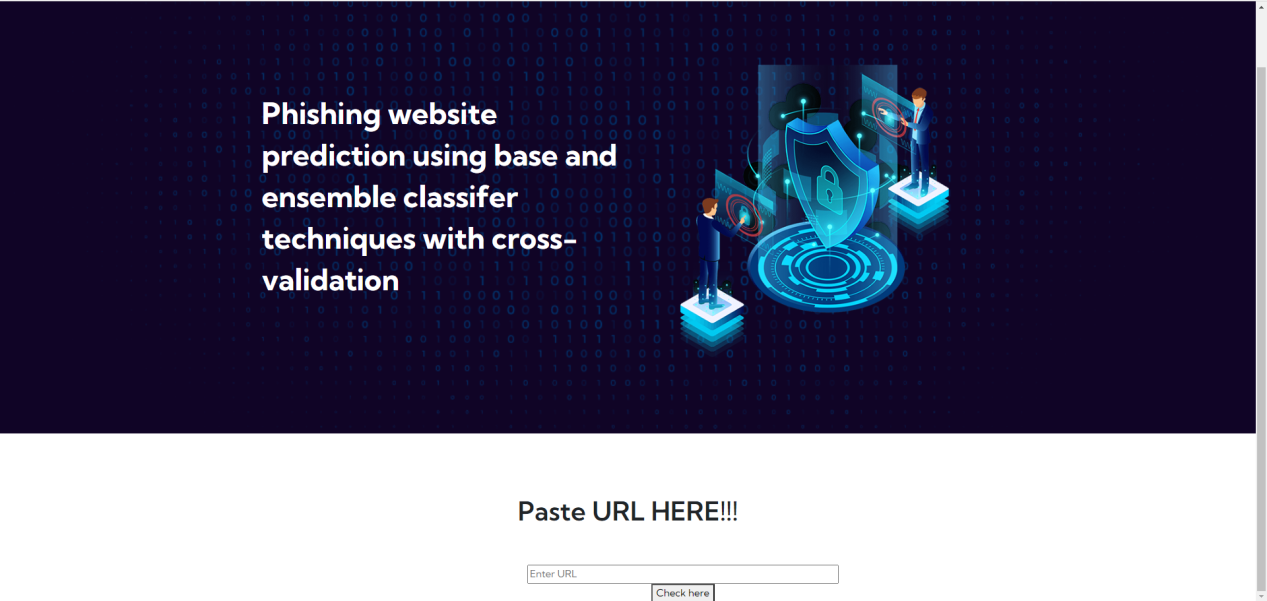




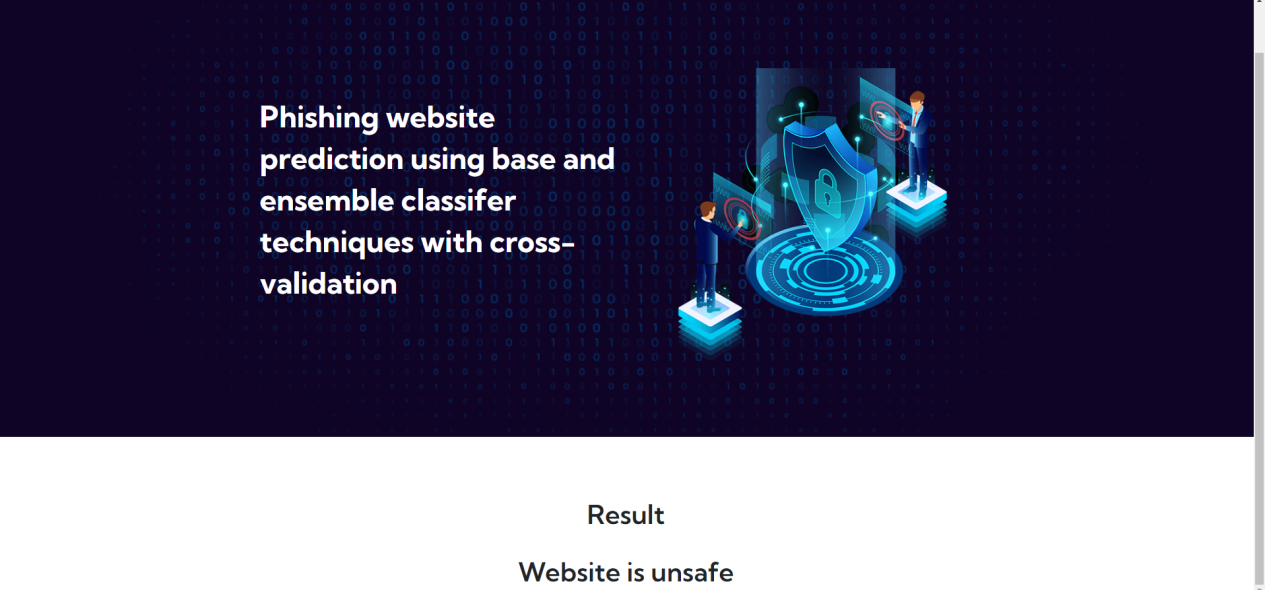


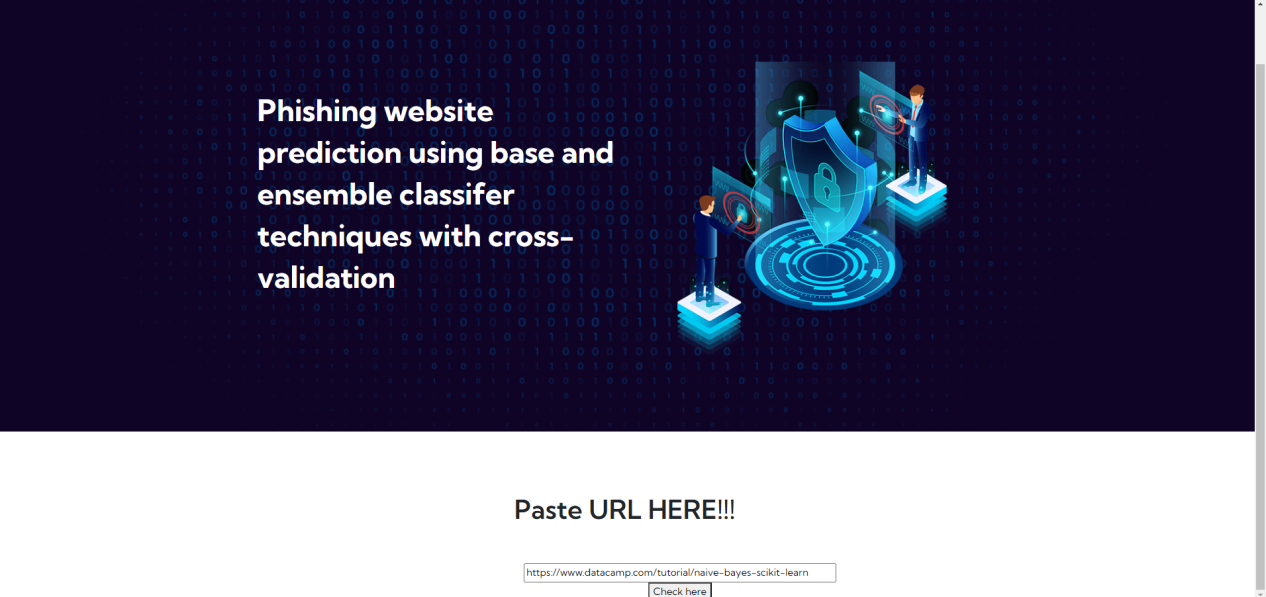


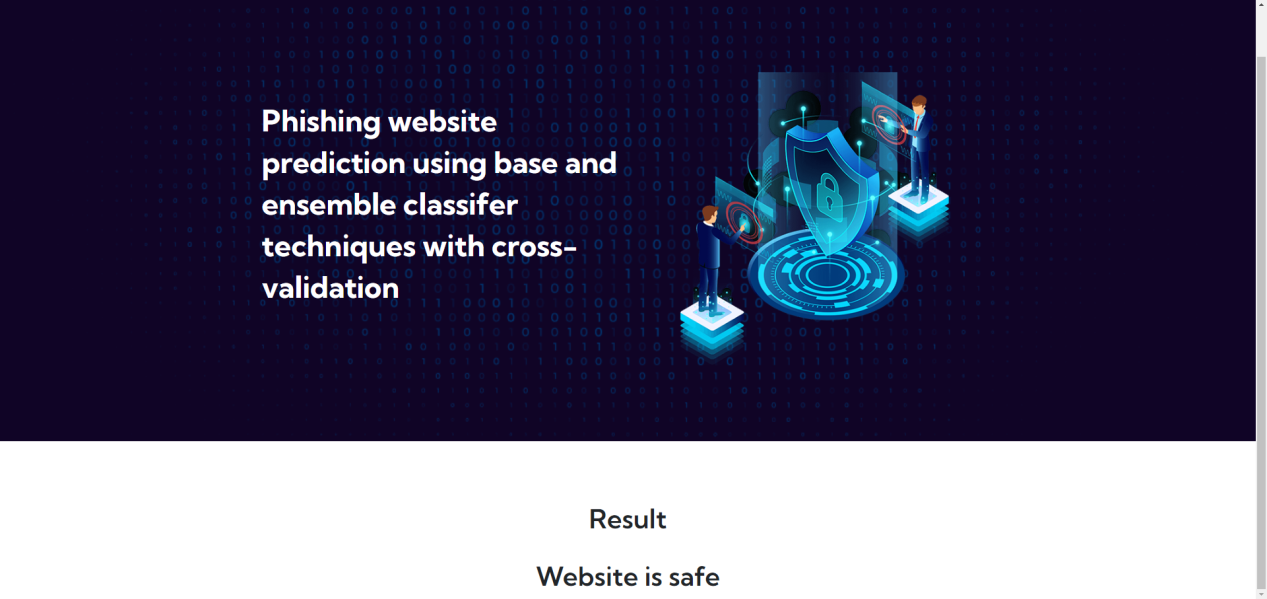












**CONCLUSION**

**10. CONCLUSION**

In conclusion, this study underscores the pressing need for robust cybersecurity measures in the face of escalating phishing threats within the expanding digital landscape. The comprehensive evaluation of diverse machine learning algorithms, including CNN, RNN, Logistic Regression, Decision Tree, SVM, KNN, Naive Bayes, LDA, and a Voting Classifier, reveals the pivotal role of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in achieving notably high accuracy levels. The proposed ensemble-based approach harnesses the collective strengths of these algorithms to fortify the system's capability in detecting and predicting phishing activities. The findings emphasize the significance of staying ahead of evolving cyber threats, with CNN and RNN emerging as powerful tools in mitigating the risks associated with phishing attacks. By combining advanced classifiers, the proposed system offers a multifaceted defense mechanism against the diverse tactics employed by cybercriminals. The incorporation of cross-validation further validates the reliability of the ensemble classifiers. This research not only contributes valuable insights to the field of cybersecurity but also advocates for the adoption of state-of-the-art techniques to ensure a more secure online environment for users across various internet domains.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] 2020 Phishing Attack Landscape. https://info.greathorn.com/report-2020- phishing-attack-landscape. Accessed 13 Sept 2020

[2] Abdelhamid N, Abdel-jaber H (2017) Learning comparison based on models content and features, pp 72–77

[3] Abutair H et al (2019) CBR-PDS: a case-based reasoning phishing detection system. J Ambient Intell Humaniz Comput 10(7):2593–2606. https://doi. org/10.1007/s12652-018-0736-0

[4] Ali W (2017) Phishing website detection based on supervised machine learning with wrapper features selection. Int J Adv Comput Sci Appl 8(9):72–78

[5] Breiman L (2001) Random forests. Mach Learn 45(1):5–32. [https://doi.org/10. 1023/A:1010933404324](https://doi.org/10.%201023/A:1010933404324)

[6] Chin T et al (2018) Phishlimiter: a phishing detection and mitigation approach using software-defned networking. IEEE Access 6:42513–42531. https:// doi.org/10.1109/ACCESS.2018.2837889

[7] Cox DR (1966) Research papers in probability and statistics (Festschrift for J. Neyman). Wiley, London

[8] Cramer JS (2005) The origins of logistic regression. SSRN Electron J. https://doi. org/10.2139/ssrn.360300

[9] El Aassal A et al (2020) An in-depth benchmarking and evaluation of phishing detection research for security needs. IEEE Access 8:22170–22192. https:// doi.org/10.1109/ACCESS.2020.2969780

[10] Fletcher R, Reeves CM (1954) The use of multiple measurements in taxonomic problems. Ann Eugen 1(1):75

[11] Friedman JH (1997): 2 What is data mining ? 1 Introduction. Statistics (Ber)

[12] Gupta BB et al (2021) A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. Comput Commun 175:47–57. <https://doi.org/10.1016/j.comcom.2021.04.023>

[13] Gupta S, Singhal A (2018) Dynamic classifcation mining techniques for predicting phishing URL. In: Advances in intelligent systems and computing. Springer, pp 537–546. <https://doi.org/10.1007/978-981-10-5699-4_50>

[14] Hong J et al (2020) Phishing URL detection with lexical features and blacklisted domains. In: Adaptive autonomous secure cyber systems. Springer, pp 253–267. <https://doi.org/10.1007/978-3-030-33432-1_12>

[15] Jain AK, Gupta BB (2018a) Towards detection of phishing websites on client-side using machine learning based approach. Telecommun Syst 68:687–700. <https://doi.org/10.1007/s11235-017-0414-0>

[16] Jain AK, Gupta BB (2018b) PHISH-SAFE : URL features-based phishing detection system using machine learning. Springer.https://doi.org/10.1007/ 978-981-10-8536-9

[17] Kleinberg EM (2000) On the algorithmic implementation of stochastic discrimination. IEEE Trans Pattern Anal Mach Intell 22(5):473–490. https://doi. org/10.1109/34.857004

[18] Koray O et al (2019) Machine learning based phishing detection from URLs. Expert Syst Appl 117:345–357. <https://doi.org/10.1016/j.eswa.2018.09.029>

[19] Kumar A, Gupta JBB (2018) A machine learning based approach for phishing detection using hyperlinks information Number of Unique Phishing Sites Detected. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/ s12652-018-0798-z

[20] Leng K et al (2019) A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Inf Sci 484:153–166. <https://doi.org/10.1016/j.ins.2019.01.064>

[21] Logistic regression—Wikipedia. https://en.wikipedia.org/wiki/Logistic\_regre ssion#cite\_note-4. Accessed 19 April 2020

[22] Mao J (2019) Phishing page detection via learning classifers from page layout feature

[23] Master Machine Learning Algorithms. https://machinelearningmastery.com/ master-machine-learning-algorithms/. Accessed 16 June 2020

[24] McFadden D (1973) Frontiers in econometrics. Academic Press, New York

[25] Module: tf.contrib | TensorFlow Core v1.15.0. https://www.tensorfow.org/versi ons/r1.15/api\_docs/python/tf/contrib?hl=JA. Accessed 18 Sept 2020

[26] Moghimi M et al (2016) New rule-based phishing detection method. Expert Syst Appl 53:231–242. <https://doi.org/10.1016/j.eswa.2016.01.028>

[27] Orunsolu AA et al (2020) A predictive model for phishing detection. J King Saud Univ Comput Inf Sci. <https://doi.org/10.1016/j.jksuci.2019.12.005>

[28] PhishTank | Join the fght against phishing. https://www.phishtank.com/. Accessed 20 April 2020

[29] Phishing website dataset | Kaggle, https://www.kaggle.com/akashkr/phishingwebsite-dataset/version/2#. Accessed 29 June 2020

[30] Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106. https:// doi.org/10.1007/bf00116251

[31] Radhakrishna Rao C (2011) Tests of signifcance in multivariate analysis. Biometrika 6(1):1–25

[32] Sahingoz OK et al (2019) Machine learning based phishing detection from URLs. Expert Syst Appl 117:345–357. [https://doi.org/10.1016/j.eswa.2018. 09.029](https://doi.org/10.1016/j.eswa.2018.%2009.029)

[33] Satapathy SK et al (2019) Classifcation of features for detecting phishing web sites based on machine learning techniques. Int J Innov Technol Explor Eng 8:424–430

[34] Shirazi H et al (2017) Fresh-Phish : a framework for auto-detection of phishing websites. <https://doi.org/10.1109/IRI.2017.40>

[35] Shiri A (2004) Introduction to modern information retrieval (2nd edition). Libr Rev 53(9):462–463. <https://doi.org/10.1108/00242530410565256>

[36] Sonowal G, Kuppusamy KS (2020) PhiDMA—a phishing detection model with multi-flter approach. J King Saud Univ Comput Inf Sci 32(1):99–112. <https://doi.org/10.1016/j.jksuci.2017.07.005>

[37] Srinivasa R et al (2019) Two level fltering mechanism to detect phishing sites using lightweight visual similarity approach. J Ambient Intell Humaniz Comput. <https://doi.org/10.1007/s12652-019-01637-z>

[38] Theil H (1969) A multinomial extension of the linear logit model. Int Econ Rev (philadelphia) 10(3):251. <https://doi.org/10.2307/2525642>

[39]UCI Machine Learning Repository: Phishing Websites Data Set. http://archive. ics.uci.edu/ml/datasets/Phishing+Websites. Accessed 19 April 2020

[40] Varoquaux G et al (2015) Scikit-learn. GetMobile Mob. Comput Commun 19(1):29–33. <https://doi.org/10.1145/2786984.2786995>

[41] WHOIS API gives access to domain registration records | WhoisXML API. https://whois.whoisxmlapi.com/. Accessed 18 Sept 2020