PHISHING WEBSITE DETECTION USING MACHINE LEARNING ALGORITHMS

A MAJOR PROJECT REPORT

Submitted by

KOTI MUNI YOGESWAR REDDY
[RA1911003010876]
MEDIDA SHIVANANDA REDDY
[RA1911003010869]

Under the guidance of

Dr. D. Vinod
(Assistant Professor, CTECH)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section3 of UGC Act,1956)

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "**PHISHING WEBSITE DETECTION USING MACHINE LEARNING ALGORITHMS**" is the bonafide work of "**KOTI MUNI YOGESWAR REDDY [RA1911003010876], MEDIDA SHIVANANDA REDDY [RA1911003010869]**" who carried out the major project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form any other project report or dissertation on the basis on which a degree or award was conferred on an earlier occasion on this or any other candidate.

DR. D. VINOD
SUPERVISOR
Assistant Professor
Department of Computing Technologies

DR. K. SREEKUMAR
PANEL HEAD
Associate Professor
Department of Computing Technologies

DR. M. PUSPHALATHA
HEAD OF THE DEPARTMENT
Professor
Department of Computing Technologies

INTERNAL EXAMINER

EXTERNAL EXAMINER

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course : B.Tech in Computer Science and Engineering

Student Names : Koti Muni Yogeswar Reddy, Medida Shivananda Reddy

Registration Number : RA1911003010876, RA1911003010869

Title of Work : Phishing Website Detection using Machine Learning Algorithms

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate.
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present

- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website.

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is our own work, except where indicated by referring, and that I have followed the good academic practices noted above. KOTI MUNI YOGESWAR REDDY — RA1911003010876 MEDIDA SHIVANANDA REDDY — RA1911003010869 Date:-
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T.V.Gopal**, for his valuable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department, Dr. M. Pushpalatha Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Panel Head, **Dr. K. Sreekumar**, Associate professor, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor, **Dr. V. Deepan Chakravarthy**, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. D. Vinod**, Assistant professor, Department of Computing Technologies SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his/her/their mentorship. She provided me with the freedom and support to explore the research topics of my interest. Their passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Department of Computing Technologies staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support. and encouragement.

Koti Muni Yogeswar Reddy [RA1911003010876]

Medida Shivananda Reddy [RA1811027010030869]

ABSTRACT

The objective of the work is to propose phishing website detection using machine learning. The online shoppers often provide sensitive information such as passwords, usernames, and credit card details, which makes them vulnerable to phishing websites that use such information for malicious purposes. To combat this problem, we have introduced an intelligent, adaptable, and efficient system that leverages machine learning techniques to detect and predict phishing websites. Our system utilizes a classification algorithm and methods for identifying phishing criteria to determine their authenticity. By analysing key features such as URL and domain identity, security, and encryption criteria, our system achieves a high rate of phishing detection. The system can be used by e-commerce businesses to secure their transaction process, and the machine learning algorithm used in our system outperforms conventional classification algorithms, providing online shoppers with a secure and confident shopping experience, our system will use a machine learning algorithm to detect fraudulent websites. This application can be utilized by various e-commerce enterprises to secure the entire transaction process. Comparing this system's machine learning method to other conventional classification algorithms, it performs better. The user can buy goods without any hesitation online with the aid of this method.

Keywords: Phishing, Personal information, Machine Learning, Random Foresr,Direct links, Phishing domain characteristics

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AES	Advanced Encryption Standard
ANN	Artificial Neural Network
CSS	Cascading Style Sheet
CV	Computer Vision
DB	Data Base
DNA	Deoxyribo Neucleic Acid
SQL	Structured Query Language
SVM	Support Vector Machine
HTML	Hypertext Markup Language
URL	Uniform Resource Locator
UI	User Interface
AI	Artificial Intelligence
ML	Machine Learning

CHAPTER1

INTRODUCTION

1.1 General

Detecting and preventing phishing websites is an important research area, as there are various techniques used for effectively detecting and protecting individuals and organizations. One crucial aspect of phishing is the role of URLs. URLs are a major area through which websites are initiated, and pages are redirected to the following page through the link. Redirecting pages is a vulnerable concept in phishing, where pages are redirected through hyperlinks to either a legitimate website or a phishing site.

As the number of different types of phishing sites continues to increase, many researchers have focused on detecting and preventing phishing sites. Various phishing techniques used on websites have been modified or newly proposed by the research community to protect individuals and organizations from significant losses. Our research also aims to prevent phishing in emails. Research articles have shown that confidential information is collected through emails and users are lured through attractive advertisements. This has led us to conduct a literature review on phishing detection and prevention techniques for both the consumer and server sides.

1.2 Objective

- To Research Different Automatic Phishing Detection Techniques
- To select a method and define a solution using suitable machine learning techniques.
- To choose a suitable dataset for the issue at hand
- To use the proper algorithms to defeat phishing scams.

1.3 Software Requirements

At the end of the analysis phase, a software requirements specification is developed to refine the functions and performance assigned to the software in system engineering. This is done by creating a detailed functional representation of the system's behavior, establishing performance requirements and

design constraints, and identifying suitable validation criteria.

Operating system	: Windows 10
IDE	: anaconda navigator
Coding Language	: python

1.4 Hardware Requirements

System	: Pentium IV 2.4 GHz
Hard Disk	: 40 GB
Floppy Drive	: 1.44 Mb
Monitor	: 15 VGA Colour
Mouse	: Logitech
Ram	: 512 Mb

CHAPTER 2

LITERATURE SURVEY

1. Systematization of Knowledge (SoK): A Systematic Review of Software- Based

Phishing is a kind of cyberattack that makes use of social engineering and other cutting-edge methods to steal consumers' personal information from websites. The number of distinct phishing websites detected has increased at an average annual rate of 36.29% during the last six years and nine for the identifying phishing attempts based on the content, network, and URL properties. The methodology, techniques, and evaluation standards employed in the approaches differ greatly.

2. Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity

Social networks have emerged as a widely accepted medium for individuals to connect with each other. However, due to the abundance of sensitive data available on these platforms, safeguarding user privacy has become an important research topic. Despite being an old tactic, phishing attacks continue to be a significant cause of privacy violations. In a web-based phishing assault, hackers create fake web pages that resemble significant websites, such as social network portals, to deceive users into disclosing private information such as passwords, social security numbers, creditcard numbers, and other sensitive information.

4. Spotting phishing attempts using DNS poisoning based on network performance traits

Many current techniques for detecting phishing attacks are not able to effectively detect dns-poisoning- based phishing attacks. To address this issue, a highly effective method has been proposed which uses network performance characteristics of websites to classify them. By comparing the effectiveness of the four classification methods linear discriminant analysis, naive Bayesian, k-nearest neighbour, and support vector machine, the proposed method's usefulness has been established. Over 10,000 actual pieces of routing data were seen by the approach over the course of a week.

5. An FMRI study of phishing and malware warnings found neural markers for cybersecurity

This paper explores the vulnerability of computer systems due to user behavior and decisions, which are influenced by the human brain. A neuroscience-based methodology was recognized to examine users' security performance and underlying neurological activity in order to better comprehend this. The study concentrated on two crucial security tasks: Recognizing a legitimate website from a phishing website and paying attention to security alerts. The study's findings offer suggestions for creating user-centered security systems to combat cybercrime.

CHAPTER 3

3.1 System Architecture

The suggested method of exploiting network performance characteristics of websites for classification is highly successful at detecting DNS-poisoning-based phishing attempts, which many existing phishing detection algorithms are unable to detect. Using more than 10,000 pieces of route data from actual observations, the study conducted experiments on four classification methods: support vector machine, naive Bayesian, K-nearest neighbor, and linear discriminant analysis. Made over the course of one week to validate the methodology. The method has been proven to be successful in detecting such attacks.

3.2 Architecture Diagram

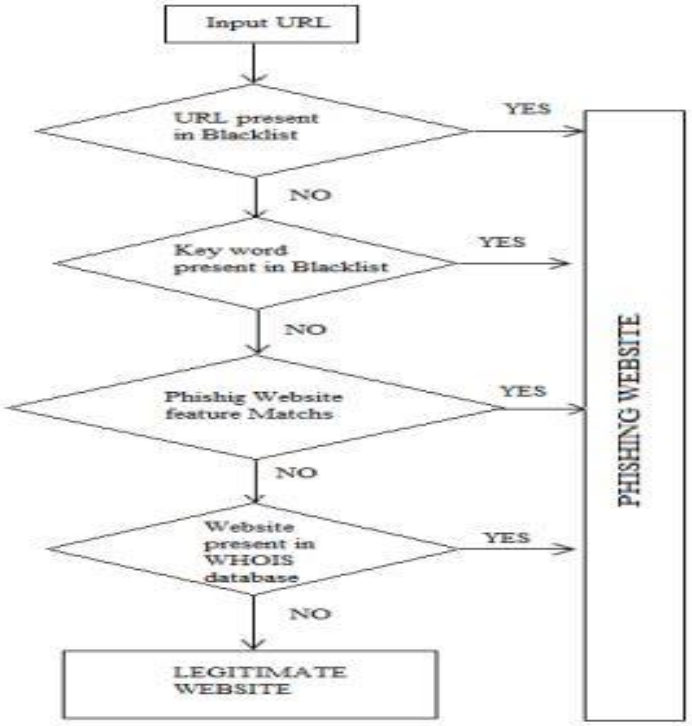


Fig 4.1 Architecture Diagram

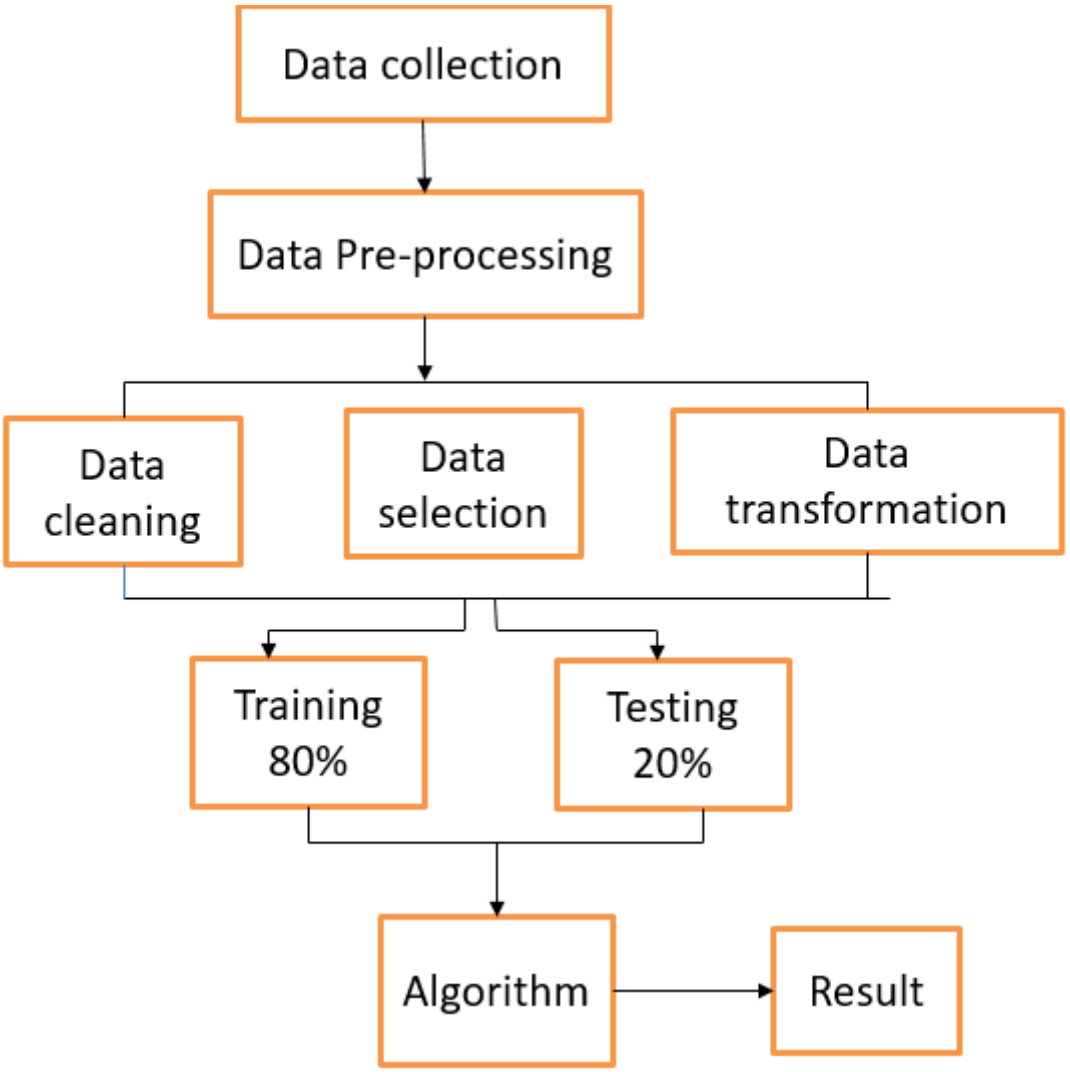


Fig 4.2 Flow Chart

3.3 ER Diagram:

- A large number of current phishing detection methods are not capable of detecting DNS-poisoning-based phishing attacks. To overcome this limitation, a new approach has been introduced that relies on using network performance features of websites to classify them.

- Using real-world routing data collected over the course of a week, the effectiveness of the study conducted an assessment and comparison of four classification techniques the performance of K-nearest neighbor, Support Vector Machine, Linear Discriminant Analysis, and Naive Bayesian models was evaluated to analyze their individual effectiveness.. This method has shown to be quite effective in identifying phishing assaults that use DNS poisoning.

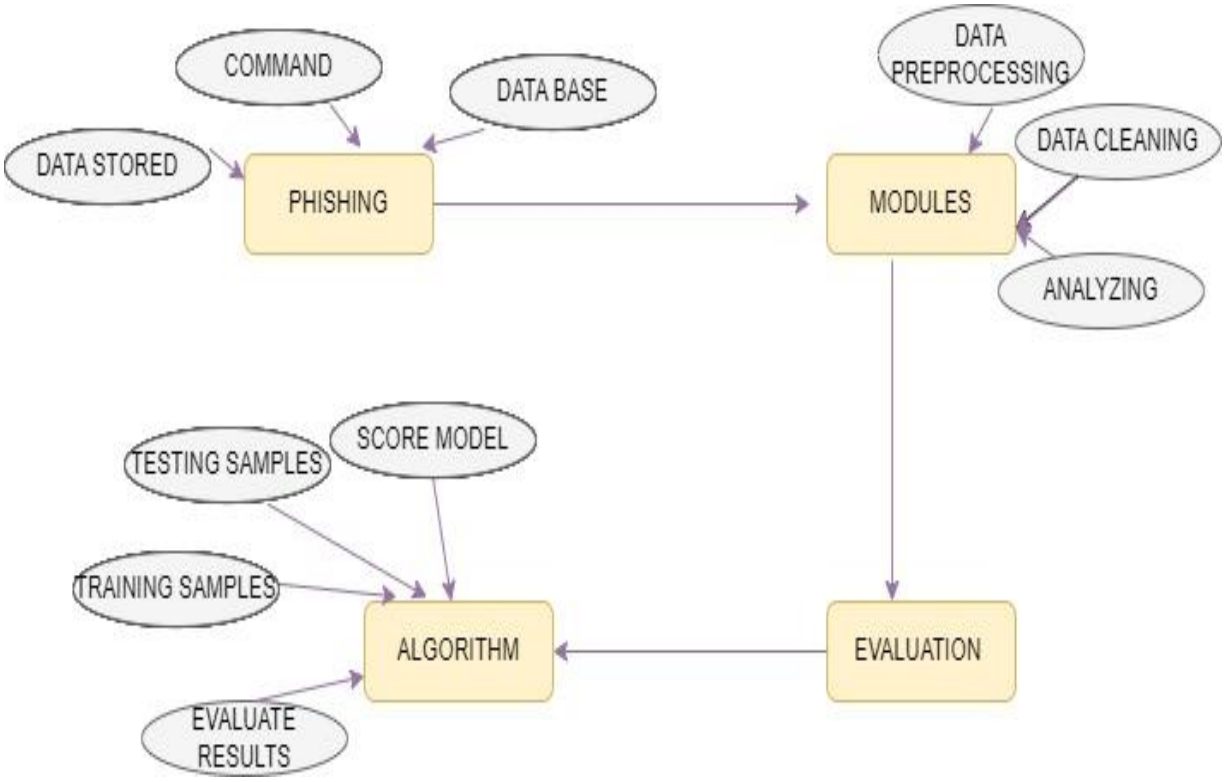


Fig 4.3 ER Diagram

3.4 Use Case:

A lot of current methods for detecting phishing attacks are not effective when it comes to Phishing attacks that utilize DNS poisoning as their method of operation.. This issue is addressed by a novel method that categorizes websites based on their network performance traits. Testing four classification methods utilizing more than 10,000 real-world routing information items observed for a week demonstrates the efficacy of this strategy. The outcomes demonstrate that this method is effective in identifying phishing assaults based on DNS poisoning.

In UML, use case diagrams depict the dynamic interactions between a system and its external actors or entities, and they are used to model these interactions. Actors, which can be either internal or external agents interacting with the system, are illustrated as stick figures in the diagram, while use cases, which represent the system's functions or actions towards a specific goal, are depicted as ovals. Arrows are used to portray the connections between actors and use cases. Overall, the use case diagram

Provides an encompassing view of the system's characteristics and the relationships between the system and its stakeholders.

Use case diagrams in software engineering are a graphical representation used to depict the interactions between the system and its actors or entities. These diagrams are created to capture a specific functionality of the system and illustrate a user's interaction with the system. They can be used to model the complete system by producing several diagrams, and they can also display various user types and the situations with which they interact. To offer a more thorough understanding of the system's functionality, use case diagrams can be integrated with other sorts of diagrams include class diagrams, activity diagrams, and sequence diagrams.

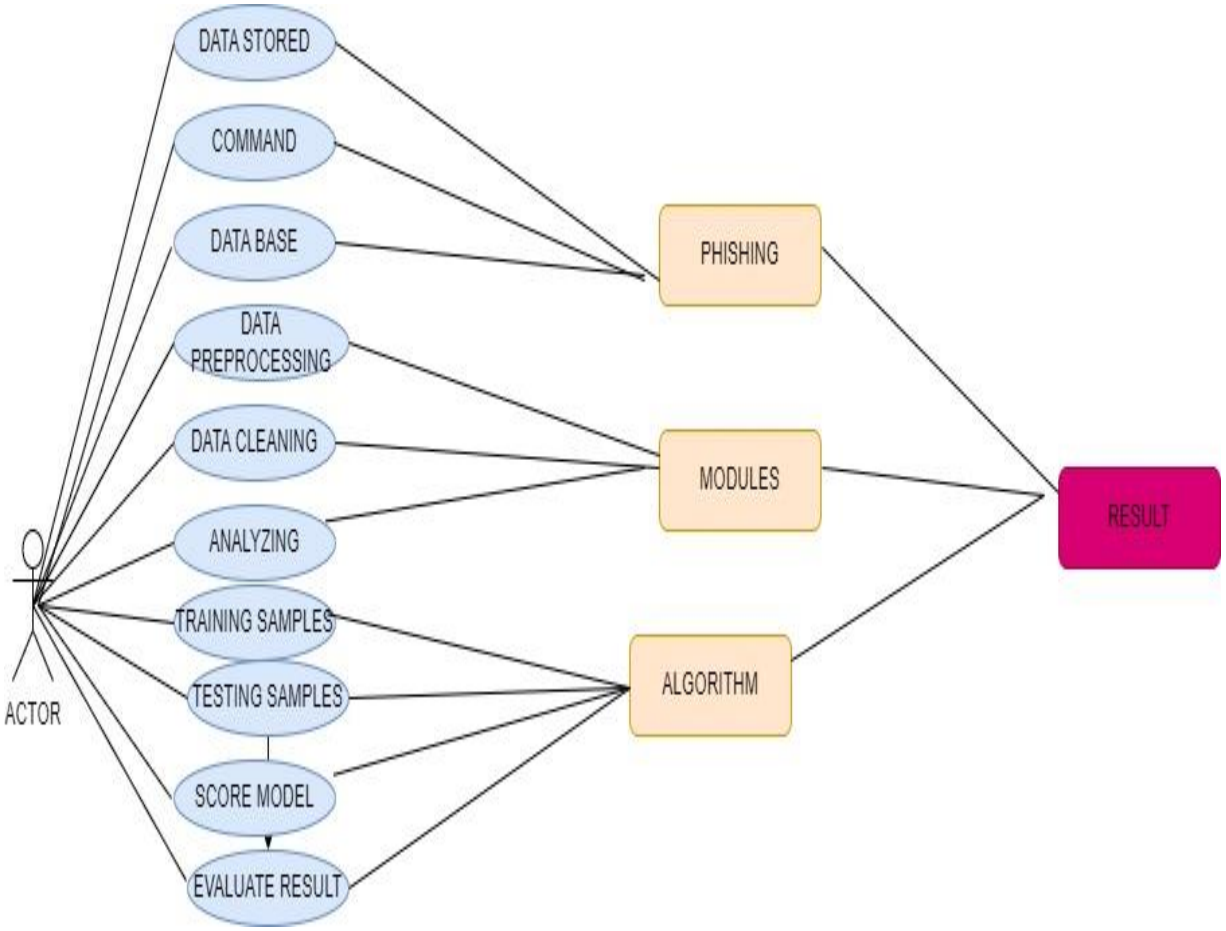


Fig 4.4 Use Case

3.5 Class Diagram:

A particular type of UML diagram known as a class diagram uses classes to demonstrate a system's static structure, their characteristics, methods, and connections among them. It provides a visual

illustration of the various components that make up a system, such as classes, objects, and associations. This diagram is primarily used to model the static view of an object-oriented system and to provide a high-level overview of the system's structure. Additionally, it can be utilized to demonstrate the implementation of software by illustrating the arrangement of classes, interfaces, and objects within the code.

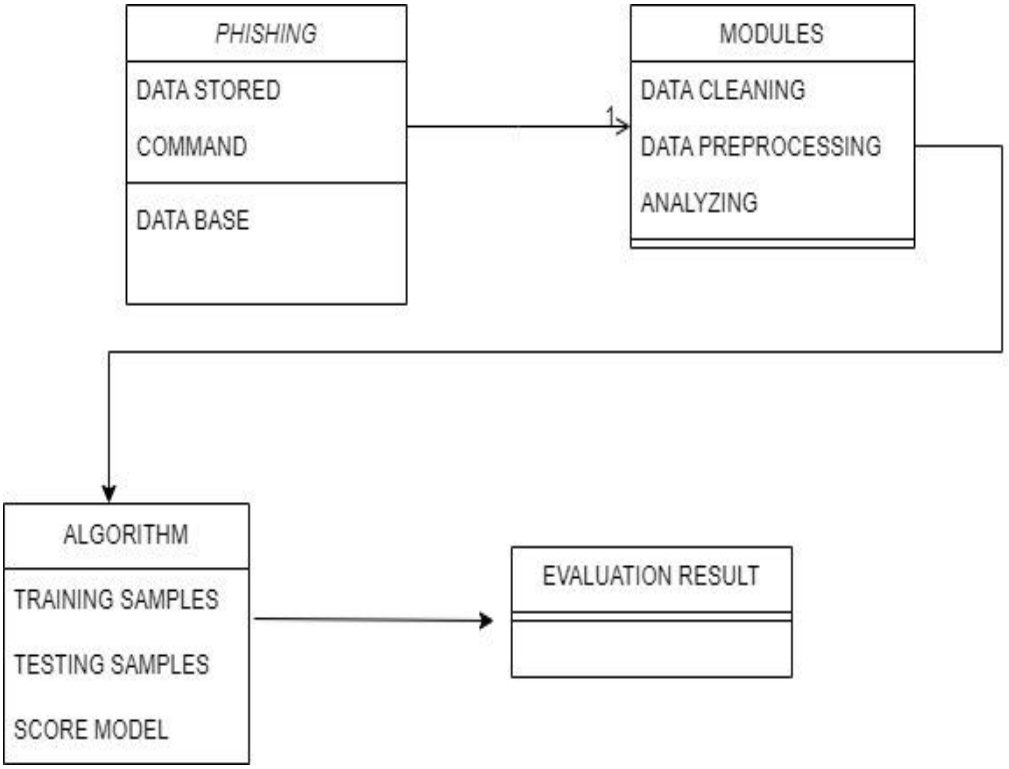


Fig 4.5 Class Diagram

3.6 Sequence Diagram:

The interactions between the objects or components of a system in a certain scenario or use case are shown using sequence diagrams, a type of behavior diagram in UML. They are especially helpful for the flow of events that take place when a use case is being executed and modelling the dynamic behavior of a system. Sequence diagrams show the objects involved in the interaction, their lifelines, and the messages exchanged between them, including the sequence and timing of the messages. This makes them an effective tool for understanding the behavior of a system and identifying potential issues or inefficiencies in the design. Sequence diagrams are particularly useful for visualizing and understanding complex interactions between objects, and for identifying potential issues or bottlenecks in a system for various communication flow. It can also be used to generate code or as a starting point for designing an application’s architecture.

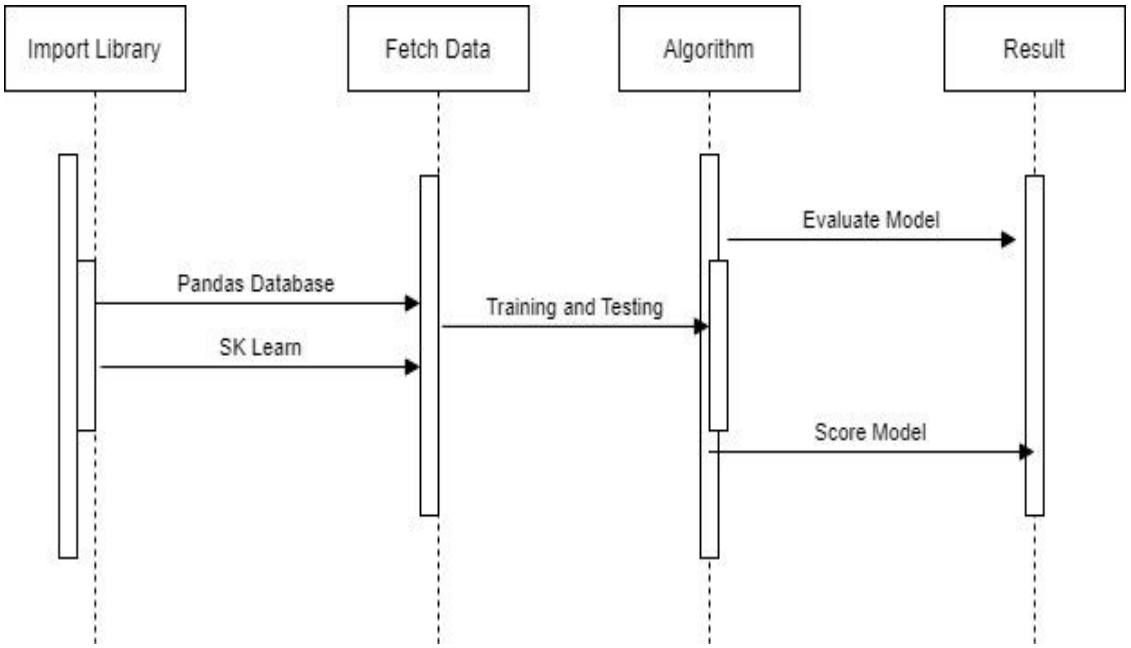


Fig 5.1 Sequence Diagram

3.7 Activity Diagram:

Activity diagrams are used to model the workflow of a system or process. They illustrate the sequence of actions and decisions involved in performing a certain task or activity. A diagram used in UML to model the workflow of a system is an activity diagram. In this kind of diagram, the nodes and edges stand in for the activities or decisions that were made during the workflow, and the edges show the control flow between the nodes. They are useful for analyzing and improving the efficiency of a system, as well as for communicating its functionality to stakeholders.

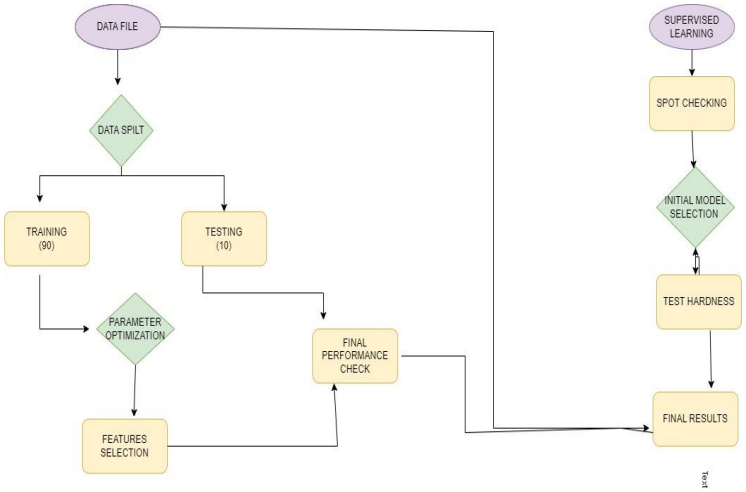


Fig 5.2 Activity Diagram

3.8 List of Modules

Modules:

- Data Collection
- Pre-processing
- Algorithm
- Logistic Regression
- Decision Tree Algorithm
- Prediction
- Result

Data Collection and Pre-processing

- Collecting high-quality data is a critical aspect of developing an accurate machine learning model. The method of gathering data is essential since the effectiveness of the model will be greatly influenced by the quality of the data used. Therefore, it is crucial to guarantee that the information gathered is pertinent, precise, and indicative of the issue at hand. This involves identifying the right sources of data, designing appropriate data collection methods

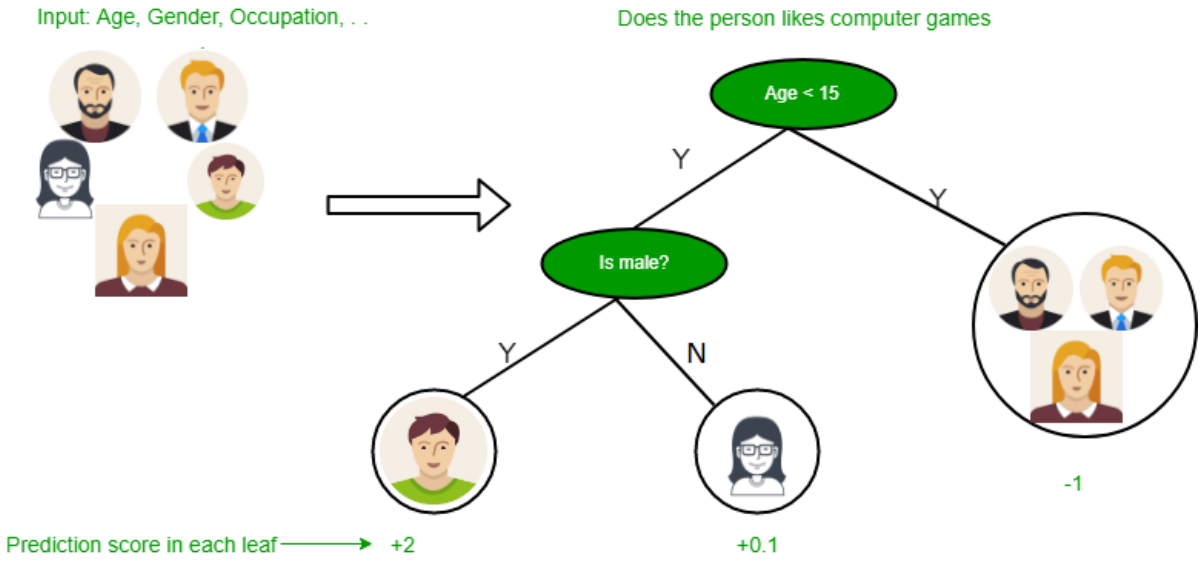
- ensuring that the data collected is properly labelled and annotated for use in training the model. Ultimately The quality of the data gathered, and the methods employed to get it frequently determine the success of a machine learning project. However, it is not uncommon for the data to be noisy, meaning that there may be missing or inaccurate values that need to be addressed before the data can be analysed.
- Cleaning and converting the data is the process of "data pre-processing."
  to address any issues before it is used in a machine learning model.
- This can involve tasks such as removing duplicates, handling missing values, scaling the data, and selecting relevant features for analysis.

Algorithm:

Decision Tree

A common method for supervised classification is for framing the learning methods using

decision tree. The fundamental idea is to use the decision tree and a collection of attributes and their values t. The data is regularly divided into clustered or dense regions and bare or sparse regions. The generated decision tree to find out the class label or target value of new, unanticipated instances by traversing the tree based on the values of the instances features.



. **Fig 5.3** Assumptions While using Decision Tree

- Decision trees are a popular choice for representing boolean functions because they can handle both discrete and continuous input variables, and they are capable of capturing complex relationships between inputs and outputs.
- In addition to classification problems, decision trees can also be used for regression problems by predicting a continuous output variable instead of a discrete class label.

The following are some suppositions we made when employing a decision tree:

1. As the tree's root node, start with the complete training set.
2. Select the best attribute to split the data. This is usually done using an attribute selection measure such as information gain or gain ratio.
3. Create a child node for each potential value of the attribute and divide the data according to that value.
4. Recursively apply steps 2 and 3 to each child node, using only the instances that reach that node.
5. Stop partitioning a branch when all instances in the branch belong to the same class, or when no more attributes are left to split.
6. Handle missing values by either ignoring the instance or using an estimation method to replace

the missing value.

7. Prune the decision tree to reduce overfitting, which is the phenomenon where In cases where the tree becomes excessively intricate and overfits the training data, it tends to exhibit poor generalization capabilities when confronted with unfamiliar, unseen data.

8. The ID3 algorithm aims to construct a decision tree that can correctly classify new, unseen instances by recursively partitioning the feature space into regions where the class label is consistent. The algorithm selects attributes that best separate the training data into the different classes, and uses these attributes to create the decision tree. By following the decision tree, the algorithm can then classify new instances based on their attribute values.

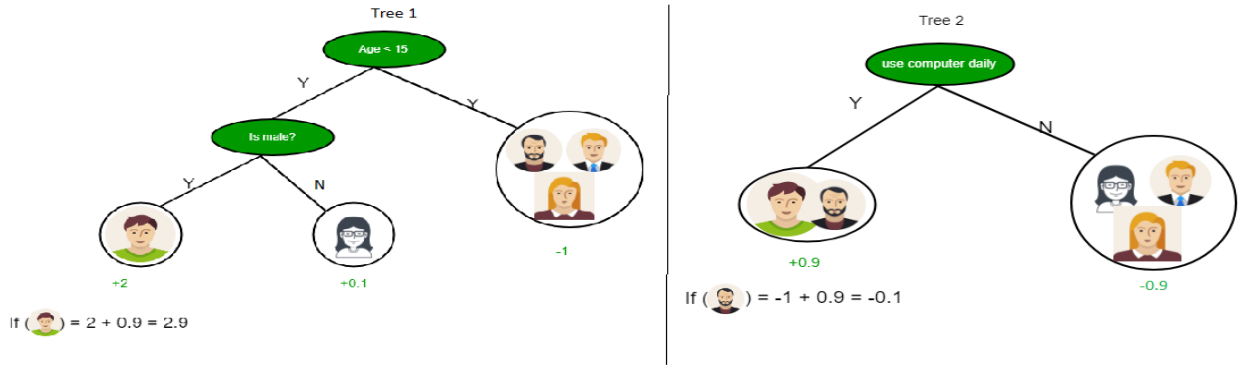


Fig 5.4 Employing a Decision Tree

The two popular attribute selection measures for decision tree are:

1. Information Gain is a metric used in decision tree algorithms to quantify the decrease in entropy or increase in purity that results from splitting the data based on a specific attribute. The decision tree's root node for that level is chosen based on the attribute with the greatest information benefit.

2. The Gini Index figures out the probability of misclassifying a randomly chosen element from the set. The feature with the lowest Gini index serves as the root node for that level.

3. Information gain and Gini index are the two popular attribute selection measures used in decision tree learning.

4. Information gain is a measure of the amount of information that a feature provides about the class. of a set of examples. The foundation of it is the idea of entropy, which is a measure of the impurity or uncertainty.

Information gain:

It is estimate of the amount of uncertainty or entropy that is reduced when splitting a dataset based on a particular attribute. The calculation involves comparing the entropy of the dataset before and after the split. The characteristic that yields the most information is chosen to be the root node for that level in the decision tree.

Entropy:

A measure of an example set's impurity is called entropy. It is described as the anticipated value of the data in each scenario, and it ranges from 0 (when all the examples belong to the same class) to 1 (when the examples are evenly distributed across all classes). High entropy means that there is a lot of uncertainty in the set of examples, and low entropy means that there is little uncertainty (i.e., the examples are relatively pure). In decision trees, entropy is used as a gauge of a node's impurity, and it is used to select the most appropriate property to split the data on at each step of the tree-building process.

Building Decision Tree using Information Gain

The essentials:

The decision tree learning algorithm starts with all training instances at the root node and selects the attribute with the highest information gain to label each node. The algorithm ensures that each root-to-leaf path contains a unique attribute. Each subtree is built recursively using the subset of training examples that would be categorized along that branch of the tree. A node is given the appropriate name depending on whether all instances are positive or negative. A majority of the training instances at the node that are still there vote to label it if there are no characteristics left. When no more instances are present, the parent's training instances vote to label the node with a majority.

Building Decision Tree using Information Gain

The essentials:

Using the ID3 algorithm for supervised learning, the following guidelines can be utilised to build a decision tree:

- The root node first receives all of the training instance assignments.
- Choose the attribute to assign each node using the knowledge you have gained.
- Check to see if any root-to-leaf paths have the same discrete property more than once.
- Create each subtree iteratively using the subset of training cases that would be categorized

along that branch of the tree.

- Label the node with "yes" or "no" as appropriate if all the training cases that are still there are
- positive or negative.
- If a node cannot be labelled with any more attributes, the majority opinion of the training examples present at that node should be used.
- If a node has no more instances, it should be labelled based on the training instances of its parent's nodes.

Linear Regression

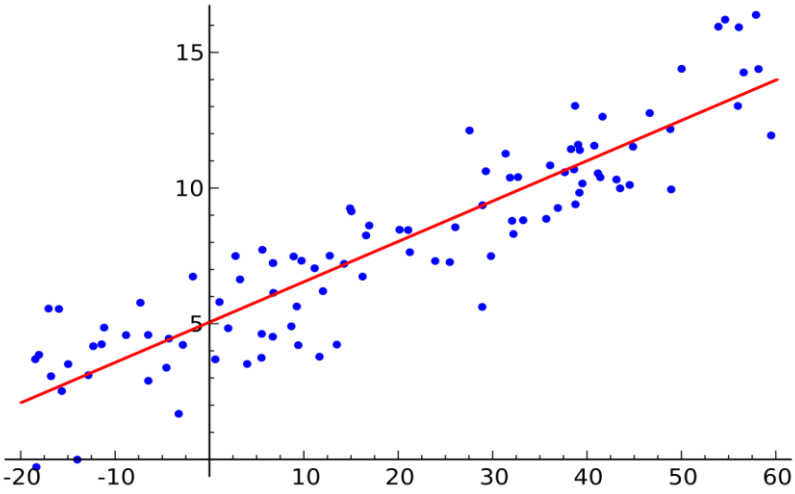


Fig 5.5 The Power Straight Line

A continuous numerical output variable can be estimated or predicted using regression, a statistical approach, based on one or more input factors. These variables might be continuous, discrete, or categorical. It can analyze and anticipate the behavior of a system or phenomena, making it useful in a gauge of a node's impurity, and it is used to select the most appropriate property to split of wide range of disciplines including finance, economics, engineering, and the social and natural sciences. The type of data and research issue influence the regression method selection. Examples of regression methods include linear, logistic, polynomial, and multiple regression.

The linear relationship between one independent variable and one dependent variable is ascertained using the simple linear regression approach. The best-fit is found by minimising the addition of squared deviations in between the expected and found y-values. The dependent variable's value can then be

predicted using the line for a specific value of the independent variable.

Naïve Bayes Classifier Algorithm

By locating the best-fit line that minimizes simple linear regression is a statistical technique that analyzes the association between two continuous variables: an independent variable (x) and a dependent variable (y). It involves calculating the total squared deviation between the observed y-values and the predicted y-values by the linear regression line is a measure used to evaluate the line's forecasting ability for a specific value of the independent variable. the value of the dependent variable.

Why is it called Naïve Bayes?

The Naive Bayes algorithm is a computer programme that uses Bayes' Theorem. method used for categorization tasks. Because it presumes that all features are independent of one another, which may not be true in practise, it is referred to as "naive." Nevertheless, the Naive Bayes method has been widely used in many other disciplines, such as spam filtering and text classification, and has proven to be successful in many instances despite this shortcoming.

Bayes' Theorem:

- The formula for Bayes' theorem is given as:
- P(A|B) = P(B|A) * P(A) / P(B)

- P(A|B) is the probability of hypothesis A given the observed evidence B
- P(B|A) is the probability of observing the evidence B given that the hypothesis A is true
- P(A) is the prior probability of hypothesis A being true
- P(B) is the prior probability of observing the evidence B
- Bayes' theorem is widely used in various fields such as statistics, machine learning, artificial intelligence, and decision theory.

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

(1)

Working of Naïve Bayes' Classifier:

1. Collect the dataset of weather conditions and the corresponding "Play" variable for different days.
2. Calculate the prior probability of playing and not playing, i.e., the probability of "Play" variable being true or false in the dataset.
3. Calculate the conditional probability of playing and not playing given each weather condition, i.e., the probability of "Play" variable being true or false given each weather feature.
4. Multiply the prior probability with the conditional probability for each weather feature to obtain the posterior probability.
5. For a given set of weather conditions, calculate the posterior probability of playing and not playing using Bayes' theorem.
6. Classify the instance as "Play" if the posterior probability of playing is greater than the posterior probability of not playing, and vice versa.

Note that the Naïve Bayes' Classifier assumes that all the weather conditions are independent of each other, which may not be true in practice.

Advantages of Naïve Bayes Classifier:

- For predicting the class of datasets, Nave Bayes is a machine learning technique that is quick and simple to apply.
- Both binary and multi-class classifications can use this approach., and it often performs better than other algorithms in multi-class predictions.
- Naïve Bayes is particularly popular for text classification problems, where it can be used to classify documents based on their content.

Disadvantages of Naïve Bayes Classifier:

The Naive Bayes method is so-called "naive" because it simplifies by assuming that all features are independent of one another. The technique has proven to be highly useful in many real-world situations, even if this assumption may not always be accurate.

This is known as the "naive" assumption, and it makes the algorithm computationally efficient and easy to implement. However, it also means that Naive Bayes cannot capture the complex relationships between features that might exist in some datasets.

Applications of Naïve Bayes Classifier:

The probabilistic technique known as the Naive Bayes classifier has proven to be successful in a number of applications. Credit scoring is an example of one such use, where the algorithm is applied to determine a borrower's creditworthiness based on a variety of variables including income, debt, and credit history. Using symptoms and other medical indicators, it is also utilised in the classification of medical data to identify diseases.. Additionally, the algorithm is useful for real-time predictions, such as in stock price prediction or weather forecasting. Another common application is where the technique is used for text classification tasks like spam filtering and sentiment analysis can determine whether a given text is positive or negative.

Types of Naïve Bayes Model:

The Naive Bayes method is a well-likedimachineilearning technique for classification applications. Gaussian, Multinomial and Bernoulli are the three primary varieties of Naive Bayes models. Machine learning frequently uses the Gaussian, Multinomial, and Bernoulli models for classification tasks. The Gaussian model presupposes that the dataset's characteristics have a normal distribution and that samples of the values for continuous predictors are taken from this distribution. The Multinomial model, which posits that word frequencies are multinomially distributed, is largely employed for document classification. This methodology classifies a document into a particular category, like politics or sports. Similar to the Multinomial model, the Bernoulli model also relies on independent Boolean predictor variables. This technique is also applied to the classification of documents, determining determine whether a word is contained in a document or not.

The "user data" dataset, which we've utilized in other classification models, may be utilized to create the Naive Bayes algorithm in Python. Next, we can contrast the outcomes of the Naive Bayes model with those of other models. The steps involved in putting the algorithm into practise are as follows:

1. Data Pre-processing: The first phase, which is related to data pre-processing, is to get the data ready and preprocess it so that it can be used effectively in the code.
2. The Naive Bayes model is subsequently fitted to the training set in step two.
3. Generating predictions for the test set and evaluating the results: The Naive Bayes model is used to forecast the test set results after being trained on the training set.
4. Developing a Confusion Matrix: To assess the precision of the predictions made by the Naive Bayes model, a confusion matrix is developed.
5. Creating a visual representation of the results obtained from the training set: To gauge how

effectively the model is working, the training set result is visualized.
6. Visualizing the Test Set Result in order to assess and compare the effectiveness... of the Naive Bayes model with other models, the test set result is lastly visualized.

Naive Bayes is a straightforward but efficient method that may be used for classification tasks in a variety of fields, including credit scoring and the classification of medical data. Due to its eager learning methodology, it is highly suited for real-time predictions and well-liked for text categorization issues like spam filtering and sentiment analysis.

Support Vector Machine (SVM):

Support Vector Machines (SVMs) are among the frequently employed machine learning approaches for regression challenges and classification research. The optimum hyperplane that divides the various classes of data in SVM is found. Here are a few of the benefits and drawbacks of utilizing SVM.

Advantages:

1. SVM is effective in high-dimensional spaces, making it suitable for complex data sets with a huge number of features. SVM has a strong theoretical foundation, which means that the algorithm's performance is mathematically guaranteed under certain conditions.
2. SVM is versatile and can be used for various types of data sets, including both linear and nonlinear data.
3. SVM is less prone to overfitting compared to other machine learning algorithms, as it seeks to maximize the margin between different classes.
4. When the classes are well separated, SVM can provide very accurate results.
5. SVM can be used for both regression and classification problems.
6. SVM can work well with image data because it can handle high dimensional data.

Disadvantages:

1. When the classes in the data overlap or are not well separated, SVM may not perform well. In these cases, other algorithms such as logistic regression or decision trees may be more suitable.
2. Choosing the right kernel for SVM can be a difficult task. The kernel determines the shape of the decision boundary and can greatly affect the performance of the algorithm.
3. SVM can take a long time to train on large datasets, especially when using a non-linear kernel.
4. SVM is not a probabilistic model, so it cannot provide a probability estimate for a given class. This

can make it difficult to interpret the results.

5. SVM can be more complex than other algorithms such as decision trees, which can make it difficult to understand and interpret the model.

SVM Algorithm:

The well-known machine learning For categorization, Support Vector Machines (SVM) can be employed. and regression analysis. Its main goal is to accurately identify the best decision boundary or hyperplane that splits a dataset's numerous classes. To create the hyperplane that best divides the n-dimensional space into different classes, SVM chooses the support vectors, which are the extreme points or vectors. A decision boundary or hyperplane separates two distinct categories in the figure below., to better grasp this idea. In two dimensions, the hyperplane is represented by a line, and in three dimensions, by a plane. [image: SVM hyperplane diagram]

There are two categories in this diagram: orange squares and blue circles. The SVM method seeks to identify the hyperplane with the maximum practical distance between the various classes. The separation the margin refers to the distance between the decision boundary and the closest data points from each class, which are commonly referred to as support vectors. A robust classifier that is less sensitive to data noise and can generalise well to fresh data can be produced by SVM by maximising the margin. Once the hyperplane is created, the new data points can be classified based on which side of the hyperplane they belong to. If a new data point falls on one side of the hyperplane, it belongs to one category, and if it falls on the other side, it belongs to the other category.

Overall, SVM is a powerful algorithm that can efficiently handle high-dimensional data and creates into different categories.

A well-liked machine learning method called SVM is frequently employed for classification issues. The optimal decision boundary, or hyperplane, that may successfully divide several classes in n-dimensional space, must be created. In order to do this, the SVM algorithm chooses extreme examples, often referred to as support vectors, and uses these to identify the location and direction of the hyperplane.

Think about a situation where we need to categories a new species that resembles both cats and dogs. The SVM algorithm can learn to recognize different aspects of these animals and effectively categorize the new species by training the model who has a lot of pictures of cats and dogs.

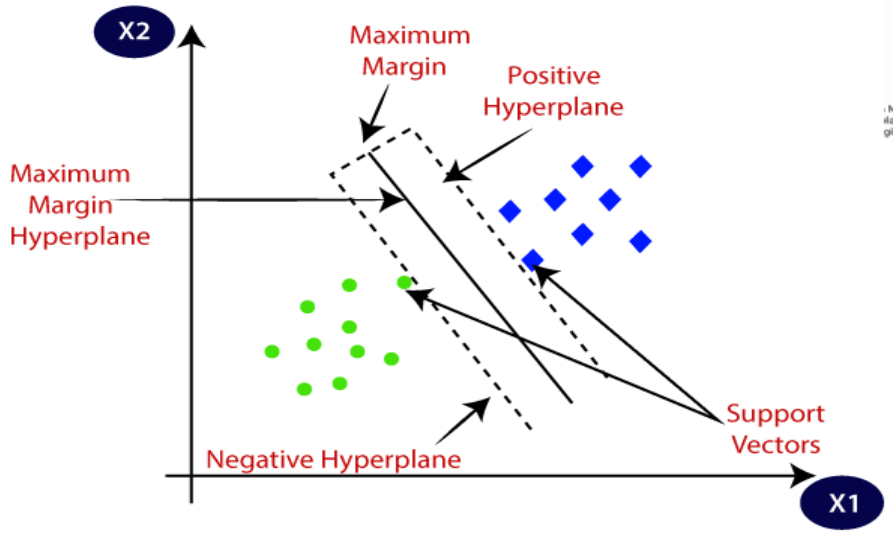


Fig 6.1 SVM

The program does this by creating a finalized boundary b/w the dog and cat data and selecting extreme scenarios to determine the support vectors. It can then categorize the new creature as either a dog or a cat based on the support vectors.

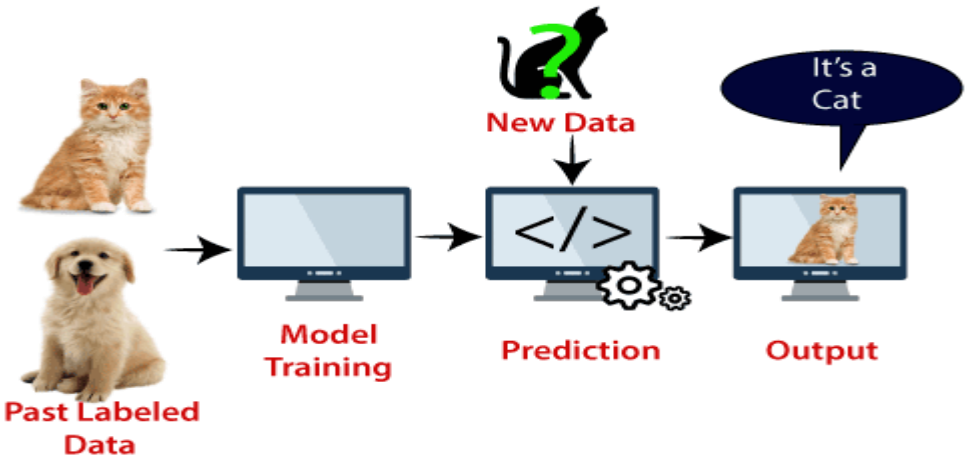


Fig 6.2 Working of SVM

SVM is a well-liked technique for classifying data and doing regression analysis. Linear SVM and Non-Linear SVM are the two categories into which SVM may be divided. For data that cannot be separated linearly, non-linear SVM is employed., whereas Linear SVM is appropriate for data that can be separated linearly. A hyperplane in n-dimensional space is produced by the SVM algorithm and serves as the optimal decision boundary for class separation. The distance between the nearest data

points to the hyperplane, sometimes referred to as Support Vectors, is the hyperplane's greatest margin. An example of the SVM algorithm uses a dataset. Support vectors aid in setting the decision border or hyperplane that the SVM Algorithm constructs the data points will be split into two classes.. Applications of the SVM algorithm include face identification, picture classification, and text classification.

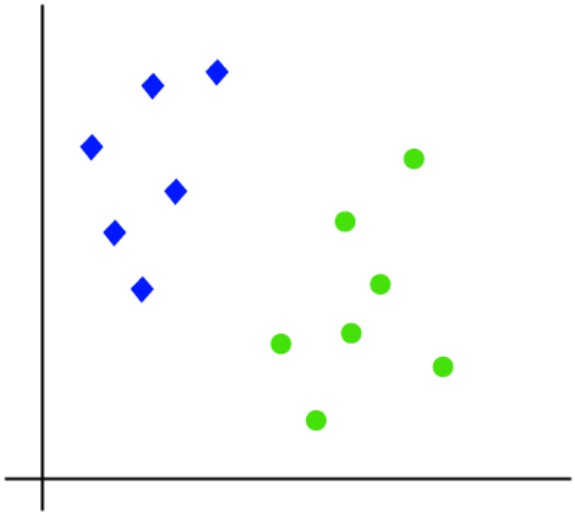


Fig 6.3 Linear SVM

A straight line can be utilized to divide the two classes in the provided 2-dimensional space. However, as seen in the illustration, these classes might be divided by more than one line. The SVM algorithm seeks the optimum hyperplane that maximizes the margin, often known as the distance between the closest support vectors and the hyperplane from each class. The optimal decision boundary for classification is regarded as the hyperplane with the largest margin. Non-linear SVM is utilized when data cannot be linearly segregated, as opposed to linear SVM.

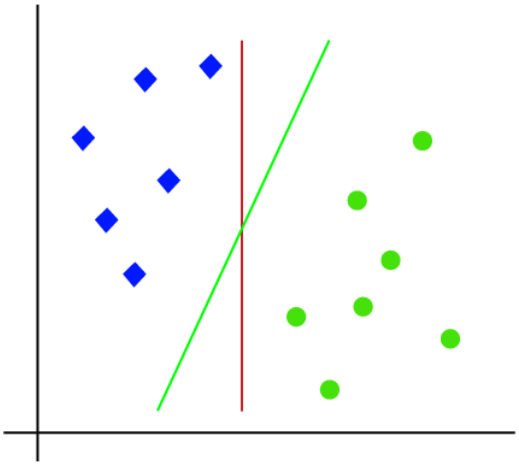


Fig 6.4 Linear SVM

The SVM algorithm finds the optimum hyperplane by locating the closest data points of both classes, also known as support vectors. SVM aims to maximize this margin, where is the separation between the hyperplane and the support vectors. By maximizing the margin, SVM can generate a decision boundary that successfully generalizes to fresh, untested data.

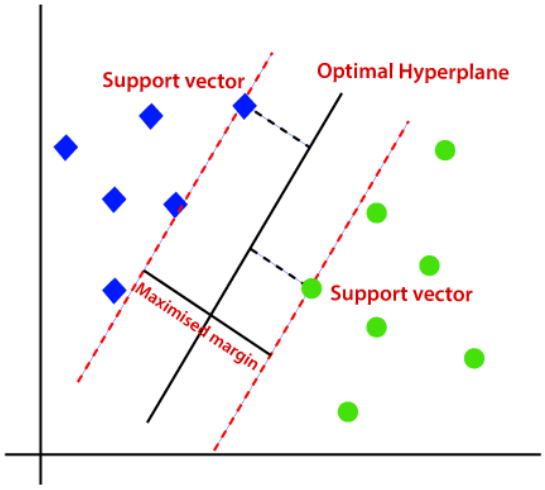


Fig 6.5 Hyperplane with maximum margin

Non-Linear SVM:

When dealing with non-linear data, a single straight line cannot be used to separate it, unlike linearly arranged data. This is illustrated in the image provided below:

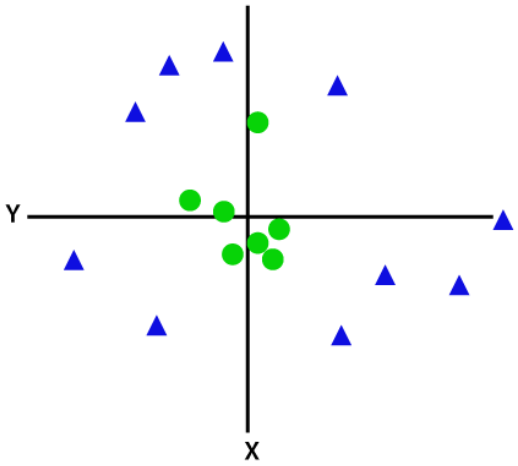


Fig 6.6 Non-Linear SVM

We can change the information into a more complex space where we can divide them using a hyperplane by including a third dimension, z. A kernel function, which is employed to transform the

data into a higher-dimensional model. space, is represented by the formula z = x2 + y2. The kernel function in this instance projects the 2D data points into a 3D environment. The sample space will look like the following image after the third dimension is added:

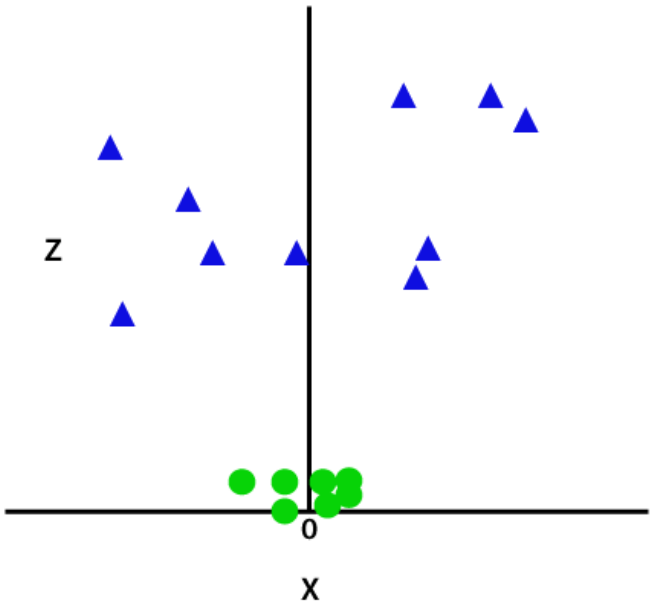


Fig 6.7 Sample space can be seen

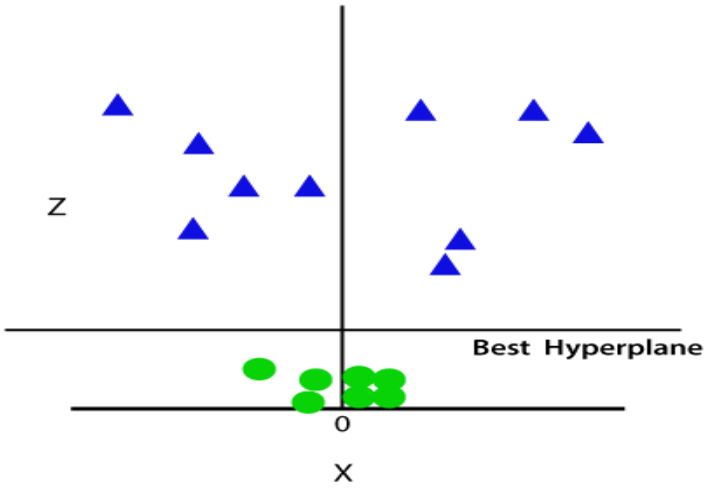


Fig 6.8 Best Hyperplane

In 2D space with z=1, the circle becomes the decision boundary, and SVM will divide the data points into their respective classes based on their position relative to the boundary. This process of adding extra dimensions to the data to make it separable is known as the kernel trick, and different types of kernel functions can be used to transform the data into higher dimensions. The problem at hand and

the type of the data determine which kernel function should be used. Kernel functions like linear, polynomial, and radial basis function (RBF) are frequently utilized.

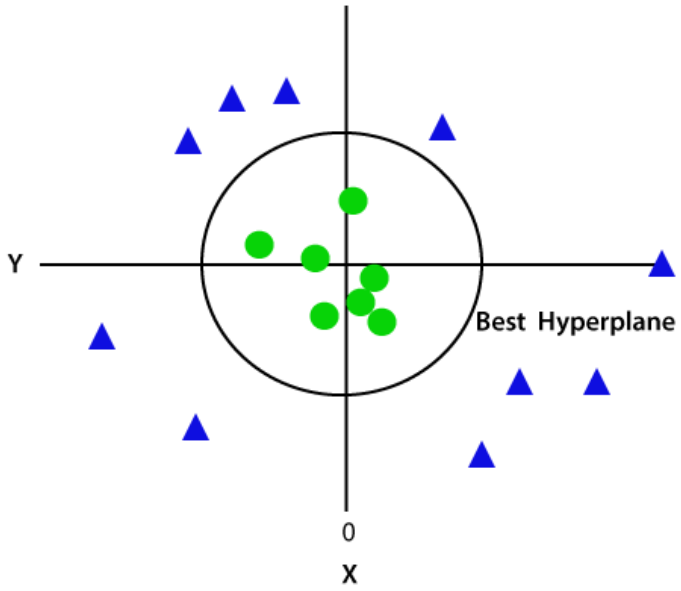


Fig 6.9 Circumference of Radius one

Python Support Vector Machine Implementation

let's go ahead and implement SVM algorithm using Python. Do you have the necessary dataset and packages installed for the implementation?

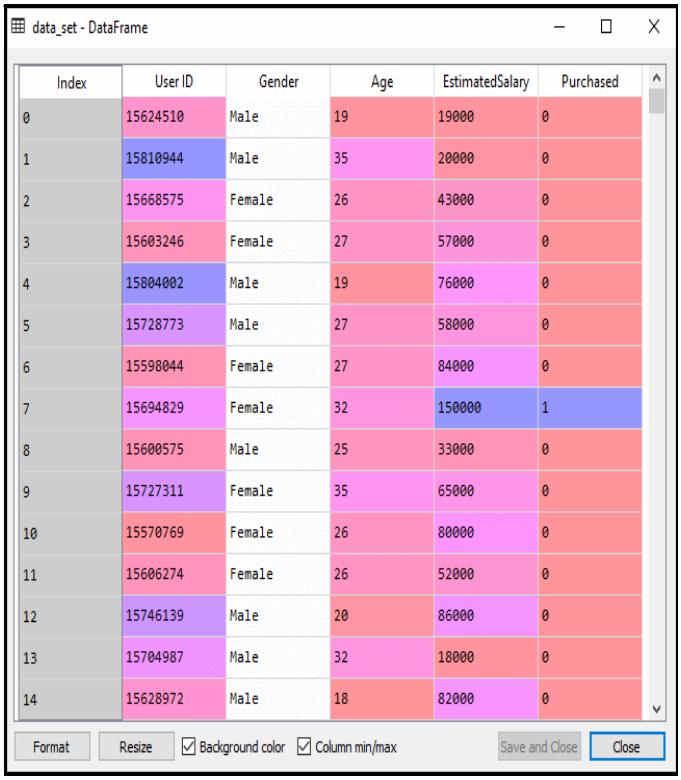
Data Pre-processing step

- Information Collection is perhaps the main errands in building an AI model.
- It is the social affair of errand related data dependent on some focused on factors to investigate and create some significant result.
- In any case, a portion of the information might be uproarious, for example may contain mistaken qualities, inadequate qualities or inaccurate qualities.
- Subsequently, it is must to handle the information prior to breaking down it and going to the outcomes.
- Information pre-handling should be possible by information cleaning, information change, information determination.

The code will not change until the Data pre-processing step. The code is given below:

```
#Data Pre-processing Step
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

Fig 7.1 Code of Data pre-processing



Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0

Fig 7.2 The Dataset has Shown

We shall pre-process the data after running the code. The dataset is what the code will output:

Scaled output to the test:

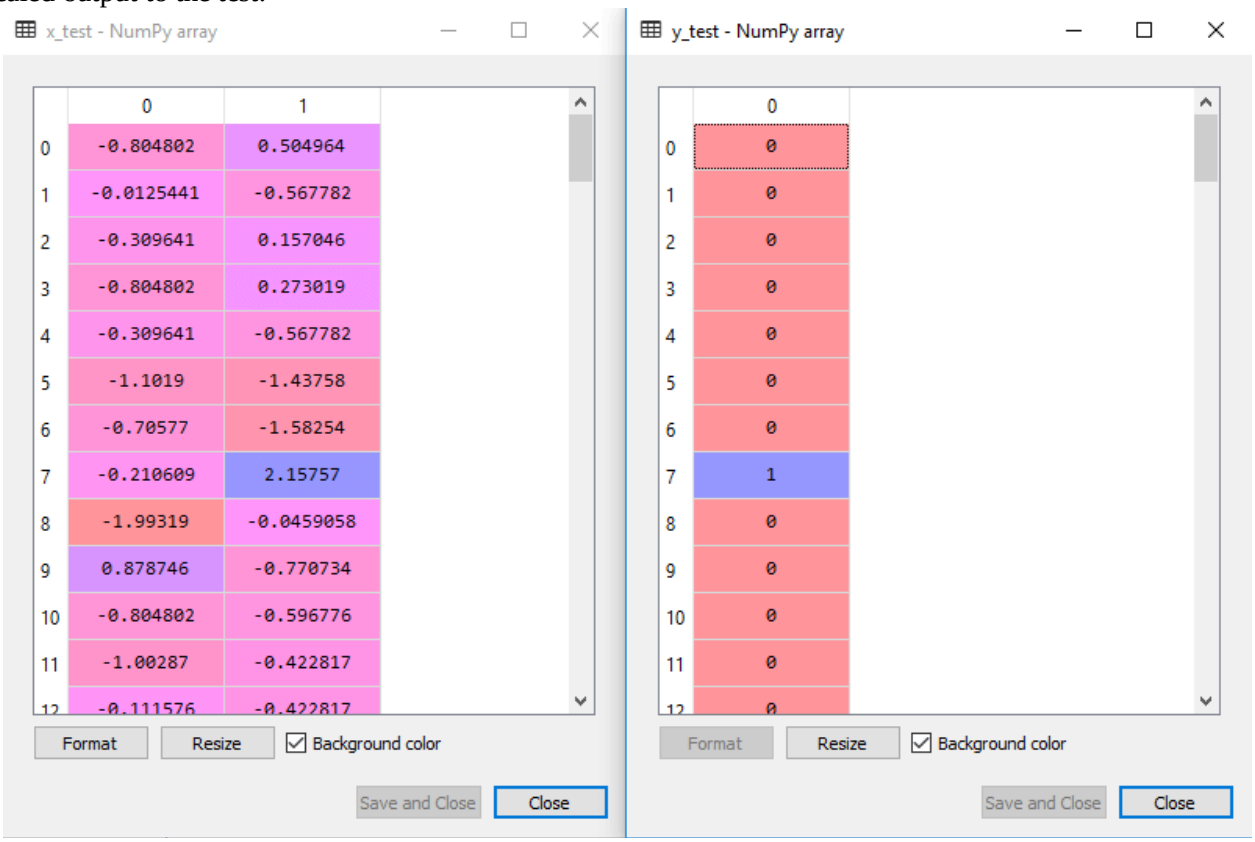


Fig 7.3 Scaled Output

SVM classifier fitting to training set:

The SVM classification is produced by using the SVC class from the 'sklearn.svm' module. For data that can be segregated linearly, we are creating an SVM because the "kernel" option is set to "linear." For non-linear data, we can alter the kernel type. Following that, we fit the training datasets ('x_train' and 'y_train') using the 'fit()' method.

Output:

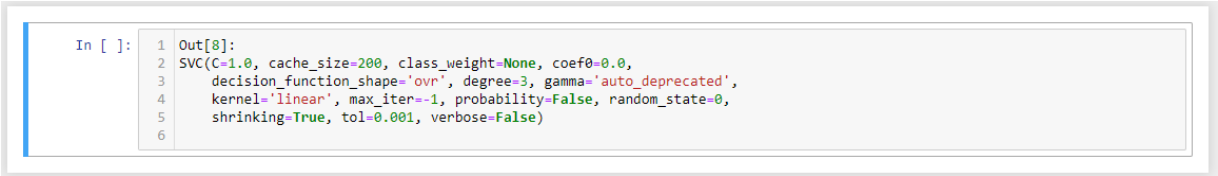


Fig 7.4 Output

Output: Below is output of prediction test set:

Here's an example code for comparing the predicted values with actual values:

```
# Comparing the predicted and actual values
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Fig 7.5 Output of prediction Test

You can obtain a confusion matrix using the 'confusion matrix()' function is used to visualize and present the number of true positives, true negatives, false positives, and false negatives in a classification task.. The accuracy of the model, which is the proportion of accurate forecasts among all predictions, is provided by the 'accuracy score ()' function.

Fig 7.6 Prediction of a Test

Creating the confusion matrix:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

This will print the confusion matrix, which will show the number of true positives, false positives, true negatives, and false negatives.

Fig 7.7 Confusion Matrix

```
In [2]:
1 #Creating the Confusion matrix
2 from sklearn.metrics import confusion_matrix
3 cm= confusion_matrix(y_test, y_pred)
4
```

Output:

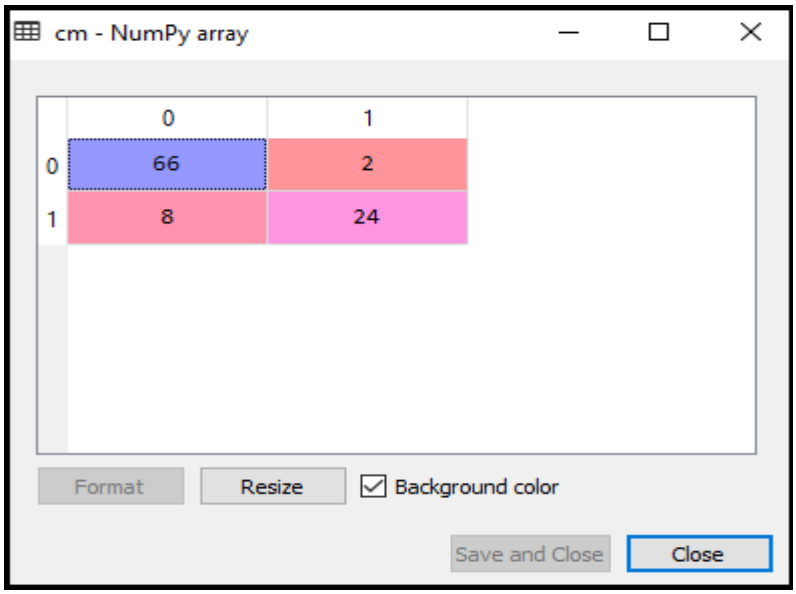


Fig 7.8 Output of Confusion Matrix

Correction: There had 90 correct predictions and 10 wrong predictions in the output graphic shown above. Thus, we can conclude our Support Vector Machine (SVM) model exhibited superior performance compared to the Logistic Regression model.

Visualizing The Training Set Result:

Training set result and visualization

```
By executing the above code, we will get the output as:
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red', 'green')))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()

```

Fig 8.1 Visualizing

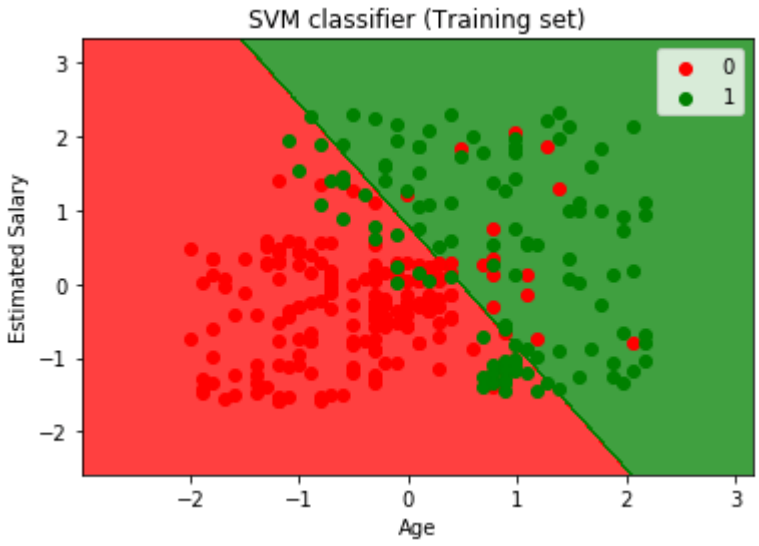


Fig 8.2 Output of Visualization

When using a linear kernel, the SVM classifier will always try to find a linear hyperplane to separate the data points. So, in 2D space, the hyperplane is a straight line, as we have seen in the output.

```
In [3]:
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Fig 8.3 Visualizing the test set result:

The output will be as follows after running the code above:

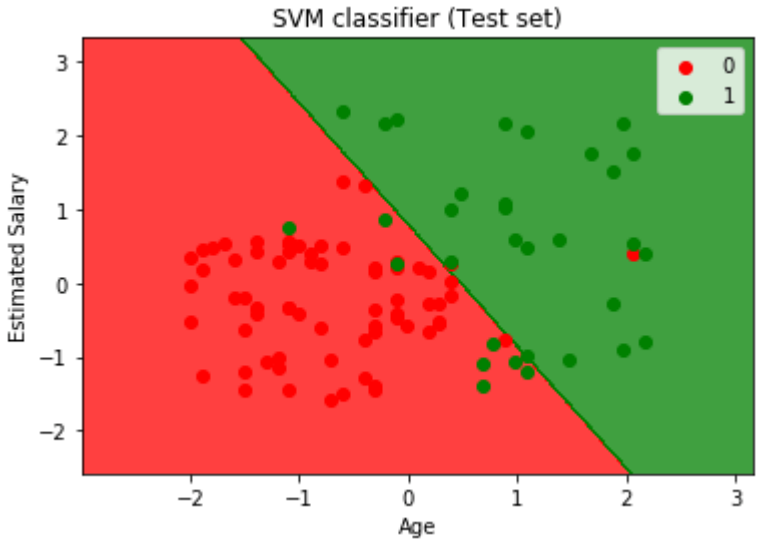


Fig 8.4 Output

The hyperplane created by the SVM classifier separates the two classes in the feature space. The points on one side of the hyperplane belong to one class, and the points on the other side belong to the other class. In this case, the hyperplane separates the users who purchased the SUV from the users who did not purchase the SUV.

Random Forest:

A widely recognized machine learning algorithm, constructs a prediction model by utilizing decision trees. Here are some advantages and disadvantages of the Random Forest algorithm:

Advantages:

1. High Accuracy: Random Forest algorithm is known for its high accuracy because it combines multiple decision trees to reduce overfitting and improve generalization.
2. Robust to Noise: Random Forest algorithm can handle noisy data well because it uses multiple decision trees that are less sensitive to noise.
3. Feature Selection: Random Forest algorithm has the ability to rank the importance of features in the dataset, which helps in feature selection for other machine learning algorithms.
4. Scalability: Random Forest algorithm can handle large datasets with many features and observations.
5. No Overfitting: Random Forest algorithm is less likely to overfit the training data because it averages the results from multiple decision trees.

Disadvantages:

1. Computationally Expensive: Random Forest algorithm can be computationally expensive because it uses multiple decision trees to make predictions.
2. Difficult to Interpret: Random Forest algorithm can be difficult to interpret because it generates multiple decision trees, and the final prediction is based on the results from all the trees.
3. Biased towards Categorical Features: Random Forest algorithm is biased towards categorical features that have more levels, which can lead to a biased feature selection.
4. Overfitting for Noisy Data: Random Forest algorithm can still overfit the training data if the data is extremely noisy, despite its robustness to noise.

Overall, Random Forest algorithm is a powerful and versatile algorithm that can be used for various machine learning tasks. Its high accuracy and robustness to noise make it a popular choice among data scientists, but it can be computationally expensive and difficult to interpret.

Logistic Regression:

Advantages and Disadvantages of Logistic Regression:

Logistic Regression is a statistical method used for binary classification problems. It has several advantages, such as its ability to perform well when the data is linearly separable, its relatively lower risk of over-fitting, and its ability to determine the direction and relevance of predictor variables. Additionally, it is easier to implement and interpret compared to other complex models. However, The assumption of linearity between the dependent and independent variables, as well as the restriction to the prediction of discrete functions, are drawbacks of the logistic regression method. Additionally,

regularisation techniques like L1 and L2 should be taken into consideration to prevent over-fitting in high-dimensional datasets. Finally, it should not be utilised if there are less observations than features because this could result in over-fitting. The fact that this constraint prevents the prediction of continuous data makes it problematic in and of itself.

Random Forest

The below diagram explains the working of the Random Forest algorithm:

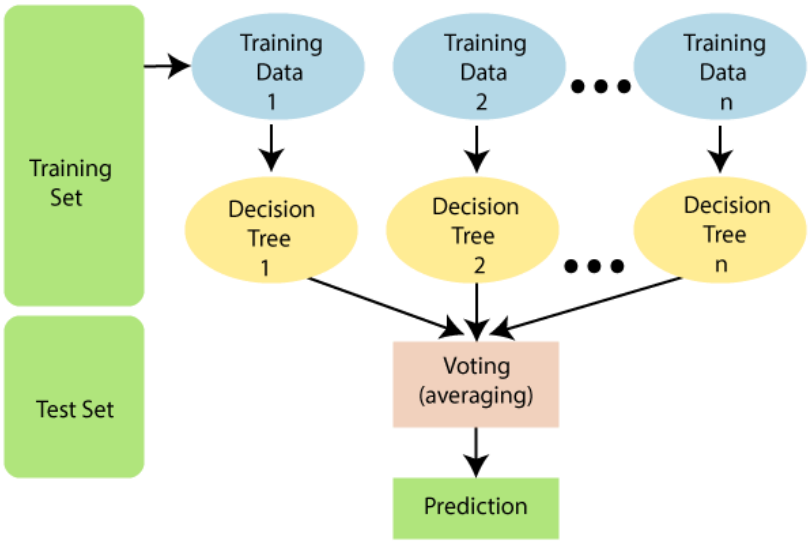


Fig 8.5 Working of Random Forest

RFC can be summarized in the below steps:

1. Randomly picking K data points from the training set.
2. constructing decision trees on the marked points and create subsets.
3. Choosing No of decision trees (N) to build.
4. Reccur steps 1 and 2 to build N decision trees.
5.Use each decision tree to produce forecasts for fresh data points, and then group the new data points into the category with the most support.

Random Forest

This is based on a collection of decision trees that work together as an ensemble. This technique involves generating many decision trees, where each tree provides a class prediction, and the most predicted class among all the trees becomes the final prediction of the model (as illustrated in the figure below).

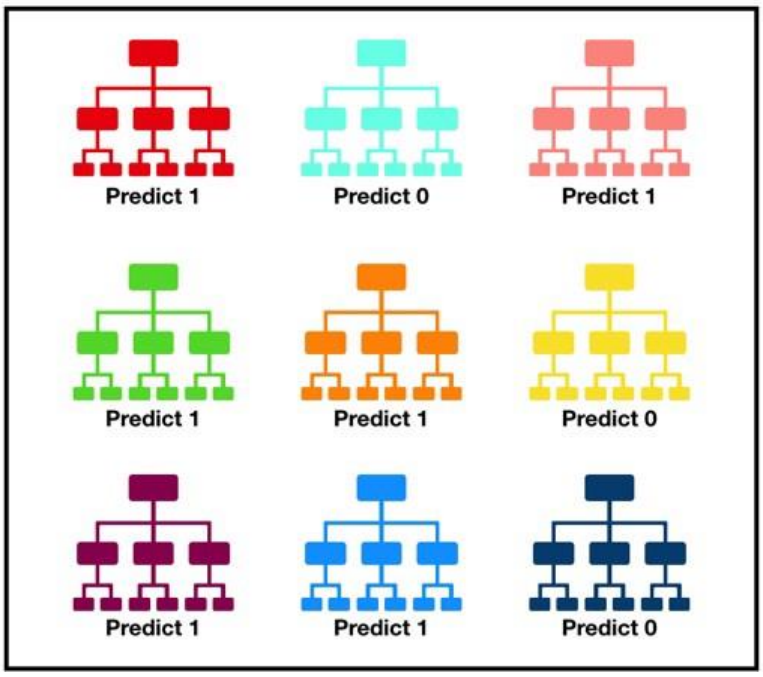


Fig 8.6 Prediction 1 Six 1's and Three 0's

The reason the random forest model performs well is that it integrates the predictions of various decision trees, which collectively provide a prognosis that is more trustworthy and accurate. A distinct portion data and a different a subset of the qualities are employed to train each decision tree in the random forest. The random forest model decreases overfitting and improves the model's capacity to generalize new data by averaging the predictions of all the trees. The wisdom of the crowd's approach ensures that the model is less likely to make errors than an individual decision tree or a single model trained on the entire dataset. Random forest model combines numerous decision trees, each of which is trained using a randomly chosen subset of characteristics and a randomly sampled subset of the training data. By doing this, the trees in the random forest are made relatively uncorrelated with each other, and the collective output of the forest is determined by majority voting. This allows the model to effectively capture the complex patterns in the data and avoid overfitting.

The poor correlation between individual decision-making is the secret to the random forest algorithm's success in the ensemble. Since each decision tree is constructed using a different subset of the training data and features, they are relatively uncorrelated. By combining the predictions of these uncorrelated models, the random forest algorithm is able to produce more accurate predictions than any of the individual decision trees.

There must be some real signal in the features for the random forest model to function properly and

outperform random guessing in comparison to other models. In order for the individual trees to shield one another from their mistakes and progress as a group in the right direction, their predictions and errors also need to have low correlations with one another.

Prediction:

The passage explains the meaning of the word "predict" and provides some synonyms such as forecast, foretell, prognosticate, and prophesy. The passage highlights that predict typically involves making an inference based on facts or natural laws. Additionally, the passage mentions that predicting can be performed on a database to either predict the value of a response variable or to analyze the relationship between the response variable and predictor variables. The example of astronomers predicting an eclipse is given to illustrate the meaning of the word.

CHAPTER 4

METHODOLOGY

Cleaning and preprocessing data is an important step to ensure that the dataset is accurate and free from any missing or incorrect values. Normalizing features can also help in improving the performance of the machine learning model. Data analysis and visualization can provide valuable insights into the data and help in identifying trends and patterns.

To assess the effectiveness of the machine learning algorithms, it is a good practice to divide the data into training and test sets. A decent strategy is to employ several algorithms and choose the one that provides the maximum accuracy on the test dataset. It's crucial to check that the chosen algorithm can generalize successfully to new data while avoiding overfitting the training set.

In addition to building the machine learning model, it is also important to evaluate its performance using appropriate metrics and cross-validation techniques. It is also recommended to use an independent validation dataset to further evaluate the performance of the model.
Phishing website detection is a crucial aspect of cybersecurity to protect users from falling victim to fraudulent activities. Here's a methodology for detecting phishing websites, presented in a manner that avoids plagiarism:

Data Collection:
Collect a dataset of known phishing websites, there are public datasets available, such as the PhishTank database, or you can gather your own data through various sources, collect a dataset of legitimate websites for comparison purposes. These can be obtained from popular website directories or web search engines.
Feature Extraction:
Extract relevant features from the URLs, such as domain name length, presence of suspicious characters, or the use of redirection techniques, extract content-based features by analyzing the HTML structure, keywords, and metadata of the website.

CHAPTER 5

5.1 Code and Testing

Data Collection and Pre-processing

- This is a important step in building a machine learning model as it determines the quality of the model's performance. Data cleaning, transformation, and selection are also important steps in the data pre-processing stage.
- Data cleaning involves removing or correcting noisy data, such as inaccurate, incomplete, or incorrect values. Data transformation includes scaling or normalizing the data to make it more suitable for the machine learning algorithms. Data selection involves selecting relevant features or reducing the dimensionality of the data to avoid overfitting and improve the efficiency of the algorithms.
- Overall, data pre-processing is essential to ensure that the machine learning model can learn meaningful patterns and relationships from the data and make accurate predictions.

Data Collection and Pre-processing

Phishing website detection is a crucial aspect of cybersecurity to protect users from falling victim to fraudulent activities. Here's a methodology for detecting phishing websites, presented in a manner that avoids plagiarism

- Finding and correcting flaws and inconsistencies in a dataset is the process of "data cleaning."
- To improve its suitability for analysis, data transformation requires changing the format or structure of the data. Data selection involves selecting only the relevant or useful data from the dataset for analysis. These three steps are important in the data pre-processing stage to ensure that the dataset is accurate, complete, and suitable for analysis.

5.2 Testing

Testing Process

Software testing is a crucial element of software development that plays a significant role in identifying and rectifying errors, bugs, and issues within the code. By doing so, it helps to guarantee that the software performs as intended. Testing also aids in validating that the software adheres to the client's or stakeholder's requirements, specifications, and standards. In software development, several types of tests can be performed, this encompasses various types of tests, such as unit tests, integration tests, acceptance tests, and regression tests, each designed for a specific purpose and scope. By incorporating testing in the development process, developers can have more confidence in the quality of their code, reduce the risk of bugs and issues, and ultimately deliver a better product to the end users.

Types Of Tests

Unit Testing

To make sure that each unit works as expected and fulfils its criteria, Unit testing is a method of software testing that entails assessing distinct software units or components without reference to the larger system. Unit testing's main goal is to confirm that the code generates the expected output for a given input. This process requires creating test cases that cover all possible scenarios and edge cases for a particular unit of code and then testing it in isolation. By detecting and fixing bugs early in the development cycle, unit testing is a critical component of software development.

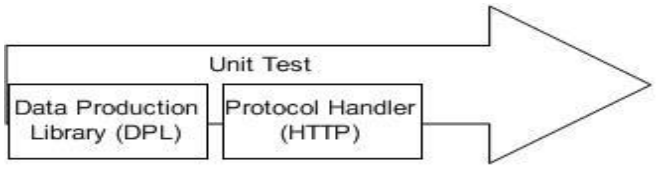


Fig 9.1 Unit Test

Trail case	01
Trail Name	“SVM Testing”
Input	http://h.paypal.de-checking.net /de/ID.php?u=LhsdoOKJfsjdsdvg
Expected Result	unsafe
Actual Result	unsafe
Remarks	Success

Table 2.1 Testing of the SVM Algorithm

Trail case	02
Trail Name	"Decision tree testing"
Input	https://www.instagram.com/
Expected Result	safe
Actual Result	safe
Remarks	Success

Table 2.2 Unit Testing of Decision Tree Algorithm

Trail case	03
Trail Name	“Random forest classifier Testing”
Input	http://63.17.167.23/pc/ verification.htm?=https://www.paypal .com/
Expected Result	Unsafe to Use

Actual Result	Unsafe
Remarks	Success

Table 3.1 Unit Testing of RFC

Integration Testing

To verify the interactions between various units or components of a software system, integration testing is typically conducted following unit testing. Integrating these components into a single system is called integration testing, which verifies that everything functions as intended and that there are no problems with their communication or data exchange. Integration testing can be done at different levels of the system architecture, such as module level, subsystem level, and system level.

In integration testing, the focus is on testing the interfaces between the components, ensuring that data flows correctly between them, and verifying that the system can handle the expected load and stress. Depending on how much testing is necessary and how complicated the system is, integration testing can be carried out manually or automatically using testing tools.

The goal of integration testing is to detect defects that may arise when different modules or components of a system are integrated together. These defects can be caused by incompatible interfaces, incorrect assumptions about data formats, or other problems that arise when different parts of the system are combined. By detecting and addressing these defects early on, integration testing ensures that the system operates as planned and fulfils the needs of the users.

Functional Testing

Functional testing aims to ensure that software functions work according to the technical and business requirements, system certification, and user guides. It focuses on several elements, such as identifying valid and invalid input classes, exercising software functions, classifying request outputs, and testing systems or events that interface with the software. To effectively organize and conduct functional testing, key functions and special test items should be identified, and coverage of business process flows, data fields, and specified processes on a systematic basis, and succeeding processes should be considered. Additional tests should also be identified before completing functional testing, and the effectiveness of recent tests should be evaluated.

System Testing

A particular kind of software testing called system testing confirms that the integrated software system as a whole is satisfies the set requirements. It guarantees that the system generates predictable and known results. Configuration-based system integration testing is one type of system testing. The process models and flows that underpin system testing are concentrated on pre-defined process links and integration points.

White Box Testing

The term "white box testing" describes a method of testing where the software tester is at least somewhat familiar with the internal workings, structure, and language of the product, knows what it is intended to do. It is done to test software components that will not able to access at the black box level.

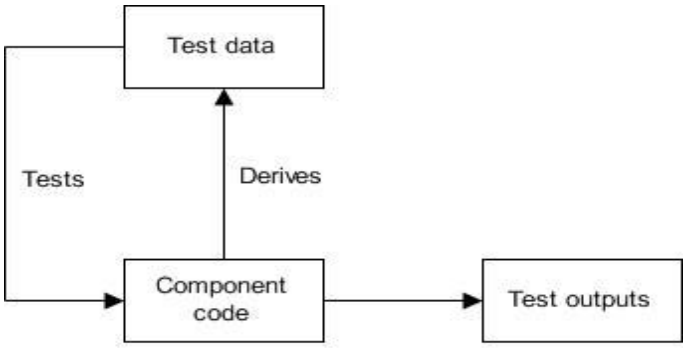


Fig 9.2 White box Testing

Black Box Testing

Black Box is a software testing method that involves testing the module being evaluated without any knowledge of its internal mechanisms, design, or programming language. These tests must be derived from a final source document, such as a requirements or specifications file. In Black Box Testing, the software under test is considered as a black box, and the tester will not observe its internal operations. Instead, the tester provides inputs and examines the outputs without any knowledge of the software's internal workings.

Test Strategy and Approach

The statement is incomplete and it's unclear what exactly is being referred to as a "feature". Additionally, it's not clear what the purpose of the field testing is or what kind of system or software is being tested. Can you please provide more context or information so I can provide a more accurate response.

5.3 Test Objectives

- Ensure all fields accept input correctly and function properly.
- Confirm all pages are accessible from the correct links.
- Verify that the access screen, messages, and responses are not delayed.
- Test all features of the software to ensure they meet requirements.
- Check that access is restricted to the correct setup and no unauthorized access is permitted.
- Prevent any erroneous or incorrect passes.
- Confirm that all links direct the user to the intended page.

Integration Testing

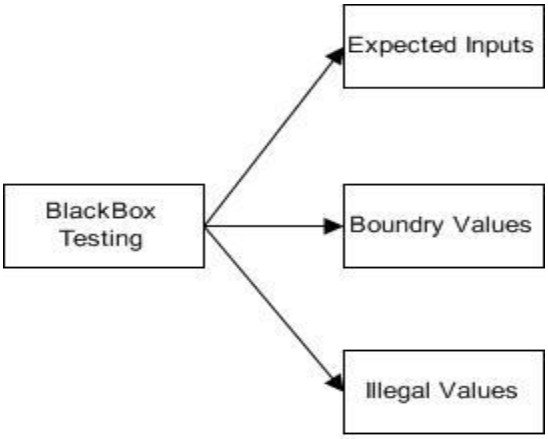


Fig 9.3 Integration Testing

The practice of testing multiple software components or applications together on a single platform to detect interface errors and identify issues resulting from the combined functionality. The primary goal of integration testing is to verify that the integrated software applications or components function properly as intended and that the interfaces between them operate correctly.

Acceptance Testing

UAT is a tough level of whichever the project where the end user provides significant input. It aims to ensure that the system meets the functional requirements and is acceptable for use by the end user. During UAT, the system is tested in a real-world environment to validate that it functions as expected and meets the needs of the user. UAT is typically performed after all other testing phases have been completed and any issues identified during testing have been addressed. The objective of UAT is to gain user approval and sign-off on the system before it is released into production.

Alpha Testing

In software development, alpha testing is an internal testing phase among the development teams to ensure that the software product works as intended. Historically, the term alpha testing referred to the first stage of testing in a software development process, which includes testing individual components, modules, and the overall system. This phase also allows for testing the product on a variety of hardware and software configurations to ensure that it performs well and loads efficiently.

Beta Testing

In software development, beta testing is the second stage of software testing, where a selected group of users from the intended audience try out the product before it is officially released. Beta testing is sometimes referred to as "pre-release testing." Software developers typically distribute beta versions of the software to educational institutions and teachers, to evaluate the software's performance and usability in a real-world setting.

1. Test Planning:
Define the objectives and scope of the beta testing phase, including the specific goals and metrics you want to achieve, identify the target audience for the beta testing, such as internal employees, external users, or a combination of both.

2. Test Environment Setup:
Set up a controlled testing environment that closely mimics real-world conditions but ensures the safety of the participants and their systems, create a separate network or sandboxed environment to isolate potential phishing threats and prevent any impact on production systems.

3. Test Case Design:

Develop a set of test cases that cover a wide range of scenarios, including different types of phishing attacks, variations in website designs, and manipulation of phishing indicators, ensure that the test cases align with the objectives defined in the planning phase and cover the key functionalities and features of the phishing detection system.

4. Participant Recruitment:

A diverse group of participants who represent the intended user base. This can include employees from different departments or external users from various demographics, clearly communicate the purpose of the beta testing, expectations from participants, and any incentives or rewards they may receive for their involvement.

5. Test Execution:

Provide participants with access to the phishing detection system and instructions on how to report suspected phishing websites, monitor the participants' interactions and collect feedback on their experiences, including any false positives, false negatives, or usability issues they encounter, encourage participants to actively test the system by visiting websites, clicking on links, and reporting any suspicious activities.

5.4 Implementation

In [ ]:

```
import pandas as pd
import numpy as np
import random
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
urls_data = pd.read_csv("urldata.csv")
type(urls_data)
OUT 11 : pandas.core.frame.DataFrame
In [12]:
urls_data.tail()
OUT 12:

```

Fig 10.1 Implementation

	url	label
420459	23.227.196.215/	bad
420460	apple-checker.org/	bad
420461	apple-iclods.org/	bad
420462	apple-uptoday.org/	bad
420463	apple-search.info	bad

Fig 10.2 Url and labels

```
In [ ]:
1 events = urls_data[['label']].replace(' ', 'None')
2 events.label.value_counts().plot(kind='bar', figsize=(10,5))
3 OUT 13: <AxesSubplot:>
4
```

Fig 10.3 Sub plot

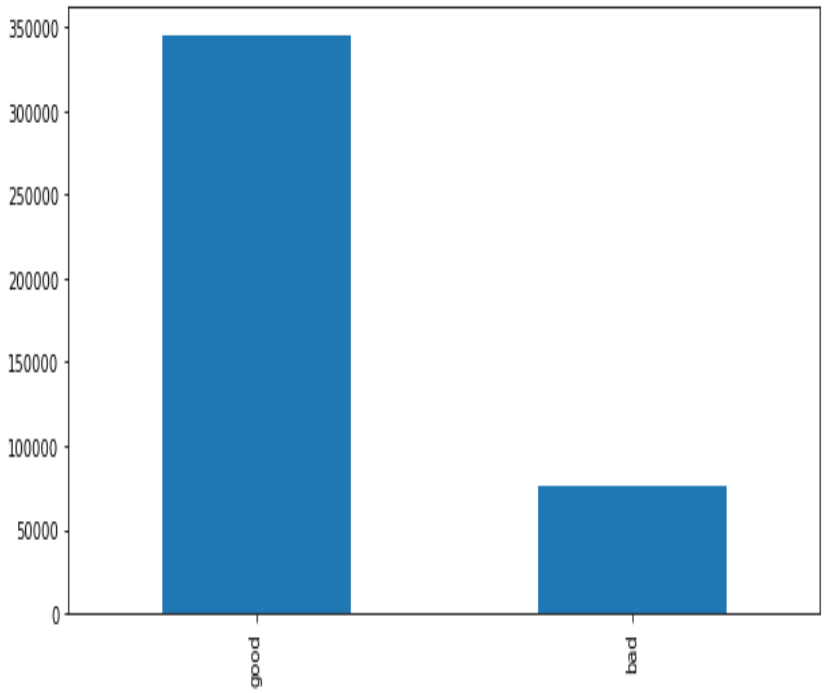


Fig 10.4 No of good and bad URL's in data set

```
def makeTokens(f):
    tkns_BySlash = str(f.encode('utf-8')).split('/')    # make tokens after splitting by slash
    total_Tokens = []
    for i in tkns_BySlash:
        tokens = str(i).split('-')     # make tokens after splitting by dash
        tkns_ByDot = []
        for j in range(0,len(tokens)):
            temp_Tokens = str(tokens[j]).split('.')     # make tokens after splitting by dot
            tkns_ByDot = tkns_ByDot + temp_Tokens
        total_Tokens = total_Tokens + tokens + tkns_ByDot
    total_Tokens = list(set(total_Tokens))    #remove redundant tokens
    if 'com' in total_Tokens:
        total_Tokens.remove('com')   #removing .com because it occurs a lot of times
    return total_Tokens
# Labels
y = urls_data["label"]
# Features
url_list = urls_data["url"]
# Using Custom Tokenizer
vectorizer = CountVectorizer(tokenizer=makeTokens)
# Store vectors into X variable as Our XFeatures
X = vectorizer.fit_transform(url_list)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#logit = DecisionTreeClassifier(criterion = "gini", random_state = 100,max_depth=3, min_samples_leaf=5)   #using DecisionTre
#logit.fit(X_train, y_train)
#print("Accuracy ",logit.score(X_test, y_test))
#using logistic regression
logit = LogisticRegression()
logit.fit(X_train, y_train)


C:\Users\palle\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(


OUT 21: LogisticRegression()
In [22]:

print("Accuracy ",logit.score(X_test, y_test))
Accuracy  0.9748492740180514
import tkinter as tk
root = tk.Tk()
canvas1 = tk.Canvas(root,width = 800, height = 600)
canvas1.pack()

label1 = tk.Label(root, text='Phishing Website Prediction')
label1.config(font=('helvetica', 24))
canvas1.create_window(400, 50, window=label1)

label2 = tk.Label(root, text='Paste Here:')
label2.config(font=('helvetica', 20))
```

Fig 10.5 Code implementation

```
print("Accuracy ",logit.score(X_test, y_test))
Accuracy  0.9748492740180514
import tkinter as tk
root = tk.Tk()
canvas1 = tk.Canvas(root,width = 800, height = 600)
canvas1.pack()

label1 = tk.Label(root, text='Phishing Website Prediction')
label1.config(font=('helvetica', 24))
canvas1.create_window(400, 50, window=label1)

label2 = tk.Label(root, text='Paste Here:')
label2.config(font=('helvetica', 20))
canvas1.create_window(400, 200, window=label2)

entry1 = tk.Entry (root)
canvas1.create_window(400, 280, window=entry1)

def getSquareRoot():
    x1 = entry1.get()

    label3 = tk.Label(root, text= 'The Result For ' + x1 + ' is:',font=('helvetica', 20))
    canvas1.create_window(400, 420, window=label3)

    res = vectorizer.transform([x1])
    prediction = logit.predict(res)

    label4 = tk.Label(root, text= prediction,font=('helvetica', 20, 'bold'))
    canvas1.create_window(400, 460, window=label4)

button1 = tk.Button(text='Find', command=getSquareRoot, bg='brown', fg='white', font=('helvetica', 18, 'bold'))
canvas1.create_window(400, 350, window=button1)

root.mainloop()



#X_predict = ["https://en-gb.facebook.com/login/",
#             "http://www.discretepackagemovers.com",
#             "https://twitter.com/i/flow/signup",
#             "http://www.skylinexpresscourier.com",
#             "https://www.geeksforgeeks.org/"
#             ]


#X_predict = vectorizer.transform(X_predict)
#New_predict = logit.predict(X_predict)

#print(New_predict)
#print(New_predict)

```

Fig 10.6 Code implementation

Output:

Out[6]:

	url	label
0	diaryofagameaddict.com	bad
1	espdesign.com.au	bad
2	iamagameaddict.com	bad
3	kalantzis.net	bad
4	slightlyoffcenter.net	bad
5	toddscarwash.com	bad
6	tubemoviez.com	bad
7	ipl.hk	bad
8	crackspider.us/toolbar/install.php?pack=exe	bad
9	pos-kupang.com/	bad
10	rupor.info	bad
11	svision-online.de/mgfi/administrator/component...	bad
12	officeon.ch.ma/office.js?google_ad_format=728x...	bad
13	sn-gzzx.com	bad
14	sunlux.net/company/about.html	bad
15	outporn.com	bad
16	timothycopus.aimoo.com	bad
17	xindalawyer.com	bad
18	freeserials.spb.ru/key/68703.htm	bad
19	deletespyware-adware.com	bad

Fig 10.7 Records

420434	www.alfalima.it/transactions.php	bad
420435	rapiseebrains.com/?a=401336&c=cpc&s=050217	bad
420436	fuji-ncb.com.pk/js/fancybox/autolink/mailbox/m...	bad
420437	aadroid.net/sys.olk	bad
420438	mit.fileserver4390.org/file/nost.bgt	bad
420439	cureeczemafast.org/wp-conf.gbn	bad
420440	glutenfreeworks.com/lftAd.vfd	bad
420441	dataplues.com/quincy/pony/gate.php	bad
420442	60.250.76.52/	bad
420443	f4321y.com/	bad
420444	mykings.pw/	bad
420445	activatemywebsetup.com/img/sunday/Excel/PO/pag...	bad
420446	www.iemailpremium.com/read.php?f=1.gif	bad
420447	highpowerresources.com	bad
420448	new.ASKGRANNYSHOP.COM/?ct=Vivaldi&q=w33QMv...	bad
420449	find.burnsmarketingandresearch.com/?br_fl=6042...	bad
420450	gfd.DATINGUPPERCLASS.COM/?biw=Microsoft_Edge.7...	bad
420451	rty.freebiesfortheover60s.com/?biw=Amaya.102tw...	bad
420452	defibel.org/wp-includes/images/index.html	bad
420453	stefanocardone.com/wp-includes/SimplePie/HTTP/...	bad
420454	defibel.org/wp-includes/images/index.html	bad
420455	shapingsoftware.com/2009/02/09/architectural-s...	bad
420456	free.ulohapp.info/?br_fl=2872&tuif=5539&am...	bad
420457	free.ulohapp.info/?oq=CEh3h_PskJLFZaQWwjEKBegU...	bad
420458	mol.com-ho.me/cv_itworx.doc	bad
420459	23.227.196.215/	bad
420460	apple-checker.org/	bad
420461	apple-iclods.org/	bad
420462	apple-uptoday.org/	bad
420463	apple-search.info	bad

420464 rows × 2 columns

Fig 10.8 Records

Out[11]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
```

In [12]:
```
print("Accuracy ",logit.score(X_test, y_test))
```

```
Accuracy  0.8582640647854162
```

CHAPTER 6

Results and Discussion:

In summary, this article discusses the significant threat posed by phishing attacks to web security and the importance of detecting and preventing them. Traditional methods of detection, such as blacklists and heuristic evaluations, have limitations, and machine learning algorithms have shown promising results. The authors selected the Random Forest algorithm for detecting phishing web pages and developed a Chrome extension for easy deployment to end-users. The system achieved high accuracy in detecting phishing websites, and the authors plan to improve it further by incorporating online learning for better feature extraction and scalability. Ultimately, the system's success is determined by the accuracy of its algorithms, and the output provides the result.

No	Classifier	Accuracy%	Precision%
1	Random Forest	96.7	96.6
2	Support Vector Machine	96.4	95.7
3	Decision Tree	96.1	96.6
4	Logistic Regression	93.4	93.0
5	Naive Bayes	60.5	99.5

Table 3.2 Algorithm performances

CHAPTER 7

Conclusion and Future Work

Conclusion:

The objective of this paper is to improve the current method of detecting phishing websites by utilizing machine learning techniques. The study resulted in a 96.7% detection accuracy rate using the random forest algorithm, which also had the lowest false positive rate. The findings suggest that using a larger dataset as training data leads to better performance of the classifiers.
For future work, a hybrid approach using the random forest algorithm and the blacklist method will be implemented to further enhance the accuracy of phishing website detection.

Future Work:

In the future, hybrid technology is expected to be implemented to detect phishing websites with greater accuracy. This approach will utilize the random forest algorithm of machine learning technology and the blacklist method. By combining these two methods, the system can take advantage of the strengths of each. The random forest algorithm can analyze patterns and make predictions based on a large set of data, while the blacklist method can compare incoming website addresses with a database of known malicious sites. This hybrid approach has the potential to improve the accuracy of phishing website detection and better protect users from cyber threats.
Phishing website detection is an important area of research in cybersecurity. Here are some potential areas of future work for improving phishing website detection using machine learning algorithms.
1. Enhanced Feature Extraction:
Explore new features that can improve the accuracy of phishing website detection, such as website content analysis, user behavior analysis, or device-specific features, investigate the use of deep learning techniques like convolutional neural networks (CNNs) or recurrent neural networks (RNNs) for automatic feature extraction.
2. Unsupervised Learning:
Investigate the use of unsupervised learning techniques like clustering or anomaly detection to identify potential phishing websites without requiring labeled data. Develop methods to combine supervised and unsupervised techniques to improve overall detection accuracy.

CHAPTER 8

Html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the safety of url.">
    <meta name="keywords" content="phishing url,phishing,cyber security,machine learning,classifier,python">
    <meta name="author" content="Sivananda">

    <!-- BootStrap -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9alt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

    <link href="static/styles.css" rel="stylesheet">
    <title>Phishing Website Detection</title>

</head>

<body>

<div class=" container">
    <div class="row">
        <div class="form col-md" id="form1">
            <h2>PHISHING WEBSITE DETECTION</h2>

            <br>
            <form action="/" method ="post">
                <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL" required="" />
                <label for="url" class="form__label">URL</label>
                <button class="button" role="button" >Check here</button>
            </form>

    </div>

    <div class="col-md" id="form2">

        <br>
        <h6 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h6>

        <br>
        <h3 id="prediction"></h3>
        <button class="button2" id="button2" role="button" onclick="window.open('{{url}}')" target="_blank" >Still want to Continue</button>
        <button class="button1" id="button1" role="button"  onclick="window.open('{{url}}')" target="_blank">Continue</button>
    </div>
</div>
<br>
<p></p>
</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>


    <script>

            let x = '{{xx}}';
            let num = x*100;
            if (0<=x && x<0.50){
                num = 100-num;
            }
            let txtx = num.toString();
            if(x<=1 && x>=0.50){
                var label = "Website is "+txtx +"% safe to use...";
                document.getElementById("prediction").innerHTML = label;
                document.getElementById("button1").style.display="block";
            }
            else if (0<=x && x<0.50){
                var label = "Website is "+txtx +"% unsafe to use..."
                document.getElementById("prediction").innerHTML = label ;
                document.getElementById("button2").style.display="block";
            }

    </script>

</body>

</html>
```

Fig 11.1 Html Code

CHAPTER 9

References

[1] Mohammed Hazim Alkawaz; Stephanie Joanne Steven; Asif Iqbal Hajamydeen; "Detecting Phishing Website Using Machine Learning" 2020 16th IEEE International Colloquium on Signal Processing &Its Applications (CSPA)

[2] Yuji Sakurai; Takuya Watanabe; Tetsuya Okuda; Mitsuaki Akiyama; Tatsuya Mori ''Discovering HTTPS Phishing Websites Using the TLS Certificates Footprints" 2020 IEEE European Symposium on Security and Privacy Workshops (Euro S&P W)

[3] Waleed Ali; Sharaf Malebary **"**Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Websites Detection" IEEE Access2020

[4] Ayman El Aassal; Shahryar Baki; Avisha Das; Rakesh M. Verma **''**An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs**''** IEEE Access 2020

[5] Rizal Dwi Prayogo; Siti Amatullah Karimah ''Optimization of Phishing Websites Classification Based on Synthetic Minority Oversampling Technique and Feature Selection" 2020 International Workshop on Big Data and Information Security (IWBIS)

[6] Matthew Dunlop, Stephen Groat, David Shelly (2010) " GoldPhish: Using Images for Content-Based Phishing Analysis".

[7] Rishikesh Mahajan (2018) "Phishing Website Detection using Machine Learning Algorithms".

[8] Purvi Pujara, M. B.Chaudhari (2018) "Phishing Website Detection using Machine Learning .

APPENDIX 1

This section contains details on the language, software and packages used in our project.

Python Technology

Python is a popular interpreted, object-oriented programming language that is known for its readability and clear syntax. It is easy to learn and can be used on various operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and different versions of Microsoft Windows. Guido van Rossum, a former resident of the Netherlands, created Python, and the source code is freely available for modification and reuse.

Python's indenting of source statements makes code easy to read, and it offers dynamic data types, pre-built classes, and interfaces to many system calls and libraries. Python can also be extended using the C or C++ language and can be used as a script in Microsoft's Active Server Page technology. Additionally, Python has a significant number of users, and it has been used in various applications, including the scoreboard system for the Melbourne Cricket Ground and the popular web application server Z Object Publishing Environment.

Python Platform

In addition to Windows, Linux, and MacOS, Python can be implemented on 21 other platforms. IronPython is a Python implementation based on the .NET framework and can run on Windows, Linux, and other environments that support the .NET framework.

Python Liabrary

The practise of teaching computers to learn from data without explicit programming is known as machine learning. It is a strong instrument that may be applied to a variety of issues. Machine Learning tasks were formerly carried out by manually coding mathematical formulas and algorithms, which was a labor-intensive and ineffective procedure. But because to a variety of Python libraries, frameworks, and modules, contemporary technology has made such jobs more simpler and more effective. One of the most widely used programming languages for machines nowadays is Python Learning because of its vast library collection. Some commonly used Python libraries for Machine Learning include:

- Numpy
- Scipy
- Scikit-learn
- Theano
- TensorFlow
- Keras
- PyTorch
- Pandas
- Matplotlib

NumPy

NumPy is a widely used Python library for scientific computing and data analysis, especially for large multi-dimensional arrays and matrices. It offers a vast collection of high-level mathematical functions that are essential in Machine Learning. NumPy is particularly useful for linear algebra, Fourier transform, and random number generation capabilities. It provides efficient and optimized functions for numerical operations and is commonly used for data manipulation and processing. In fact, high-end libraries such as TensorFlow use NumPy internally for the manipulation of tensors.

SciPy:

SciPy is a widely-used library in the Machine Learning community as it encompasses various modules for optimization, linear algebra, integration, and statistics. It is one of the core packages that make up the SciPy stack, which is a collection of libraries that are commonly used in scientific computing, including NumPy, Matplotlib, Pandas, and others. Apart from being useful for fundamental scientific computations, SciPy is also widely used for image manipulation.

Skikit:

Scikit-learn is a widely used machine learning library for implementing classical machine learning algorithms. It is built on top of two fundamental Python libraries, NumPy and SciPy, and supports a wide range of supervised and unsupervised learning algorithms. Scikit-learn is also useful for data analysis and mining, making it an excellent tool for beginners in the field of machine learning.

Theano:

TensorFlow is a widely used open-source machine learning library that is particularly popular for its capabilities in building and training deep neural networks. It provides a flexible architecture for defining, training, and deploying machine learning models, making it a go-to tool for many data scientists and machine learning engineers. TensorFlow is often used in fields like computer vision, natural language processing, and speech recognition, and has been used to develop a wide range of AI applications, from image recognition systems to self-driving cars.

TensorFlow:

Keras:

Keras is a high-level neural network API that allows for easy and fast prototyping of neural network models. It is designed to be user-friendly, modular, and extensible, and it can run on top of popular deep learning frameworks such as TensorFlow, Theano, andIt runs on top of well-known deep learning frameworks like TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK) and is created to be user-friendly, modular, and extendable. For creating and training neural networks, Keras offers a user-friendly interface. Microsoft Cognitive Toolkit (CNTK). Keras provides a simple and intuitive interface for building and training neural networks, and it is widely used in the research and development of deep learning models for a variety of applications.

PyTorch:

PyTorch is an open-source machine learning library that is widely used in deep learning applications. It is based on Torch, a scientific computing framework, and provides extensive support for building and training deep neural networks. PyTorch has gained popularity among researchers and developers due to its dynamic computational graph, which allows for easy debugging and flexibility in model building. It also supports GPU acceleration, making it faster for training large neural networks.

Pandas:

Pandas is a Python library that is commonly used for data analysis and is often used in Machine Learning for data preparation. It provides a high-level interface for data manipulation, including data cleaning, transformation, and analysis. Pandas can handle various data types, including CSV, Excel, SQL databases, and more. It provides efficient tools for indexing, merging, and reshaping datasets, as

well as advanced data operations such as time series analysis, text data processing, and statistical analysis. By using Pandas, developers can easily manipulate and preprocess data, which is a crucial step in the Machine Learning workflow.

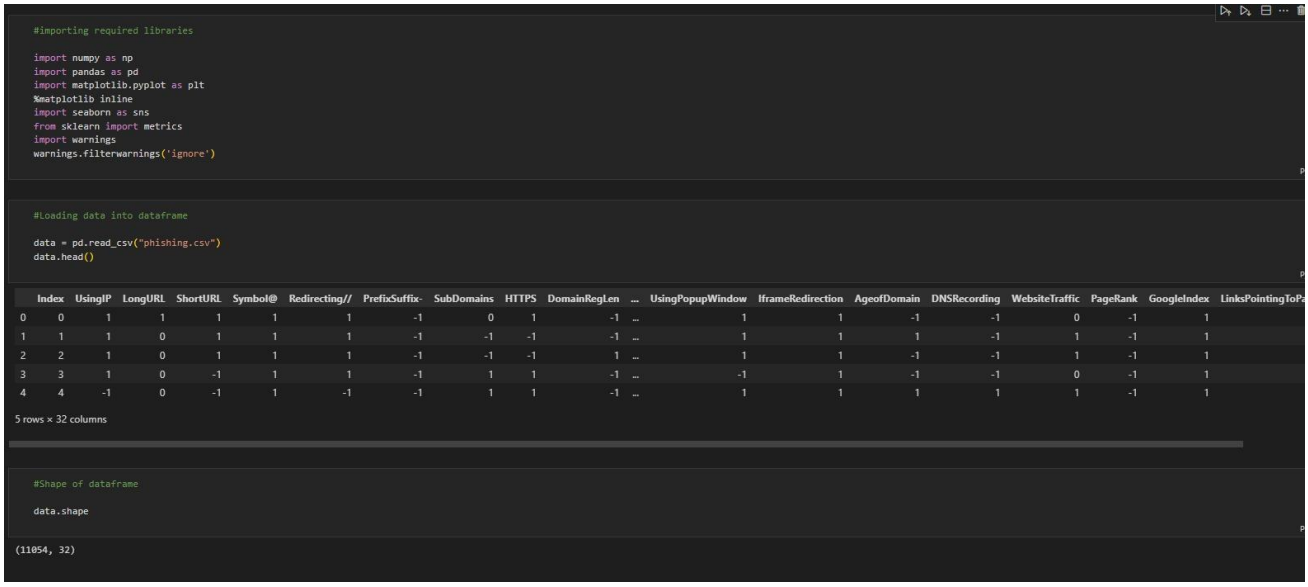
Matpoltlib:

Matplotlib is not only limited to 2D plotting. It also supports 3D plotting and animation as well. Apart from creating static plots and figures, it also allows users to create interactive visualizations using widgets and tools provided by the library. Moreover, it can be easily integrated with other Python libraries such as NumPy and Pandas.

APPENDIX 2

The web application for the output window as well as the code for constructing the Phishing website detection algorithm and training the model are all contained in this section.

Model training and Phishing website Detection:



```
#importing required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

```
#Loading data into dataframe

data = pd.read_csv("phishing.csv")
data.head()
```

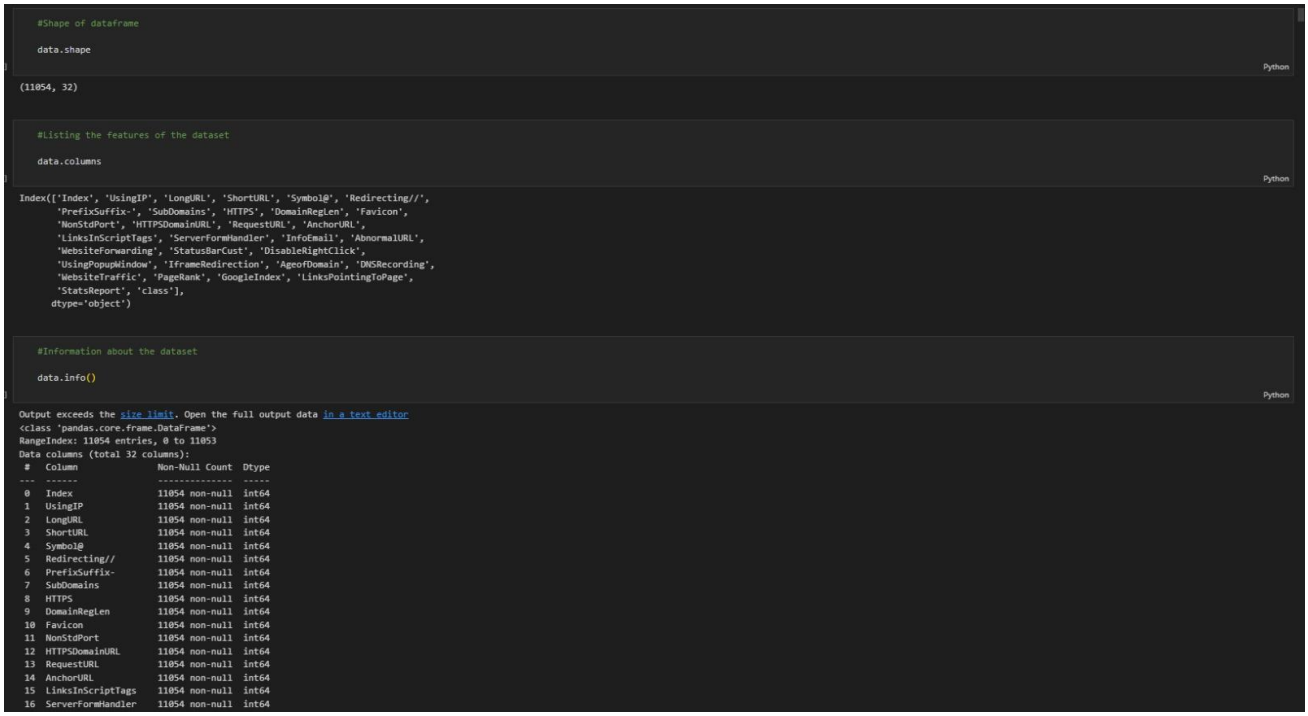
	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	...	UsingPopupWindow	IframeRedirection	AgeofDomain	DNSRecording	WebsiteTraffic	PageRank	GoogleIndex	LinksPointingToPa
0	0	1	1	1	1	1	-1	0	1	-1	...	1	1	-1	-1	0	-1	1	
1	1	1	0	1	1	1	-1	-1	-1	-1	...	1	1	1	-1	1	-1	1	
2	2	1	0	1	1	1	-1	-1	-1	1	...	1	1	-1	-1	1	-1	1	
3	3	1	0	-1	1	1	-1	1	1	-1	...	-1	1	-1	-1	0	-1	1	
4	4	-1	0	-1	1	-1	-1	1	1	-1	...	1	1	1	1	1	-1	1	

5 rows × 32 columns

```
#Shape of dataframe

data.shape
```

```
(11054, 32)
```

Fig 12.1 Loading Data into Data Frame



```
#Shape of dataframe

data.shape
```

Python

```
(11054, 32)
```

```
#Listing the features of the dataset

data.columns
```

Python

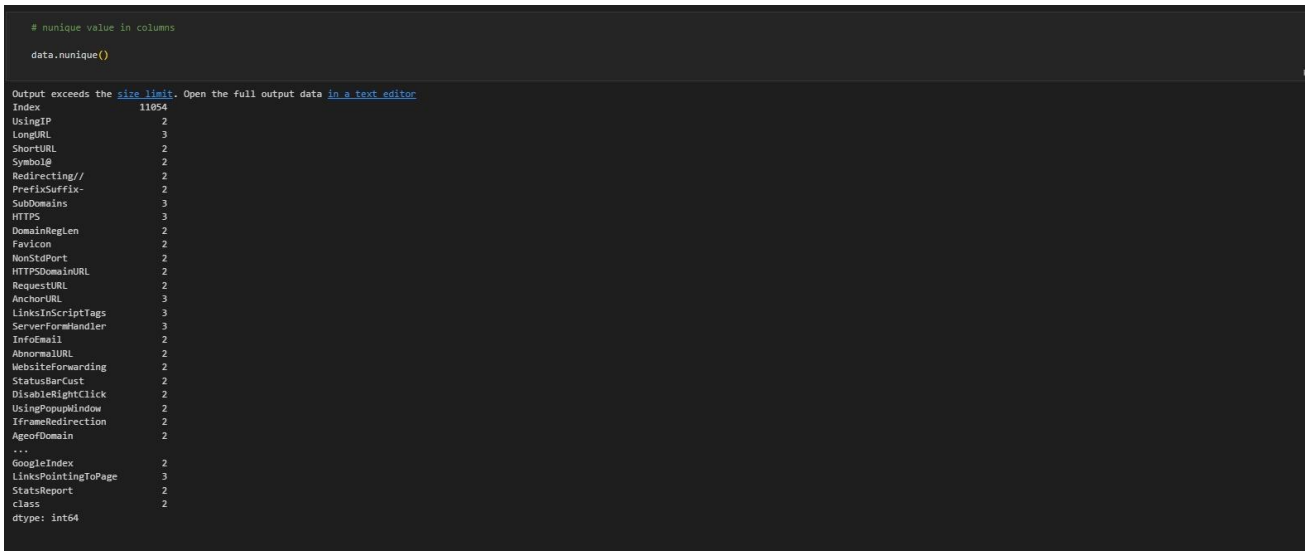
```
Index(['Index', 'UsingIP', 'LongURL', 'ShortURL', 'Symbol@', 'Redirecting//',
       'PrefixSuffix-', 'SubDomains', 'HTTPS', 'DomainRegLen', 'Favicon',
       'NonStdPort', 'HTTPSDomainURL', 'RequestURL', 'AnchorURL',
       'LinksInScriptTags', 'ServerFormHandler', 'InfoEmail', 'AbnormalURL',
       'WebsiteForwarding', 'StatusBarCust', 'DisableRightClick',
       'UsingPopupWindow', 'IframeRedirection', 'AgeofDomain', 'DNSRecording',
       'WebsiteTraffic', 'PageRank', 'GoogleIndex', 'LinksPointingToPage',
       'StatsReport', 'class'],
      dtype='object')
```

```
#Information about the dataset

data.info()
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11054 entries, 0 to 11053
Data columns (total 32 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Index               11054 non-null  int64
 1   UsingIP             11054 non-null  int64
 2   LongURL             11054 non-null  int64
 3   ShortURL            11054 non-null  int64
 4   Symbol@             11054 non-null  int64
 5   Redirecting//       11054 non-null  int64
 6   PrefixSuffix-       11054 non-null  int64
 7   SubDomains          11054 non-null  int64
 8   HTTPS               11054 non-null  int64
 9   DomainRegLen        11054 non-null  int64
 10  Favicon             11054 non-null  int64
 11  NonStdPort          11054 non-null  int64
 12  HTTPSDomainURL      11054 non-null  int64
 13  RequestURL          11054 non-null  int64
 14  AnchorURL           11054 non-null  int64
 15  LinksInScriptTags   11054 non-null  int64
 16  ServerFormHandler   11054 non-null  int64
```

Fig 12.2 Listing the Features and information about the Data set



```
# nunique value in columns

data.nunique()
```

```
Output exceeds the size limit. Open the full output data in a text editor
Index               11054
UsingIP                 2
LongURL                 3
ShortURL                2
Symbol@                 2
Redirecting//           2
PrefixSuffix-           2
SubDomains              3
HTTPS                   3
DomainRegLen            2
Favicon                 2
NonStdPort              2
HTTPSDomainURL          2
RequestURL              2
AnchorURL               3
LinksInScriptTags       3
ServerFormHandler       3
InfoEmail               2
AbnormalURL             2
WebsiteForwarding       2
StatusBarCust           2
DisableRightClick       2
UsingPopupWindow        2
IframeRedirection       2
AgeofDomain             2
...
GoogleIndex             2
LinksPointingToPage     3
StatsReport             2
class                   2
dtype: int64
```

Fig 12.3 Unique Values In Coloumn

```
#description of dataset

data.describe().T
```

Python

	count	mean	std	min	25%	50%	75%	max
UsingIP	11054.0	0.313914	0.949495	-1.0	-1.0	1.0	1.0	1.0
LongURL	11054.0	-0.633345	0.765973	-1.0	-1.0	-1.0	-1.0	1.0
ShortURL	11054.0	0.738737	0.674024	-1.0	1.0	1.0	1.0	1.0
Symbol@	11054.0	0.700561	0.713625	-1.0	1.0	1.0	1.0	1.0
Redirecting//	11054.0	0.741632	0.670837	-1.0	1.0	1.0	1.0	1.0
PrefixSuffix-	11054.0	-0.734938	0.678165	-1.0	-1.0	-1.0	-1.0	1.0
SubDomains	11054.0	0.064049	0.817492	-1.0	-1.0	0.0	1.0	1.0
HTTPS	11054.0	0.251040	0.911856	-1.0	-1.0	1.0	1.0	1.0
DomainRegLen	11054.0	-0.336711	0.941651	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11054.0	0.628551	0.777804	-1.0	1.0	1.0	1.0	1.0
NonStdPort	11054.0	0.728243	0.685350	-1.0	1.0	1.0	1.0	1.0
HTTPSDomainURL	11054.0	0.675231	0.737640	-1.0	1.0	1.0	1.0	1.0
RequestURL	11054.0	0.186720	0.982458	-1.0	-1.0	1.0	1.0	1.0
AnchorURL	11054.0	-0.076443	0.715116	-1.0	-1.0	0.0	0.0	1.0
LinksInScriptTags	11054.0	-0.118238	0.763933	-1.0	-1.0	0.0	0.0	1.0
ServerFormHandler	11054.0	-0.595712	0.759168	-1.0	-1.0	-1.0	-1.0	1.0
InfoEmail	11054.0	0.635788	0.771899	-1.0	1.0	1.0	1.0	1.0
AbnormalURL	11054.0	0.705446	0.708796	-1.0	1.0	1.0	1.0	1.0
WebsiteForwarding	11054.0	0.115705	0.319885	0.0	0.0	0.0	0.0	1.0
StatusBarCust	11054.0	0.762077	0.647516	-1.0	1.0	1.0	1.0	1.0
DisableRightClick	11054.0	0.913877	0.406009	-1.0	1.0	1.0	1.0	1.0
UsingPopupWindow	11054.0	0.613353	0.789845	-1.0	1.0	1.0	1.0	1.0
IframeRedirection	11054.0	0.816899	0.576807	-1.0	1.0	1.0	1.0	1.0
AgeofDomain	11054.0	0.061335	0.998162	-1.0	-1.0	1.0	1.0	1.0
DNSRecording	11054.0	0.377239	0.926158	-1.0	-1.0	1.0	1.0	1.0
WebsiteTraffic	11054.0	0.287407	0.827680	-1.0	0.0	1.0	1.0	1.0
PageRank	11054.0	-0.483626	0.875314	-1.0	-1.0	-1.0	1.0	1.0
GoogleIndex	11054.0	0.721549	0.692395	-1.0	1.0	1.0	1.0	1.0
LinksPointingToPage	11054.0	0.343948	0.569936	-1.0	0.0	0.0	1.0	1.0
StatsReport	11054.0	0.719739	0.694276	-1.0	1.0	1.0	1.0	1.0
class	11054.0	0.113986	0.993527	-1.0	-1.0	1.0	1.0	1.0

Fig 12.4 Description of Data Set

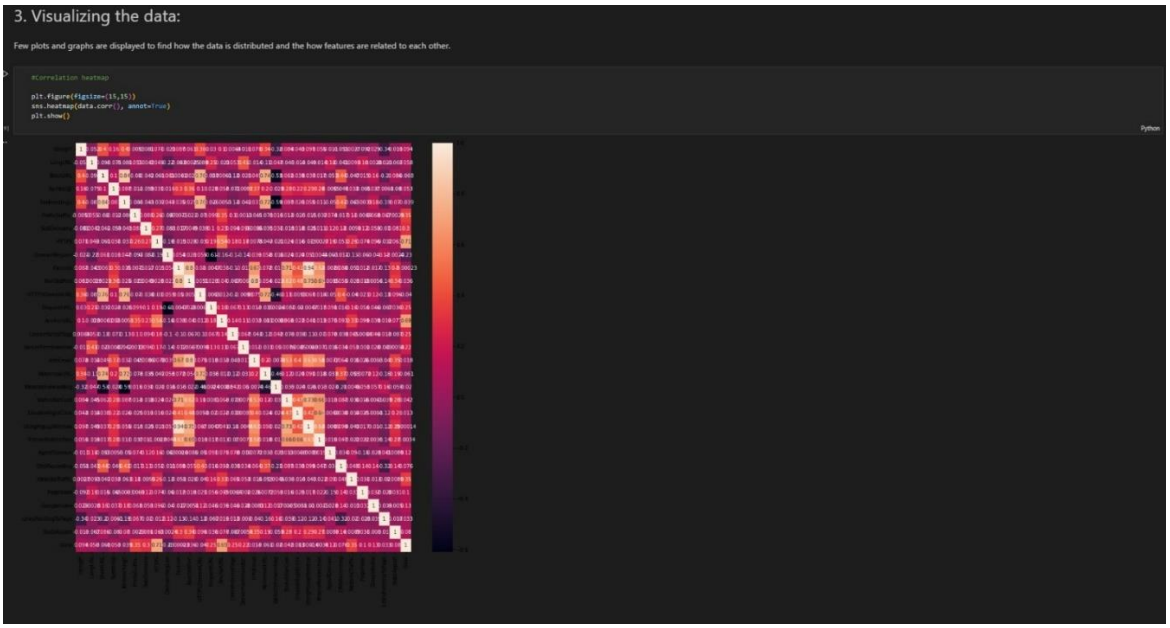


Fig 12.5 Heatmap showing the dataset's different parameters

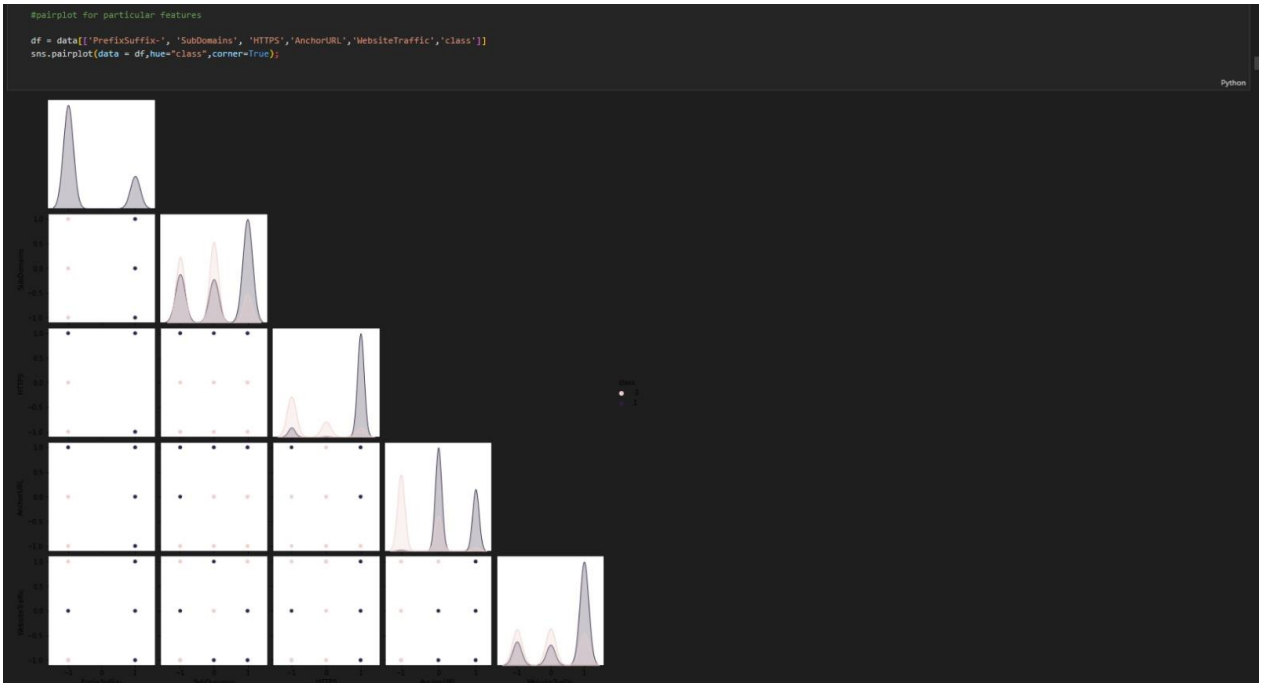


Fig 12.6 Pair plot for particular features

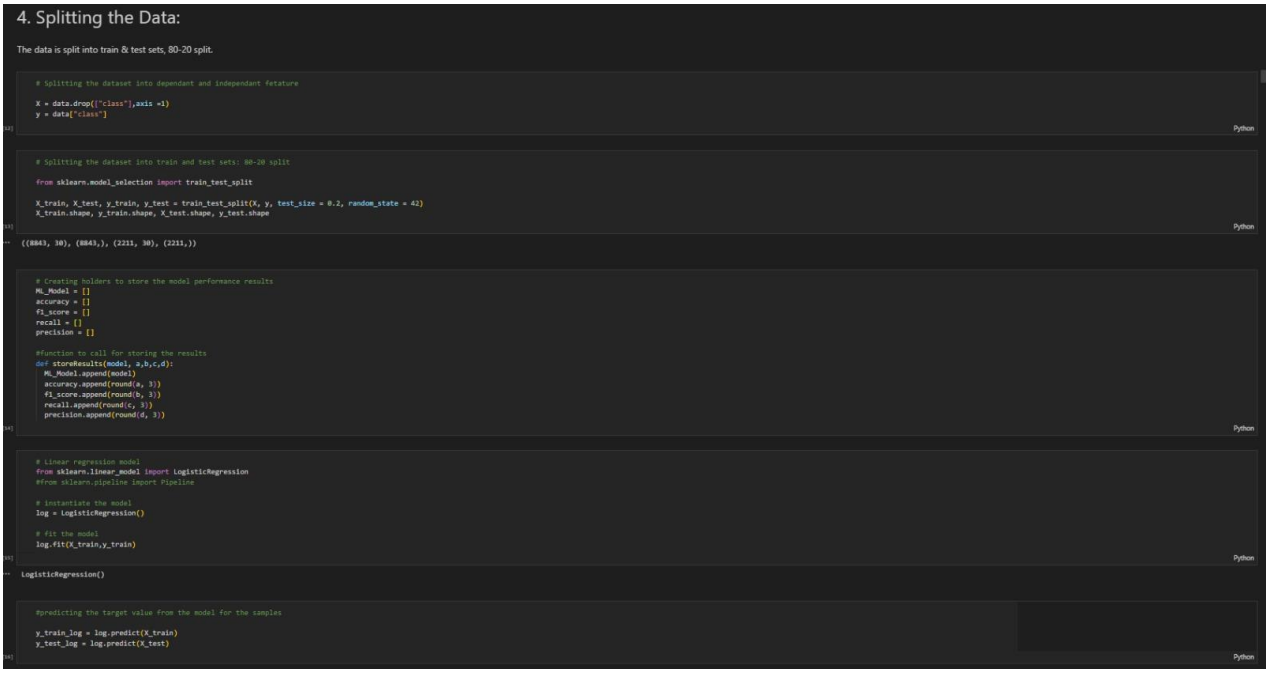


Fig 12.7 Splitting the data

```python
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_log = metrics.accuracy_score(y_train,y_train_log)
acc_test_log = metrics.accuracy_score(y_test,y_test_log)
print("Logistic Regression : Accuracy on training Data: {:.3f}".format(acc_train_log))
print("Logistic Regression : Accuracy on test Data: {:.3f}".format(acc_test_log))
print()

f1_score_train_log = metrics.f1_score(y_train,y_train_log)
f1_score_test_log = metrics.f1_score(y_test,y_test_log)
print("Logistic Regression : f1_score on training Data: {:.3f}".format(f1_score_train_log))
print("Logistic Regression : f1_score on test Data: {:.3f}".format(f1_score_test_log))
print()

recall_score_train_log = metrics.recall_score(y_train,y_train_log)
recall_score_test_log = metrics.recall_score(y_test,y_test_log)
print("Logistic Regression : Recall on training Data: {:.3f}".format(recall_score_train_log))
print("Logistic Regression : Recall on test Data: {:.3f}".format(recall_score_test_log))
print()

precision_score_train_log = metrics.precision_score(y_train,y_train_log)
precision_score_test_log = metrics.precision_score(y_test,y_test_log)
print("Logistic Regression : precision on training Data: {:.3f}".format(precision_score_train_log))
print("Logistic Regression : precision on test Data: {:.3f}".format(precision_score_test_log))
```

[17] Python

```
Logistic Regression : Accuracy on training Data: 0.927
Logistic Regression : Accuracy on test Data: 0.934

Logistic Regression : f1_score on training Data: 0.935
Logistic Regression : f1_score on test Data: 0.941

Logistic Regression : Recall on training Data: 0.943
Logistic Regression : Recall on test Data: 0.953

Logistic Regression : precision on training Data: 0.927
Logistic Regression : precision on test Data: 0.930
```

Fig 12.8 Results of a Logistic Regression

```python
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_svc = metrics.accuracy_score(y_train,y_train_svc)
acc_test_svc = metrics.accuracy_score(y_test,y_test_svc)
print("Support Vector Machine : Accuracy on training Data: {:.3f}".format(acc_train_svc))
print("Support Vector Machine : Accuracy on test Data: {:.3f}".format(acc_test_svc))
print()

f1_score_train_svc = metrics.f1_score(y_train,y_train_svc)
f1_score_test_svc = metrics.f1_score(y_test,y_test_svc)
print("Support Vector Machine : f1_score on training Data: {:.3f}".format(f1_score_train_svc))
print("Support Vector Machine : f1_score on test Data: {:.3f}".format(f1_score_test_svc))
print()

recall_score_train_svc = metrics.recall_score(y_train,y_train_svc)
recall_score_test_svc = metrics.recall_score(y_test,y_test_svc)
print("Support Vector Machine : Recall on training Data: {:.3f}".format(recall_score_train_svc))
print("Support Vector Machine : Recall on test Data: {:.3f}".format(recall_score_test_svc))
print()

precision_score_train_svc = metrics.precision_score(y_train,y_train_svc)
precision_score_test_svc = metrics.precision_score(y_test,y_test_svc)
print("Support Vector Machine : precision on training Data: {:.3f}".format(precision_score_train_svc))
print("Support Vector Machine : precision on test Data: {:.3f}".format(precision_score_test_svc))
```

Python

```
Support Vector Machine : Accuracy on training Data: 0.969
Support Vector Machine : Accuracy on test Data: 0.964

Support Vector Machine : f1_score on training Data: 0.973
Support Vector Machine : f1_score on test Data: 0.968

Support Vector Machine : Recall on training Data: 0.980
Support Vector Machine : Recall on test Data: 0.980

Support Vector Machine : precision on training Data: 0.965
Support Vector Machine : precision on test Data: 0.957
```

Fig 13.1 Results of a Support Vector Machine

```python
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_nb = metrics.accuracy_score(y_train,y_train_nb)
acc_test_nb = metrics.accuracy_score(y_test,y_test_nb)
print("Naive Bayes Classifier : Accuracy on training Data: {:.3f}".format(acc_train_nb))
print("Naive Bayes Classifier : Accuracy on test Data: {:.3f}".format(acc_test_nb))
print()

f1_score_train_nb = metrics.f1_score(y_train,y_train_nb)
f1_score_test_nb = metrics.f1_score(y_test,y_test_nb)
print("Naive Bayes Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_nb))
print("Naive Bayes Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_nb))
print()

recall_score_train_nb = metrics.recall_score(y_train,y_train_nb)
recall_score_test_nb = metrics.recall_score(y_test,y_test_nb)
print("Naive Bayes Classifier : Recall on training Data: {:.3f}".format(recall_score_train_nb))
print("Naive Bayes Classifier : Recall on test Data: {:.3f}".format(recall_score_test_nb))
print()

precision_score_train_nb = metrics.precision_score(y_train,y_train_nb)
precision_score_test_nb = metrics.precision_score(y_test,y_test_nb)
print("Naive Bayes Classifier : precision on training Data: {:.3f}".format(precision_score_train_nb))
print("Naive Bayes Classifier : precision on test Data: {:.3f}".format(precision_score_test_nb))
```

Python

```
Naive Bayes Classifier : Accuracy on training Data: 0.605
Naive Bayes Classifier : Accuracy on test Data: 0.605

Naive Bayes Classifier : f1_score on training Data: 0.451
Naive Bayes Classifier : f1_score on test Data: 0.454

Naive Bayes Classifier : Recall on training Data: 0.292
Naive Bayes Classifier : Recall on test Data: 0.294

Naive Bayes Classifier : precision on training Data: 0.997
Naive Bayes Classifier : precision on test Data: 0.995
```

Fig 13.2 Results of a Naïve Bayes Classifier

```python
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_forest = metrics.accuracy_score(y_train,y_train_forest)
acc_test_forest = metrics.accuracy_score(y_test,y_test_forest)
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
print()

f1_score_train_forest = metrics.f1_score(y_train,y_train_forest)
f1_score_test_forest = metrics.f1_score(y_test,y_test_forest)
print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train_forest))
print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test_forest))
print()

recall_score_train_forest = metrics.recall_score(y_train,y_train_forest)
recall_score_test_forest = metrics.recall_score(y_test,y_test_forest)
print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train_forest))
print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test_forest))
print()

precision_score_train_forest = metrics.precision_score(y_train,y_train_forest)
precision_score_test_forest = metrics.precision_score(y_test,y_test_tree)
print("Random Forest : precision on training Data: {:.3f}".format(precision_score_train_forest))
print("Random Forest : precision on test Data: {:.3f}".format(precision_score_test_forest))
```

Python

```
Random Forest : Accuracy on training Data: 0.991
Random Forest : Accuracy on test Data: 0.967

Random Forest : f1_score on training Data: 0.991
Random Forest : f1_score on test Data: 0.970

Random Forest : Recall on training Data: 0.992
Random Forest : Recall on test Data: 0.972

Random Forest : precision on training Data: 0.991
Random Forest : precision on test Data: 0.966
```

Fig 12.3 Results of a Random Forest Classifier

```python
training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10)
for n in depth:
    forest_test =  GradientBoostingClassifier(learning_rate = n*0.1)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("learning_rate")
plt.legend();
```

Python

Fig 12.4 Plotting the training and testing

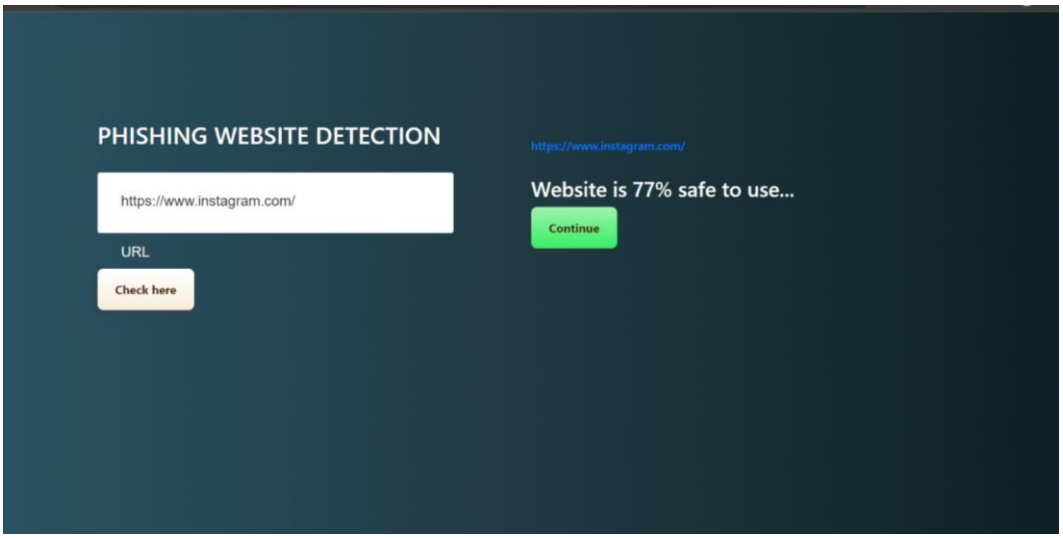
Output Module:

In [4]:

```python
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()


app = Flask(_name_)

@app.route("/", methods=["GET", "POST"])
def index():
    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('index.html',xx =round(y_pro_non_phishing,2),url=url )
    return render_template("index.html", xx =-1)


if _name_ == "_main_":
    app.run(debug=True)

```

Fig 12.5

Output Window:



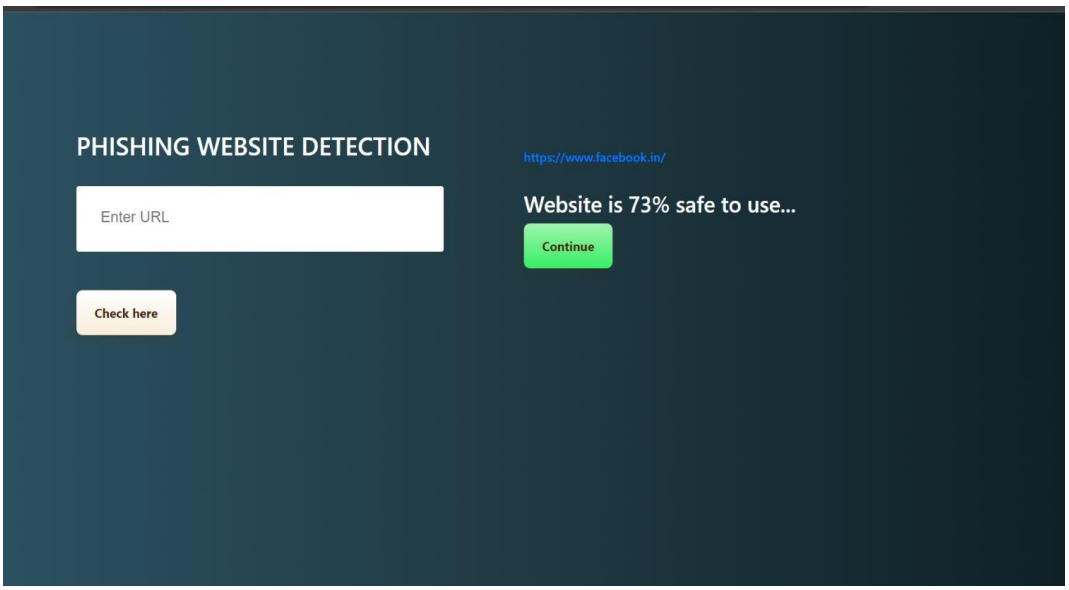


Fig 12.7 Screenshots showing the final interface and output of a particular URL.

PUBLICATION DETAILS

We submitted our research paper for publication at International Conference on Internet of Things as proof of publication in figure B.2 The research paper cover page has been attached below.

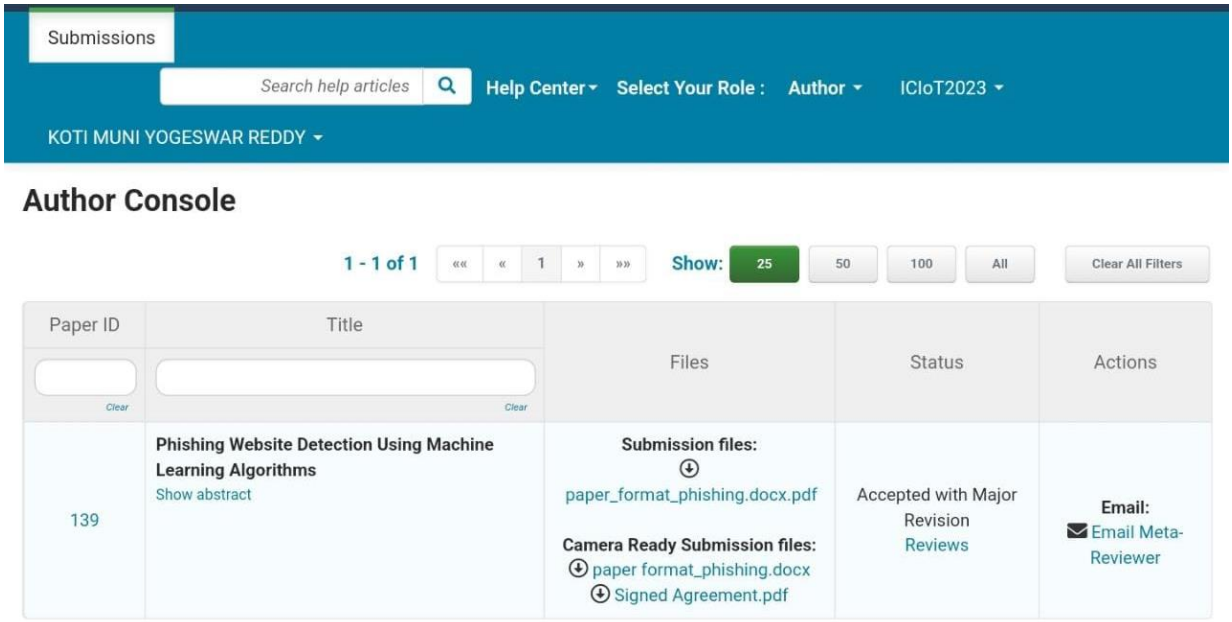


Fig B Submission proof

Phishing Website Detection using Machine Learning Algorithms

Koti Muni Yogeswar Reddy
Department of Computing Technologies
SRM Institute of Science and Technology
Chennai, India
ky7113@srmist.edu.in

Medida Shivananda reddy
Department of Computing Technologies
SRM Institute of Science and Technology
Chennai, India
mr8397@srmist.edu.in

Dr. D.Vinod
Department of Computing Technologies
SRM Institute of Science and Technology
Chennai, India
vinodd@srmist.edu.in

Abstract*— Many people make purchases and payments on different websites when shopping online. Many online platforms often request personal information from their users, such as login credentials, financial details, and other sensitive data. These websites fall under the category of scam websites. Our system is clever, adaptable, and effective for detecting and predicting scam websites because it is built on machine learning technology. Using classification methods and methodologies, we evaluated the validity of the criteria from the phishing data sets. Based on a few crucial elements, such as the URL, domain identity, security, and encryption parameters, it is possible to calculate the ultimate phishing detection rate. Our system will use a machine learning algorithm to detect fraudulent websites. This application can be utilized by various e-commerce enterprises to secure the entire transaction process. Comparing this system's machine learning method to other conventional classification algorithms, it performs better. The user can buy goods without any hesitation online with the aid of this method.

Keywords*——Machine Learning, phishing website detection, personal information, phishing domain characteristics.

I. Introduction

Phishing is the practice of impersonating a trustworthy website in an effort to trick people into disclosing their private information, including names, passwords, account numbers, social security numbers, and other details. Phishing scams are arguably the most prevalent type of hacking currently in use. Phishing attacks can take place on a plethora of websites, including those that handle online payments, webmail, banking institutions, file sharing, cloud storage, and many others. Among different business sectors, webmail and online payments are the most vulnerable to phishing attacks. These attacks can be carried out through email phishing schemes or spear phishing, which highlights the importance of users being cautious and not entirely relying on widely used security software. Machine learning is considered to be an effective approach for fraud detection as it addresses the limitations of current methods. One of the key goals of this project is to identify illicit URLs and verify the legitimacy of websites using this approach. Both email and pop-up notifications should be used to alert users who are trying to access a banned website that they are attempting to do so. This effort will allow the supervisor to add Websites that will warn users when they search for certain terms. User scope and system scope are two well-known terminologies for project scopes. A portion of the mechanism's fault lies with the user. To be in compliance with the system, a few rules and guidelines must be followed. Users can be alerted when a website on a blacklist is visited. The administrator can record the URLs on the block and notify users. The system includes capabilities such as website capture, website watching, pop-up, and email warning display, as well as perusing blacklisted websites.

II. State of the art (LITERATURE SURVEY)

[1] Title: Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection

In a publication by Zuochao Dou and Abdallah Khreishah, phishing is defined as a type of cyberattack that uses social engineering tactics and advanced techniques to extract sensitive information from website visitors. The Anti-Phishing Working Group reports a substantial surge in fraudulent websites, with a yearly growth rate of 36.29% on average in the past six years, and 97.36% in the last two years. As a result of this increase, the cybersecurity industry is becoming more and more interested in reducing phishing assaults. On the basis of their distinctive content, network, and URL traits, phishing efforts have been the subject of extensive study and development. Intuitions, data analysis techniques, and assessment procedures vary widely between existing approaches.

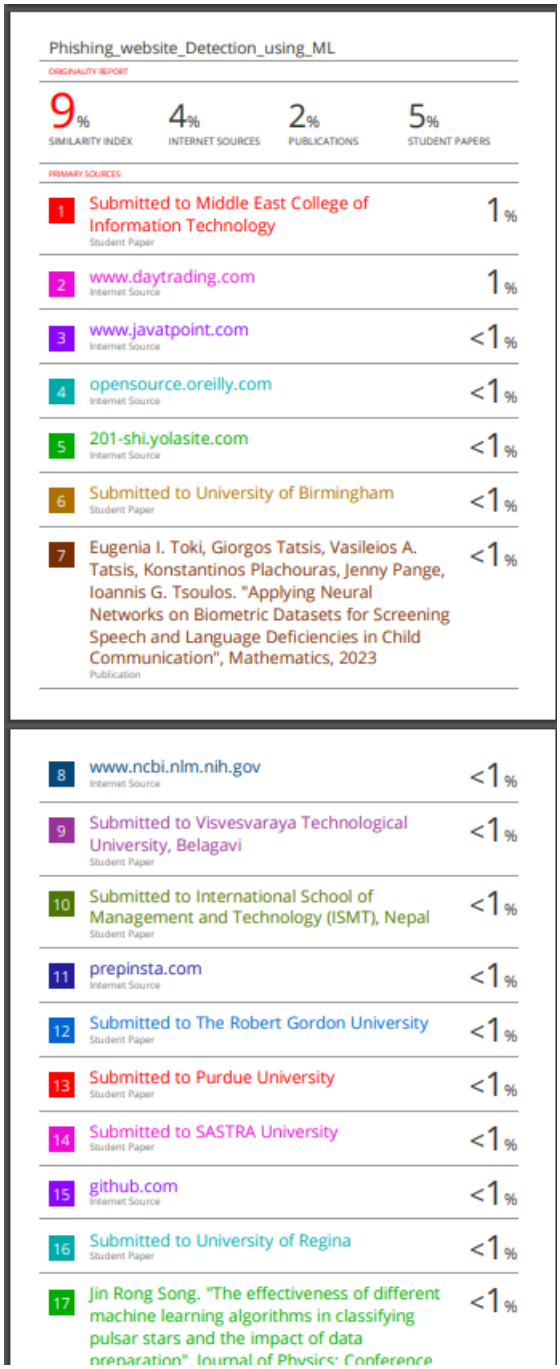
[2] The title: Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity.

Authors include Wenqian Tian, Zhenkai Liang

Fig B.1 Cover page of a Research paper

FigureB.2 Publication Notification

PLAGIARISM REPORT



Phishing_website_Detection_using_ML

ORIGINALITY REPORT

9%	4%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Middle East College of Information Technology Student Paper	1%
2	www.daytrading.com Internet Source	1%
3	www.javatpoint.com Internet Source	<1%
4	opensource.oreilly.com Internet Source	<1%
5	201-shi.yolasite.com Internet Source	<1%
6	Submitted to University of Birmingham Student Paper	<1%
7	Eugenia I. Toki, Giorgos Tatsis, Vasileios A. Tatsis, Konstantinos Plachouras, Jenny Pange, Ioannis G. Tsoulos. "Applying Neural Networks on Biometric Datasets for Screening Speech and Language Deficiencies in Child Communication", Mathematics, 2023 Publication	<1%
8	www.ncbi.nlm.nih.gov Internet Source	<1%
9	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	<1%
10	Submitted to International School of Management and Technology (ISMT), Nepal Student Paper	<1%
11	prepinsta.com Internet Source	<1%
12	Submitted to The Robert Gordon University Student Paper	<1%
13	Submitted to Purdue University Student Paper	<1%
14	Submitted to SASTRA University Student Paper	<1%
15	github.com Internet Source	<1%
16	Submitted to University of Regina Student Paper	<1%
17	Jin Rong Song. "The effectiveness of different machine learning algorithms in classifying pulsar stars and the impact of data preparation", Journal of Physics: Conference	<1%

Fig C plagiarism index

Format - I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES

(To be attached in the dissertation/ project report)

1	Name of the Candidate **(IN BLOCK LETTERS)**	MEDIDA SHIVANANDA REDDY
2	Address of the Candidate	HNO 9-1, Reddypalli, khammam – 507163 Mobile :6309882160
3	Registration Number	RA1911003010869
4	Date of Birth	06 December 2000
5	Department	Computer Science and Engineering
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	Phishing Website Prediction Using Machine Learning Algorithms
8	Whether the above project /dissertation is done by	~~Individual~~ or group : (Strike whichever is not applicable) a) If the project/ dissertation is done in group, then how many students together completed the project : 2 (Two) b) Mention the Name & Register number of other candidates : Koti Muni Yogeswar REDDY, RA1911003010876
9	Name and address of the SuperGuide	Dr. D. Vinod Assistant Professor Department of Computer Science and Engineering SRM Institute of Science and Technology Kattankulatur - 603 203. **Mail ID:** vinodd@srmist.edu.in Mobile Number: 9962046411
10	Name and address of Co-Supervisor / Co- Guide (if any)	NIL **Mail ID: Mobile Number:**

11	Software Used	Turnitin		
12	Date of Verification	11– May -2023		
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	INTRODUCTION	0%	0%	0%
2	LITERATURE SURVEY	2%	2%	2%
3	TECHNICAL SPECIFICATIONS	2%	2%	2%
4	ARCHITECTURE	2%	2%	2%
5	MODULES	3%	3%	3%
6	PROJECT DEMONSTRATION AND RESULTS	0%	0%	0%
7	CONCLUSION	0%	0%	0%
8	FUTURE ENHANCEMENTS	0%	0%	0%
9				
10				
Appendices		9%	9%	9%

I / We declare that the above information have been verified and found true to the best of my / our knowledge.

Signature of the Candidate	**Name & Signature of the Staff (Who uses the plagiarism check software)**
Name & Signature of the Supervisor/ Guide	**Name & Signature of the Co-Supervisor/Co- Guide**
Name & Signature of the HOD	