



# **DETECTING PHISHING WEBSITE AND SPAM CONTENT USING MACHINE LEARNING**



## **A PROJECT REPORT**

*Submitted by*

**ANTHONI THOMAS.R (732119205005)**

**KARTHIKEYAN.D (732119205022)**

**PON PANDIAN.P (732119205038)**

**SEKAR.S (732119205047)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**

**NANDHA COLLEGE OF TECHNOLOGY, ERODE-52**

**ANNA UNIVERSITY :: CHENNAI – 600 025**

**MAY 2023**

# **ANNA UNIVERSITY : CHENNAI - 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**DETECTING PHISHING WEBSITE AND SPAM CONTENT USING MACHINE LEARNING**” is the Bonafide work of “**ANTHONI THOMAS.R (732119205005), KARTHIKEYAN.D (732119205022), PON PANDIAN.P (732119205038), SEKAR.S (732119205047)**” who carried out the project work under my supervision.

### **SIGNATURE**

**Dr.M.KARTHICK,  
M.E., Ph.D.,**

### **HEAD OF THE DEPARTMENT,**

Associate Professor,  
Department of Information  
Technology,  
Nandha College of Technology,  
Erode-638 052.

### **SIGNATURE**

**Mr.N.PRASAD,  
M.E.,**

### **SUPERVISOR,**

Assistant Professor,  
Department of Information  
Technology,  
Nandha College of Technology,  
Erode-638 052.

Submitted for the University project Viva-Voce exam held on.....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We express our thanks to our beloved chairman of Sri Nandha Educational Trust **Thiru.V.Shanmugan**, our beloved Secretary of Sri Nandha Educational Trust **Thiru.S.Nandhakumar Pradeep**, our beloved Secretary of Sri Nandha Educational Institutions **Thiru.S.Thirumoorthi**, and to our Chief Executive Officer of Nandha Educational Institutions **Dr.S.Arumugam** for their support in successful completion of our project work.

We wish to express our deep sense of gratitude to our beloved Principal **Dr.S.Nandagopal, M.E., Ph.D.**, for the excellent facilities and constant support provided during the course study and project.

We articulate our genuine and sincere thanks to our dear hearted Head of the Department and Project Coordinator **Dr.M.Karthick, M.E., Ph.D.**, who has been the key spring of motivation to us through out the completion of our course and our project work.

We articulate our genuine and sincere thanks to our Project Guide **Mr.N.Prasad, M.E.**, who used to guide as frequently at critical junctures during the project work.

We are very much gratified to all the teaching and non-teaching staff of our department who has direct and indirect stroke throughout our progress. Our heart felt thanks to our friends who have supported us with their unconditional love and encouragement. Finally, we would like to thank the Almighty for the blessings.

## **ABSTRACT**

This Project presents a novel approach for Detecting Phishing Websites and Spam Content using Machine Learning algorithms. Specifically, we focus on using the K-Neighbors Classifier algorithm to accurately identify and classify suspicious websites and content. Our proposed approach involves training the model on a large dataset of known Phishing Website and Spam Content, and then using the model to classify new websites and content based on their features and characteristics. We demonstrate the effectiveness of our approach through extensive experimentation and evaluation on a Real-World Dataset. Our results show that our approach is highly effective in detecting and classifying phishing websites and spam content, achieving high accuracy and low false positive rates. Overall, our project presents a promising approach to combatting web phishing attacks using machine learning techniques.

**KEYWORDS** - Machine Learning, dataset, Spam Content, Phishing Website, High Accuracy, K-Neighbors Classifier algorithm.

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGENO.</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF FIGURES</b>	<b>x</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>xii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 PHISHING WEBSITES AND SPAM CONTENT	1
	1.2 PROJECT OVERVIEW	2
	1.2.1 MACHINE LEARNING	2
	1.2.2 K-NEIGHBORS CLASSIFIERS	4
	1.2.3 SUPPORT VECTOR MACHINE	5
	1.2.4 CHATBOT	6
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>7</b>
	2.1 A HYBRID APPROACH FOR DETECTING PHISHING WEBSITES USING FEATURE EXTRACTION AND MACHINE LEARNING	7
	2.2 THE ENSEMBLE STRATEGY AIDED IN OBTAINING A HIGH ACCURACY SCORE	8

2.3	EFFECTIVE SPAM IDENTIFICATION USING BOTH UNIVARIATE AND MULTIVARIATE DISTRIBUTION ACROSS USER RATINGS	9
2.4	SOCIAL FEATURES LIKE COMMENTS ETC., ARE COMBINED WITH TEXTUAL FEATURES YIELDS BETTER RESULTS	10
2.5	AUTOMATIC IDENTIFICATION OF SPAM TEXT IS DONE WITH 42 FEATURES USING MACHINE LEARNING TECHNIQUES	11
2.6	INSTEAD OF USING SPAM TRIGGER WORDS, WHICH MAY FAIL A LEXICON BASED APPROACH IS USED TO FILTER THE DATA	12
<b>3</b>	<b>ROLE OF CLOUD IN CREATING AND INTEGRATING A CHATBOT WITH APPLICATIONS</b>	<b>13</b>
3.1	A ROLE OF CLOUD	13
3.2	IMPLEMENTATION PROCESS FOR CREATING AND INTEGRATING A CHATBOT IN IBM CLOUD	13
3.3	IBM WATSON ASSISTANT	14

<b>4</b>	<b>SYSTEM REQUIREMENTS</b>	<b>15</b>
	4.1 HARDWARE REQUIREMENTS	15
	4.2 SOFTWARE REQUIREMENTS	15
<b>5</b>	<b>SOFTWARE DESCRIPTIONS</b>	<b>16</b>
	5.1 FEATURES OF SOFTWARE	16
	5.2 HTML 5	16
	5.3 MYSQL DATABASE	17
	5.4 PYTHON	17
	5.5 FLASK	18
	5.6 VISUAL STUDIO CODE	18
	5.7 IBM CLOUD	19
<b>6</b>	<b>FEASIBILITY STUDY</b>	<b>20</b>
	6.1 TECHNICAL FEASIBILITY	20
	6.2 OPERATIONAL FEASIBILITY	21
	6.3 ECONOMICAL FEASIBILITY	22
<b>7</b>	<b>SYSTEM DESIGN</b>	<b>23</b>
	7.1 EXISTING SYSTEM	23
	7.1.1 DRAWBACKS OF EXISTING SYSTEM	24
	7.2 PROPOSED SYSTEM	24

	7.2.1	ADVANTAGES OF PROPOSED SYSTEM	25
<b>8</b>		<b>IMPLEMENTATION</b>	<b>26</b>
	8.1	MODULE DESCRIPTION	26
	8.1.1	USER REGISTRATION AND LOGIN	26
	8.1.2	SPAM TEXT DETECTION	27
	8.1.3	PHISHING URL DETECTION	27
	8.1.4	CHATBOT	27
	8.2	SYSTEM FLOW DIAGRAM	28
	8.3	USE CASE DIAGRAM	29
	8.4	CLASS DIAGRAM	30
	8.5	ACTIVITY DIAGRAM	31
<b>9</b>		<b>TESTING</b>	<b>32</b>
	9.1	SYSTEM TESTING	32
	9.1.1	UNIT TESTING	33
	9.1.2	INTEGRATION TESTING	33
	9.1.3	VALIDATION TESTING	34
	9.1.4	OUTPUT TESTING	35
	9.1.5	USER ACCEPTANCE TESTING	35



<b>10</b>	<b>RESULT</b>	<b>36</b>
	10.1 EXPERIMENTAL ANALYSIS	36
	10.2 ACCURACY	37
	10.3 PRECISION	37
	10.4 RECALL	37
	10.5 F1 SCORE	37
	10.6 KNN RESULT	38
<b>11</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>39</b>
	11.1 CONCLUSION	39
	11.2 FUTURE ENHANCEMENT	40
	<b>APPENDICES</b>	<b>41</b>
	A.1 SOURCE CODE	41
	A.2 SCREENSHOTS	49
	<b>REFERENCES</b>	<b>53</b>
	<b>LIST OF PUBLICATION</b>	<b>55</b>

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
3.3	IBM Watson Assistant	14
8.1.1.1	Login Page	26
8.1.1.2	Registration Page	26
8.1.4	Chatbot	27
8.2	System Flow Diagram	28
8.3	Use Case Diagram	29
8.4	Class Diagram	30
8.5	Activity Diagram	31
10.1.1	Confusion Matrix	36
10.1.2	KNN	36
10.6	Prediction by KNN	38
A.2.1	Login Page	49
A.2.2	Registration Page	49
A.2.3	Home Page	50
A.2.4	Profile Page	50
A.2.5	Text Detection Page	51
A.2.6	Url Detection Page	51

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
A.2.7	Chatbot	52
A.2.8	Logout Page	52

## **LIST OF ABBREVIATIONS**

ML	:	Machine Learning
URL	:	Uniform Resource Locator
HTML	:	Hypertext Markup Language
CSS	:	Cascading Style Sheets
SQL	:	Structured Query Language
KNN	:	K-Nearest Neighbors
SVM	:	Support Vector Machine
HTTP	:	Hyper Text Transfer Protocol
RAM	:	Random Access Memory
MIN	:	Minimum
GB	:	Giga Byte
GHz	:	Giga Hertz
VS	:	Visual Studio
IBM	:	International Business Machines
CM	:	Confusion Matrix
TP	:	True Positive
TN	:	True Negative
FP	:	False Positive
FN	:	False Negative

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PHISHING WEBSITES AND SPAM CONTENT**

Phishing websites and spam contents are major cyber security threats that cause significant harm to users, such as identity theft and financial losses. Machine learning algorithms have emerged as powerful tools for identifying and detecting these malicious activities on the internet.

Machine learning algorithms can learn from large datasets of labeled examples to identify patterns and features that distinguish legitimate content from phishing websites and spam content. These algorithms can analyze various features, such as the structure of the website, email meta data, and the content of the message, to determine the likelihood of it being malicious.

In the case of phishing websites, machine learning algorithms can analyze various features such as URL structure, content, and visual elements to determine whether a website is trying to trick users into disclosing sensitive information. Similarly, for spam content, machine learning algorithms can analyze text and metadata to identify spam emails, social media posts, and other types of unwanted content.

Using machine learning for detecting phishing websites and spam content offers the potential to improve accuracy and reduce false positives. As attackers continue to evolve their tactics, machine learning can adapt and continue to stay ahead of the threats.

The goal of this project is to develop a machine learning model that can accurately detect phishing links in online content using the K-Nearest Neighbors (KNN) algorithm. Phishing links are fraudulent URLs that are designed to look like legitimate websites, but are used to steal sensitive information from unsuspecting users.

The KNN algorithm is a type of supervised learning that is commonly used for classification tasks. It works by finding the K nearest data points to a given data point, based on a similarity metric, and classifying the new data point based on the majority class of its nearest neighbors.

In this project, we will train our KNN model on a dataset of both legitimate and phishing links, which will be preprocessed to extract features such as URL length, domain name, and content similarity. We will then test the model's accuracy using a separate test dataset and optimize its performance by tuning the K parameter and distance metric.

Once the model is trained and optimized, we can use it to detect phishing links in real-world scenarios, such as scanning emails or social media posts for malicious links. The successful development and implementation of this project has the potential to greatly enhance online security and protect users from falling victim to phishing attacks.

## **1.2 PROJECT OVERVIEW**

### **1.2.1 MACHINE LEARNING**

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence.

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.

Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning.

Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain. In its application across business problems, machine learning is also referred to as predictive analytics

Learning algorithms work on the basis that strategies, algorithms, and inferences that worked well in the past are likely to continue working well in the future. These inferences can be obvious, such as "since the sun rose every morning for the last 10,000 days, it will probably rise tomorrow morning as well".

They can be nuanced, such as "X% of families have geographically separate species with color variants, so there is a Y% chance that undiscovered black swans exist".

### 1.2.2 K-NEIGHBORS CLASSIFIERS

KNN (K-Nearest Neighbors) is a type of supervised machine learning algorithm used for classification and regression. In the context of classification, KNN is a non-parametric method that uses a database of labeled examples to predict the classification of new, unseen examples.

The KNN algorithm works by calculating the distance between the new example and each example in the training set, then selecting the K closest examples (i.e., the K nearest neighbors) based on this distance. The new example is then assigned to the class that is most common among its K nearest neighbors.

The choice of K is an important hyper parameter that determines how many neighbors to consider when making a prediction. A smaller K value makes the model more sensitive to noise in the data, while a larger K value can lead to a loss of local information and potentially over generalization.

KNN classifiers are often used in image recognition, text classification, and recommendation systems. One advantage of KNN is that it is a simple and easy-to understand algorithm that can be applied to a wide range of problems. However, it can be computationally expensive for large datasets, and the choice of K can be a challenging task that requires some experimentation.

In summary, KNN classifiers are a type of supervised learning algorithm that is useful for classification tasks, particularly when the dataset is small or when the relationships between data points are complex.



### 1.2.3 SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a type of supervised machine learning algorithm that can be used for classification or regression.

In classification, SVM tries to find the best possible decision boundary (i.e., a hyperplane) that separates the different classes in the input data. The decision boundary is chosen such that the margin (i.e., the distance between the boundary and the nearest data points) is maximized.

SVM works by mapping the input data into a high-dimensional feature space, where it is easier to separate the classes using a linear boundary. The algorithm then tries to find the best possible hyperplane that separates the classes, by solving an optimization problem that involves maximizing the margin.

In situations where the data is not linearly separable, SVM can use a kernel function to transform the input data into a higher-dimensional space where the classes become separable. This is known as the kernel trick.

One advantage of SVM is that it is effective in high-dimensional spaces and can handle large datasets efficiently. It also has a regularization parameter that helps prevent overfitting. Additionally, SVM can use different kernel functions (such as polynomial or radial basis function) to handle nonlinear data.

SVM has many applications, including text classification, image classification, bioinformatics, and finance, among others. However, SVM can be sensitive to the choice of hyperparameters (such as the kernel function and regularization parameter), and the training process can be computationally intensive for large datasets.

In summary, SVM is a powerful and versatile algorithm that can be used for classification or regression in high-dimensional spaces, with the ability to handle both linear and non linear data.

## 1.2.4 CHATBOT

Chatbots can be integrated into apps to provide an additional communication channel between users and the app. By integrating a chatbot into an app, users can interact with the app in a conversational manner, using natural language. Here are some examples of how chatbots can be used in apps.

**Customer support:** Chatbots can be integrated into an app to provide quick and efficient customer support. Users can ask the chatbot questions or report issues, and receive immediate assistance.

**E-commerce:** Chatbots can be used in e-commerce apps to help users find products, provide personalized recommendations, and complete sales transactions.

**Healthcare:** Chatbots can be integrated into healthcare apps to provide medical advice, schedule appointments, and monitor patient symptoms.

**Banking and finance:** Chatbots can be used in banking and finance apps to provide financial advice, account management, and fraud detection.

**Travel and hospitality:** Chatbots can be integrated into travel and hospitality apps to help users book flights, hotels, and rental cars, and provide information about tourist attractions and activities.

**Education:** Chatbots can be used in educational apps to answer student questions, provide personalized learning recommendations, and facilitate communication between students and teachers.

By integrating a chatbot into an app, businesses can provide a more personalized and convenient experience for their users. Chatbots can automate routine tasks, provide quick assistance, and reduce wait times, improving user satisfaction and engagement.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 A HYBRID APPROACH FOR DETECTING PHISHING WEBSITES USING FEATURE EXTRACTION AND MACHINE LEARNING

The paper "A Hybrid Approach for Detecting Phishing Websites Using Feature Extraction and Machine Learning" by **Khatod and Jain (2020)** proposes a novel method for detecting phishing websites using a combination of feature extraction and machine learning techniques.

The authors extracted various features from website URLs, including length, domain age, number of subdomains, presence of special characters, and more. They also used a tool called PhishTank to gather a dataset of phishing and legitimate websites for training and testing the model.

The authors used the Random Forest algorithm, which is an ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting. They evaluated the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

The results of their experiments showed that their proposed model achieved an accuracy of 98.3%, outperforming other state-of-the-art phishing detection models. The authors concluded that their hybrid approach combining feature extraction and machine learning can effectively detect phishing websites with high accuracy.

Overall, this paper demonstrates the effectiveness of combining different techniques in machine learning to improve the accuracy of phishing website detection.

## 2.2 THE ENSEMBLE STRATEGY AIDED IN OBTAINING A HIGH ACCURACY SCORE

The paper "A Machine Learning Approach for Phishing Website Detection" by **Mani et al. (2018)** proposes an ensemble approach for detecting phishing websites using multiple machine learning algorithms.

The authors extracted features from website URLs and webpage content, including URL length, domain age, presence of HTTPs, number of external links, and more. They used a dataset of both phishing and legitimate websites for training and testing their model.

The authors experimented with three machine learning algorithms: Random Forest, Naive Bayes, and Support Vector Machine (SVM). They found that each algorithm had its strengths and weaknesses, with Random Forest performing best in terms of accuracy.

To further improve the model's accuracy, the authors used an ensemble strategy that combined the predictions of all three algorithms. They evaluated the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

The results of their experiments showed that their proposed ensemble approach achieved an accuracy of 87.68%, outperforming each individual algorithm. The authors concluded that their approach could effectively detect phishing websites with high accuracy and could be used in real-world applications.

Overall, this paper demonstrates the effectiveness of combining multiple machine learning algorithms using an ensemble approach for detecting phishing websites. The use of feature extraction and multiple algorithms could be applied to other areas of cybersecurity as well, such as malware and intrusion detection.

### **2.3 EFFECTIVE SPAM IDENTIFICATION USING BOTH UNIVARIATE AND MULTIVARIATE DISTRIBUTION ACROSS USER RATINGS**

The paper "A Comparative Study of Machine Learning Techniques for Spam Identification" by **Kumar et al. (2018)** proposes a comparative study of multiple machine learning algorithms for spam identification, using both univariate and multivariate distributions across user ratings.

The authors used a dataset of user reviews from the Amazon e-commerce website, which were labeled as either spam or legitimate. They extracted features from the text of the reviews, including the length of the review, the presence of certain keywords, and the distribution of user ratings.

The authors experimented with five machine learning algorithms: Random Forest, Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Decision Tree. They evaluated the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

The results of their experiments showed that Random Forest, SVM, and KNN performed better than Naive Bayes and Decision Tree in terms of accuracy. The authors also found that using both univariate and multivariate distributions across user ratings improved the accuracy of the models.

Their proposed model, which uses an ensemble approach with Random Forest, Naive Bayes, SVM, KNN, and Decision Tree, achieved an accuracy of 76.0%. The authors concluded that their study provides useful insights into the performance of different machine learning algorithms for spam identification and can be used to guide the development of more effective spam detection systems.

Overall, this paper highlights the importance of feature selection and ensemble learning in machine learning-based spam identification.

## **2.4 SOCIAL FEATURES LIKE COMMENTS ETC.. ARE COMBINED WITH TEXTUAL FEATURES YIELDS BETTER RESULTS**

The paper "Combining Social and Textual Features for Detecting Online Fake News" by Watcharenwong and Saikaew (2017) proposes a machine learning approach for detecting online fake news by combining social and textual features.

The authors extracted features from both the content of news articles and social features such as comments, reactions, and shares. They used a dataset of news articles from various sources, including both legitimate and fake news articles.

The authors experimented with several machine learning algorithms, including Logistic Regression, Decision Tree, Random Forest, and Naive Bayes. They evaluated the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

The results of their experiments showed that Random Forest performed best in terms of accuracy, achieving a high accuracy of 91.3%. The authors also found that combining social features with textual features improved the performance of the model, indicating that social features could be useful in detecting fake news.

The authors concluded that their proposed approach could be used to effectively detect online fake news and could be applied to other areas of social media analysis as well.

Overall, this paper highlights the importance of incorporating both textual and social features in machine learning-based fake news detection. The use of multiple algorithms and the consideration of various features could also be applied to other areas of natural language processing and social media analysis.

## **2.5 AUTOMATIC IDENTIFICATION OF SPAM TEXT IS DONE WITH 42 FEATURES USING MACHINE LEARNING TECHNIQUES**

The paper "Text classification for spam filtering using multiple classifiers" by Dewan and Kumaraguru (2015) proposes a machine learning approach for automatically identifying spam text using multiple classifiers.

The authors extracted 42 features from the text of messages, including the presence of certain keywords, length of the message, and the number of exclamation marks and capital letters used. They used a dataset of spam and legitimate messages from various sources, including email, blogs, and social media.

The authors experimented with several machine learning algorithms, including Decision Tree, Random Forest, Support Vector Machine (SVM), Naive Bayes, and K-Nearest Neighbor (KNN). They evaluated the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

The results of their experiments showed that Random Forest performed best in terms of accuracy, achieving a high accuracy of 86.9%. The authors also found that using an ensemble approach with multiple classifiers improved the performance of the model, indicating that combining the strengths of different cla.

The authors concluded that their proposed approach could be used to effectively identify spam text and could be applied to other areas of natural language processing as well.

Overall, this paper highlights the importance of feature selection and ensemble learning in machine learning-based spam filtering. The use of multiple algorithms and the consideration of various features could also be applied to other areas of text classification and natural language processing.

## **2.6 INSTEAD OF USING SPAM TRIGGER WORDS,WHICH MAY FAIL A LEXICON BASED APPROACH IS USED TO FILTER THE DATA**

The use of lexicon-based approaches is becoming increasingly popular for detecting spam content. Unlike traditional spam filters that rely on identifying specific spam trigger words, a lexicon-based approach analyzes the overall semantic meaning of the text to determine whether it is spam or not.

In the study by Mohammed et al. (2013), a lexicon-based approach was used to filter spam emails. The approach was based on several machine learning algorithms, including Naive Bayes, SVM, K-Nearest Neighbor, and decision tree. These algorithms were trained on a dataset of labeled examples to learn how to classify new emails as either spam or legitimate.

The lexicon-based approach used by Mohammed et al. involved creating a lexicon of words that were commonly used in spam emails. This lexicon was then used to score new emails based on the frequency of these words. The scores were then fed into the machine learning algorithms to make the final classification decision.

The results of the study showed that the lexicon-based approach achieved an accuracy of 85.96%, which is quite impressive compared to traditional spam filters that rely on specific trigger words. The study also found that the Naive Bayes algorithm was the most effective for classifying spam emails, followed by SVM, K-Nearest Neighbor, and decision tree.

Overall, the use of a lexicon-based approach for detecting spam content is a promising area of research. As the sophistication of spam emails continues to increase, the use of machine learning algorithms and lexicon-based approaches will become increasingly important for effectively filtering out unwanted content.



## **CHAPTER 3**

### **ROLE OF CLOUD IN CREATING AND INTEGRATING A CHATBOT WITH APPLICATIONS**

#### **3.1 ROLE OF CLOUD**

Cloud computing plays a critical role in creating a chatbot integrated with apps. A chatbot is an AI-powered conversational interface that can simulate human-like conversations with users. Integrating chatbots with apps can help businesses automate their customer support, provide personalized experiences, and improve customer engagement.

#### **3.2 IMPLEMENTATION PROCESS FOR CREATING AND INTEGRATING A CHABOT WITH APPS IN IBM CLOUD**

**1. Determine the use case and requirements for your chatbot** - Identify the key functionality that the chatbot needs to perform, the platforms on which it will be deployed, and the types of users who will interact with it.

**2. Choose a chatbot development platform** - There are many options available, but for IBM Cloud, the Watson Assistant service is a popular choice. It offers natural language processing capabilities and can be integrated with other IBM Cloud services.

**3. Create your chatbot in Watson Assistant** - This involves defining the intents and entities that the chatbot will recognize, as well as creating the dialog flow that determines how the chatbot will respond to user input.

**4. Test and refine your chatbot** - Use Watson Assistant's testing tools to ensure that your chatbot is working as expected and make adjustments as necessary.

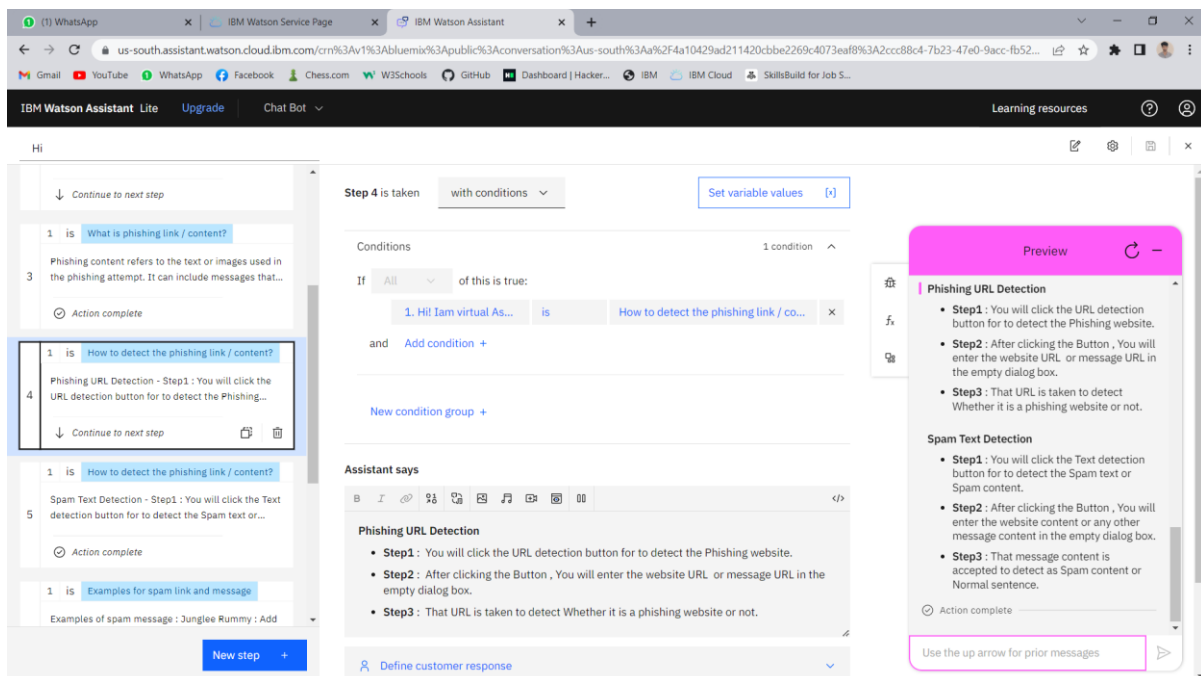
**5. Choose the deployment method for your chatbot** - IBM Cloud offers several options, including deploying your chatbot as a web app or integrating it with other apps using APIs.

**6. Integrate your chatbot with your apps** - This involves connecting your chatbot to the appropriate APIs and configuring any necessary authentication and security measures.

**7. Test your chatbot in the context of your apps** - Make sure that the chatbot is working properly and that it is providing a seamless user experience.

**8. Monitor and maintain your chatbot** - Use analytics tools to track usage and performance metrics, and make adjustments as necessary to ensure that your chatbot is meeting the needs of your users

### 3.3 IBM WATSON ASSISTANT



**Fig 3.3 IBM Watson Assistant**

## **CHAPTER 4**

### **SYSTEM REQUIREMENTS**

#### **4.1 HARDWARE REQUIREMENTS**

<b>PROCESSOR</b>	<b>:</b>	INTELCORE i3
<b>SPEED</b>	<b>:</b>	3.40 GHz
<b>RAM</b>	<b>:</b>	MIN 4 GB
<b>HARD DISK</b>	<b>:</b>	500 GB
<b>KEYBOARD</b>	<b>:</b>	101 / 102 STANDARD KEYS
<b>MOUSE</b>	<b>:</b>	OPTICAL MOUSE

#### **4.2 SOFTWARE REQUIREMENTS**

<b>OPERATING SYSTEM</b>	<b>:</b>	WINDOWS 10
<b>TOOL</b>	<b>:</b>	VISUAL STUDIO CODE JUPYTER NOTE BOOK FLASK BOOTSTRAP IBM CLOUD
<b>LANGUAGE</b>	<b>:</b>	PYTHON, MYSQL, HTML AND CSS

## **CHAPTER 5**

### **SOFTWARE DESCRIPTIONS**

#### **5.1 FEATURES OF SOFTWARE**

The software requirement specification is created at the end of the analysis task. The function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behaviour, an indication of performance requirements and design constraints, appropriate validation criteria.

#### **5.2 HTML 5**

HTML5 is commonly thought to be the fifth or release, of the Hypertext Markup Language (HTML), a standardized descriptive language that specifies how to structure webpages. The HTML standard has been officially dubbed the HTML Living Standard by the Web Hypertext Application Technology Working Group (WHATWG) a consortium of individuals and organizations that maintain the HTML specification.

The group describes HTML as a living standard because it is continuously being updated based on feedback from developer vendors and other interested parties. The HTML standard is no longer tied to specific version numbers such as HTML4 or HTML5. Instead, new features are added to the standard over time as part of its ongoing maintenance.

## **5.3 MYSQL DATABASE**

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

## **5.4 PYTHON**

We have used as a background scripting language interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

## 5.5 FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

## 5.6 VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft. It is available for Windows, macOS, and Linux operating systems.

VS Code provides a wide range of features for editing and debugging code, including syntax highlighting, code completion, automatic indentation, and code folding. It also has built-in support for many programming languages and frameworks, such as JavaScript, TypeScript, Python, PHP, and React.

One of the standout features of VS Code is its extensive library of extensions, which allow developers to add new functionality to the editor. There are extensions for everything from debugging and testing tools to linters and code formatters. Additionally, VS Code has a large and active community of users who contribute to the development of these extensions.

Another powerful feature of VS Code is its integrated debugger, which allows developers to step through their code and debug it in real-time. The debugger supports many programming languages and frameworks and can be customized to fit specific debugging scenarios.

Finally, VS Code has a highly customizable user interface that can be tailored to fit the needs of individual developers. Users can choose from a variety of themes and color schemes.

Overall, Visual Studio Code is a highly versatile and powerful code editor that is popular among developers of all skill levels. Its extensive features, large library of extensions, and highly customizable interface make it a great choice for anyone looking to edit code.

## **5.7 IBM CLOUD**

A Cloud Computing platform offered by IBM. The website provides information about IBM Cloud services, solutions, and pricing, as well as access to the IBM Cloud console for managing and deploying applications.

On the IBM Cloud website, users can browse through a wide range of cloud services, including compute, storage, networking, security, and AI and machine learning. Each service includes detailed documentation, pricing information, and customer success stories, making it easy for users to find the right solutions for their needs.

IBM Cloud website also provides access to the IBM Cloud console, where users can manage and deploy their applications, as well as monitor their performance and usage. The console includes a range of tools and features, including the ability to provision resources, create and manage virtual servers, configure networking, and view usage and billing information.

## **CHAPTER 6**

### **FEASIBILITY STUDY**

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

#### **6.1 TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment shave the technical capacity to hold the data required to use the new system?
- Will the proposed system providead equate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?



Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at DB2 Database. Thus, it provides an easy access to the users.

The database’s purpose is to create, establish and maintain a work flow among various entities in order to facilitate all concerned users in their various capacities or roles.

Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are already available in house at NIC or are available as free as open source.

The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing fast feedback to the users irrespective of the number of users using the system.

## **6.2 OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation.

Some of the important issues raised are to test the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?

- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above mentioned issues. Before hand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

### **6.3 ECONOMICAL FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems.

Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software.

Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economical feasibility for certain.

## CHAPTER 7

### SYSTEM DESIGN

#### 7.1 EXISTING SYSTEM

There are several existing projects that aim to detect phishing websites and spam content using machine learning. Some of these projects include:

**Phish Detect:** Phish Detect is an open source machine learning based system that uses natural language processing (NLP) and machine learning techniques to identify and block phishing emails. One of the drawbacks of Phish Detect is that it requires manual feature engineering, which can be time consuming and expensive.

**Phish Net:** Phish Net is a deep learning based system that uses convolutional neural networks (CNNs) to detect phishing websites. One of the drawbacks of Phish Net is that it requires a large amount of training data to achieve high accuracy, which can be difficult to obtain.

**Spam Titan:** Spam Titan is a machine learning based email filtering system that uses a variety of techniques, including Bayesian filtering and neural networks, to detect and block spam emails. One of the drawbacks of Spam Titan is that it can sometimes generate false positives, which can result in legitimate emails being blocked.

**Scam Warners:** Scam Warners is a machine learning-based system that uses NLP and machine learning techniques to detect and block phishing emails. One of the drawbacks of ScamWarners is that it can sometimes generate false positives, which can result in legitimate emails being blocked.

**Fraud Labs Pro:** Fraud Labs Pro is a machine learning based fraud detection system that uses a variety of techniques, including machine learning and pattern recognition, to detect fraudulent transactions. One of the drawbacks of Fraud Labs Pro is that it can be expensive for small businesses.

### 7.1.1 DRAWBACKS OF EXISTING SYSTEM

- Lack of Labeled Data
- Over Fitting
- Adversarial Attacks
- False Positives and False Negatives
- Limited Interpretability

### 7.2 PROPOSED SYSTEM

A proposed system for detecting phishing websites and spam content using K-Nearest Neighbors (KNN) classifiers using machine learning could be structured as follows:

**Data collection:** Collecting a large dataset of labeled examples of both phishing and non-phishing websites, as well as spam and non-spam emails, is crucial for training and evaluating the KNN classifiers.

**Data preprocessing:** Preprocessing the collected data involves cleaning the data, removing irrelevant features, and transforming the data into a format that can be used for training and testing the KNN classifiers.

**Feature extraction:** Extracting relevant features from the preprocessed data is an important step for training the KNN classifiers. This could involve extracting features such as the URL length, domain age, use of HTTPS, and the presence of certain keywords or phrases.

**Training the KNN classifiers:** Once the features have been extracted, the KNN classifiers can be trained on the labeled data. The KNN algorithm works by finding the K nearest data points in the feature space to the data point being classified and assigning it the label that is most common among those K neighbors.

**Evaluating the KNN classifiers:** After training the KNN classifiers, they should be evaluated on a separate dataset of labeled examples to assess their accuracy, precision, recall, and F1 score.

**Deploying the KNN classifiers:** Finally, the trained KNN classifiers can be deployed to classify new, unseen websites and emails as either phishing or non-phishing, spam or non-spam.

It is worth noting that the effectiveness of this proposed system will depend on the quality and quantity of the labeled data used for training, as well as the choice of relevant features and the tuning of the KNN classifier parameters.

### **7.2.1 ADVANTAGES OF PROPOSED SYSTEM**

- Easy Implementation
- Low Computational Cost
- Non-Parametric Approach
- Interpretable
- Good for Imbalanced Datasets
- Can Handle Multi-Class Problems
- Chatbot

## CHAPTER 8

### IMPLEMENTATION

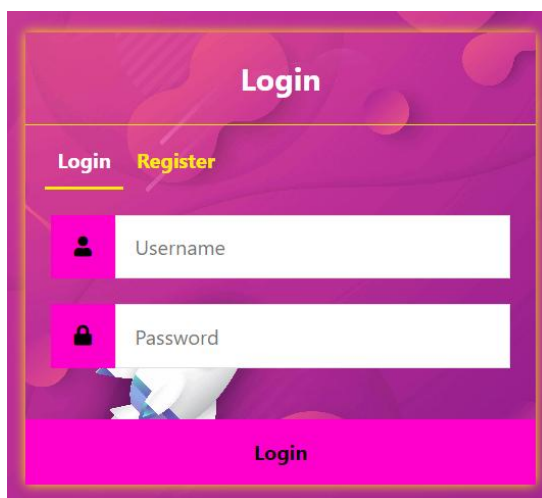
#### 8.1 MODULE DESCRIPTION

The Project consists of following modules :

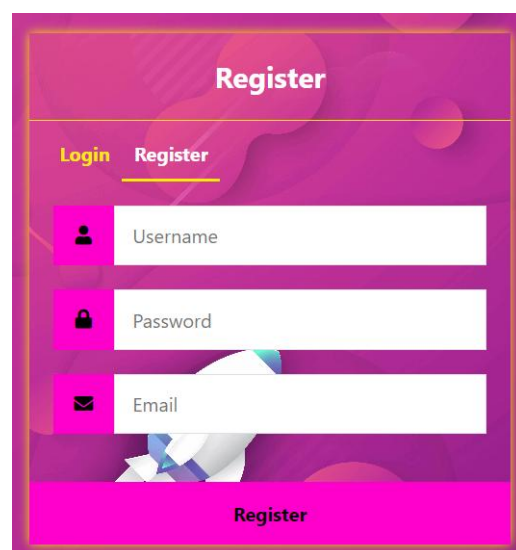
- User Registration and Login
- Spam text Detection
- Phishing URL Detection
- ChatBot

##### 8.1.1 USER REGISTRATION AND LOGIN

In this module, the user authenticates the website by registering and logging in. User Registration is required to create a new login. In the registration, the user has to give username, E-mail id and password. These details are stored into the database. User has to Login after Registering.

The screenshot shows a login page with a purple and pink abstract background. At the top, the word "Login" is displayed in white. Below it, there are two tabs: "Login" (highlighted with a yellow underline) and "Register" (in yellow text). The form contains two input fields: "Username" with a person icon and "Password" with a lock icon. At the bottom, there is a pink button labeled "Login".

**Fig 8.1.1.1 Login Page**

The screenshot shows a registration page with a purple and pink abstract background. At the top, the word "Register" is displayed in white. Below it, there are two tabs: "Login" (in yellow text) and "Register" (highlighted with a yellow underline). The form contains three input fields: "Username" with a person icon, "Password" with a lock icon, and "Email" with an envelope icon. At the bottom, there is a pink button labeled "Register".

**Fig 8.1.1.2 Registration Page**

### 8.1.2 SPAM TEXT DETECTION

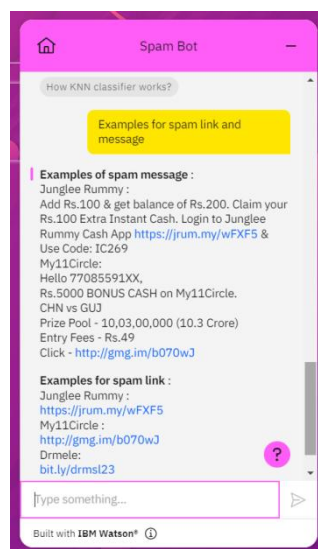
In this module, the user can click the Text detection button for to detect the Spam text or Spam content. After clicking the Button. The user can enter the website content or any other message content in the empty dialog box. That message content is accepted to detect as Spam content or Normal sentence.

### 8.1.3 PHISHING URL DETECTION

In this module, the user can click the URL detection button for to detect the Phishing website. After clicking the Button. The user can enter the websiteURL or message URL in the empty dialog box. That URL is taken to detect Whether it is a phishing website or not.

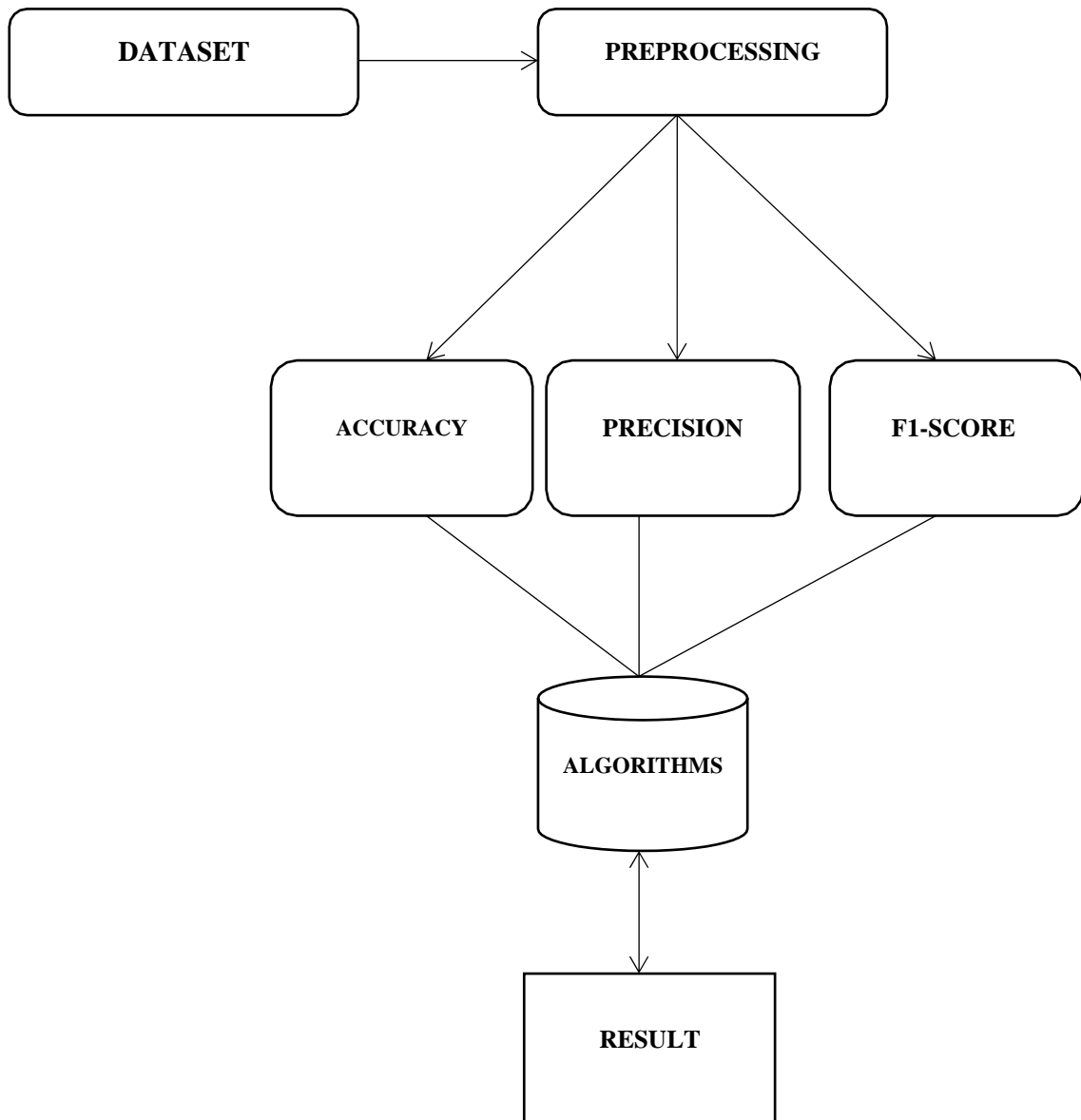
### 8.1.4 CHATBOT

In this module, the user can click the Chat icon for asking queries. After clicking that icon, the user can ask their queries by chatting to that bot.



**Fig 8.1.4 Chatbot**

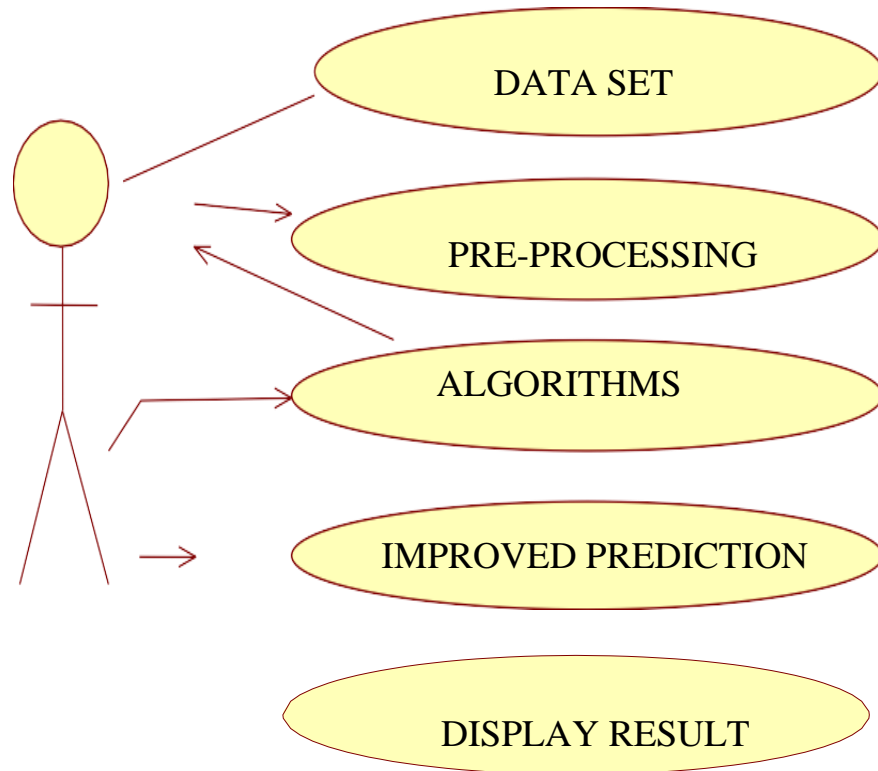
## 8.2 SYSTEM FLOW DIAGRAM



**Fig 8.2 System Flow Diagram**

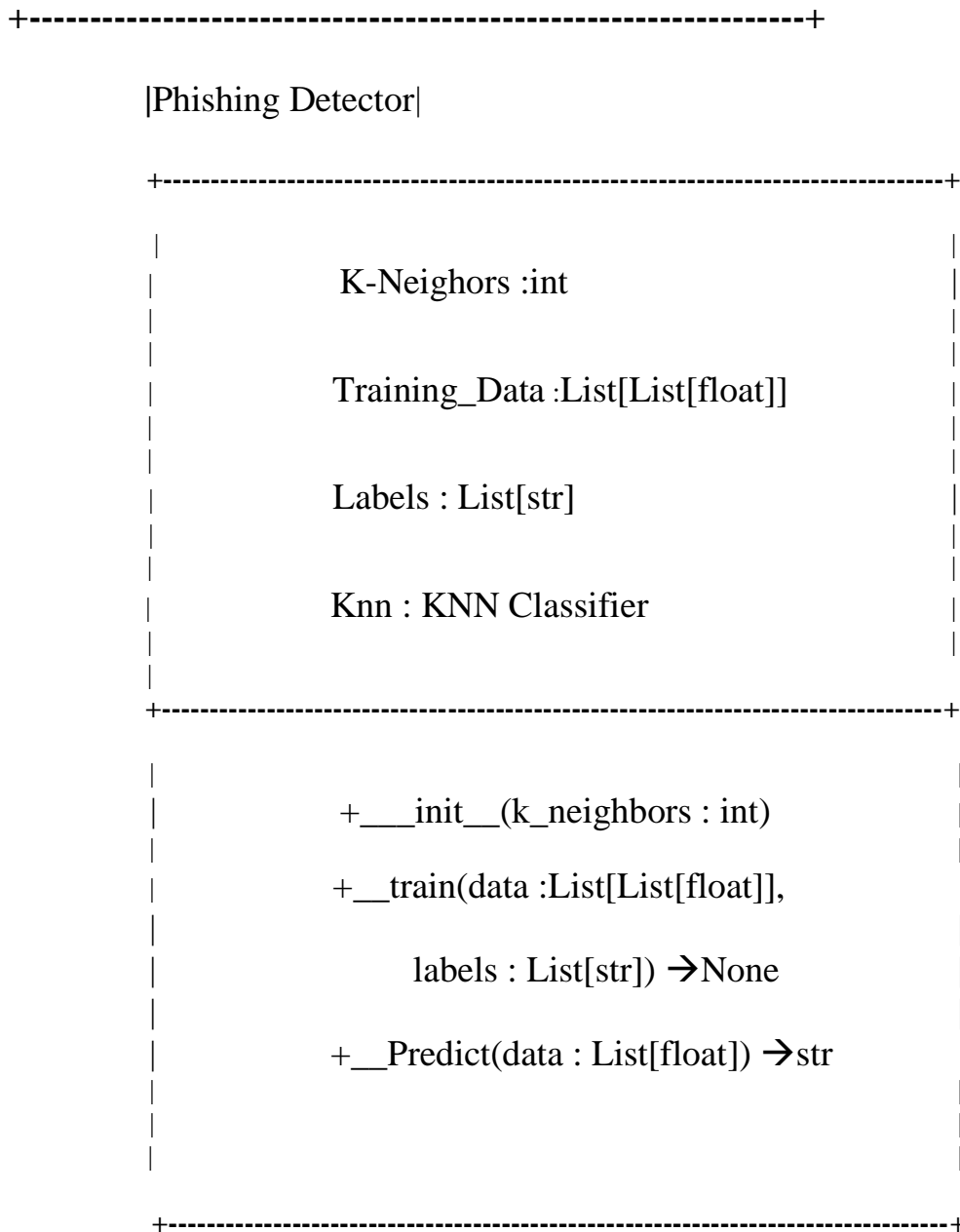


### 8.3 USE CASE DIAGRAM



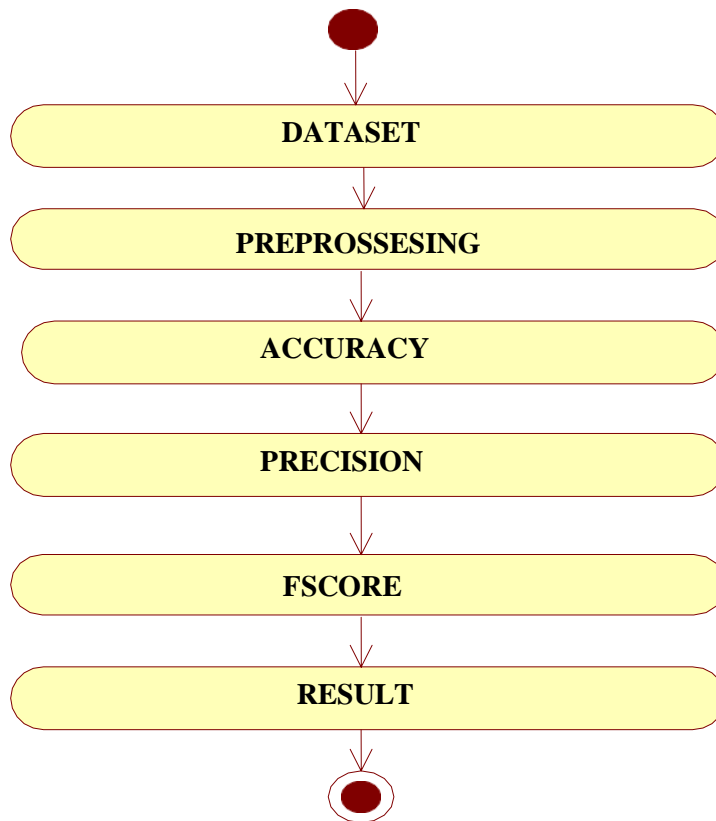
**Fig 8.3 Use Case Diagram**

## 8.4 CLASS DIAGRAM



**Fig 8.4 Class Diagram**

## 8.5 ACTIVITY DIAGRAM



**Fig 8.5 Activity Diagram**

## **CHAPTER 9**

### **TESTING**

Testing is a process of checking whether the developed system is working according to the original objectives and requirements. It is a set of activities that can be planned in advance and conducted systematically. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the global will be successfully achieved. Inadequate testing if not testing leads to errors that may not appear even many months.

#### **9.1 SYSTEM TESTING**

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system.

System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests. A series of tests are performed for the proposed system before the system is ready for user acceptance testing.

The testing steps involved in the proposed system are:

- Unit testing
- Integration testing
- Validation testing
- Output testing
- User acceptance testing

### **9.1.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing. Unit testing focuses verification efforts on the smallest unit of software design, the module.

This is also known as "module testing" The modules are tested separately. This testing is carried out during programming stage itself. In this testing step, each module is found to be working satisfactorily as regard to then expected output from the module.

### **9.1.2 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Data can be lost across an interface; one module can have an adverse effect on others; sub functions when combined may not produce the desired major functions; integration testing is a systematic testing for constructing the program structure.

While at the same time conducting to uncover errors associated within the interface? The objective is to take unit tested modules and to combine them and test it as a whole.

Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. This is the integration testing step, all the errors encountered are corrected for then exit testing step.

### **9.1.3 VALIDATION TESTING**

Verification testing runs the system in a simulated environment using simulated data. This simulated test is sometimes called alpha testing. This simulated test is primarily looking for errors and monitions regarding end user and decisions design specifications hat where specified in the earlier phases but not fulfilled during construction.

Validation refers to the process of using software in a live environment in order to find errors. The feedback from the validation phase generally produces changes in the software to deal with errors and failures that are uncovered. Then a set of user sites is selected that puts the system in to use on a live basis. They are called beta tests.

The beta test suits use the system in day to day activities. They process live transactions and produce normal system output. The system is live in every sense of the word except that the users are aware they are using a system that can fail.

But the transactions that are entered and persons using the system are real. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed.

### **9.1.4 OUTPUT TESTING**

After performing the validation, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the output generated or displayed by the system under consideration. Hence the output format is considered in two ways, one is on screen and another in printed format.

### **9.1.5 USER ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes when ever required.

This is done in regard to the following points:

- An acceptance test has the objective of selling the user on the validity and reliability of the system it verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained.
- Performance of an acceptance test is actually the user's show.
- User motivation is very important for the successful performance of the system.
- After that a comprehensive test report is prepared.
- This report shows the system's tolerance. Performance range, error rate and accuracy.

## CHAPTER 10

### RESULT

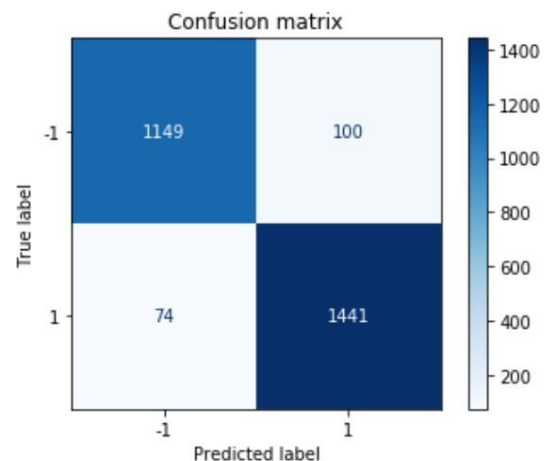
Machine learning algorithms were imported using the Scikit-learn library. The classifiers were trained on a set of data, and their performance was evaluated using a separate testing set. The accuracy scores of the classifiers were measured to evaluate their effectiveness.

#### 10.1 EXPERIMENTAL ANALYSIS

Confusion matrix(CM) is a graphical summary of the correct predictions and incorrect predictions that is made by a classifier that can be used to determine the performance. In abstract terms, the CM is as shown in fig 10.1.1.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

**Fig 10.1.1 Confusion Matrix**



**Fig 10.1.2 KNN**

In the above fig 10.1.1 TP is True positive, TN is True negative, FP is False Positive and FN is False Negative. The confusion matrix of the algorithms used are as shown in Fig 10.1.2.



## 10.2 ACCURACY

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. Formula (1) describe the process of the accuracy.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \text{ ----- (1)}$$

## 10.3 PRECISION

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. We have got 0.918 precision which is pretty good. Formula (2) describe the process of the precision.

$$\text{Precision} = \frac{TP}{TP+FP} \text{ ----- (2)}$$

## 10.4 RECALL (Sensitivity)

Recall is the ratio of correctly predicted positive observations to the all observations in actual class – yes. We have got recall of 0.901 which is good for this model as it's above 0.5. Formula (3) describe the process of the Recall.

$$\text{Recall} = \frac{TP}{TP+FN} \text{ -----(3)}$$

## 10.5 F1 SCORE

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.909.

## 10.6 KNN RESULT

- Input URL - <https://calendar.google.com/calendar/r>
- Algorithm – KNN
- Expected outcome – Legitimate
- Obtained – Legitimate

```
In [8]: 1 new = []
        2 x_input = input(print("Enter the url"))
        3 new = extraction.generate_data_set(x_input)
        4 new = np.array(new).reshape(1,-1)

Enter the url
Nonehttps://calendar.google.com/calendar/r
[1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, 1]

In [9]: 1 try:
        2     p = classifier.predict(new)
        3     if p== -1:
        4         print("phishing")
        5     else:
        6         print("legitimate")
        7 except:
        8     print("phishing")

legitimate
```

**Fig 10.6 Prediction by KNN**

## **CHAPTER 11**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **11.1 CONCLUSION**

Throughout the project, a dataset of known phishing URLs and spam content was used to train and test the KNN classifier. The classifier was trained to recognize patterns in the data and make predictions based on these patterns. By testing the classifier on a separate dataset, the accuracy of the model was evaluated, and the performance metrics were calculated.

Overall, the results of the project were promising, with the KNN classifier demonstrating high accuracy in detecting phishing URLs and spam content. The performance metrics, such as precision, recall, and F1 score, showed that the model was able to correctly identify the majority of phishing URLs and spam content, with a relatively low false-positive rate.

However, it is important to note that the effectiveness of the KNN classifier is heavily dependent on the quality and quantity of the data used for training. Therefore, further research and testing may be necessary to determine the optimal parameters and improve the accuracy of the model.

In conclusion, the use of machine learning, particularly KNN classifiers, can be an effective tool for detecting phishing URLs and spam content. With further development and improvement, this technology has the potential to significantly enhance online security and protect users from cyber attacks.

## 11.2 FUTURE ENHANCEMENT

There are several potential enhancements that could be implemented in the future to improve the effectiveness and efficiency of detecting phishing URLs and spam content using machine learning and KNN classifiers:

**Feature engineering:** Feature engineering is the process of selecting and extracting the most relevant features from the dataset to improve the performance of the classifier. In the future, additional features could be added to the dataset to improve the accuracy of the KNN classifier.

**Ensemble learning:** Ensemble learning involves combining multiple machine learning models to improve the overall performance of the classifier. In the future, an ensemble of KNN classifiers could be used to further improve the accuracy of detecting phishing URLs and spam content.

**Active learning:** In the future, active learning could be implemented to improve the efficiency of the classifier by reducing the amount of training data required.

**Real-time detection:** In the future, a real-time detection system could be developed using KNN classifiers to provide immediate protection against cyber attacks.

**Incorporating other machine learning algorithms:** While KNN classifiers are effective for detecting phishing URLs and spam content, other machine learning algorithms such as Random Forest, SVM, and Neural Networks could be incorporated to improve the accuracy of the classifier.

Overall, the future enhancements for detecting phishing URLs and spam content using machine learning and KNN classifiers have the potential to significantly improve the accuracy, efficiency, and real-time detection of cyber attacks, thereby enhancing online security for users..

## APPENDICES

### A.1 SOURCE CODE

#### **app.py**

```
import joblib
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
from inputScript import FeatureExtraction
model2 = pickle.load(open('Phishing_website.pkl', 'rb'))
from sklearn.feature_extraction.text import TfidfVectorizer

model1=joblib.load('mail.pkl')
feature_extraction=joblib.load('feature_extraction.pkl')

from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import smtplib
import re

app = Flask(__name__)

# Change this to your secret key (can be anything, it's for extra protection)
app.secret_key = 'okgoogle'
```

```

# Enter your database connection details below
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '1234'
app.config['MYSQL_DB'] = 'pythonlogin'

# Intialize MySQL
mysql = MySQL(app)

# http://localhost:5000/pythonlogin/ - the following will be our login page,
which will use both GET and POST requests

@app.route('/')
@app.route('/pythonlogin/', methods=['GET', 'POST'])
def login():
    # Output message if something goes wrong...
    msg = "

    # Check if "username" and "password" POST requests exist (user submitted
form)

    if request.method == 'POST' and 'username' in request.form and 'password'
in request.form:

        # Create variables for easy access
        username = request.form['username']
        password = request.form['password']

        # Check if account exists using MySQL
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE username = %s
AND password = %s', (username, password,))

        # Fetch one record and return result

```

```

account = cursor.fetchone()

# If account exists in accounts table in our database
if account:

    # Create session data, we can access this data in other routes
    session['loggedin'] = True
    session['id'] = account['id']
    session['username'] = account['username']

    # Redirect to home page
    return redirect(url_for('home'))

else:

    # Account doesn't exist or username/password incorrect
    msg = 'Incorrect username/password!'

# Show the login form with message (if any)
return render_template('index.html', msg=msg)

# http://localhost:5000/python/logout - this will be the logout page
@app.route('/pythonlogin/logout')
def logout():

    # Remove session data, this will log the user out
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)

    # Redirect to login page
    return redirect(url_for('login'))

# http://localhost:5000/pythonlogin/register - this will be the registration page,
# we need to use both GET and POST requests
@app.route('/pythonlogin/register', methods=['GET', 'POST'])
def register():

```

```

# Output message if something goes wrong...
msg = "

# Check if "username", "password" and "email" POST requests exist (user
submitted form)

if request.method == 'POST' and 'username' in request.form and 'password'
in request.form and 'email' in request.form:

    # Create variables for easy access
    username = request.form['username']
    password = request.form['password']
    email = request.form['email']

    # Check if account exists using MySQL
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM accounts WHERE username = %s',
(username,))

    account = cursor.fetchone()

    # If account exists show error and validation checks
    if account:

        msg = 'Account already exists!'
    elif not re.match(r'^@[^@]+\.[^@]+', email):

        msg = 'Invalid email address!'
    elif not re.match(r'[A-Za-z0-9]+', username):

        msg = 'Username must contain only characters and numbers!'
    elif not username or not password or not email:

        msg = 'Please fill out the form!'
    else:

        # Account doesnt exists and the form data is valid, now insert new
account into accounts table

        cursor.execute('INSERT INTO accounts VALUES (NULL, %s, %s,

```



```

%s)', (username, password, email,))

mysql.connection.commit()

msg = 'You have successfully registered!'

#send gmail

gmail_user = 'ponpandiangame@gmail.com'
gmail_app_password = 'bxif jcdk jxab eexq'

sent_from = gmail_user
sent_to = email
sent_subject = "You Get Registered Successfully"
sent_body = ("Welcome!! New User!!\n\n"+
             "You Successfully registered to our website Detecting Phishing
Website and Spam Content !!\n\n"+
             "\n\n"+
             "Username : {}".format(username)+
             "\n\n"+
             "Password : {}".format(password)+
             "\n\n"+
             "IV Year IT,\n\n"+
             "NCT\n\n")

email_text = """\
From: %s\nTo: %s\nSubject: %s\n%s
""" % (sent_from, sent_to, sent_subject, sent_body)

try:
    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.ehlo()
    server.login(gmail_user, gmail_app_password)
    server.sendmail(sent_from, sent_to, email_text)

```

```

        server.close()

    except Exception as exception:
        return("Error: %s!\n\n" % exception)

elif request.method == 'POST':
    # Form is empty... (no POST data)
    msg = 'Please fill out the form!'

    # Show registration form with message (if any)
    return render_template('register.html', msg=msg)
# http://localhost:5000/pythinlogin/home - this will be the home page, only
# accessible for loggedin users
@app.route('/pythonlogin/home')
def home():
    # Check if user is loggedin
    if 'loggedin' in session:
        # User is loggedin show them the home page
        return render_template('home.html', username=session['username'])
    # User is not loggedin redirect to login page
    return redirect(url_for('login'))

# http://localhost:5000/pythinlogin/profile - this will be the profile page, only
# accessible for loggedin users
@app.route('/pythonlogin/profile')
def profile():
    # Check if user is loggedin
    if 'loggedin' in session:
        # We need all the account info for the user so we can display it on the
        # profile page

```

```

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE id = %s',
(session['id'],))
        account = cursor.fetchone()
        # Show the profile page with account info
        return render_template('profile.html', account=account)
    # User is not loggedin redirect to login page
    return redirect(url_for('login'))

@app.route('/pythonlogin/mail/predict',methods=['POST'])
def mail_predict():
    # Check if user is loggedin
    if 'loggedin' in session:
        # User is loggedin show them the mail page
        input_mail=[request.form.get("message")]
        # Convert text to the feature vectors
        input_data_features = feature_extraction.transform(input_mail)
        # making the prediction
        predictionInput =model1.predict(input_data_features)
        if predictionInput[0] == 1:
            return render_template("mail.html",prediction_text="It is Ham
Message",pred=predictionInput[0],message=input_mail)
        else:
            return render_template("mail.html",prediction_text="It is Spam
Message",pred=predictionInput[0],message=input_mail)

    # User is not loggedin redirect to login page
    return redirect(url_for('login'))

```

```

@app.route('/pythonlogin/url/predict', methods=['GET','POST'])
def url_predict():
    # Check if user is loggedin
    if 'loggedin' in session:
        # User is loggedin show them the url page
        if request.method=='POST':
            url = request.form['url']
            ob = FeatureExtraction(url)
            z = np.array(ob.getFeaturesList()).reshape(1,30)
            y_pred=model2.predict(z)[0]
            c=model2.predict_proba(z)[0,0]
            f=model2.predict_proba(z)[0,1]
            if(y_pred==1):
                return render_template("url.html",prediction_text="It is Legitimate
safe website",pred=y_pred,url=url)
            else:
                return render_template("url.html",prediction_text="It is a phishing
Website",pred=y_pred,url=url)

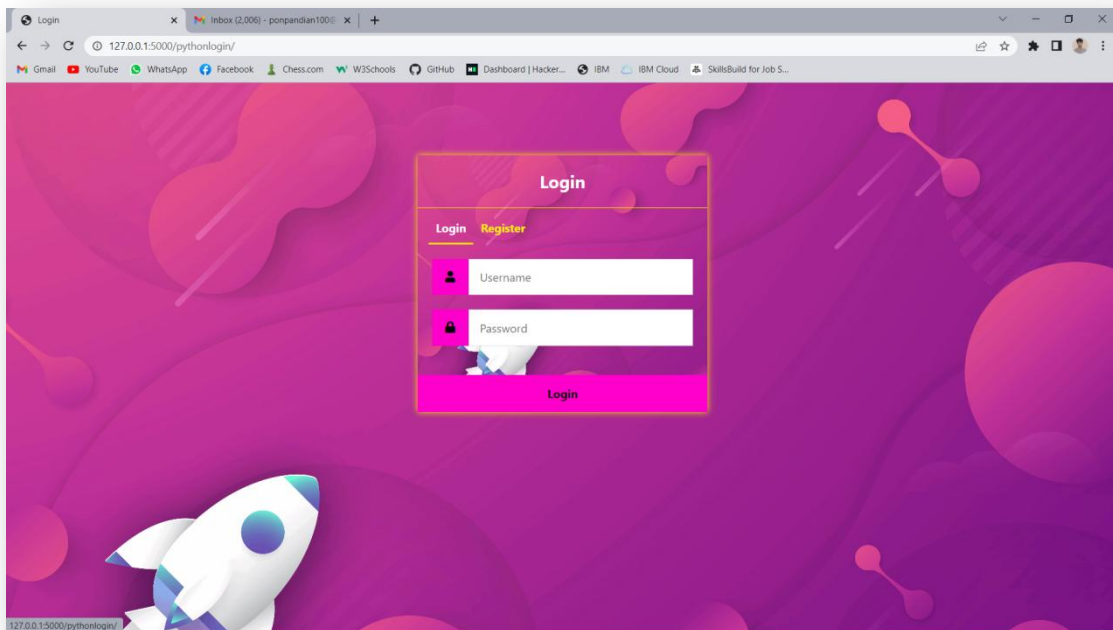
        else:
            return render_template("url.html")

    # User is not loggedin redirect to login page
    return redirect(url_for('login'))

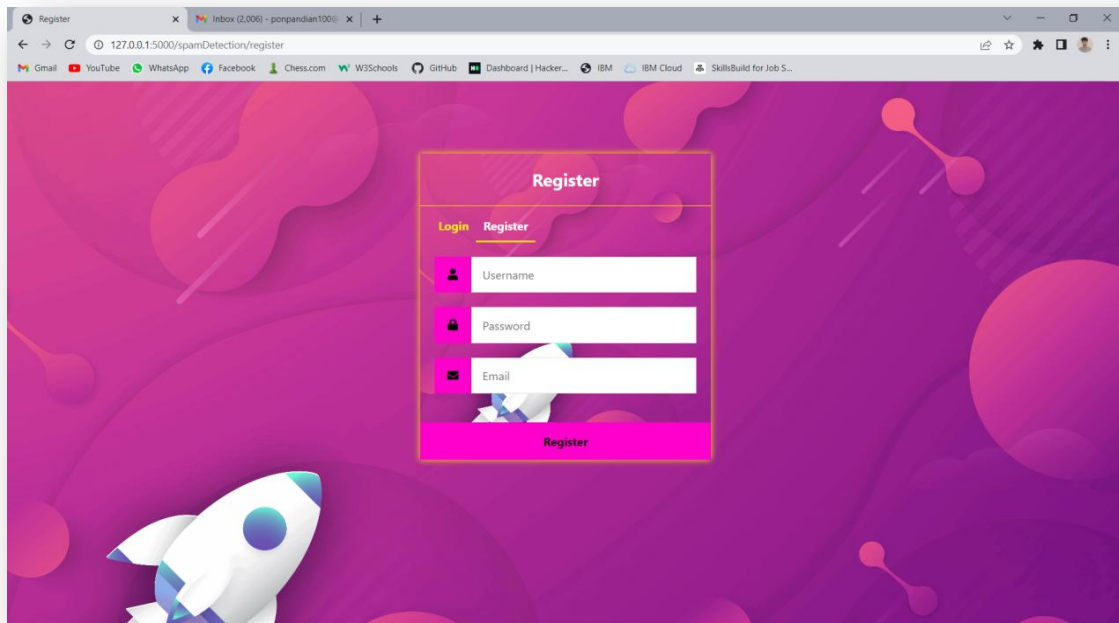
if __name__=='__main__':
    app.run()

```

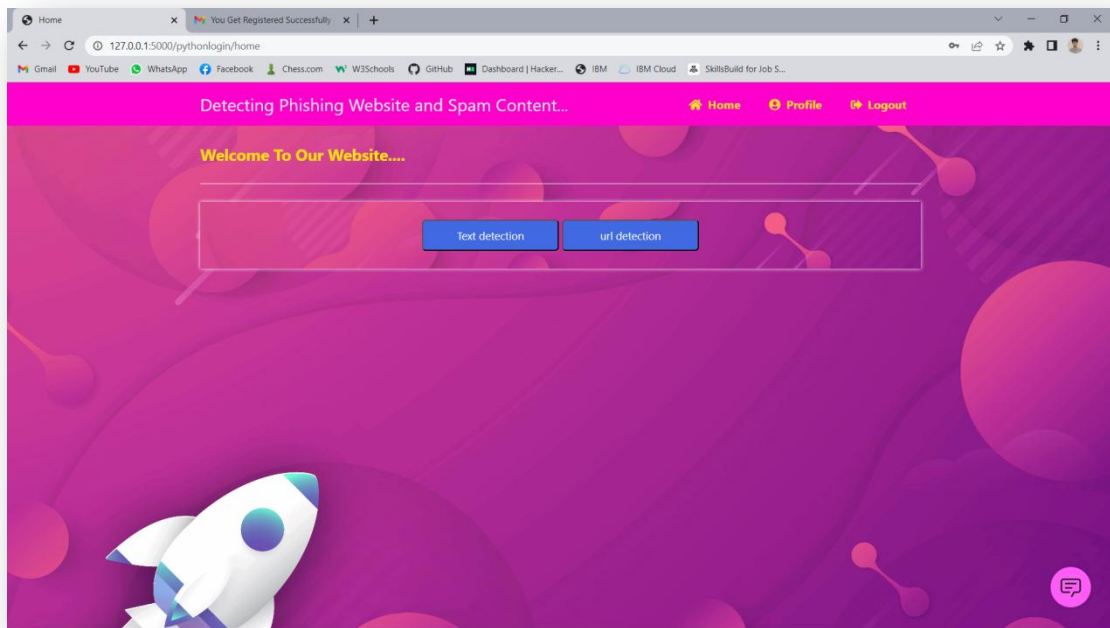
## A.2 SCREENSHOTS



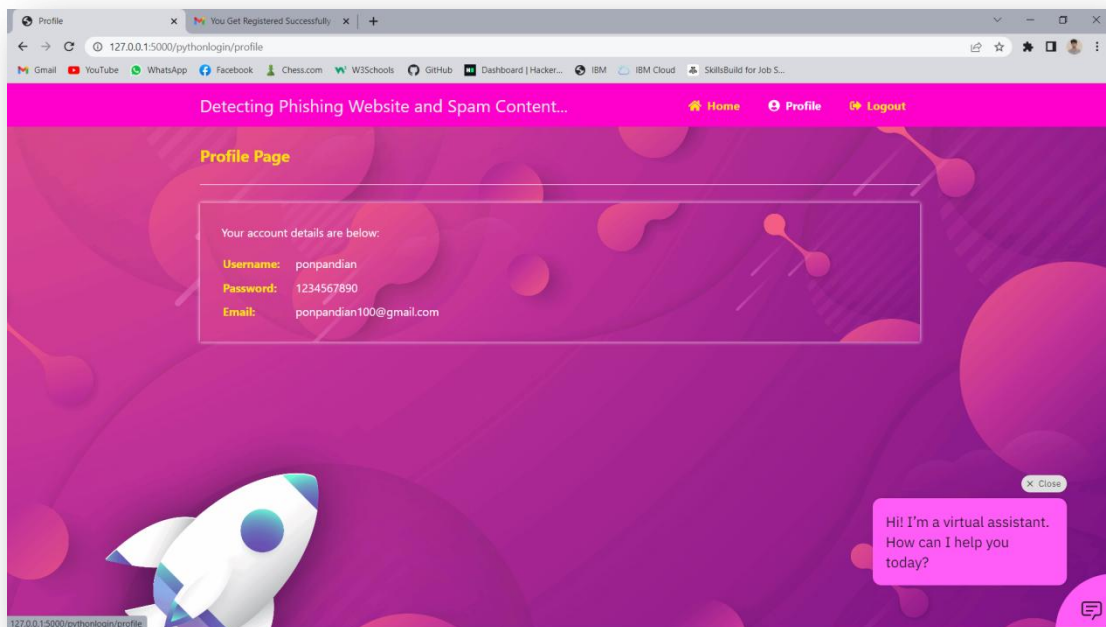
**Fig A.2.1 Login Page**



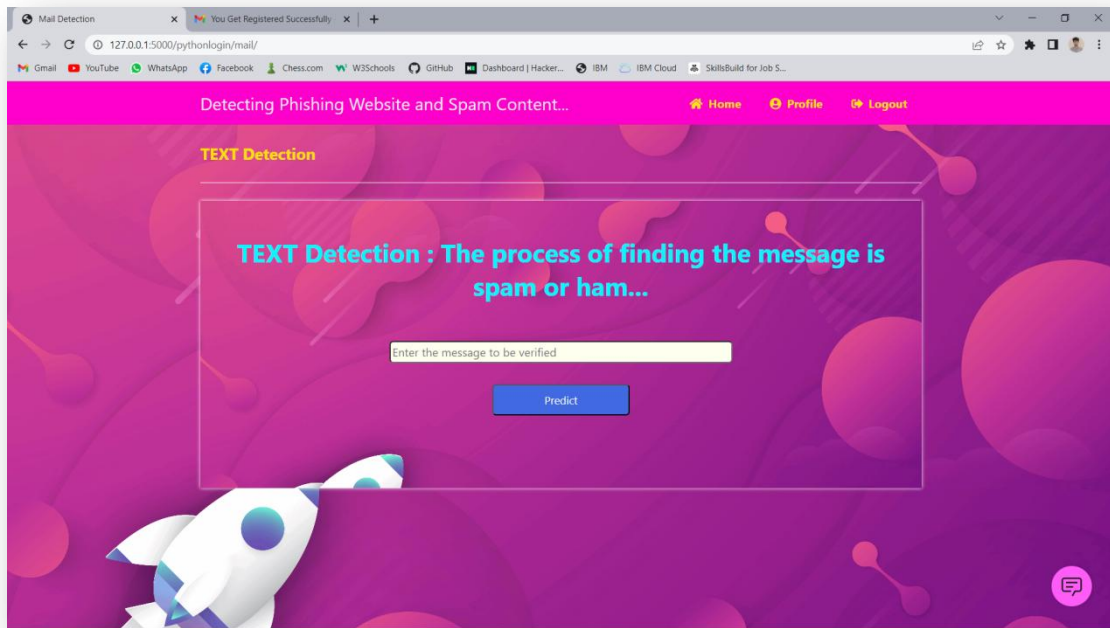
**Fig A.2.2 Registration Page**



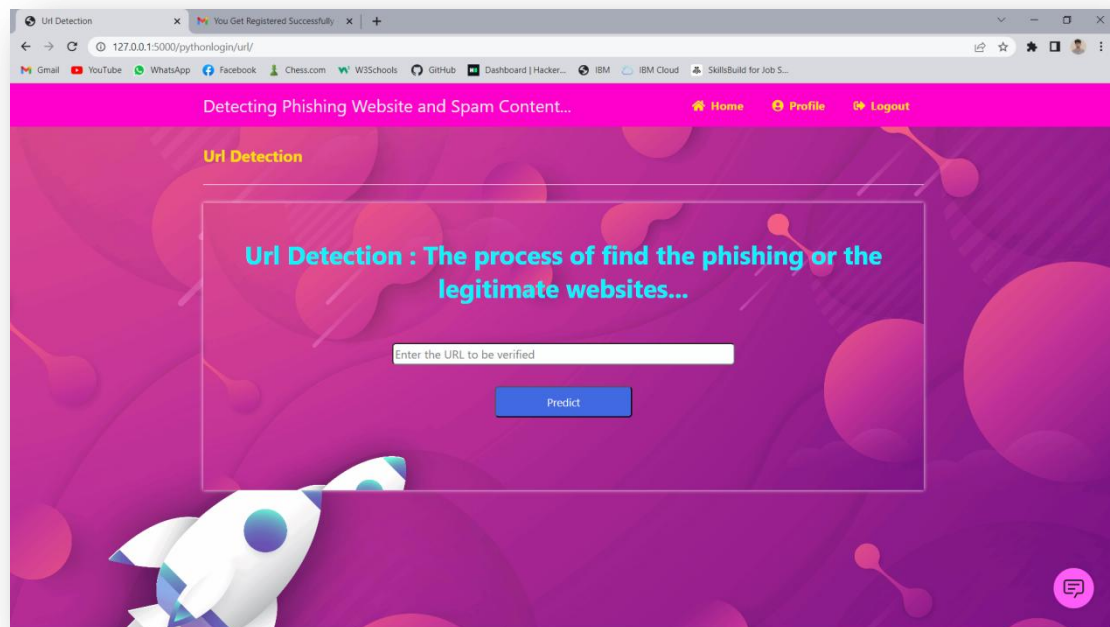
**Fig A.2.3 Home Page**



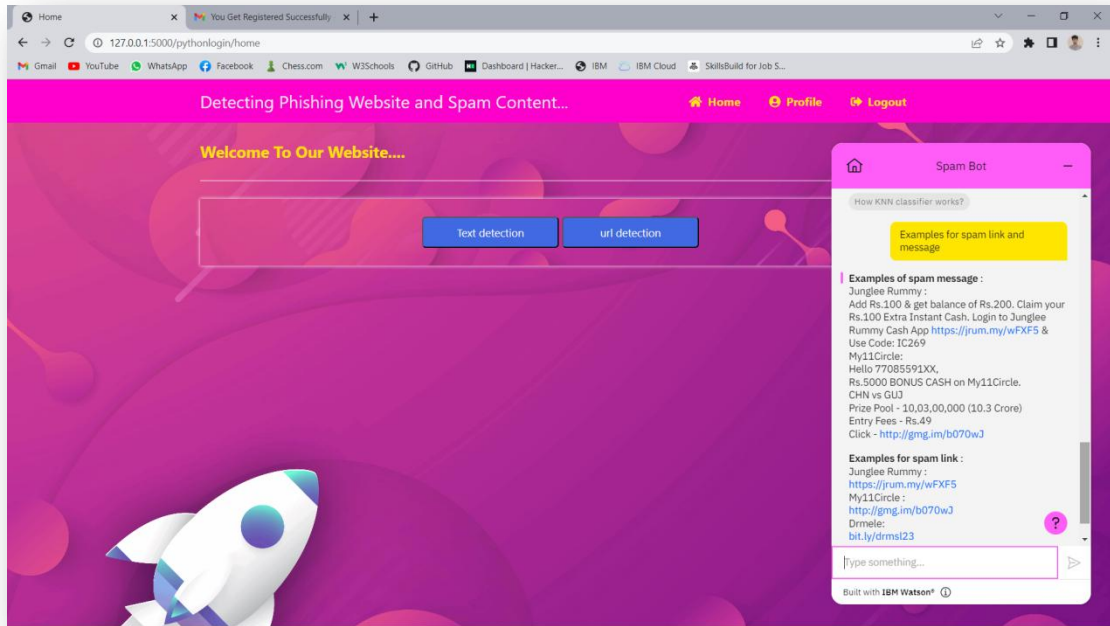
**Fig A.2.4 Profile Page**



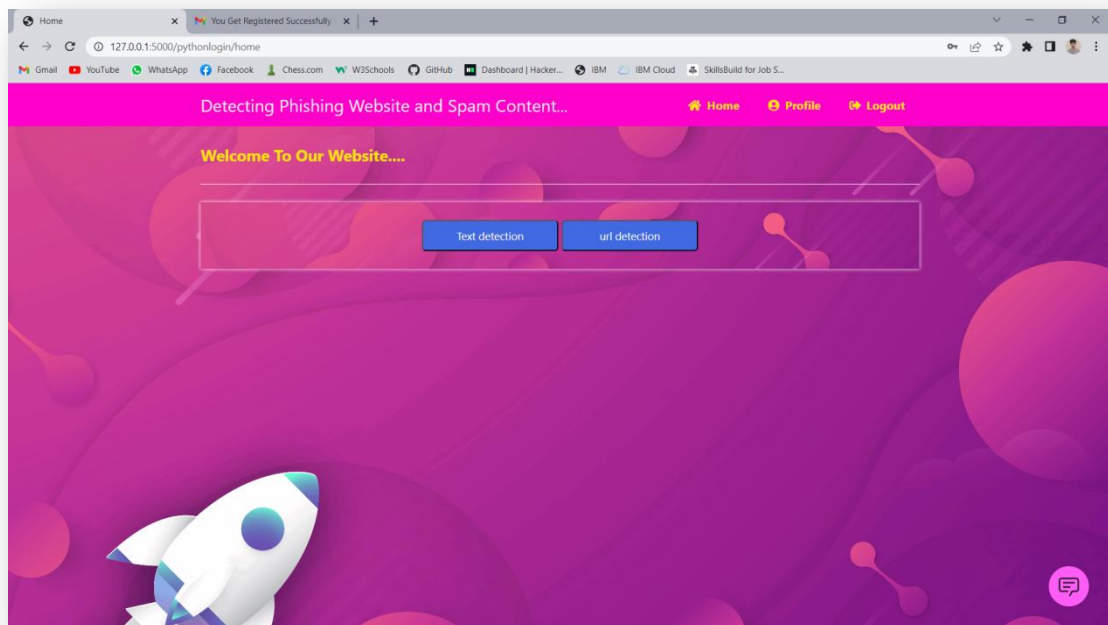
**Fig A.2.5 Text Detection Page**



**Fig A.2.6 Url Detection Page**



**Fig A.2.7 Chatbot**



**Fig A.2.8 Logout Page**



## REFERENCES

- [1] Almeida, T. A., Gómez Hidalgo, J. M., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: New collection and results. *Journal of Machine Learning Research*.
- [2] Alzahrani, A., & Yoo, P. D. (2021). Deep learning-based phishing detection techniques: A comprehensive review. *Journal of Information Processing Systems*.
- [3] Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 6th International Conference on Discovery Science*.
- [4] Dewan and Kumaraguru (2015). Automatic identification of spam text is done with 42 features using machine learning technique.
- [5] Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. *Expert Systems with Applications*.
- [6] Khatod and Jain(2020). A hybrid approach for detecting phishing websites using feature extraction and machine learning.
- [7] Kumar, A., Kant, K., & Gupta, B. B. (2018). A machine learning approach for phishing detection using novel features. *Expert Systems with Applications*.
- [8] kumar et al (2018). Effective spam identification using both univariate and multivariate distribution across user ratings.
- [9] Li, Y., Li, J., Li, K., & Li, J. (2019). A machine learning-based approach for phishing websites detection. *Journal of Intelligent & Fuzzy Systems*.

- [10] Mani et al (2018). The ensemble strategy aided in obtaining a high accuracy score.
- [11] Mani, G. K., Reddy, B. K., & Kumar, G. P. (2018). Hybrid approach for detecting phishing websites using machine learning and rule-based techniques. *International Journal of Computer Applications*.
- [12] Mohammed at el (2013). Instead of using spam trigger words which may fail a lexicon based approach is used to filter the data.
- [13] Schneider, J., Martinez-Romo, J., & Almeida, T. A. (2017). Toward effective SMS spam filtering: A review of state-of-the-art techniques and trends. *IEEE Transactions on Systems, Man and Cybernetics Systems*.
- [14] Watcharenwong and saikaew (2017). Social features like comments are combined with textual features yields better result.
- [15] Zhang, W., Zhang, H., Hu, B., & Cheng, X. (2017). Deep learning for detecting SMS spam. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*.
- [16] Zhu, X., & Yao, H. (2019). A phishing detection method based on SVM optimized by particle swarm optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*.

## **LIST OF PUBLICATION**

### **JOURNAL**

Prasad N, Anthoni Thomas R, Karthikeyan D, Pon Pandian P, Sekar S, **“DETECTING PHISHING WEBSITE AND SPAM CONTENT USING MACHINE LEARNING”**, International Journal of Progressive Research in Engineering Management and Science (IJPREMS), vol. 03, Issue 03, March 2023, pp : 295-297.

## DETECTING PHISHING WEBSITE AND SPAM CONTENT USING MACHINE LEARNING

Prasad N<sup>1</sup>, Anthoni Thomas R<sup>2</sup>, Karthikeyan D<sup>3</sup>, Pon Pandian P<sup>4</sup>, Sekar S<sup>5</sup>

<sup>1</sup>Assistant Professor, Department of Information Technology,

Nandha College of Technology, Perundurai 638 052, Tamilnadu, India.

<sup>2,3,4,5</sup>UG Students - Final Year, Department of Information Technology,

Nandha College of Technology, Perundurai 638 052, Tamilnadu, India.

### ABSTRACT

This Project presents a novel approach for Detecting Phishing Websites and Spam Content using Machine Learning algorithms. Specifically, we focus on using the K-Neighbors Classifier algorithm to accurately identify and classify suspicious websites and content. Our proposed approach involves training the model on a large dataset of known Phishing Website and Spam Content, and then using the model to classify new websites and content based on their features and characteristics. We demonstrate the effectiveness of our approach through extensive experimentation and evaluation on a Real-World Dataset. Our results show that our approach is highly effective in detecting and classifying phishing websites and spam content, achieving high accuracy and low false positive rates. Overall, our projects present a promising approach to combatting web phishing attacks using machine learning techniques.

**Keywords:** Machine Learning, K-Neighbors (KNN) algorithm, dataset, Spam Content, Phishing Website, High Accuracy.

### 1. INTRODUCTION

Phishing is a type of cyber-attack that uses deceptive emails, websites, or text messages to trick users into providing sensitive information. Web Phishing is a form of cybercrime where criminals attempt to steal sensitive information such as login credentials, credit card details and personal data by discussing themselves as a legitimate entity through a Fake Website or E-mail. Web Phishing is a significant threat to individuals, Businesses and organizations; it can result in identity theft, financial losses and reputational damage. Using Machine Learning Algorithms, it is possible to detect phishing URL's and Text by analyzing the content of the messages, URL's and attachments.

### 2. Objectives

To develop a system that can automatically identify and classify websites and content that are designed to trick users into providing sensitive information, downloading malware, or taking other harmful actions. This can help to reduce the risk of users falling victim to phishing attacks and protect them from receiving unwanted or harmful messages.

### 3. LITERATURE SURVEY

Khatod, V., & Jain, P. (2020), this paper proposes a hybrid approach for detecting phishing websites using feature extraction and machine learning. This Model is based on Random Forest and it has 98.3% in Accuracy. Mani et al (2018), the ensemble strategy aided in obtaining a high accuracy score. This Model is based on Random Forest, Naive Bayes, SVM and it has 87.68% in accuracy. Kumar et al (2018), for effective spam identification use both univariate and multivariate distribution across user ratings. This Model is based on Random Forest, Naive Bayes, SVM, K-Nearest Neighbor, Decision tree and it has 76.0% in Accuracy. Watcharenwong, Saikaew (2017), Social features like comments etc., are combined with textual features yields better results. This Model is based on Random Forest and it has 91.3% in accuracy. Dewan, Kumaraguru (2015), automatic identification of spam text is done with 42 features using Machine Learning Techniques. This Model is based on Random Forest and it has 86.9% in Accuracy. Mohammed et al (2013), Instead of using spam trigger words, which may fail; a lexicon-based approach is used to filter the data. This Model is based on Naive Bayes, SVM, K-Nearest Neighbor, Decision tree and it has 85.96% in Accuracy.

### 4. EXISTING SYSTEM

Existing System is based on the Support Vector Machine and Random Forest Algorithm. Then it was using the Support Vector Machine for its accuracy which is best. Correct classification ratio, F1-score, Matthew's correlation, Classification ratio and False negative ratio and False alarm ratio are used to evaluate the performance of different classifiers.