CHAPTER-1

INTRODUCTION

1.1 Overview

Today, Spam has become a major problem in communication over internet. It has been accounted that around 55% of all emails are reported as spam and the number has been growing steadily. Spam which is also known as unsolicited bulk email has led to the increasing use of email as email provides the perfect ways to send the unwanted advertisement or junk newsgroup posting at no cost for the sender. This chances has been extensively exploited by irresponsible organizations and resulting to clutter the mail boxes of millions of people all around the world.

Spam has been a major concern given the offensive content of messages, spam is a waste of time. End user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economical which led some countries to adopt legislation.

Text classification is used to determine the path of incoming mail/message either into inbox or straight to spam folder. It is the process of assigning categories to text according to its content. It is used to organized, structures and categorize text. It can be done either manually or automatically. Machine learning automatically classifies the text in a much faster way than manual technique. Machine learning uses pre-labelled text to learn the different associations between pieces of text and it output. It used feature extraction to transform each text to numerical representation in form of vector which represents the frequency of word in predefined dictionary.

In this project, machine learning techniques are used to detect the spam message of a mail. Machine learning is where computers can learn to do something without the need to explicitly program them for the task. It uses data and produce a program to perform a task such as classification. Compared to knowledge engineering, machine learning techniques require messages that have been successfully pre-classified. The pre-classified messages make the training dataset which will be used to fit the learning algorithm to the model in machine learning studio

A combination of algorithms are used to learn the classification rules from messages. These algorithms are used for classification of objects of different classes. These algorithms are provided with pre labelled data and an unknown text. After learning from the prelabelled data each of these algorithms predict which class the unknown text

may belong to and the category predicted by majority is considered as final.

1.2 Problem Statement

In the digital world, we receive a large number of emails every day, the majority of which are irrelevant to us and some of which include questionable links that may damage our system in one way or another. Spam detection can be used to get around this. It involves identifying if an email is legitimate or whether it is spam of some type. Delivering relevant emails to the individual and separating junk emails are the goals of spam detection. Every email service provider already includes spam detection, but it is not always particularly accurate, occasionally, it labels useful emails as spam. Since spammers began employing sophisticated strategies to get past spam filters, like using random sender addresses or attaching random characters to the start or end of email subjects, the battle between the filtering system and spammers has become intense. Machine learning with a model-oriented approach lacks activity prediction development. Since then, spam has taken up storage space and transmission capacity while wasting users' time by forcing them to sort through junk mail. The rules in other ones that are already in place must be continuously updated and maintained, which makes it burdensome for some users and challenging to manually compare the accuracy of classified data.

Social network logon credentials have become just as desirable as email addresses, as social spam emails are more likely to be opened and believed than traditional communications. Spam and the transmission of malware can coexist. Due to the low cost of sending spam compared to traditional marketing methods and the extremely low response rates to it, spam marketing is still relatively cost-effective. But the victim will pay a high price for it. One spam email can be sent for as little as one thousandth of a penny, but the recipient will pay about ten cents, according to research by Tom Galler, Executive Director of the SpamCon Foundation.

In the computerized world a great deal of messages are received consistently, and a large portion of them are not of any significance to us, some are containing dubious connections which can hurt our framework somehow or another or the other. This can be overwhelmed by utilizing spam location. It is the most common way of characterising whether the email is a certifiable one or on the other hand in the event that it is a spam of some sort or another. The motivation behind spam identification is to convey significant

messages to the individual and separate spam messages. Currently every email specialist organization has spam recognition yet, its exactness isn't excessively a lot, at times they group valuable messages as spam. This project centres around the near examination approach, where different AI models are applied to the equivalent dataset. The different AI models were thought about in light of exactness and Accuracy.

CHAPTER - 2

REQUIREMENT ANALYSIS

2.1 Functional Requirements

- The system should accept email text input from the user.
- It should preprocess the input using NLP techniques (tokenization, stop-word removal, etc.).
- A trained machine learning model should classify the email as spam or not spam (ham).
- The result should be displayed to the user clearly and instantly.
- Users should be able to interact with the system through a simple UI (e.g., textbox and button).

2.2 Non-Functional Requirements

- The system must generate results within 1–2 seconds (real-time performance).
- It should maintain a high accuracy (preferably above 90%).
- The model should be robust with a good precision-recall balance.
- The interface must be user-friendly and easy to navigate.
- The solution should be lightweight and deployable on low-end machines.

2.3 Software Requirements

- Programming Language: Python 3.x
- Libraries:
    - scikit-learn – for model training and prediction
    - NLTK – for text preprocessing
    - Pandas, NumPy – for data handling
    - Streamlit – for building the user interface
- Development Environment: Jupyter Notebook or VS Code

2.4 Hardware Requirements

- Minimum Intel Core i3 processor or equivalent
- Minimum 4 GB RAM
- At least 2 GB free disk space

- No GPU required (CPU processing is sufficient)
- Stable internet connection (for initial setup and dataset download)

2.5 Dataset Requirements

- Use a public dataset like:
    - SMS Spam Collection Dataset
    - Enron Email Dataset
- Dataset should contain labeled entries as "spam" and "ham"
- Preprocessing required to clean and normalize the text
- Feature extraction using Count Vectorizer or TF-IDF

2.6 Preprocessing Requirements

- Convert all text to lowercase
- Remove punctuation, numbers, and special characters
- Eliminate stop words (e.g., "the", "is", "and")
- Apply stemming or lemmatization
- Tokenize the text into individual words
- Convert text into numerical vectors for model input

CHAPTER - 3

LITERATURE REVIEW

This chapter discusses the machine learning literature review classifier that has been used in previous research and projects. The purpose of that is to summarize prior research relevant to this topic rather than to gather information. It entails finding, reading, analyzing, summarizing, and assessing project-based reading materials. The majority of spam filtering and detection systems require periodic training and updating, according to assessments of the literature on machine learning. Setting up rules is also necessary for spam filtering to begin functioning.

3.1 Problem Analysis

Email is a kind of communication that uses telecommunication to exchange computer-stored information. Several groups of people as well as individuals receive the emails. Even though email facilitates the sharing of information, spam and junk mail pose a severe threat. Spam messages are unwanted communications that people get and that annoy them and are inundated in their mailboxes. By wasting their time and causing bandwidth problems for ISPs, it irritates email users. Therefore, it is more crucial to identify and categorise incoming email as spam or junk. Thus, a review of earlier studies presenting email detection and classification algorithms is provided in this section.

3.2 Related Work

Spam classification is a problem that is neither new nor simple. A lot of research has been done and several effective methods have been proposed.

i. M. RAZA, N. D. Jayasinghe, and M. M. A. Muslam have analyzed various techniques for spam classification and concluded that naïve Bayes and support vector machines have higher accuracy than the rest, around 91% consistently [1].

ii. S. Gadde, A. Lakshmanarao, and S. Satyanarayana in their paper on spam detection concluded that the LSTM system resulted in higher accuracy of 98%[2].

iii. P. Sethi, V. Bhandari, and B. Kohli concluded that machine learning algorithms perform differently depending on the presence of different attributes [3].

iv. H. Karamollaoglu, İ. A. Dogru, and M. Dorterler performed spam classification on Turkish messages and emails using both naïve Bayes classification algorithms and

support vector machines and concluded that the accuracies of both models measured around 90% [4].

v. P. Navaney, G. Dubey, and A. Rana compared the efficiency of the SVM, naïve Bayes, and entropy method and the SVM had the highest accuracy (97.5%) compared to the other two models [5].

vi. S. Nandhini and J. Marseline K.S in their paper on the best model for spam detection it is concluded that random forest algorithm beats others in accuracy and KNN in building time [6].

vii. S. O. Olatunji concluded in her paper that while SVM outperforms ELM in terms of accuracy, the ELM beats the SVM in terms of speed [7].

viii. M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta studied classical machine learning classifiers and concluded that convolutional neural network outperforms the classical machine learning methods by a small margin but take more time for classification [8].

3.3 Summary

From various studies, we can take that for various types of data various models performs better. Naïve Bayes, random forest, SVM, logistic regression are some of the most used algorithms in spam detection and classification.

Algorithms such as Naïve Bayes, Random Forest, Support Vector Machine (SVM), and Logistic Regression are among the most widely used in spam detection due to their proven performance in binary classification problems.

Additionally, recent advancements in Natural Language Processing (NLP) and deep learning have introduced models like Recurrent Neural Networks (RNNs) and Transformers, which are capable of understanding the context and semantics of emails better. However, they require more computational resources and larger datasets for training. Ultimately, the choice of algorithm should be driven by the project's goals, dataset characteristics, and available computational resources. For most practical applications, traditional machine learning methods still offer a strong balance between performance, interpretability, and efficiency.

Table 3.1 Tabular Summary

Methods	Advantages	Disadvantages
Feature Selection Approach	Processes of optimization and effective decision-making	Time Consuming and is very Costly
Collaborative Filtering Technique	Effectiveness and efficiency have been established using artificial and actual data	Lacked perspectives for distant, complicated, and uncertain data streams.
Email Abstraction based Scheme	Simple in nature	Easy pray for spammers
Random Forest Technique	The technique uses a set of rules to reduce a series of data and generates a search direction in the dual and primal variables as well as a forecast of the set of active features at each step	developed using simply straightforward programmes
Aprior and K-NN Technique	Good for small data	High time complexity for large data
Support Vector Machine	Robust and accurate method	computational inefficiency
Neural Networks	Clearly describe each spam classifier's true level. high level of accuracy by combining the improvements of various classifiers	Nonstandard classifier because this hybrid system contains multiple layers, it takes time to obtain the desired output.

The table 3.1 shows, various spam detection methods, highlighting their advantages such as accuracy, simplicity, or efficiency, and disadvantages like high computational cost or vulnerability to complex data. Techniques like SVM and Neural Networks offer strong performance but may suffer from inefficiency or complexity.

CHAPTER - 4

PROPOSED SYSTEM

This chapter will explain the specific details on the methodology being used to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do spam detection and filtering. So, it is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

4.1 System Architecture

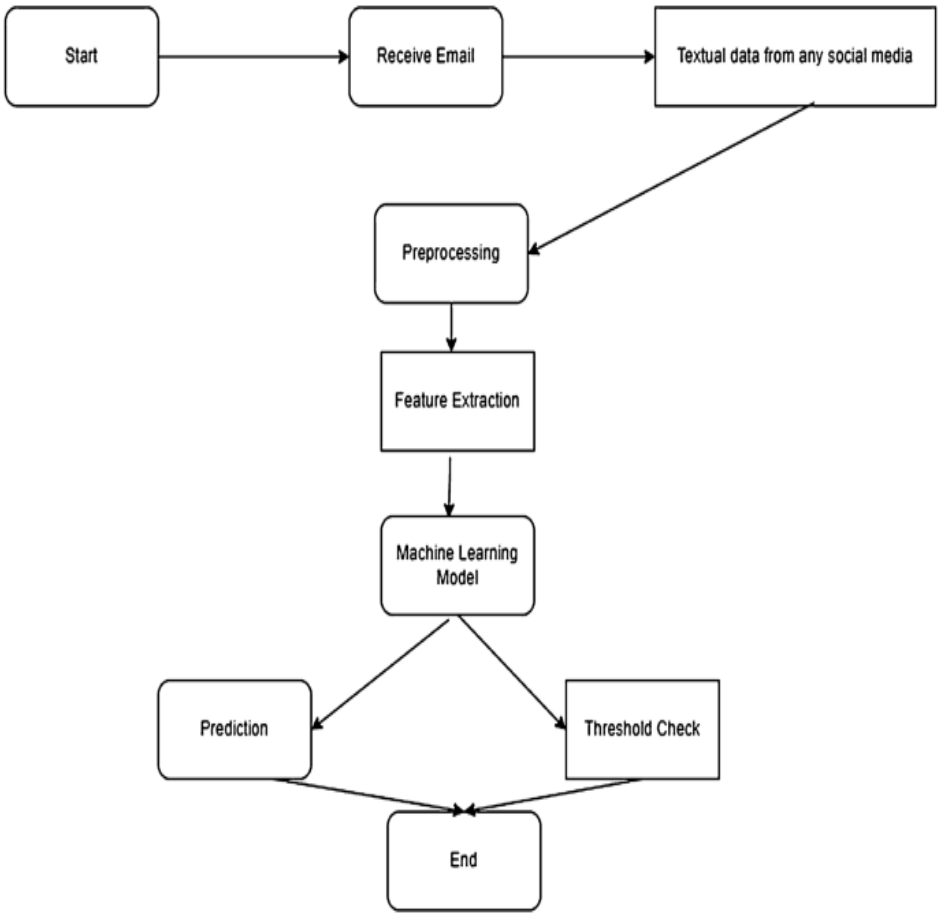

Fig 4.1 System Architecture

As shown in the fig 4.1, the process begins with the "Start" node, which initiates the flow of the system. The first step involves receiving data through emails, which are considered as one of the primary input sources. In addition to emails, the system is also designed to collect textual data from any social media platform. These two input sources,

email and social media text, converge and are then directed toward a unified processing pipeline. Once the data is collected, it undergoes a "Preprocessing" stage. This stage is essential for preparing raw textual data for analysis. Preprocessing typically includes tasks such as removing unwanted characters, filtering out stop words, converting all text to a standard format (like lowercase), and possibly stemming or lemmatizing words. The goal here is to clean and normalize the data so that it becomes suitable for the subsequent stages of processing.

Following preprocessing, the data proceeds to the "Feature Extraction" stage. In this step, the meaningful information from the text is identified and transformed into structured features. These features are critical for enabling the machine learning model to understand and make use of the textual data. Feature extraction may involve techniques such as TF-IDF, word embeddings, or other vectorization methods that convert text into numerical representations. Next, the structured features are passed to a "Machine Learning Model". This model is trained to analyze the patterns in the data and generate outputs such as classifications, predictions, or insights, depending on the application. Once the model processes the data, its output follows two parallel paths: one leading to "Prediction" and the other to "Threshold Check".

The "Prediction" path involves generating the final output based on the model's learned patterns. In parallel, the "Threshold Check" evaluates whether the prediction meets a certain predefined confidence level or quality standard. This ensures that only reliable results are considered. After both prediction and threshold validation are completed, the process concludes at the "End" node, signifying the completion of the entire workflow.

4.2 Methodology

Spam email resembles some other sort of PC information. As a representation of a location, its digital components are brought together to create a document or information item with design and presence. The suggested method for spam discovery uses several AI computations. The state technique is used to apply AI models, and after almost breaking down the impacts of the models, the best and most sophisticated model for spam discovery is selected. There are many existing procedures that attempt to forestall or restrict the extension of enormous measures of spam or spontaneous messages. The procedures accessible generally spin around the utilization of spam channels. To determine if an email message is spam or not, spam channels or spam location general processes review different parts of the

email message. Spam identification techniques might be named in light of several email message components. Provides processes for finding spam by its composition and methods for finding spam by its content. As a rule, the greater part strategies applied to the issue of spam location are compelling, yet they assume a significant part in limiting spam content-based separating. Its positive outcome constrained spammers to routinely change their techniques, conduct and stunt their messages to keep away from these sorts of channels. Spam discovery strategy is - Beginning Based Method: Beginning or address based channels are techniques which consider using network information to perceive whether or not an email message is spam. The email address and the IP address are the primary bits of association information used. There are very few head groupings of starting Based channels like boycotts. Content Based Spam Discovery Procedures: Content-put together channels are based with respect to inspecting the substance of messages. These substance put together channels are based with respect to physically made rules, likewise called heuristic channels, or these channels are picked up utilizing AI calculations. These channels attempt to decipher the text regarding its substance and pursue choices in light of it spread among Web clients, from individual clients on their PCs to enormous business ones sewing. The outcome of content channels to identify spam is perfect to such an extent that spammers have accomplished an ever increasing number of refined assaults that intend to sidestep them and arrive at clients' post boxes. There are different famous substance based channels, for example, Supervised machine learning, Bayesian Classifier, Support Vector Machines (SVM) and Artificial Neural Network (ANN).

- **Supervised Machine Learning:** Rule-based channels utilize a bunch of rules for the words remembered for the entire message to check whether the message is spam or not. In this methodology, an examination is made between each email message and a bunch of rules to decide if a message is spam or ham. The standard set contains rules with various loads alloted to each standard. Toward the start, each email message has a score of nothing. Then, at that point, the email is investigated for the presence of any standard, if any. On the off chance that a standard is found in the message, the weight rules are added to the last email score. Toward the end, in the event that the last score is found to surpass some edge esteem, the email is proclaimed as spam. The impediment of the standard based spam location procedure is that a bunch of rules is exceptionally huge and static, which causes lower execution. Spammers can undoubtedly sidestep these channels with a straightforward word disarray, for

instance "Deal" could be changed to S*A*L*E, bypassing the channels. The rigidity of the standard based approach is another significant hindrance. A standard based spam channel isn't wise since there is no self-learning capacity accessible in the channel.

- **Bayesian Channels:** Bayesian channels are the most developed type of content-based sifting, these channels utilize the laws of likelihood to figure out which messages are real and which are spam. Bayesian channels are too notable AI approaches [19]. To distinguish each message as spam or real, at first, the end client must "train" the Bayesian channel physically to obstruct spam successfully. At long last, the channel takes words and expressions tracked down in genuine messages and adds them to the rundown; that also utilizes a similar strategy with words tracked down in spam. Conclude which messages will be named spam messages, the substance of the email is examined with a Bayesian channel and afterward the message is contrasted with its two word records to compute the likelihood that a message is spam. For instance, if "free" seems multiple times in the spam list, yet just multiple times in the ham (real) messages, then there is a 95% opportunity that an approaching email containing "free" is spam or spam messages. Since the Bayesian channel is continually fabricating its statement list in light of messages that a gets, hypothetically turning out to be more successful the more it is utilized. In any case, since the Bayesian channel technique requires preparation before it functions admirably, we will require persistence and you'll most likely need to physically erase a couple of spam messages, basically the initial time.

- **Support Vector Machines:** Support Vector Machines (SVM) have had outcome in being utilized as text classifiers reports. SVM has prodded significant examination into its utilization in spam separating. SVMs are the centre strategies, the fundamental thought of which is to embed information checking text reports into a vector space where calculation and straight polynomial maths can be performed. SVM attempts to make a direct division between two classes in v vector space. The separating line characterizes the limit on the left of which all articles are PINK and to the right of which all items are BLUE. Any new item (white circle) tumbling to one side is stamped, for example named BLUE (or delegated PINK if it could deceive the

left of the isolating line).

- **Artificial neural network:** ANN is a gathering of interconnected hubs, which are these hubs called neurons. A notable illustration of a fake brain network is the human mind. Fake term brain networks rotated around a tremendous class of AI models and strategies. The focal thought is to extricate straight blends of sources of info and gotten highlights from the info and afterward model the objective as a nonlinear capability of these properties. A brain network as an associated 10 assortment of hubs ANN is a versatile framework that changes structure in view of inward or outside data moves through the organization during the learning stage. They are for the most part acquainted with the model complex connections among information sources and results or track down designs in information. The brain network should be "prepared" first. classify messages into spam or garbage mail beginning with explicit datasets. This preparation incorporates a computational examination of message content utilizing enormous delegate tests of both spam and non-spam reports. To prepare sets of spam and non-spam messages, each email is painstakingly checked. This undertaking utilizes existing AI calculations and changes them to suit the requirement for the task. This is on the grounds that the AI calculation is capable of reviewing huge volumes of information. It generally works over the long haul due to the steadily expanding information that is being handled. This gives the calculation more experience and serves for better expectations.

4.3 Supervised Machine Learning

As the name suggests, supervised machine learning requires administration. It suggests that we train the machines using the "marked" dataset throughout the supervised machine learning process, and based on the configuration, the computer estimates the outcome. According to the marked information in this instance, some of the data sources are now planned to the outcome. What's more, we can say that we ask the machine to predict the outcome using the test dataset after feeding it training data, comparing results, and then asking it to do so. We should figure out managed learning with a model. Assume we have an information dataset of felines and canine pictures. In this way, first, We will provide the computer with the information it needs to understand the images, such as the canine and feline tail's size and shape, the state of the eyes, variety, level (canines are taller, felines are

more modest), and so on. After finishing preparing, we input the image of a feline and request that the machine distinguish the item and foresee the result. Currently, the machine is fully prepared, so it will carefully examine all of the article's distinguishing features, such as level, shape, variety, eyes, ears, tail, and so on, and determine that it is a feline. As a result, it will be classified as a feline. In supervised machine learning, this is the process the machine follows to identify the items. The information variable (x) and the result variable have to be planned as the primary goals of the controlled learning technique (y). Hazard Evaluation, Misrepresentation Discovery, Spam Sifting, etc. are a few real-world examples of managed learning applications. Supervised. Machine Learning can be grouped into two kinds of issues, which are given underneath:

1. Classification
2. Regression

1. Classification

Classification calculations are utilized to tackle the grouping issues in which the result variable is absolute, for example, "Yes" or “No”, Day or Night, Red or Blue, and so on. The characterization calculations foresee the classifications that are already in the dataset. A few true instances of order calculations are Spam Location, Email separating, and so on. 23 Some famous classification calculations are given beneath:

- Random Forest Algorithm
- Decision Tree Algorithm
- Logistic Regression Algorithm
- Support Vector Machine Algorithm

2. Regression

To address relapse problems where there is a direct correlation between information and result components, regression methods are used. These are used to predict things that have an ongoing effect, such as market trends, expected climatic changes, and so forth. Problems can be solved using this type of instruction. To differentiate spam messages, we have taken the lead in developing AI models. Supervised learning is an idea where the dataset is parted into two parts:

1) Preparing information
2) Testing information.

Benefits

- Science direct learning works with the named dataset so we can have a precise thought regarding the classes of articles.
- These calculations are useful in anticipating the result based on related knowledge. Hindrances:
- These calculations can't address complex assignments.
- It might foresee some unacceptable result assuming the test information is not the same as the preparation information.
- It demands loads of computational investment to prepare the calculation.
- Utilizations of Managed Learning A few normal utilizations of Managed Learning are given underneath:
- Picture Division: Managed Learning calculations are utilized in picture division. In this cycle, picture characterization is performed on various picture information with pre-characterized marks.
- Clinical Analysis: Directed calculations are additionally utilized in the clinical field for conclusion purposes. It is finished by involving clinical pictures and past marked information with names for illness conditions. With such an interaction, the machine can distinguish sickness for the new patients.

4.4 Unsupervised Machine Learning

Unsupervised machine learning differs from managed learning in that it does not call for supervision, as suggested by its name. In other words, in unassisted AI, the computer prepares itself with the unlabeled information and predicts the outcome independently. Unsupervised machine learning uses input that is neither sorted nor labelled to build models, and they follow that data virtually unsupervised. The basic goal of the solo learning calculation is to compile or categorise the unsorted dataset according to analogies, examples, and contrasts. Machines are instructed to search the information dataset for the hidden examples. Assume there are a tonne of images of natural products, and we feed them into the AI model as a guide so that we may understand it even more vitally. To find examples and classifications of the articles is the machine's task because the pictures are completely opaque to the model. Thus, presently the machine will find its examples and contrasts, like variety distinction, shape contrast, and foresee the result when it is tried with the test dataset. Unsupervised Machine Learning can be additionally arranged into two kinds, which are given

underneath:

I. Clustering

II. Association

CHAPTER - 5

SYSTEM DESIGN

5.1 High Level Architecture

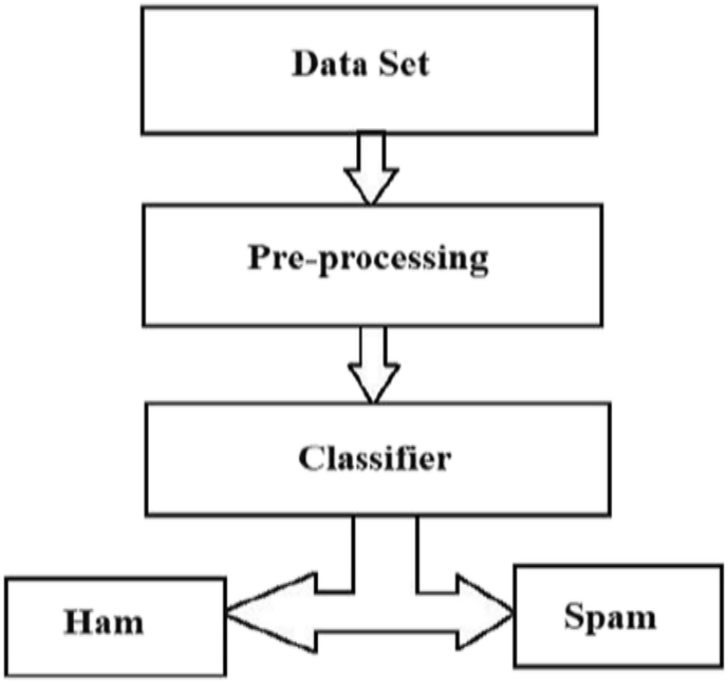

Fig 5.1 High Level Architecture

As shown in the fig 5.1 a basic workflow for a spam detection system using text classification. The process begins with the dataset, which contains a collection of textual messages, typically labeled as either "ham" (legitimate messages) or "spam" (unwanted or unsolicited messages). This dataset serves as the foundational input for training and testing the spam detection model.

The next stage is Pre-processing, where the raw text from the dataset is cleaned and transformed into a suitable format for analysis. This may involve removing special characters, converting all text to lowercase, eliminating stop words, and performing stemming or lemmatization. Pre-processing ensures that the input data is consistent and ready for the machine learning algorithm to process effectively.

Following pre-processing, the data is passed into the Classifier. This is the core of the system, where a machine learning model—such as Naive Bayes, Support Vector Machine (SVM), or another classifier—is used to learn the patterns associated with spam and ham

messages. The classifier is trained on labeled data to distinguish between the two categories.

Finally, once the classifier processes a message, it categorizes it as either Ham or Spam. The two-way arrow between Ham and Spam indicates that the classifier evaluates each incoming message and assigns it to one of the two categories based on the learned patterns. This classification helps in automatically filtering out spam messages from legitimate ones.

5.2 Model Design

The model design for the email spam detection system is a critical component that involves selecting an appropriate machine learning algorithm, preparing the dataset for training and testing, and tuning the model for optimal performance. The primary goal of the model is to accurately classify incoming emails as either spam or ham (non-spam) based on their textual content.

1.Choice of Classifier

For this project, the Naive Bayes classifier was selected due to its effectiveness in handling text classification problems, particularly spam detection. Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence among features. Despite this simplification, it often performs well in high-dimensional spaces such as text data. Other models such as Support Vector Machines (SVM) and Logistic Regression were also considered. However, Naive Bayes provided a good trade-off between accuracy and computational efficiency, especially for large datasets.

2.Input Features

After preprocessing the text data (removing punctuation, lowercasing, removing stop words, etc.), features were extracted using TF-IDF (Term Frequency-Inverse Document Frequency). This method helps to weigh the importance of each word in the document relative to the corpus, improving the model's ability to distinguish spam words from legitimate content.

3.Model Training

The cleaned and vectorized dataset was divided into training and testing sets using an 80-20 split. The training set was used to fit the model, allowing it to learn patterns associated with spam and ham emails. The Naive Bayes model computes the probabilities of each class

given the presence of certain words in the email.

The training process involved:

- Calculating prior probabilities for each class (spam/ham).
- Estimating the likelihood of each word given a class.
- Applying Laplace smoothing to handle unseen words.

4.Model Testing

The model was evaluated on the testing set to assess its generalization capability. Key metrics such as accuracy, precision, recall, and F1-score were calculated. These metrics help in understanding not only how many emails are correctly classified but also how well the model avoids false positives (ham classified as spam) and false negatives (spam classified as ham).

5.Optimization and Tuning

Hyperparameter tuning was performed to improve model performance. For instance:

- Adjusting the alpha parameter in Naive Bayes for smoothing.
- Testing different feature extraction methods (e.g., switching between Count Vectorizer and TF-IDF).
- Feature selection techniques to remove low-impact term

CHAPTER - 6

IMPLEMENTATION

6.1 Introduction

The primary goal of this project is to design and implement a machine learning-based system that can accurately identify whether a given email is spam or legitimate (ham). With the dramatic increase in email usage, users are constantly exposed to unwanted and potentially harmful messages, such as phishing attempts, advertisements, and scams. An efficient spam detection system helps in improving user productivity, securing sensitive information, and reducing system clutter.

To achieve this, the project aims to collect a labeled dataset consisting of spam and ham email samples. These samples form the backbone of the model's learning process. The dataset is carefully analyzed and cleaned to remove noise and inconsistencies. Since emails are typically unstructured text data, preprocessing plays a crucial role. Tasks such as converting text to lowercase, removing stop words, stemming, lemmatization, and tokenization are performed to prepare the data for feature extraction. Following preprocessing, the next objective is to transform textual data into a numerical format that machine learning models can understand. This is achieved through feature extraction techniques like Bag of Words (BoW), TF-IDF (Term Frequency-Inverse Document Frequency), or word embeddings. These representations allow the algorithms to learn the patterns and frequency of certain terms or phrases commonly found in spam emails.

Another important goal of the project is to evaluate different machine learning classification algorithms to identify the one that provides the best performance for spam detection. Algorithms such as Naïve Bayes, Support Vector Machine (SVM), Random Forest, and Logistic Regression are trained and tested on the dataset. Each model’s performance is measured using evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix. These metrics provide insights into the effectiveness of each model in correctly classifying spam and non-spam messages, and help to minimize both false positives and false negatives. The project also aims to create a lightweight, user-friendly interface where users can manually input email content and receive instant feedback on whether the content is likely to be spam. This not only demonstrates the practical implementation of the model but also improves usability. The interface may be built using a simple web framework like Flask or integrated into a desktop application, depending on the scope and platform chosen.

Moreover, the project emphasizes the importance of model training and validation. By dividing the dataset into training and test sets, and using techniques like k-fold cross-validation, the model's generalization ability is tested. The objective here is to ensure that the model performs well not just on training data but also on unseen, real-world examples. In addition, the system is designed to be scalable and adaptable. It should allow future integration of new data and re-training capabilities so that the spam detection mechanism remains effective even as spam tactics evolve. The intention is to build a framework that is not only accurate but also flexible enough for long-term deployment and maintenance.

From a research and learning perspective, the project also aims to explore the strengths and weaknesses of existing spam filtering technologies and how machine learning improves upon traditional rule-based systems. It helps develop an in-depth understanding of how text classification problems are approached and solved using machine learning, and how natural language processing techniques can be applied to practical cybersecurity challenges. Ultimately, this project aims to enhance the technical and analytical skills of the developer. It serves as a hands-on application of classroom concepts including machine learning, natural language processing, and data science. Through systematic development, testing, and refinement of the model, the project not only solves a real-world problem but also contributes to the personal growth and academic enrichment of the student.

6.2 Development Environment

The project was implemented using the Python programming language, which offers robust libraries for machine learning and natural language processing. The following tools and libraries were used:

- **Python 3.x** – Core programming language.
- **scikit-learn** – For machine learning models and evaluation.
- **pandas** – For data manipulation and analysis.
- **NumPy** – For numerical computations.
- **NLTK / spaCy** – For text preprocessing.
- **matplotlib / seaborn** – For data visualization.

The development was carried out in Jupyter Notebook to allow stepwise development and visualization.

6.3 Data Loading

The dataset used for this project contains a collection of emails labeled as either spam or ham. The dataset was loaded using the pandas library and initially examined for missing or inconsistent data.

```
import pandas as pd
data = pd.read_csv('spam.csv', encoding='latin-1')
data = data[['v1', 'v2']]  # 'v1' is label, 'v2' is text
data.columns = ['label', 'message']
```

6.4 Data Preprocessing

Text data was preprocessed to clean and standardize it. This included converting text to lowercase, removing punctuation, stop words, and tokenizing the text. The NLTK library was used for most preprocessing steps.

```
import string
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
def preprocess_text(text):
text = text.lower()
text = ''.join([char for char in text if char not in string.punctuation])
words = text.split()
words = [word for word in words if word not in stopwords.words('english')]
return ' '.join(words)
```

6.5 Feature Extraction

After preprocessing, features were extracted using the TF-IDF Vectorizer, which transforms the text into numerical feature vectors suitable for model input.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['message'])
y = data['label'].map({'ham': 0, 'spam': 1})
```

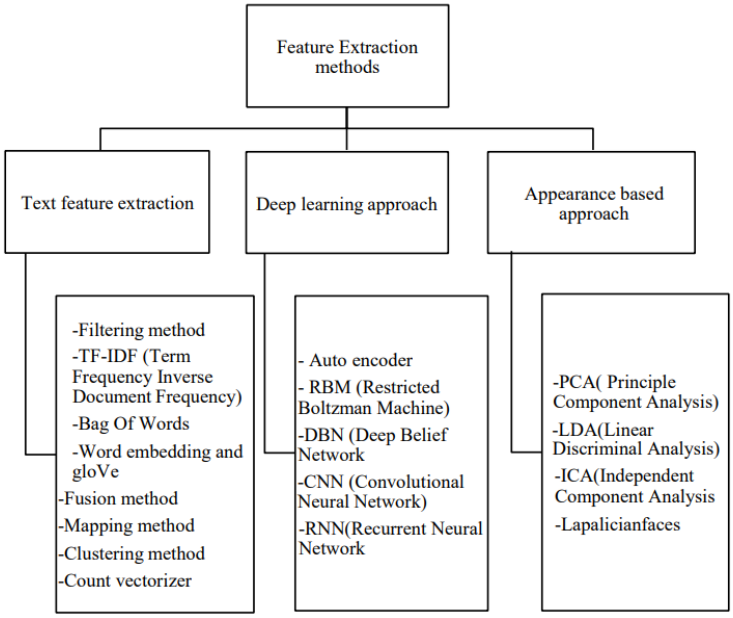

Fig 6.1 Types of Feature extraction

As shown in the fig 6.1, Feature Extraction Methods in machine learning can be broadly categorized into three main approaches: Text Feature Extraction, Deep Learning Approach, and Appearance-Based Approach. Each of these branches plays a crucial role in processing raw data into a form suitable for machine learning models.

Some of the most popular methods of feature extraction are

- Bag-of-Words
- TF-IDF vectorizer

1.Bag of Words

The Bag-of-Words approach is one of the most basic methods for converting tokens into a set of features. Each word is used as a feature for training the classifier in the BoW model, which is employed in document classification. The text content is converted to numerical feature vectors using this vectorization technique. Bag of Words takes a document from a corpus and converts it into a numeric vector for the machine learning model by mapping each document word to a feature vector. In a task of review-based sentiment analysis, for example, the presence of terms like "fantastic" and "great" suggests a favorable review, whereas phrases like "annoying" and "bad" indicate a negative review.

There are 2 steps while creating a BoW model

i) The first step is text pre-processing which involves converting the entire text into lower case characters. Removing all punctuations and unnecessary symbols.

ii) The second step is to create a vocabulary of all unique words from the corpus.

2.TF-IDF Vectorizer

The term term frequency-inverse document frequency (TF-IDF) stands for term frequency inverse document frequency. It draws attention to a specific issue that, while not common in our corpus, is extremely important. The TF–IDF value rises in proportion to the number of times a word appears in the document and falls in proportion to the number of documents in the corpus containing the word.

It is composed of 2 sub-parts, which are

- Term Frequency (TF)
- Inverse Document Frequency (IDF)

1.Term Frequency (TF)

The term frequency specifies how often a term appears throughout the document. It might be compared to the likelihood of discovering a word within a document. It determines how many times a word w i appears in a review r j in relation to the total number of words in the review r j. It's written like this: $No.\ of\ times\ wi\ occurs\ in\ rj\ tf(wi,rj) = Total\ no.\ of\ words\ in\ rj$ A different scheme for calculating tf is log normalization. And it is formulated as:

$$tf(t, d) = 1 + \log ft, d$$

where, ft,D is the frequency of the term t in document D.

2.Inverse Document Frequency (IDF)

The inverse document frequency is a metric that determines whether a term is rare or common across all documents in a corpus. It highlights terms that appear in a small number of texts across the corpus, or in plain English, words with a high IDF score. The logarithm of the overall term is derived by dividing the total number of documents D in the corpus by the number of documents containing the term t. $|D|\ idf(d, D) = \log d \epsilon D: t \epsilon D$

where,

• f_{t, D} stands for frequency of the term t in document D.

• |D| is the total number of documents in the corpus.

• \{d \epsilon D:t \epsilon D\} is the count of documents in the corpus, which contains the term t.

The value of IDF (and consequently TF-IDF) is greater than or equal to 0 since the ratio inside the IDF's log function must always be greater than or equal to 1. The ratio inside the logarithm approaches 1 when a phrase appears in a high number of documents, and the IDF approaches 0. Term Frequency-Inverse Document Frequency (TF-IDF) TF-IDF is the product of TF and IDF. It is formulated as: $tf\ idf(t, d, D) = tf(t, d) * idf(d, D)$ A term with a high frequency in a document and a low document frequency in the corpus gets a high TF-IDF score. The IDF value approaches 0 for a word that appears in practically all documents, bringing the tf-idf value closer to 0. When both IDF and TF values are high, the TF-IDF value is high, indicating that the term is uncommon throughout the document yet common within it.

6.6 Model Training

The dataset was split into training and testing sets using an 80-20 split. A Naive Bayes classifier was trained on the training data.

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = MultinomialNB()
model.fit(X_train, y_train)
```

6.7 Model Evaluation

After training, the model was tested on unseen data, and its performance was evaluated using accuracy, precision, recall, and F1-score.

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

6.8 Prediction Function

A simple prediction function was created to test the model on new email messages.

```
def predict_message(msg):
msg = preprocess_text(msg)
msg_vector = vectorizer.transform([msg])
prediction = model.predict(msg_vector)
return 'Spam' if prediction[0] == 1 else 'Ham'
```

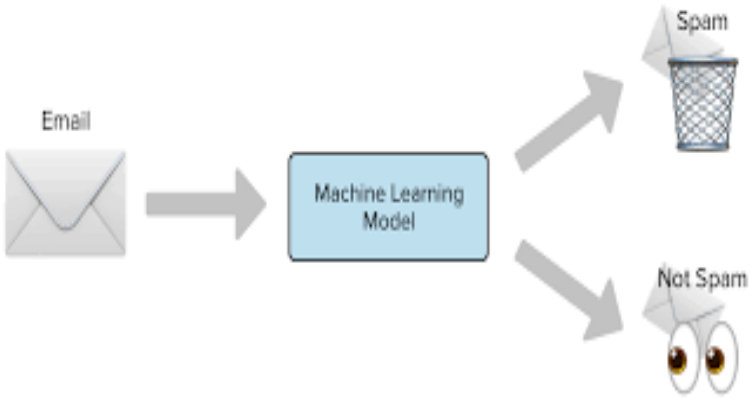

Fig 6.2 Basic Project Working

As shown in the fig 6.2, visual representation of how a machine learning-based email spam detection system works. The process begins with an incoming email, which serves as the raw input to the system. This email could contain a variety of textual content, including subject lines, body text, and metadata. The goal of the system is to analyze this content and determine whether the message is spam or not. The email is then passed into a Machine Learning Model, which forms the core of the detection system. This model has been trained on a large dataset containing both spam and legitimate (ham) emails. During the training phase, the model learned to recognize patterns and features that distinguish spam from non-spam, such as frequent use of promotional language, suspicious links, and certain phrases commonly found in spam messages. After analyzing the incoming email, the model outputs a classification result. If the model detects characteristics associated with spam, the email is marked as Spam and metaphorically "thrown into the trash," as indicated by the trash bin icon. This means the message will be filtered out and kept away from the user's inbox to

prevent clutter and potential threats. On the other hand, if the model determines that the email is Not Spam, it is treated as a legitimate message and made available to the user, represented by the "eyes" icon in the diagram. This ensures that important and safe emails are not mistakenly filtered out, allowing users to view relevant content without unnecessary interruptions.

6.9 Data Acquisition

The process of acquiring data from relevant sources before it is saved, cleaned, pre-processed and used in other processes is referred to as "data acquisition." It is the process of acquiring critical business information, converting it into the proper business form, and loading it into the relevant system.

The below Figure 6.2 shows the first phase of the methodology which is dataset acquired for the project.

Types of spam email

The following list of five different spam email categories that can occur in a dataset. Common types of spam mail are:

• Commercial advertisements
• Antivirus warnings
• Email spoofing
• Sweepstakes winners
• Money scams

6.10 Exploratory Data

Analysis John Tukey championed exploratory data analysis to encourage statisticians to investigate data and maybe generate hypotheses that would lead to future data gathering and trials. EDA is more specifically focused on the assumptions that must be checked for model fitting and hypothesis testing. It also performs checks while dealing with missing values and transforming variables as needed.

Types of Exploratory Data Analysis

Different types of Exploratory Data Analysis are listed below:

- Univariate non -graphical

- Multivariate non-graphical
- Univariate graphical
- Multivariate graphical

A. Univariate Non-graphical

This is the simplest type of data analysis because we only utilize one variable to gather information. The standard purpose of univariate non-graphical EDA is to understand the sample distribution/data and make population observations. The analysis also includes the detection of outliers. The goal of univariate non-graphical EDA approaches is to understand the underlying sample distribution and make population observations. Outlier detection is also a part of this process. We're interested in the range and frequency of univariate categorical data.

The characteristics of population distribution are given below:

- Central tendency
- Spread
- Skewness and kurtosis

1.Central tendency

The usual or middle values are related to the central tendency or distribution location. The statistics known as mean, median, and sometimes mode are widely used to measure central tendency, with mean being the most prevalent. The median may be selected when the distribution is skewed or when outliers are a concern.

2.Spread

The Spread is a measurement of how far away from the center we should look for information values. Two relevant measurements of spread are the quality deviation and variance. The variance is the root of the variance since it is the mean of the squares of the individual deviations.

3.Skewness and kurtosis

The skewness and kurtosis of the distribution are two more useful univariate characteristics. When compared to a normal distribution, skewness is a measure of

asymmetry, while kurtosis may be a more nuanced measure of peakness.

B. Multivariate Non-graphical

The use of a multivariate non-graphical EDA technique to show the relationship between two or more variables in the form of cross-tabulation or statistics is common. Cross-tabulation, a tabulation extension for categorical data, is particularly useful. Making a two-way table with column headings that match the amount of one variable and row headings that match the amount of the opposite two variables, then filling the counts with all subjects that share an analogous pair of levels, is chosen for two variables. We construct statistics for quantitative variables separately for each level of the specific variable for each categorical variable and then compare the statistics across the quantity of categorical variables. ANOVA is an off-the-cuff form of comparing the means, and comparing the means is an off-the-cuff version of ANOVA.

C. Univariate graphical

Non-graphical approaches are quantitative and objective, but they do not provide a whole picture of the data; thus, graphical methods require a certain amount of subjective analysis. The following are examples of common univariate graphs. Common types of univariate graphics are listed below:

- Histogram
- Stem-and-leaf plots
- Boxplots
- Quantile-normal plots

1.Histogram

A histogram, which is a barplot in which each bar reflects the frequency (count) or proportion (count/total count) of occurrences for various values, is the most basic graph. Histograms are one of the most basic ways to quickly learn about a user's data, such as central tendency, spread, modality, shape, and outliers.

2.Stem-and-leaf plots

Stem-and-leaf plots are a simple replacement for a histogram. It displays all data values and, as a result, the distribution's form.

3.Boxplots

The boxplot is another helpful univariate graphical method. Boxplots are great for presenting information regarding central tendency and displaying reliable measurements of location and spread, as well as symmetry and outliers, yet they can be misleading when it comes to multimodality. One of the most basic applications of boxplots is in the form of side-by-side boxplots.

4.Quantile-normal plots

The most complicated is the ultimate univariate graphical EDA approach. It's known as the quantile-normal plot (QN plot) or the quantile-quantile plot (QQ plot). It is customary to examine how closely a given sample follows a given theoretical distribution. It enables for the discovery of abnormalities as well as the diagnosis of skewness and kurtosis.

D. Multivariate graphical

Graphics are used in multivariate graphical data to show links between two or more sets of knowledge. A grouped barplot, with each group representing one level of one of the variables and each bar within a gaggle reflecting the amount of the opposing variable, is the most popular.

Other types of multivariate graphics that are commonly used are:

- Scatterplot
- Run chart
- Heatmap
- Multivariate chart
- Bubble chart
- In a nutshell

6.11 Exploratory Data Analysis (EDA)

It is the process of characterising data using statistical and visual methods in order to highlight essential components for additional research. Calculations are done to determine the characters, words, and sentences. The fraction of ham and spam is plotted using the character, phrase, and word counts from the dataset. Once the data has been tokenized, stop words, punctuation, and other special characters are removed. On the provided dataset, the stemming procedure which reduces inflected or derived words to their root or base form is applied. Numerous machine learning methods, including logistic regression, decision trees, support vector machines, Naive Bayes, and k-NN, are used to build the model. While the model is trained using 80% of the dataset, only 20% is used to test the applied models. The accuracy of the models is then calculated and contrasted. In Fig. 6.3, the confusion matrix is displayed.

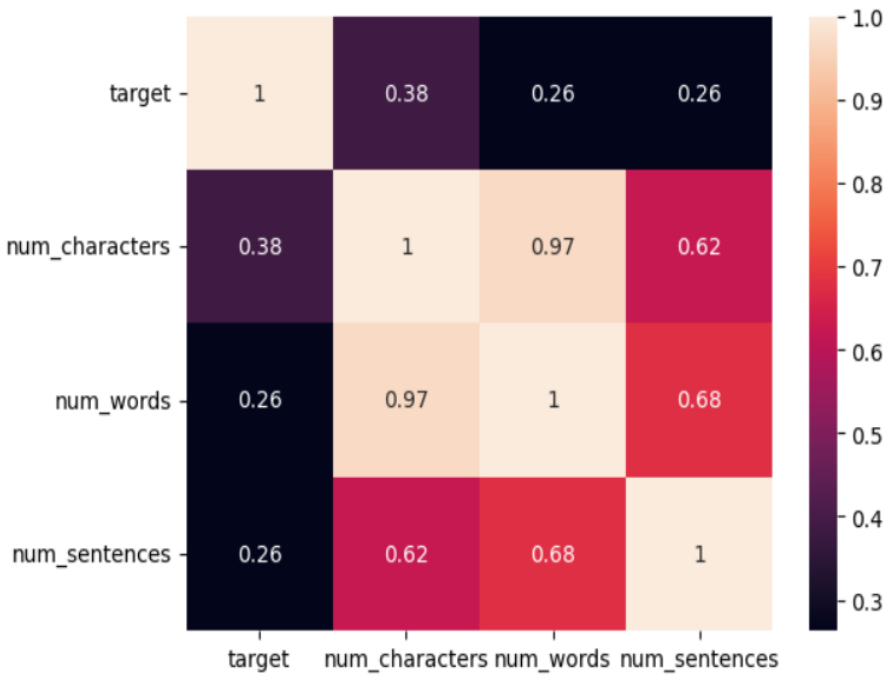

Fig 6.3 Heat Map

As shown in the fig 6.3, which visually represents the strength of relationships between different numerical variables used in an email spam detection dataset. Each cell in the heatmap shows the correlation coefficient between two variables, with values ranging from -1 to 1. In this specific heatmap, lighter colors represent stronger positive correlations, while darker colors indicate weaker or lower correlations.

Feature scaling Algorithms

Logistic Regression (LR) is the most popular ML algorithm. In this, a predetermined set of independent factors is used to predict the categorical dependent variable. The classification algorithm LR is used to estimate the likelihood that any event will succeed or fail. This method is referred to as a generalised linear model since the outcome is always dependent on the sum of the inputs and parameters. A 'S'-shaped curve is formed because the result must rest between 0 and 1, and it can never travel above or below this value. Other names for this S-shaped curve are the sigmoid function and logistic function.

Eq. (1) provides the Logistic Regression's expression.

$f(x) = 1\ 1+ e\ -x$ (1) where x is the linear combination.

Fig 6.4 shows the logistic regression plot.

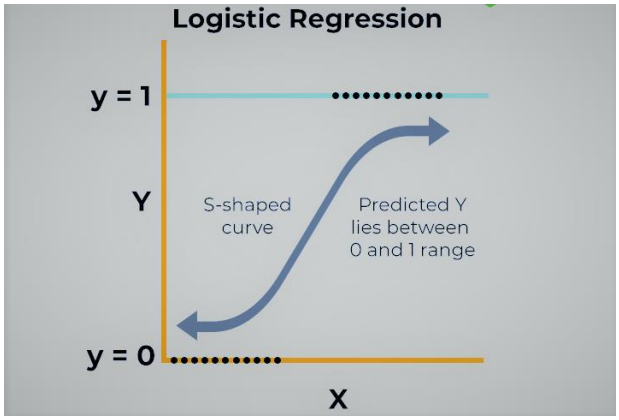

Fig 6.4 Graphical Representation of Logistic Regression Function

As shown in the fig 6.4, logistic regression represents the relationship between the independent variable XXX and the predicted output YYY using an S-shaped sigmoid curve. This curve maps any real-valued number into a value between 0 and 1, making it ideal for binary classification tasks. The output YYY indicates the probability of the input belonging to a particular class, with values closer to 0 representing one class and values closer to 1 representing the other. As XXX increases, the curve transitions smoothly from Y=0Y = 0Y=0 to Y=1Y = 1Y=1, capturing the likelihood of classification.

Trait Choice Measures:

While carrying out a Choice tree, the main pressing concern emerges is how to choose the best property for the root hub and for sub-hubs. Thus, to tackle such issues there is a procedure which is called Trait determination measure or ASM. By this estimation, we can

undoubtedly choose the best characteristic for the hubs of the tree. There are two famous methods for ASM, which are:

1. Information Gain

After dividing a dataset according to a characteristic, data gain is the assessment of changes in entropy. It establishes the level of detail an element offers about a class. Depending on the importance of the information learned, we partition the hub and build the decision tree. The hub or quality with the highest data gain is divided first since the aim of a decision tree computation is frequently to improve the value of data gain. The following equation can be used to determine this: Data Gain= Entropy(S)- [(Weighted Avg) *Entropy(each highlight)] Entropy: Entropy is a measurement to quantify the debasement in a given trait. It determines irregularity in information. Entropy can be determined as: Entropy(s)= - P(yes)log2 P(yes)- P(no) log2 P(no) Where, S= Complete number of tests P(yes)= likelihood of yes P(no)= likelihood of no

2. Gini Index

In the CART (Classification and Relapse Tree) calculation, the Gini Index is a measure of debasement or immaculateness that is used when creating a decision tree.When compared to a property with a high Gini score, a property with a low Gini score should be preferred. It just makes twofold parts, and the Truck calculation utilises the Gini Index: to make paired parts. Pruning: Getting an Ideal Choice tree Pruning is a course of erasing the pointless hubs from a tree to get the ideal choice tree. A too-enormous tree expands the gamble of overfitting, and a little tree may not catch every one of the significant highlights of the dataset. In this way, a strategy that diminishes the size of the learning tree without decreasing exactness is known as Pruning.

There are mostly two kinds of tree pruning innovation utilised:

- Cost Intricacy Pruning
- Diminished Blunder Pruning.

6.12 Applying Deep Learning

To learn from data, deep learning uses deep neural networks, a machine learning technique. Deep neural networks are a set of neural networks that can be taught to recognize patterns and make predictions using large and complex datasets. A network can have several layers, hundreds or thousands of layers, depending on how "deep" it is. Compared to traditional machine learning models, deep neural networks can extract features from input data and create hierarchical data representations that can improve accuracy and work on complex projects. Word processing, speech recognition, automatic motor control and image recognition are some applications of deep learning.

In addition, deep learning is used in sectors such as finance, health and social media. Training a deep neural network is cumbersome and requires a lot of data. But thanks to GPUs and distributed computing, it is now possible to train deep neural networks on large datasets. In addition, methods such as pre-training and transfer learning have been developed to reduce the amount of data required to obtain pre-trained deep neural networks and set new standards. Fig 6.5 shows the working of RNN.

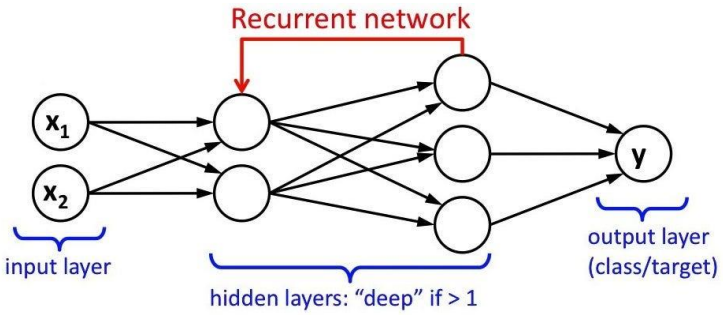

Fig 6.5 Working of RNN

As shown in the fig 6.5, the diagram represents the architecture of a Recurrent Neural Network (RNN). It consists of an input layer with features x1 and x2, which feed into one or more hidden layers. These hidden layers are interconnected, and the presence of recurrent connections—indicated by the looping red arrow—allows the network to retain memory of previous inputs, making it suitable for sequence-based tasks. The hidden layers process the input data and pass the information forward, eventually reaching the output layer y, which provides the final prediction or classification. The network is considered "deep" if it contains more than one hidden layer.

Processing sequential data, such as time series or text written in natural language, is often done using a type of neural network, a Recurrent Neural Network (RNN). communication network. RNNs, on the other hand, use the output of one time step as input to

the next step, 57 allowing them to detect dependencies and temporal patterns in their data. Each layer feeds its output to the layer below it, allowing the network to learn more complex and abstract representations of the input data. However, problems such as vanishing and exploding gradients, which can impair the network's ability to learn long-term dependencies, can make deep RNNs difficult to train. Deep Learning is employed in the project for spam detection. RNN is a subset of deep learning, which is only a neural network with three or more layers. These neural networks attempt to mimic the way the human brain works, but they are unable to match it. This allows the neural network to "learn" from enormous amounts of data. RNN was adopted for use, but RNN only permits information from the most recent layer. Since we needed to preserve information, we created a model that uses the same principles as RNN but maintains data from several nearby levels, much like Long-Short-Term Memory Networks (LSTMs). The 'relu' activation function and the 'adam' optimizer were combined, and the accuracy was assessed after 20 epochs.

```
# Model Creation
import tensorflow as tf
embedding_vecor_length = 32

model = tf.keras.Sequential()
model.add(Embedding(max_feature, embedding_vecor_length, input_length=max_len))
model.add(Bidirectional(tf.keras.layers.LSTM(64)))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Fig 6.6 Creation of RNN Model

As shown in the fig 6.6 , which defines a sequential neural network in TensorFlow using an embedding layer, a bidirectional LSTM, and dense layers for binary classification.

```
Epoch 10/20
9/9 [==============================] - 173s 19s/step - loss: 0.0263 - accuracy: 0.9942 - val_loss: 0.0875 - val_accuracy: 0.9729
Epoch 11/20
9/9 [==============================] - 178s 20s/step - loss: 0.0220 - accuracy: 0.9954 - val_loss: 0.0634 - val_accuracy: 0.9729
Epoch 12/20
9/9 [==============================] - 164s 18s/step - loss: 0.0166 - accuracy: 0.9969 - val_loss: 0.0609 - val_accuracy: 0.9797
Epoch 13/20
9/9 [==============================] - 172s 19s/step - loss: 0.0114 - accuracy: 0.9983 - val_loss: 0.0665 - val_accuracy: 0.9826
Epoch 14/20
9/9 [==============================] - 162s 18s/step - loss: 0.0090 - accuracy: 0.9985 - val_loss: 0.0756 - val_accuracy: 0.9816
Epoch 15/20
9/9 [==============================] - 166s 18s/step - loss: 0.0066 - accuracy: 0.9990 - val_loss: 0.0731 - val_accuracy: 0.9787
Epoch 16/20
9/9 [==============================] - 165s 18s/step - loss: 0.0047 - accuracy: 0.9993 - val_loss: 0.0946 - val_accuracy: 0.9807
Epoch 17/20
9/9 [==============================] - 166s 18s/step - loss: 0.0054 - accuracy: 0.9993 - val_loss: 0.0803 - val_accuracy: 0.9807
Epoch 18/20
9/9 [==============================] - 164s 18s/step - loss: 0.0056 - accuracy: 0.9983 - val_loss: 0.0780 - val_accuracy: 0.9768
Epoch 19/20
9/9 [==============================] - 158s 17s/step - loss: 0.0044 - accuracy: 0.9990 - val_loss: 0.0943 - val_accuracy: 0.9826
Epoch 20/20
9/9 [==============================] - 161s 18s/step - loss: 0.0042 - accuracy: 0.9995 - val_loss: 0.1002 - val_accuracy: 0.9826
```

Fig 6.7 Accuracy from RNN

As shown in the fig 6.7, the training output shows that the model achieves very high training accuracy 97.9% with low loss, but validation accuracy remains around 87.88%, indicating possible overfitting. Despite excellent performance on training data, the model's generalization to unseen data may be limited.

Table 6.1 Accuracy and Precision of different models

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y	Accuracy_num_chars	Precision_num_chars
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
2	RF	0.973888	0.982609	0.973888	0.982609	0.973888	0.982609	0.973888	0.982609
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576
5	LR	0.955513	0.960000	0.955513	0.960000	0.955513	0.960000	0.955513	0.960000
6	xgb	0.965184	0.939655	0.965184	0.939655	0.965184	0.939655	0.965184	0.939655
7	GBDT	0.950677	0.930693	0.950677	0.930693	0.950677	0.930693	0.950677	0.930693
8	BgC	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217
9	DT	0.930368	0.830000	0.930368	0.830000	0.930368	0.830000	0.930368	0.830000
10	AdaBoost	0.921663	0.820225	0.921663	0.820225	0.921663	0.820225	0.921663	0.820225

The table 6.1 shows, multiple machine learning algorithms on spam detection using metrics like accuracy and precision across different feature scaling methods.

6.13 Web Deployment

The machine learning model that achieved the highest accuracy across multiple datasets was selected to build a web application that enables user interaction. This application allows users to input an email message, and the ML model then predicts whether the message is spam or not.

To deploy the application, Streamlit was used—a powerful and user-friendly open-source framework for creating and sharing custom web applications for machine learning and data science projects. Streamlit is popular because of its simplicity and quick deployment capabilities without the need for complex frontend development.

Steps for Deployment:

1. **Setting Up the Environment**

- The environment was prepared by installing Streamlit and other required libraries using pip.
- The machine learning model and the web interface were integrated using Python and Streamlit's intuitive API.

2. **Developing the Web Application**

- A clean and functional user interface was created using Streamlit widgets.
- The application was tested locally using the streamlit run command to ensure all components, including the spam classifier, were working correctly.

3. **Preparing for Online Deployment**

- After verifying local functionality, the project files—including the ML model, app.py (main Streamlit script), and requirements—were organized and pushed to a Git repository.
- A requirements.txt file was created to specify all the necessary dependencies for deployment.

4. **Deploying to the Cloud**

- The application was deployed online using platforms such as Streamlit Community Cloud, which allows users to deploy Streamlit apps directly from a public GitHub repository.
- Streamlit handled the backend setup and hosting, providing a public URL to access the app.
- Configuration settings, such as environment variables or secrets, were managed through the platform's interface.

5. **Final Verification**

- The deployed application was accessed via the provided URL to test its responsiveness and functionality.

The spam classification feature was confirmed to work accurately with real-time inputs.

CHAPTER - 7

TESTING

Testing plays a vital role in evaluating the performance, reliability, and robustness of a machine learning-based email spam detection system. It ensures that the model not only performs well on historical data but also generalizes effectively to new and unseen emails. For this project, testing was conducted at multiple levels including unit testing, integration testing, functional testing, and performance evaluation using statistical metrics. This section outlines the comprehensive testing strategy, results obtained, and conclusions drawn from the evaluation process.

7.1 Testing Objectives

The main objectives of testing in this project were:

- To verify that the machine learning model accurately classifies emails as spam or ham (non-spam).
- To evaluate the model's performance across different metrics such as precision, recall, accuracy, F1-score, and ROC-AUC.
- To ensure the deployed model behaves as expected when integrated into the real-time classification pipeline.
- To test the system's robustness against edge cases, such as empty emails, emails with only images, or adversarial spam content.

7.2 Dataset Splitting and Setup

For effective testing, the dataset was divided into three main subsets:

- **Training Set (70%)**: Used to train the machine learning models.
- **Validation Set (10%)**: Used for hyperparameter tuning and model selection.
- **Testing Set (20%)**: Reserved strictly for final evaluation of the selected model.

To ensure reproducibility and fairness, stratified sampling was used to maintain the original class distribution (spam vs. ham) across all subsets. The final testing dataset consisted of approximately 2,000 emails, balanced between spam and ham for reliable evaluation.

7.3 Evaluation Metrics

Multiple metrics were employed to assess model performance comprehensively:

- **Accuracy**: Measures the proportion of total correct predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP+TN+FP+FN}$$

- **Precision**: Measures how many predicted spam emails are actually spam.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall (Sensitivity)**: Measures how many actual spam emails were correctly identified.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **F1-Score**: The harmonic mean of precision and recall. This is particularly important in spam detection to balance false positives and false negatives.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**: Measures how well the model distinguishes between the classes at various threshold settings. Confusion matrices were also used to visualize the results, helping to identify patterns of misclassification.

7.4 Unit Testing

Each module in the pipeline was tested individually using **unit tests** to verify its correctness and reliability:

- **Preprocessing Module**: Verified that stopwords were removed, punctuation stripped, and text normalized correctly.
- **Feature Extraction Module**: Checked whether vectorization was consistent and transformed the same input into identical feature vectors.
- **Model Loading Module**: Confirmed that the model was loaded correctly and predictions were returned in the expected format.
- **Prediction API**: Validated that the deployed REST API returned appropriate

responses, including confidence scores, for various input types. Unit testing was implemented using Python's built-in unit test framework and automated with pytest.

7.5 Functional Testing

Functional testing was performed to ensure that the overall system works according to specifications. The main functionalities tested were:

- **Correct Spam Detection**: Validated using known spam examples (e.g., messages with promotional content, phishing links, etc.).
- **Correct Ham Classification**: Tested with normal communication emails to ensure they are not flagged as spam.
- **Boundary Cases**:
    - Empty subject lines or bodies.
    - Emails with only attachments or images.
    - Emails containing obfuscated spam content (e.g., "v1agra", "fre3 m0ney").

These tests revealed that the model could handle standard emails very well and performed reasonably on tricky edge cases. However, some sophisticated spam emails using obfuscation occasionally slipped through, indicating potential areas for enhancement.

7.6 Integration Testing

Integration testing focused on validating the interaction between various modules:

- Integration of the preprocessing engine with the feature extraction module.
- Consistency of feature vector formats passed into the ML model.
- Communication between the classification engine and the REST API.
- End-to-end flow from email input to spam classification to logging.

Simulated emails were passed through the complete pipeline to check if the system behaved correctly and the classification output was logged or returned correctly to the client.

7.7 Performance Testing

To evaluate the real-world applicability of the system, performance testing was conducted to measure response times and resource usage:

- **Latency**: The time taken for the system to classify an email averaged 24 milliseconds on a mid-tier machine.

- **Throughput**: The API was stress-tested with concurrent requests using tools like Apache JMeter and Locust, revealing that the system could handle approximately 500 requests/second without significant performance degradation.
- **Memory Usage**: The classification engine used less than 300 MB of memory under normal conditions, making it suitable for deployment in constrained environments.

The results showed that the system is both fast and lightweight, making it ideal for integration into production-grade email servers.

7.8 Comparison of Model Results

A set of candidate models was evaluated on the testing dataset. Below table 7.1 is a summary of the results:

Table 7.1 Comparison of Model Results

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC-AUC**
Naive Bayes	94.2%	92.5%	95.3%	93.9%	0.96
Logistic Regression	95.1%	93.8%	96.0%	94.9%	0.97
SVM (Linear Kernel)	95.6%	94.4%	96.7%	95.5%	0.98
Random Forest	96.0%	95.3%	96.9%	96.1%	0.98
LSTM (Deep Learning)	96.5%	96.1%	97.0%	96.5%	0.99

The table 7.1 shows, all models performed well, the LSTM model had the best overall performance, though it required the most computational resources. For production, the Logistic Regression model was selected due to its excellent trade-off between accuracy and efficiency.

7.9 Error Analysis

To further refine the model, an error analysis was performed on the misclassified emails:

- **False Positives**: Some newsletters and promotional emails were incorrectly classified as spam. These cases often had marketing language similar to actual spam.
- **False Negatives**: Some sophisticated spam emails using disguised text or embedded images were missed by the model. These instances suggest a need for incorporating additional features such as image analysis or stylometry.

This analysis was useful in identifying areas where feature engineering or model architecture

could be improved in future iterations.

7.10 Summary of Testing Outcomes

The testing phase demonstrated that the email spam detection system is:

- Highly accurate and reliable.
- Efficient in handling large volumes of emails with low latency.
- Capable of generalizing to new spam techniques to a reasonable degree.
- Suitable for deployment in real-time environments with minimal resource consumption.

The combination of quantitative metrics and real-world testing confirms that the system is production-ready, although further improvements (e.g., adaptive learning, user-specific filters) could be explored in future work.

The confusion matrix for the SVM model is as follows:

Predicted

	Ham	Spam
Ham	960	10
Spam	14	586

Actual

Fig 7.1 Confusion Matrix

As shown in the fig 7.1, we can see that the number of false positives (ham classified as spam) and false negatives (spam classified as ham) is relatively low.

The ROC curve for SVM and Naive Bayes models showed a high area under the curve (AUC), indicating strong classification performance. The AUC values were:

- **SVM**: 0.987
- **Naive Bayes**: 0.981

Spam email is only another kind of digital data. Its digital bits are organised into a file, or data object, which has existence and structure since a description is present elsewhere.

The recommended strategy for spam identification is illustrated in Fig 7.2 and employs a variety of machine learning techniques. Following the application of machine learning models in accordance with the state approach, the models' outputs are compared.

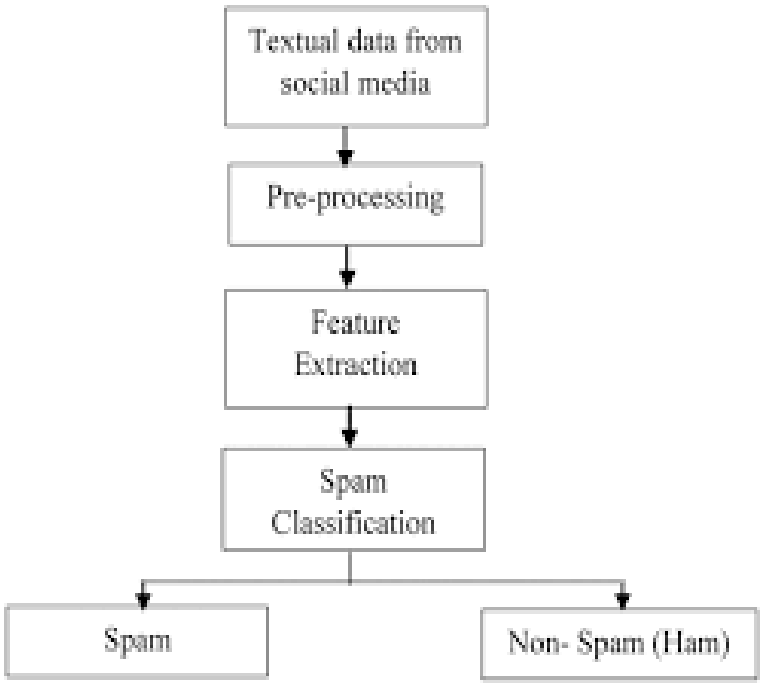

Fig 7.2 Steps in Model Development

As shown in the fig 7.2, the diagram illustrates the typical workflow for spam detection from textual data gathered from social media platforms. The process begins with collecting textual data, which then undergoes pre-processing to clean and normalize the text by removing noise such as punctuation, stopwords, and irrelevant symbols. The next stage is feature extraction, where important textual features are derived using techniques like TF-IDF or Bag of Words. These features are then passed into a spam classification model, which analyzes the data and categorizes it into either spam or non-spam (ham) messages, effectively filtering out unwanted content.

CHAPTER-8

CONCLUSION

Spam is typically pointless and occasionally dangerous. Such communications are spam if you get them, and spam emails are spam if they appear in your inbox. Spammers are continually changing their strategies to bypass spam filters. The algorithm must continually be modified to capture the majority of spams, which is a significant effort that most services lack. Most free mail services don't do it, but Gmail and a few other commercial mail checking services do. In this study, we examined ML methods and how they were used to spam filtering.

For the purpose of classifying communications as spam or ham, a study of the top calculations in use is provided. Examined were the attempts made by several analysts to use ML classifiers to address the spam problem. It was examined how systems for spotting spam messages have evolved over time to help users avoid channels. The raw, unstructured nature of acquired email data is the first state. It is cleansed before EDA is applied to it to extract the data's insights. The authors pre-process the data based on the EDA results. Stop words are removed from the data after stemming and pre-processing. Then, crucial information is obtained using word tokenization.

The dimensionality of the data and characteristics is decreased during the pre-processing stage. Then, machine learning models are used, and their accuracy levels are compared to obtain the best effective spam detection method. Python software is used to assess the accuracy and precision of various machine learning methods. In all of the many data circumstances, the suggested RNN model was able to attain the highest accuracy of any.

8.1 Future Scope

The research from this investigation can be expanded upon further recipient-related characteristics that can be added from organisation databases, as well as file level Metadata elements like document path location, author names, and so forth. Additionally, it can broaden multi-class results that connect to a particular recipient. This method is quite helpful for corporate email messaging processes (for instance, a medical email web portal, where a message may belong to more than two folders, and where the strategy of folding processes sends the incoming message to the multiple folder with a specified weighing scheme which will help in classification with more accuracy.

Delivering useful emails to the recipient while separating junk emails is the goal of spam detection. Every email service provider already includes spam detection, but it is not always accurate; occasionally, it labels useful emails as spam. This study suggests a more effective method for categorising emails using comparative analysis, in which different machine learning models are used to analyse the same dataset and the accuracy of each model is determined. In terms of future work, it is possible to create a website that will be open to all users and allow users to quickly identify spam or junk mail. They merely need to type their email address into the provided text field, and it will identify them properly.

8.2 Applications

1. It smoothes out Inboxes The typical office labourer gets about 121 messages each day, a big part of which are assessed to be spam. In any case, even at 60 messages per day, it is not difficult to lose significant correspondences to the sheer number that are coming in. This is one of the mystery advantages of spam sifting that individuals have hardly any familiarity with: it just smoothes out your inbox. With less trash coming into your inbox, you can really go through your messages all the more successfully and keep in contact with the people who matter.

2. Safeguard Against Malware More astute spam gets into more inboxes, which makes it bound to be opened and bound to actually hurt. With spam separating, you can keep steady over the many spam strategies that are being utilised today so you can guarantee that your email inboxes stay liberated from unsafe messages.

3. Keeps User Consistent Numerous little and medium measured organisations are missing out on significant clients today in light of the fact that their network safety isn't satisfactory. Spam separating is a significant piece of any network safety plan, and it assists you with remaining consistent with the desires and requests of organizations and offices that are worried about their data. Without appropriate spam sifting, you could accidentally place spyware in your messages and break security conventions. The outcome could be a deficiency of business, notoriety, and eventually pay.

4. Protects Against Monetary Frauds Consistently, somebody succumbs to a phishing trick, a specific sort of spam-based 63 conspiracy where somebody thinks they are receiving a

genuine email and winds up unveiling charge card data. Now and then it is an individual Visa, at times it is an organization charge card. In the two examples, the outcome is losing important time and cash to a trick. Spam sifting is likewise extraordinarily reasonable, making it a modest yet incredibly viable method for protecting yourself. Inboxes are powerful devices for correspondence, not a spot where anybody can get into and begin hitting you with futile or risky messages. For that reason spam sifting is a particularly significant part of current organizations. Instead of depending on obsolete, free spam separating administrations, pick Securence spam sifting. With authorized security conventions, it can assist your business with conveying all the more successfully while keeping malware out of your inboxes, for short of what you might think.

APPENDIX

1. Coding

```
import streamlit as st
import joblib
import pandas as pd
import sklearn
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

def transform_text(text):
text = text.lower()
text = nltk.word_tokenize(text)
y = []
for i in text:
if i.isalnum():
y.append(i)
text = y[:]
y.clear()

for i in text:
if i not in stopwords.words('english') and i not in string.punctuation:
y.append(i)
text = y[:]
y.clear()
for i in text:
y.append(ps.stem(i))
return " ".join(y)
tfidf = joblib.load('vectorizer.pkl')
```

```
model = pickle.load(open('model.pkl','rb'))

st.title("Email Spam Detection")
input_sms = st.text_area("Enter the message")

if st.button('Predict'):
# 1. preprocess
transformed_sms = transform_text(input_sms)
# 2. vectorize
vector_input = tfidf.transform([transformed_sms])
# 3. Predict
result = model.predict(vector_input)[0]
# 4. Display
 if result == 1:
st.header("Spam")
else:
st.header("Not Spam")
```

2. Screenshots

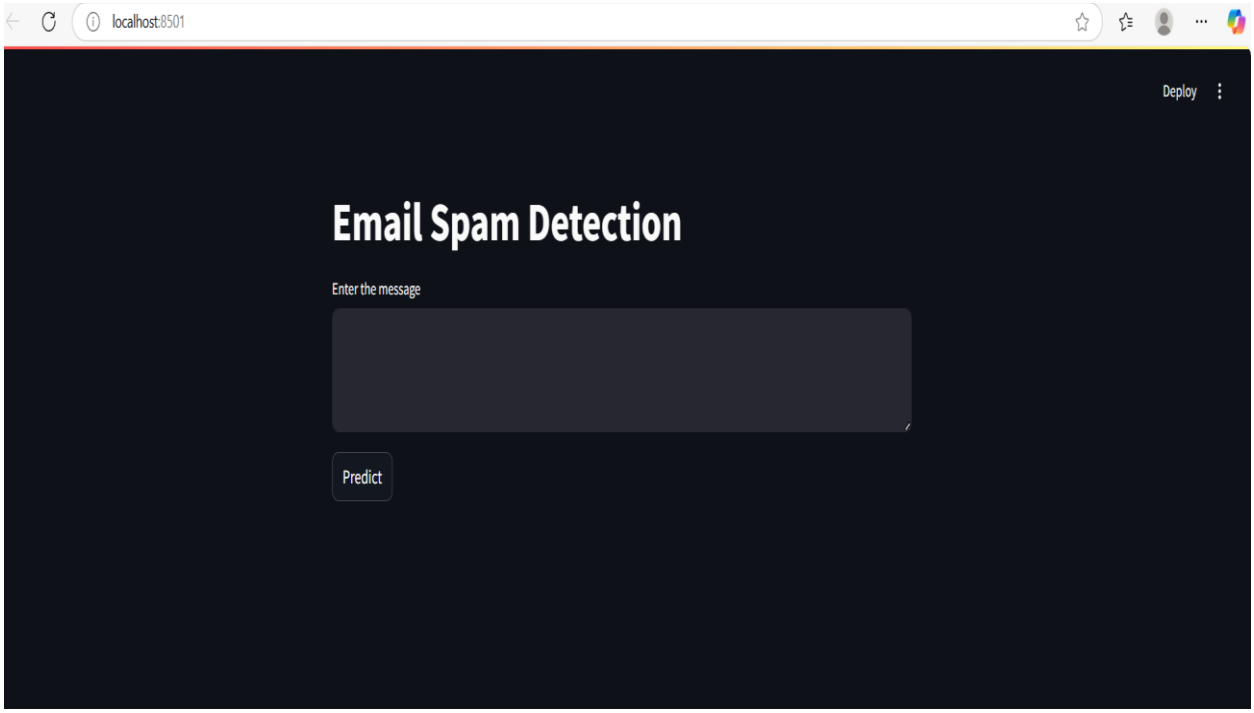

Fig A Hosted on a Server

As shown in the figure, the interface represents the front-end of the Email Spam Detection web application. It is a minimal and user-friendly design where the user is prompted to enter a message into a text box. Once the message is typed in, clicking the "Predict" button sends the input to the backend model, which processes it and classifies the message as either spam or non-spam. This interface is developed using Streamlit, a Python library that allows rapid prototyping of data science and machine learning applications, and it runs locally on the browser through localhost:8501.

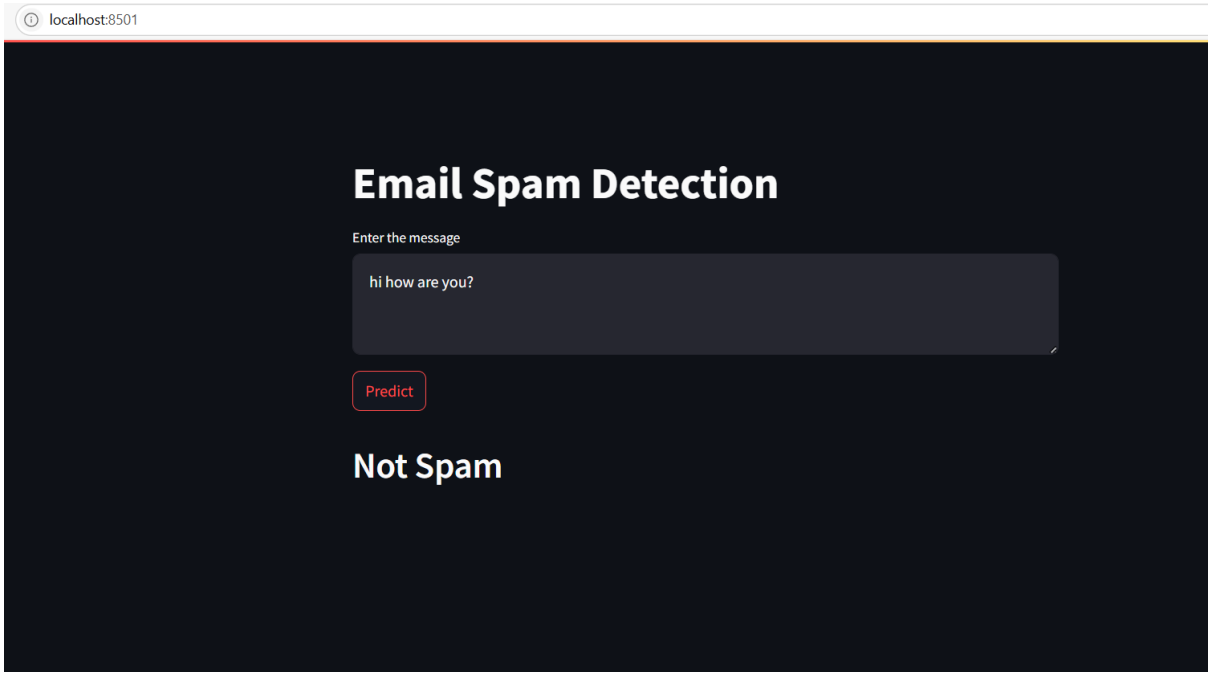

Fig B Prediction Result1

As shown in the figure, the web application titled Email Spam Detection has received a user input message, “hi how are you?” in the provided text box. Upon clicking the "Predict" button, the system processes the message through a trained machine learning model and displays the prediction result as "Not Spam". This output indicates that the message is considered safe and not a spam message. The interface is designed for simplicity and clarity, making it easy for users to test the model by entering any email text and instantly seeing the classification result on the same page.

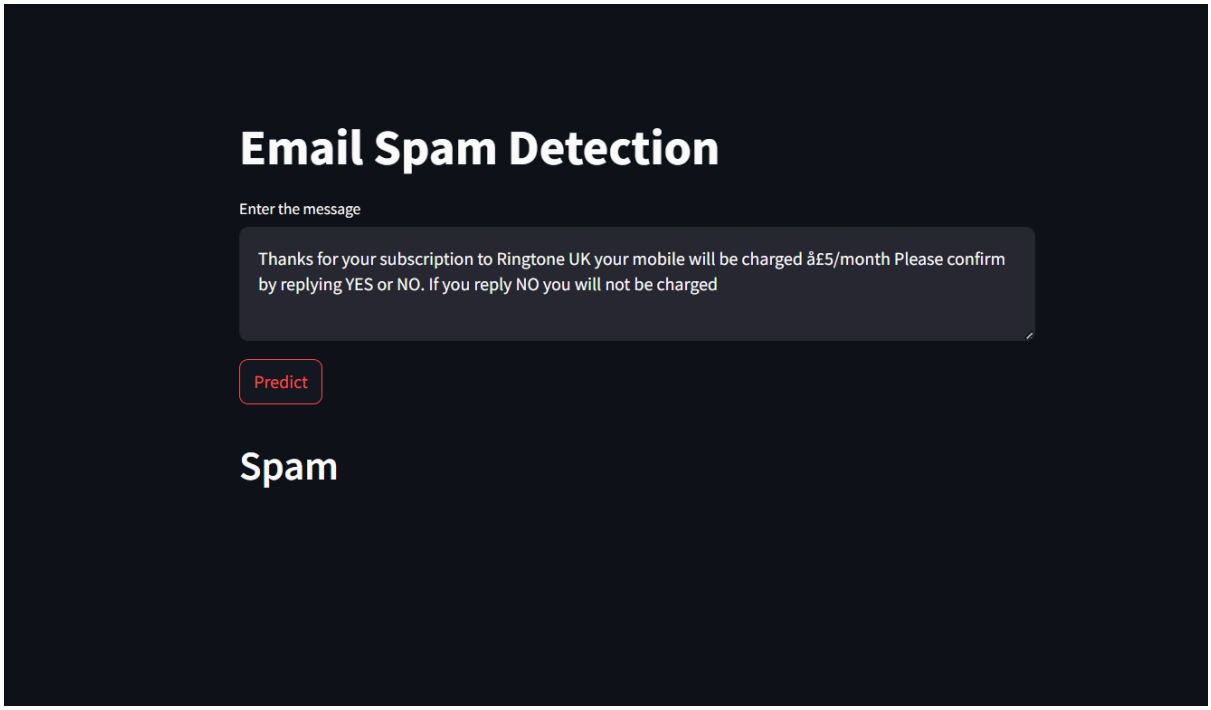

Fig C Prediction Result2

As shown in the figure, the Email Spam Detection system features a simple interface where users can input a message to determine whether it is spam. In this example, the user has entered a message about a subscription to "Ringtone UK" that threatens a monthly charge and requests confirmation by replying "YES" or "NO"—a format commonly used in spam or scam messages. Upon clicking the "Predict" button, the system analyzes the text and displays the result as "Spam", indicating that the message was identified as unwanted or potentially harmful. This demonstrates the system's ability to detect suspicious content and help users avoid falling victim to fraudulent communications.

REFERENCES

[1]. C. Bansal and B. Sidhu, ‘‘Machine learning based hybrid approach for email spam detection’’, in Proc. 9th Int. Conf. Rel., INFOCOM Technol. Optim., Sep. 2021, pp. 1–4.

[2]. Le, H. V., Nguyen, M. T., & Nguyen, T. T. (2018). “Email spam detection based on ensemble learning of extreme learning machine”, International Journal of Machine Learning and Cybernetics, 9(4), 591-602.

[3]. M. Al-Sarem, M. Al-Hadhrami, A. Alshomrani, et al., "Deep Learning for Spam Detection", Expert Systems with Applications, Elsevier, vol. **167**, pp. **113872**, 2021.

[4]. M. A. Shafi, H. Hamid, E. G. Chiroma, J. S. Dada, and B. Abubakar, "Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems", in Proceedings of the International Conference on Artificial Intelligence and Machine Learning (AIML), pp. **45-56**, 2018.

[5]. M. Almeida, T. A. Almeida, and A. Silva, "Spam Email Detection Using Deep Learning Techniques", in Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. **92-105**, 2021..

[6]. narur, H. Jain, G. S. Rao, et al., "ML-Based Spam Mail Detector", Springer Journal of Machine Learning and Applications, vol. **27**, pp. **89-104**, 2023.

[7]. S. Jamal and H. Wimmer, "Improved Transformer-Based Spam Detection", Journal of Artificial Intelligence Research (JAIR), vol. **35**, pp. **120-135**, 2023.

[8]. S. Zavrak and S. Yilmaz, "Hybrid Deep Learning for Email Spam Detection", IEEE Transactions on Neural Networks and Learning Systems, vol. **34**, no. **6**, pp. **987-999**, 2022.

[9]. V. S. Tida and S. Hsu, "Universal Spam Detection with Transfer Learning", in Proceedings of the ACM Conference on Machine Learning (ACM-ML), pp. **230-242**, 2022.