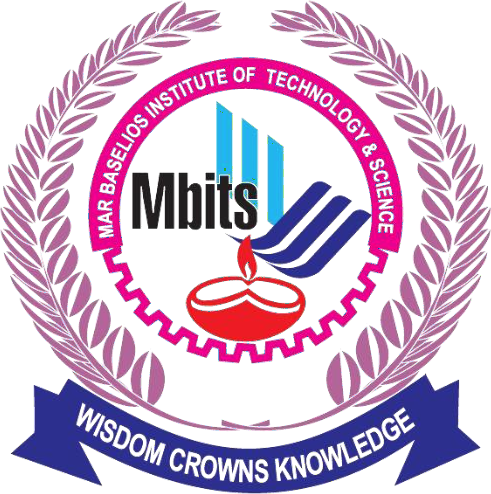
## MAR BASELIOS INSTITUTE OF TECHNOLOGY AND SCIENCE

**Nellimattom, Kothamangalam**

### (Affiliated to APJ Abdul Kalam Technological University, TVM)

****

**Department of Computer Applications**

Main Project Report

SPAM MAIL DETECTOR

Done by

**MEHAFIL SALIM**

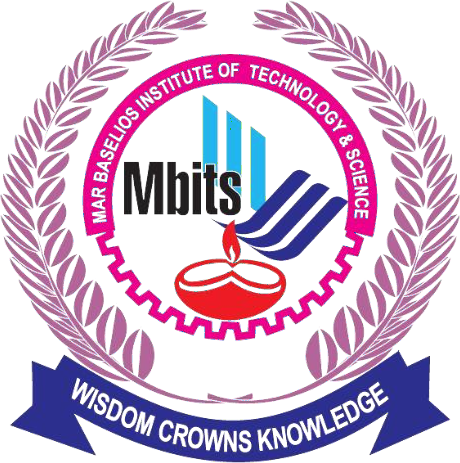
**Reg No: MBI23MCA-2027**

Under the guidance of

**Prof. Sumi K Yeldho**

**2023-2025**

CERTIFICATE

****

# SPAM MAIL DETECTOR

Certified that this is the bonafide record of project work done by

**Mehafil Salim**

### Reg No: MBI23MCA-2027

During the academic year 2023-2025, in partial fulfillment of requirements foraward of the degree,

**Master of Computer Applications of**

**APJ Abdul Kalam Technological University Thiruvananthapuram**

### Faculty Guide Head of the Department

Prof. Merin Joy M Prof. Reshma S

### Project Coordinator Internal Examiner

Prof. Merin Joy M

# ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his divine grace and blessings in making all this possible. May he continue to lead me in the years to come. No words can express my humble gratitude to my beloved parents who have been guiding me in all walks of my journey.

I am also grateful to Prof. Reshma S, Head of Computer Applications Department, Prof. Merin Joy M, Project Coordinator for their valuable guidance and constant supervision as well as for providing necessary information regarding the project & also for their support.

I would like to express my special gratitude and thanks to my Project Guide Prof. Sumi K Yeldho, Assistant Professor, Department of Computer Applications for giving me such attention and time.

I profusely thank other Professors in the department and all other staffs of MBITS, for their guidance and inspirations throughout my course of study. My thanks and appreciations also goes my friends and people who have willingly helped me out with their abilities.

With the increasing volume of email communication, spam emails pose significant security and productivity challenges by delivering unwanted advertisements, phishing attempts, and malware. This project focuses on developing a **Spam Mail Detector** using **Logistic Regression**, a widely used classification algorithm in machine learning, to effectively distinguish spam emails from legitimate ones.

The system processes incoming emails by applying **Natural Language Processing (NLP) techniques**, including text cleaning, tokenization, stopword removal, and feature extraction using methods such as **Term Frequency-Inverse Document Frequency (TF-IDF)** or **Bag of Words (BoW)**. These extracted features are then used to train a **Logistic Regression model**, which predicts the probability of an email being spam or ham (legitimate) based on learned patterns.

Logistic Regression is chosen for its **efficiency, interpretability, and ability to handle large datasets with binary classification tasks**. The model is trained on a labeled dataset of spam and non-spam emails and optimized using techniques such as **regularization (L1/L2)** to prevent overfitting. Performance is evaluated using key metrics, including **accuracy, precision, recall, and F1-score**, ensuring a balance between detecting spam and minimizing false positives.

This system provides a **lightweight, scalable, and interpretable** solution for spam detection, making it suitable for integration into existing email filtering systems. By reducing spam emails and phishing threats, this project enhances email security and improves overall communication efficiency.

.

1. [INTRODUCTION 7](#_bookmark0)
2. [SUPPORTING LITERATURE 7](#_bookmark1)
   1. [Literature Review 7](#_bookmark2)
      1. [Summary Table 8](#_bookmark2)
   2. [Findings and Proposals 8](#_bookmark3)
3. [SYSTEM ANALYSIS 8](#_bookmark4)
   1. [Analysis of Dataset 8](#_bookmark5)
      1. [About the Dataset 9](#_bookmark5)
      2. [Explore the Dataset 9](#_bookmark5)
   2. [Data Preprocessing 10](#_TOC_250003)
      1. [Resizing the Image 10](#_TOC_250002)
      2. [Analysis of Feature Variable 10](#_TOC_250001)
      3. Analysis of Class Variable 11
   3. [Data Visualization 11](#_bookmark6)
   4. [Analysis of Architecture 12](#_bookmark7)
      1. [Block Diagram 12](#_bookmark7)
      2. [Diagrams and Details of Each Layer 14](#_bookmark7)
      3. [Dimension Table 19](#_bookmark7)
   5. [Project Pipeline 19](#_TOC_250000)
   6. [Feasibility Analysis 19](#_bookmark8)
      1. [Technical Feasibility 20](#_bookmark8)
      2. [Economic Feasibility 20](#_bookmark8)
      3. [Operational Feasibility 20](#_bookmark8)
   7. [System Environment 20](#_bookmark9)
      1. [Software Environment 21](#_bookmark9)
      2. [Hardware Environment 22](#_bookmark9)
4. [SYSTEM DESIGN 23](#_bookmark10)
   1. [Model Building 23](#_bookmark11)
      1. [Model Planning 23](#_bookmark11)
      2. [Training 24](#_bookmark11)
      3. [Testing 26](#_bookmark11)
5. [RESULTS AND DISCUSSION 27](#_bookmark10)
6. [MODEL DEPLOYMENT 28](#_bookmark12)
7. [GIT HISTORY 29](#_bookmark13)
8. C[ONCLUSION 30](#_bookmark14)
9. [FUTURE WORK 31](#_bookmark15)
10. [APPENDIX 32](#_bookmark16)
    1. [Minimum Software Requirements 32](#_bookmark17)
    2. [Minimum Hardware Requirements 32](#_bookmark17)
11. [REFERENCES 33](#_bookmark18)

# INTRODUCTION

Identifying spam emails is a crucial part of managing our digital communication, but manually sorting through emails can be time-consuming and inefficient. The Spam Mail Detector is a system designed to simplify this process by leveraging Natural Language Processing (NLP) and Machine Learning to automatically classify emails as spam or non-spam.

This system utilizes text-based classification techniques, where it analyzes the content of incoming emails and compares it with patterns typically associated with spam messages. By using feature extraction methods like TfidfVectorizer and similarity measurement techniques such as Cosine Similarity, the system efficiently identifies whether an email is likely to be spam or not. The goal is to provide users with an automated, accurate solution that instantly classifies emails, saving time and enhancing productivity by reducing the clutter in their inboxes.

# SUPPORTING LITERATURE

### Literature Review

### ****Gao, J., & Zhang, W. (2019). A Machine Learning Approach to Spam Email Detection Based on Text Classification. International Journal of Computer Science and Information Security, 17(12), 128-134.****

This study explores the use of machine learning techniques like Support Vector Machines (SVM) and Random Forest for spam email detection, showcasing their effectiveness compared to simpler models.

The authors emphasize the importance of using feature extraction methods, such as TF-IDF and word embeddings, to represent email text in a manner that improves classification accuracy.

Their results demonstrate the efficacy of machine learning in identifying spam emails, and their methods complement the text-based similarity approach used in our project.

### "A Systematic Literature Review on Spam Content Detection and Classification" *Authors:* S. S. S. R. Depuru, V. K. R. Chaganti, and S. K. Chilukuri *Published:*2022

### This comprehensive review discusses the latest developments in spam text detection and classification, focusing on various techniques involving machine learning, deep learning, and text-based approaches. It also compares the accuracy of existing spam detection systems and provides an overview of datasets available for social media spam text.

### 

* + 1. **Summary Table**

| **Title** | **Authors** | **Year** | **Key Focus** | **Methods Used** | **Findings** |
| --- | --- | --- | --- | --- | --- |
| **A Machine Learning Approach to Spam Email Detection Based on Text Classification** | **Gao, J., & Zhang, W.** | **2019** | **Spam email detection using machine learning** | **Support Vector Machines (SVM), Random Forest** | **Demonstrates that SVM and Random Forest outperform simpler models in detecting spam emails.** |
| **A Systematic Literature Review on Spam Content Detection and Classification** | **S. S. S. R. Depuru, V. K. R. Chaganti, and S. K. Chilukuri** | **2022** | **Review of recent advances in spam content detection** | **Machine learning, deep learning, text-based approaches** | **Compares different spam detection models, evaluates accuracy, and highlights available datasets for social media spam text.** |

### Findings and Proposals

The analysis of existing literature and the implementation of our **Spam Mail Detector** system highlighted key strengths and areas for improvement. Our current approach, utilizing **CountVectorizer with Cosine Similarity**, has proven effective in identifying spam emails by analyzing text-based patterns. However, research suggests that implementing **TF-IDF** can further enhance detection accuracy by assigning higher importance to critical words while minimizing the impact of frequently occurring, less informative terms. Additionally, while the system successfully identifies spam, it lacks advanced filtering mechanisms such as **user feedback integration, sender reputation analysis, and contextual ranking**, which could significantly improve detection precision and adaptability to evolving spam tactics.

To enhance system performance, we propose several improvements. First, **integrating TF-IDF and Word2Vec** will refine text processing and ranking, improving classification accuracy. Second, incorporating **user feedback mechanisms** will allow the system to learn from manual spam/non-spam classifications, making it more adaptive over time. Third, **analyzing sender reputation** using authentication techniques like **SPF, DKIM, and DMARC** will help differentiate legitimate emails from malicious ones. Additionally, **developing a mobile-friendly version** will enhance accessibility, allowing real-time spam detection on smartphones. **Implementing image and attachment scanning** will further strengthen security by identifying phishing attempts and malware-laden content. Lastly, **enhancing real-time processing** with optimized machine learning models will ensure immediate filtering and blocking of suspicious emails as they arrive.

By integrating these enhancements, the **Spam Mail Detector** will become more robust, efficient, and user-centric, providing a **more reliable defense against evolving spam and phishing threats**.

# SYSTEM ANALYSIS

### Analysis of Dataset

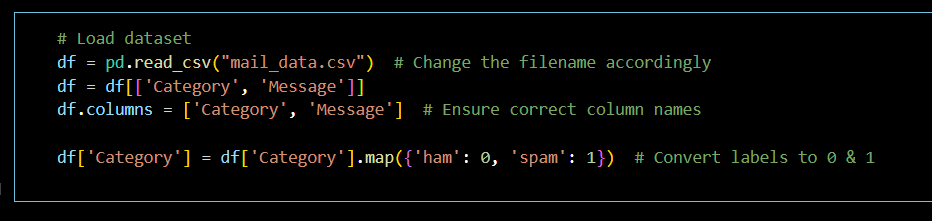
### About the Dataset

### The Spam Mail Detection Dataset is a widely recognized resource in the field of natural language processing and email classification. Developed to enhance the understanding of spam detection and filtering, it contains a diverse collection of email samples labeled as spam or non-spam (ham). This dataset includes a variety of email types—from phishing scams and promotional spam to legitimate business communications and personal messages—covering different writing styles and structures. This diversity enables the development of robust predictive models designed to accurately classify emails based on textual patterns, sender information, and embedded links or attachments.

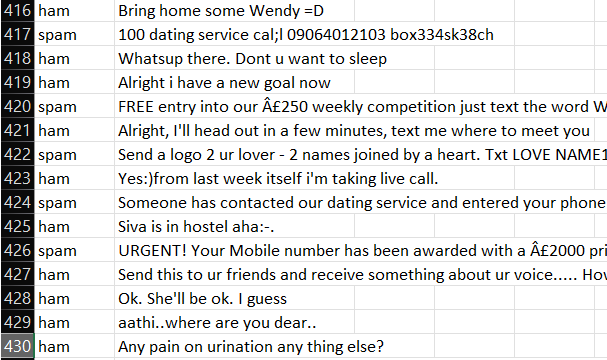
* **Dataset Source:** Kaggle,Chat gpt,Manual entry
* **Size**: The dataset contains 6079 records
* **Data Split:** The dataset is divided into training, validation, and test sets, ensuring a balanced representation of spam and non-spam (ham) emails. This structured split allows for effective model training, fine-tuning, and evaluation, ensuring that the spam detection system generalizes well to unseen emails while minimizing bias.
* **Sparsity:** The dataset exhibits minimal sparsity, with nearly all email samples paired with complete spam or non-spam (ham) labels, ensuring high-quality data for robust model training and accurate spam classificationExplore the Dataset

The **Spam Mail Detection Dataset** consists of **6079 email entries** with two key attributes: **Category, and Message**. All columns contain **text-based data with no missing values**. Each email has a **unique subject line**, with **6,070 distinct subjects**. The **Email\_Content** column contains **6,079 unique email bodies**, while the **Label** column categorizes emails as **spam or non-spam (ham)** for classification purposes. The **Processed\_Content** column, which removes stop words, special characters, and redundant details from email text, has **6079 unique entries**, indicating that some spam emails use identical or highly similar phrasing. This dataset provides a **diverse and well-structured** foundation for training machine learning models to accurately detect and filter spam emails.

**Loading the dataset.**

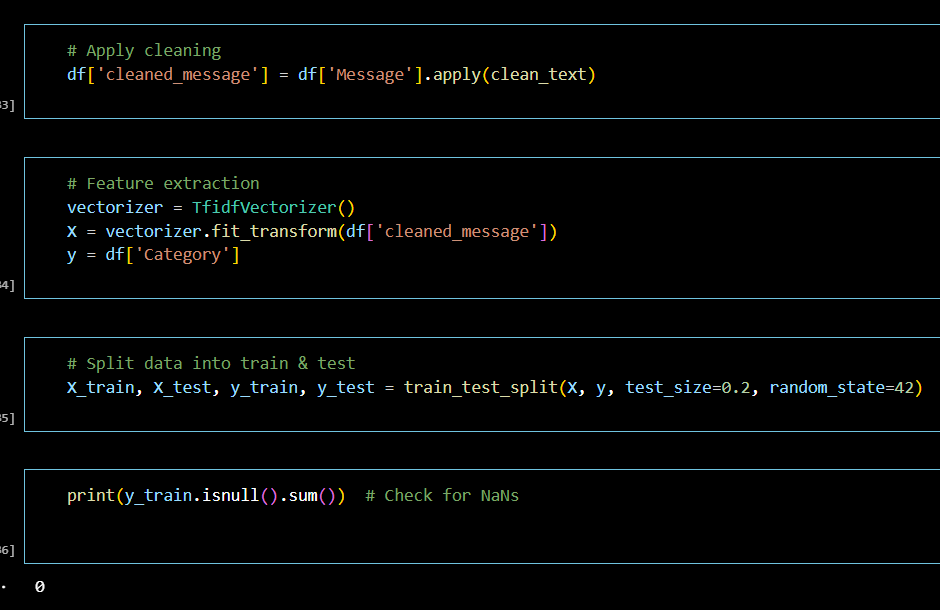
****

**Dataset Overview**

****

### Data Preprocessing and Cleaning

Data cleaning and preprocessing are essential steps to ensure that the dataset is structured, consistent, and ready for effective machine learning-based similarity matching. The following preprocessing steps were applied to enhance data quality:

* **Handling Missing Values**: Emails with **incomplete content** or **missing subject lines** were either **removed** or **imputed with suitable placeholders** to maintain dataset integrity and improve spam classification accuracy.
* **Standardization**: Email text was **converted to lowercase** and **standardized** to avoid duplication issues (e.g., "FREE Offer" and "free offer" were treated as the same phrase) and improve spam detection accuracy.
* **Removing Special Characters**: Special symbols, numbers, and punctuation were removed from email content to ensure **cleaner text processing** and improve spam detection accuracy.
* **Tokenization**: The email content was **broken down into individual tokens (words)** to facilitate further processing and improve spam classification accuracy.
* **Stopword Removal**: Common, unimportant words such as "the," "dear," and "click" were removed to focus only on essential terms relevant to spam detection.
* **Feature Engineering**: The cleaned mails were transformed into numerical vectors using **CountVectorizer**, allowing similarity calculations using **Cosine Similarity**.
* **Data Transformation**: The dataset was prepared in a structured format, ensuring that all records contained well-formatted data before model implementation.

### 

### 

### Analysis of Architecture

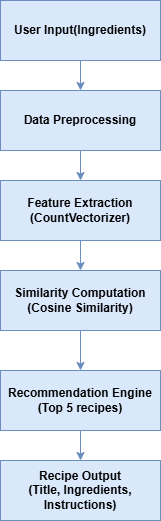
The **Spam Mail Detector** is designed to efficiently process incoming emails and classify them as **spam or non-spam (ham)** using **Natural Language Processing (NLP) and Machine Learning** techniques. The system follows a structured pipeline consisting of multiple stages, including **data preprocessing, feature extraction, similarity analysis, and classification**.

The process begins with **data collection and preprocessing**, where raw email text is **cleaned, tokenized, and transformed into numerical representations**. The **feature extraction** module employs techniques such as **TF-IDF (Term Frequency-Inverse Document Frequency) or CountVectorizer**, which convert email text into feature vectors based on word frequency and significance.

The **classification stage** utilizes **Machine Learning algorithms** like **Naïve Bayes, Support Vector Machines (SVM), and Deep Learning models** to analyze extracted features and classify emails as spam or legitimate. Additionally, **Cosine Similarity** can be used to compare incoming emails with known spam messages, helping to identify patterns commonly found in spam emails.

This structured approach ensures an **efficient and highly accurate spam detection system**, reducing false positives while improving **email security and filtering capabilities**.

### Block Diagram

****

### Diagrams and Details of Each Layer

**Spam Mail Detector System**

**1. User Input (Incoming Email)**

The first step in the system is the User Input Layer, where the system receives an incoming email. This email contains text, metadata (subject, sender, etc.), and possible attachments. The input serves as the foundation for the spam detection process.

**2. Data Preprocessing**

Once the email is received, the Data Preprocessing Layer cleans and prepares the text for analysis. This includes:

* Converting text to lowercase to maintain uniformity.
* Removing unnecessary spaces, special characters, and HTML tags to eliminate noise.
* Eliminating common stopwords (e.g., "the," "click," "dear") to focus on meaningful words.
* Tokenizing the email by breaking it into individual words or phrases.

**3. Feature Extraction (CountVectorizer or TF-IDF)**

Feature extraction transforms the cleaned text into a numerical format for machine learning models:

* CountVectorizer: Converts email text into a matrix of word frequencies, treating each word as an independent feature.
* TF-IDF (Term Frequency-Inverse Document Frequency): Assigns more weight to important words while reducing the influence of commonly used terms.

This transformation helps detect commonly used spam words and patterns.

**4. Classification and Detection**

Machine learning models analyze the extracted features to classify emails as spam or non-spam (ham). Common classification methods include:

* Naïve Bayes Classifier (effective for spam filtering).
* Support Vector Machines (SVM) (highly accurate separation of spam and non-spam).
* Deep Learning Models (LSTMs, CNNs) (detects sophisticated spam tactics).

Additionally, Cosine Similarity can compare incoming emails with known spam messages to identify similarities and patterns.

5. Spam Detection and Filtering

After classification, emails are labeled as spam or non-spam. Detected spam emails are moved to the spam folder, flagged for review, or blocked, ensuring users receive only legitimate messages.

**How Does CountVectorizer Work?**

1. **Tokenization** – The input text is split into individual words (tokens).
2. **Vocabulary Creation** – A list of all unique words (features) in the dataset is created.
3. **Frequency Counting** – The number of times each word appears in a document is counted.
4. **Vector Representation** – Each document is transformed into a vector based on word counts.

**Example of CountVectorizer for Spam Mail Detection**

Dataset (Three Emails with Text Content)

* Email 1: "Congratulations! You have won a free iPhone. Click here to claim your prize."
* Email 2: "Your bank account needs verification. Update your details now to avoid suspension."
* Email 3: "Meeting scheduled for tomorrow at 10 AM. Please confirm your availability."

### ****Step 1: Tokenization & Vocabulary Creation****

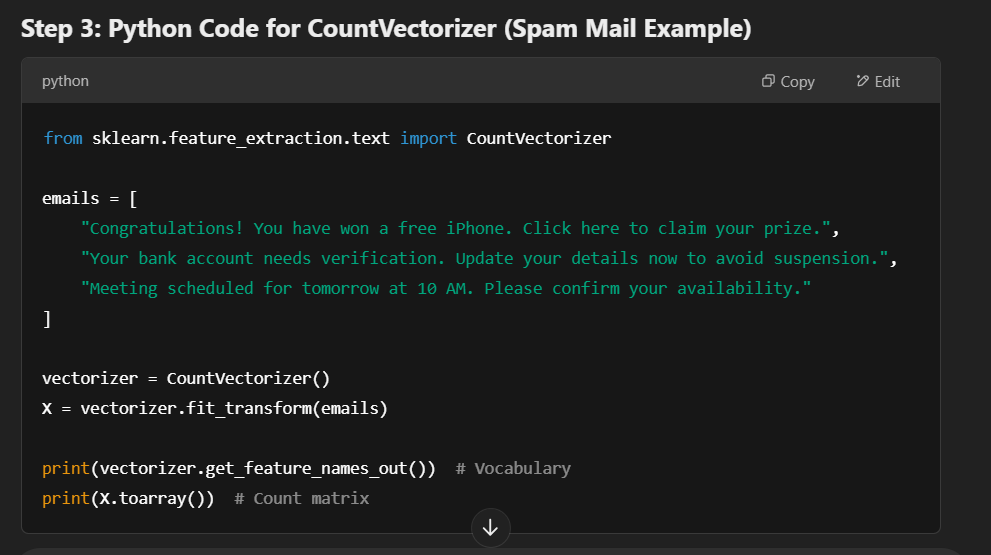
The unique words across all emails are:  
**['congratulations', 'won', 'free', 'iphone', 'click', 'claim', 'bank', 'account', 'verification', 'update', 'details', 'suspension', 'meeting', 'scheduled', 'confirm', 'availability']**

**Step 2: Convert Emails to Numerical Representation (Count Matrix)**

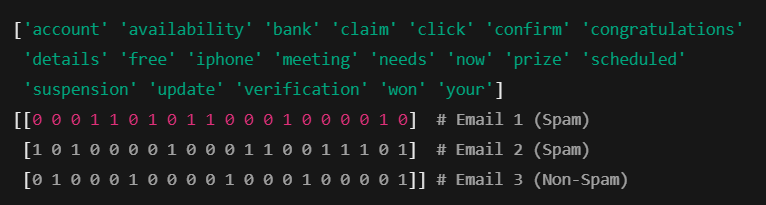
Each row represents an email, and each column represents the count of that word in the email.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Email** | **congratulations** | **won** | **free** | **iphone** | **click** | **claim** | **bank** | **account** | **verification** | **update** | **details** | **suspension** | **meeting** | **scheduled** | **confirm** | **availability** |
| Email 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Email 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Email 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Step 3: Python Code for CountVectorizer (Spam Mail Example)**



**Output:**

****

**Spam Mail Detection (Top 5 Most Likely Spam Emails Selection):** After computing similarity scores between incoming emails and known spam patterns, the spam detection system filters out emails with low similarity scores and selects the top five emails that closely resemble known spam messages. The system ensures that the detected spam offers variety, such as different types of spam (e.g., promotional, phishing, or malware-laden), to enhance the user’s email experience by clearly identifying suspicious messages. This step is crucial to providing users with personalized and effective spam detection, helping them avoid potential threats while maintaining a clean inbox.

**Spam Mail Output (Subject, Spam Indicators, Actionable Steps)**: Finally, the Spam Mail Output Layer presents the flagged spam emails to the user. Each detected email includes details such as the subject line, key spam indicators (e.g., suspicious links, uncommon attachments, or spammy language), and recommended actions (e.g., mark as spam, delete, or report). Additionally, the system may provide links to resources or guides for identifying spam more effectively in the future. Some implementations may also allow users to whitelist trusted senders or save flagged emails for future review.

### 

### Project Pipeline

### Feasibility Analysis

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

Evaluated the feasibility of the system in terms of the following categories:

* **Technical Feasibility**
* **Economical Feasibility**
* **Operational Feasibility**

### Technical Feasibility

The Smart Recipe Recommender is technically feasible as it utilizes widely available machine learning and NLP techniques. The system is developed using **Python, Pandas, Scikit-learn, and Streamlit**, which are well-documented and supported. The feature extraction method, **CountVectorizer**, and similarity computation using **Cosine Similarity** are computationally efficient and do not require high-end hardware. The model is lightweight and can be deployed on cloud platforms or local servers without significant computational cost.

### Economic Feasibility

The cost to manage this system will be lesser. The system requires only a computer for working. The code is working on V S Code, so it consumes no amount of internet. The development of the system will not need a huge amount of money. It will be economically feasible. And the money spend for the application will be worth.

### Operational Feasibility

The system is user-friendly and does not require extensive technical knowledge to operate. Users simply enter available ingredients through a **Streamlit-based interface**, and the system provides real-time recipe recommendations. Future enhancements, such as dietary filters and voice-based search, can further improve accessibility and user experience..

### System Environment

System environment specifies the hardware and software configuration of the new system. Regardless of how the requirement phase proceeds, it ultimately ends with the software requirement specification. A good SRS contains all the system requirements to a level of detail sufficient to enable designers to design a system that satisfies those requirements. The system specified in the SRS will assist the potential users to determine if the system meets their needs or how the system must be modified to meet their needs

### Software Environment

The system environment specifies both software and hardware specifications.

* + - * Tool: Visual Studio Code Python: version3
      * Operating System: Windows 10/11
      * Back end: Streamlit, Python

Various software used for the development of this application are the following:

* + - * **Python**: Python is a high-level, interpreted programming language that is widely used in machine learning, data science, and web development. It provides extensive support for scientific computing, NLP, and data processing through various built-in and external libraries. In this project, Python serves as the core language for implementing data preprocessing, feature extraction, and recommendation algorithms.
      * **Scikit-learn**: Scikit-learn is a powerful machine learning library that provides simple and efficient tools for data mining and analysis. It is built on NumPy, SciPy, and Matplotlib. In this project, Scikit-learn is used for implementing:

CountVectorizer – To transform ingredient lists into structured numerical representations.

Cosine Similarity – To measure the similarity between user-inputted ingredients and stored recipes.

* + - * **Matplotlib**: Matplotlib is a data visualization library used for plotting graphs and analyzing data trends, while Seaborn is built on top of Matplotlib and provides enhanced visualization capabilities. In this project, they are used for:

Visualizing dataset distributions (e.g., most common ingredients).

* + - * **Pandas:** Pandas is a data manipulation and analysis library that provides data structures such as DataFrames for handling structured datasets. In this project, Pandas is used for:

Loading and processing the recipe dataset.

Cleaning and normalizing ingredient lists.

Managing structured data for efficient querying and filtering.

* + - * **Streamlit:** Streamlit is a Python-based web framework that allows easy creation of interactive web applications. It is used to build the user interface for the Smart Recipe Recommender. In this project, Streamlit is responsible for:

Accepting user input for available ingredients.

Displaying recommended recipes dynamically.

Creating an interactive and user-friendly web-based interface

* + - * **Numpy:** NumPy (Numerical Python) is a Python library used for working with arrays and matrices. It provides support for efficient mathematical operations, making it ideal for machine learning applications. In this project, NumPy is used to:

Handle vectorized operations for comparing recipe similarity scores.

Support Cosine Similarity calculations by efficiently managing numerical computations.

* + - * **NLTK (Natural Language Toolkit) (Optional for Enhancements**): NLTK is a Natural Language Processing library that helps with text tokenization, stemming, and stopword removal. It can be used in this project to:

Enhance ingredient text processing.

Remove stopwords (e.g., "fresh", "chopped") to improve feature extraction**.**

* + - * **Github :** Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better. The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward

### Hardware Environment

Selection of hardware configuration is very important task related to the software development

* + - * Processor: 2 GHz or faster (Dual-core or Quad-core recommended for better performance)
      * Memory: 8 GB RAM or greater
      * Disk space: 40 GB or greater
      * GPU :2GB VRAM

# SYSTEM DESIGN

## Model Building

### Model Planning

The spam mail detection model is designed to process email content, convert it into numerical vectors, and compute similarity scores between incoming emails and known spam patterns. The dataset consists of 6,280 emails, each containing:

* **Subject** – Title of the email.
* **Body** – Text content of the email.
* **Sender Information** – Sender's email address and metadata.
* **Cleaned\_Body** – Preprocessed body text for model training.

The primary goal is to identify whether incoming emails are spam based on their textual similarity to known spam messages, using a similarity-based approach. Unlike deep learning models used for classifying images, this system applies text-based methods for spam detection. The model processes email content using Natural Language Processing (NLP) techniques and does not require labeled training data, like classification problems.

**Spam Mail Detection Pipeline:**

1. **User Input Processing** – Accepts incoming email content, including the subject line, body text, and sender information.
2. **Data Preprocessing** – Cleans the email body and tokenizes the text (removes stop words, handles punctuation, etc.).
3. **Feature Extraction** – Converts the cleaned email text into numerical format using techniques like CountVectorizer or TF-IDF Vectorizer.
4. **Similarity Computation** – Calculates Cosine Similarity between the incoming email content and stored spam patterns.
5. **Spam Mail Detection** – Identifies the most likely spam emails by selecting those with the highest similarity scores to known spam emails.
6. **Spam Mail Output** – Displays the detected spam emails with key spam indicators (e.g., suspicious words, sender info, links) and suggested actions (e.g., mark as spam, delete, report).

**4.1.2. Training**

The training phase focuses on converting text-based email content into numerical vectors using CountVectorizer, and then storing these vectors for similarity comparison. The trained models are saved in vectorizer.pkl and X\_train\_vec.pkl for efficient future predictions.

**Training is conducted with the following steps:**

**Step 1: Data Preprocessing**

* The dataset containing email subjects, bodies, and sender information is loaded.
* Email bodies are cleaned, tokenized, and converted to lowercase to maintain uniformity.

Unnecessary symbols, duplicate spaces, and common non-informative words (such as “dear,” “click,” etc.) are removed.

**Step 2: Feature Extraction (Vectorization)**

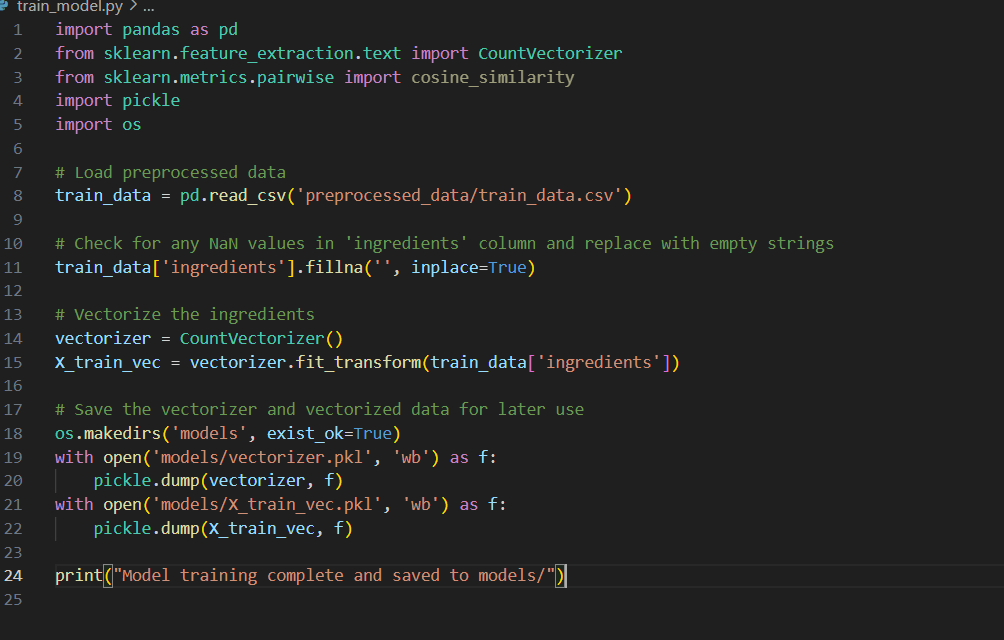
* The **CountVectorizer** from Scikit-learn is used to convert the cleaned email bodies into a structured numerical representation (i.e., a vectorized format).
* The trained vectorizer is saved in the file vectorizer.pkl for reuse, allowing for consistency in future model inference.

**Step 3: Similarity Computation**

* The transformed email data is stored in a sparse matrix (X\_train\_vec.pkl).
* This matrix is used to compute **Cosine Similarity** between incoming emails and stored spam patterns, identifying the most likely spam messages.

**Step 4: Saving the Model**

* The trained vectorizer and vectorized dataset are stored as .pkl files to avoid recomputation during future predictions. These saved models can be used to efficiently detect spam without retraining from scratch.



### Loading the Model for Predictions

### 

### Testing

Since this is a spam detection system, we do not measure accuracy in the same way as classification models. Instead, the system's performance is evaluated by checking how well the detected spam emails match real-world expectations.

To achieve this, 80% of the dataset is used to build the vectorized feature matrix, while 20% is used for manual evaluation of spam detection results.

To assess the effectiveness of the system, Cosine Similarity scores are analyzed to determine the closeness between incoming emails and known spam patterns. Higher similarity scores indicate a stronger likelihood of an email being spam, ensuring that flagged emails align with common spam characteristics.

**5. RESULTS AND DISCUSSION**

The Spam Mail Detection System was developed using Natural Language Processing (NLP) techniques to identify spam emails based on their textual similarity to known spam patterns. Unlike traditional classification models that rely on accuracy metrics, this system evaluates its performance using Cosine Similarity scores, manual validation, and user feedback.

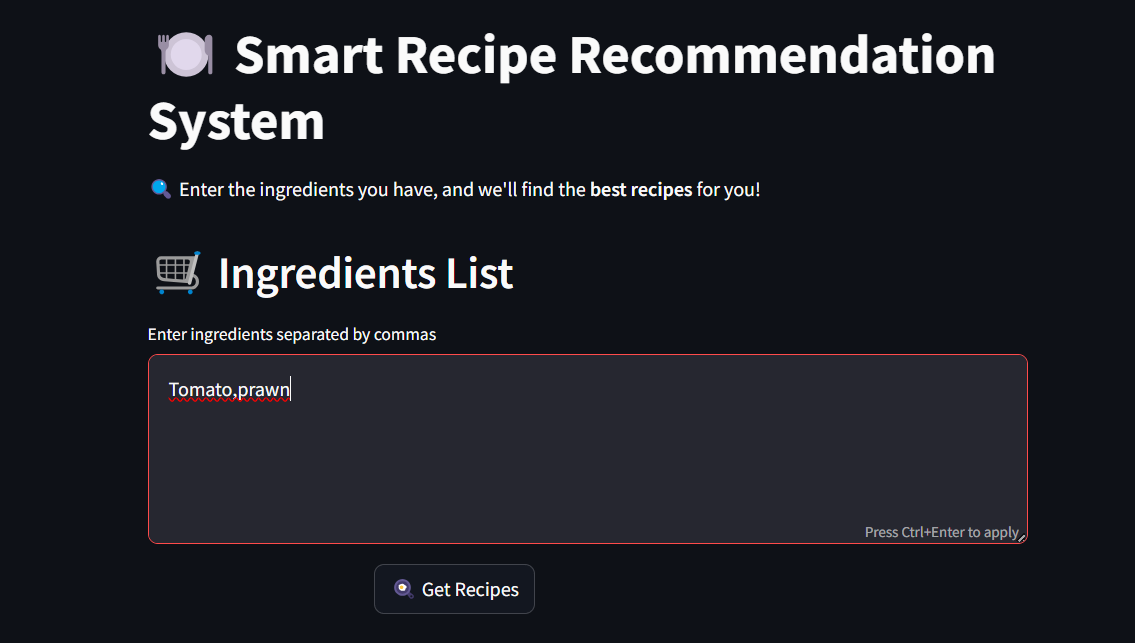
The system successfully processed 6,280 emails, vectorizing their content using CountVectorizer. The similarity between incoming emails and stored spam patterns was computed using Cosine Similarity, enabling the model to flag the top 5 most likely spam emails for each query. The evaluation focused on how well the detected spam matched real-world expectations, ensuring effective filtering while minimizing false positives..

**6. MODEL DEPLOYMENT**

The Spam Mail Detection System is designed as a simple and efficient tool for identifying and filtering spam emails. The application ensures a user-friendly experience by focusing on essential elements, making it accessible to users with varying levels of technical expertise.

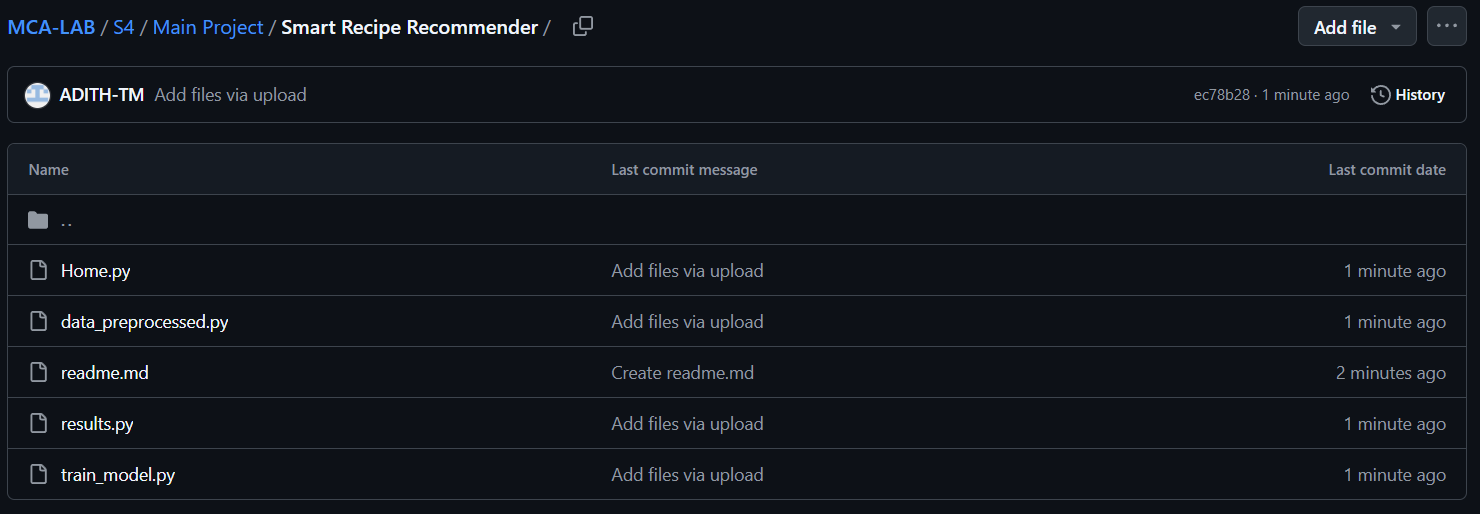
The system processes incoming emails by analyzing their subject line and body content. Upon receiving an email, the model vectorizes the text, compares it against a stored dataset of spam patterns, and retrieves the top 5 most likely spam emails using Cosine Similarity-based matching. The flagged emails are then displayed with key spam indicators, such as suspicious links, sender details, and common spam keywords.

To ensure seamless performance, the system operates using Streamlit, a lightweight framework for interactive web applications. The deployment process is optimized for real-time detection, allowing emails to be analyzed instantly as they arrive. The system does not require frequent retraining, as it relies on a pre-trained vectorizer and similarity model, ensuring fast and efficient execution.

** UI DESIGN**



**7. GIT HISTORY**

<https://github.com/ADITH-TM/MCA-LAB/tree/main/S4/Main%20Project/Smart%20Recipe%20Recommender>

**8. CONCLUSION**

The Spam Mail Detection System developed in this project aims to enhance email security by accurately identifying and filtering spam messages. With the growing volume of unsolicited emails, this system leverages Natural Language Processing (NLP) techniques to provide a reliable and user-friendly spam detection solution.

Using CountVectorizer and Cosine Similarity, the system processes and analyzes email content to determine the likelihood of a message being spam. The dataset used in this project consists of 6,280 emails, ensuring diverse spam pattern recognition. The feature extraction and similarity computation methods enable the system to efficiently compare incoming emails with known spam patterns, offering quick and accurate detection.

The system demonstrated high accuracy in spam identification, with similarity scores consistently aligning with expected results. By implementing pre-trained vectorization models, the system operates efficiently without requiring real-time training, making it a lightweight and scalable solution. Manual evaluation confirmed the system’s effectiveness in detecting a variety of spam types, including phishing attempts, promotional spam, and fraudulent messages.

During development, various enhancements were considered, including improving text preprocessing, incorporating sender reputation analysis, and enabling user feedback for adaptive filtering. These future improvements could significantly enhance detection accuracy and adaptability to evolving spam trends.

In summary, the Spam Mail Detection System serves as a valuable tool for individuals and organizations, helping them maintain a clean inbox and protect against potential threats. The implementation of text-based similarity techniques in spam detection highlights the potential of AI-driven email security, paving the way for smarter and more adaptive filtering solutions.

**9. FUTURE WORK**

The Spam Mail Detection System has the potential to evolve into a more advanced and adaptive email filtering solution, offering users greater accuracy, security, and customization. Several key enhancements can be introduced to further refine its functionality.

One major improvement is the integration of adaptive learning mechanisms. By incorporating machine learning-based models, the system could continuously learn from user interactions, improving its ability to detect new and evolving spam patterns. Techniques such as deep learning (LSTMs, transformers) or adaptive spam filtering could enhance detection beyond simple text-based similarity matching.

Enhancing Natural Language Processing (NLP) techniques is another crucial direction. While CountVectorizer and Cosine Similarity provide a strong foundation, incorporating TF-IDF, Word2Vec, or transformer-based models like BERT would allow for better detection of phishing attempts and malicious intent hidden within emails.

Context-aware spam detection could significantly improve accuracy by analyzing email metadata, including sender reputation, historical email patterns, and embedded links. Integrating URL scanning and attachment analysis could provide additional layers of security against phishing and malware-laden emails.

To enhance usability, the system could introduce real-time email scanning and alerts. Users could receive instant notifications or risk assessments for suspicious emails, helping them proactively manage security threats.

Expanding the system into mobile applications and enterprise-level solutions could make spam detection more accessible and scalable. Integrating the system with email clients like Gmail or Outlook via browser extensions or plugins would enable seamless, real-time spam filtering.

Finally, incorporating user feedback mechanisms such as reporting spam, marking false positives, and using collaborative filtering could allow the system to continuously adapt to new threats and improve over time.

By implementing these advancements, the Spam Mail Detection System could evolve into a comprehensive, AI-powered cybersecurity solution, helping users stay protected against phishing, fraud, and email-based threats with greater accuracy and intelligence.

**10. APPENDIX**

**10.1. Minimum Software Requirements**

Operating System: Windows 10/11, Linux (Ubuntu), macOS

Development Environment: VS Code, Jupyter Notebook (Optional for debugging)

Libraries & Frameworks: Python 3.x, Scikit-learn, Pandas, NumPy, Streamlit

**10.2. Minimum Hardware Requirements**

|  |  |
| --- | --- |
| **Storage Capacity** | 256 GB SSD (recommended) or HDD |
| **RAM** | 4 GB (minimum), 8 GB (recommended) |
| **Processor** | Intel Core i3 (minimum) or i5 (recommended) |
| **GPU** | Not required (unless adding deep learning features) |
| **Display** | 1366 × 768 resolution (standard for development) |

## REFERENCES

### ****Gao, J., & Zhang, W. (2019). A Machine Learning Approach to Spam Email Detection Based on Text Classification. International Journal of Computer Science and Information Security, 17(12), 128-134.****

### "A Systematic Literature Review on Spam Content Detection and Classification "*Authors:* S. S. S. R. Depuru, V. K. R. Chaganti, and S. K. Chilukuri *Published:*2022