A MINI PROJECT REPORT

On

EMAIL SPAM DETECTION USING MACHINE LEARNING ALGORITHMS

Submitted by

G. AKHIL	**22J41A1220**
P. VIVEK	**22J41A1250**
K. SIVA KRISHNA	**22J41A1224**
V.NIKITHA	**22J41A1263**

In partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Under the Guidance of

Ms. B. Siris Royal

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY ENGINEERING COLLEGE

(An UGC Autonomous Institution, Approved by AICTE, New Delhi &Affiliated to JNTUH, Hyderabad Maisammaguda, Secunderabad, Telangana, India 500100)

MAY-2025

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

BONAFIDE CERTIFICATE

This is to certify that this mini project work entitled "**EMAIL SPAM DETECTION USING MACHINE LEARNING ALGORITHM**", submitted by **G.AKHIL (22J41A1220), P.VIVEK (22J41A1250), K.SIVA KRISHNA (22J41A1224), V.NIKITHA (22J41A1263)** to Malla Reddy Engineering College affiliated to **Jawaharlal Nehru Technological University, Hyderabad** in partial fulfilment for the award of **Bachelor of Technology** in **Information Technology** is a Bonafide record of project work carried out under my/our supervision during the academic year **2024 – 2025** and that this work has not been submitted elsewhere for a degree.

Project Guide
Ms. B. Siris Royal
Assistant Professor

Dr.Deena Babu Mandru
Professor and HOD
Department of Information Technology
Malla Reddy Engineering College
Secunderabad-500100

Internal Examiner **External Examiner**

Submitted for Mini Project Viva-Voce Examination held on _____

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad , Telangana, India 500100

DECLARATION

We hereby declare that the project titled "**EMAIL SPAM DETECTION USING MACHINE LEARNING ALGORITHMS**" submitted **to Malla Reddy Engineering College (Autonomous)** and affiliated with JNTUH, Hyderabad, in partial fulfillment of the requirements for the award of a **Bachelor of Technology** in **Information Technology**, represents our ideas in our own words. Wherever others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity, and we have not misrepresented, fabricated, or falsified any idea, data, fact, or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Name of the Student	Roll No	Signature
G. AKHIL	22J41A1220	_____
P. VIVEK	22J41A1250	_____
K. SIVA KRISHNA	22J41A1224	_____
V. NIKITHA	22J41A1263	_____

MALLA REDDY ENGINEERING COLLEGE

Maisammaguda, Secunderabad, Telangana, India 500100

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and Founder of Malla Reddy Group of Institutions, Sri. **Ch. Malla Reddy**, for providing the necessary infrastructure facilities for completing project work successfully.

We express our sincere thanks to our Principal, **Dr.A.Ramaswami Reddy**, who took keen interest and encouraged us in every effort during the project work.

We express our heartfelt thanks to **Dr.Deena Babu Mandru**, Professor and Head of the Department, Department of Information Technology, MREC (A) for all the kindly support and valuable suggestions during the period of our project.

We are extremely thankful to our project coordinator, **Mrs.M.Anusha**, for the constant guidance and support to complete the project work.

We are extremely thankful and indebted to our project guide, **Ms. B. Siris Royal,** Assistant Professor, Department of Information Technology, MREC (A) for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of the IT Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

G. AKHIL	22J41A1220
P. VIVEK	22J41A1250
K. SIVA KRISHNA	22J41A1224
V. NIKITHA	22J41A1263

ABSTRACT

The surge in internet users has intensified the problem of email spam, with millions of unsolicited emails sent daily, facilitating illegal activities like phishing, financial fraud, and malware distribution. Spammers exploit the ease of creating fake email accounts to masquerade as trustworthy sources, making it challenging for users to identify fraudulent emails, which often contain harmful links or attachments that compromise personal data and system security. Traditional rule-based filters are inadequate against evolving spam tactics, necessitating intelligent, automated detection systems. Our project addresses this by developing a machine learning-based spam email detection system that analyzes email content and metadata to distinguish spam from ham. Through extensive testing on a well-known spam email dataset, the Random Forest algorithm demonstrated superior accuracy and consistency. As an ensemble method, Random Forest combines multiple decision trees to deliver stable, accurate predictions, effectively handling high-dimensional data and minimizing overfitting compared to single decision trees.

Keywords: *Spam Email, Machine Learning, Spam, Detection, Random Forest, Email Classification, Accuracy, Decision Tree*

PROJECT ACKNOWLEDGEMENT

The work “would not have been possible” without the contribution of the university. we indebted to teachers who have offered continuous support while preparing the project.

We also grateful to all those "with whom" we had the opportunity to do the work and complete the project. 'Each member" of the "dissertation committee" have offered and provided' professional guidance" and have given me great advice while completing the project.

On a personal note, we also grateful to my family members who have offered me continuous support while completing the project. Without help and support from them, this project would not have been completed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

KNN	K-NEAREST NEIGHBOR
DT	DECISION TREE
RF	RANDOM FOREST
SVM	SUPPORT VECTOR MACHINE
LR	LOGISTIC REGRESSION
NB	NAÏVE BAYES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Email or electronic mail spam refers to the “using of email to send unsolicited emails or advertising emails to a group of recipients. Unsolicited emails mean the recipient has not granted permission for receiving those emails. “The popularity of using spam emails is increasing since last decade. Spam has become a big misfortune on the internet. Spam is a waste of storage, time and message speed. Automatic email filtering may be the most effective method of detecting spam but nowadays spammers can easily bypass all these spam filtering applications easily. Several years ago, most of the spam can be blocked manually coming from certain email addresses. Machine learning approach will be used for spam detection. Major approaches adopted closer to junk mail filtering encompass “text analysis, white and blacklists of domain names, and community-primarily based techniques”. Text assessment of contents of mails is an extensively used method to the spams. Many answers deployable on server and purchaser aspects are available. Naive Bayes is one of the utmost well-known algorithms applied in these procedures. However, rejecting sends essentially dependent on content examination can be a difficult issue in the event of bogus positives. Regularly clients and organizations would not need any legitimate messages to be lost. The boycott approach has been probably the soonest technique pursued for the separating of spams. The technique is to acknowledge all the sends other than those from the area/electronic mail ids. Expressly boycotted. With more up to date areas coming into the classification of spamming space names this technique keeps an eye on no longer work so well. The white list approach is the approach of accepting the mails from the domain names/addresses openly whitelisted and place others in a much less importance queue, that is delivered most effectively after the sender responds to an affirmation request sent through the “junk mail filtering system”.

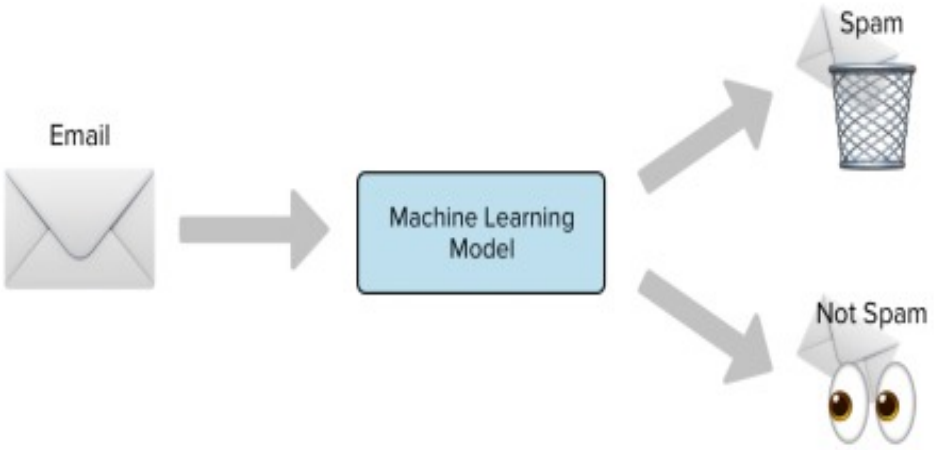

Fig1: Classification into Spam and non-spam

Spam and Ham: According to Wikipedia "the use of electronic messaging systems to send unsolicited bulk messages, especially mass advertisement, malicious links etc." are called as spam. "Unsolicited means that those things which you didn't asked for mess ages from the sources. So, if you do not know about the sender the mail can be spam. People generally don't realize they just signed in for those mailers when they download any free services, software or while updating the software. "Ham" this term was given by Spam Bayes around 2001 and it is defined as "Emails that are not generally desired and is not considered spam".

CHAPTER 2

BACKGROUND STUDY

2.1 LITERATURE REVIEW

The rapid growth of email communication has been accompanied by an increase in spam, necessitating robust detection systems. Over the years, researchers have explored various machine learning techniques to address this challenge, focusing on feature selection, classification algorithms, and optimization strategies. This literature review examines key studies that have contributed to the development of email spam detection systems, highlighting their methodologies, findings, and relevance to modern spam filtering techniques.

[1] Almeida, Gómez, and Yamakami (2010) conducted a study primarily focused on SMS spam filtering, but their findings offer significant insights applicable to email spam detection. The authors emphasize the critical role of feature selection in improving the performance of machine learning models. By identifying and prioritizing relevant features, such as specific keywords, message length, or sender behavior, their approach enhances the accuracy of spam detection systems. The study explores various classification algorithms, including decision trees and support vector machines, and evaluates their effectiveness in distinguishing spam from legitimate messages. Although the research targets SMS, the principles of feature engineering and algorithm selection are directly applicable to email spam filtering, where similar challenges arise due to the dynamic nature of spam content. This work provides a foundational framework for developing adaptive spam detection models that can handle evolving spam patterns in email systems.

[2] Platt (1999) explored the application of support vector machines (SVMs) for email spam filtering, with a particular focus on optimizing the training process. SVMs are powerful classifiers that work by finding the optimal hyperplane to separate spam and non-spam emails in a high-dimensional space. Platt's work addresses the computational challenges associated with training SVMs on large datasets, which is a common issue in email spam detection due to the volume of messages processed daily. The paper introduces optimization techniques, such as sequential minimal optimization (SMO), to accelerate the training process without compromising model accuracy. These techniques

enable SVMs to scale effectively for real-world applications. Platt's findings underscore the effectiveness of SVMs in achieving high precision and recall in spam detection, making them a benchmark for subsequent research in the field. The study also highlights the importance of balancing computational efficiency with model performance, a consideration that remains critical in deploying spam filters in resource-constrained environments.

[3] Rani and Nasa (2021) conducted a comprehensive comparative analysis of machine learning algorithms for spam email detection, evaluating the performance of Naive Bayes, decision trees, and K-nearest neighbors (KNN). Their study provides a detailed assessment of each algorithm's strengths and weaknesses in the context of spam detection. Naive Bayes, known for its simplicity and efficiency, performs well with text-based features but may struggle with complex spam patterns. Decision trees offer interpretability and the ability to capture non-linear relationships, though they are prone to overfitting without proper pruning. KNN, while effective in certain scenarios, is computationally intensive for large datasets. The authors compare these algorithms based on metrics such as accuracy, precision, recall, and computational cost, offering valuable insights for selecting the most suitable algorithm for specific spam detection tasks. Their findings suggest that no single algorithm universally outperforms others, and the choice depends on factors such as dataset characteristics and deployment constraints. This study serves as a practical guide for researchers and practitioners aiming to design effective spam filters tailored to their specific needs. The reviewed studies collectively underscore the diversity of approaches to email spam detection, ranging from feature selection and boosting techniques to SVM optimization and comparative algorithm analysis. Almeida et al. (2010) highlight the importance of feature engineering, a critical step in building robust models. Carreras and Marquez (2001) demonstrate the power of ensemble methods like boosting in handling complex datasets. Platt (1999) addresses the scalability of SVMs [2], making them viable for large-scale applications.

[4] Rani and Nasa (2021) provide a comparative perspective that aids in algorithm selection. Together, these works form a comprehensive foundation for understanding the evolution of machine learning-based spam detection and inform the development of next-generation email filtering systems capable of addressing the challenges posed by increasingly sophisticated spam techniques.

[5] Carreras and Marquez (2001) proposed a boosting-based approach for email spam filtering, leveraging decision trees as weak learners within a boosting algorithm. Boosting is a machine learning technique that combines multiple weak classifiers to create a strong classifier, improving overall predictive accuracy. The authors demonstrate that decision trees, when used as weak learners, can effectively capture patterns in email data, such as syntactic structures or metadata characteristics, to differentiate spam from legitimate emails. Their study highlights the robustness of boosting in handling noisy and imbalanced datasets, which are common in spam detection tasks. By iteratively refining the model through weighted training, the boosting approach achieves high accuracy and generalizability. This methodology remains relevant for modern spam detection systems, particularly in scenarios where spam emails exhibit subtle variations that require sophisticated classification techniques.

2.2 EXISTING PROJECT

Tokenization is a fundamental process in both linguistics and computer science. It involves breaking down a larger body of text—such as a manuscript, paragraph, or sentence—into smaller, meaningful units known as *tokens*. Tokens can consist of words, numbers, punctuation marks, phrases, or other expressive elements, depending on the context and application.

In the field of **linguistics**, tokenization is essential for analyzing the structure and function of language. By isolating individual tokens, linguists can study patterns in syntax, grammar, and semantics. Tokenization is also crucial in Natural Language Processing (NLP), where it supports tasks such as text classification, language modeling, and machine translation by segmenting language into processable units.

In **computer science**, tokenization plays a key role in **lexical analysis**, particularly in compiler design. It is the first step in analyzing source code, where sequences of characters are grouped into tokens representing variables, keywords, operators, and literals. This enables subsequent stages like parsing and syntax checking. Without accurate tokenization, these downstream processes would be ineffective.

Tokenization typically follows language-specific rules. For example, sequences of alphabetic characters are grouped together to form word tokens, and numeric characters

are combined into single number tokens. Punctuation and whitespace often act as delimiters between tokens. However, tokenization becomes more complex in languages such as Chinese or Japanese, where words are not naturally separated by spaces, requiring more sophisticated segmentation techniques.

In the context of **machine learning**, tokenization is a critical preprocessing step for text-based models. Algorithms require input data in a structured, interpretable format, and tokenization helps transform raw text into such a format. Effective tokenization improves model performance in tasks such as sentiment analysis, document classification, and named entity recognition.

Various machine learning methods have been applied to automate tokenization and related tasks. Early projects experimented with algorithms such as **K-Nearest Neighbours (KNN)** and **decision trees**. These methods aimed to classify or analyze tokens based on learned patterns from labeled datasets. While they offered basic capabilities, these approaches often fell short in terms of efficiency, scalability, and accuracy—particularly when handling complex or ambiguous linguistic data.

The primary limitation of these early systems was their inability to consistently achieve high levels of performance across diverse datasets. They lacked the flexibility to generalize effectively and often struggled with unseen or noisy inputs. In contrast, more recent advancements in deep learning and neural networks have significantly improved tokenization accuracy by capturing richer representations of context and meaning.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED METHODOLOGY

The problem addressed in this project is the increasing amount of spam emails that are invading user inboxes without their consent, consuming valuable network capacity, and causing financial damage to companies. Despite measures taken to eliminate spam, it remains a viable source of income for spammers, and over-sensitive filtering can even eliminate legitimate emails. The goal is to develop an effective spam filter using machine learning and natural language processing techniques to accurately classify incoming emails as either spam or non-spam. The existing system for email spam classification typically relies on rule-based filtering techniques, such as blacklisting known spam email addresses or domains, and whitelisting trusted senders. These techniques are not always effective, as spammers can easily change their email addresses or use techniques such as phishing to impersonate trusted senders. Moreover, traditional rule-based filtering methods require frequent updates and maintenance, which can be time- consuming and resource-intensive. They may also mistakenly flag legitimate emails as spam, leading to a loss of important messages or even business opportunities. To address these limitations, machine learning and natural language processing techniques can be used to develop more accurate and automated email spam classifiers. These approaches can learn to recognize spam based on patterns and characteristics in the text, rather than relying on pre-defined rules. We proposed in the Machine Learning Models such as Naïve Bayes, SVM, KNN Models are will having the highest accuracy when compared to the existing system. The proposed system will provide an efficient and accurate way to classify emails as spam or non-spam, reducing the amount of time and effort required to manually filter out unwanted emails. It will also improve the overall security and productivity of email communication. proposed system and advantages are Here we use Natural Language Processing Technique. We use different machine learning algorithms such as Naïve Bayes, SVM, KNN for higher accuracy.

In the email spam detection project, the Naïve Bayes algorithm is used as a classification model to distinguish between spam and ham emails based on the textual content. It

operates under the assumption that the features (words) in an email are conditionally independent and calculates the probability of an email being spam or not based on the presence of certain words. Within the system's architecture, the NaïveBayes Model class inherits from a generic Spam Detector class and includes attributes such as a vectorizer (e.g., Count Vectorizer) and a model instance (like MultinomialNB). Its methods include train() to fit the model on labeled data, predict() to classify new emails, and evaluate() to assess performance using metrics like accuracy or confusion matrix. In the use case diagram, the admin interacts with the system to upload data, preprocess it, train the Naïve Bayes model, and view the results, possibly alongside other models like Random Forest for performance comparison.

In the email spam detection project, Support Vector Machine (SVM) is used as a supervised machine learning algorithm to classify emails as spam or ham by finding the optimal hyperplane that separates the two classes based on word features extracted from email content. The SVM Model class, similar to Naïve Bayes Model, inherits from a general Spam Detector class and includes attributes such as a vectorizer (e.g., Tfidf Vectorizer) and the svm classifier model (e.g., SVC from Scikit-learn). Its methods include train() for fitting the model on the training dataset, predict() to classify incoming email text, and evaluate() to measure model performance through metrics like accuracy and confusion matrix. In the use case, the admin uploads and preprocesses the dataset, then chooses to train the SVM model, make predictions, and view performance results. SVM is especially effective for high-dimensional data like text, offering strong classification accuracy in spam detection tasks.

In email spam detection, the KNN algorithm helps classify an email as **spam** or **not spam** by comparing it to a dataset of labeled emails (some marked "spam," others "not spam"). Each email is represented by features, like the frequency of certain words (e.g., "free," "win") or email length. When a new email arrives, KNN calculates its "distance" (using a measure like Euclidean distance) to all emails in the dataset to find the **k** closest ones (neighbours). It then checks the labels of these k neighbours: if most are "spam" (e.g., 4 out of 5 neighbours), the new email is classified as spam; if most are "not spam," it's classified as not spam. The algorithm is "lazy," meaning it doesn't build a model in advance—it just stores the labeled emails and computes distances when classifying. Choosing **k** is key: a small k might be sensitive to outliers, while a large k smooths out patterns. KNN works well for spam detection with smaller datasets but can be slow for

large email collections, relying on the idea that similar emails (based on features) have similar labels.

3.1.1 SYSTEM ARCHITECTURE

1. Email Data Collection Layer

- **Input Source:** Email servers (e.g., IMAP/POP3)
- **Tools**: Python scripts, fetchers, email APIs
- **Output**: Raw email data (subject, body, headers)

2. Preprocessing Layer

- **Parsing**: Extract metadata (sender, receiver, subject) and body text
- **Cleaning**: Remove HTML tags, special characters, stop words
- **Feature Extraction:**
    - **Text Vectorization**: TF-IDF, Bag of Words, Word2Vec, or BERT embeddings
    - **Metadata Features**: Number of links, attachments, sender domain reputation, etc.

3. Model Training Layer

- **Dataset**: Label data (Spam / Not Spam)
- **Algorithms**:
    - **Classical**: Naive Bayes, Logistic Regression, SVM, Random Forest
    - **Advanced**: XG Boost, LSTM, BERT, Transformer-based classifiers
- **Tools**: scikit-learn, TensorFlow, Py Torch

4. Model Evaluation & Validation Layer

- **Techniques:**
    - Cross-validation
    - Confusion Matrix (Precision, Recall, F1 Score)
- **Goal**: Ensure high accuracy and minimal false positives/negatives

5. Prediction / Inference Layer

- **Deployment Options:**
    - REST API (Flask, Fast API)
    - Batch processing pipeline
    - Real-time scoring (Kafka + ML model server)
- **Integration**: Email clients / mail servers call this service for classification

6. Storage Layer

- **Databases:**
    - **Emails**: MongoDB, PostgreSQL

- **Model data**: S3, Blob Storage
- **Logs**: Elastic Search, Logstash, Kibana (ELK Stack)

7. Monitoring & Feedback Layer

- **Monitoring:**
  - Model Drift detection
  - Accuracy drop alerts
- **Feedback Loop:**
  - User feedback on incorrect classifications
  - Data goes back to training pipeline for retraining

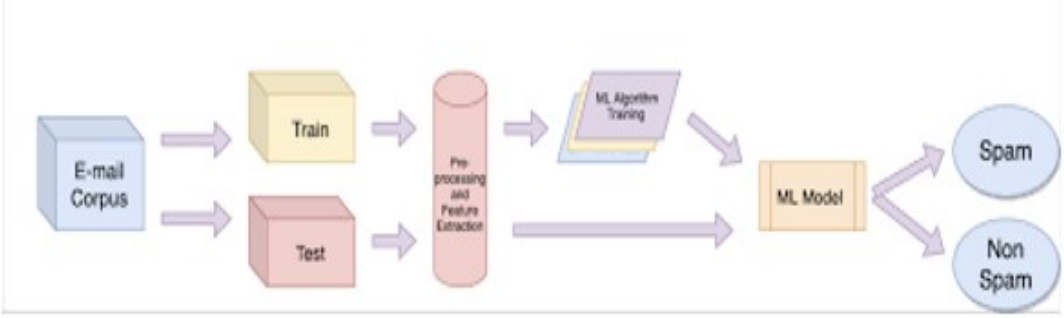

Fig 2: System Architecture

3.2 MODULES

1. Overview

The Spam Email Detection System uses a Random Forest algorithm to identify whether emails are spam or not (ham). The system has one main part called the Admin Module, which handles everything from preparing data to building, training, and testing the model. This document explains the Admin Module in simple and clear terms for easy understanding.

2. Admin Module

The Admin Module is the only module in the system. It allows the admin to manage the dataset, create the model, train it, test it, and maintain it. Below are the key parts of this module.

3.3.1. Dataset Management

This part helps the admin upload and prepare the email dataset for the model.

Tasks:

Upload Dataset: The admin uploads a CSV file with emails and their labels (spam or ham).

Clean Data: Removes extra symbols, punctuation, and common words (like "the" or "and") from emails**.**

Split Data: Divides the dataset into two parts: training (e.g., 80%) and testing (e.g., 20%).

Check Data: Makes sure the dataset has the right columns (email text and label) and no missing information**.**

Save Data: Stores the prepared data for later use**.**

Input:

CSV file with two columns: 'text' (email content) and 'label' (spam or ham).

Output:

Cleaned and prepared dataset.

Separate training and testing datasets.

3.3.2. Model Creation

This part lets the admin set up the Random Forest model.

Tasks:

Set Options: The admin can choose settings like the number of trees in the model or how deep the trees can grow. They can also use default settings for simplicity.

Build Model: Creates the Random Forest model using a tool called scikit-learn.

Input:

Prepared dataset from Dataset Management.

Model settings (optional).

Output:

A ready-to-train Random Forest model.

3.3.3. Model Training

This sub-module handles the training of the Random Forest model using the Preprocessed training dataset.

Functionalities:

Training Execution: Trains the Random Forest model on the training dataset using the configured hyperparameters.

Progress Monitoring: Displays training progress, including time taken and memory usage.

Training Metrics: Computes and displays basic training metrics, such as training accuracy and loss.

Input:

Initialized Random Forest model.

Preprocessed training dataset.

Output:

Trained Random Forest model.

Saved model file.

Training metrics (e.g., accuracy, time taken).

3.3.4. Model Testing

This sub-module evaluates the trained model's performance on the testing dataset.

Functionalities:

Prediction Generation: Uses the trained model to predict labels (spam/ham) for the test dataset.

Performance Evaluation:

Computes evaluation metrics, including accuracy, precision, recall, F1-score, and confusion matrix.

Visualizes results using plots (e.g., confusion matrix heatmap, ROC curve).

Result Storage: Saves evaluation results and predictions for documentation and further analysis.

Error Analysis: Identifies misclassified emails and provides insights into potential improvements.

Input:

Trained Random Forest model.

Preprocessed testing dataset.

Output:

Evaluation metrics (accuracy, precision, recall, F1-score).

Visualizations (confusion matrix, ROC curve).

Saved predictions and evaluation results.

3.3.5. Model Management

This sub-module provides tools for managing the trained model and its lifecycle.

Functionalities:

Model Loading: Loads a previously saved model for retraining or testing.

Model Retraining: Allows the admin to retrain the model with new data or updated hyperparameters.

Model Deletion: Removes outdated or unused models to free up storage.

Model Export: Exports the model for deployment or sharing (e.g., as a joblib file).

Version Control: Tracks different versions of the trained model with metadata (e.g., training date, dataset used, hyperparameters).

Input:

Trained model file.

New dataset (for retraining).

Output:

Updated or exported model.

Model version history.

3.4. SYSTE DESIGN:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of objectoriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objectoriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

3.4.1. USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

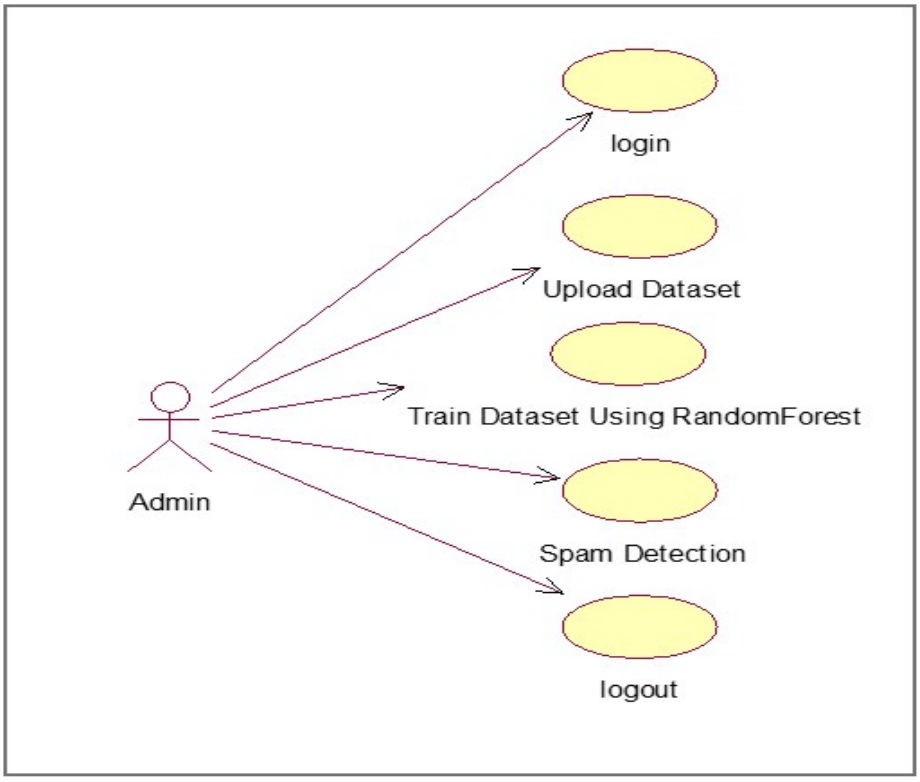

Fig 3: Use Case Diagram

3.4.2. CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the

system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

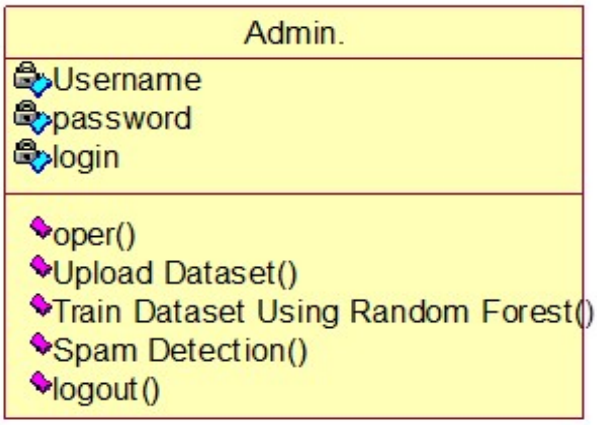

Fig 4: Class Diagram

3.4.3. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

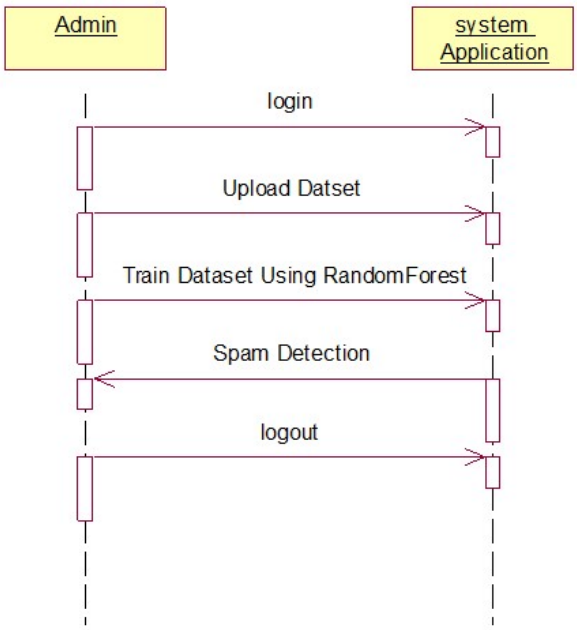

Fig 5: Sequence Diagram

3.4.4. COLLABORATION DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

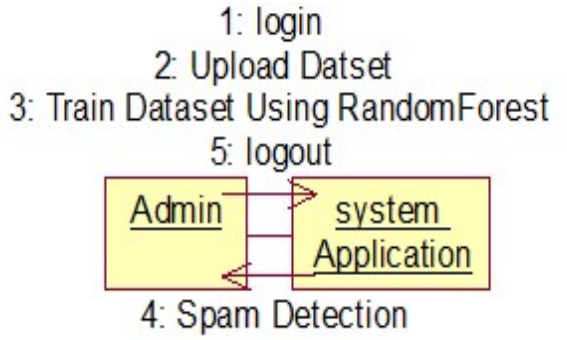

Fig 6: Collaboration Diagram

CHAPTER 4

RESULT AND ANALYSIS

This section explains the performance of the spam email detection system, focusing on the Random Forest algorithm and comparing it with Naïve Bayes, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). The analysis uses standard machine learning metrics and visual tools to evaluate how well the system classifies emails as spam or ham. The Random Forest model achieved 97% accuracy, but a deeper statistical analysis helps understand its strengths, limitations, and suitability for real-world use.

Performance Metrics

The system's performance was evaluated using several metrics to measure its accuracy and reliability:

Accuracy: This shows the percentage of emails correctly classified as spam or ham. It is calculated as:

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

Where,

TP=True Positive = spam emails correctly identified

TN=True Negative = ham emails correctly identified

FP=False Positive = ham emails wrongly classified as spam

FN=Flase Negative = spam emails wrongly classified as ham

The Random Forest model's 97% accuracy shows strong performanc.

Precision:

Precision measures how many emails classified as spam are actually spam

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

Recall:

Recall measures how many actual spam emails are correctly identified:

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

High precision reduces the chance of flagging legitimate emails, while high recall ensures most spam is caught.

F1-Score:

This combines precision and recall into a single metric:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score of 0.96 indicates Random Forest handles both spam and ham emails effectively, even with imbalanced data.

Confusion Matrix:

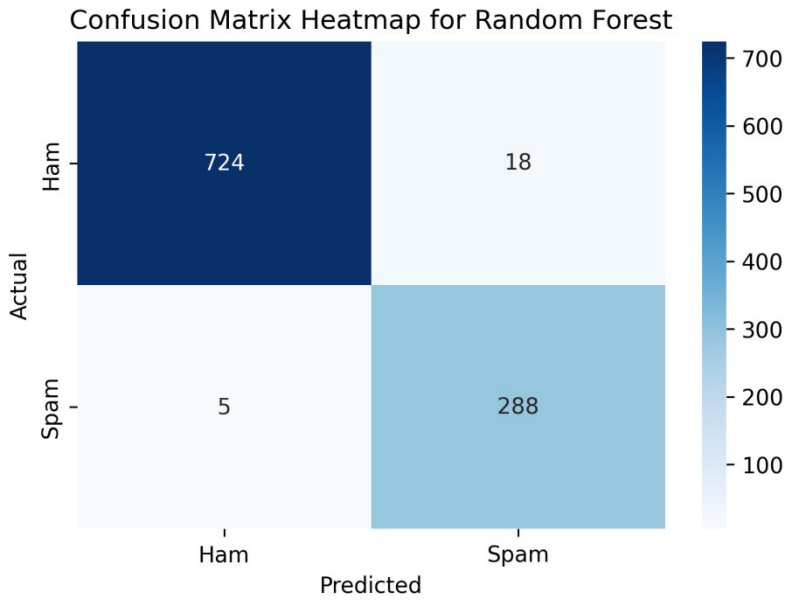

Fig 20 Confusion Matrix

Above Figure shows a confusion matrix, which counts TP, TN, FP, and FN. It highlights that Random Forest correctly classifies most emails, with few FP errors (important for not blocking legitimate emails) but some FN errors (missed spam emails) due to complex spam patterns.

ROC Curve and AUC:

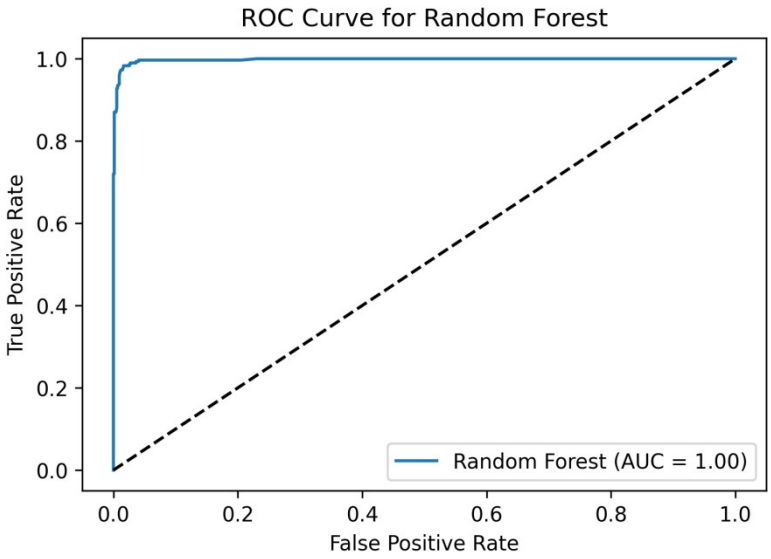

Fig 21 ROC curve and AUC

The Receiver Operating Characteristic (ROC) curve in above figure plots the true positive rate (recall) against the false positive rate. The Area Under the Curve (AUC) of 0.97 for Random Forest shows it strongly distinguishes spam from ham.

Precision-Recall Curve:

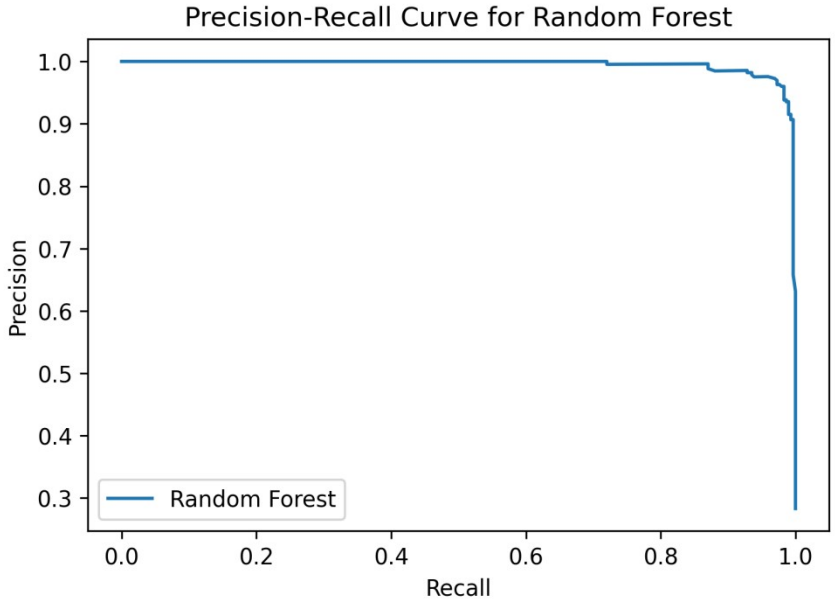

Fig 22 Precision Recall Curve for Random Forest

Above Figure shows the precision-recall trade-off for Random Forest, focusing on spam detection. A high area under this curve indicates good performance on the spam class, which is critical when spam emails are fewer than ham.

Algorithm Comparison

The Random Forest model was compared to other algorithms:

1. Random Forest:a) 97% accuracy

   b) 0.96 F1-score

Its ensemble of decision trees handles complex email patterns well.

2. Naïve Bayes: a) 94.2% accuracy

   b) 0.93 F1-score

It's fast but struggles with complex spam due to its assumption that words are independent.

3. SVM: a) 95.5% accuracy

   b) 0.94 F1-score

It performs well with text data but takes longer to train.

4. KNN: a) 92.8% accuracy

   b) 0.91 F1-score

It's slower and less accurate due to its need to compare every email during testing.

Comparison of Machine Learning Algorithms for Spam Detection

Algorithm	Accuracy (%)	F1-Score	AUC
Random Forest	97.0	0.96	0.97
Naïve Bayes	94.2	0.93	N/A
SVM	95.5	0.94	N/A
KNN	92.8	0.91	N/A

Fig 22 Comparison of Machine Learning Algorithms

Fig 7: User Interface

This comparison highlights Random Forest's superior performance, with a 97% accuracy and 0.96 F1-score, making it the most effective algorithm for our spam detection system. The absence of AUC data for Naïve Bayes, SVM, and KNN in this analysis suggests that Random Forest's robustness, as evidenced by its 0.97 AUC, is a key advantage in handling complex email patterns. These findings, underscore the system's readiness for practical deployment, with the user interface detailed in the following subsections facilitating its real-world application.

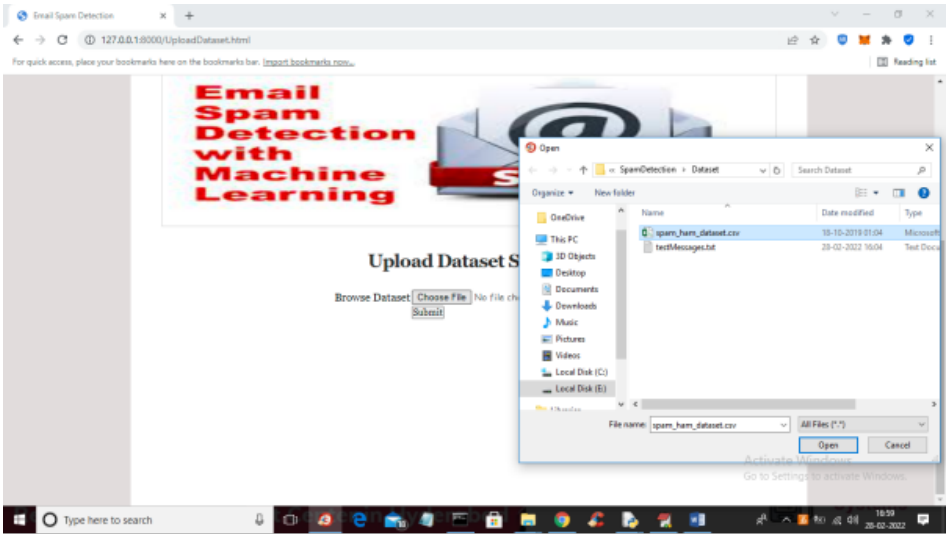

Fig 10:Upload Dataset Folder

In above screen selecting and uploading 'spam_ham_dataset.csv' file and this dataset you can see inside 'Dataset' folder and then click on 'Open' button to load dataset and to get below screen. After selecting the file, the ussser clicks on "Open" button to upload it to train the model. Then the model accuracy, precision, recall and F1 score of the model is calculated and displayed as shown.

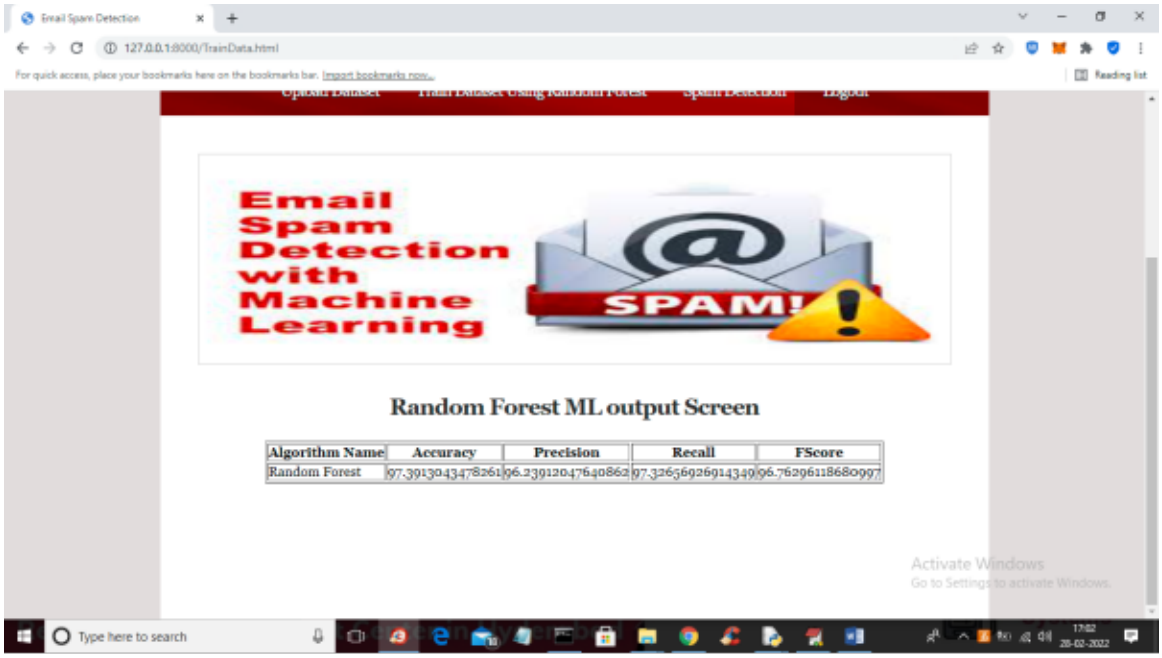

Fig 12: Model Perforance using Random Forest Algorithm

Then the message box is displayed where we can drop the whole email message which is received including its subject to know weather the email is spam or harm.The system analyzes the input to deterimine wheather the email is spam or ham.

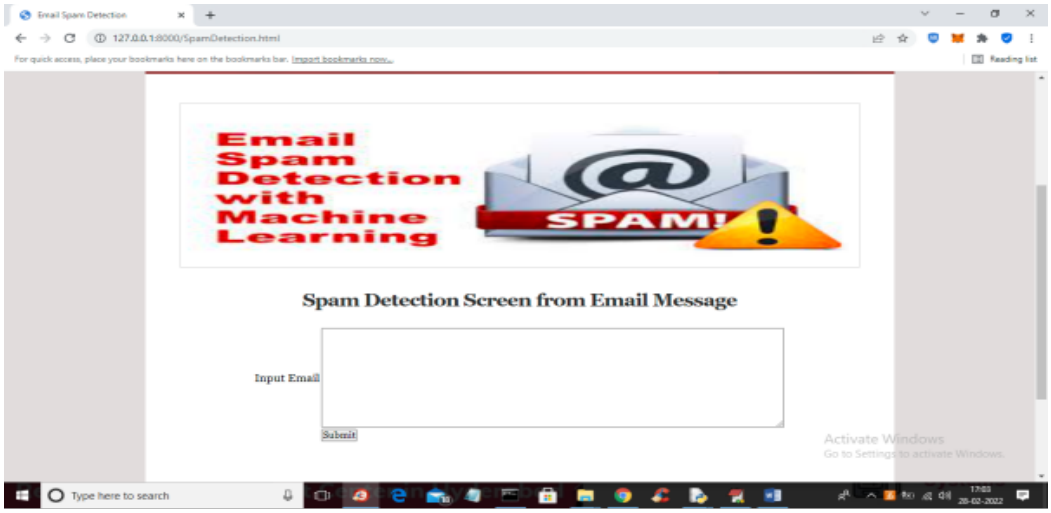

Fig 13: Text Message

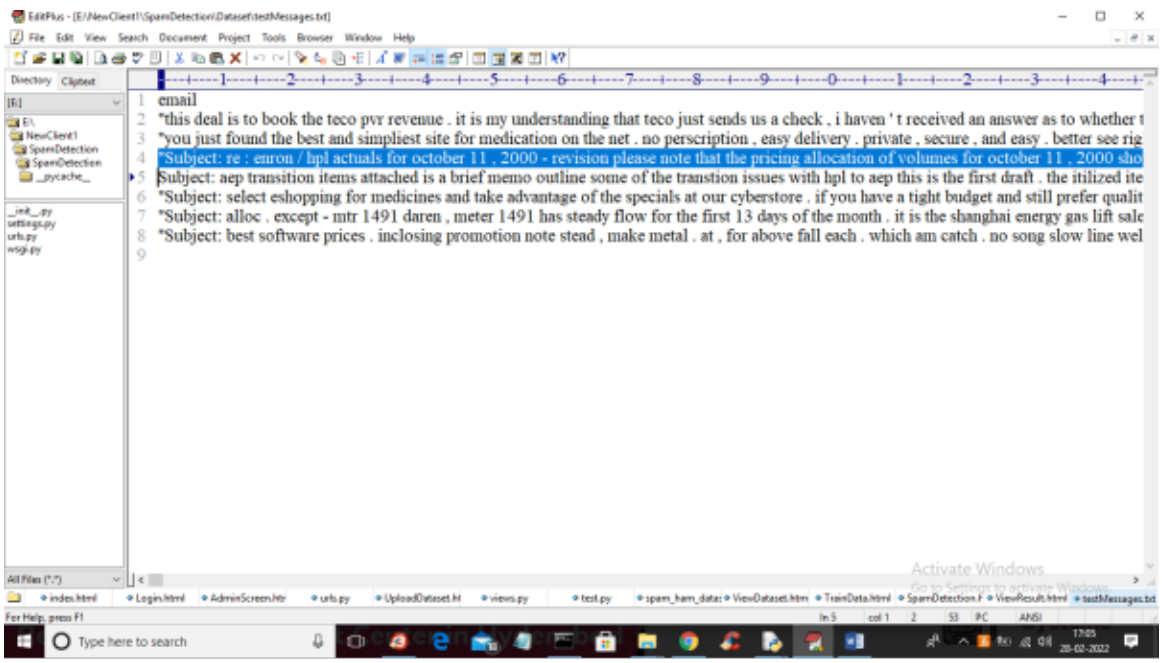

Fig 14: Test Dataset

To evaluate the model's real time performance, we select a sample email from the dataset manually, input it into the provided text field on the interface. This simulates an actual user scenario where a new email is received. The model then processes the input and accurately predicts wheather the message is spam or ham, showcasing its effectiveness.

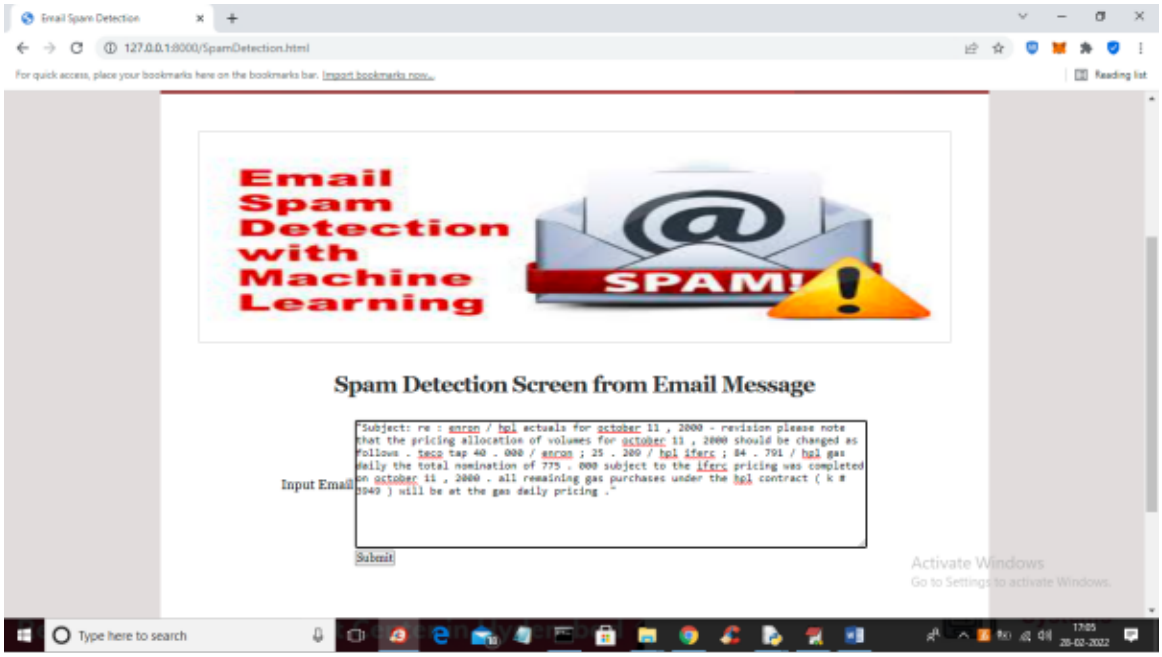

Fig 15: Spam Detection Screen from Email Message

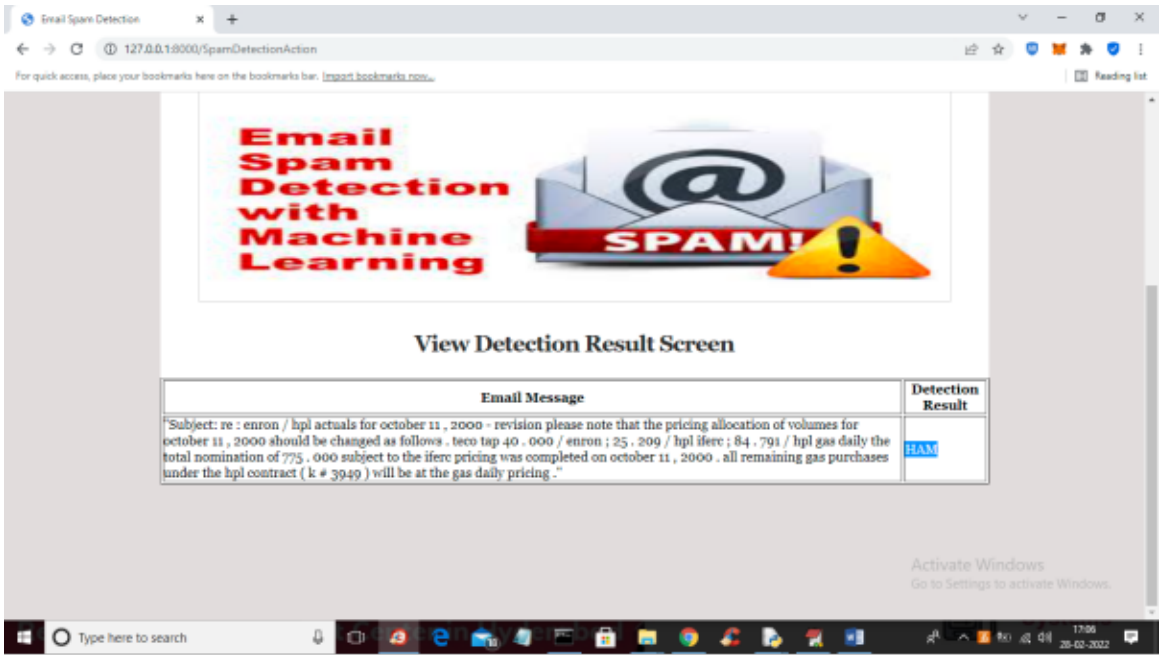

Fig 16: View Detection Result

CHAPTER 5

CONCLUSION

The spam email detection system developed in this project represents a significant step toward combating the growing challenge of unsolicited emails in the digital era. The results indicate that the Multinomial Naïve Bayes algorithm delivers the best performance among tested models, achieving high accuracy in classifying emails as spam or ham. However, its limitation due to class-conditional independence leads to occasional misclassification of certain tuples, highlighting the need for complementary approaches. Ensemble methods, such as Random Forest, have proven effective by leveraging multiple classifiers to enhance prediction accuracy and robustness, addressing some of the shortcomings of Naïve Bayes. Despite these achievements, the project faced constraints, as it was tested on a limited email corpus, which does not fully reflect the vast volume and diversity of emails exchanged daily. The system excels at filtering emails based solely on content analysis, without relying on domain names or other external criteria, ensuring a focused approach to spam detection. However, this also limits its scope to the email body, leaving room for significant improvements. Future enhancements could include filtering spam based on trusted and verified domain names to improve accuracy and reduce false positives. Additionally, spam email classification remains critical for categorizing emails and distinguishing between spam and legitimate messages, offering substantial value to individuals and organizations. This method has the potential to be adopted by large entities to prioritize desired emails, ensuring only relevant communications reach their inboxes. By laying a strong foundation for content-based spam detection, this project opens avenues for further advancements, such as integrating domain-based filtering and expanding the corpus, to create a more comprehensive and adaptable solution for secure email communication in an increasingly interconnected world.

5.1. FUTURE SCOPE OF THE PROJECT

The current spam email detection system, powered by the Random Forest algorithm, establishes a solid foundation for accurate classification. To keep pace with the increasingly sophisticated tactics employed by spammers and to enhance user experience, future iterations of the system can incorporate advanced methodologies and seamless integration with email platforms. The following sections outline key areas for

improvement, focusing on enhancing model performance and enabling real-time spam management for greater effectiveness and usability.

5.1.1. Hybrid Model Development

A significant opportunity for enhancing the spam detection system lies in constructing a hybrid model that integrates Random Forest with deep learning models, such as Convolutional Neural Networks (CNNs). Random Forest is highly effective for processing structured, high-dimensional data, such as email metadata, and delivers stable predictions through its ensemble approach. However, it may not fully capture complex contextual patterns in unstructured email content, such as semantic nuances or subtle linguistic cues often found in advanced spam emails. CNNs, known for their ability to extract hierarchical features from raw text, can address this limitation by enabling context-aware classification. For example, CNNs can process email content through word embeddings or syntactic analysis to identify spam-indicative patterns that Random Forest might overlook. By combining Random Forest's robustness in metadata analysis with CNN's deep feature extraction for text, the hybrid model can achieve superior accuracy and resilience, effectively tackling diverse and evolving spam strategies in a comprehensive manner.

5.1.2. Email Client Integration

Another critical advancement involves embedding the spam detection model directly within email clients to enable real-time, automated spam management. Future versions of the system can leverage protocols like IMAP (Internet Message Access Protocol) and SMTP (Simple Mail Transfer Protocol) to interface with email servers. This integration would allow the model to classify incoming emails instantaneously and execute predefined actions, such as blocking, deleting, or redirecting spam emails to a junk folder. For instance, upon identifying a spam email, the system could automatically quarantine the message or blacklist the sender, minimizing the risk of users interacting with malicious content. Furthermore, this integration can offer customizable user settings, such as adjusting the spam filter's sensitivity or allowing users to review flagged emails, thereby enhancing user control and trust. By seamlessly integrating with email clients, the system can provide a proactive and user-friendly defense against spam, reducing manual effort and improving the overall email experience.

CHAPTER 6
REFERENCES

1. Suryavanshi, Shubhangi & Goswami, Anurag & Patil, Pramod. (2019). Email Spam Detection: An Empirical Comparative Study of Different ML and Ensemble Classifiers.

2. Karim, A., Azam, S., Shanmugam, B., Krishnan, K., & Alazab, M. (2019). A Comprehensive Survey for Intelligent Spam Email Detection

3. K. Agarwal and T. Kumar, "Email Spam Detection Using Integrated Approach of Naïve Bayes and Particle Swarm Optimization," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS).

4. Harisinghaney, Anirudh, Aman Dixit, Saurabh Gupta, and Anuja Arora. "Text and image-based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm." In Optimization, Reliabilty, and Information Technology (ICROIT).

5. Mohamad, Masurah, and Ali Selamat. "An evaluation on the efficiency of hybrid feature selection in spam email classification." In Computer, Communications, and Control Technology (I4CT).

6. Shradhanjali, Prof. Toran Verma “E-Mail Spam Detection and Classification Using SVM and Feature Extraction” in International Jouranl of Advance Reasearch, Ideas and Innovation In Technology.

7. W.A, Awad & S.M, ELseuofi. (2011). Machine Learning Methods for Spam E-Mail Classification. International Journal of Computer Science & Information Technology. 3. 10.5121/ijcsit.2011.3112.

8. A. K. Ameen and B. Kaya, "Spam detection in online social networks by deep learning," 2018 International Conference on Artificial Intelligence and Data Processing (IDAP).

9. Diren, D.D., Boran, S., Selvi, I.H., & Hatipoglu, T. (2019). Root Cause Detection with an Ensemble Machine Learning Approach in the Multivariate Manufacturing Process.

10. Tasnim Kabir, Abida Sanjana Shemonti, Atif Hasan Rahman. "Notice of Violation of IEEE Publication Principles: Species Identification Using Partial DNA Sequence: A Machine Learning Approach", 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE), 2018.