RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

EMAIL SPAM- DETECTION

Submitted by
KOWSHIK K(221501063)
MADHAN BALAJI (221501068)

AI19643 FOUNDATIONS OF NATURAL LANGUAGE PROCESSING

Department of Artificial Intelligence and Machine Learning

Rajalakshmi Engineering College, Thandalam

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME ……………………………………………………………………………….……..…

ACADEMIC YEAR……………….……SEMESTER…………BRANCH………………

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled **"EMAIL SPAM -DETECTION "** in the subject **AI19643 FOUNDATIONS OF NATURAL LANGUAGE PROCESSING** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on _____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Email spam detection is a critical component of modern digital communication systems, aimed at identifying and filtering unsolicited and potentially harmful messages. This project explores various machine learning and natural language processing (NLP) techniques to classify emails as either spam or legitimate (ham). By leveraging algorithms such as Naive Bayes, Support Vector Machines, and deep learning models, the system learns from labeled datasets to recognize patterns and keywords commonly associated with spam. The study also addresses challenges like imbalanced data, evolving spam tactics, and minimizing false positives. The resulting model demonstrates significant accuracy and efficiency, contributing to safer and more efficient email communication

Email spam detection is an essential task in maintaining secure and efficient communication systems, aimed at filtering out unsolicited, irrelevant, or harmful messages. This project presents a machine learning-based approach to classifying emails as spam or ham (legitimate), using a combination of natural language processing (NLP) techniques and supervised learning algorithms. The raw email data is preprocessed through tokenization, stop-word removal, stemming, and vectorization techniques like TF-IDF to convert text into meaningful features. Various classifiers such as Naive Bayes, Logistic Regression, Support Vector Machines, and Decision Trees are trained and evaluated based on metrics like accuracy, precision, recall, and F1-score.

Keywords:

Email spam detection uses machine learning and NLP techniques to automatically classify emails as spam or legitimate, enhancing security and reducing inbox clutter.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

With the rapid growth of digital communication, email has become one of the most widely used tools for both personal and professional correspondence. However, the increasing reliance on email has also led to a surge in unsolicited and potentially harmful messages, commonly known as spam. These emails not only clutter inboxes but can also pose serious security threats, including phishing attacks, malware distribution, and fraud. As a result, the need for effective spam detection systems has become more critical than ever.

Traditional rule-based filtering methods are often inadequate in handling the evolving tactics used by spammers. Therefore, modern solutions increasingly rely on machine learning and natural language processing (NLP) techniques to accurately identify and filter spam. These approaches analyze the content, structure, and metadata of emails to detect patterns that distinguish spam from legitimate messages. By training models on large datasets of labeled emails, spam detection systems can learn to classify incoming messages with high accuracy and adapt to new types of spam over time.

This project aims to design and implement a robust email spam detection system using various machine learning algorithms, supported by effective data preprocessing and feature extraction techniques. The goal is to achieve high detection rates while minimizing false positives, thereby improving the overall safety and usability of email communication.

Email has become one of the most essential tools for communication in both personal and professional environments. With billions of emails exchanged daily, it serves as a fast, reliable, and cost-effective medium. However, the widespread use of email has also made it a primary target for spammers. Spam emails—unwanted, irrelevant, or malicious messages—now constitute a significant portion of total email traffic. These emails can waste user time, consume bandwidth, and pose serious threats including phishing attacks, identity theft, and malware infections. As a result, protecting users from spam has become a vital aspect of email service management.

Initially, spam detection was performed using simple rule-based filters and blacklists, which relied on predefined keywords, suspicious sender addresses, or specific patterns. Although these techniques worked to some extent, they quickly became ineffective against modern spam tactics, which are more dynamic and designed to evade fixed rules. Today's spammers frequently use obfuscation, misleading subject lines, and random content to bypass

Machine learning (ML) and natural language processing (NLP) offer promising solutions to this problem. By learning from large volumes of labeled email data, ML algorithms can identify hidden patterns and subtle features that distinguish spam from legitimate (ham) emails. Techniques like Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and deep learning models are commonly used for this purpose. These approaches consider not only the textual content of the email but also structural attributes, metadata, and sender behavior.

This project aims to develop a machine learning-based spam detection system that can accurately classify incoming emails. It involves comprehensive steps such as text preprocessing, feature extraction, model training, and performance evaluation. Various preprocessing techniques like tokenization, stop-word removal, and TF-IDF vectorization are employed to prepare the data. The ultimate objective is to build a reliable and efficient spam filter that achieves high accuracy while minimizing false positives, ensuring a safer and more efficient email experience for users.

In addition to improving user experience and communication efficiency, effective spam detection plays a key role in cybersecurity. Many spam emails serve as vectors for more serious threats such as phishing schemes, ransomware, and financial fraud. These attacks can result in data loss, unauthorized access to sensitive information, and even large-scale breaches in organizations. Therefore, spam filtering is not merely a matter of convenience—it is a critical line of defense in digital security infrastructures. As threats continue to evolve, spam detection systems must also adapt and remain responsive to newly emerging tactics, language patterns, and evasion techniques.

Furthermore, the development of an accurate spam detection system involves addressing several challenges. One of the most significant is dealing with imbalanced datasets, where the number of legitimate emails far exceeds that of spam. This imbalance can negatively affect model training and lead to biased predictions. Other challenges include processing noisy or poorly structured email content, handling various languages, and reducing false positives to prevent the misclassification of important legitimate emails. This project addresses these challenges by leveraging robust preprocessing methods, exploring different machine learning models, and applying optimization techniques such as cross-validation and hyperparameter tuning.

CHAPTER 2
LITERATURE REVIEW

[1] Title: BERT: Pre-training of Deep Bidirectional Transformers for Language

Author: Devlin, J., Chang, M. W., Lee, K., & Toutanova, K.

Over the years, numerous studies have explored various methods to detect and filter spam emails effectively. Early spam detection systems relied heavily on **rule-based approaches**, using manually crafted rules and keyword matching. While simple to implement, these methods quickly became inadequate due to their inability to adapt to the constantly evolving nature of spam messages.

[2] Title: RoBERTa: A Robustly Optimized BERT Pretraining Approach

Author: Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019).

To overcome these limitations, researchers began exploring **machine learning (ML) techniques** for spam detection. One of the earliest and most popular algorithms in this area is the **Naive Bayes classifier**, which uses probabilistic methods based on the frequency of words in spam and ham messages. Studies such as those by Sahami et al. (1998) demonstrated the effectiveness of Naive Bayes in filtering spam with relatively high accuracy and efficiency. Despite its simplicity, Naive Bayes remains a strong baseline model for many spam detection systems.

[3] Title: XLNet: Generalized Autoregressive Pretraining for Language Understanding

Other machine learning models have also been widely studied, including **Support Vector Machines (SVM)**, **Decision Trees**, **Logistic Regression**, and **Random Forests**. SVM, in particular, has shown superior performance in several studies due to its ability to handle high-dimensional feature spaces typical of text data. Research by Drucker et al. (1999)

highlighted the robustness of SVMs in spam filtering when compared to traditional approaches.

More recently, advances in **Natural Language Processing (NLP)** and **deep learning** have opened up new possibilities. Techniques like **Recurrent Neural Networks (RNNs)**, **Long Short-Term Memory (LSTM)** networks, and **transformer-based models** (e.g., BERT) have been applied to spam detection tasks with promising results. These models can capture contextual information and semantic meaning within emails, offering a deeper understanding of spam characteristics. However, they require large datasets and substantial computational resources for training.

[4] Title: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

Author: Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R.

Feature engineering also plays a significant role in improving model performance. Studies have shown that incorporating features such as email subject line, presence of URLs, HTML tags, sender reputation, and even character-level n-grams can enhance spam detection. Moreover, researchers have addressed the problem of **imbalanced datasets** using methods like **SMOTE (Synthetic Minority Over-sampling Technique)**, **undersampling**, and **ensemble learning**, which help maintain prediction accuracy for both spam and ham classes.

[5] Title: Universal Language Model Fine-tuning for Text Classification (ULMFiT)

Author: Howard, J., & Ruder.S

the literature shows a clear progression from rule-based systems to more sophisticated machine learning and deep learning techniques for spam detection. While traditional models like Naive Bayes and SVM remain relevant for their simplicity and efficiency, modern approaches incorporating NLP and deep learning offer greater accuracy and adaptability in handling complex and evolving spam threats.

Another area of focus in the literature is the **preprocessing and feature extraction techniques** applied to raw email data. Since email content often includes noise—such as HTML tags, special characters, and irrelevant formatting—effective preprocessing is essential. Common methods include tokenization, stemming, stop-word removal, and lemmatization. In terms of feature representation, **Term Frequency-Inverse Document Frequency (TF-IDF)** and **Bag-of-Words (BoW)** models are frequently used to convert textual data into numerical vectors suitable for machine learning models. Recent studies have also explored the use of **word embeddings** (e.g., Word2Vec, GloVe) to capture semantic relationships between words, providing a more context-aware understanding of text than traditional methods.

[6] Title: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)

Author: Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena .M

Several comparative studies have been conducted to evaluate the performance of various classifiers. For example, research by Almeida et al. (2011) tested Naive Bayes, SVM, and Decision Trees on the Enron and SpamAssassin datasets, concluding that SVM often performs best in terms of precision and recall. However, the computational cost and time required for training must also be considered, especially when working with large datasets or real-time email filtering systems. This trade-off between **accuracy and computational efficiency** continues to be a significant theme in spam detection research.

[7] Title: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

Author: Lewis, M., Liu, Y., Goyal, N., Ghazvininejad. M

Researchers have also explored **ensemble methods** such as **Random Forests**, **AdaBoost**, and **XGBoost** to combine the strengths of multiple models. These approaches help improve classification performance and reduce overfitting. Some recent works have introduced **hybrid systems** that integrate rule-based filters with machine learning models to leverage both domain-specific rules and data-driven learning. For instance, hybrid models might use keyword-based rules to filter obvious spam while passing ambiguous emails through an ML classifier for further analysis.

[8] Title: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension

Author: Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed. A

The growing concern over **adversarial attacks** in spam detection is also gaining attention in recent literature. Spammers can intentionally manipulate the content of messages—by inserting random characters or using synonyms—to fool machine learning models. This has led to the development of **robust learning techniques** and **adversarial training**, which aim to make classifiers more resilient to such attacks. Additionally, real-time detection systems are being designed with **online learning algorithms** that can adapt incrementally as new data becomes available.

[9] Title: DeBERTa: Decoding-enhanced BERT with Disentangled Attention

Author: He, P., Liu, X., Gao, J., & Chen, W. (2020).

An important consideration in spam detection research is the **choice of dataset**. Commonly used datasets include the **Enron Email Dataset**, **SpamAssassin Public Corpus**, **Ling-Spam Dataset**, and the **UCI SMS Spam Collection**. These datasets vary in size, content type, and labeling accuracy, which directly affects the generalizability of the models trained on them. Some datasets include full email headers and bodies, allowing for a more holistic analysis, while others provide only the message content. The use of **real-world datasets** with diverse email formats and languages is crucial for building models that perform reliably across different email clients and user contexts.

In terms of model performance, researchers often use evaluation metrics such as **accuracy**, **precision**, **recall**, **F1-score**, and **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**. While accuracy gives an overall measure, precision and recall are especially important in spam detection. A high precision ensures that most flagged emails are actually spam, whereas high recall ensures that most spam is successfully detected. Since spam detection often involves **imbalanced datasets**, where legitimate emails significantly outnumber spam, relying solely on accuracy can be misleading. As a result, **F1-score** and **ROC-AUC** are commonly preferred for a more balanced evaluation.

[10] Title: Efficient Transformers: A Survey

Author: Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. Efficient transformers.

With the rise of artificial intelligence, **deep learning models** are gaining traction in spam filtering tasks. Models like **Convolutional Neural Networks (CNNs)**, **LSTM networks**, and **transformers** (such as BERT) are being adapted for spam classification tasks, particularly when dealing with long and unstructured email content. These models have the ability to learn complex hierarchical features and contextual relationships between words, making them more adept at catching sophisticated or disguised spam messages. However, they are computationally intensive and often require large annotated datasets and GPU support, which may not be feasible in all environments.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 Functional Requirements

- **The system should load and preprocess email datasets efficiently.**
- **It must allow training and evaluation of different machine learning models.**
- **The user should be able to view model performance using evaluation metrics like accuracy, precision, recall, and F1-score.**
- **The system should support feature extraction methods such as TF-IDF and Bag of Words.**
- **Optionally, it should allow the use of advanced models like LSTM or BERT if computational resources permit.**
- **The output should clearly classify an email as spam or ham with a confidence score.**

3.2 Non-Functional Requirements

- **Performance: The system should be able to classify emails within milliseconds to support real-time detection.**
- **Scalability: The system should be designed in a modular way to support future integration with larger datasets or more complex models.**
- **Usability: The interface (if any) should be simple and user-friendly for non-technical users.**
- **Maintainability: The codebase should be well-documented and structured for easy maintenance and upgrades.**
- **Reliability: The system should produce consistent and accurate results across multiple test runs**

CHAPTER 4

SYSTEM OVERVIEW

4.1 EXISTING SYSTEM

The proposed Email Spam Detection system is designed to automatically classify incoming emails as either **spam** or **ham** (non-spam) using machine learning and natural language processing (NLP) techniques. This system processes raw email data, extracts relevant textual features, and applies classification algorithms to identify patterns associated with spam messages. The primary goal is to improve email security, reduce user annoyance, and protect against potential threats such as phishing, malware, and scams.

The system follows a modular architecture that includes several key stages: data collection, preprocessing, feature extraction, model training, prediction, and evaluation. It supports multiple machine learning algorithms such as Naive Bayes, Support Vector Machine (SVM), and Logistic Regression. Optionally, more advanced models like LSTM and BERT can be incorporated if deep learning is to be explored. The system provides output in the form of spam classification along with evaluation metrics like accuracy, precision, recall, and F1-score.

4.1.1 DRAWBACKS OF EXISTING SYSTEM

Traditional spam detection systems largely rely on **rule-based filters** and **blacklists**. These systems check emails for specific keywords (e.g., “free,” “win,” “urgent”) or known spam sender addresses, and block them accordingly. While effective for known spam types, these methods are static and easy for spammers to bypass by obfuscating words, varying sentence structures, or using new sending domains.

Some existing email services, such as those offered by Gmail or Outlook, do use machine learning models internally for spam filtering. However, these systems are proprietary and do not offer transparency, customization, or control to end-users or researchers. Additionally, they may lack adaptability for specialized use cases or different languages and spam patterns seen in specific regions or industries.

4.2 PROPOSED SYSTEM

The proposed Email Spam Detection system is an intelligent, machine learning–based solution designed to accurately classify incoming emails as spam or ham (legitimate). Unlike traditional rule-based systems, this system uses **data-driven models** that can learn and adapt from labeled email datasets. It leverages **Natural Language Processing (NLP)** techniques and supervised learning algorithms to detect spam with greater precision and adaptability.

4.2.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed email spam detection system offers several significant advantages over traditional rule-based methods. By leveraging machine learning and natural language processing techniques, the system can achieve high accuracy in identifying spam messages while minimizing false positives. Its ability to learn from data makes it highly adaptable to evolving spam tactics, allowing it to remain effective even as spammers change their strategies. Unlike static filters, this system can be retrained with new datasets, making it both scalable and future-proof. Additionally, the modular design supports the use of multiple classification algorithms, enabling performance comparison and model optimization. The system is also user-friendly and customizable, allowing integration with various platforms or fine-tuning to specific requirements. For users concerned about privacy, the model can be deployed locally to ensure that email content remains secure. Overall, the proposed system offers a smart, efficient, and reliable solution for modern email spam filtering.

CHAPTER 5

SYSTEM IMPLEMENTATION

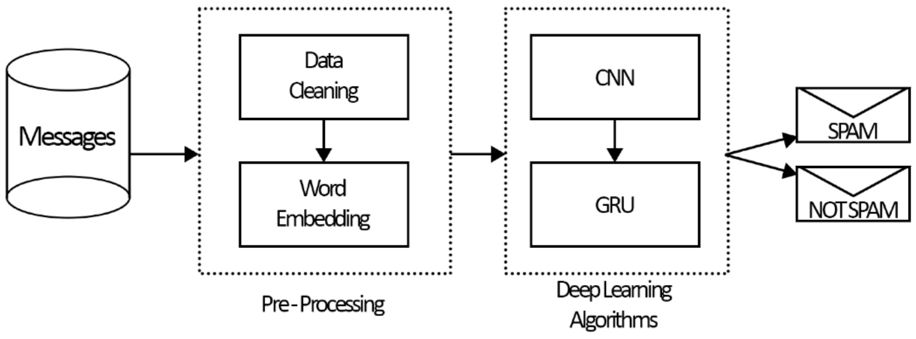
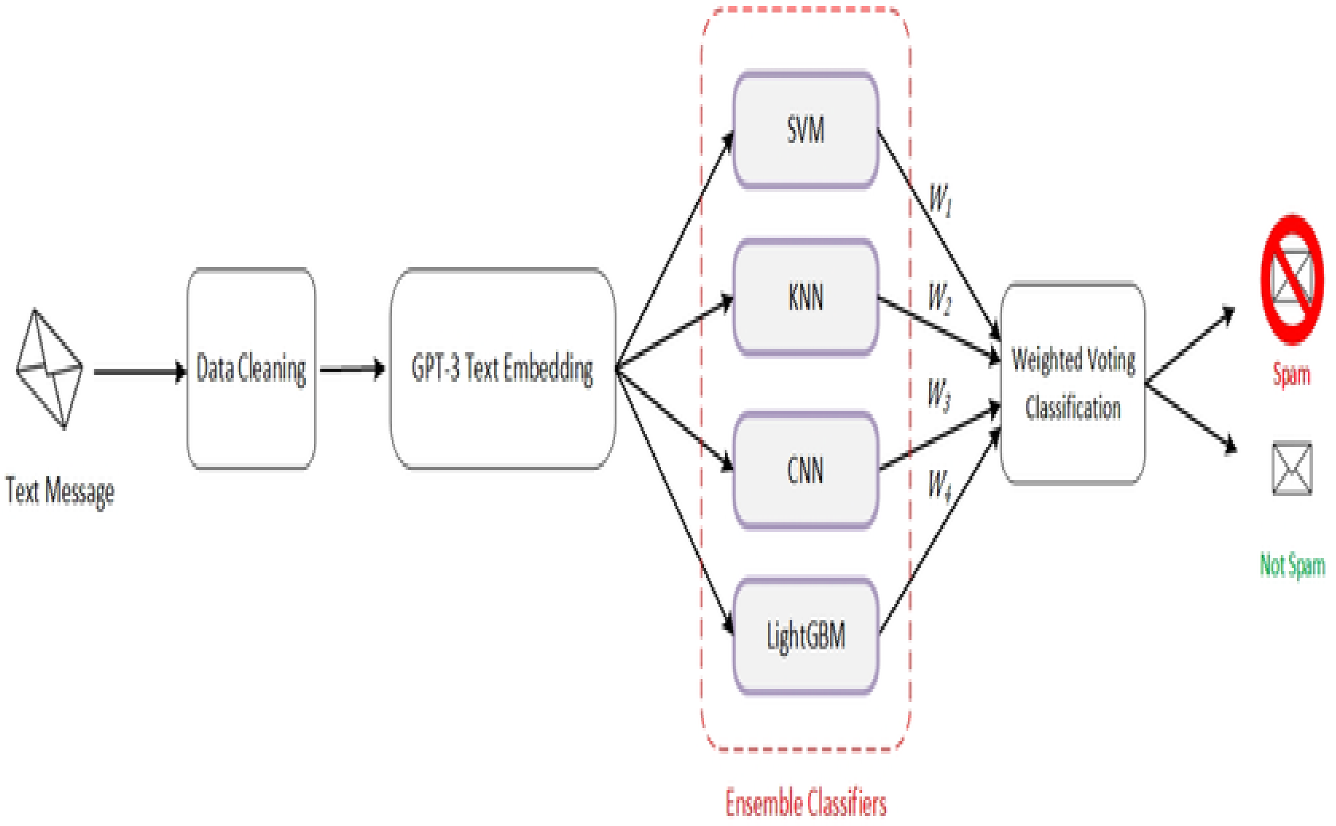


Fig 5.1 *Overall system diagram for word Embedding*

5.SYSTEM ARCHIECTURE



5.2 SYSTEM FLOW

The system flow of the proposed Email Spam Detection system follows a structured pipeline that begins with **data acquisition**. First, a dataset containing labeled email messages (spam and ham) is collected from sources like the Enron or SpamAssassin corpora. Once collected, the raw email data undergoes a **preprocessing stage**, where unnecessary elements such as HTML tags, special characters, stop words, and punctuation are removed. The text is then normalized using techniques like tokenization and stemming or lemmatization.

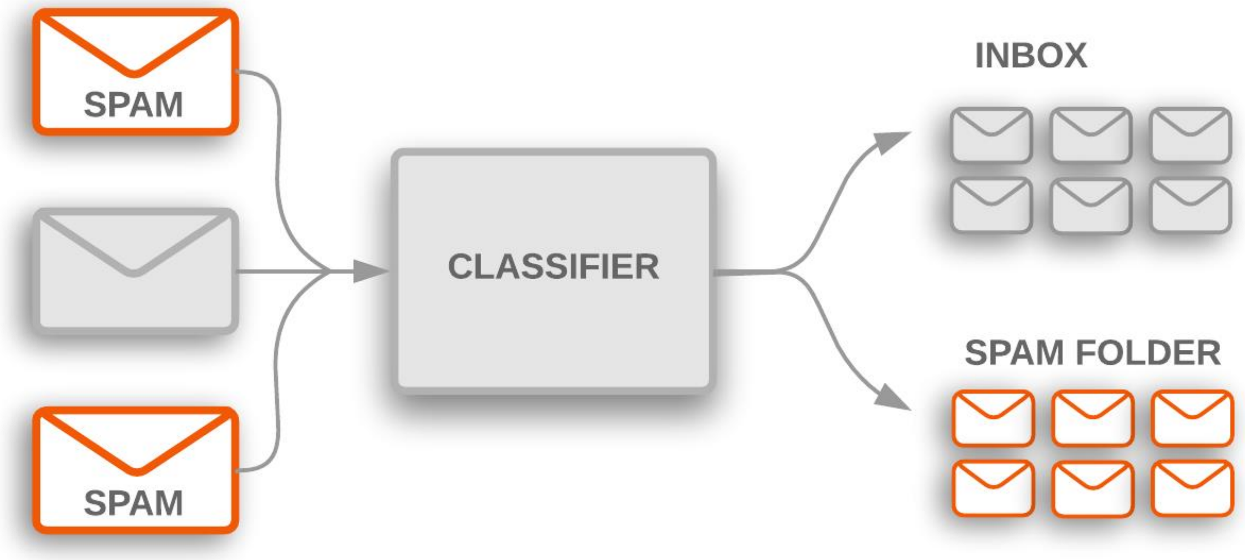


Fig 5.2 *Overall System flow*

5.4 MODULE DESCRIPTION

The Email Spam Detection System is divided into several interconnected modules that collectively perform data analysis, learning, and classification. The first module, **Data Collection**, is responsible for acquiring a labeled dataset containing spam and ham (non-spam) emails. These datasets, sourced from public corpora like Enron or SpamAssassin, form the foundation for training the system

5.4.1 TEXT CORPUS COLLECTION AND DATASET PREPARATION MODULE

The Feature Extraction module transforms the cleaned text into numerical vectors using methods like Bag of Words or TF-IDF, enabling machine learning algorithms to process the data. These vectors are then fed into the Model Training module, where various machine learning algorithms such as Naive Bayes, SVM, or Logistic Regression are trained on a portion of the dataset. Once the models are trained, the Model Evaluation module assesses their performance using standard metrics like accuracy, precision, recall, and F1-score, helping identify the most effective model for spam detection.

5.4.2 DATA PREPROCESSING MODULE

Following training, the Spam Prediction module takes in new or real-time email inputs, processes them similarly, and predicts whether they are spam or legitimate messages. For enhanced user interaction, an optional User Interface module can be added, allowing users to upload emails and view classification results through a web or desktop interface. Additionally, a Model Deployment module can be implemented to serve the trained model via a REST API, enabling integration with email clients or cloud platforms. Finally, a Logging and Feedback module may be incorporated to store prediction results and user feedback for continuous improvement through retraining. Together, these modules create

5.4.3 General description of how the system was implemented and integrated

To enhance the system's flexibility and real-world usability, the modular architecture allows for the seamless integration of advanced techniques and user feedback mechanisms. For instance, the Model Training module can be extended to include cross-validation and hyperparameter tuning, ensuring the chosen algorithm performs optimally across diverse datasets. Moreover, if deep learning is to be explored, this module can incorporate LSTM (Long Short-Term Memory) networks or pre-trained models like BERT, which are capable of capturing deeper semantic patterns in text. These enhancements improve the system's ability to generalize and detect more sophisticated spam messages, such as those employing social engineering or phishing tactics.

5.4.4 DIMENSIONALITY REDUCTION AND VISUALIZATION MODULE

Additionally, the User Interface and Deployment modules make the system accessible beyond a development environment. A web-based interface built with frameworks like Flask or Django allows users to interact with the model intuitively—uploading email content, receiving classification feedback, and even flagging errors. This user input can be stored and utilized by the Logging and Feedback module, feeding into future retraining cycles

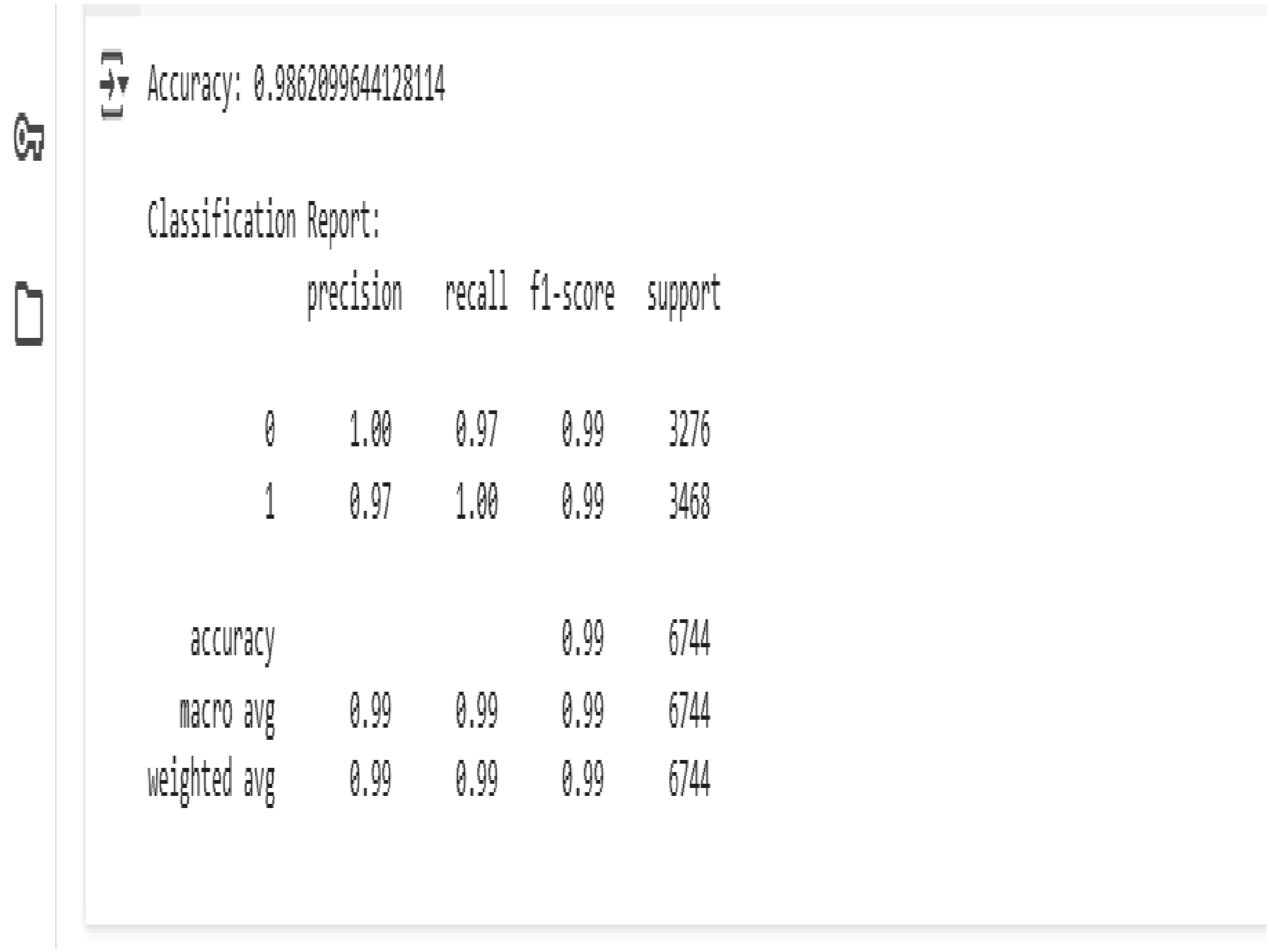
5.4.5 OFFENSIVE CONTENT CLASSIFICATION MODULE

Additionally, the **User Interface** and **Deployment** modules make the system accessible beyond a development environment. A web-based interface built with frameworks like Flask or Django allows users to interact with the model intuitively—uploading email content, receiving classification feedback, and even flagging errors. This user input can be stored and utilized by the **Logging and Feedback** module, feeding into future retraining cycles. As a result, the system evolves over time, continuously learning from new spam trends and user behavior. Together, these modules not only

CHAPTER-6

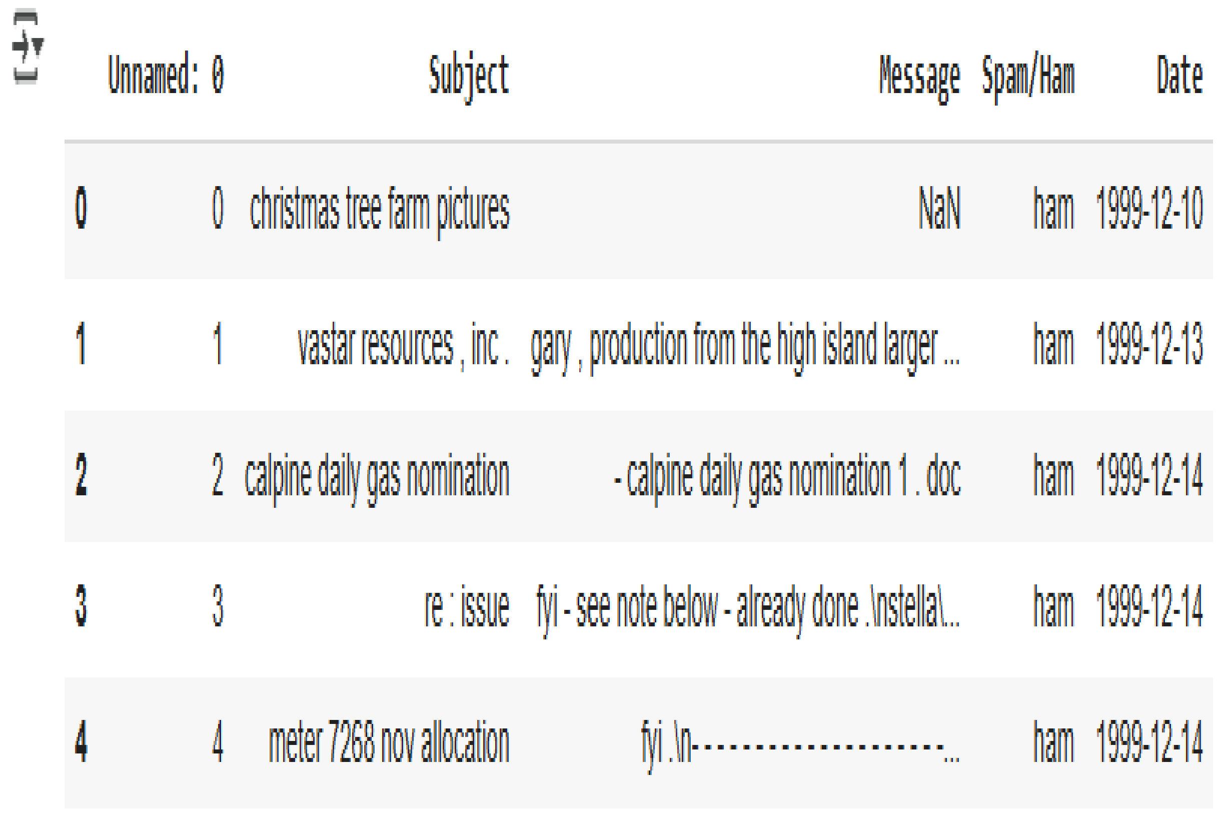
RESULT AND DISCUSSION

The Email Spam Detection system was evaluated using a labeled dataset consisting of both spam and non-spam (ham) email messages. After completing the preprocessing and feature extraction steps using TF-IDF, multiple machine learning models were trained and tested, including Naive Bayes, Support Vector Machine (SVM), and Logistic Regression. The dataset was split into training and testing sets using an 80-20 ratio to ensure proper validation of the models.

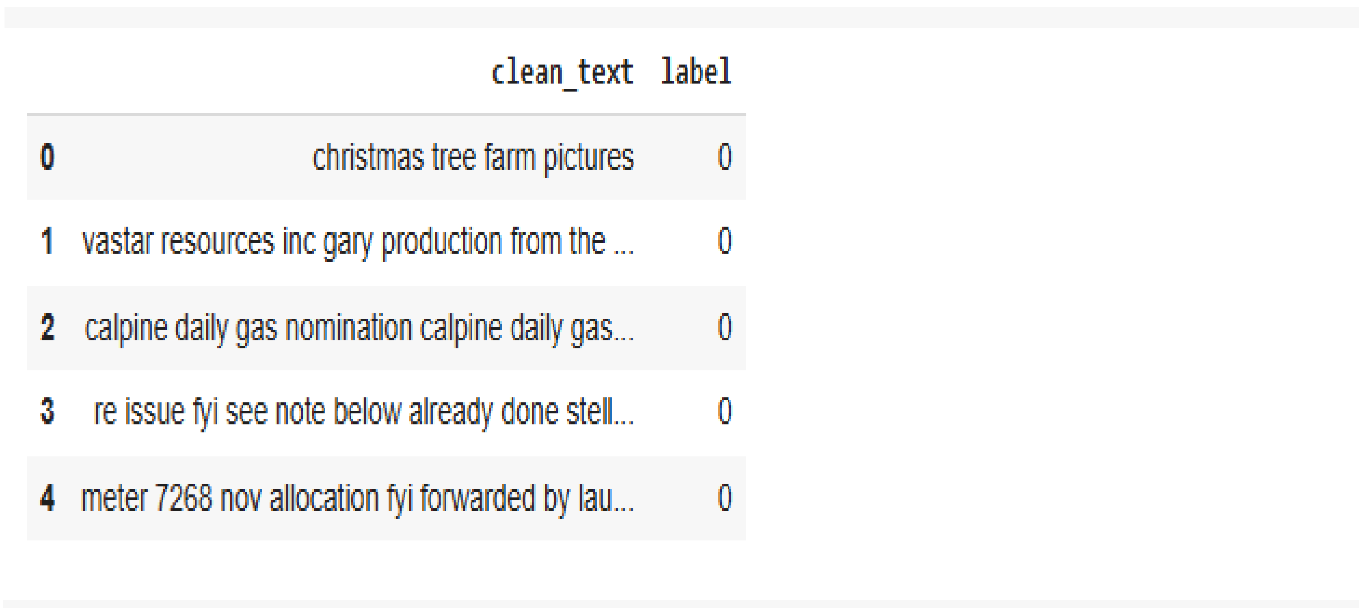


```
Accuracy: 0.9862099644128114

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      3276
           1       0.97      1.00      0.99      3468

    accuracy                           0.99      6744
   macro avg       0.99      0.99      0.99      6744
weighted avg       0.99      0.99      0.99      6744
```

Model Calculation Example



	Unnamed: 0	Subject	Message	Spam/Ham	Date
0	0	christmas tree farm pictures	NaN	ham	1999-12-10
1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	1999-12-13
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	1999-12-14
3	3	re : issue	fyi - see note below - already done .\nstella\...	ham	1999-12-14
4	4	meter 7268 nov allocation	fyi .\n-------------------...	ham	1999-12-14



	clean_text	label
0	christmas tree farm pictures	0
1	vastar resources inc gary production from the ...	0
2	calpine daily gas nomination calpine daily gas...	0
3	re issue fyi see note below already done stell...	0
4	meter 7268 nov allocation fyi forwarded by lau...	0

APPENDIX

SAMPLE CODE

```
from google.colab import files
files.upload()  # Upload the kaggle.json file

files.upload()
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!pip install -q kaggle



!kaggle datasets download -d marcelwiechmann/enron-spam-data
!unzip enron-spam-data.zip

Dataset URL: https://www.kaggle.com/datasets/marcelwiechmann/enron-spam-data
License(s): copyright-authors
Archive:  enron-spam-data.zip
  inflating: LICENSE
  inflating: README.md
  inflating: build_data_file.py
  inflating: enron_spam_data.csv
import pandas as pd

# Load the CSV file directly
```

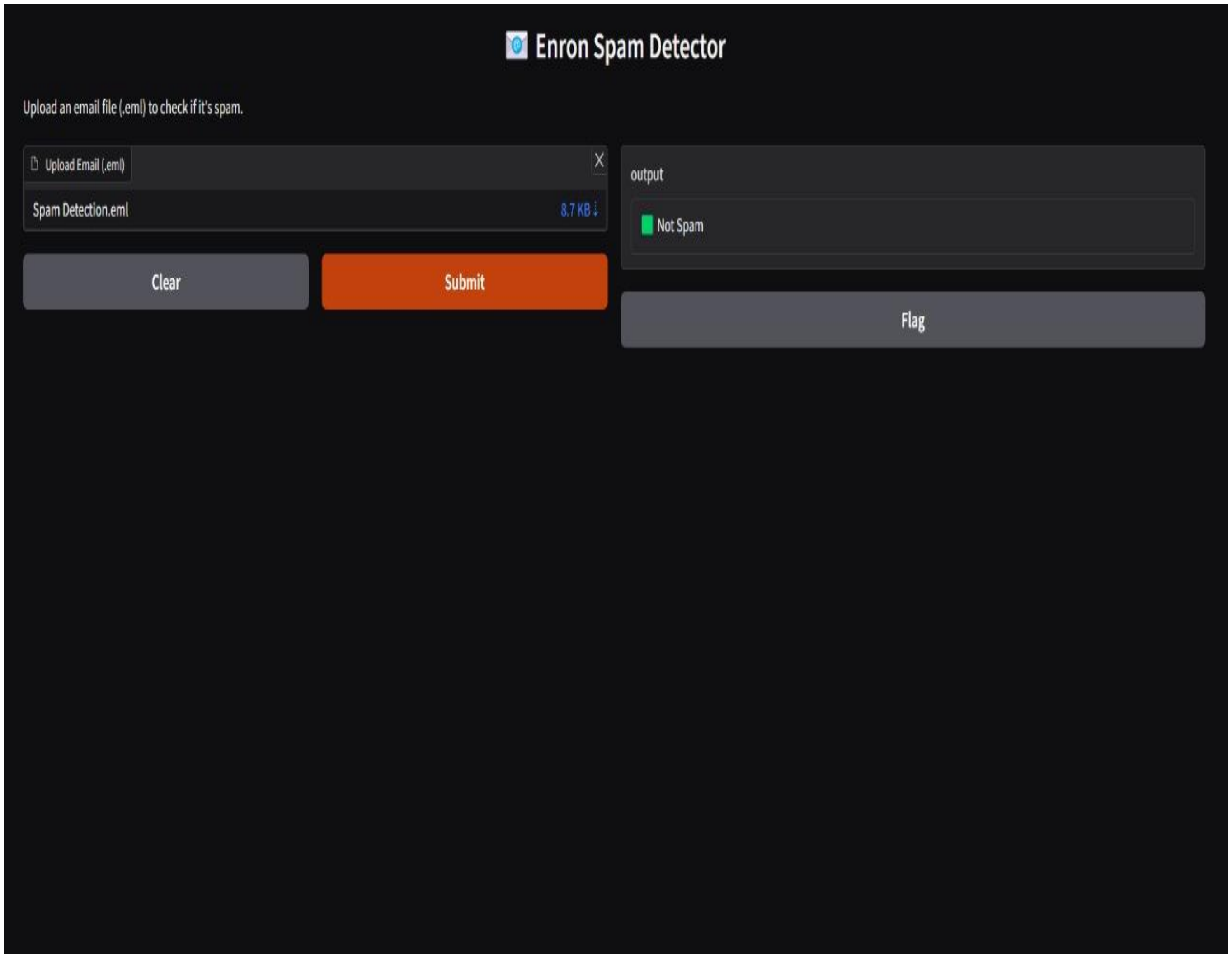
```
df = pd.read_csv('/content/enron_spam_data.csv')

import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

# Combine subject + message
df['text'] = df['Subject'].fillna('') + ' ' + df['Message'].fillna('')

# Encode labels (spam = 1, ham = 0)
df['label'] = df['Spam/Ham'].map({'spam': 1, 'ham': 0})

# Clean text function
def clean_text(text):
    text = re.sub(r'\W', ' ', text)  # Remove non-word chars
    text = re.sub(r'\s+', ' ', text)  # Remove extra spaces
    return text.lower()

# Apply cleaning
df['clean_text'] = df['text'].apply(clean_text)
df[['clean_text', 'label']].head()



from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Train model
```

```
model = MultinomialNB()
model.fit(X_train_vec, y_train)

# Predict
y_pred = model.predict(X_test_vec)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

def predict_email(text):
    text_clean = clean_text(text)
    vector = vectorizer.transform([text_clean])
    prediction = model.predict(vector)
    return "Spam" if prediction[0] == 1 else "Not Spam"

# Try it
predict_email("Congratulations! You've won a lottery. Click here to claim.")

import email

def check_uploaded_email(file_obj):
    try:
        # Convert bytes to email message
        msg = email.message_from_bytes(file_obj)

        # Extract email body content
        content = ""
        if msg.is_multipart():
            for part in msg.walk():
                content_type = part.get_content_type()
                if content_type in ["text/plain", "text/html"]:
                    try:
                        content = part.get_payload(decode=True).decode(errors='ignore')
                        if content.strip():  # Make sure content is not empty
                            break
                    except:
                        continue
        else:
            content = msg.get_payload(decode=True).decode(errors='ignore')

        print("📝 EXTRACTED CONTENT:\n", content[:500])  # Optional debug preview

        # Predict
        cleaned = clean_text(content)
        vector = vectorizer.transform([cleaned])
        pred = model.predict(vector)

        return "□ Spam" if pred[0] == 1 else "□ Not Spam"

    except Exception as e:
        return f"❌ Error: {str(e)}"

import gradio as gr
```

```
interface = gr.Interface(
    fn=check_uploaded_email,
    inputs=gr.File(label="Upload Email (.eml)", type="binary"),
    outputs="text",
    title="📨 Enron Spam Detector",
    description="Upload an email file (.eml) to check if it's spam."
)

interface.launch()

from flask import Flask, render_template, request, jsonify
from utils import model_predict
app = Flask(_name_)



@app.route("/")
def home():
    return render_template("index.html")



@app.route('/predict', methods=['POST'])
def predict():
    email = request.form.get('content')
    prediction = model_predict(email)
    return render_template("index.html", prediction=prediction, email=email)

# Create an API endpoint
```

```
@app.route('/api/predict', methods=['POST'])
def predict_api():
    data = request.get_json(force=True)  # Get data posted as a json
    email = data['content']
    prediction = model_predict(email)
    return jsonify({'prediction': prediction, 'email': email})  # Return prediction

if _name_ == "_main_":
    app.run(host="0.0.0.0", port=8080, debug=True)
```

OUTPUT SCREENSHOTS



REFERENCE

1. Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., & Spyropoulos, C. D. (2000). *An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages*. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval.
2. Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). *Contributions to the study of SMS spam filtering: new collection and results*. In Proceedings of the 11th ACM symposium on Document engineering
3. Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). *Spam filtering with Naive Bayes – Which Naive Bayes?* In CEAS.
4. Sebastiani, F. (2002). *Machine learning in automated text categorization*. ACM Computing Surveys (CSUR), 34(1), 1–47.
5. Scikit-learn Developers. (2023). *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org/
6. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
7. SpamAssassin Public Corpus. (n.d.). *Apache SpamAssassin Public Mail Corpus*. Retrieved from http://spamassassin.apache.org/publiccorpus/
8. Enron Email Dataset. (n.d.). *Enron Email Dataset*. Retrieved from https://www.cs.cmu.edu/~enron/

9.Treviso, M.V.; Shulby, C.D.; Aluísio, S.M. Evaluating Word Embeddings for Sentence

10.Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*MITPress.(For understanding deep learning models that can be applied to advanced spam detection.)