Spam Mail Detection

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning

with

TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Fowziya Shaik, fowziyashaik7@gmail.com

Under the Guidance of

Rathod Jay

ACKNOWLEDGEMENT

I would like to express my profound gratitude to all the individuals who guided and supported me during this project.

Firstly, I extend my heartfelt thanks to my supervisor, Mr.Rathod Jay **Sir**, for his exceptional guidance, constructive feedback, and consistent encouragement throughout the duration of this project. His expertise and insights were invaluable and instrumental in completing this project successfully.

I also wish to acknowledge the **TechSaksham i**nitiative by **Microsoft & SAP** for providing this transformative learning opportunity. Lastly, I thank my family, peers, and friends for their unwavering support and encouragement.

FowziyaShaik

ABSTRACT

This report focuses on the project made, titled **Spam Mail Detection**, which is about developing an efficient and reliable system to classify emails as spam or ham (non-spam). With the rapid increase in unsolicited emails, spam detection has become a critical necessity to enhance user experience and ensure cybersecurity. The project leverages natural language processing (NLP) techniques and machine learning algorithms to address these challenges effectively.

The methodology begins with data preprocessing, including processes like tokenization, lemmatization, text normalization, punctuation removal, and case conversion, to prepare the raw email data for analysis. Feature extraction is performed using CountVectorizer, transforming text data into numerical representations suitable for machine learning models. A Naive Bayes classifier is chosen for its simplicity and efficiency in handling text classification tasks. The system is deployed using a web-based application built with Streamlit, enabling users to input emails in real time and receive instant classification results.

The implementation is tested on the widely used **spam.csv** dataset, which contains a collection of spam and ham emails. The model demonstrates high accuracy in distinguishing between spam and legitimate emails, achieving a overall score of 98.56 %.

The project concludes with a discussion on its impact and future improvements. Potential enhancements include support for multilingual datasets, integration of deep learning models for greater adaptability, and advanced techniques to counter adversarial spam tactics. Overall, the system offers a practical and scalable solution to combat email spam, showcasing the power of AI-driven automation in solving real-world challenges.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Problem Statement:

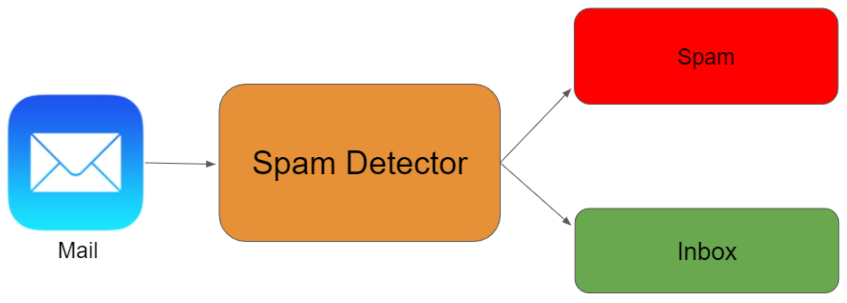
Unsolicited and harmful emails, commonly referred to as spam, have become a significant problem in the digital age. These emails not only clutter inboxes but also pose risks such as phishing attacks, malware distribution, and financial fraud. As email communication continues to be a primary means of interaction in personal and professional domains, the need to accurately distinguish between legitimate (ham) and spam emails becomes critical.

Current solutions, while effective to some extent, often fail in adapting to evolving spam tactics and result in misclassifications, causing user frustration or security breaches. Addressing this challenge requires robust machine learning models capable of adapting to dynamic datasets and effectively classifying emails with high accuracy.

1.2 Motivation:

The motivation for this project stems from the increasing dependence on email as a mode of communication and the adverse effects of spam emails on user productivity and security. The rise in phishing scams and malware attacks disguised as genuine emails highlights the urgent need for advanced detection systems. Moreover, manual filtering of emails is impractical due to the sheer volume of daily email exchanges.

This project seeks to bridge this gap by leveraging machine learning and natural language processing (NLP) techniques to build an automated spam detection system.



(Figure 1: Basic approach for building the system)

By automating the classification of spam and legitimate emails, this system not only enhances user experience but also ensures greater security for individuals and organizations.

Impact and Potential Applications :-

1. **Email Service Providers**: Integrating the classifier into email platforms (e.g., Gmail, Outlook) to improve spam filters.
2. **Enterprise Communication**: Ensuring secure and efficient communication channels within businesses.
3. **User Productivity**: Reducing the time spent manually filtering out spam emails.
4. **Security**: Preventing phishing and malware attacks, thereby protecting sensitive information and financial data.
5. **Educational Platforms**: Demonstrating the application of machine learning and NLP in real-world scenarios for students and researchers.

1.3 Objective:

The primary objectives of this project are:

1. **Developing a Robust Classifier**: To design and implement a machine learning model that accurately classifies emails as spam or ham.
2. **Preprocessing Email Data**: Applying tokenization and lemmatization techniques to clean and prepare text data for analysis.
3. **Feature Extraction**: Utilizing CountVectorizer to convert textual data into numerical format suitable for machine learning models.
4. **Model Training and Evaluation**: Training a Naive Bayes classifier and evaluating its performance using metrics like model score on training and testing data.
5. **Building a User-Friendly Application**: Developing a Streamlit-based web interface for real-time email classification.
6. **Ensuring Scalability and Adaptability**: Designing the system to adapt to new datasets and evolving spam patterns.

1.4 Scope of the Project:

The project focuses on designing a spam email detection system leveraging machine learning and NLP techniques. While it successfully addresses the core issue of email classification, several aspects define its scope and limitations:

Key Deliverables :-

1. **Spam Detection Model**: A machine learning model trained to classify spam and legitimate emails.
2. **Streamlit Application**: A functional web application for real-time email classification, enabling users to test the system.
3. **Preprocessing Techniques**: Implementation of tokenization and lemmatization for preparing email text data.

Limitations :-

1. **Dataset Dependency**: The accuracy of the model is dependent on the quality and diversity of the training dataset. A limited dataset may result in reduced performance on unseen data.
2. **Language Limitations**: The model focuses on English text and may not perform well with emails in other languages.
3. **Dynamic Nature of Spam**: Evolving spam tactics may require periodic retraining and updates to the model.
4. **Resource Constraints**: The use of computational resources for training and deploying the model may limit scalability in resource-constrained environments.

CHAPTER 2

Literature Survey

2.1 Literature Review

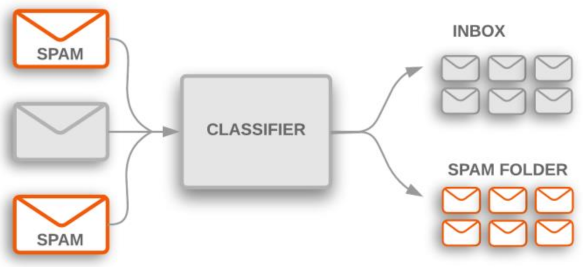
Spam email detection has been a topic of significant research due to its importance in safeguarding user data and preventing unwanted disruptions in communication. Over the years, numerous techniques have been proposed for spam classification, ranging from rule-based approaches to machine learning-based methods. Early approaches focused on keyword-based filtering, where specific terms associated with spam emails were identified and blocked. However, these methods often resulted in high false positive rates and were easily circumvented by spammers using obfuscation techniques.

The advent of machine learning provided a more robust solution to spam detection. Algorithms like Naive Bayes, Support Vector Machines (SVM), and decision trees were applied to classify emails as spam or ham (non-spam). Studies have shown that Naive Bayes, in particular, performs well due to its simplicity and efficiency, especially when combined with techniques like term frequency-inverse document frequency (TF-IDF) for feature extraction. More recent studies have shifted towards deep learning models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to capture complex patterns in text data.

2.2 Some Existing Models, Techniques or Methodologies

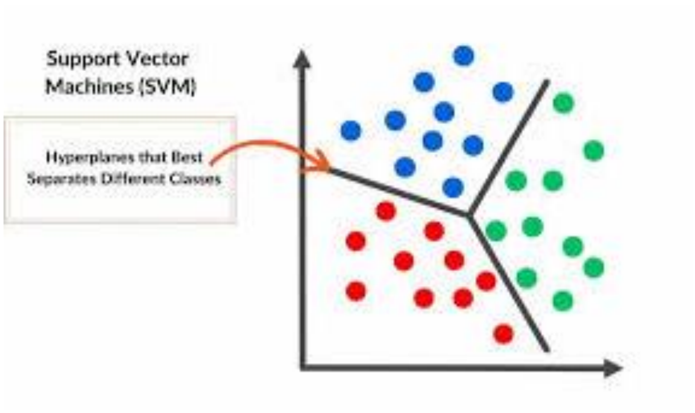
Several models and techniques have been applied to spam email detection:

- **Naive Bayes Classifier**: One of the most commonly used models for spam detection due to its simplicity and effectiveness. It assumes that the presence of a word in an email is independent of the presence of other words and calculates the probability of the email being spam or not.



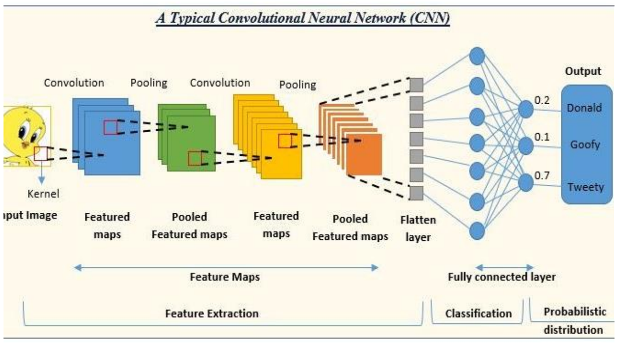
(Figure 2 : Naive Bayes Classifier for SPAM Mails)

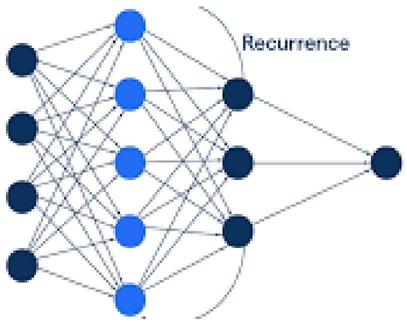
- **Support Vector Machines (SVM)**: This model is used in spam detection due to its ability to create complex decision boundaries in high-dimensional spaces. It is particularly effective when the dataset is linearly separable.


Support Vector Machines (SVM)
Hyperplanes that Best Separates Different Classes

(**Figure 3: Support Vector Machines [SVM]**)

- **Deep Learning Models**: Techniques like CNNs and RNNs are used for spam detection, particularly when large datasets are available. These models are capable of capturing complex relationships and patterns in text, leading to improved accuracy.





(**Figure 4: Convolutional Neural Network**)

(**Figure 5: Recurrent Neural Network**)

- **Ensemble Methods**: Combining multiple classifiers such as decision trees and Naive Bayes models can enhance performance by reducing overfitting and improving robustness.

- **Feature Extraction Methods**: TF-IDF, Word2Vec, and CountVectorizer are popular methods for transforming raw text into numerical data that can be used by machine learning models.

2.3 Limitations in Existing Methodologies or Systems

Despite the progress in spam email detection, existing models still have some limitations:

1. **High False Positive Rate**: Many spam filters suffer from a high number of legitimate emails being classified as spam. This is especially a problem in more complex models where the line between spam and non-spam becomes blurred due to the vast diversity in email content.

2. **Dependence on Keyword-based Features**: While techniques like TF-IDF and CountVectorizer are effective, they may fail to capture context and semantic meaning, which can limit their ability to classify more sophisticated spam.

3. **Evolving Nature of Spam**: Spammers constantly adapt their techniques, using more sophisticated obfuscation methods that bypass traditional models.

How This Project Will Address These Gaps:

The current project, **Spam Email Detection using Naive Bayes**, addresses some of these gaps by focusing on a simplified but effective approach for spam classification. The use of **CountVectorizer** for feature extraction offers a simple yet effective way to convert text into numerical data, allowing the Naive Bayes classifier to identify spam.

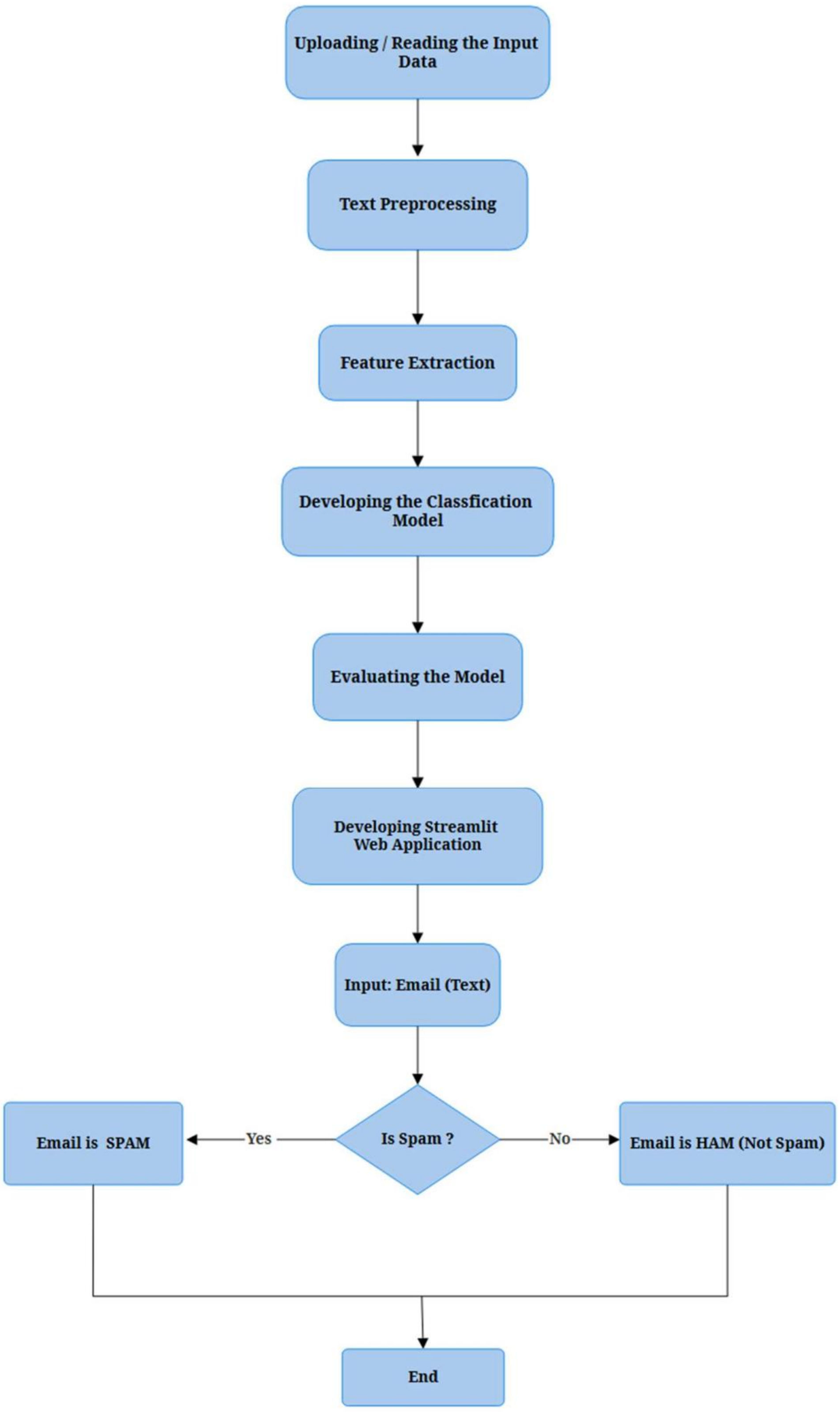
While **deep learning models** can capture more complex patterns, this project opts for a more balanced and interpretable approach using Naive Bayes, which is more suited for smaller datasets or when computational efficiency is a priority. The focus will be on improving the feature extraction process and experimenting with different configurations to improve the overall performance.

Additionally, the project will use **Streamlit** to build an interactive application for users to easily classify emails, providing a more accessible interface for real-world application. This is in line with the growing trend towards user-friendly AI-powered applications, addressing the limitation of complex models that may require specialized knowledge to operate.

CHAPTER 3

Proposed Methodology

3.1 System Design (Workflow Diagram)



The design of the Spam Email Classifier system is represented through a flowchart comprising two main sections: **Model Development** and **Application Deployment**. Each section outlines specific processes that collectively ensure the system's accuracy and usability.

Model Development Phase:

The first part of the flowchart explains the step-by-step procedure for creating a machine learning model to classify emails as spam or ham:

1) **Uploading/Reading the Input Data**:
   - The system starts by loading the dataset (spam.csv) into the environment. This dataset contains pre-labeled email samples, where each email is categorized as spam or ham.
   - This data serves as the basis for training and validating the classifier.

2) **Text Preprocessing**:
   - Raw email texts undergo cleaning and normalization, including:
     - Removing unnecessary characters (e.g., punctuation, special symbols).
     - Converting all text to lowercase to maintain uniformity.
     - Tokenizing the text into individual words or terms for further processing.

3) **Feature Extraction**:
   - The cleaned and tokenized text is converted into a numerical representation using **CountVectorizer**.
   - This step transforms textual data into a matrix of word frequencies, which acts as input for the classification model.

4) **Developing the Classification Model**:
   - A **Naive Bayes Classifier** is trained on the feature vectors generated in the previous step.
   - The model learns to differentiate between spam and ham emails based on word patterns and distributions in the dataset.

5) **Evaluating the Model**:
   - The model is rigorously tested on unseen data to assess its performance based on model score.
   - Evaluation ensures that the classifier is reliable and robust for real-world use.

Application Deployment Phase

The second part of the flowchart transitions from model creation to user interaction, explaining the deployment of the classifier through a web application:

1. **Developing the Streamlit Web Application**:
   - The trained Naive Bayes model is deployed as an interactive application using **Streamlit**, enabling real-time email classification.
   - The app features an intuitive interface for users to input email content.
2. **Input: Email (Text)**:
   - Users provide the content of an email (as plain text) through the web application's input form.
3. **Is Spam? (Decision Point)**:
   - The input email is processed using the same text preprocessing and feature extraction pipeline as during model training.
   - The classifier predicts whether the email is spam or ham:
     - **If "Yes"**: The email is labeled as spam and the result is displayed to the user.
     - **If "No"**: The email is classified as ham (not spam), and this result is also presented.
4. **Output Presentation**:
   - The classification result is displayed in the application with clear labels indicating whether the email is spam or ham.
5. **End of Workflow**:
   - The process ends, and the system becomes ready for the next email input.

3.2 Requirement Specification

3.2.1 Functional Requirements

1. **Data Preprocessing**:
   - Tokenization: Splitting email text into individual tokens (words).
   - Lemmatization: Reducing words to their base forms for better analysis.
   - Removal of unnecessary characters, stopwords, and special symbols.
2. **Feature Extraction**:
   - Convert text data into numerical format using CountVectorizer.
3. **Model Development**:
   - Train a Naive Bayes classifier on preprocessed and vectorized data.
   - Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.
4. **Web Application**:
   - Build an interactive user interface using Streamlit.
   - Provide real-time email classification (spam or ham) based on user input.

3.2.2 Non-Functional Requirements

1. **Usability**:
   - Ensure a user-friendly interface for seamless interaction.
2. **Scalability**:
   - Design the system to handle a larger volume of emails and adapt to new datasets.
3. **Performance**:
   - Optimize the application for quick and accurate classifications.
4. **Maintainability**:
   - Write modular and well-documented code to facilitate future updates.
5. **Security**:
   - Ensure that user data is processed securely and confidentially.

3.2.3 Software Requirements

1. **Programming Language**: Python
2. **Libraries**:
   - Pandas: For data manipulation.
   - Scikit-learn: For machine learning algorithms and evaluation metrics.
   - Streamlit: For building the web application.
3. **Development Environment**: Jupyter Notebook, VS Code
4. **Dataset**: "spam.csv" dataset

3.2.4 Hardware Requirements

1. **Processor**: Minimum 2 GHz dual-core CPU
2. **RAM**: At least 4 GB (8 GB recommended for better performance)
3. **Storage**: Minimum 500 MB of free space
4. **Internet Connection**: Required for Streamlit application deployment and testing.

Constraints

1. **Dataset**: The system's accuracy depends on the diversity and quality of the dataset.
2. **Language**: The model is trained only on English-language emails.
3. **Real-time Performance**: Latency issues may arise with limited hardware resources.

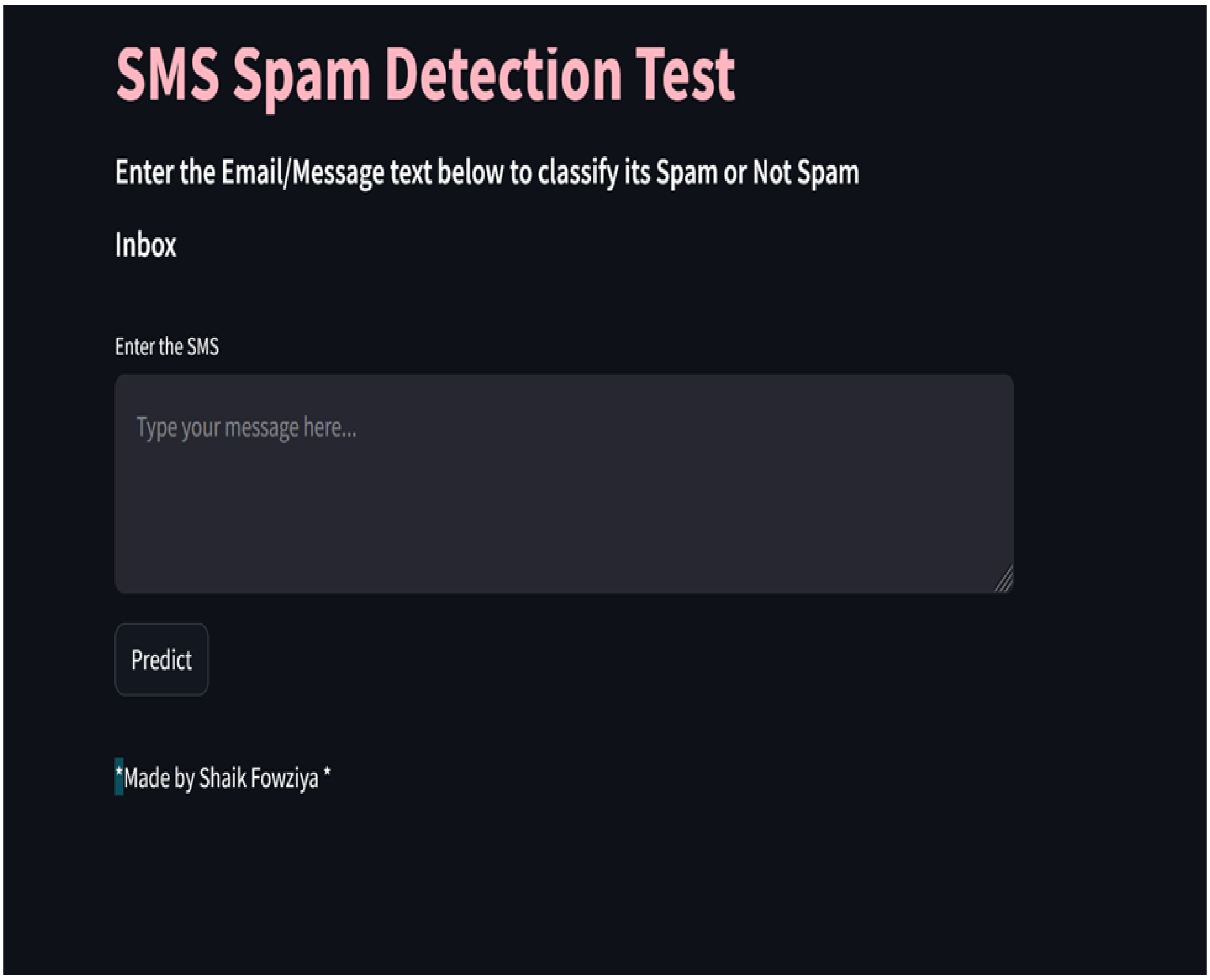
Assumptions

1. The input email data is primarily in English.
2. The system users are familiar with basic email classification concepts.
3. The dataset used for training and testing represents a realistic mix of spam and legitimate emails.

CHAPTER 4

Implementationand Result

4.1 Snap Shots of Result:


SMS Spam Detection Test

Enter the Email/Message text below to classify its Spam or Not Spam

Inbox

Enter the SMS

Type your message here...

Predict

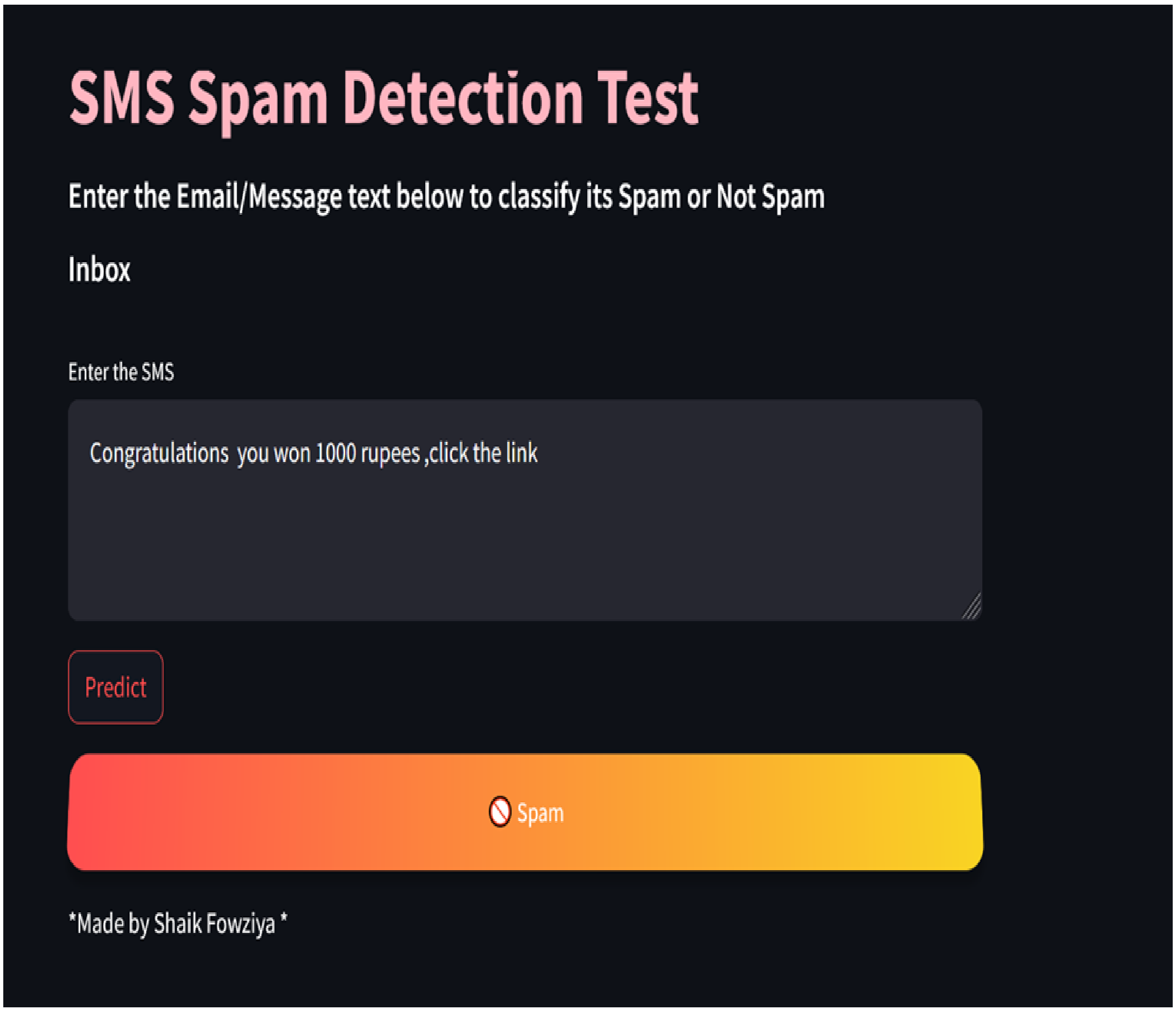
*Made by Shaik Fowziya *

(Snapshot 1: Main Interface)

Explanation:

The first snapshot shows the main interface of the Spam Email Classifier.

- This interface is built using **Streamlit**, providing a user-friendly platform where users can input their email text.
- Users can paste or type their email content into the text box provided and click the **"Classify Email"** button to get the result.


SMS Spam Detection Test

Enter the Email/Message text below to classify its Spam or Not Spam

Inbox

Enter the SMS

Congratulations you won 1000 rupees ,click the link

*Made by Shaik Fowziya *

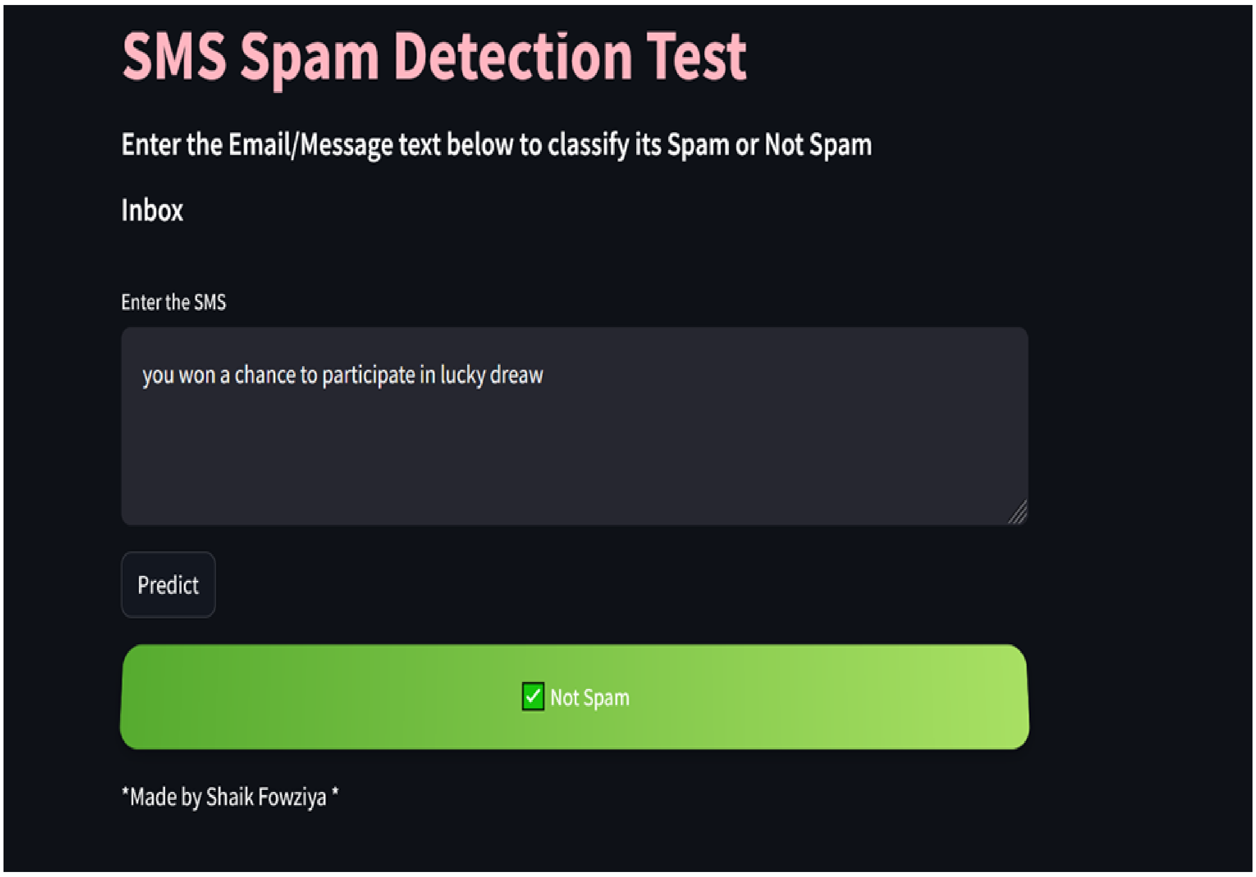
(Snapshot 2: Email Predicted as SPAM)

Explanation:

The second snapshot demonstrates the system classifying an email as **SPAM**.

- The email input contains typical spam-like content, such as promotional offers, clickbait phrases, or deceptive language.
- The classifier processes the input text through its **Naive Bayes model**, using previously trained data to identify spam-related patterns.
- The result, **"Spam"**, is displayed along with optional confidence scores or a message indicating why it was classified as spam.

This shows the system's ability to detect spam emails effectively.


SMS Spam Detection Test

Enter the Email/Message text below to classify its Spam or Not Spam

Inbox

Enter the SMS

you won a chance to participate in lucky dreaw

✅ Not Spam

*Made by Shaik Fowziya *

(Snapshot 3: Email Predicted as HAM [Not Spam])

4.2 GitHub Link for Code:

https://github.com/fowziyashaik/Spam.github.io.git

CHAPTER 5

Discussion and Conclusion

The Spam Email Classifier system presented in this project effectively addresses the need for automated email filtering using a combination of machine learning and natural language processing techniques. The implementation using a **Naive Bayes Classifier** ensures simplicity and efficiency, while the **CountVectorizer** provides a robust method for feature extraction.

The project demonstrates the practical utility of such systems in real-world scenarios, particularly in reducing the burden of manual email filtering. By leveraging a labeled dataset (`spam.csv`), the model achieves reliable classification of spam and ham emails. However, certain limitations exist, and these form the basis for further improvement:

1. **Performance on Complex Data**:
   The current model may struggle with more sophisticated spam emails, such as those using obfuscation techniques or multilingual content. This highlights the need for advanced feature extraction and dynamic model updating.
2. **Scalability**:
   While the system is effective for a modest dataset, scaling it to handle larger datasets or integrate with enterprise-level email services may require architectural modifications.
3. **Explainability and User Trust**:
   Providing users with insights into how classification decisions are made (e.g., keyword highlighting) could enhance the system's usability and transparency.
4. **Evolving Threat Landscape**:
   As spam techniques evolve, the static nature of the trained model could result in reduced effectiveness over time. Continuous model training on updated datasets is necessary to maintain accuracy.

The discussion emphasizes the project's strengths and its potential for future enhancements, paving the way for broader adoption and improved functionality.

5.1 Future Work:

While the project achieves its primary objectives, several avenues for improvement remain:

➢ **Enhanced Feature Extraction**:
Using advanced vectorization techniques like **TF-IDF**, **Word2Vec**, or **BERT** embeddings could capture more nuanced features in email content, leading to better classification accuracy.

➢ **Adopting Advanced Models**:
Exploring deep learning approaches, such as **LSTM**, **RNN**, or **Transformers**, can improve the model's ability to handle complex spam patterns and context-rich emails.

➢ **Support for Multilingual Emails**:
Expanding the system's capability to process non-English emails would significantly increase its applicability across diverse user bases.

➢ **Dynamic Model Updating**:
Implementing mechanisms for real-time model retraining or periodic updates with newly labeled datasets can help adapt to evolving spam patterns.

➢ **User-Focused Features**:
Adding features like keyword-based explanations for classification or interactive visual insights (e.g., word clouds, spam trends) can improve user engagement and trust.

➢ **Integration with Real-World Systems**:
Embedding the classifier into email services (e.g., Gmail, Outlook) or deploying it as a browser extension could enhance its practical usability.

5.2 Conclusion:

The Spam Email Classifier system successfully demonstrates the application of machine learning to tackle the pervasive problem of email spam. By leveraging the computational efficiency of the **Naive Bayes Classifier** and the simplicity of **CountVectorizer**, the project delivers accurate and reliable results.

Key Contributions:

- Developed a lightweight and accessible system with a user-friendly **Streamlit** interface.
- Demonstrated the effectiveness of a Naive Bayes-based approach for email classification.
- Provided a foundation for future enhancements in spam detection systems.

This project underscores the importance of integrating machine learning with user-centric design to solve real-world problems. The Spam Email Classifier lays the groundwork for more sophisticated and scalable solutions, ensuring a cleaner and more efficient email communication environment.

REFERENCES

[1] Al-Ghamdi, J., Al-Harbi, S., & Al-Harthi, S. (2020). **"Spam Email Classification Using Machine Learning Techniques."** *International Journal of Advanced Computer Science and Applications (IJACSA)*, 11(2), 1-8.

- DOI: 10.14569/IJACSA.2020.0110201

[2] Ramos, J. (2003). **"Using TF-IDF to Determine Word Relevance in Document Queries."** *Proceedings of the First International Conference on Machine Learning (ICML).*

- Available at: **https://www.cs.rutgers.edu**

[3] McCallum, A., & Nigam, K. (1998). **"A Comparison of Event Models for Naive Bayes Text Classification."** *AAAI-98 Workshop on Learning for Text Categorization.*

- URL: **https://www.cs.cmu.edu**

[4] Bird, S., Klein, E., & Loper, E. (2009). **"Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit."** *O'Reilly Media.*

- ISBN: 978-0596516499

[5] Zhang, Y., & Jin, R. (2021). **"Text Classification Algorithms: A Survey."** *ACM Transactions on Knowledge Discovery from Data*, 15(5), 1-37.

- DOI: 10.1145/3422811

[6] Streamlit Documentation. (n.d.). **"Streamlit: Build Data Apps in Python."**

- Available at: **https://docs.streamlit.io**

[7] The SpamAssassin Project. (n.d.). **"Spam Email Datasets for Machine Learning."**

- URL: **https://spamassassin.apache.org**

[8] Kaur, P., & Chhabra, A. (2014). **"Improved Spam Detection Using Machine Learning Techniques."** *International Journal of Computer Science and Information Technologies (IJCSIT)*, 5(2), 1165-1170.

- Available at**: http://ijcsit.com**