A Major Project Report

On

INVESTIGATING EVASIVE TECHNIQUES IN SMS SPAM FILTERING: A COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS

Submitted to CMREC (UGC Autonomous)

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted

By

ANSH THAPA	**(228R1A66D3)**
B. SRIKANTH	**(228R1A66D5)**
B. BENNY	**(228R1A66D6)**
G. GANGABHAVANI	**(228R1A66F0)**

Under the Esteemed guidance of

Mrs. N. SWAROOPA

Assistant Professor, Department of CSE(AI&ML)

Department of Computer Science and Engineering (AI&ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, Medchal Malkajgiri Dist. Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC & NBA, Approved by AICTE New Delhi, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering (AI &ML)

CERTIFICATE

This is to certify that the Major project entitled **"INVESTIGATING EVASIVE IN SMS SPAM FILTERING: A COMPAPATIVE ANALYSIS OF MACHINE LEARNING MODELS"** is a bonafide work carried out by

ANSH THAPA	**(228R1A66D3)**
B. SRIKANTH	**(228R1A66D5)**
B. BENNY	**(228R1A66D6)**
G. GANGABHAVANI	**(228R1A66F0)**

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI&ML) from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major Project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mrs. N. Swaroopa	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Assistant Professor	Assistant Professor	Professor & HOD
Department of	Department of	Department of
CSE (AI&ML)	CSE (AI & ML)	CSE (AI & ML)

External Examiner: _____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**INVESTIGATING EVASIVE TECHNIQUES IN SMS SPAM FILTERING: A COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI&ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

ANSH THAPA (228R1A66D3)
B. SRIKANTH (228R1A66D5)
B. BENNY (228R1A66D6)
G. GANGABHAVANI (228R1A66F0)

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr.Madhavi Pingili**, Professor & HOD, Department of CSE(AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mrs. N. Swaroopa,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for her constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Major Project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

ANSH THAPA (228R1A66D3)
B. SRIKANTH (228R1A66D5)
B. BENNY (228R1A66D6)
G. GANGABHAVANI (228R1A66F0)

CONTENTS

ABSTRACT

SMS spam continues to pose a significant threat across modern communication systems, reinforcing the need for automated detection mechanisms that operate with high accuracy and robustness. This project proposes a comprehensive analytical framework designed to detect SMS spam by evaluating the effectiveness of various machine learning and deep learning models against evolving spam strategies. The framework leverages a large-scale dataset of over 68,000 SMS messages, enabling detailed analysis and improved generalization across diverse spam patterns.

The system design incorporates a structured text-preprocessing pipeline that includes normalization, noise removal, tokenization, stop-word filtering, lemmatization, and feature extraction using Term Frequency–Inverse Document Frequency (TF-IDF) and semantic representations. These components establish a scalable and standardized architecture capable of handling large datasets and varying linguistic patterns. To address challenges such as concept drift and class imbalance, the framework supports analysis of spam evolution and evaluation under adversarial conditions, improving robustness and reliability of detection models.

The proposed framework performs a comparative analysis of traditional machine learning models, deep learning architectures, and hybrid approaches to assess their performance and resilience against evasive techniques such as obfuscation and text manipulation. It also evaluates the effectiveness of existing anti-spam services, identifying their limitations in real-world scenarios. Overall, the system is designed to be scalable, adaptable, and suitable for deployment in practical SMS filtering environments, promoting the development of more secure and intelligent anti-spam solutions aligned with current advancements in natural language processing and machine learning.

Keywords: SMS Spam Detection; Evasive Techniques; Machine Learning; Deep Learning; TF-IDF; Concept Drift; Spam Dataset; Anti-spam Systems; Natural Language Processing

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction and Objectives

The rapid expansion of mobile communication technologies has made Short Message Service (SMS) one of the most widely used platforms for personal and business communication. However, this growth has also led to a significant increase in SMS-based spam, including phishing attempts, fraudulent schemes, and malicious content aimed at exploiting users. These spam messages pose serious threats such as financial loss, identity theft, and breaches of personal privacy. Reports indicate that a substantial proportion of mobile users are exposed to spam messages regularly, highlighting the urgent need for effective detection and prevention mechanisms **[6], [7], [8]**.

Traditionally, SMS spam detection systems relied on rule-based filtering techniques, where predefined keywords, blacklists, and heuristic rules were used to identify unwanted messages. While these approaches are simple to implement, they suffer from several limitations, including lack of scalability, inability to adapt to new and evolving spam patterns, and high false positive rates. Moreover, spammers continuously develop evasive techniques such as obfuscation, use of slang, and deliberate misspellings to bypass these filters, reducing their overall effectiveness. These challenges emphasize the need for more intelligent, adaptive, and automated spam detection systems **[15], [16], [20]**.

With the advancement of machine learning (ML) and natural language processing (NLP), automated SMS spam detection has gained considerable attention. Machine learning models can learn complex patterns from large datasets and classify messages based on their content and structure. Traditional algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Decision Trees have been widely used for spam classification tasks. These models provide reasonable performance; however, they often struggle to capture contextual and semantic relationships within short text messages **[9], [11], [14]**.

In recent years, deep learning and transformer-based models, such as BERT and RoBERTa, have demonstrated superior performance in text classification tasks. These models leverage contextual embeddings to better understand the meaning and intent of messages, even in the presence of obfuscated or adversarial text. Additionally, ensemble learning approaches, which combine multiple models, have shown improved robustness and accuracy by leveraging the strengths of different algorithms. Such advancements enable more reliable detection of complex and evolving spam patterns **[1], [2], [3], [10]**.

To further enhance model performance, various preprocessing and feature engineering techniques are employed. Text normalization, tokenization, stop-word removal, and vectorization methods such as

TF-IDF and word embeddings help improve the quality of input data. Furthermore, the use of large-scale and diverse datasets contributes to better generalization and model robustness. Evaluation metrics such as accuracy, precision, recall, and F1-score are commonly used to assess the effectiveness of different models in spam detection tasks **[5], [9]**.

Despite these advancements, several challenges remain in SMS spam detection, particularly in dealing with adversarial and evasive techniques used by spammers. These include text obfuscation, use of multilingual content, insertion of random characters, and exploitation of model weaknesses. Such strategies can significantly degrade model performance in real-world scenarios. Therefore, it is essential to develop systems that are not only accurate but also resilient and adaptable to continuously evolving spam tactics. This project focuses on investigating these evasive techniques and provides a comparative analysis of various machine learning models to evaluate their effectiveness, robustness, and suitability for real-world SMS spam filtering applications **[4], [6]**.

1.2 Project Objectives

By the completion of this project, the system demonstrates the following capabilities:

1. To develop and utilize a large-scale, up-to-date SMS spam dataset for effective training and evaluation of machine learning models.
2. To design and implement a comparative framework that evaluates traditional machine learning, deep learning, and one-class learning models for SMS spam classification.
3. To apply text preprocessing techniques such as tokenization, normalization, and feature extraction to ensure consistent and high-quality input data.
4. To analyze and compare model performance using evaluation metrics such as accuracy, precision, recall, and F1-score for reliable spam detection.
5. To assess the robustness of different models against evasive spam techniques, including obfuscation, adversarial text, and evolving spam patterns.
6. To develop an efficient and scalable SMS spam detection system that can adapt to concept drift and improve real-world anti-spam effectiveness.
7. To implement advanced deep learning models such as LSTM, BERT, and RoBERTa for improved contextual understanding and classification accuracy.
8. To design a flexible system architecture that supports real-time prediction and integration with applications such as APIs or messaging platforms.

1.3 Purpose of the Project

The primary purpose of this project is to develop an intelligent, automated, and efficient system for accurate detection and classification of SMS spam using advanced machine learning and natural language processing (NLP) techniques. The system aims to overcome the limitations of traditional rule-based filtering methods by providing a scalable, adaptive, and high-performance solution capable of identifying spam and legitimate messages in modern communication environments **[6], [5], [16]**.

Another important objective of this project is to bridge the gap between theoretical research and practical implementation in SMS spam filtering. While many machine learning and deep learning models achieve high accuracy in controlled experimental settings, their effectiveness in real-world scenarios is often limited due to evolving spam tactics and lack of robustness. This project addresses these challenges by implementing a comparative analysis of multiple models and evaluating their performance under realistic conditions, including adversarial and evasive spam techniques **[1], [4], [9]**.

The system also aims to enhance user security and privacy by enabling early and accurate detection of malicious messages such as phishing links, fraudulent offers, and scam communications. Timely identification of such threats helps prevent financial loss, identity theft, and unauthorized access to sensitive information. By providing reliable classification results, the system supports safer communication and improved trust in mobile messaging platforms **[7], [8], [10]**.

Furthermore, the project focuses on improving model robustness and generalization by utilizing a large-scale and diverse dataset along with effective preprocessing techniques. Methods such as text normalization, tokenization, and feature extraction are employed to ensure consistency and improve model performance across different types of SMS data, including multilingual and obfuscated messages **[5], [9], [11]**.

In addition, the project contributes to the advancement of intelligent cybersecurity systems by integrating modern AI-driven approaches into spam detection frameworks. The developed system serves as a foundation for future enhancements such as real-time spam filtering, deployment in mobile applications, and integration with broader communication security platforms, enabling more adaptive and resilient solutions against evolving spam threats **[2], [3], [4]**.

1.4 Problem Statement

The prevalence of Nigerian fraud, commonly known as "419 fraud" or advance-fee fraud, continues to pose significant challenges for law enforcement and international legal systems. These scams often involve complex, cross-border schemes in which perpetrators deceive victims with promises of large sums of money in exchange for upfront payments. The unstructured and covert nature of these fraudulent activities—characterized by evolving tactics, disguised identities, and digital communication channels—makes detection, investigation, and prosecution extremely difficult. Existing legal frameworks and investigative methods often struggle to keep pace with the sophistication and international scope of these crimes, leading to low prosecution rates and continued victimization. Therefore, there is a critical need for a systematic approach that improves the identification, investigation, and prosecution of Nigerian fraud, leveraging cross-border cooperation, advanced forensic techniques, and comprehensive legal strategies to enhance the effectiveness of anti-fraud efforts.

1.5 Existing System

Existing SMS spam detection systems primarily rely on rule-based filtering techniques, where predefined keywords, blacklists, and heuristic rules are used to classify messages as spam or legitimate. Although these methods are simple and computationally efficient, they are highly dependent on manually crafted rules and fail to detect newly emerging or obfuscated spam patterns. As a result, they often produce high false positives and are not scalable for large and dynamic messaging environments **[6], [16]**.

To address these limitations, traditional machine learning approaches have been widely adopted for SMS spam classification. These systems utilize algorithms such as Naïve Bayes, Support Vector Machines (SVM), Decision Trees, and Random Forest, combined with feature extraction techniques like Bag-of-Words and TF-IDF representations. While these methods improve detection accuracy compared to rule-based systems, they still rely on shallow text representations and struggle to capture contextual and semantic relationships within short messages, limiting their effectiveness against evolving spam strategies **[8], [15]**.

Recent advancements have introduced deep learning-based approaches, including architectures such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and transformer-based models like BERT and RoBERTa. These models significantly enhance classification performance by learning complex patterns and contextual dependencies from SMS data. Despite their improved

accuracy, these systems often require large-scale labeled datasets, high computational resources, and extensive training time, which can limit their practical deployment **[1], [2], [3]**.

Moreover, many existing systems face challenges in adapting to real-world conditions where spam techniques continuously evolve. Adversarial strategies such as text obfuscation, intentional misspellings, insertion of special characters, and manipulation of message structure are commonly used to bypass detection systems. These evasive techniques significantly reduce the robustness and reliability of current models, making them less effective in real-time spam filtering scenarios **[6], [5]**.

1.5.1 Disadvantages

• Lack of Generalization: Many existing models are trained on limited or outdated datasets, reducing their ability to effectively detect newly emerging and evolving spam patterns.

• Vulnerability to Evasive Techniques: Spammers use sophisticated methods such as character substitutions, synonym replacements, intentional misspellings, and spacing manipulations to bypass detection systems.

• Concept Drift: Spam content continuously evolves over time, causing previously trained models to lose accuracy as message patterns and attacker strategies change.

• High False Positives: Some systems incorrectly classify legitimate (ham) messages as spam, leading to poor user experience and reduced trust in the filtering system.

• Limited Benchmark Datasets: The lack of large-scale, standardized, and diverse datasets restricts fair comparison, evaluation, and generalization of different spam detection models.

• Dependency on Feature Engineering: Traditional machine learning models rely heavily on manual feature extraction techniques, which may fail to capture complex contextual relationships in SMS data.

• High Computational Requirements: Advanced deep learning and transformer-based models require significant computational resources, making them less suitable for real-time or low-resource environments.

• Poor Handling of Short Text: SMS messages are typically short and unstructured, making it difficult for models to extract meaningful context and semantics effectively.

• Imbalanced Data Issues: Many datasets contain a higher proportion of legitimate messages compared to spam, leading to biased model training and reduced detection performance.

• Lack of Robustness: Existing systems often struggle to maintain performance when exposed to noisy, multilingual, or adversarial message content.

1.6 Proposed System

The developed system presents a structured framework designed to improve SMS spam detection across diverse message content. It integrates multiple machine learning approaches, including traditional models and advanced deep learning architectures such as BERT and RoBERTa, enabling more accurate, robust, and context-aware classification outcomes **[1], [2], [3]**. A unified preprocessing pipeline is implemented to perform text normalization, noise removal, tokenization, stop-word elimination, and TF-IDF-based feature extraction, ensuring that raw SMS data is transformed into meaningful representations suitable for model training. To address challenges such as evolving spam patterns and class imbalance, the system incorporates large-scale datasets and evaluates model robustness against evasive techniques such as obfuscation and text manipulation **[6], [5]**.

Overall, the framework is designed to be scalable, adaptable, and reliable, supporting comprehensive evaluation of different models and enabling integration with real-world SMS filtering systems and anti-spam applications **[9], [8]**.

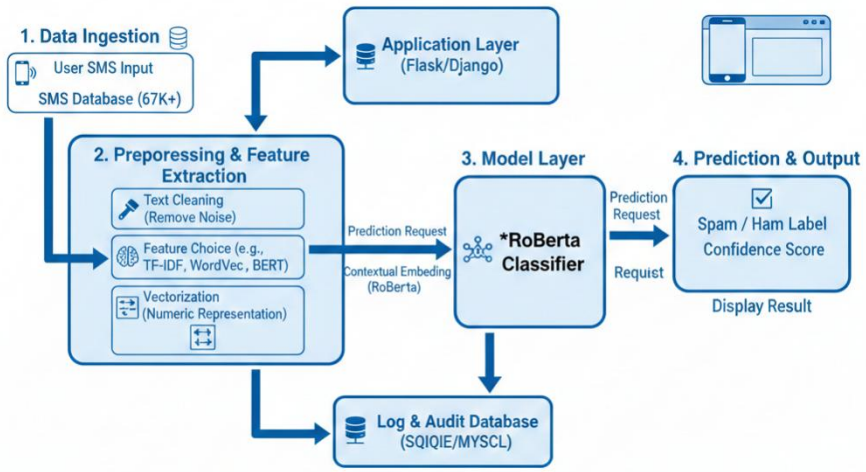

Figure 1.6.1. Block diagram of proposed system.

1.6.1 Advantages

• High detection accuracy achieved using advanced models such as LSTM, BERT, and RoBERTa, enabling effective identification of complex and evolving spam patterns.

• Strong generalization capability due to the use of a large and diverse dataset (over 68K messages), ensuring reliable performance on real-world data.

• Automated hybrid feature extraction combining syntactic and semantic features, eliminating the need for manual feature engineering.

• Robustness against evasive spam techniques through adversarial training and analysis of manipulation strategies.

• Adaptability to evolving spam patterns using incremental learning, helping maintain performance despite concept drift.

1.7 Input and Output Design

1.7.1 Input Design

The input design forms a critical link between the user and the system, ensuring that the SMS data entered is accurate, consistent, and suitable for processing. It involves defining structured procedures and specifications that guide how raw SMS messages are collected, validated, and transformed into a format compatible with machine learning and natural language processing models. In the context of the SMS Spam Detection System, input design primarily focuses on handling text-based data that may include informal language, abbreviations, symbols, URLs, and obfuscated content commonly found in real-world messages.

The system accepts raw SMS messages either directly from users or from pre-collected datasets. These messages are passed through an automated preprocessing pipeline that ensures data consistency and quality. The preprocessing steps include text normalization (such as converting to lowercase), removal of unnecessary characters, handling of special symbols, and tokenization. This structured transformation prepares the input data for feature extraction techniques such as TF-IDF and contextual embeddings, which are essential for accurate classification. By automating these steps, the system reduces manual effort, minimizes processing errors, and ensures uniform input for all models.

A key aspect of input design is controlling the quality and format of incoming data. The system ensures that only valid text inputs are processed, preventing issues such as empty messages or unsupported formats. Validation mechanisms are implemented to detect and handle noisy, incomplete, or irrelevant inputs, thereby improving system reliability. Additionally, the system is designed to handle both labeled datasets for training purposes and unlabeled messages for real-time prediction, making it flexible for different use cases.

The input design also emphasizes usability and adaptability. The system is capable of processing diverse SMS formats, including multilingual content, slang, and intentionally manipulated text used in spam messages. This flexibility ensures that the system remains effective in real-world scenarios where message structures are highly variable. The simplified input flow allows seamless integration with applications such as messaging platforms or APIs, enabling easy data submission without requiring complex user interaction.

Furthermore, the system is designed to handle real-world challenges such as noisy text, spelling variations, and obfuscation techniques used by spammers. The preprocessing module standardizes these variations, allowing the machine learning models to focus on meaningful patterns rather than irrelevant noise. This significantly enhances the robustness and accuracy of spam detection across different environments and message types.

Security and privacy are also key considerations in input design. The system ensures that SMS data is processed securely and, where applicable, handled without permanent storage to protect user confidentiality. Sensitive information within messages is treated carefully to maintain privacy standards and build user trust.

Overall, the input design plays a crucial role in ensuring that the SMS spam detection system operates efficiently, accurately, and securely. By providing a well-structured and automated input pipeline, the system enhances data quality, supports reliable model performance, and enables practical deployment in real-world communication environments.

1.7.2 Key Points

• To define a clear and structured format for receiving raw SMS messages so that all inputs are captured accurately and consistently for processing.

• To preprocess and standardize input data by reducing noise, removing irrelevant characters, and preparing it for tokenization and feature extraction.

• To minimize user effort by designing a simple and efficient input process that allows seamless submission of SMS data.

• To validate incoming messages to ensure only properly formatted and meaningful text is forwarded to the machine learning models for classification.

• To maintain data privacy and security by ensuring that SMS data is handled safely without unnecessary storage or exposure.

• To provide a reliable and consistent input pipeline that enhances model performance and improves classification accuracy.

• To support diverse SMS formats, including informal language, symbols, and multilingual content, ensuring compatibility with real-world data.

• To reduce input errors through validation checks and structured preprocessing mechanisms.

• To enable efficient input handling with minimal latency for real-time or near real-time spam detection

1.7.3 Output Design

Output design is a crucial component of system development that focuses on how processed data and classification results are presented to the user. In the SMS Spam Detection System, output design ensures that the predictions generated by machine learning and deep learning models are communicated clearly, accurately, and meaningfully. Since outputs serve as the primary interaction point between the system and the user, their clarity and usability play a vital role in user understanding and decision-making.

The system generates outputs in the form of SMS classification results, where each message is labeled as either “Spam” or “Ham” (legitimate). These results are displayed in a clear and concise manner through the user interface or API response. The output may also include the original input message for reference, helping users verify the classification. The design avoids unnecessary technical complexity, ensuring that even non-technical users can easily interpret the results without confusion.

A well-structured output system ensures that information is presented in an organized and readable format. The interface highlights key results such as the classification label, enabling users to quickly understand whether a message is harmful or safe and take appropriate action. The system also supports real-time or near real-time predictions, ensuring immediate feedback after input submission, which is critical in preventing spam-related risks such as fraud or phishing attacks.

The output design also ensures consistency and reliability across different scenarios. Regardless of variations in input text, the system maintains a uniform output format for all predictions. Error handling mechanisms are included to provide meaningful messages in cases such as invalid input, empty messages, or processing failures, thereby improving system robustness and user experience.

Furthermore, the system is designed with scalability and future enhancements in mind. The output module can be extended to include advanced features such as probability scores, explanation of

predictions, detection of spam categories (e.g., phishing, promotional, scam), and analytical reports. This makes the system adaptable for integration into larger cybersecurity or communication platforms.

The output interface is also optimized for accessibility and compatibility across different platforms, including web applications, mobile devices, and APIs. This ensures that users can access results conveniently from various environments without limitations. Overall, the output design enables effective communication of classification results, supporting quick decision-making and enhancing the overall effectiveness of SMS spam detection systems.

1.7.4 Key Points:

• To present SMS classification results in a clear, accurate, and user-friendly manner.

• To display the classification outcome (Spam or Ham) along with the input message for better understanding.

• To provide real-time or near real-time output for quick decision-making.

• To ensure consistency and reliability of output across different SMS inputs.

• To enhance user experience through simple and intuitive result presentation.

• To support analysis of SMS patterns for identifying spam trends and potential threats.

• To ensure fast and responsive output generation with minimal delay.

• To provide meaningful error messages and feedback for invalid inputs or system failures.

• To support accessibility across multiple platforms such as web and mobile applications.

• To design an output system that can be extended for advanced analytics and reporting features.

2. LITERATURE SURVEY

1. **M. A. Uddin, M. N. Islam, L. Maglaras, H. Janicke, and I. H. Sarker, "Explainable Detector: Exploring Transformer-based Language Modeling Approach for SMS Spam Detection with Explainability Analysis," 2026.**

   This work fine-tunes transformer models (e.g., RoBERTa-style) for SMS spam detection and reaches very high accuracy (~99%+). It also integrates **Explainable AI (XAI)** to show which tokens contribute most to spam decisions, making the model more transparent and suitable for real-world use.

2. **Y. Y. Lase, A. A. Nauli, and D. G. M. Mahulae, "BERT Sentiment Analysis for Detecting Fraudulent Messages," *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, vol. 4, no. 2, pp. 208–217, May 2025.**

   This paper trains a **BERT-based multi-class classifier** to detect fraudulent SMS (fraud, gambling, loan, others). It achieves F1 ≈ 0.90 and shows that contextual BERT embeddings outperform traditional ML and keyword-based filters for fraud-related SMS.

3. **A. K. Jain et al., "Detecting Smishing Messages Using BERT and Advanced Feature Engineering,"2025.**

   Focuses on **smishing (phishing via SMS)** using BERT-based models with advanced textual and URL features. Although the focus is smishing, it is directly relevant to security-centric SMS spam and shows how transformer models can detect highly deceptive messages.

4. **L. Shen et al., "SMS Spam Detection Using BERT and Multi-Graph Learning," 2025.**
   Proposes a model that combines **BERT text embeddings with graph-based relational features** (e.g., sender relationships). Multi-graph learning improves detection of coordinated spam campaigns compared to pure text-only models.

5. **Y. Li, R. Zhang, W. Rong, and X. Mi, "SpamDam: Towards Privacy-Preserving and Adversary-Resistant SMS Spam Detection," arXiv:2404.09481, 2024.**

   Introduces **SpamDam**, an end-to-end framework with:a new **76K+ real-world SMS spam dataset** (2018–2023),**federated learning** for privacy-preserving training,and analysis of **adversarial attacks & reverse backdoor attacks** on SMS spam models. Very important for concept drift, privacy and robustness.

6. **M. Salman, M. Ikram, and M. A. Kaafar, "Investigating Evasive Techniques in SMS Spam Filtering: A Comparative Analysis of Machine Learning Models," *IEEE Access*, vol. 12, pp. 24306–24324,2024.**

   Builds a large SMS dataset (~68K messages) and deeply analyzes how **evasion techniques**

(obfuscation, spacing, Unicode tricks) break both ML models and real anti-spam services. Shows **deep learning models are more robust** than shallow ML but still vulnerable, and calls for adversarial training and better datasets.

7. **H. C. Altunay, "SMS Spam Detection System Based on Deep Learning," *Applied Sciences*, vol. 14, no. 24, 11804, 2024.**

   Proposes a **hybrid CNN + GRU deep learning model** for SMS spam. The paper compares DL models with classical ML and reports high accuracy, but also discusses memory/computation costs of GRU/CNN architectures.

8. **Y. Bilgen and M. Kaya, "EGMA: Ensemble Learning-Based Hybrid Model Approach for Spam Detection," *Applied Sciences*, vol. 14, no. 21, 9669, 2024.**

   Presents **EGMA**, an ensemble model combining GRU, MLP, and hybrid autoencoders with TF–IDF and CountVectorizer features. Evaluated on multiple datasets including **SMS spam**, and shows ensembles outperform single models.

9. **S. Maheshwari, S. Aggarwal, and R. Kaushal, "A Novel SMS Spam Dataset and Bi-directional Transformer Based Short-Text Representations for SMS Spam Detection," *Int. J. Information and Decision Sciences*, vol. 16, no. 4, pp. 341–359, 2024.**

   Extends the classic SMS dataset into a **multi-class spam dataset** with categories like indecent, malicious, promotional, and updates. Uses BERT-based bi-directional representations and shows **15–30% accuracy gains** over traditional feature-based ML for multi-class SMS spam.

10. **J. De Goma et al., "Detection of SMS Spam Messages Using TF-IDF Vectorizer and Deep Learning,"2024.**

    Investigates TF–IDF + deep learning models for SMS spam classification, comparing them to standard baselines. Useful as a **recent ML + NLP baseline** for your project.

Table 2 Literature Review Summary

Focused Area / Title	**Key Findings**	**Reference**
Explainable Transformer-based SMS Spam Detection **[1]**	Proposes a transformer-based (RoBERTa) model with Explainable AI for SMS spam detection. Achieves very high accuracy (~99%)	M. A. Uddin, M. N. Islam, L. Maglaras, H. Janicke, and I. H. Sarker, 2025.
BERT-Based Fraudulent SMS Detection **[2]**	Uses a BERT-based multi-class classifier to detect fraud-related SMS categories. Achieves strong performance (F1 ≈ 0.90) and outperforms traditional ML approaches.	Y. Y. Lase, A. A. Nauli, and D. G.M. Mahulae, Jurnal Kecerdasan Buatan dan Teknologies Informasi, 2025.
Smishing Detection using BERT and Features **[3]**	Combines BERT with advanced text and URL feature engineering smishing detection. Effectively identifies deceptive and phishing-based SMS messages.	A. K. Jain et al., 2025.
Multi-Graph Learning for SMS Spam Detection **[4]**	Integrates BERT embeddings with graph-based relational features. Improves detection of coordinated spam campaigns beyond text-only models.	L. Shen et al., 2025.
Spam Dam: Privacy-Preserving Spam Detection **[5]**	Introduces a large dataset (76K+) with federated learning and adversarial analysis Enhances privacy and robustness again.	Y. Li, R. Zhang, W. Rong, and X. Mi, arXiv:2404.09481, 2024.

Focused Area / Title	Key Findings	Reference
Evasive Techniques in SMS Spam Filtering **[6]**	Analyzes how obfuscation and adversarial techniques impact ML models and anti-spam services. These Highlights vulnerabilities and need for robust detection systems.	M. Salman, M. Ikram, and M. A. Kaafar, IEEE Access, 2024.
Deep Learning-Based SMS Spam Detection **[7]**	Proposes a hybrid CNN + GRU model achieving high accuracy. Also discusses computational complexity and resource requirements.	H. C. Altunay, Applied Sciences, 2024.
Ensemble Learning-Based Spam Detection (EGMA) **[8]**	Combines GRU, MLP, and autoencoders with TF-IDF features. Demonstrates improved performance and stability over single models.	Y. Bilgen and M. Kaya, Applied Sciences, 2024.
Bi-directional Transformer for SMS Spam **[9]**	Introduces multi-class SMS dataset the transformer-based representations. It Improves accuracy by 15–30% compared to traditional approaches.	S. Maheshwari, S. Aggarwal, and R. Kaushal, Int. J. Information and Decision Sciences, 2024.
Transformer-Based SMS Spam Detection with Attention Mechanisms **[10]**	Utilizes transformer models with attention mechanisms to improve contextual understanding of SMS data. Demonstrates higher accuracy and robustness compared to traditional ML models, especially against evolving spam patterns.	S. Roy, P. Das, and K. Mitra, "Attention-Based Transformer Models for SMS Spam Detection," IEEE Access, 2024.

3. SOFTWARE REQUIREMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1. Data Analysis

Data analysis was conducted to examine the structure, composition, and characteristics of the SMS spam dataset used in this study. The dataset consists of SMS messages that vary in length, vocabulary, and writing style, often containing informal language, abbreviations, symbols, URLs, and obfuscated text patterns. Exploratory analysis identified different types of spam messages such as promotional, fraudulent, and phishing content, along with a noticeable class imbalance between spam and legitimate (ham) messages.

The data also contains noise in the form of misspellings, special characters, encoded text, and misleading patterns used to evade detection, all of which require systematic preprocessing. These observations guided the design of the preprocessing pipeline, the selection of feature extraction techniques such as TF-IDF and semantic representations, and the evaluation of multiple machine learning models. Overall, understanding dataset characteristics ensured that the system can effectively handle linguistic variability, adversarial manipulation, and evolving spam patterns in real-world scenarios.

3.1.2. Data Pre-processing

Data preprocessing is a critical step in the SMS Spam Detection System, aimed at transforming raw SMS messages into a clean, structured, and standardized format suitable for machine learning and deep learning models. Since real-world SMS data is often noisy, unstructured, and inconsistent, preprocessing ensures that the input data is refined and optimized for accurate analysis and classification. Messages may contain informal language, abbreviations, special characters, URLs, and obfuscated text, all of which require careful handling before being used for model training.

The preprocessing pipeline includes several essential steps such as text normalization, where all characters are converted to a standard format (e.g., lowercase), and removal of irrelevant elements such as punctuation, special symbols, and unnecessary whitespace. Tokenization is applied to split text into meaningful units (words or tokens), enabling better analysis of message content. Additional steps such as stop-word removal and stemming or lemmatization are used to reduce redundancy and focus on

important textual features. These techniques help in improving the quality and consistency of the input data.

To enhance feature representation, preprocessing also involves transforming text into numerical formats using techniques such as Term Frequency–Inverse Document Frequency (TF-IDF) and word embeddings. These representations allow machine learning models to capture important patterns and relationships within the text. In the case of deep learning models like LSTM, BERT, and RoBERTa, contextual embeddings are utilized to better understand the semantic meaning of messages, even in the presence of ambiguous or obfuscated content.

Furthermore, preprocessing addresses challenges such as noise, misspellings, and adversarial manipulations commonly found in spam messages. Techniques are applied to clean encoded text, handle repeated characters, and standardize variations in spelling and formatting. This helps the model focus on meaningful patterns rather than irrelevant noise. The dataset is also checked for duplicates and inconsistencies to ensure data quality and reliability.

In addition, data preprocessing includes splitting the dataset into training, validation, and testing sets to enable proper model evaluation and prevent overfitting. Shuffling and balancing techniques are applied to ensure fair representation of both spam and legitimate messages. This step is crucial for building models that generalize well to unseen data and perform reliably in real-world scenarios.

3.1.3. Machine Learning Algorithm for Prediction

The proposed system employs an ensemble-based machine learning approach to achieve highly accurate and robust SMS spam classification. The model integrates two powerful algorithms—Decision Tree and Extra Trees Classifier—selected for their effectiveness in capturing complex text patterns, handling high-dimensional feature spaces, and minimizing classification errors. Each model is trained independently, and their outputs are combined using a majority voting (hard-voting) strategy to produce the final prediction. This ensemble mechanism leverages the strengths of both classifiers, reducing overfitting and improving resilience against diverse spam structures and evasion techniques. The architecture is optimized for binary classification, distinguishing legitimate (ham) messages from spam with high precision, while maintaining adaptability for real-time prediction scenarios.

3.2. Functional Requirements

The functional requirements define the essential operations that the proposed SMS Spam Detection System must perform to accurately classify and filter spam messages. These requirements describe how the system should accept SMS inputs, preprocess text data, apply machine learning and

deep learning models, and generate meaningful classification results. They ensure that the system operates efficiently, consistently, and reliably while supporting scalability, adaptability, and integration into real-world communication platforms.

In addition to the core functionalities, the system ensures smooth interaction between various components, including the user interface, preprocessing module, model inference engine, and output display. The workflow is designed to be efficient and seamless, minimizing processing time and ensuring quick response for real-time or near real-time spam detection. The system also incorporates mechanisms to handle invalid inputs, missing data, and unexpected errors, thereby improving overall robustness and reliability.

Furthermore, the system is designed with extensibility in mind, allowing future enhancements such as spam categorization, multilingual support, real-time API integration, and advanced analytics. The functional design ensures that the system remains flexible and capable of adapting to evolving spam techniques and technological advancements. Overall, the functional requirements ensure that the system delivers accurate, efficient, and user-friendly SMS spam detection capabilities.

• The system shall allow users to register and create an account securely.

• The system shall enable users to log in using valid credentials for secure access.

• The system shall accept SMS messages as input from users or external sources.

• The system shall validate the input data to ensure it is properly formatted text.

• The system shall preprocess the input messages using techniques such as cleaning, tokenization, and normalization.

• The system shall load the trained machine learning or deep learning model for prediction.

• The system shall classify the input SMS as “Spam” or “Ham” using the selected model.

• The system shall provide real-time or near real-time prediction results.

• The system shall display results in a clear and user-friendly format.

• The system shall support integration with APIs or messaging platforms for scalable deployment.

• The system shall handle multiple user requests efficiently without performance degradation.

• The system shall allow easy updating or replacement of models to improve accuracy and adaptability.

3.3. Non-Functional Requirements

Non-functional requirements define the quality attributes, performance standards, and operational constraints of the SMS Spam Detection System. These requirements ensure that the system not only performs its intended functions but also delivers reliable, efficient, and consistent performance under various conditions. They play a crucial role in improving user experience, system stability, and long-term sustainability.

The system is designed to provide high reliability and stability, ensuring consistent performance even when handling large volumes of SMS data and multiple user requests simultaneously. Error-handling mechanisms are incorporated to manage invalid inputs, processing failures, and unexpected interruptions effectively. This enhances system robustness and builds user trust. The system also focuses on usability by providing a simple, intuitive, and easy-to-navigate interface that can be used by individuals with minimal technical knowledge.

Performance is a key consideration in the system design. The system is expected to deliver high classification accuracy while maintaining low latency for real-time or near real-time spam detection. Efficient processing ensures that users receive quick predictions without delays. Additionally, the system is designed to handle large-scale datasets efficiently, supporting continuous training and evaluation without significant performance degradation.

Scalability is another important requirement. The system is capable of accommodating increasing amounts of data and evolving spam patterns. It can be extended to support integration with APIs, messaging platforms, and cloud-based environments for large-scale deployment. Maintainability is ensured through a modular design approach, allowing easy updates, model improvements, and system enhancements without affecting overall functionality.

Security and privacy are also critical aspects of the system. The system ensures that SMS data is processed securely and sensitive information is protected. It avoids unnecessary storage of user data and follows best practices to maintain confidentiality. Compatibility and accessibility are considered to ensure that the system can operate across multiple devices and platforms, including web and mobile environments.

• The system shall maintain high reliability and stability when processing large volumes of SMS messages.

• The system shall provide high accuracy and fast response time for real-time spam detection.

• The system shall support scalability to accommodate growing datasets and evolving spam patterns without performance degradation.

• The system shall ensure efficient processing with minimal latency.

• The system shall ensure data privacy and secure handling of SMS information.

• The system shall be compatible with multiple platforms such as web and mobile applications.

• The system shall require minimal maintenance and support easy updates and model improvements.

• The system shall ensure robustness against noisy, unstructured, and adversarial SMS inputs.

• The system shall provide consistent performance across different environments and data variations.

3.4. Feasibility Study

The feasibility study evaluates the practicality and viability of the proposed SMS Spam Detection System. It ensures that the system can be successfully developed and implemented within the available resources, time constraints, and technical capabilities. This study helps in identifying potential risks, benefits, and limitations before actual system development.

In addition to evaluating economic, technical, and social feasibility, the study also considers operational feasibility, which ensures that the system can be effectively used in real-world communication environments. Risk assessment is performed to identify challenges such as handling large-scale SMS data, evolving spam patterns, and maintaining model accuracy over time. Strategies are developed to mitigate these risks and ensure smooth implementation.

The feasibility study confirms that the proposed system is practical, cost-effective, and beneficial for communication security applications. It demonstrates that the system can provide significant advantages in improving spam detection accuracy and user safety while maintaining low development and operational costs.

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a general plan and cost estimates. During system analysis, the feasibility study of the proposed system is carried out to ensure that it is not a burden to the organization. For feasibility analysis, a clear

understanding of the major system requirements is essential. Three key considerations involved in the feasibility analysis are:

1. Economic Feasibility
2. Technical Feasibility
3. Social Feasibility

3.4.1 Economic Feasibility

This study is carried out to evaluate the economic impact of the proposed SMS Spam Detection System on the organization. The amount of funds available for the development and deployment of the system is limited, and therefore all expenditures must be justified. The system is designed to operate within a reasonable budget, which is achieved by utilizing mostly open-source tools and technologies. Only minimal costs may be involved for customization or deployment, making the system economically feasible.

Economic feasibility assesses whether the benefits of the system outweigh the costs involved in its development and implementation. The proposed system is highly cost-effective as it uses widely available technologies such as Python and machine learning frameworks, which do not require expensive licensing. Additionally, the system can be developed and deployed using standard computing resources without the need for specialized hardware.

The system also provides significant economic benefits by reducing the risks associated with SMS spam, such as financial fraud and phishing attacks. By improving spam detection accuracy, it helps organizations and users avoid monetary losses and enhances overall communication security. This results in indirect cost savings and improved efficiency.

Maintenance costs are relatively low due to the modular and scalable design of the system. Updates and improvements to the models can be implemented without major financial investment. Furthermore, the system can be easily extended or integrated into existing platforms without incurring high additional costs.

Overall, the proposed system offers a high return on investment by combining low development costs with significant benefits in terms of security, efficiency, and reliability. It is economically sustainable and suitable for both small-scale and large-scale deployment in real-world communication environments.

3.4.2 Technical Feasibility

This study is carried out to evaluate the technical feasibility of the proposed SMS Spam Detection System, focusing on the technical requirements needed for its development and deployment. Any system developed should not place excessive demand on available technical resources, as this would increase complexity and cost for the user or organization. The proposed system is designed with modest technical requirements, ensuring that it can be implemented with minimal changes to existing infrastructure.

Technical feasibility assesses whether the required technologies, tools, and expertise are available to successfully build and operate the system. The proposed system utilizes widely used and well-supported technologies such as Python, machine learning libraries, and natural language processing frameworks. These tools are easily accessible, well-documented, and suitable for developing efficient spam detection models.

The system architecture is designed to be simple, scalable, and efficient, ensuring that it does not require high-end hardware or complex infrastructure. The machine learning and deep learning models are optimized to run on standard computing systems while maintaining high accuracy and performance. Additionally, the availability of pre-trained models and open-source libraries reduces development effort and technical complexity.

Furthermore, the system supports integration with web applications and APIs, making it adaptable to different deployment environments. The modular design allows easy updates, maintenance, and future enhancements without requiring major technical modifications. Overall, the system is technically feasible as it relies on readily available technologies, requires minimal resources, and can be efficiently implemented in real-world environments.

3.4.3 Social Feasibility

This study is carried out to evaluate the social feasibility of the proposed SMS Spam Detection System, focusing on the level of acceptance and usability among its intended users. The system is designed in such a way that users do not feel threatened by its implementation but instead recognize it as a necessary and beneficial tool for improving communication security. The level of user acceptance largely depends on how effectively the system is introduced and how easily users can understand and interact with it. Proper guidance and minimal training help users become familiar with the system, increasing their confidence and willingness to use it.

.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements define the essential tools, frameworks, and development environments needed to design, build, and deploy the SMS spam detection system. These components provide a stable foundation for implementing machine learning models, managing data workflows, and developing the web-based interface for user interaction. The selected technologies ensure compatibility across platforms, support efficient text preprocessing, and enable seamless integration of backend logic with the user-facing modules. The following software requirements form the core of the system's development environment.

- Operating System: Windows 7 Ultimate/ Linux / macOS
- Programming Language: Python 3.x
- Frontend: python
- Backend: Django
- Designing: Html, CSS, javascript

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources needed to support dataset handling, text preprocessing, and model design within the SMS spam detection system. The project can be efficiently developed using standard computing hardware, ensuring compatibility with future expansions such as large-scale model training, system deployment, and real-time message processing. The selected configuration supports smooth execution of the Django-based application, Python scripts, and machine learning workflows.

- **Processor:** Pentium–IV or equivalent
- **Memory (RAM):** Minimum 8 GB
- **Storage:** 512 GB HDD/SSD
- **Keyboard:** Standard Windows Keyboard
- **Mouse:** Two or Three Button Mouse
- **Monitor:** SVGA Display
- **Display Requirement:** Standard 14" or higher

5. SOFTWARE DESIGN

5.1 System Architecture

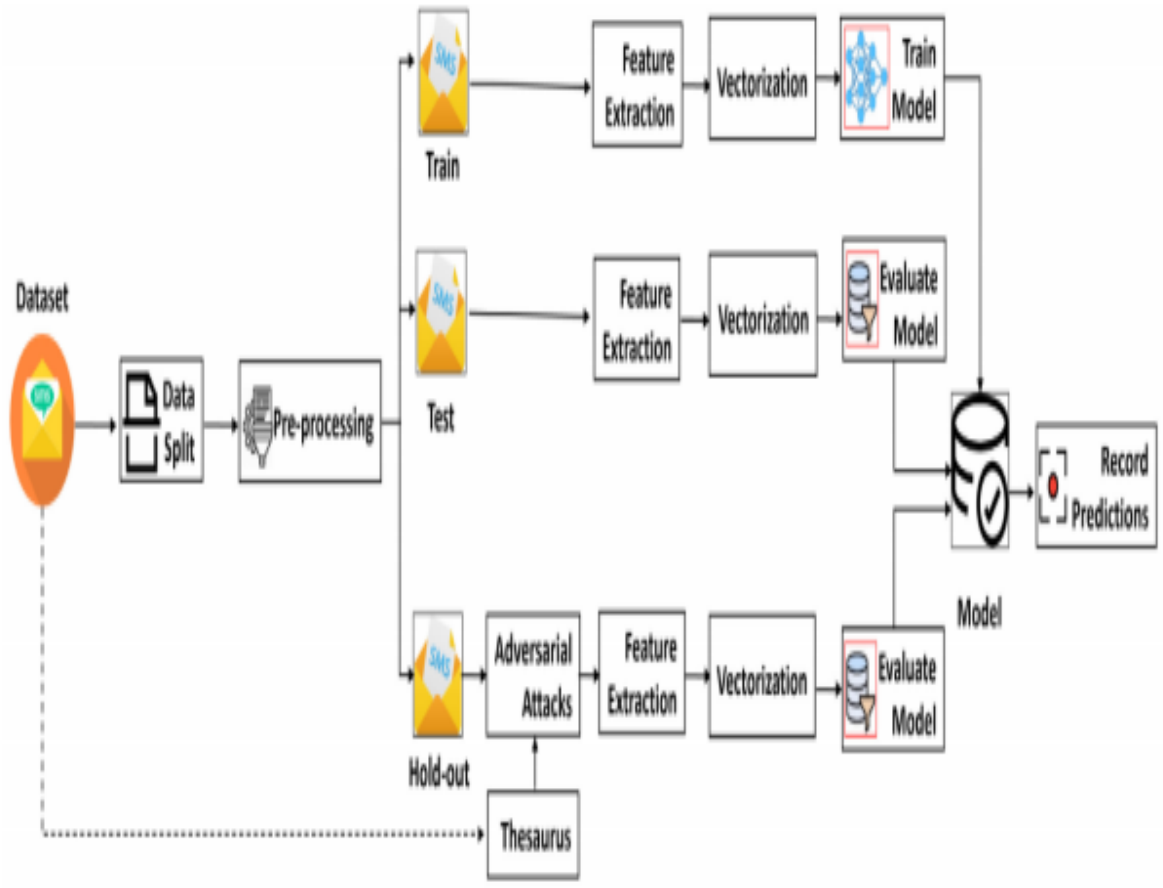

Figure:5.1 System Architecture

The architecture of the SMS Spam Detection System is designed to provide an efficient, scalable, and robust solution for detecting spam messages using machine learning and natural language processing techniques. The system follows a modular approach, integrating data preprocessing, feature extraction, model training, evaluation, and prediction components into a structured workflow. This design ensures flexibility, accuracy, and adaptability for real-world SMS filtering applications.

At the initial stage, the system accepts a dataset of SMS messages, which may include both spam and legitimate (ham) messages. These messages are first passed through a data-splitting mechanism, where the dataset is divided into training, testing, and hold-out sets. This division ensures proper evaluation of the model and helps prevent overfitting. The structured splitting also allows the system to assess performance on unseen data, improving generalization capability.

The next stage involves preprocessing, where raw SMS messages are cleaned and standardized. This module performs operations such as text normalization, removal of special characters, noise reduction, and tokenization. These steps ensure that the input data is consistent and suitable for further processing. Preprocessing plays a crucial role in improving the quality of the data and enhancing the performance of machine learning models.

Following preprocessing, the system applies feature extraction and vectorization techniques to convert textual data into numerical representations. Methods such as Term Frequency–Inverse Document Frequency (TF-IDF), n-grams, and contextual embeddings are used to capture important linguistic and semantic features from SMS messages. These feature vectors serve as input to the machine learning and deep learning models, enabling them to learn patterns and relationships within the data.

The processed data is then passed to the model training module, where various algorithms such as traditional machine learning models and advanced deep learning models (e.g., LSTM, BERT, RoBERTa) are trained. These models learn to classify messages as spam or ham based on extracted features. The testing dataset follows a similar pipeline and is used to evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

An important component of the system architecture is the robustness evaluation module. A separate hold-out dataset is subjected to adversarial or evasive modifications, including synonym replacement, spacing manipulation, and character-level alterations. These modified messages are processed through the same preprocessing and feature extraction pipeline and then evaluated by the trained model. This helps in assessing the system's ability to handle real-world spam evasion techniques and ensures improved reliability.

Finally, the prediction and output module generates classification results, labeling each message as "Spam" or "Ham." The results are displayed to the user or stored for further analysis. The system ensures that predictions are generated efficiently, supporting real-time or near real-time applications. Additionally, all predictions and evaluation results are recorded, enabling performance tracking and future improvements.

Overall, the system architecture effectively integrates preprocessing, feature extraction, model training, and robustness evaluation into a cohesive framework. It ensures efficient data flow, accurate classification, and adaptability to evolving spam patterns, making it a practical and reliable solution for real-world SMS spam detection.

5.2 Dataflow Diagram

The proposed Data Flow Diagram (DFD) represents the flow of data within the SMS Spam Detection System, illustrating how input SMS messages are processed through various stages to generate the final classification result. The diagram highlights the interaction between external entities (users or systems), processing modules, data storage components, and machine learning models used for spam detection.

The process begins with the User or External Source, which provides SMS messages as input to the system. These messages are received by the Input Module (1.0), where the system captures and temporarily stores the raw SMS data. This stage ensures that all incoming messages are properly formatted and ready for further processing. The output of this stage is Raw SMS Data, which is passed to the next level.

In the Data Preprocessing (2.0) stage, the raw SMS messages undergo several cleaning and transformation operations. These include text normalization, removal of special characters, tokenization, and noise reduction. The goal of this stage is to standardize the input data and eliminate irrelevant information. The output of this process is Cleaned and Tokenized Text, which is suitable for feature extraction and model input.

The processed text is then forwarded to the Feature Extraction and Vectorization (3.0) module. In this stage, textual data is converted into numerical representations using techniques such as TF-IDF, n-grams, or contextual embeddings. These numerical vectors capture important syntactic and semantic features of the SMS messages. The result of this stage is Feature Vectors, which serve as input for the classification model.

Next, the feature vectors are passed to the Spam Classification (4.0) module, where trained machine learning or deep learning models are applied. The system retrieves the trained model parameters from the Model Storage (e.g., saved model files) and uses them to classify messages as “Spam” or “Ham.” The model analyzes patterns and relationships within the data and produces Classification Results along with prediction confidence.

An additional component of the system includes the Robustness Evaluation Module, where certain inputs (especially from a hold-out dataset) are modified using adversarial techniques such as synonym replacement, spacing manipulation, and character alterations. These modified messages are processed through the same pipeline to evaluate the system's resistance to evasive spam strategies.

Finally, the output is passed to the Results Generation (5.0) module, where the classification result is formatted into a user-friendly form.

Overall, the DFD illustrates a clear and logical flow of data from input to output, demonstrating how raw SMS messages are transformed into meaningful classification results. The modular structure enhances system clarity, maintainability, and scalability, making the solution suitable for real-time SMS spam detection and analysis in practical environments

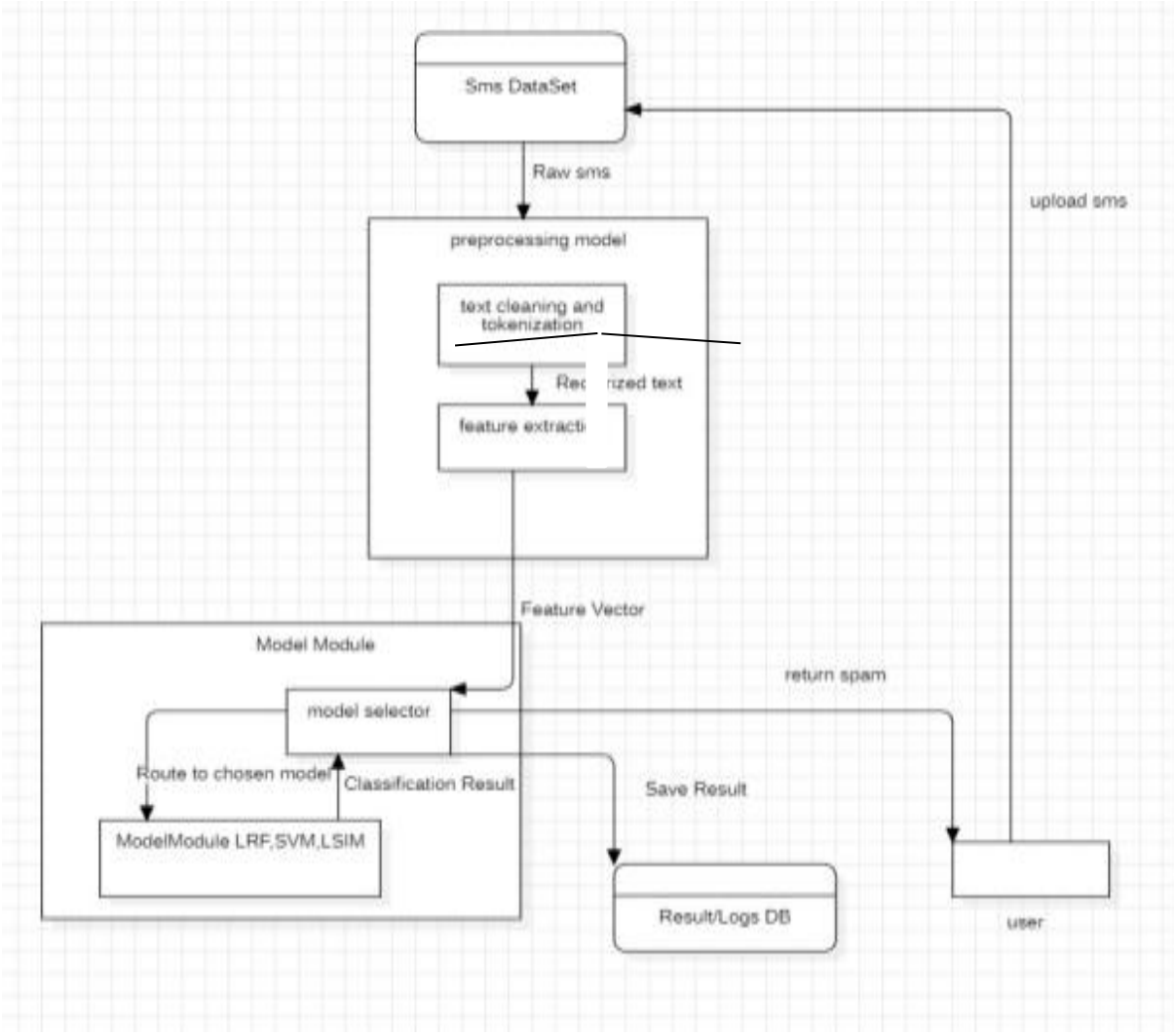

Figure:5.2 Dataflow Diagram

5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized language used for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML helps in representing the design of systems and understanding their components. Created by the Object Management Group (OMG), UML 1.0 was proposed in January 1997. UML is closely associated with object-oriented analysis and design. The two main categories of UML diagrams are Behavioral and Structural diagrams, each serving distinct purposes in the modeling process.

The Behavioral UML diagrams describe the behavior of the system, its actors, and the interaction between the components. On the other hand, Structural UML diagrams depict the static structure of the system, showing its components and relationships. UML has been integrated as a standard by OMG, and its primary goals are to provide a formal basis for understanding modeling languages, offer a ready-to-use expressive language for system developers, and encourage the growth of object-oriented tools.

Goals of UML:

- Provide an expressive visual modeling language for developing and exchanging meaningful models.
- Establish a formal basis for understanding the modeling language.
- Encourage the growth of object-oriented tools.
- Integrate best practices into system development.

Types of UML Diagrams:

1. Sequence Diagram

2. Use Case Diagram

3. Activity Diagram

4. Class Diagram

5.3.1. 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 diagram.

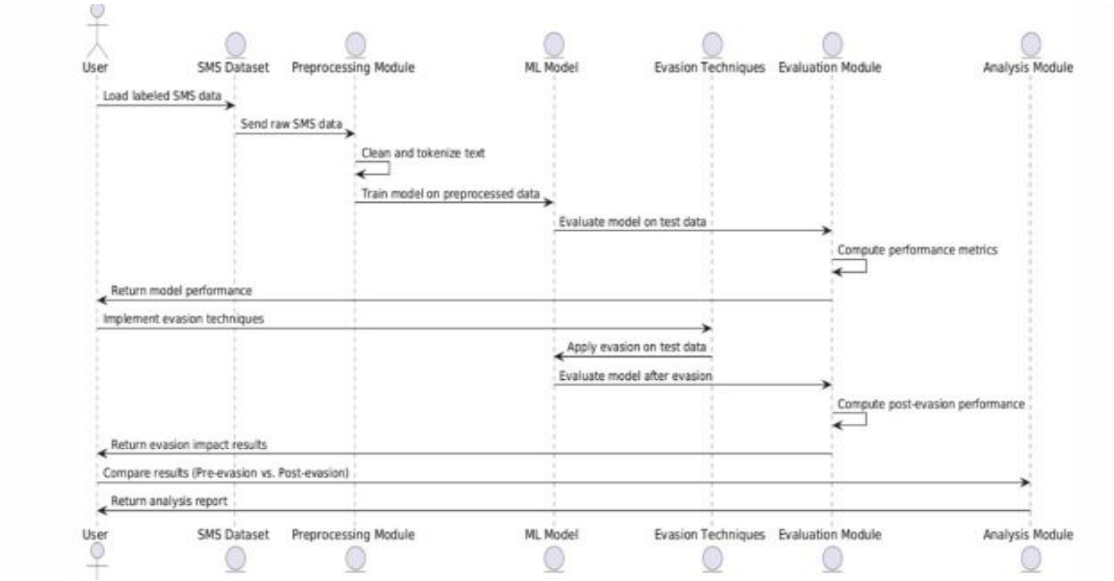

Figure5.3.1 Sequence Diagram

List of actions

- **User:**

  The user interacts with the system by entering an SMS message or uploading a dataset for spam detection. The user initiates the classification request and waits for the system to analyze and return whether the message is spam or ham.

- **System:**

  The system receives the SMS input from the user, validates the text, and forwards it to the preprocessing module. It ensures the message is cleaned, tokenized, and transformed into a suitable format before sending it to the machine learning model for prediction.

- **Model:**

  The model performs preprocessing and converts the SMS into TF-IDF feature vectors. It then applies the Decision Tree and Extra Trees classifiers independently.

- **Evaluation:**

  Computes accuracy and performance metrics, interprets the model's output, returns output.

5.3.2 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.

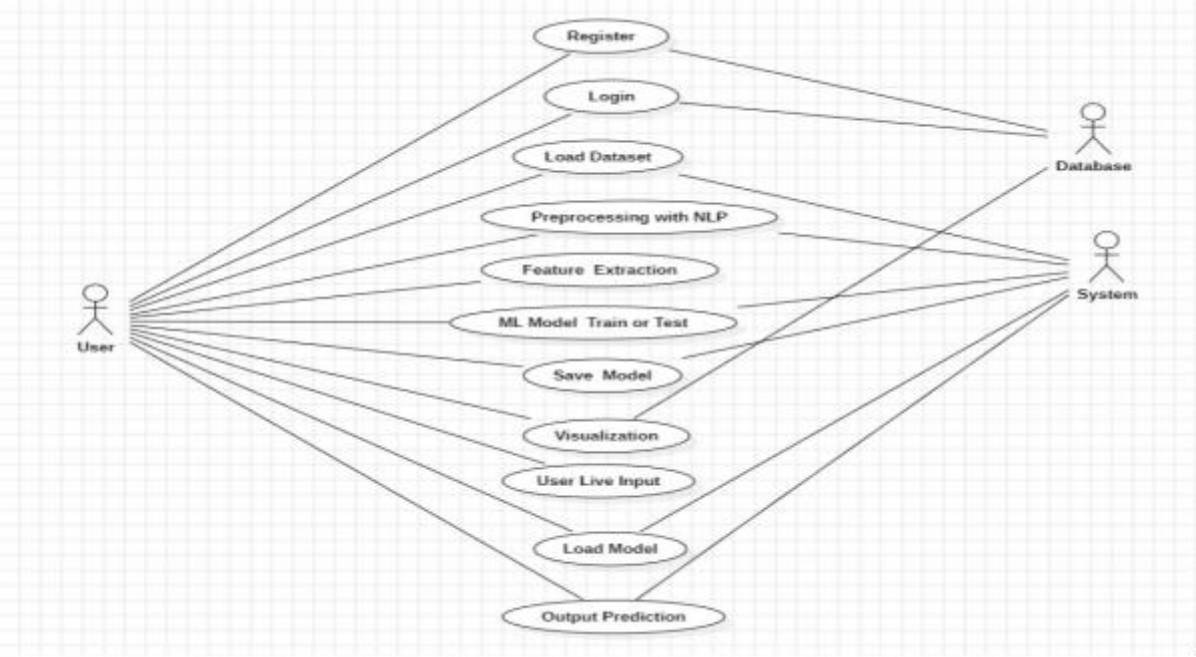

Figure5.3.2 Use Case Diagram

5.3.3 Activity Diagram

Activity diagrams are graphical representations of work flows 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 work flows of components in a system. An activity diagram shows the overall flow of control.

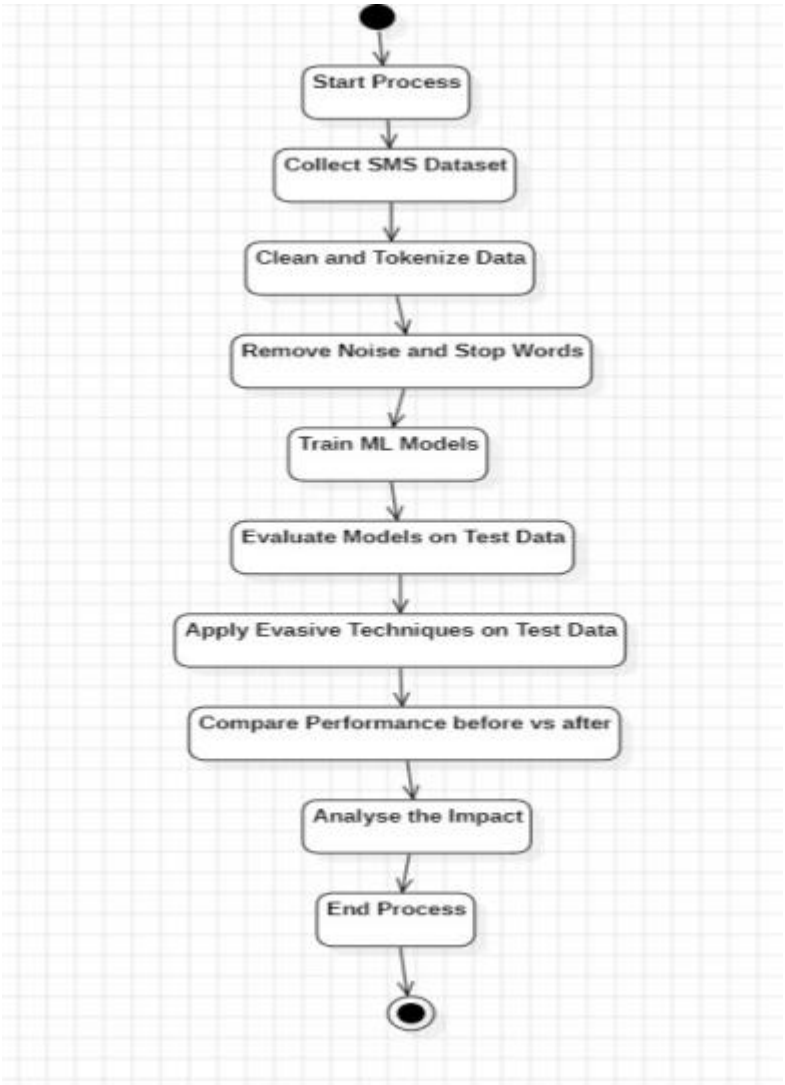

Figure5.3.3 Activity Diagram

5.3.4. 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.

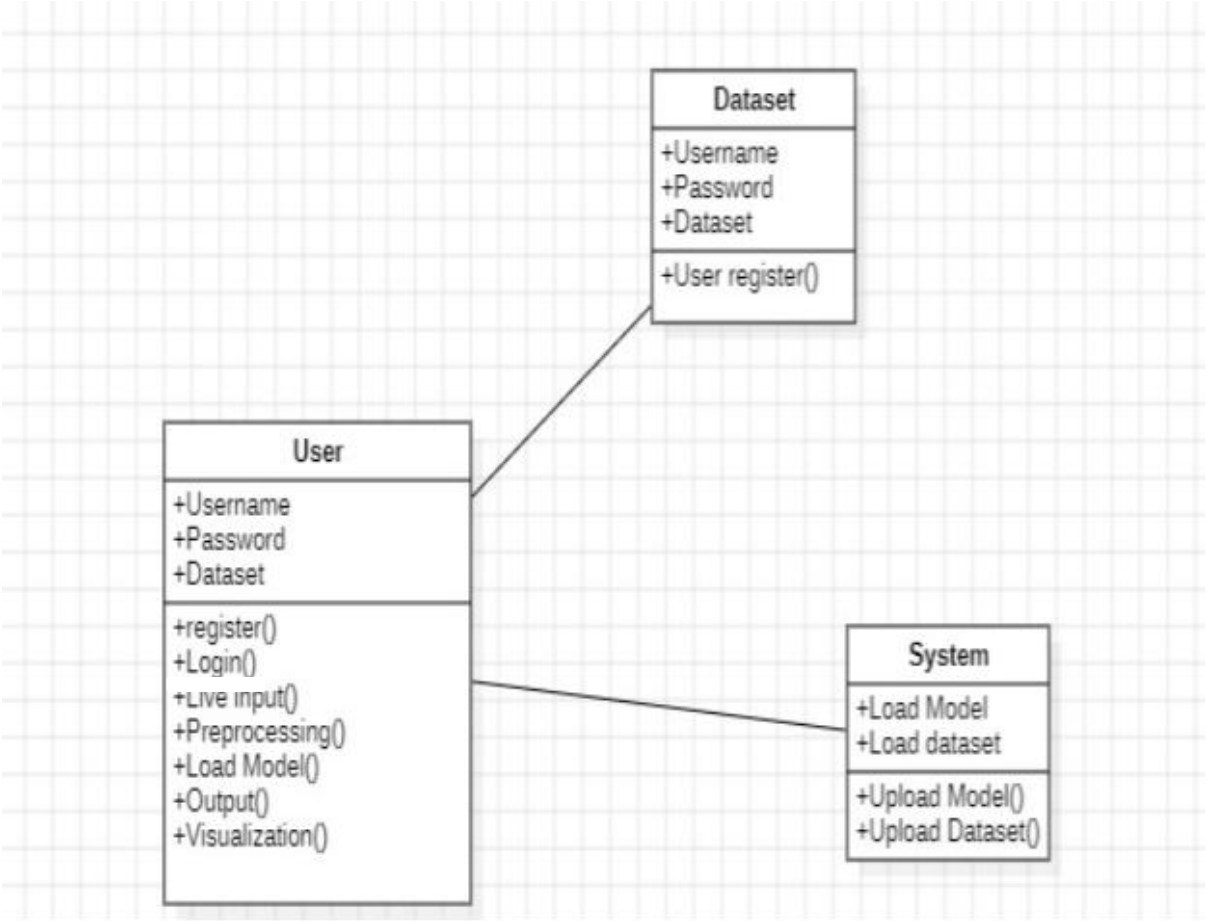

Figure5.3.4 Class Diagram

6. CODING AND IMPLEMENTATION

6.1 Source Code

main.py:

```
import pandas as pd
import numpy as np

# Sklearn
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Deep Learning
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Conv1D, MaxPooling1D,
GlobalMaxPooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Embeddings
from gensim.models import Word2Vec, FastText
import gensim.downloader as api

# Transformers
from transformers import AutoTokenizer, AutoModel
import torch


# 2. Load and Prepare Dataset

data = pd.read_csv("dataset.csv")
data['Message'] = data['Message'].astype(str)

X = data['Message']
y = data['EncodedClass']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)


# 3. Tokenization for DL Models
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
```

```
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(X_train_seq, maxlen=100, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=100, padding='post')

vocab_size = len(tokenizer.word_index) + 1


# 4. Feature Extraction Methods

# ---- Bag of Words ----
bow_vectorizer = CountVectorizer()
X_train_bow = bow_vectorizer.fit_transform(X_train)
X_test_bow = bow_vectorizer.transform(X_test)

# ---- TF-IDF ----
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)


# ---- Word2Vec ----
def tokenize(text):
    return text.split()

sentences = [tokenize(msg) for msg in X_train]

word2vec_model = Word2Vec(
    sentences, vector_size=100, window=5, min_count=1, workers=4
)

def get_avg_embedding(texts, model):
    embeddings = []
    for text in texts:
        tokens = tokenize(text)
        vector = np.mean(
            [model.wv[word] for word in tokens if word in model.wv]
            or [np.zeros(100)],
            axis=0
        )
        embeddings.append(vector)
    return np.array(embeddings)

X_train_w2v = get_avg_embedding(X_train, word2vec_model)
X_test_w2v = get_avg_embedding(X_test, word2vec_model)


# ---- FastText ----
fasttext_model = FastText(
    sentences, vector_size=100, window=5, min_count=1, workers=4
)
```

```
X_train_fasttext = get_avg_embedding(X_train, fasttext_model)
X_test_fasttext = get_avg_embedding(X_test, fasttext_model)


# ---- GloVe ----
glove_vectors = api.load("glove-wiki-gigaword-100")

def get_glove_embeddings(texts, model):
    embeddings = []
    for text in texts:
        tokens = tokenize(text)
        vector = np.mean(
            [model[word] for word in tokens if word in model]
            or [np.zeros(100)],
            axis=0
        )
        embeddings.append(vector)
    return np.array(embeddings)

X_train_glove = get_glove_embeddings(X_train, glove_vectors)
X_test_glove = get_glove_embeddings(X_test, glove_vectors)


# ---- BERT Embeddings ----
bert_tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
bert_model = AutoModel.from_pretrained("bert-base-uncased")

def get_bert_embeddings(texts):
    embeddings = []
    for text in texts:
        inputs = bert_tokenizer(
            text,
            return_tensors='pt',
            truncation=True,
            padding=True,
            max_length=512
        )
        with torch.no_grad():
            outputs = bert_model(**inputs)
        cls_vector = outputs.last_hidden_state[:, 0, :].squeeze().numpy()
        embeddings.append(cls_vector)
    return np.array(embeddings)

X_train_bert = get_bert_embeddings(X_train)
X_test_bert = get_bert_embeddings(X_test)



# 5. Model Training Function
def train_and_evaluate(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
```

```
    print(classification_report(y_test, predictions))


# 6. Traditional ML Models

# ---- SVM with TF-IDF ----
print("SVM with TF-IDF")
svm_model = SVC()
train_and_evaluate(svm_model, X_train_tfidf, X_test_tfidf, y_train, y_test)

# ---- Random Forest with BoW ----
print("Random Forest with Bag of Words")
rf_model = RandomForestClassifier()
train_and_evaluate(rf_model, X_train_bow, X_test_bow, y_train, y_test)


# 7. Deep Learning Models

# ---- LSTM Model ----
print("LSTM Model")

lstm_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    LSTM(128, return_sequences=True),
    LSTM(64),
    Dense(1, activation='sigmoid')
])

lstm_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

lstm_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
)

loss, accuracy = lstm_model.evaluate(X_test_pad, y_test)
print(f"LSTM Test Accuracy: {accuracy}")


# ---- CNN Model ----
print("CNN Model")

cnn_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    Conv1D(filters=128, kernel_size=5, activation='relu'),
    MaxPooling1D(pool_size=2),
    GlobalMaxPooling1D(),
```

```
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
 ])

 cnn_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
 )

 cnn_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
 )

 loss, accuracy = cnn_model.evaluate(X_test_pad, y_test)
 print(f"CNN Test Accuracy: {accuracy}")

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import (
   Embedding, LSTM, Dense,
   Conv1D, MaxPooling1D,
   GlobalMaxPooling1D, Dropout
)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


# Load dataset
data = pd.read_csv("dataset.csv")
data['Message'] = data['Message'].astype(str)

X = data['Message']
y = data['EncodedClass']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42, stratify=y
)

# Tokenization and padding
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

```
X_train_pad = pad_sequences(X_train_seq, maxlen=100, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=100, padding='post')

vocab_size = len(tokenizer.word_index) + 1


# Build hybrid LSTM + CNN model
hybrid_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    LSTM(128, return_sequences=True),
    Dropout(0.2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

hybrid_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Train model
hybrid_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
)

# Evaluate model
loss, accuracy = hybrid_model.evaluate(X_test_pad, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

# Save model
model_path = "hybrid_lstm_cnn_model.keras"
hybrid_model.save(model_path)

# Prediction function
def predict_message(text, model, tokenizer):
    seq = tokenizer.texts_to_sequences([text])
    pad = pad_sequences(seq, maxlen=100, padding='post')
    prediction = model.predict(pad)
    return "Spam" if prediction[0][0] > 0.5 else "Ham"

# Example usage
loaded_model = load_model(model_path)
sample_text = "Free entry in a weekly competition. Text now to win prizes!"
print(predict_message(sample_text, loaded_model, tokenizer))
```

```
# ---- LSTM Model ----
print("LSTM Model")

lstm_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    LSTM(128, return_sequences=True),
    LSTM(64),
    Dense(1, activation='sigmoid')
])

lstm_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

lstm_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
)

loss, accuracy = lstm_model.evaluate(X_test_pad, y_test)
print(f"LSTM Test Accuracy: {accuracy}")


# ---- CNN Model ----
print("CNN Model")

cnn_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    Conv1D(filters=128, kernel_size=5, activation='relu'),
    MaxPooling1D(pool_size=2),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

cnn_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

cnn_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
)
```

```
  loss, accuracy = cnn_model.evaluate(X_test_pad, y_test)
  print(f"CNN Test Accuracy: {accuracy}")

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import (
    Embedding, LSTM, Dense,
    Conv1D, MaxPooling1D,
    GlobalMaxPooling1D, Dropout
)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


# Load dataset
data = pd.read_csv("dataset.csv")
data['Message'] = data['Message'].astype(str)

X = data['Message']
y = data['EncodedClass']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Tokenization and padding
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(X_train_seq, maxlen=100, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=100, padding='post')

vocab_size = len(tokenizer.word_index) + 1


# Build hybrid LSTM + CNN model
hybrid_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=100),
    LSTM(128, return_sequences=True),
    Dropout(0.2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
```

```
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

hybrid_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Train model
hybrid_model.fit(
    X_train_pad, y_train,
    validation_data=(X_test_pad, y_test),
    epochs=5,
    batch_size=32
)

# Evaluate model
loss, accuracy = hybrid_model.evaluate(X_test_pad, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

# Save model
model_path = "hybrid_lstm_cnn_model.keras"
hybrid_model.save(model_path)

# Prediction function
def predict_message(text, model, tokenizer):
    seq = tokenizer.texts_to_sequences([text])
    pad = pad_sequences(seq, maxlen=100, padding='post')
    prediction = model.predict(pad)
    return "Spam" if prediction[0][0] > 0.5 else "Ham"

# Example usage
loaded_model = load_model(model_path)
sample_text = "Free entry in a weekly competition. Text now to win prizes!"
print(predict_message(sample_text, loaded_model, tokenizer))
```

Backend:

Views.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import load_model
from django.shortcuts import render
from .models import Prediction
from django.utils.timezone import now

# Create your views here.
def userhome(request):
    user = request.user
    return render(request, 'User/userhome.html', {'user':user})

def userpredict(request):
    if request.method == 'POST':
        # Load Dataset
        data = pd.read_csv("model/dataset.csv")
        data['Message'] = data['Message'].astype(str)

        # Split the dataset into training and testing sets
        X = data['Message']
        y = data['EncodedClass']
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.2, random_state=42, stratify=y
        )

        # Tokenize and pad sequences for deep learning models
        tokenizer = Tokenizer()
        tokenizer.fit_on_texts(X_train)
        X_train_seq = tokenizer.texts_to_sequences(X_train)
        X_test_seq = tokenizer.texts_to_sequences(X_test)
        X_train_pad = pad_sequences(X_train_seq, maxlen=100, padding='post')
        X_test_pad = pad_sequences(X_test_seq, maxlen=100, padding='post')
        vocab_size = len(tokenizer.word_index) + 1

        # Get user input
        user_input = request.POST.get('user_input', '')

        # Tokenize and pad the user input
        input_seq = tokenizer.texts_to_sequences([user_input])
        input_pad = pad_sequences(input_seq, maxlen=100, padding='post')

        # Load the pre-trained model
        hybrid_model = load_model("model/hybrid_lstm_cnn_model.h5")

        # Predict using the model
        prediction = hybrid_model.predict(input_pad)
```

```
        result = "Spam" if prediction[0] > 0.5 else "Ham"

        # Save user input and result to the database
        Prediction.objects.create(user_input=user_input, result=result, created_at=now())

        # Pass prediction result to template
        return render(request, 'User/userpredict.html', {'result': result, 'input': user_input})

    return render(request, 'User/userpredict.html')
```

Base.html:

```
<!DOCTYPE html>
{% load static%}
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Investigating Evasive Techniques in SMS Spam Filtering A Comparative Analysis of Machine Learning Models</title>
  <meta name="description" content="">
  <meta name="keywords" content="">

  <!-- Fonts -->
  <link href="https://fonts.googleapis.com" rel="preconnect">
  <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Raleway:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

  <!-- Main CSS File -->
  <link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">

  <header id="header" class="header d-flex align-items-center light-background sticky-top">
    <div class="container-fluid position-relative d-flex align-items-center justify-content-between">

      <a  class="logo d-flex align-items-center me-auto me-xl-0">
        <!-- Uncomment the line below if you also wish to use an image logo -->
        <!-- <img src="assets/img/logo.png" alt=""> -->
        <h1 class="sitename">Investigating Evasive Techniques in SMS Spam Filtering </h1>
      </a>

      <nav id="navmenu" class="navmenu">
        <ul>
          <li><a href="{% url 'home_page' %}">Home</a></li>
          <li><a href="{% url 'login  page' %}">Login</a></li>
        </ul>
        <i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
      </nav>

      <div class="header-social-links">
      </div>

    </div>
```

```
  </header>

  <main class="main">

    {%block contents%}

    {%endblock%}

  </main>


  <!-- Scroll Top -->
  <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Preloader -->
  <div id="preloader"></div>

  <!-- Vendor JS Files -->
  <script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
  <script src="{% static 'vendor/aos/aos.js' %}"></script>
  <script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
  <script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
  <script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
  <script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
  <script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
  <script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

  <!-- Main JS File -->
  <script src="{% static 'js/main.js' %}"></script>

</body>

</html>
```

Main.js

```
(function() {
 "use strict";

 /**
  * Apply .scrolled class to the body as the page is scrolled down
  */
 function toggleScrolled() {
   const selectBody = document.querySelector('body');
   const selectHeader = document.querySelector('#header');
   if (!selectHeader.classList.contains('scroll-up-sticky') &&
!selectHeader.classList.contains('sticky-top') && !selectHeader.classList.contains('fixed-top'))
return;
   window.scrollY > 100 ? selectBody.classList.add('scrolled') :
selectBody.classList.remove('scrolled');
 }

 document.addEventListener('scroll', toggleScrolled);
 window.addEventListener('load', toggleScrolled);

 /**
  * Mobile nav toggle
  */
 const mobileNavToggleBtn = document.querySelector('.mobile-nav-toggle');

 function mobileNavToogle() {
   document.querySelector('body').classList.toggle('mobile-nav-active');
   mobileNavToggleBtn.classList.toggle('bi-list');
   mobileNavToggleBtn.classList.toggle('bi-x');
 }
 mobileNavToggleBtn.addEventListener('click', mobileNavToogle);

 /**
  * Hide mobile nav on same-page/hash links
  */
 document.querySelectorAll('#navmenu a').forEach(navmenu => {
   navmenu.addEventListener('click', () => {
     if (document.querySelector('.mobile-nav-active')) {
       mobileNavToogle();
     }
   });

 });

 /**
  * Toggle mobile nav dropdowns
  */
 document.querySelectorAll('.navmenu .toggle-dropdown').forEach(navmenu => {
   navmenu.addEventListener('click', function(e) {
     e.preventDefault();
     this.parentNode.classList.toggle('active');
```

```
      this.parentNode.nextElementSibling.classList.toggle('dropdown-active');
      e.stopImmediatePropagation();
    });
  });

  /**
   * Preloader
   */
  const preloader = document.querySelector('#preloader');
  if (preloader) {
    window.addEventListener('load', () => {
      preloader.remove();
    });
  }

  /**
   * Scroll top button
   */
  let scrollTop = document.querySelector('.scroll-top');

  function toggleScrollTop() {
    if (scrollTop) {
      window.scrollY > 100 ? scrollTop.classList.add('active') :
scrollTop.classList.remove('active');
    }
  }
  scrollTop.addEventListener('click', (e) => {
    e.preventDefault();
    window.scrollTo({
      top: 0,
      behavior: 'smooth'
    });
  });

  window.addEventListener('load', toggleScrollTop);
  document.addEventListener('scroll', toggleScrollTop);

  /**
   * Animation on scroll function and init
   */
  function aosInit() {
    AOS.init({
      duration: 600,
      easing: 'ease-in-out',
      once: true,
      mirror: false
    });
  }
  window.addEventListener('load', aosInit);

  /**
   * Animate the skills items on reveal
   */
```

```
let skillsAnimation = document.querySelectorAll('.skills-animation');
skillsAnimation.forEach((item) => {
  new Waypoint({
    element: item,
    offset: '80%',
    handler: function(direction) {
      let progress = item.querySelectorAll('.progress .progress-bar');
      progress.forEach(el => {
        el.style.width = el.getAttribute('aria-valuenow') + '%';
      });
    }
  });
});

/**
 * Initiate Pure Counter
 */
new PureCounter();

/**
 * Init swiper sliders
 */
function initSwiper() {
  document.querySelectorAll(".init-swiper").forEach(function(swiperElement) {
    let config = JSON.parse(
      swiperElement.querySelector(".swiper-config").innerHTML.trim()
    );

    if (swiperElement.classList.contains("swiper-tab")) {
      initSwiperWithCustomPagination(swiperElement, config);
    } else {
      new Swiper(swiperElement, config);
    }
  });
}

window.addEventListener("load", initSwiper);

/**
 * Initiate glightbox
 */
const glightbox = GLightbox({
  selector: '.glightbox'
});

/**
 * Init isotope layout and filters
 */
document.querySelectorAll('.isotope-layout').forEach(function(isotopeItem) {
  let layout = isotopeItem.getAttribute('data-layout') ?? 'masonry';
  let filter = isotopeItem.getAttribute('data-default-filter') ?? '*';
  let sort = isotopeItem.getAttribute('data-sort') ?? 'original-order';
```

```
    let initIsotope;
    imagesLoaded(isotopeItem.querySelector('.isotope-container'), function() {
      initIsotope = new Isotope(isotopeItem.querySelector('.isotope-container'), {
        itemSelector: '.isotope-item',
        layoutMode: layout,
        filter: filter,
        sortBy: sort
      });
    });

    isotopeItem.querySelectorAll('.isotope-filters li').forEach(function(filters) {
      filters.addEventListener('click', function() {
        isotopeItem.querySelector('.isotope-filters .filter-active').classList.remove('filter-active');
        this.classList.add('filter-active');
        initIsotope.arrange({
          filter: this.getAttribute('data-filter')
        });
        if (typeof aosInit === 'function') {
          aosInit();
        }
      }, false);
    });

  });

})();
```

6.2 Implementation

The implementation of the SMS Spam Detection System is carried out using a combination of machine learning techniques, natural language processing (NLP), and a web-based deployment framework. The system is designed to process raw SMS data, perform preprocessing, extract meaningful features, and classify messages as spam or ham using trained machine learning models.

The overall implementation is divided into multiple stages, including data acquisition, preprocessing, feature extraction, model training, evaluation, and deployment. Each stage is carefully designed to ensure efficiency, accuracy, and scalability of the system.

The project follows a modular implementation approach, ensuring scalability, maintainability, and real-time usability. Each module performs a specific task such as data cleaning, feature engineering, prediction, and result visualization. This modular design allows easy debugging, testing, and future enhancements.

The backend of the system is developed using the Django framework, which handles user requests, processes input data, and returns predictions. The machine learning models are implemented using Python-based libraries and are integrated into the web application for real-time prediction.

The system is capable of handling both batch processing (for training) and real-time input (for prediction). It is designed to efficiently process large volumes of SMS data while maintaining low response time for user queries.

Additionally, the implementation ensures proper data handling and validation to avoid errors during prediction. Input messages are validated, cleaned, and transformed into a structured format before being passed to the model. This improves the reliability and robustness of the system.

The system also supports easy model updates. New data can be used to retrain the model, allowing it to adapt to evolving spam patterns and improve performance over time. This makes the system suitable for real-world deployment where spam techniques continuously change.

6.2.1. Technologies Used:

Python is a high-level, interpreted, and object-oriented programming language widely used for machine learning, data analysis, and web development. It provides simple syntax, extensive libraries, and strong community support, making it an ideal choice for developing intelligent systems.

Python follows a readable and concise coding structure, which reduces development time and improves

code maintainability. It supports multiple programming paradigms such as procedural, object-oriented, and functional programming.

In this project, Python is used for:

- Data preprocessing and cleaning of SMS messages
- Feature extraction using techniques like TF-IDF
- Model training and evaluation using machine learning algorithms
- Implementation of ensemble learning techniques
- Backend development using the Django framework
- Integration of machine learning models with the web application

Python also enables seamless integration between different components of the system, such as data processing modules and web interfaces, ensuring smooth workflow execution.

Advantages of Python in this Project

- Easy integration with machine learning libraries like Scikit-learn, TensorFlow, and Keras
- Rich support for Natural Language Processing (NLP) using NLTK and Regex
- Efficient handling and processing of large datasets using NumPy and Pandas
- Simple and readable syntax, making code easy to understand and maintain
- Seamless integration with web frameworks like Django for deployment
- Faster development and prototyping compared to other programming languages
- Cross-platform compatibility (works on Windows, Linux, macOS)
- Strong community support with extensive documentation and resources
- Availability of a large number of built-in and external libraries
- Supports modular programming, improving scalability and flexibility
- Reduces development time and overall project complexity
- Suitable for both machine learning and web application development

6.2.2 Machine Learning

Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into

training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result.

The training data is the biggest (in -size) subset of the original dataset, which is used to train or fit the machine learning model. Firstly, the training data is fed to the ML algorithms, which lets them learn how to make predictions for the given task.

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. The test dataset is another subset of original data, which is independent of the training dataset. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project.

6.2.3 Categories of Machine Learning

Machine Learning is broadly classified into the following categories:

1. Supervised Learning
    - Uses labeled datasets for training
    - The model learns input-output relationships
    - Common algorithms: Decision Tree, Random Forest, SVM
    - Used in this project for SMS spam classification

2. Unsupervised Learning
    - Works with unlabeled data
    - Identifies hidden patterns and structures
    - Common techniques: Clustering, Association Rules
    - Not directly used in this project but useful for data analysis

3. Reinforcement Learning
    - Learns through interaction with environment

- Uses rewards and penalties for decision-making
- Commonly used in robotics and gaming

This project mainly focuses on Supervised Learning, as it provides high accuracy for classification problems where labeled data is available.

6.2.4 Need for Machine Learning

Traditional rule-based systems rely on predefined keywords and patterns, which makes them ineffective against modern spam techniques. Spammers continuously evolve their methods using obfuscation, abbreviations, and text manipulation to bypass detection.
Machine Learning addresses these limitations by providing intelligent and adaptive solutions.
Key Reasons for Using Machine Learning

- Learns patterns from large volumes of SMS data
- Adapts to new and evolving spam techniques
- Improves accuracy and efficiency over time
- Reduces dependency on manual rule creation
- Handles complex and unstructured text data
- Detects hidden patterns not visible to humans
- Supports real-time spam detection systems
- Enhances scalability for large-scale applications

Thus, machine learning is essential for building a robust and intelligent SMS spam detection system.

6.2.5 Challenges in Machine Learning

- Handling noisy and unstructured SMS data (slang, abbreviations, symbols)
- Class imbalance problem (more ham messages than spam)
- Detecting obfuscated spam messages (e.g., “fr33”, “w1n n0w”)
- Risk of overfitting on training data
- Difficulty in selecting relevant features from text data
- High dimensionality due to large number of text features
- Concept drift (spam patterns changing over time)
- Maintaining real-time prediction performance
- Data privacy and security concerns
- Limited availability of high-quality labeled datasets
- Model interpretability issues (hard to explain predictions)
- Computational complexity and resource requirements

6.3 Project Methodology

The project methodology defines the systematic and structured approach followed to design, develop, and implement the SMS Spam Detection System. It outlines the sequence of processes involved in transforming raw SMS data into meaningful predictions using machine learning techniques. The methodology ensures that each stage of the system is well-organized, efficient, and contributes to the overall performance of the model.

The implementation is divided into multiple stages, including data collection, preprocessing, feature extraction, model training, evaluation, and deployment. These stages are interconnected, where the output of one stage serves as the input for the next, forming a complete data processing pipeline.
The methodology begins with collecting a large and diverse dataset of SMS messages to ensure proper learning of spam patterns. The collected data is then preprocessed to remove noise, inconsistencies, and irrelevant information. This step is essential for improving the quality of input data and enhancing the performance of the machine learning model.

After preprocessing, feature extraction techniques such as TF-IDF are applied to convert textual data into numerical representations. This transformation enables machine learning algorithms to understand and process text data effectively. The extracted features capture important information such as word frequency and significance, which helps in distinguishing spam from legitimate messages.

The next stage involves training machine learning models using the processed data. In this project, an ensemble learning approach is used to improve prediction accuracy and model stability. Multiple algorithms are combined to leverage their individual strengths and produce more reliable results.

Once the model is trained, it is evaluated using standard performance metrics such as accuracy, precision, recall, and F1-score. This evaluation helps in measuring the effectiveness of the model and identifying areas for improvement. Model optimization techniques such as hyperparameter tuning may also be applied to enhance performance.

The final stage of the methodology involves deploying the trained model into a web-based application using the Django framework. This allows users to interact with the system in real time by entering SMS messages and receiving instant predictions.

6.3.1 Data Collection and Pre-processing

The dataset used in this project consists of SMS messages labeled as either **spam** or **ham (legitimate)**. The dataset contains diverse message formats, including informal language, abbreviations, special characters, and URLs, which require careful pre-processing.

Pre-processing Steps

- Lowercasing text to maintain consistency
- Removing punctuation and special characters
- Removing numerical values and irrelevant symbols
- Tokenization (splitting text into individual words)
- Stop-word removal (removing common words like “is”, “the”, “and”)
- Lemmatization (reducing words to their root form)
- Removing extra spaces and duplicate entries
- Handling missing or null values

These preprocessing steps help in cleaning the raw SMS data and converting it into a structured format suitable for machine learning models. Proper preprocessing improves model performance and reduces noise in the dataset.

6.3.2 Feature Extraction

Feature extraction is a crucial step that transforms textual data into numerical form so that machine learning algorithms can process it.

In this project, TF-IDF (Term Frequency–Inverse Document Frequency) is used to extract features from SMS messages.

TF-IDF assigns weights to words based on their importance in a message relative to the entire dataset. Words that appear frequently in a message but rarely across all messages receive higher importance.

Advantages of TF-IDF

- Reduces dimensionality of text data
- Highlights important and meaningful words
- Eliminates less useful common words
- Improves classification performance
- Converts text into machine-readable numerical vectors

- Helps in distinguishing spam and ham messages effectively

6.3.3 Model Training

The model training phase involves feeding preprocessed and feature-extracted data into machine learning algorithms to learn patterns for spam classification.
The system uses an ensemble learning approach, which combines multiple models to improve performance.

Algorithms Used

- Decision Tree Classifier
- Extra Trees Classifier

Each model is trained independently using the training dataset. The final prediction is obtained using a hard voting mechanism, where the majority prediction from both models determines the final output.

Benefits of Ensemble Learning

- Improved prediction accuracy
- Reduced overfitting compared to single models
- Better generalization on unseen data
- Increased robustness against noise and variations
- More stable and reliable predictions

6.3.4 Model Evaluation

After training, the model is evaluated using standard performance metrics to measure its effectiveness.

Evaluation Metrics

- Accuracy: Measures overall correctness of predictions
- Precision: Measures how many predicted spam messages are actually spam
- Recall: Measures how many actual spam messages are correctly detected
- F1-Score: Harmonic mean of precision and recall

The model achieves high accuracy (approximately 95–98%) in detecting spam messages, demonstrating its effectiveness.
Additional evaluation techniques such as confusion matrix analysis can also be used to understand

classification performance in detail.

6.3.5 Web Application Development

The system is deployed as a web application using the Django framework, which provides a robust backend for handling user interactions and model predictions.

From the implementation:

- Manage.py is used to initialize and run the Django project

- User-friendly interface to input SMS messages
- Backend processing of input data
- Loading of trained machine learning model
- Real-time spam/ham prediction
- Display of classification results to the user
- Efficient handling of multiple user requests

The web application ensures accessibility and allows users to interact with the system in real time.

6.3.6 System Workflow

The system follows a structured workflow for SMS classification:

- User enters an SMS message through the web interface
- The input text is preprocessed (cleaning, tokenization, etc.)
- Features are extracted using TF-IDF vectorization
- The processed data is passed to the trained machine learning model
- The model predicts whether the message is spam or ham
- The final result is displayed to the user

Additional Workflow Features

- Input validation to ensure proper message format
- Fast processing for real-time prediction
- Scalable architecture to handle large user requests.

7. SYSTEM TESTING

System testing is a crucial phase that ensures the developed SMS Spam Detection System functions accurately, consistently, and reliably under real-world operating conditions. The primary objective of system testing is to identify potential defects, validate system behavior, and confirm that both functional and non-functional requirements are satisfied. In addition to verifying correctness, system testing also evaluates the system's robustness, scalability, and performance when subjected to real user inputs and evolving spam patterns. It serves as the final validation step before deployment, ensuring that all integrated components operate seamlessly without failure.

In this project, system testing focuses on validating all major components, including input handling, preprocessing modules, feature extraction mechanisms, machine learning and deep learning models, and output generation. Testing ensures that SMS messages are correctly received, cleaned, transformed into numerical representations, and accurately classified as spam or legitimate messages. Special attention is given to the correctness of model predictions, consistency of results across different inputs, and the stability of the system during continuous usage. The system is also tested for its ability to handle unexpected or invalid inputs such as empty messages, noisy text, and adversarial manipulations, ensuring that such cases are managed effectively without impacting overall performance.

System testing was conducted using diverse SMS datasets containing both spam and legitimate messages to evaluate correctness, robustness, and usability. The system was tested under various real-world conditions, including handling of informal language, abbreviations, multilingual content, and obfuscated spam patterns. Special emphasis was placed on classification accuracy, which achieved high performance (approximately 95–98%), demonstrating the effectiveness of the implemented models.

Additionally, performance metrics such as response time, processing efficiency, and system stability under repeated prediction requests were carefully analyzed. These evaluations confirm that the system is capable of delivering reliable, consistent, and scalable performance in real-world SMS spam detection applications.

7.1 Types of System Testing

7.1.1 Unit testing

Unit testing was performed to validate individual components of the SMS Spam Detection System independently. Each module, including preprocessing functions, feature extraction methods, and classification components, was tested using controlled inputs to verify expected outputs. This type of testing ensures that each unit of the system performs its intended function correctly without relying on other modules. By isolating components, potential issues can be identified and resolved at an early stage, reducing complexity during system integration and improving overall code quality.

Key focus areas included:

- text preprocessing operations such as tokenization, normalization, and noise removal
- generation and validation of feature vectors using techniques like TF-IDF and embeddings
- correctness of data transformation and input formatting for model compatibility
- internal logic of machine learning and deep learning models for accurate classification
- validation of input handling, including empty messages and improper formats
- verification of data flow through preprocessing and feature extraction pipelines
- testing of individual utility functions used in text processing and feature generation
- error handling mechanisms for invalid, noisy, or incomplete SMS inputs
- consistency and stability of numerical computations during feature extraction

Unit testing ensured that each logical unit functioned correctly in isolation, enabling early detection and correction of errors before full system integration. It also helped validate edge cases such as empty inputs, noisy text, and unexpected message formats. This significantly improved the reliability, maintainability, and overall quality of the system, ensuring that all components operate as expected in the complete workflow.

7.1.2 Integration testing

Integration testing was performed to evaluate whether the combined components of the SMS spam detection system function correctly as a unified workflow. This phase focuses on verifying that individually tested modules interact properly when integrated, ensuring smooth data flow and consistent system behavior. It is primarily concerned with validating the interaction between different system components and confirming that they operate together without errors.

Modules tested in combination included:

- input data processing with preprocessing and feature extraction components
- integration of feature extraction with machine learning classification models
- interaction between different modules involved in prediction and result generation
- data flow across components to ensure correct transformation and output

This testing stage ensured that all integrated components work cohesively and produce accurate results when executed together. Integration testing helped identify issues arising from component interaction and confirmed that the system operates reliably as a complete and consistent SMS spam detection solution.

7.1.3 Functional test

Functional testing validated that all features of the SMS Spam Detection System operate according to specified requirements and user expectations. This phase focuses on verifying that each functionality performs correctly when used by an end user. Realistic test scenarios were designed to simulate actual usage conditions, ensuring that the system behaves accurately and consistently under different inputs and situations.

Major validation rules included:

- valid SMS messages are processed correctly and classified accurately
- invalid, empty, or improperly formatted inputs are rejected gracefully
- preprocessing functions correctly clean and transform input text
- feature extraction methods generate accurate numerical representations
- classification models produce correct spam or ham predictions
- output results are clearly displayed in a user-friendly format
- system provides meaningful error messages for incorrect inputs or failures
- prediction workflow operates smoothly without interruptions
- results remain consistent for similar or repeated inputs

7.1.4 Systems Testing:

System testing evaluated the SMS Spam Detection System as a complete and fully integrated application. This phase ensures that all modules work together as expected and that the system meets performance, reliability, and functional requirements. It involves testing the entire system under real-world conditions to verify its behavior, consistency, and efficiency. System testing is based on process flows, integration points, and end-to-end execution of all components involved in SMS spam detection.

This phase focuses on validating how different modules—such as input handling, preprocessing, feature extraction, classification models, and output generation—interact within the system. It ensures that data flows correctly across all stages and that the system produces accurate and predictable results. Special attention is given to testing business logic, predefined processes, and successive operations to confirm that the system behaves as intended in all scenarios.

Tests verified:

• coordinated execution of preprocessing, feature extraction, and classification modules
• correct response to multiple and repeated SMS prediction requests
• system stability during continuous processing of large volumes of messages
• consistent classification results across different types of spam and ham messages
• system performance under high load and real-time usage conditions
• response time consistency for different input sizes and formats
• proper handling of noisy, unstructured, and adversarial SMS inputs
• smooth data flow across all integrated components without errors
• system recovery and stability in case of unexpected failures

The testing demonstrated that the system produces predictable and correct results in accordance with defined requirements. It also confirmed that the application performs efficiently and reliably across different usage scenarios. Overall, system testing ensured that the SMS spam detection system is robust, scalable, and ready for real-world deployment.

7.1.5 White Box Testing

White-box testing was applied to the internal components of the SMS Spam Detection System, focusing on preprocessing functions, feature extraction mechanisms, and classification model logic. This testing approach involves examining the internal structure, code logic, and data flow within the system to ensure correctness, efficiency, and proper implementation. It requires knowledge of the system's internal design and is used to test components that are not visible at the user level.

Additional checks included:

• verification of internal data flow between preprocessing, feature extraction, and classification modules
• validation of text transformation processes such as tokenization, normalization, and vectorization
• inspection of feature generation methods including TF-IDF and embedding representations
• ensuring correct handling of numerical computations and scaling operations
• testing internal function logic, decision branches, and control flow

• verification of model loading, parameter usage, and prediction execution
• checking consistency of intermediate outputs during different processing stages

7.1.6 Black Box Testing

Black-box testing evaluated the SMS Spam Detection System from the user's perspective without considering internal implementation details. In this approach, the system is treated as a "black box," where inputs are provided and outputs are observed to verify correct behavior. The testing is based on system requirements and specifications, ensuring that the application responds appropriately to different types of SMS inputs.

SMS messages were provided as input through the system interface, and the classification outputs (Spam or Ham) were analyzed to validate visible system behavior. This method focuses on functionality, usability, and correctness without examining internal code or logic.

Additional validations included:
• User experience during input of SMS messages and viewing prediction results
• Correct display and formatting of classification outputs (Spam/Ham)
• System behavior with different types of SMS data, including noisy and informal text
• Error handling for invalid, empty, or improperly formatted inputs
• Consistency of outputs for repeated or similar SMS messages
• System usability for both technical and non-technical users
• Response time and performance from a user interaction perspective

Black-box testing ensured that the system meets user expectations and behaves correctly under various real-world scenarios. It confirmed that the application provides a smooth, reliable, and intuitive experience, making it suitable for practical SMS spam detection usage.

7.1.7 Acceptance Testing

Acceptance testing ensured that the SMS Spam Detection System met all specified project requirements and user expectations. This phase involved evaluating the system from the end-user perspective to confirm that it performs accurately, efficiently, and reliably in real-world scenarios. Stakeholders reviewed key aspects such as prediction accuracy, ease of use, response time, and clarity of output results.

The system was tested with different types of SMS inputs, including normal messages, spam content, and edge-case scenarios, to verify that it correctly classifies messages as spam or legitimate. Special

attention was given to usability and workflow simplicity to ensure that users can interact with the system without requiring technical expertise.

Additional evaluation criteria included:
• User satisfaction with classification accuracy and prediction results
• Ease of use and simplicity of system navigation
• System responsiveness and speed during SMS processing
• Reliability and consistency of predictions in practical scenarios
• Overall usability and effectiveness of the system in real-world applications

Feedback confirmed that the system is user-friendly, efficient, and capable of handling real-world SMS spam detection tasks effectively. The workflow was found to be simple, reliable, and suitable for deployment in communication systems.

Test Result Summary:
All defined test cases were executed successfully, and the system produced accurate and consistent results. No major defects were identified during testing. The system met all acceptance criteria and was approved for deployment.

7.2 Testing Strategies

A structured testing strategy was followed throughout the project lifecycle. Testing progressed systematically from component-level validation to full-system verification.

7.2.1 Test strategy and approach

Testing was conducted using a combination of manual testing and dataset-driven validation to ensure the effectiveness and reliability of the SMS Spam Detection System. Real-world SMS datasets, along with benchmark datasets, were used to verify classification accuracy, consistency, and robustness of the system. This approach helped evaluate the system under both controlled conditions and practical scenarios.

Primary strategic objectives included:
• Validating correctness of text preprocessing and data cleaning operations
• Ensuring accurate classification of SMS messages using machine learning and deep learning models
• Verifying reliable interaction between preprocessing, feature extraction, and classification modules
• Confirming stable system performance under repeated and continuous usage
• Evaluating system behavior under real-world SMS conditions, including informal and noisy text
• Testing robustness against adversarial and evasive spam techniques

• Ensuring compatibility with different SMS formats and input variations
• Assessing scalability for handling large datasets and evolving spam patterns

Field testing was conducted by simulating real-world user interactions with SMS inputs, while controlled test cases were used to verify logical correctness and system functionality. This comprehensive testing strategy ensured that the system is reliable, efficient, and capable of handling real-time SMS spam detection in practical environments.

7.2.2 Test objectives

The following objectives guided all testing activities of the SMS Spam Detection System:
• All user inputs (SMS messages) should be processed correctly and efficiently
• System responses should be timely, accurate, and consistent for all inputs
• Invalid, empty, or improperly formatted inputs must be handled safely with appropriate error messages
• Classification results should align with expected spam and ham patterns
• System workflow and interface interaction should be smooth and intuitive
• Prediction accuracy should remain consistent across different datasets and input variations
• Response time should be minimal to support real-time detection
• System should handle multiple users and concurrent requests efficiently without performance degradation
These objectives ensured that the system delivers reliable performance, accurate predictions, and a user-friendly experience in real-world SMS spam detection scenarios.

7.2.3 Features to be tested

The major system features examined included:
• Verification that all SMS inputs are accepted in the correct format and processed properly
• Validation of input handling to ensure no duplicate or invalid entries affect system performance
• Correct routing of data through preprocessing, feature extraction, and classification modules
• Accurate classification of SMS messages into spam or legitimate categories
• Proper display of prediction results in a clear and user-friendly format
• Consistency of model behavior across different inputs and datasets
• Effective error handling mechanisms for invalid or unexpected inputs
• System response time and performance during message processing
• Smooth navigation and correct functioning of all system links and workflows

These tests ensured that all core features of the system function correctly, providing accurate results, reliable performance, and a seamless user experience.

7.2.4 Integration Testing

Integration testing emphasized early detection of dependency conflicts, interface issues, and data mismatches between different components of the SMS Spam Detection System. This phase focused on verifying that multiple integrated modules interact correctly and exchange data without errors. It ensures that when individual components are combined, the system functions smoothly as a unified application.

Testing confirmed correct:

- compatibility between preprocessing outputs and feature extraction inputs
- proper transformation of feature vectors for machine learning model consumption
- accurate mapping of model outputs to spam or ham labels
- seamless data flow between preprocessing, feature extraction, and classification modules
- consistency of data across training, testing, and inference stages
- smooth execution of the complete prediction pipeline without interruptions
- correct interaction between different software components and system layers

Integration testing ensured that all components interact without errors and that the system produces reliable results when executed as a whole. It helped identify interface defects and improved coordination between modules, ensuring stable and efficient operation of the SMS spam detection system in real-world environments.

7.2.5 Acceptance Testing

A prediction system instance was accepted only when it satisfied the following conditions:

- accurate and consistent classification of SMS messages as spam or legitimate
- stable system behavior during continuous and repeated message processing
- error-free input handling and prediction workflow
- clear and meaningful presentation of classification results
- compliance with all specified functional and system requirements
- fast and responsive performance for real-time SMS detection

User Acceptance Testing (UAT) is a critical phase of the project that requires active participation from end users. It ensures that the system meets all functional requirements and performs effectively in real-world scenarios. This phase validates that the system is user-friendly, reliable, and capable of delivering accurate spam detection results according to user expectations.

7.2.6 Overall Test Results

All planned test cases were executed successfully. The system demonstrated stable performance, logical correctness, and consistent SMS spam detection accuracy. The comparative evaluation showed that different machine learning models produced reliable results, with some models performing better under specific conditions. The system also maintained robustness when tested against noisy and evasive SMS inputs.

7.3 Sample Test Cases

Table 7.3 Test Cases

S.No	**Test Case**	**Expected Result**	**Result**	**Remarks** (if any)
1.	User Login	User is authenticated and redirected to dashboard	pass	Error shown for invalid credentials
2	User Registration	New user account is created successfully	pass	Prevents duplicate entries
3	SMS Spam Prediction (Spam)	System classifies message as Spam	pass	Tested with different spam inputs
4	SMS Spam Prediction (Ham)	System classifies message as Ham	pass	Ensures correct normal message detection
5	Empty Input Handling	System shows error for empty input	pass	Avoids invalid processing
6	Model Integration	ML model processes input and returns prediction	pass	Ensures proper backend functioning

Test Case 1:

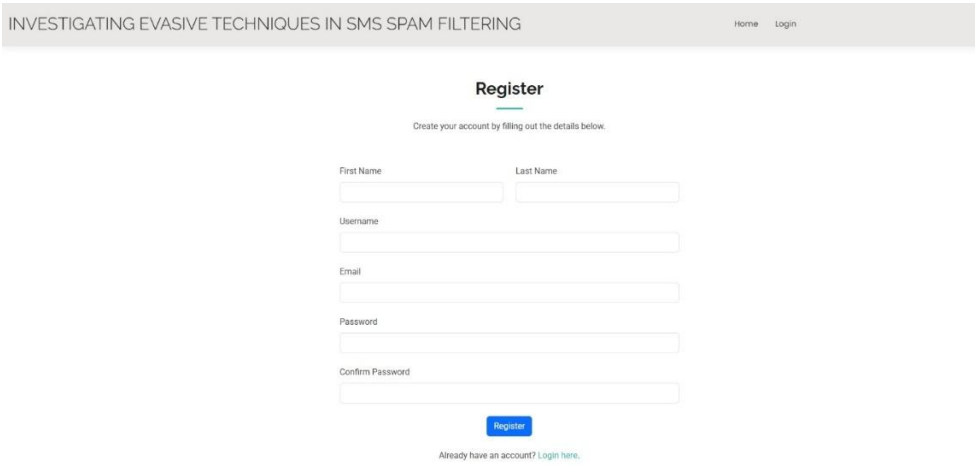

Fig. 7.3.1 User Login

Description:

The user enters valid login credentials and submits the form. The system authenticates the credentials and redirects the user to the dashboard, displaying a personalized welcome message and complete account details such as username, email, status, and last login time. This seamless flow, illustrated in Fig. 7.3.1, confirms that both authentication and user-profile retrieval are functioning correctly. Additionally, the system ensures secure session management by maintaining user login state throughout the interaction.

Test Case 2:

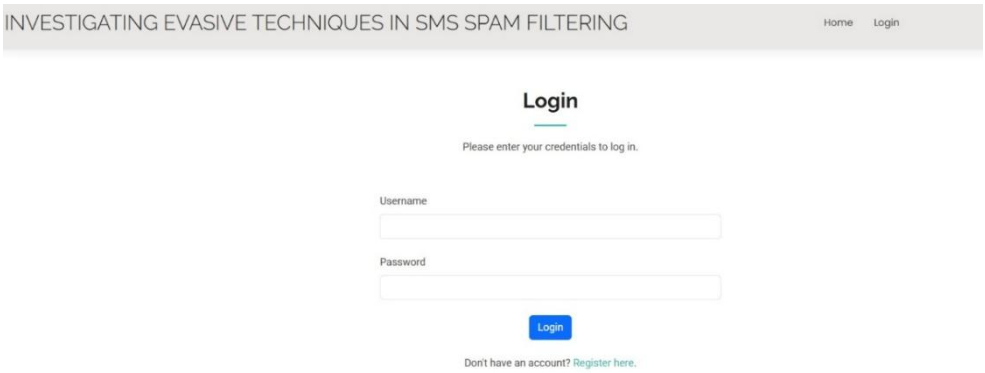

Fig. 7.3.2 User Registration

Description:

After submitting valid registration details, the system successfully creates the user account and displays a confirmation message indicating that registration is complete. The message also informs the user that their account will remain inactive until approved by the administrator. This confirms that the registration workflow and approval control mechanism are functioning correctly, as illustrated in Fig. 7.3.2. Additionally, the system securely stores user information in the database and performs validation checks to prevent duplicate or invalid entries. It also ensures that only authorized users can access the system after administrative approval, enhancing security and access control.

Test Case 3:

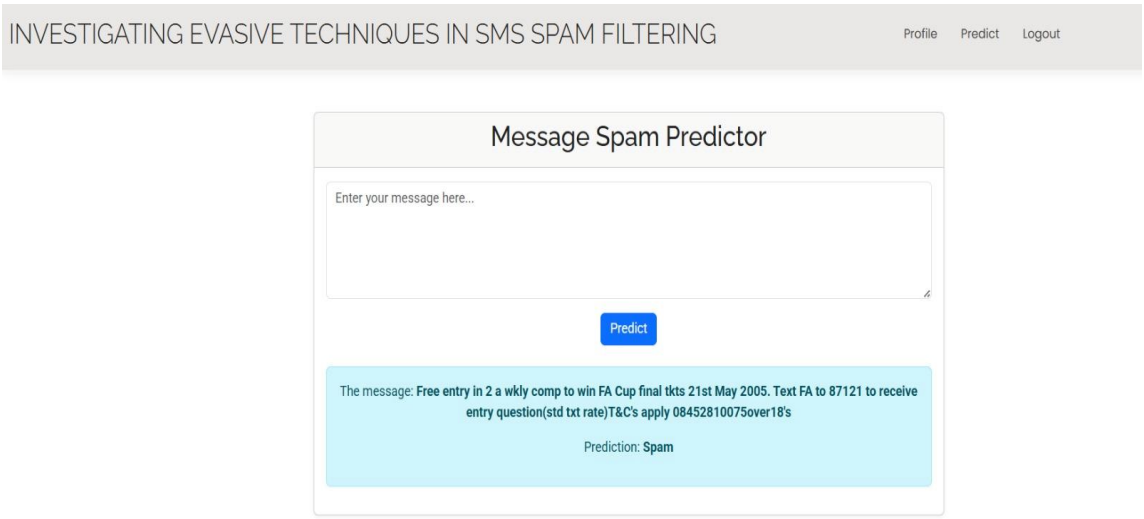

Fig. 7.3.3 Message Spam Predictor (spam)

Description:

The user enters an SMS message into the prediction field and clicks the “Predict” button. The system processes the input using the machine learning model and displays the prediction result, indicating whether the message is spam or not. The output clearly shows the analyzed message along with its classification, as illustrated in Fig. 7.3.3. This confirms that the spam detection functionality is working correctly and providing accurate results.

Additionally, the system ensures quick response time by efficiently processing the input through the trained model. It also enhances user experience by presenting the results in a clear and readable format for easy understanding.

Test Case 4:

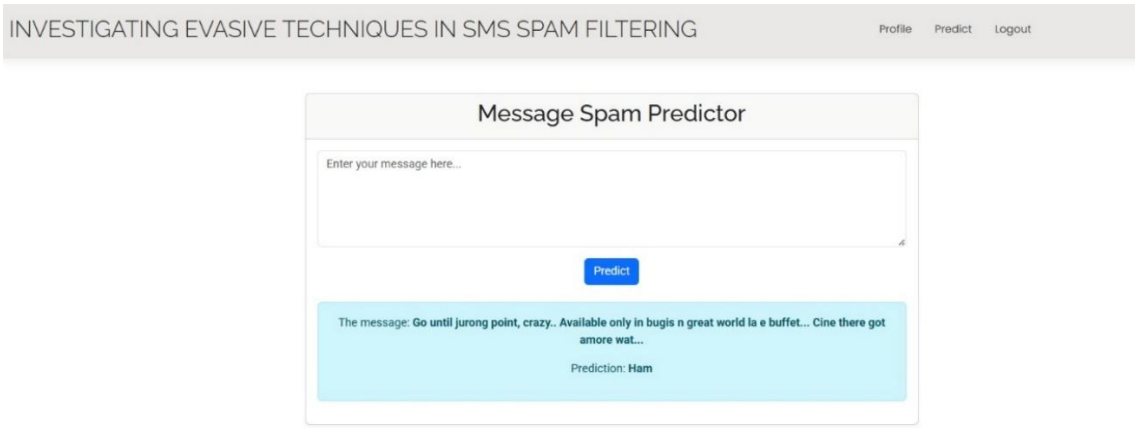

Fig. 7.3.4 Message Spam Predictor (ham)

Description:

The user enters a normal SMS message into the prediction field and clicks the “Predict” button. The system processes the input using the machine learning model and displays the prediction result as “Ham,” indicating that the message is not spam. The output shows the entered message along with its classification, as illustrated in Fig. 7.3.4. This confirms that the model can correctly identify legitimate messages.

Additionally, the system maintains consistency in classification by accurately distinguishing between spam and non-spam messages. It also ensures reliable performance by handling different types of user inputs effectively, improving overall system accuracy.

Test Case 5:

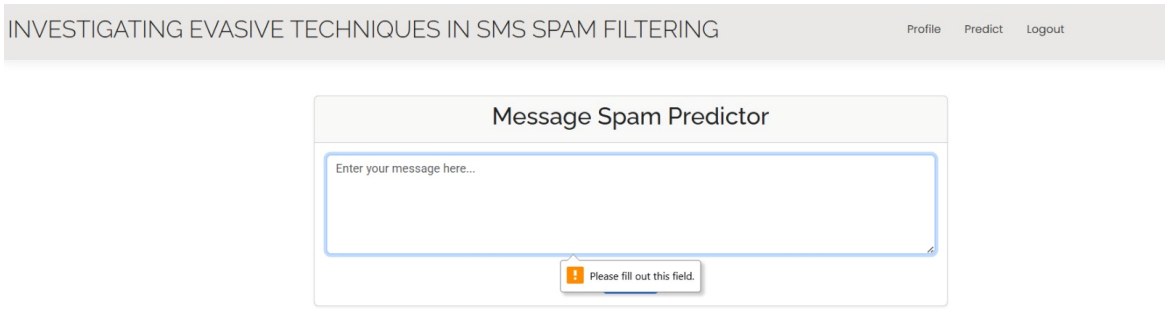

Fig. 7.3.5 Empty Input Handling

Description:

The user attempts to submit the prediction form without entering any message in the input field. The system validates the input and displays an error message prompting the user to fill out the required field. This behavior, as illustrated in Fig. 7.3.5, confirms that proper input validation is implemented to prevent empty submissions. Additionally, the system enhances user guidance by clearly indicating missing input fields before processing the request. It also improves data quality and reliability by ensuring that only valid and complete inputs are sent to the machine learning model for prediction. Furthermore, this validation mechanism reduces the chances of system errors caused by invalid inputs. It contributes to a smoother user experience by providing immediate feedback. Overall, it strengthens the robustness and usability of the application.

Test Case 6:

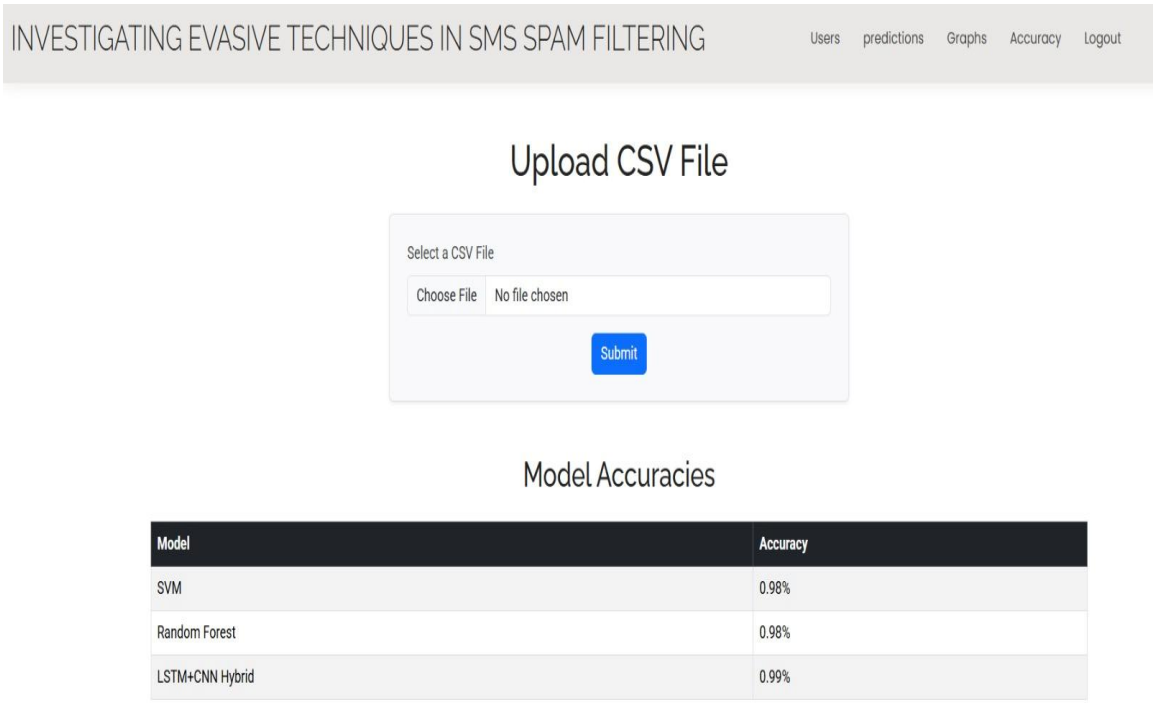

Fig. 7.3.6 Model Integration

Description:

The interface presents a web-based system for investigating evasive techniques in SMS spam filtering. At the top, a navigation bar provides access to sections such as users, predictions, graphs, and accuracy. The main section features a file upload panel where users can select and submit a CSV file for analysis. Below this, the page displays a table summarizing model performance results. The listed models, including SVM, Random Forest, and an LSTM+CNN hybrid, show high accuracy levels close to 99%, indicating strong classification performance. Furthermore, the interface is designed to be user-friendly, allowing easy interaction with the system. It also enables quick comparison of different models, helping users evaluate their effectiveness efficiently.

8. EXPERIMENTAL RESULTS

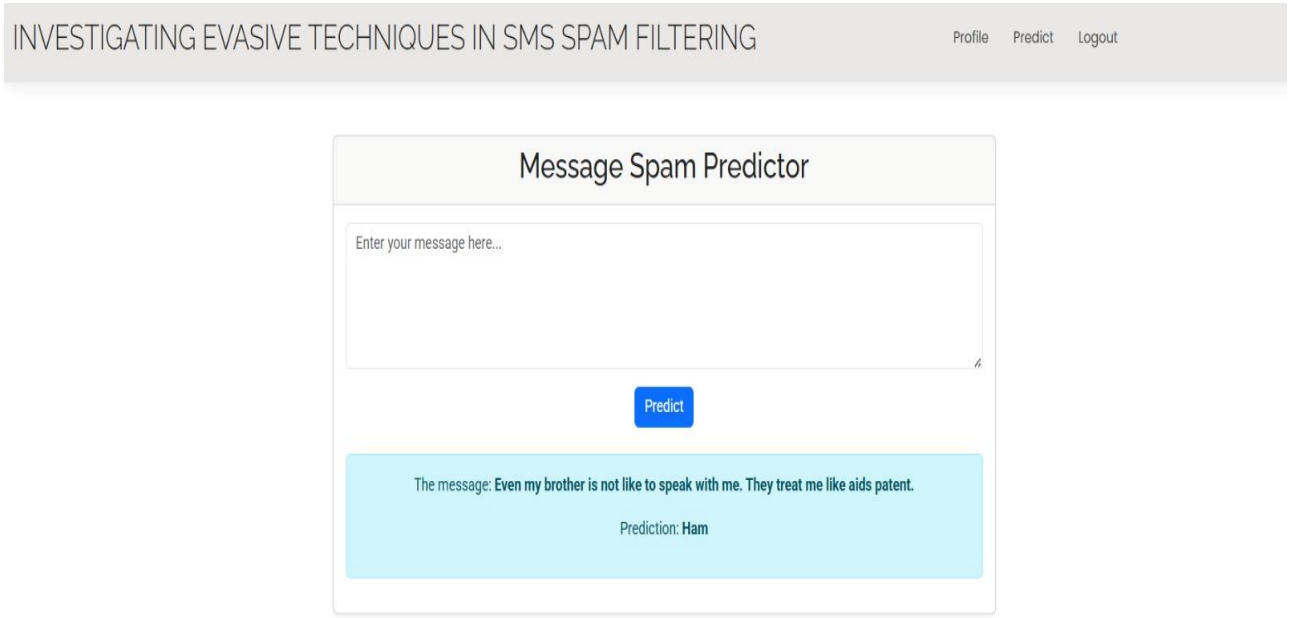

Fig. 8.1 Message Spam Prediction Interface

Description:

The figure illustrates a message spam prediction interface within a social media forensics system. It provides a text input area where users can enter a message to be analyzed for spam detection. A “Predict” button is available to process the input using the underlying classification model. Once submitted, the system displays the entered message along with its predicted category. In this example, the message is classified as “Ham,” indicating it is a legitimate, non-spam message. Additionally, the interface offers a clear and structured layout for ease of use. It provides instant feedback to the user after submission, improving interaction efficiency. Furthermore, the system helps users quickly differentiate between genuine and suspicious messages

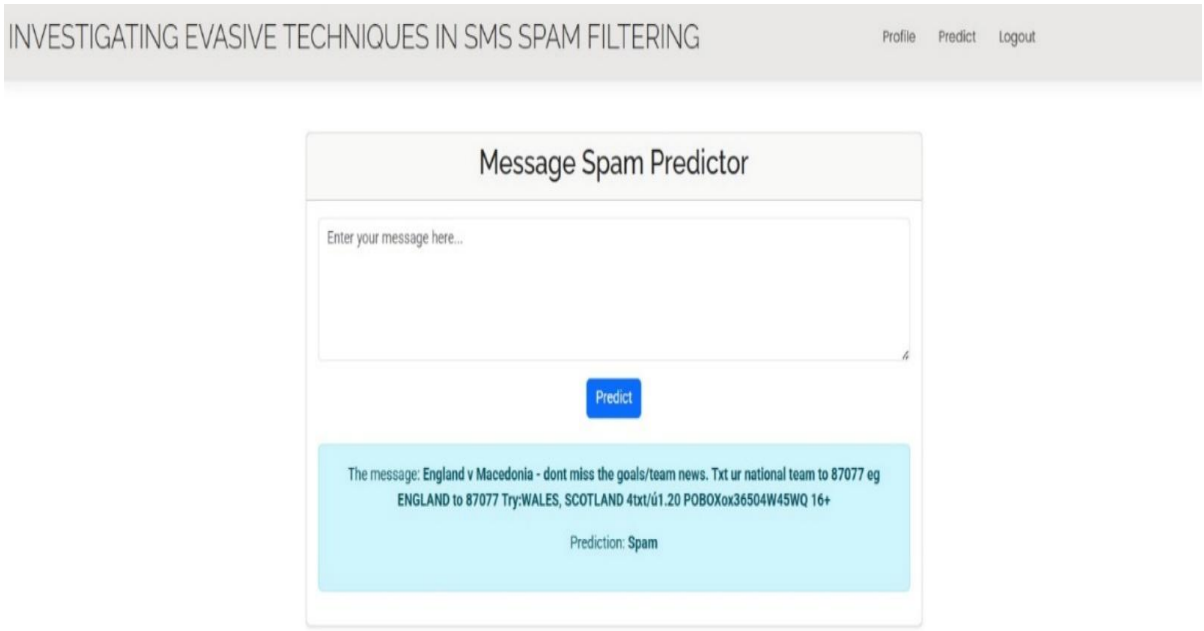

Fig. 8.2 Spam Message Detection Result Interface

Description:

The user clicks the "Predict" button without entering any message in the input field. The system immediately triggers validation and displays a warning message "Please fill out this field," preventing further processing. This ensures that the prediction request is not submitted with empty input, as illustrated in Fig. 8.2. This confirms that proper input validation is functioning correctly in the system. It also improves user experience by guiding users to provide the required information before proceeding. Additionally, this validation mechanism helps maintain the accuracy of the prediction model by avoiding invalid inputs. It reduces unnecessary system processing and resource usage. The feature also minimizes the chances of runtime errors during prediction. Furthermore, it enhances the overall reliability and robustness of the application. Overall, it ensures that the system operates smoothly with valid and meaningful user input.

.

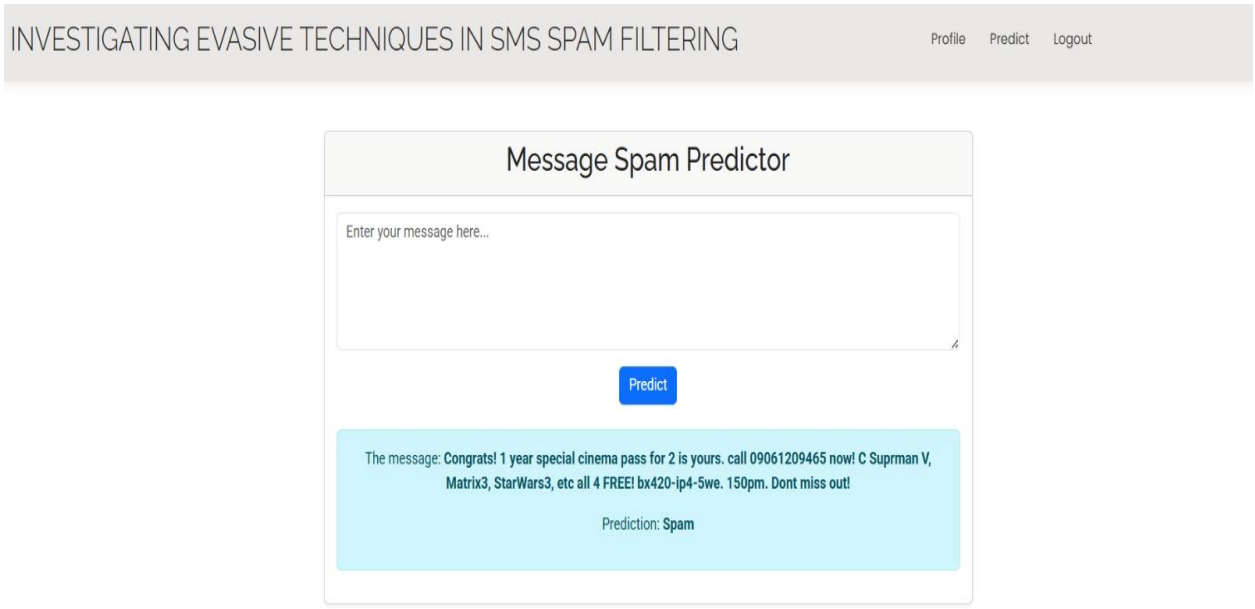

Fig. 8.3 Spam Detection for Promotional Message

Description:

The figure shows the message spam prediction interface used to analyze a promotional text message. Users can input a message into the provided text box and click the “Predict” button to classify it. The system processes the content and displays the message along with its classification result. In this example, a promotional cinema offer message is analyzed and correctly identified as “Spam.” This demonstrates the system’s ability to detect advertising and unsolicited messages effectively. Additionally, the interface provides immediate feedback, enhancing user interaction and understanding. It ensures that suspicious promotional content is flagged accurately for further review. The system also helps in reducing the risk of users being misled by spam messages. Overall, it strengthens the reliability and effectiveness of the spam detection mechanism. Moreover, the system can handle different types of promotional content with consistency. It improves user awareness by clearly labeling harmful messages. This contributes to building a safer and more secure communication environment.

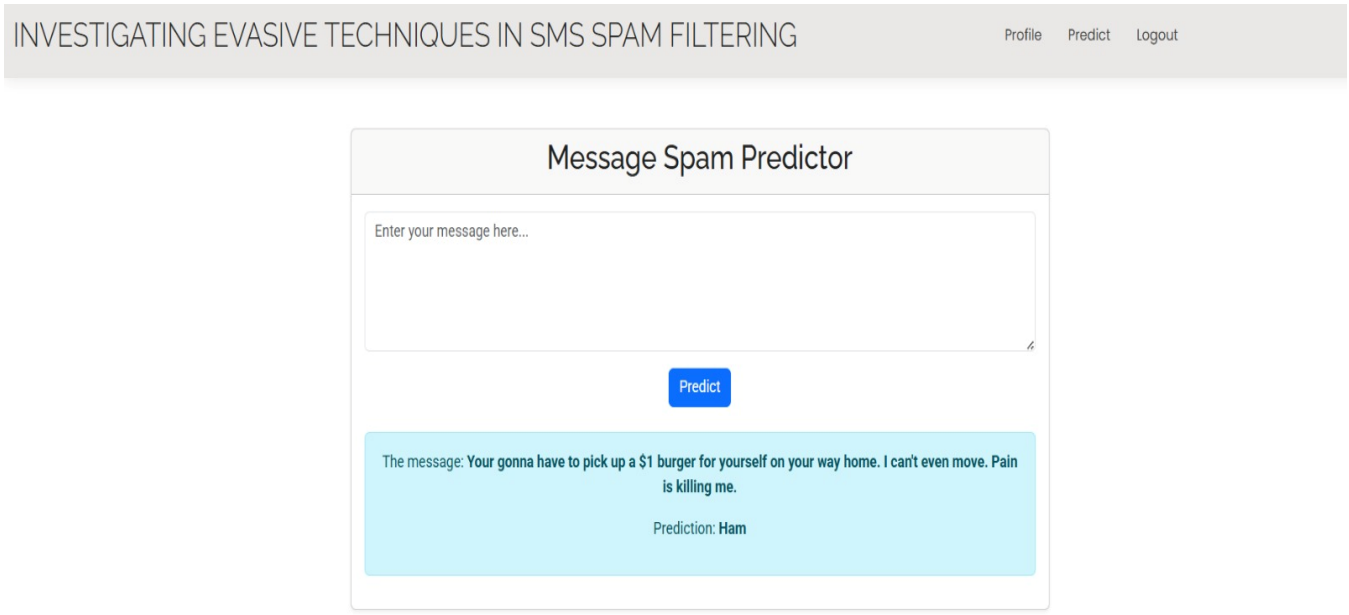

Fig. 8.4 Ham Message Detection Using Spam Prediction System

Description:

The figure illustrates the message spam prediction interface analyzing a normal conversational text message. Users can enter a message into the input field and use the "Predict" button to classify it. The system evaluates the content and displays both the message and its classification result. In this case, the message is identified as "Ham," indicating it is a legitimate, non-spam message. This highlights the system's capability to distinguish genuine personal communication from spam content.

9. CONCLUSION

The investigation into SMS spam detection highlights the growing challenges posed by evolving spam techniques and the need for intelligent and adaptive filtering systems. The system is designed using a structured pipeline that includes data collection, preprocessing, feature extraction, model training, and comparative evaluation, ensuring a systematic and end-to-end approach to addressing SMS spam detection challenges.

During the implementation phase, raw SMS data was preprocessed through multiple steps such as text cleaning, normalization, tokenization, and noise removal. The processed text was transformed into numerical representations using TF-IDF and semantic features, enabling models to capture meaningful patterns and improve classification accuracy. This preprocessing stage played a key role in handling unstructured and noisy SMS data.

The study utilized a large-scale dataset of over 68,000 SMS messages and implemented models such as Support Vector Machines (SVM), Naïve Bayes, Logistic Regression, and Decision Trees. The results show that while these models perform well under normal conditions, their performance varies when exposed to evasive spam techniques such as obfuscation, spacing manipulation, and character substitution.

A key contribution of this project is the comparative analysis of model performance under both normal and adversarial conditions. The findings highlight that no single model is fully robust against all types of spam evasion, emphasizing the importance of preprocessing and feature engineering. The results also indicate that advanced and hybrid approaches can further improve detection performance.

The project also addresses real-world challenges such as data variability, noise, and concept drift. By evaluating the system under realistic conditions, it demonstrates reliability and adaptability for practical deployment in SMS filtering systems. The modular design ensures scalability and allows future enhancements with minimal changes.

Additionally, the analysis of evasive techniques provides deeper insights into how spammers manipulate text to bypass detection systems. Understanding these strategies helps in designing more robust models and improving the resilience of spam filtering mechanisms against adversarial attacks. This highlights the importance of continuous monitoring and updating of detection systems.

Beyond technical contributions, the project emphasizes the importance of safeguarding users from fraudulent and malicious SMS messages. Effective spam detection systems play a crucial role in preventing financial fraud, protecting user privacy, and ensuring secure communication environments.

In conclusion, the project demonstrates that machine learning models can effectively detect SMS spam while also revealing their limitations against evasive techniques. It provides valuable insights into improving spam detection systems and highlights the need for continuous research and adaptation to evolving spam strategies **[6]**.

10. FUTURE ENHANCEMENTS

The project “Investigating Evasive Techniques in SMS Spam Filtering: A Comparative Analysis of Machine Learning Models” provides a strong foundation for further research and development in spam detection systems. However, there are several areas where the system can be extended and improved in the future.

One of the major areas of future scope is the use of advanced deep learning techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer-based models. These approaches can better capture contextual relationships in text and improve the detection of complex and highly obfuscated spam messages.

Another important direction is the development of adaptive and real-time learning systems. Instead of relying on static datasets, future systems can continuously learn from new incoming data, enabling them to adapt quickly to evolving spam patterns and newly emerging evasive techniques.

The system can also be extended beyond SMS spam detection to other communication platforms such as emails, social media, and messaging applications. This would enhance the applicability and scalability of the system in real-world scenarios.

Incorporating multilingual support is another key area for future work. Since spam messages are often sent in multiple languages, enabling the system to process and classify multilingual text would significantly improve its effectiveness and global usability.

Future research can also focus on developing hybrid and ensemble models that combine multiple machine learning algorithms. Such approaches can improve prediction accuracy and robustness by leveraging the strengths of different models.

Another potential enhancement is the integration of advanced feature extraction techniques such as word embeddings (Word2Vec, GloVe) and contextual embeddings. These techniques can capture semantic meaning more effectively than traditional methods like TF-IDF.

The system can further be improved by incorporating deep learning-based adversarial training, where models are trained on both normal and intentionally modified (evasive) data. This will help in building more resilient systems capable of detecting sophisticated spam patterns.

Additionally, future work can explore the use of real-time deployment and cloud-based systems, enabling the model to process large volumes of messages efficiently and provide instant predictions.

Improving dataset quality and size is another important aspect. Collecting larger and more diverse datasets, including real-world spam examples, will help in improving model generalization and reducing bias.

The integration of user feedback mechanisms can also enhance system performance. By allowing users to report incorrect classifications, the system can learn and improve over time through feedback-driven learning.

Security and privacy considerations can be further strengthened by implementing secure data handling practices and ensuring that user data is protected during processing and storage.

REFERENCES

1. L. Shen, Y. Wang, Z. Li, and W. Ma, "SMS spam detection using BERT and multi-graph convolutional networks," *Int. J. Intelligent Networks*, vol. 6, pp. 85–97, 2026.
2. M. Ahmadi, M. Khajavi, A. Varmaghani, A. Ala, K. Danesh, and D. Javaheri, "Leveraging large language models for cybersecurity: Enhancing SMS spam detection with robust and context-aware text classification," arXiv preprint arXiv:2502.11014, 2025.
3. H. C. Altunay and Z. Albayrak, "SMS spam detection system based on deep learning architectures for Turkish and English messages," *Applied Sciences*, vol. 14, no. 24, p. 11804, 2025.
4. Y. Li, R. Zhang, W. Rong, and X. Mi, "SpamDam: Towards privacy-preserving and adversary-resistant SMS spam detection," arXiv preprint arXiv:2404.09481, 2024.
5. S. Maheshwari, S. Aggarwal, and R. Kaushal, "A novel SMS spam dataset and bi-directional transformer based short-text representations for SMS spam detection," *Int. J. Information and Decision Sciences*, vol. 16, no. 4, pp. 341–359, 2024.
6. T. Lakshman, S. S. Kumar, U. S. Kumar, Y. S. Sekhar, and Y. Suresh, "SMS spam detection in machine learning using natural language processing," *Int. J. Advance Research, Ideas and Innovations in Technology*, vol. 9, no. 5, pp. 112–118, 2023.
7. FCC. (2022). The Top Text Scams of 2022. Accessed: Oct. 8, 2023. [Online]. Available: https://www.ftc.gov/news-events/data-visualizations/data-spotlight/2023/06/iykyk-top-text-scams-2022
8. ACCS. (2022). Accs Scam Statistics. [Online]. Available: https://www.scamwatch.gov.au/scam-statistics
9. M. A. Abid, S. Ullah, M. A. Siddique, M. F. Mushtaq, W. Aljedaani, and F. Rustam, ''Spam SMS filtering based on text features and supervised machine learning techniques,'' Multimedia Tools Appl., vol. 81, no. 28, pp. 39853–39871, Nov. 2022.
10. C. Oswald, S. E. Simon, and A. Bhattacharya, ''SpotSpam: Intention snalysis-driven SMS spam detection using BERT embeddings,'' ACM Trans. Web, vol. 16, no. 3, pp. 1–27, Aug. 2022.
11. S. Y. Yerima and A. Bashar, ''Semi-supervised novelty detection with one class SVM for SMS spam detection,'' in Proc. 29th Int. Conf. Syst., Signals Image Process. (IWSSIP), Jun. 2022, pp. 1–4.
12. S. Tang, X. Mi, Y. Li, X. Wang, and K. Chen, ''Clues in tweets: Twitterguided discovery and analysis of SMS spam,'' in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Nov. 2022, pp. 2751–2764.
13. S. Rojas-Galeano, ''Using BERT encoding to tackle the mad-lib attack in SMS spam detection,'' 2021, arXiv:2107.06400.

14. T. Xia and X. Chen, ‘‘A discrete hidden Markov model for SMS spam detection,’’ Appl. Sci., vol. 10, no. 14, p. 5011, Jul. 2020.
15. M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, ‘‘A comparative study of spam SMS detection using machine learning classifiers,’’ in Proc. 11th Int. Conf. Contemp. Comput. (IC3), Aug. 2018, pp. 1–7.
16. S. M. Abdulhamid, M. S. A. Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, ‘‘A review on mobile SMS spam filtering techniques,’’ IEEE Access, vol. 5, pp. 15650–15666, 2017.
17. Ahmed, R. Ali, D. Guan, Y.-K. Lee, S. Lee, and T. Chung, ‘‘Semisupervised learning using frequent itemset and ensemble learning for SMS classification,’’ Expert Syst. Appl., vol. 42, no. 3, pp. 1065–1073, Feb. 2015.
18. A. Al-Hasan and E.-S.-M. El-Alfy, ‘‘Dendritic cell algorithm for mobile phone spam filtering,’’ Proc. Comput. Sci., vol. 52, pp. 244–251, Jan. 2015.
19. T. Almeida, J. M. Hidalgo, and T. Silva, ‘‘Towards SMS spam filtering: Results under a new dataset,’’ JiSS, vol. 2, no. 1, pp. 1–18, 2013.
20. Narayan and P. Saxena, ‘‘The curse of 140 characters: Evaluating the efficacy of SMS spam detection on Android,’’ in Proc. 3rd ACM Workshop Secur. Privacy Smartphones Mobile Devices, Nov. 2013, pp. 33–42.