



**Prifysgol Cymru**  
Y Drindod Dewi Sant  
**University of Wales**  
Trinity Saint David

# **A Comparative Evaluation of Lightweight Machine Learning Models for SMS Spam and Smishing Detection**

**By:**

Bryan Alexis Rivas Dominguez

**Supervisor:** Renuga Jayakumar

Project submitted as part of the requirements for the award of  
MSc Cyber Security and Digital Forensics

February 2026

## **Appendix 1: Declaration and Form of Consent**

### **DECLARATION**

I, Bryan Alexis Rivas Dominguez declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Bryan Rivas

Date: 03/02/2026

### **FORM OF CONSENT**

**I Bryan Alexis Rivas Dominguez** hereby consent that my Project, submitted in candidature for the MSc.CYBER SECURITY AND DIGITAL FORENSICS, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Bryan Rivas

Date: 03/02/2026

Where necessary replace M.Sc.Cyber Security and Digital Forensics with your named award.

## Table of contents

### Contents

|                                                                        |           |
|------------------------------------------------------------------------|-----------|
| <b>Appendix 1: Declaration and Form of Consent.....</b>                | <b>2</b>  |
| <b>List of Figures.....</b>                                            | <b>6</b>  |
| <b>List of Tables .....</b>                                            | <b>7</b>  |
| <b>ABSTRACT .....</b>                                                  | <b>8</b>  |
| <b>CHAPTER 1: INTRODUCTION .....</b>                                   | <b>9</b>  |
| 1.1 Background and Context .....                                       | 9         |
| 1.2 Problem Statement .....                                            | 9         |
| 1.3 Research Aim.....                                                  | 10        |
| 1.4 Research Objectives .....                                          | 10        |
| 1.5 Research Question.....                                             | 10        |
| 1.6 Significance of the Study.....                                     | 10        |
| 1.7 Structure of the Report.....                                       | 11        |
| <b>CHAPTER 2: LITERATURE REVIEW.....</b>                               | <b>12</b> |
| 2.1 Overview of SMS Spam and Smishing .....                            | 12        |
| 2.1.1 Definitions and Key Concepts.....                                | 12        |
| 2.1.2 Impact on Users and Organisations .....                          | 12        |
| 2.2 Machine Learning for SMS Spam Detection .....                      | 13        |
| 2.2.1 Supervised and unsupervised learning approaches.....             | 13        |
| 2.2.2 Classical machine learning models for SMS spam detection.....    | 14        |
| 2.2.3 Feature extraction techniques for text-based classification..... | 14        |
| 2.3 Datasets Used in Previous Studies.....                             | 15        |
| 2.4 Evaluation Metrics in Prior Work.....                              | 15        |
| 2.5 Identified Research Gap.....                                       | 16        |
| 2.6 Summary of the Literature Review .....                             | 16        |
| <b>CHAPTER 3: RESEARCH METHODOLOGY .....</b>                           | <b>17</b> |
| 3.1 Research Philosophy and Approach .....                             | 17        |
| 3.2 Research Strategy.....                                             | 17        |
| 3.3 Overall Experimental Workflow .....                                | 17        |
| 3.4 Dataset Description .....                                          | 19        |

|                                                                            |           |
|----------------------------------------------------------------------------|-----------|
| <b>3.5 Data Cleaning and Preprocessing .....</b>                           | <b>19</b> |
| <b>3.6 Feature Extraction and Selection .....</b>                          | <b>20</b> |
| <b>3.7 Model Development Overview .....</b>                                | <b>20</b> |
| <b>3.8 Ethical Considerations .....</b>                                    | <b>20</b> |
| <b>3.9 Chapter Summary .....</b>                                           | <b>21</b> |
| <b>3.10 Conceptual Design of the SMS Classification Application .....</b>  | <b>21</b> |
| <b>CHAPTER 4: DESIGN PROCESS (EXPERIMENT DESIGN / SYSTEM DESIGN) .....</b> | <b>22</b> |
| <b>4.1 Overall Experimental Design.....</b>                                | <b>22</b> |
| <b>4.1.1 High-Level Overview of the Experiment .....</b>                   | <b>22</b> |
| <b>4.1.2 Input, Processing and Output Stages.....</b>                      | <b>22</b> |
| <b>4.2 System Architecture for SMS Classification.....</b>                 | <b>22</b> |
| <b>4.2.1 Architecture Diagram .....</b>                                    | <b>22</b> |
| <b>4.2.2 Components and Their Roles.....</b>                               | <b>24</b> |
| <b>4.3 Data Processing and Model Pipeline Design.....</b>                  | <b>25</b> |
| <b>4.3.1 SMS Ingestion and Preprocessing Flow.....</b>                     | <b>25</b> |
| <b>4.3.2 Feature Vector Generation (TF-IDF Pipeline).....</b>              | <b>25</b> |
| <b>4.3.3 Model Selection and Training Flow .....</b>                       | <b>26</b> |
| <b>4.4 Test Design and Evaluation Plan.....</b>                            | <b>26</b> |
| <b>4.4.1 Train-Test Split and Validation Strategy.....</b>                 | <b>26</b> |
| <b>4.4.2 Planned Evaluation Metrics and Comparisons .....</b>              | <b>26</b> |
| <b>CHAPTER 5: IMPLEMENTATION (EXPERIMENT).....</b>                         | <b>27</b> |
| <b>5.1 Implementation Environment.....</b>                                 | <b>27</b> |
| <b>5.2 Project Structure and Dataset Preparation .....</b>                 | <b>27</b> |
| <b>5.3 Feature Extraction Implementation (TF-IDF) .....</b>                | <b>29</b> |
| <b>5.4 Model Implementation and Training.....</b>                          | <b>29</b> |
| <b>5.4.1 Naive Bayes Implementation .....</b>                              | <b>30</b> |
| <b>5.4.2 Random Forest Implementation .....</b>                            | <b>31</b> |
| .....                                                                      | 31        |
| <b>5.5 Model Evaluation and Comparative Analysis .....</b>                 | <b>32</b> |
| <b>5.6 Proof-of-Concept SMS Classification Application .....</b>           | <b>32</b> |
| <b>5.7 Implementation Challenges and Solutions .....</b>                   | <b>34</b> |

|                                                                           |           |
|---------------------------------------------------------------------------|-----------|
| <b>CHAPTER 6: EVALUATION OF RESULTS .....</b>                             | <b>35</b> |
| <b>6.1 Model Performance Evaluation .....</b>                             | <b>35</b> |
| 6.1.1 Evaluation of Naive Bayes Model Performance.....                    | 35        |
| 6.1.2 Evaluation of Random Forest Model Performance.....                  | 36        |
| <b>6.2 Comparative Evaluation of Naïve Bayes and Random Forest.....</b>   | <b>38</b> |
| <b>6.3 Comparison with Existing Studies .....</b>                         | <b>39</b> |
| <b>6.4 Critical Discussion of Results .....</b>                           | <b>40</b> |
| <b>6.5 Strengths of the Experimental Setup .....</b>                      | <b>41</b> |
| <b>6.6 Limitations and Threats to Validity.....</b>                       | <b>41</b> |
| <b>CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS.....</b>                    | <b>42</b> |
| 7.1 Conclusions.....                                                      | 42        |
| 7.2 Contributions of the Study .....                                      | 42        |
| 7.3 Recommendations .....                                                 | 43        |
| 7.4 Suggestions for Further Work .....                                    | 43        |
| 7.5 Final Remarks.....                                                    | 44        |
| <b>CHAPTER 8: REFLECTION.....</b>                                         | <b>45</b> |
| 8.1 Time Management and Planning .....                                    | 45        |
| 8.2 Development of Study and Research Skills .....                        | 45        |
| 8.3 Learning About My Own Learning Style.....                             | 45        |
| 8.4 Areas for Improvement .....                                           | 46        |
| 8.5 Reflection on Overall Learning.....                                   | 46        |
| <b>REFERENCES: .....</b>                                                  | <b>47</b> |
| <b>APPENDICES .....</b>                                                   | <b>49</b> |
| <b>Appendix A – Code Snippets.....</b>                                    | <b>49</b> |
| Appendix A.1 – Text Preprocessing and TF-IDF Feature Extraction .....     | 49        |
| Appendix A.2 – Model Training and Evaluation .....                        | 49        |
| <b>Appendix B – Dataset Standardisation and Class Harmonisation .....</b> | <b>50</b> |
| <b>Appendix C – Dataset Samples .....</b>                                 | <b>51</b> |

## List of Figures

|                                                                                                                                                                                                                                                           |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1 Illustrates the overall experimental workflow adopted in this study, from dataset selection through preprocessing, model training, and performance evaluation. ....                                                                              | 18 |
| Figure 2 Proposed SMS spam and smishing classification architecture illustrating the end-to-end pipeline from dataset input and text preprocessing to feature extraction, model classification, evaluation metrics, and conceptual validation layer. .... | 23 |
| Figure 3: Naive Bayes classification results using TF-IDF features on the Kaggle SMS dataset.....                                                                                                                                                         | 29 |
| Figure 4: Naive Bayes model training and evaluation results obtained using TF-IDF features on the Kaggle SMS dataset. ....                                                                                                                                | 30 |
| Figure 5 Random Forest classification results using TF-IDF features on the Kaggle SMS dataset.....                                                                                                                                                        | 31 |
| Figure 6 Streamlit-based proof-of-concept SMS spam/smishing classification interface .                                                                                                                                                                    | 33 |
| Figure 7 Dataset standardisation and class harmonisation across Kaggle, Mendeley and UCI datasets. ....                                                                                                                                                   | 50 |
| Figure 8 SMS spam and smishing detection system validation using a Streamlit-based interface. ....                                                                                                                                                        | 51 |

## List of Tables

|                                                                                       |    |
|---------------------------------------------------------------------------------------|----|
| Table 1 Summarises the performance of the Naive Bayes model across all datasets. .... | 35 |
| Table 2 Performance of Random Forest Model Across Datasets .....                      | 37 |
| Table 3 Model Comparison Summary .....                                                | 39 |

## ABSTRACT

This project investigates the effectiveness of lightweight machine learning techniques for detecting SMS spam and smishing messages, with an emphasis on practicality, reproducibility, and suitability for resource-constrained environments. Smishing remains a growing cybersecurity threat, and automated text classification can support earlier identification of malicious messages before users interact with harmful links or social engineering content. The aim of this study was to design, implement, and evaluate an end-to-end SMS classification pipeline using traditional supervised learning models and publicly available datasets.

A quantitative, experiment-based methodology was adopted. Three datasets were used to test generalisability across different message characteristics: the UCI SMS Spam Collection, a Kaggle SMS spam dataset, and the Mendeley Smishing 2022 dataset. Messages were pre-processed using a consistent pipeline and transformed into numerical features using TF-IDF. Two classification algorithms were implemented and evaluated under the same conditions: Multinomial Naive Bayes (as a lightweight baseline) and Random Forest (as a stronger ensemble benchmark). Model performance was assessed using accuracy, precision, recall, and F1-score, with particular focus on the malicious class due to its direct relevance to smishing detection.

Results show that both models achieved high overall accuracy across datasets, with stronger performance on the Mendeley Smishing 2022 and UCI datasets. Naive Bayes achieved consistently very high precision for the malicious class (above 99% across all datasets), indicating a low false-positive tendency, although recall was notably lower on the Kaggle dataset. Random Forest produced a more balanced performance, improving recall and F1-score across datasets, and achieved the strongest results on the Mendeley dataset. Based on these findings, Naive Bayes was selected as the primary model for a simple demonstration SMS classification application due to its lower computational cost, faster training, and ease of deployment, while Random Forest served as a comparative benchmark to validate robustness. Overall, the project demonstrates that lightweight machine learning remains effective for SMS spam and smishing detection and provides a reproducible experimental framework for future extensions.

## **CHAPTER 1: INTRODUCTION**

### **1.1 Background and Context**

Mobile communication has become an essential part of daily life, yet this dependency has also created new opportunities for cybercriminals. One of the most widespread threats affecting users today is SMS spam and smishing fraudulent text messages designed to trick individuals into revealing personal information, clicking malicious links, or responding to deceptive prompts. These attacks continue to grow in volume and sophistication, largely because sending SMS messages remains cheap, fast, and difficult to trace. As a result, users with limited technical experience are often the most vulnerable, particularly when the message appears to come from trusted organisations such as delivery companies, banks, or government agencies.

In the UK, this issue has become increasingly visible. Many individuals regularly receive fake delivery notifications, bank alerts or tax-related messages that appear legitimate at first glance. The rise of such incidents has created significant concern among everyday users, as well as among organisations trying to protect their customers. This growing trend highlights the urgent need for reliable and accessible detection methods that can help filter malicious SMS content before users interact with it.

Because of this rising threat, researchers and industry professionals have developed various automated techniques to detect and filter malicious SMS messages. Machine Learning (ML) has shown strong potential in this area, especially for classifying large volumes of text efficiently. Traditional ML models such as Naive Bayes and Random Forest remain popular due to their simplicity, interpretability, and relatively low computing requirements. However, despite ongoing progress, many users continue to be exposed to smishing attempts, highlighting the need for practical and effective classification models that can support real-world protection.

### **1.2 Problem Statement**

Although a wide range of research has explored methods for SMS spam detection, several challenges remain. Many existing solutions rely on complex architectures that require large datasets, specialised hardware or deep learning techniques that may not be feasible for all environments. Furthermore, performance varies significantly between models depending on dataset characteristics, preprocessing techniques and evaluation methods. Users with limited technical experience may still struggle to identify fraudulent messages, and existing

automated systems do not always generalise well across different types of smishing attempts. Therefore, a focused evaluation of lightweight and accessible ML models is needed to understand how effective they can be at detecting malicious SMS content.

### **1.3 Research Aim**

The aim of this study is to design, implement and evaluate a Machine Learning-based approach for classifying SMS messages as legitimate or malicious. The project focuses on accessible ML algorithms that can be deployed in real-world mobile communication contexts without requiring extensive computational resources.

### **1.4 Research Objectives**

**To achieve this aim, the study is guided by the following objectives:**

1. To review existing research on SMS spam and smishing detection, with particular attention to classical ML methods.
2. To design an experimental pipeline that includes preprocessing, feature extraction and model development using lightweight ML algorithms.
3. To implement and train selected ML models (Naive Bayes and Random Forest) using a publicly available SMS dataset.
4. To evaluate model performance using standard classification metrics and compare the results against findings from existing studies.

### **1.5 Research Question**

The central research question addressed in this project is:

To what extent can lightweight Machine Learning models accurately detect SMS spam and smishing messages in modern mobile communication environments?

### **1.6 Significance of the Study**

Understanding which ML techniques provide the most reliable and accessible solution for SMS spam detection is important for improving user protection, especially for individuals who may not have advanced technical knowledge. By focusing on models that are easy to implement and computationally efficient, this study contributes to the development of practical defence mechanisms that can be integrated into mobile applications or security

tools. The findings may also support further research by highlighting strengths and limitations in current approaches.

## **1.7 Structure of the Report**

**The remainder of this report is organised as follows:**

- **Chapter 2:** reviews relevant literature on SMS spam, smishing and ML-based classification techniques.
- **Chapter 3:** outlines the methodological framework, including the research philosophy, strategy and dataset description.
- **Chapter 4:** presents the design of the experimental system, detailing the architecture, processing steps and planned evaluation approach.
- **Chapter 5:** describes the implementation of the system and the development of the classification models.
- **Chapter 6:** evaluates the results obtained through experimentation and compares them with previous studies.
- **Chapter 7:** summarises the main findings, discusses the achievement of the research objectives and offers recommendations for future work.
- **Chapter 8:** provides a personal reflection on the learning experience and the development of academic and technical skills throughout the project.

## **CHAPTER 2: LITERATURE REVIEW**

### **2.1 Overview of SMS Spam and Smishing**

Short Message Service (SMS) spam and smishing represent a growing cybersecurity threat affecting both individuals and organisations worldwide. Smishing refers to phishing attacks delivered via SMS messages, where attackers attempt to deceive recipients into disclosing sensitive information, clicking malicious links, or performing fraudulent actions. Unlike traditional email-based phishing, smishing exploits the perceived trust and immediacy associated with mobile text messages, increasing the likelihood of successful attacks.

Recent reports indicate a significant rise in the volume and sophistication of SMS-based attacks [1]. Cybercriminals increasingly rely on social engineering techniques combined with technical evasion strategies to bypass existing spam filters. As mobile devices have become a primary communication channel, SMS has emerged as an attractive attack vector due to its widespread usage and limited native security mechanisms.

#### **2.1.1 Definitions and Key Concepts**

SMS spam typically refers to unsolicited bulk messages sent for advertising, scams or malicious purposes. Smishing is a more targeted form of SMS spam, where attackers impersonate trusted entities such as banks, delivery services or government organisations to manipulate users into revealing personal or financial information.

Modern smishing campaigns often employ deceptive tactics such as URL shortening, character substitution and message fragmentation to evade detection mechanisms. These techniques allow malicious messages to closely resemble legitimate communications, making them particularly difficult to identify using traditional filtering approaches [2].

#### **2.1.2 Impact on Users and Organisations**

The impact of SMS spam and smishing extends beyond individual financial losses. For end users, successful attacks can result in identity theft, unauthorised account access and long-term financial harm. Non-technical users are especially vulnerable, as smishing messages are frequently designed to exploit urgency, fear or trust.

From an organisational perspective, smishing attacks contribute to increased support costs, reputational damage and potential regulatory consequences. Organisations are often required to invest additional resources in incident response, customer support and security awareness

training to mitigate the effects of these attacks. As smishing campaigns continue to evolve, existing detection mechanisms struggle to maintain effectiveness, highlighting the need for adaptive and intelligent solutions [3].

## **2.2 Machine Learning for SMS Spam Detection**

Machine learning (ML) has become one of the most widely adopted approaches for addressing large-scale cybersecurity challenges, including SMS spam and smishing detection. Unlike traditional rule-based or keyword filtering techniques, machine learning systems are capable of learning patterns from data and adapting to new or previously unseen attack strategies. This makes ML particularly suitable for text-based classification problems, where malicious messages often evolve rapidly to evade static detection mechanisms.

In the context of SMS spam detection, machine learning enables automated and scalable analysis of large volumes of messages, allowing for faster identification and mitigation of malicious content. By analysing linguistic patterns, word distributions and contextual features within SMS messages, ML models can distinguish between legitimate and malicious communications with minimal human intervention [4].

### **2.2.1 Supervised and unsupervised learning approaches**

Machine learning techniques are commonly categorised into supervised and unsupervised learning. Supervised learning relies on labelled datasets, where messages are pre-classified as legitimate or spam, allowing the model to learn the relationship between input features and known outcomes. This approach is particularly effective for SMS spam detection, as labelled datasets are widely available and enable reliable performance evaluation.

Unsupervised learning, in contrast, does not require labelled data and instead focuses on identifying hidden patterns or anomalies within the dataset. While unsupervised methods can be useful for detecting novel or previously unseen threats, they are less commonly applied in SMS spam detection due to the difficulty of interpreting results and evaluating accuracy.

Given the availability of labelled SMS datasets and the objective of accurately classifying messages as spam or legitimate, this study adopts a supervised machine learning approach, which is widely recognised as the standard paradigm for text classification tasks when annotated data is available [8].

### **2.2.2 Classical machine learning models for SMS spam detection**

Classical supervised machine learning models continue to play an important role in SMS spam detection due to their computational efficiency, interpretability, and suitability for deployment in resource-constrained environments. These characteristics make them particularly appropriate for real-world applications where fast inference and low overhead are essential.

This study focuses on two widely adopted classical classifiers: Naive Bayes and Random Forest. Both models have been extensively evaluated in prior research and have demonstrated reliable performance in SMS spam and phishing detection tasks [5].

Naive Bayes is a probabilistic classifier derived from Bayes' theorem and operates under the assumption of feature independence. Despite this simplifying assumption, it has shown strong effectiveness in text classification problems, especially when dealing with sparse and high-dimensional feature spaces such as those produced by SMS messages. Its low computational complexity makes it a suitable baseline model for lightweight detection systems.

Random Forest, on the other hand, is an ensemble-based algorithm that combines multiple decision trees to improve predictive accuracy and reduce overfitting. By leveraging randomness in both data sampling and feature selection, Random Forest is able to capture more complex patterns within textual data, which can be beneficial when dealing with diverse or obfuscated smishing messages.

Selecting these two models enables a comparative evaluation between a lightweight probabilistic classifier and a more robust ensemble approach, directly supporting the objective of analysing the trade-offs between performance and computational cost in SMS spam and smishing detection.

### **2.2.3 Feature extraction techniques for text-based classification**

In order to apply machine learning algorithms to SMS data, textual messages must first be converted into numerical representations. Feature extraction techniques are therefore a fundamental component of text-based classification systems.

Common approaches include Bag of Words (BoW) and Term Frequency–Inverse Document Frequency (TF-IDF), which represent messages based on term occurrence and relative

importance. These techniques allow classifiers to identify distinguishing patterns between legitimate and malicious SMS messages.

In this project, TF-IDF is adopted as the primary feature extraction method due to its effectiveness, simplicity, and widespread use in SMS spam detection research. TF-IDF is particularly suitable for short text messages, as it emphasises discriminative terms while reducing the impact of commonly occurring words [9]. The practical implementation of TF-IDF within the experimental pipeline is detailed in later chapters.

### **2.3 Datasets Used in Previous Studies**

Previous research on SMS spam and smishing detection has largely relied on publicly available and labelled datasets, which enable supervised machine learning evaluation. These datasets typically consist of collections of legitimate and spam SMS messages derived from real-world communications or user reports.

However, existing literature also highlights several limitations associated with commonly used SMS datasets. These include class imbalance, limited representation of modern smishing tactics, and simplified linguistic patterns, which may restrict the generalisability of trained models when applied to real-world scenarios.

To address these limitations, this study adopts a multi-dataset evaluation approach, enabling the assessment of model behaviour across datasets with differing characteristics. This approach supports a more robust analysis of lightweight machine learning models under varied and realistic conditions.

### **2.4 Evaluation Metrics in Prior Work**

To assess the effectiveness of SMS spam detection models, prior studies commonly employ standard classification metrics, including accuracy, precision, recall and F1-score. These metrics provide insight into both overall performance and the model’s ability to correctly identify malicious messages while minimising false positives. In addition, confusion matrices are frequently used to visualise classification outcomes and analyse error distribution across classes. Such evaluation measures are widely adopted in the literature and form the basis for comparing model performance in SMS spam detection research.

Recent literature also emphasises that relying solely on accuracy may be misleading, particularly in imbalanced classification problems. Obi (2023) demonstrates that accuracy can remain high even when sensitivity (recall) is zero, suggesting that precision, recall,

specificity and F1-score should be examined alongside accuracy to obtain a more reliable evaluation of classifier performance. These metrics are derived directly from the confusion matrix and provide a clearer understanding of error distribution across classes [7].

## **2.5 Identified Research Gap**

Despite extensive research on SMS spam and smishing detection, several limitations remain evident in existing studies. Many prior works focus on achieving high classification accuracy using controlled or clean datasets, which may not fully reflect real-world smishing scenarios where messages are short, noisy, and intentionally obfuscated.

In addition, while a wide range of machine learning techniques has been proposed, limited research has systematically evaluated how lightweight and computationally efficient classifiers perform across multiple datasets with differing characteristics. In particular, there is a lack of comparative evaluation of classical machine learning models under consistent preprocessing and evaluation conditions.

Given the practical constraints of real-world deployment environments, such as limited computational resources and the need for fast inference, further investigation is required to assess whether accessible models like Naive Bayes and Random Forest can provide reliable and robust smishing detection performance.

This research addresses this gap by conducting a comparative evaluation of two classical supervised machine learning algorithms using multiple publicly available SMS datasets and standard evaluation metrics.

## **2.6 Summary of the Literature Review**

This chapter reviewed existing literature on SMS spam and smishing detection, highlighting the growing threat posed by SMS-based attacks and the application of classical machine learning techniques to address this challenge. The review identified a research gap in evaluating lightweight and computationally efficient machine learning models under realistic smishing conditions, which forms the basis for the methodology presented in the next chapter.

## **CHAPTER 3: RESEARCH METHODOLOGY**

### **3.1 Research Philosophy and Approach**

This study follows a pragmatic and quantitative approach, focused on evaluating measurable model performance through controlled experimentation. A deductive strategy is adopted, as the research tests established machine learning techniques (Naive Bayes and Random Forest) against predefined evaluation metrics (accuracy, precision, recall, and F1-score) to assess their suitability for SMS spam and smishing detection. This approach supports objective comparison, reproducibility, and clear validation of findings.

### **3.2 Research Strategy**

This study adopts an experiment-based research strategy to evaluate the effectiveness of machine learning models for SMS spam and smishing detection. The focus of the research is on assessing lightweight supervised classifiers in terms of performance, practicality, and suitability for deployment in environments with limited computational resources.

An experimental approach is appropriate for this study as it allows controlled testing of machine learning models using consistent datasets, preprocessing steps, and evaluation criteria. This enables meaningful comparison of model behaviour and performance while maintaining transparency and reproducibility throughout the research process.

### **3.3 Overall Experimental Workflow**

The research follows a structured experimental workflow designed to ensure consistency across all stages of the study. The workflow consists of the following key phases:

1. Dataset collection
2. Data cleaning and preprocessing
3. Feature extraction
4. Model training
5. Model evaluation
6. Comparative analysis of results

This workflow ensures that raw SMS data is systematically prepared prior to training and that evaluation outcomes accurately reflect the generalisation ability of the models on unseen messages.

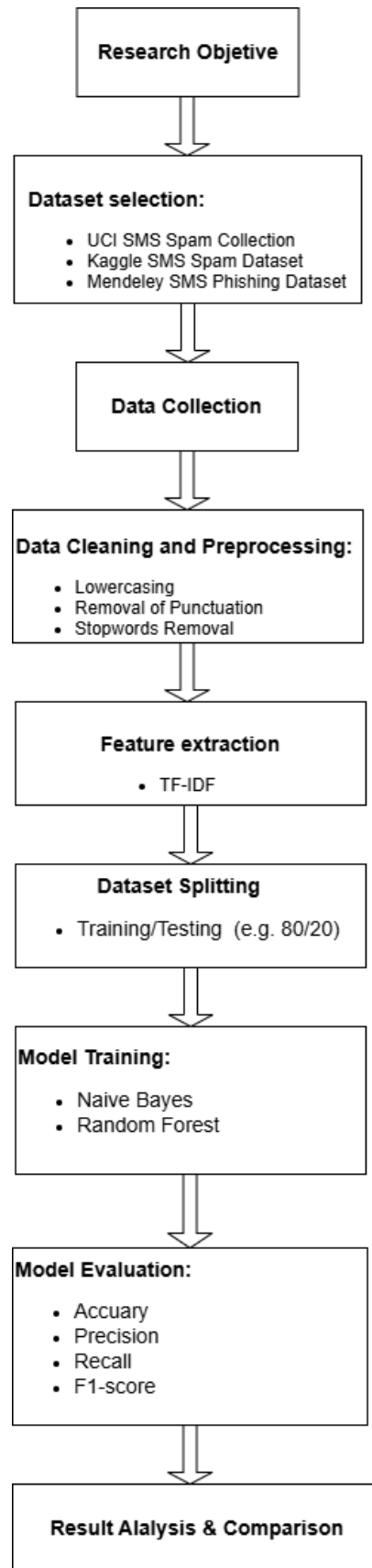


*Figure 1 Illustrates the overall experimental workflow adopted in this study, from dataset selection through preprocessing, model training, and performance evaluation.*

### 3.4 Dataset Description

This study utilises three publicly available SMS datasets that have been widely adopted in prior research on spam and smishing detection. The datasets were selected to support transparency, reproducibility, and robustness by evaluating model performance across different message characteristics and attack patterns.

The UCI SMS Spam Collection contains 5,572 English-language messages labelled as *ham* or *spam* and was used as a benchmark dataset due to its frequent use in SMS spam detection studies [10]. A larger dataset from Kaggle, consisting of 10,286 labelled SMS messages, was included to assess model performance on higher-volume inputs and to examine scalability effects [11]. To incorporate more recent, phishing-oriented SMS content, the Mendeley SMS Phishing Dataset (2022) was also included; this dataset provides labels for *legitimate*, *spam*, and *smishing* messages, enabling evaluation against more explicit smishing threats [12].

Each dataset was processed and evaluated independently using the same text preprocessing steps and TF-IDF feature extraction pipeline, ensuring a fair comparison between the selected classifiers. All datasets are publicly available, anonymised, and do not contain personally identifiable information.

### 3.5 Data Cleaning and Preprocessing

Prior to model training, the SMS datasets undergo a series of preprocessing steps aimed at reducing noise and improving feature quality. Text preprocessing is a critical stage in SMS classification, as messages often contain informal language, abbreviations, and irrelevant symbols.

The preprocessing steps applied in this study include:

- Conversion of text to lowercase
- Removal of punctuation and special characters
- Tokenisation of messages into individual terms
- Removal of common stopwords
- Elimination of duplicate messages where applicable

These steps standardise textual input and reduce variability that could negatively affect model performance.

### **3.6 Feature Extraction and Selection**

Following preprocessing, SMS messages are transformed into numerical representations suitable for machine learning algorithms. This study employs Term Frequency–Inverse Document Frequency (TF-IDF) as the feature extraction technique.

TF-IDF assigns weights to terms based on their frequency within individual messages and their importance across the dataset. This method is well suited to SMS classification tasks, as it highlights discriminative terms while reducing the influence of commonly occurring words.

Only text-based features derived from message content are considered. Additional metadata, such as sender information or temporal features, is excluded to ensure consistency across datasets and to focus the evaluation solely on textual characteristics.

### **3.7 Model Development Overview**

Two supervised machine learning models are selected for experimental evaluation:

- Naive Bayes
- Random Forest

These models are chosen based on their widespread adoption in SMS spam detection research and their contrasting characteristics in terms of computational complexity and classification capability.

Each dataset is divided into training and testing subsets using a standard train–test split to evaluate model generalisation on unseen messages. The training data is used to construct the classification models, while the testing data is reserved exclusively for performance evaluation.

Detailed implementation and training procedures are presented in Chapter 5.

### **3.8 Ethical Considerations**

This research exclusively utilises publicly available and anonymised datasets that do not contain personal or sensitive information. No human participants are involved, and no private data is collected or processed.

The study complies with academic ethical standards and dataset usage conditions. Results are reported objectively, without selective reporting or manipulation of findings.

### **3.9 Chapter Summary**

This chapter has outlined the research methodology adopted in the study, including the experimental strategy, dataset selection, preprocessing techniques, feature extraction approach, and model development overview. The methodology is designed to support reproducibility, transparency, and meaningful comparative evaluation.

The next chapter focuses on the system and experimental design, detailing the architecture, processing pipeline, and evaluation plan.

### **3.10 Conceptual Design of the SMS Classification Application**

To demonstrate the practical applicability of the proposed SMS spam and smishing detection framework, a simple conceptual classification application is introduced. The purpose of this application is to validate the trained machine learning models by classifying previously unseen SMS messages as either legitimate or malicious.

The application follows a minimal design focused on functional validation rather than interface complexity. Users input an SMS message, which is processed using the same preprocessing and feature extraction pipeline applied during model training. The processed message is then classified using the selected machine learning model, and the prediction result is presented to the user.

The choice of model integrated into the application is informed by the comparative evaluation conducted during the experimental phase. The application is intended solely as a proof-of-concept, demonstrating real-world applicability of the framework rather than serving as a fully deployed system. Technical implementation details are presented in subsequent chapters.

## CHAPTER 4: DESIGN PROCESS (EXPERIMENT DESIGN / SYSTEM DESIGN)

### 4.1 Overall Experimental Design

This project adopts a lightweight experimental design aimed at evaluating the effectiveness of simple and accessible machine learning models for SMS spam and smishing detection. The design prioritises practicality and reproducibility, avoiding complex architectures or specialised hardware requirements.

The experimental system is designed to operate as an offline text classification pipeline, where SMS messages are processed, transformed into numerical features, and classified using supervised machine learning algorithms.

#### 4.1.1 High-Level Overview of the Experiment

The experimental process is structured as a linear pipeline consisting of data input, preprocessing, feature extraction, model training, and evaluation.

Two classical machine learning models, Naive Bayes and Random Forest are trained and evaluated under identical conditions to enable a fair and meaningful comparison of performance.

#### 4.1.2 Input, Processing and Output Stages

The system design is divided into three main stages:

- **Input:** A publicly available SMS dataset containing labelled messages classified as legitimate or spam/smishing.
- **Processing:** Text preprocessing operations followed by feature extraction using the TF-IDF technique and supervised model training.
- **Output:** Classification predictions and evaluation metrics that indicate each model's ability to correctly detect malicious SMS messages.

### 4.2 System Architecture for SMS Classification

#### 4.2.1 Architecture Diagram

The system architecture for this project is designed as a simple and modular SMS classification pipeline. The architecture focuses on clarity and practicality, ensuring that each component performs a specific role within the overall detection process.

The system operates offline and processes SMS messages sequentially, transforming raw text into numerical representations that can be used by machine learning classifiers. This architectural approach supports transparency, reproducibility, and ease of implementation.

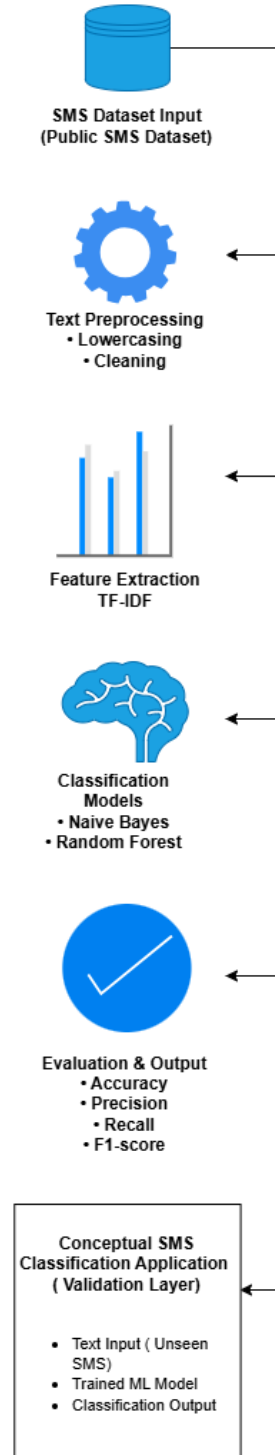


*Figure 2 Proposed SMS spam and smishing classification architecture illustrating the end-to-end pipeline from dataset input and text preprocessing to feature extraction, model classification, evaluation metrics, and conceptual validation layer.*

#### **4.2.2 Components and Their Roles**

**The proposed system architecture consists of the following core components:**

##### **1. SMS Dataset Input**

This component represents the entry point of the system. It consists of a publicly available dataset containing labelled SMS messages classified as legitimate or spam/smishing. The dataset serves as the foundation for training and evaluating the machine learning models.

##### **2. Text Preprocessing Module**

The preprocessing module is responsible for preparing raw SMS text for analysis. This includes basic text cleaning operations such as lowercasing, removal of punctuation, and elimination of irrelevant tokens. The purpose of this stage is to reduce noise and ensure consistency across all messages before feature extraction.

##### **3. Feature Extraction Module (TF-IDF)**

After preprocessing, SMS messages are transformed into numerical feature vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) technique. This component converts textual data into a structured format that captures the importance of words within individual messages relative to the entire dataset, enabling effective classification by machine learning algorithms.

##### **4. Classification Models**

The classification component includes two supervised machine learning models: Naive Bayes and Random Forest. Both models receive the same TF-IDF feature vectors as input and are trained using labelled SMS data. This design ensures a fair comparison of model performance under identical experimental conditions.

##### **5. Evaluation and Output Module**

The final component of the architecture evaluates model predictions using standard classification metrics, including accuracy, precision, recall, and F1-score. The output of the system consists of predicted class labels for SMS messages and quantitative performance results that support comparative analysis between models.

The implementation details of the validation interface and the execution of the experiments are presented in Chapter 5.

In addition, the evaluated models are conceptually validated through a simple SMS classification interface to demonstrate potential real-world applicability.

### **4.2.3 Architectural Design Rationale**

The chosen architecture prioritises simplicity and accessibility over complexity. By avoiding deep learning frameworks and specialised hardware requirements, the system can be implemented on a standard personal computer using widely available machine learning libraries. This design aligns with the project's objective of evaluating lightweight and practical solutions for SMS spam and smishing detection.

## **4.3 Data Processing and Model Pipeline Design**

### **4.3.1 SMS Ingestion and Preprocessing Flow**

The data processing pipeline begins with the ingestion of SMS messages from the selected dataset. Each message is read sequentially and prepared for analysis through a series of preprocessing steps.

Preprocessing includes converting text to lowercase, removing punctuation and special characters, and eliminating unnecessary tokens that do not contribute to classification. These steps aim to reduce noise and ensure that all SMS messages follow a consistent textual format prior to feature extraction.

The preprocessing flow is applied uniformly across the entire dataset to maintain consistency and avoid bias during model training and evaluation.

### **4.3.2 Feature Vector Generation (TF-IDF Pipeline)**

Following preprocessing, SMS messages are transformed into numerical feature vectors using the Term Frequency–Inverse Document Frequency (TF-IDF) method. This pipeline assigns weights to words based on their frequency within individual messages and their relative importance across the dataset.

The TF-IDF representation enables machine learning models to identify distinguishing textual patterns between legitimate and malicious SMS messages. This feature extraction method was selected due to its simplicity, effectiveness, and widespread adoption in text classification research.

### **4.3.3 Model Selection and Training Flow**

The processed TF-IDF feature vectors are used to train two supervised machine learning models: Naive Bayes and Random Forest. These models were selected based on their demonstrated effectiveness in prior SMS spam detection studies and their low computational requirements.

Both models are trained using the same dataset, preprocessing steps, and feature representations. This design ensures a fair comparison of performance and allows differences in results to be attributed to model characteristics rather than variations in data processing.

## **4.4 Test Design and Evaluation Plan**

### **4.4.1 Train-Test Split and Validation Strategy**

To evaluate model performance, the dataset is divided into training and testing subsets using a standard train-test split. The training set is used to train the models, while the testing set is reserved for performance evaluation on unseen data.

This validation strategy allows the study to assess how well each model generalises beyond the data used for training, providing a realistic measure of classification effectiveness.

### **4.4.2 Planned Evaluation Metrics and Comparisons**

Model performance is evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. These metrics provide insight into both overall classification performance and the model's ability to correctly identify malicious SMS messages while minimising false positives.

The results obtained from Naive Bayes and Random Forest are systematically compared to determine which model offers the most effective and practical solution for SMS spam and smishing detection within the scope of this study.

## **CHAPTER 5: IMPLEMENTATION (EXPERIMENT)**

### **5.1 Implementation Environment**

This project was implemented using a local development environment in order to maintain full control over the experimental process and ensure reproducibility. All experiments were executed on a personal computer without the use of specialised hardware or cloud-based services, in line with the project's objective of evaluating lightweight and accessible machine learning solutions.

The implementation was carried out using Python as the primary programming language due to its extensive support for data analysis and machine learning tasks. Development and execution were performed using Visual Studio Code (VS Code) as the integrated development environment, alongside Windows PowerShell for environment configuration and script execution.

The following core libraries were utilised during implementation:

- Pandas for dataset loading and manipulation
- Scikit-learn for feature extraction, model training, and evaluation
- Numpy for numerical operations
- Streamlit for developing a lightweight proof-of-concept SMS classification interface

These libraries were selected due to their stability, widespread adoption in academic research, and compatibility with lightweight experimental setups. All dependencies were installed locally using Python's package management tools.

### **5.2 Project Structure and Dataset Preparation**

To ensure clarity, reproducibility, and ease of experimentation, the project followed a simple and well-organised directory structure. All datasets were stored within a dedicated data folder, while preprocessing, training, and evaluation scripts were maintained separately. This separation reduced ambiguity during execution and supported consistent experimentation across multiple datasets.

Three publicly available SMS datasets were prepared and used independently during the experimental phase:

- UCI SMS Spam Collection
- Kaggle SMS Spam Dataset
- Mendeley SMS Phishing Dataset (2022)

The UCI SMS Spam Collection contains 5,572 English-language SMS messages labelled as legitimate (ham) or spam. This dataset was used as a benchmark dataset due to its widespread adoption in SMS spam detection research. During preparation, a practical issue was encountered as the dataset was distributed without a standard file extension, which initially caused loading errors. This issue was resolved by explicitly specifying the appropriate file format during data ingestion.

The Kaggle SMS Spam Dataset consists of approximately 10,286 labelled SMS messages classified as spam or legitimate. This dataset provided a larger message volume, allowing evaluation of model scalability and behaviour on higher-volume data.

The Mendeley SMS Phishing Dataset focuses specifically on smishing-related threats and contains SMS messages labelled as legitimate, spam, or smishing. This dataset was included to evaluate model effectiveness against modern phishing-oriented SMS attacks.

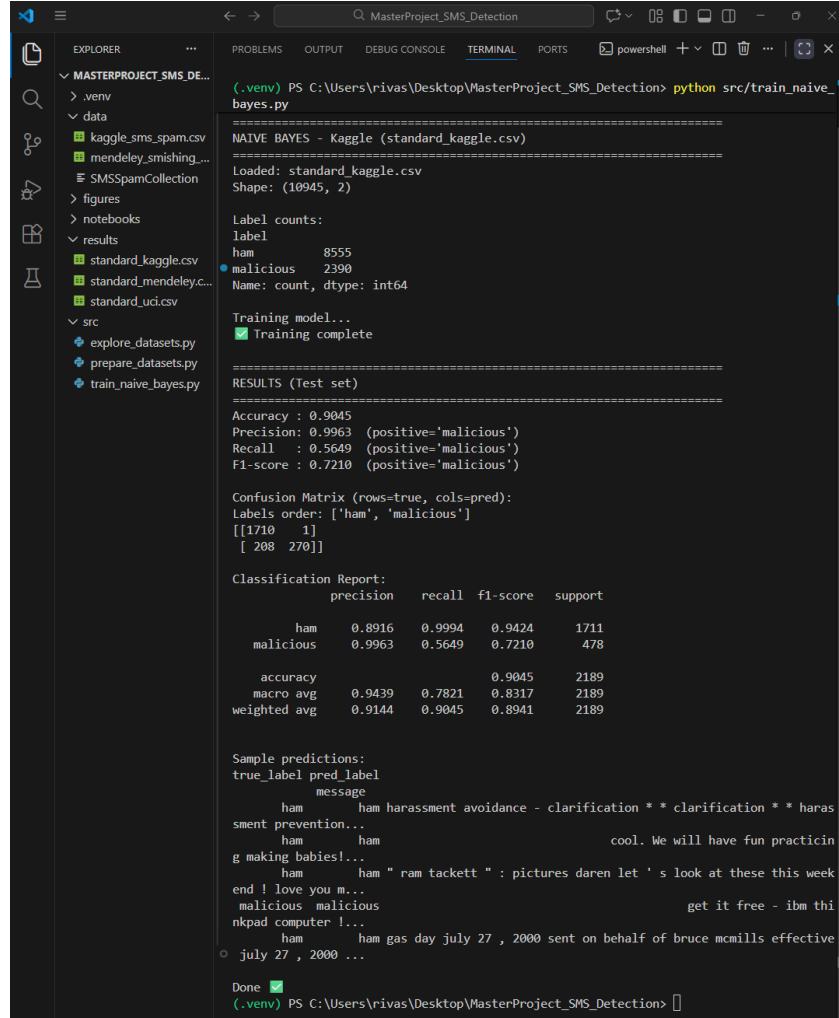
Each dataset was verified individually to confirm successful loading, correct label interpretation, and structural consistency prior to preprocessing. No datasets were merged; instead, each dataset was processed and evaluated independently to ensure a fair comparison across different data distributions and message characteristics.

For experimental evaluation, a fixed train–test split of 80/20 was applied consistently across all datasets. Approximately 80% of the messages in each dataset were used for model training, while the remaining 20% were reserved for testing on unseen data. This approach ensured that performance results reflected the models’ ability to generalise beyond the training data.

Screenshots captured during dataset loading and verification are provided in the appendices to document successful execution and to support the practical implementation of the experimental setup.

### 5.3 Feature Extraction Implementation (TF-IDF)

The TF-IDF feature extraction process was executed as part of the training pipeline using scikit-learn's TfidfVectorizer. Figure 3 demonstrates the successful transformation of SMS text into numerical feature vectors and the subsequent training and evaluation of the Naive Bayes classifier on the Kaggle dataset.



```
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection> python src/train_naive_bayes.py

=====
NAIVE BAYES - Kaggle (standard_kaggle.csv)
=====
Loaded: standard_kaggle.csv
Shape: (18945, 2)

Label counts:
Label
ham      8555
malicious 2390
Name: count, dtype: int64

Training model...
[✓] Training complete

=====
RESULTS (Test set)
=====
Accuracy : 0.9045
Precision: 0.9963 (positive='malicious')
Recall   : 0.5649 (positive='malicious')
F1-score : 0.7210 (positive='malicious')

Confusion Matrix (rows=true, cols=pred):
Labels order: ['ham', 'malicious']
[[1710   1]
 [ 208 270]]

Classification Report:

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| ham          | 0.8916    | 0.9994 | 0.9424   | 1711    |
| malicious    | 0.9963    | 0.5649 | 0.7210   | 478     |
| accuracy     |           |        | 0.9045   | 2189    |
| macro avg    | 0.9439    | 0.7821 | 0.8317   | 2189    |
| weighted avg | 0.9144    | 0.9045 | 0.8941   | 2189    |

```
Sample predictions:
true_label pred_label
message
ham ham harassment avoidance - clarification * * clarification * * harassment prevention...
ham ham cool. We will have fun practicing making babies!...
ham ham "ram tackett" : pictures daren let 's look at these this week end ! love you m...
malicious malicious get it free - ibm thinkpad computer !...
ham ham gas day july 27 , 2000 sent on behalf of bruce mcmills effective
o july 27 , 2000 ...

Done [✓]
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection>
```

Figure 3: Naive Bayes classification results using TF-IDF features on the Kaggle SMS dataset.

### 5.4 Model Implementation and Training

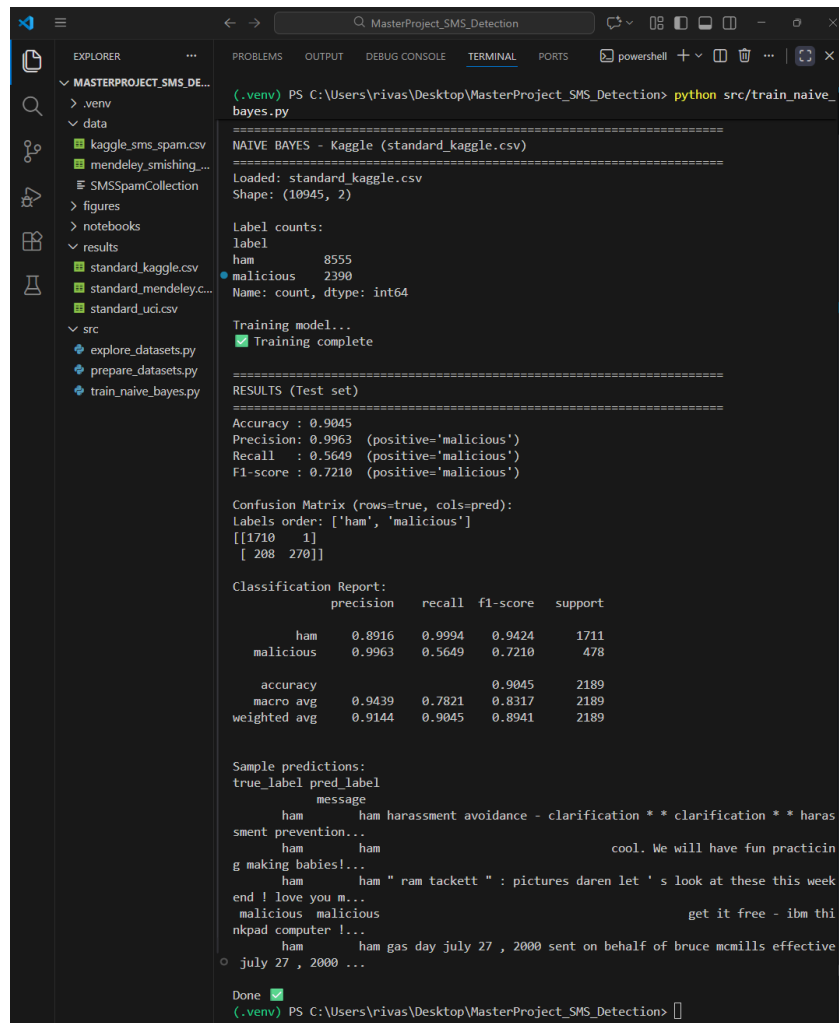
Two supervised machine learning models were implemented and evaluated in this study: Naive Bayes and Random Forest. Both models were trained using the TF-IDF feature vectors derived from each dataset.

The same split strategy described in Section 5.2 was applied for model training and evaluation.

#### 5.4.1 Naive Bayes Implementation

The Naive Bayes classifier was implemented as the primary model due to its suitability for text classification and its low computational cost. In this project, the Multinomial Naive Bayes algorithm was trained using TF-IDF feature vectors generated from the pre-processed SMS messages.

The dataset was divided into training and testing subsets using a stratified train-test split to preserve class distribution. Model training was performed locally, and evaluation was conducted on unseen test data. Performance was measured using accuracy, precision, recall, F1-score, and a confusion matrix.



```
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection> python src/train_naive_bayes.py

=====
NAIVE BAYES - Kaggle (standard_kaggle.csv)
=====
Loaded: standard_kaggle.csv
Shape: (10945, 2)

Label counts:
label
ham      8555
malicious 2390
Name: count, dtype: int64

Training model...
[✓] Training complete

=====
RESULTS (Test set)
=====
Accuracy : 0.9045
Precision: 0.9963 (positive='malicious')
Recall   : 0.5649 (positive='malicious')
F1-score : 0.7210 (positive='malicious')

Confusion Matrix (rows=true, cols=pred):
Labels order: ['ham', 'malicious']
[[1710  1]
 [ 208 270]]

Classification Report:
      precision    recall  f1-score   support

ham      0.8916      0.9994      0.9424        1711
malicious 0.9963      0.5649      0.7210         478

accuracy      0.9045      0.9045      0.9045        2189
macro avg      0.9439      0.7821      0.8317        2189
weighted avg      0.9144      0.9045      0.8941        2189

Sample predictions:
true_label pred_label
message
ham      ham harassment avoidance - clarification * * clarification * * harass
sment prevention...
ham      ham      cool. We will have fun practicing
g making babies!...
ham      ham " ram tackett " : pictures daren let ' s look at these this week
end ! love you m...
malicious malicious      get it free - ibm thi
nkpap computer !...
ham      ham gas day july 27 , 2000 sent on behalf of bruce mcmills effective
o july 27 , 2000 ...

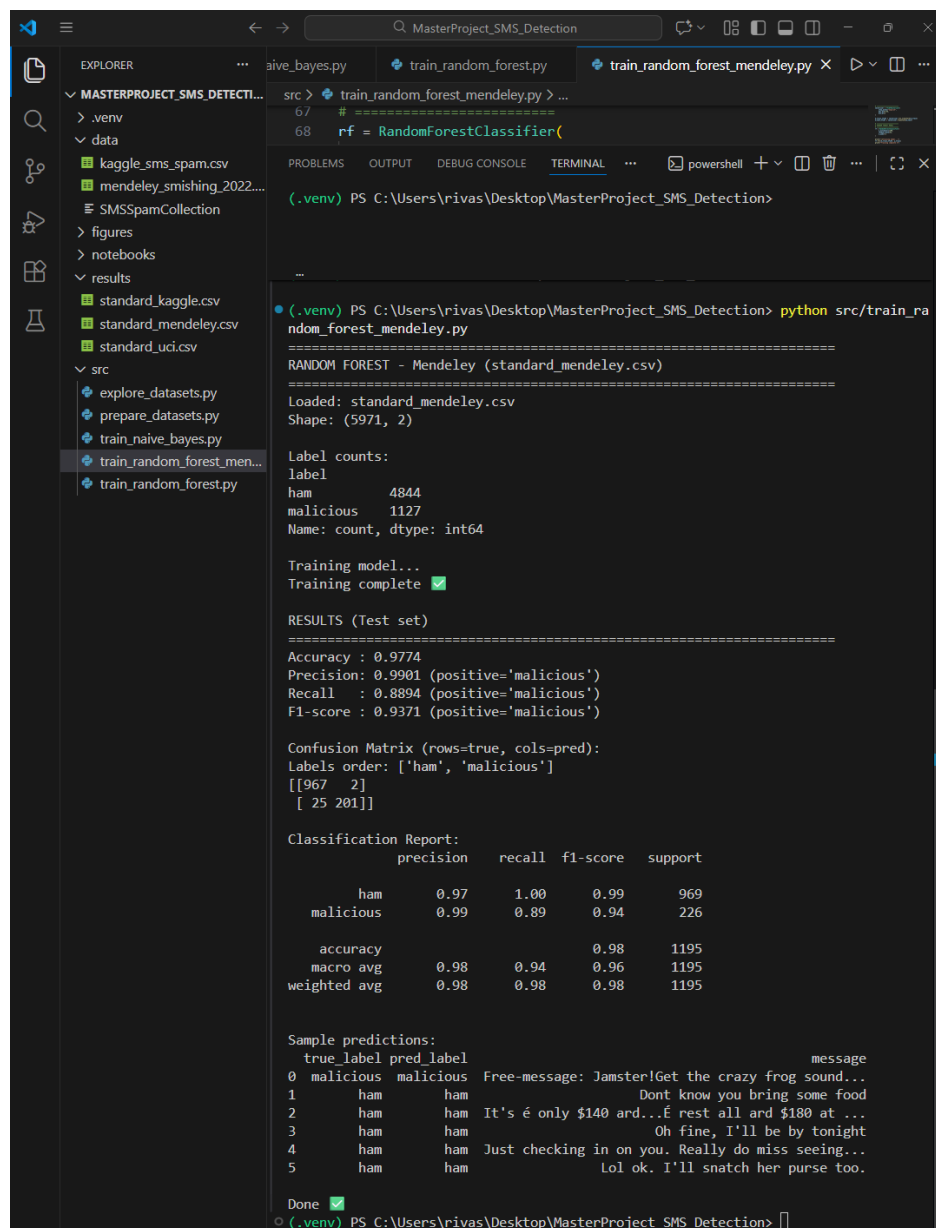
Done [✓]
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection>
```

Figure 4: Naive Bayes model training and evaluation results obtained using TF-IDF features on the Kaggle SMS dataset.

## 5.4.2 Random Forest Implementation

The Random Forest classifier was implemented as a comparative model to evaluate whether an ensemble-based approach could improve classification performance. The model was trained using the same TF-IDF feature representation and dataset splits applied to the Naive Bayes classifier, ensuring consistency across experiments.

Although Random Forest achieved competitive performance, it required higher computational resources and longer execution time compared to Naive Bayes, particularly when applied to larger datasets.



```
src > train_random_forest_mendeley.py > ...
67 # =====
68 rf = RandomForestClassifier(

(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection>

...

(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection> python src/train_random_forest_mendeley.py
=====
RANDOM FOREST - Mendeley (standard_mendeley.csv)
=====
Loaded: standard_mendeley.csv
Shape: (5971, 2)

Label counts:
label
ham      4844
malicious 1127
Name: count, dtype: int64

Training model...
Training complete ✓

RESULTS (Test set)
=====
Accuracy : 0.9774
Precision: 0.9901 (positive='malicious')
Recall   : 0.8894 (positive='malicious')
F1-score : 0.9371 (positive='malicious')

Confusion Matrix (rows=true, cols=pred):
Labels order: ['ham', 'malicious']
[[967  2]
 [ 25 201]]

Classification Report:
              precision    recall  f1-score   support

      ham           0.97         1.00         0.99         969
    malicious       0.99         0.89         0.94         226

   accuracy              0.98              0.98         1195
  macro avg           0.98         0.94         0.96         1195
 weighted avg           0.98         0.98         0.98         1195

Sample predictions:
true_label pred_label message
0  malicious  malicious  Free-message: Jamster!Get the crazy frog sound...
1      ham      ham      Dont know you bring some food
2      ham      ham  It's é only $140 ard...É rest all ard $180 at ...
3      ham      ham      Oh fine, I'll be by tonight
4      ham      ham  Just checking in on you. Really do miss seeing...
5      ham      ham      Lol ok. I'll snatch her purse too.

Done ✓
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection> 
```

Figure 5 Random Forest classification results using TF-IDF features on the Kaggle SMS dataset.

## **5.5 Model Evaluation and Comparative Analysis**

Model evaluation was conducted using standard classification metrics, including accuracy, precision, recall, and F1-score, in order to assess and compare the performance of the implemented machine learning models. Both Naive Bayes and Random Forest were evaluated across the same datasets using identical preprocessing steps and TF-IDF feature representations, ensuring a fair and consistent comparison.

The results indicate that both models achieved strong overall performance in SMS spam and smishing detection tasks. Naive Bayes demonstrated consistently high precision across datasets, suggesting a low false-positive rate and reliable identification of legitimate messages. In addition, the model showed fast execution times and low computational overhead, reinforcing its suitability for lightweight deployment scenarios.

Random Forest, on the other hand, achieved higher recall and F1-score values across most datasets, particularly on the Mendeley Smishing 2022 dataset. This indicates a stronger ability to detect a higher proportion of malicious messages, including more complex or socially engineered smishing patterns. However, this improved detection capability came at the cost of increased computational complexity and longer training time.

Overall, the comparative evaluation highlights a trade-off between model simplicity and detection capability. While Random Forest provided more balanced performance in terms of recall and F1-score, Naive Bayes offered a more efficient and practical solution within the scope of this project. These findings directly informed the selection of Naive Bayes as the primary model for the proof-of-concept SMS classification application.

## **5.6 Proof-of-Concept SMS Classification Application**

To demonstrate the practical applicability of the proposed SMS spam and smishing detection pipeline, a lightweight proof-of-concept application was developed using Streamlit. The primary purpose of this application is to validate the trained machine learning model by enabling classification of previously unseen SMS messages through a simple and interactive interface.

The application allows users to manually enter an SMS message via a text input field. Once submitted, the message undergoes the same preprocessing and TF-IDF feature extraction pipeline used during model training. The processed input is then classified using the trained

Naive Bayes model, and the predicted label is displayed as either legitimate (ham) or malicious (spam/smishing).

The interface was intentionally designed to remain minimal, focusing on functional validation rather than user interface complexity. This design choice aligns with the scope of the project, which aims to demonstrate the feasibility and end-to-end functionality of the proposed approach rather than deliver a fully deployed system. The application also highlights the potential for future extension into a more comprehensive web or mobile-based solution.

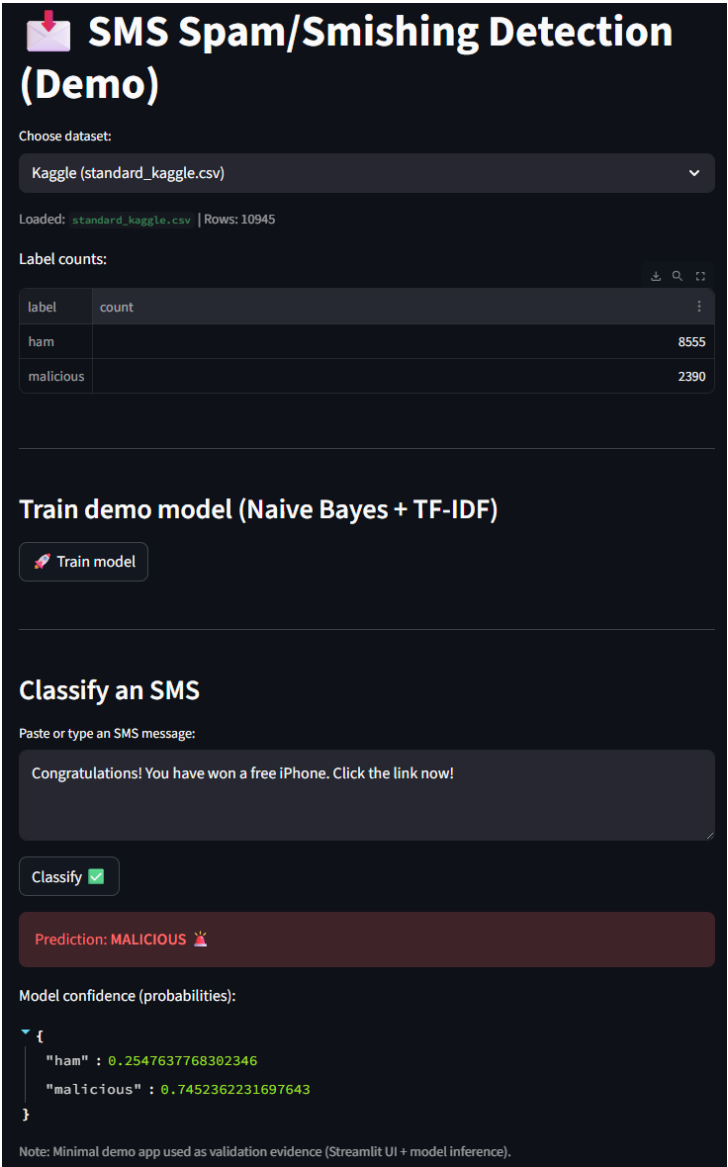


Figure 6 Streamlit-based proof-of-concept SMS spam/smishing classification interface

## 5.7 Implementation Challenges and Solutions

Several practical challenges were encountered during the implementation of the SMS spam and smishing detection experiments. One of the primary issues involved dataset inconsistencies across the three selected sources. The datasets differed in file formats, label naming conventions, and structural organisation. For example, the UCI SMS Spam Collection dataset was provided without a standard file extension, which initially caused file-loading errors. This issue was resolved by explicitly specifying the file format during data ingestion and standardising label representations across all datasets.

Another challenge related to dataset harmonisation and preprocessing. The datasets varied in size and message distribution, requiring careful verification to ensure that preprocessing steps such as text cleaning and TF-IDF feature extraction were applied consistently. This was addressed by implementing a unified preprocessing pipeline that ensured comparable feature representations across all experiments.

During model training, additional challenges were observed when comparing Naive Bayes and Random Forest classifiers. While Naive Bayes trained efficiently across all datasets, Random Forest required increased computational resources and longer execution times, particularly for larger datasets. This behaviour reinforced the importance of considering computational efficiency alongside classification performance when evaluating model suitability.

All identified issues were addressed through systematic troubleshooting, including incremental testing, dataset validation, and controlled experimentation. These challenges provided valuable insight into real-world machine learning implementation and highlighted the importance of robust preprocessing, model selection, and experimental consistency.

## CHAPTER 6: EVALUATION OF RESULTS

### 6.1 Model Performance Evaluation

This section evaluates the performance of the machine learning models implemented in this project using standard classification metrics. The Naive Bayes and Random Forest models are assessed across three publicly available SMS datasets, with a specific focus on their ability to detect malicious messages relevant to smishing attacks.

Together, these evaluations provide a comprehensive understanding of how both models perform across datasets with different characteristics, forming the basis for the comparative analysis presented in the next section.

All results reported in this chapter were obtained on the held-out test subset (20%) for each dataset, ensuring evaluation on previously unseen messages.

#### 6.1.1 Evaluation of Naive Bayes Model Performance

The Naive Bayes classifier combined with TF-IDF feature extraction was evaluated across three publicly available SMS datasets: Kaggle SMS Spam, Mendeley Smishing 2022, and the UCI SMS Spam Collection. The evaluation focused on standard classification metrics including accuracy, precision, recall, and F1-score, with particular attention given to the malicious class due to its relevance for smishing detection.

*Table 1 Summarises the performance of the Naive Bayes model across all datasets.*

| <b>Dataset</b>                     | <b>Accuracy</b> | <b>Precision<br/>(Malicious)</b> | <b>Recall<br/>(Malicious)</b> | <b>F1-score<br/>(Malicious)</b> |
|------------------------------------|-----------------|----------------------------------|-------------------------------|---------------------------------|
| <b>Kaggle SMS Spam</b>             | 0.9045          | 0.9963                           | 0.5649                        | 0.7210                          |
| <b>Mendeley Smishing<br/>2022</b>  | 0.9699          | 0.9948                           | 0.8451                        | 0.9139                          |
| <b>UCI SMS Spam<br/>Collection</b> | 0.9695          | 0.9915                           | 0.7785                        | 0.8722                          |

Across all datasets, the model achieved consistently high overall accuracy, ranging from 90.45% on the Kaggle dataset to approximately 97% on both the Mendeley and UCI datasets.

This indicates that the Naive Bayes approach is effective for general SMS classification tasks, even when applied to datasets with different characteristics and levels of message complexity.

A notable observation is the consistently high precision for the malicious class across all datasets, exceeding 99% in each case. This suggests that when the model identifies a message as malicious, it is very likely to be correct.

The F1-score further highlights these differences. The Mendeley dataset achieved the highest F1-score (0.9139), indicating a strong balance between precision and recall. This suggests that Naive Bayes performs best when trained on datasets that explicitly contain modern smishing patterns.

Overall, these results demonstrate that Naive Bayes provides a robust and computationally efficient baseline model for SMS spam and smishing detection.

Despite some limitations in recall on older datasets, its strong precision and consistent performance justified its selection as the primary model for the demonstration application implemented in this project.

### **6.1.2 Evaluation of Random Forest Model Performance**

The Random Forest classifier was evaluated across the same three publicly available SMS datasets used for the Naive Bayes model: Kaggle SMS Spam, Mendeley Smishing 2022, and the UCI SMS Spam Collection. This ensured a fair and consistent comparison between both machine learning approaches. The evaluation focused on accuracy, precision, recall, and F1-score for the malicious class, which is the most critical category for smishing detection.

Across all datasets, Random Forest demonstrated consistently high overall performance, achieving accuracy values above 94% in all cases. The highest accuracy was observed on the Mendeley Smishing 2022 dataset (97.74%), followed closely by the UCI dataset (97.66%). This indicates that Random Forest performs particularly well when trained on datasets that contain more recent and security-focused smishing messages.

A key strength of the Random Forest model was its strong balance between precision and recall for the malicious class. On the Kaggle dataset, the model achieved a precision of 95.98% and a recall of 79.92%, resulting in an F1-score of 0.8721. While the recall is lower than precision, this result still represents a significant improvement over Naive Bayes on the

same dataset, indicating that Random Forest is better at identifying malicious messages that may be more subtle or diverse in structure.

The Mendeley Smishing 2022 dataset produced the strongest results overall. The model achieved a precision of 99.01% and a recall of 88.94%, leading to the highest F1-score (0.9371) across all experiments. This suggests that Random Forest is particularly effective when applied to datasets specifically curated for smishing detection, where message patterns are more aligned with real-world attack scenarios.

On the UCI SMS Spam Collection dataset, Random Forest achieved perfect precision (100%) for the malicious class, meaning that all messages classified as malicious were correctly identified. However, recall was slightly lower (82.55%), indicating that some malicious messages were still misclassified as benign. Despite this, the overall F1-score of 0.9044 confirms strong classification performance and highlights the robustness of the model even when applied to older datasets.

Overall, these results demonstrate that Random Forest provides strong and reliable performance across different datasets, particularly excelling in scenarios where higher recall and balanced classification are required. However, this improved performance comes at the cost of increased computational complexity compared to simpler models such as Naive Bayes.

*Table 2 Performance of Random Forest Model Across Datasets*

| <b>Dataset</b>                     | <b>Accuracy</b> | <b>Precision<br/>(Malicious)</b> | <b>Recall<br/>(Malicious)</b> | <b>F1-score<br/>(Malicious)</b> |
|------------------------------------|-----------------|----------------------------------|-------------------------------|---------------------------------|
| <b>Kaggle SMS Spam</b>             | 0.9488          | 0.9598                           | 0.7992                        | 0.8721                          |
| <b>Mendeley Smishing<br/>2022</b>  | 0.9774          | 0.9901                           | 0.8894                        | 0.9371                          |
| <b>UCI SMS Spam<br/>Collection</b> | 0.9766          | 1.0000                           | 0.8255                        | 0.9044                          |

## 6.2 Comparative Evaluation of Naïve Bayes and Random Forest

This section compares the performance of the Naive Bayes and Random Forest models to determine which approach is more suitable for SMS spam and smishing detection within the scope of this project.

When comparing overall accuracy, both models achieved strong results across all datasets. However, Random Forest consistently outperformed Naive Bayes, particularly on the Mendeley Smishing 2022 dataset, where it achieved higher accuracy, recall, and F1-score. This suggests that Random Forest is more effective at capturing complex patterns present in modern smishing messages.

Precision for the malicious class was high for both models, often exceeding 99%. This indicates that both approaches are reliable in minimising false positives, which is critical in security-related applications where incorrectly flagging legitimate messages can negatively impact user trust. Nevertheless, Random Forest demonstrated slightly stronger precision consistency across datasets.

The most notable difference between the two models lies in recall. Naive Bayes showed lower recall on the Kaggle dataset, indicating that a larger proportion of malicious messages were missed. In contrast, Random Forest achieved substantially higher recall values across all datasets, making it more suitable for applications where detecting as many malicious messages as possible is a priority.

Despite the superior performance of Random Forest, Naive Bayes was selected as the primary model for the demonstration application. This decision was based on its lower computational cost, faster training time, and ease of deployment, while still providing strong and reliable performance. Random Forest was therefore used as a comparative benchmark to validate the robustness of the chosen approach rather than as the core deployment model.

*Table 3 Model Comparison Summary*

| <b>Dataset</b>  | <b>Model</b>  | <b>Accuracy</b> | <b>Precision</b> | <b>Recall</b> | <b>F1-Score</b> |
|-----------------|---------------|-----------------|------------------|---------------|-----------------|
| <b>Kaggle</b>   | Naive Bayes   | 0.9045          | 0.9963           | 0.5649        | 0.7210          |
| <b>Kaggle</b>   | Random Forest | 0.9488          | 0.9598           | 0.7992        | 0.8721          |
| <b>Mendeley</b> | Naive Bayes   | 0.9699          | 0.9948           | 0.8451        | 0.9139          |
| <b>Mendeley</b> | Random Forest | 0.9774          | 0.9901           | 0.8894        | 0.9371          |
| <b>UCI</b>      | Naive Bayes   | 0.9695          | 0.9915           | 0.7785        | 0.8722          |
| <b>UCI</b>      | Random Forest | 0.9766          | 1.0000           | 0.8255        | 0.9044          |

### **6.3 Comparison with Existing Studies**

The results obtained in this project are broadly consistent with findings reported in existing literature on SMS spam and smishing detection.

Prior studies frequently report that lightweight probabilistic models such as Naive Bayes provide strong baseline performance for SMS classification when combined with sparse text representations like TF-IDF, while ensemble-based approaches such as Random Forest often achieve improved recall by capturing more complex feature interactions [5], [6].

In this study, Random Forest demonstrated superior recall and F1-score across all three datasets, especially on the Mendeley Smishing 2022 dataset, which contains more recent and security-focused smishing messages. This aligns with prior research indicating that ensemble methods are better suited for capturing non-linear relationships and subtle feature interactions in textual data.

Conversely, Naive Bayes achieved consistently high precision across all datasets, a result also commonly observed in related work. This confirms its effectiveness as a strong baseline

classifier for text classification tasks, particularly when computational efficiency and simplicity are prioritised.

Overall, the alignment between the results of this project and previous machine learning studies reinforces the validity of the experimental approach and supports the reliability of the findings presented in this chapter.

#### **6.4 Critical Discussion of Results**

The evaluation results highlight several important insights regarding the performance of machine learning models for SMS spam and smishing detection. While both Naive Bayes and Random Forest achieved high accuracy values across all datasets, accuracy alone does not fully capture the effectiveness of a model in security-related contexts.

A key observation is the trade-off between precision and recall. Naive Bayes consistently achieved very high precision for the malicious class, indicating that when a message is classified as malicious, it is highly likely to be correct. However, its lower recall on the Kaggle dataset suggests that a proportion of malicious messages were misclassified as benign. In real-world scenarios, this could result in potentially harmful messages bypassing detection.

In contrast, Random Forest demonstrated a more balanced performance, particularly in terms of recall and F1-score. This indicates a stronger ability to correctly identify a higher proportion of malicious messages, which is especially important in smishing detection where missing an attack may have serious consequences. The improved recall observed across all datasets suggests that Random Forest is more effective at generalising across diverse message patterns.

These findings demonstrate that model selection should be guided by the specific objectives of the application. If minimising false positives is the priority, Naive Bayes remains a strong candidate. However, if detecting as many malicious messages as possible is the primary concern, Random Forest offers a more suitable solution despite its higher computational cost.

## **6.5 Strengths of the Experimental Setup**

One of the main strengths of the experimental setup in this project is the use of multiple publicly available datasets with differing characteristics. By evaluating both models on the Kaggle SMS Spam dataset, the UCI SMS Spam Collection, and the Mendeley Smishing 2022 dataset, the study ensures a broader and more reliable assessment of model performance.

Additionally, the use of standard evaluation metrics, including accuracy, precision, recall, and F1-score, provides a comprehensive and well-established framework for performance comparison. The consistent application of TF-IDF feature extraction across all experiments further supports a fair and controlled comparison between the two models.

Finally, implementing both models using the same preprocessing pipeline and evaluation methodology enhances the reproducibility and transparency of the experimental results.

## **6.6 Limitations and Threats to Validity**

Despite the positive results achieved, this study has several limitations that should be acknowledged. Firstly, although three datasets were used, two of them (Kaggle and UCI) contain relatively older SMS messages, which may not fully reflect modern smishing techniques that increasingly incorporate personalised language, URLs, and social engineering strategies.

Secondly, the models were evaluated using a fixed train-test split rather than cross-validation. While this approach is sufficient for comparative analysis, it may limit the generalisability of the results across different data distributions.

Another limitation relates to the use of traditional machine learning models. While Naive Bayes and Random Forest provide strong baseline and ensemble performance, more advanced deep learning approaches may achieve improved results at the cost of higher computational complexity.

These limitations do not invalidate the findings but highlight areas for future improvement and extension. Addressing these factors in future work could further strengthen the robustness and applicability of SMS spam and smishing detection systems.

## **CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS**

### **7.1 Conclusions**

This project investigated the effectiveness of lightweight machine learning techniques for detecting SMS spam and smishing attacks, with a focus on practicality, reproducibility, and real-world applicability. The research aim and objectives outlined in Chapter 1 were successfully addressed through a structured experimental methodology, supported by a comprehensive literature review and systematic model evaluation.

The results demonstrate that traditional supervised machine learning models, when combined with TF-IDF feature extraction, can achieve strong performance in SMS spam and smishing detection tasks. Both Naive Bayes and Random Forest produced high accuracy and reliable classification results across multiple publicly available datasets. While Random Forest generally achieved higher recall and F1-score values, particularly on the Mendeley Smishing 2022 dataset, Naive Bayes consistently delivered very high precision and stable performance.

Overall, the findings confirm that lightweight machine learning approaches remain effective and suitable for SMS spam and smishing detection, particularly in environments where computational efficiency and ease of deployment are important considerations.

### **7.2 Contributions of the Study**

This project makes several practical and academic contributions:

- It provides a structured and reproducible experimental framework for evaluating SMS spam and smishing detection models using multiple datasets.
- It offers a comparative analysis of Naive Bayes and Random Forest models under identical preprocessing and evaluation conditions.
- It demonstrates the practical applicability of machine learning models through a conceptual SMS classification application.
- It reinforces existing research findings while providing updated experimental evidence using a modern smishing dataset.

These contributions enhance the broader understanding of how traditional machine learning techniques can be effectively applied to cybersecurity problems involving text-based attacks.

### **7.3 Recommendations**

Based on the experimental findings presented in Chapter 6, model selection for SMS spam and smishing detection should be guided by the intended deployment context and operational constraints. While both evaluated models demonstrated strong classification performance, clear trade-offs were observed between computational efficiency and detection capability. These trade-offs are particularly relevant in real-world environments where system resources, scalability, and response time must be considered alongside detection accuracy.

For lightweight or resource-constrained environments, Naive Bayes remains a strong and reliable choice due to its simplicity, fast training and inference times, and consistently high precision. Its low computational overhead makes it suitable for practical deployment scenarios where efficiency and ease of integration are prioritised.

In scenarios where detecting as many malicious messages as possible is critical, Random Forest is recommended due to its superior recall and more balanced performance across evaluation metrics. This makes it more appropriate for security-sensitive contexts where minimising missed smishing attempts is a primary concern.

Future implementations of SMS filtering systems may also benefit from combining precision-oriented and recall-oriented models, depending on the operational context and risk tolerance. Such hybrid or layered approaches could further enhance detection effectiveness while maintaining acceptable performance and resource usage.

### **7.4 Suggestions for Further Work**

While this project achieved its stated aims, several opportunities for future research and improvement have been identified:

- The use of cross-validation techniques instead of a fixed train-test split could improve the robustness and generalisability of evaluation results.
- Incorporating more recent and larger-scale smishing datasets could better reflect evolving real-world attack patterns.
- Future work could explore deep learning approaches, such as recurrent or transformer-based models, to assess potential performance improvements.

- The conceptual SMS classification application could be extended into a fully deployed mobile or web-based system, allowing real-time message classification and user feedback.
- Additional features, such as URL analysis or sender behaviour profiling, could be integrated to enhance detection accuracy.

Addressing these areas would further strengthen the effectiveness and applicability of SMS spam and smishing detection systems.

### **7.5 Final Remarks**

In conclusion, this project demonstrates that well-designed and carefully evaluated machine learning solutions can play a significant role in mitigating SMS-based cyber threats. By balancing performance, simplicity, and practicality, the proposed approach provides a solid foundation for both academic research and real-world security implementations.

## **CHAPTER 8: REFLECTION**

This project provided a valuable opportunity to reflect not only on the technical aspects of machine learning and cybersecurity, but also on my personal learning process, time management, and academic development throughout the MSc programme.

### **8.1 Time Management and Planning**

At the beginning of the project, managing time effectively was one of the main challenges. Balancing full-time work commitments with academic deadlines required careful planning and prioritisation. Initially, I underestimated the amount of time required for experimentation, troubleshooting, and refining written sections. As the project progressed, I learned to break tasks into smaller, manageable milestones and to allocate dedicated time slots for research, implementation, and writing.

A key lesson was the importance of documenting implementation evidence, such as code outputs, screenshots, and execution results, during experimentation rather than attempting to collect them retrospectively close to submission.

This experience improved my ability to plan realistically and highlighted the importance of starting practical work early to avoid unnecessary pressure close to submission deadlines.

### **8.2 Development of Study and Research Skills**

Throughout the project, my study skills improved significantly, particularly in relation to independent research and critical evaluation of academic literature. I became more confident in identifying relevant sources, comparing methodologies across studies, and linking theoretical findings to my own experimental results.

I also strengthened my ability to interpret classification metrics beyond overall accuracy, particularly precision, recall, and F1-score, when analysing imbalanced datasets in a security context.

The requirement to justify design choices and interpret results critically strengthened my analytical thinking and helped me move beyond simply “implementing a solution” towards understanding *why* certain approaches perform better in specific contexts.

### **8.3 Learning About My Own Learning Style**

This project helped me understand that I learn most effectively through a combination of structured guidance and hands-on experimentation. While reviewing academic papers

provided essential theoretical grounding, the greatest learning occurred during practical implementation, testing, and debugging.

Encountering unexpected errors and performance variations forced me to engage more deeply with the data and models, reinforcing concepts that would not have been fully understood through reading alone. This insight will influence how I approach future technical projects, ensuring a stronger balance between theory and practice.

#### **8.4 Areas for Improvement**

In retrospect, there are several aspects of the project that could have been improved. Future iterations of this work could incorporate k-fold cross-validation and light hyperparameter tuning (for example, adjusting the alpha parameter in Naive Bayes or tree depth in Random Forest) to improve robustness while maintaining the lightweight focus of the project. Earlier integration of the conceptual application design might also have allowed for a more polished demonstration component.

Additionally, maintaining more detailed progress notes throughout the project would have further supported reflection and documentation, particularly during intensive implementation phases.

#### **8.5 Reflection on Overall Learning**

Overall, this project was a challenging but rewarding learning experience. It reinforced the importance of adaptability, perseverance, and reflective thinking in both academic and professional contexts. The process demonstrated that effective cybersecurity solutions require not only technical knowledge, but also careful planning, critical evaluation, and continuous self-improvement.

This reflective experience has increased my confidence in managing complex, independent projects and has provided valuable lessons that will directly inform my future academic work and professional development.

## REFERENCES:

- [1] S. Kotecha and G. Lambert, "Text scammers make tens of thousands a month – and spend it on designer shoes and bags," *BBC News*, Nov. 29, 2025. [Online]. Available: <https://www.bbc.co.uk/news/articles/cg4nzp6ezg7o>. [Accessed: Jan. 30, 2026].
- [2] C. Edwards, "Watchdog proposes new rules to increase blocking of scam texts," *BBC News*, Oct. 29, 2025. [Online]. Available: <https://www.bbc.co.uk/news/articles/c6e4x55s29o>. [Accessed: Jan. 30, 2026].
- [3] K. Peachey, "Parcel delivery texts now the most common con-trick," *BBC News*, Aug. 16, 2021. [Online]. Available: <https://www.bbc.co.uk/news/business-58323743>. [Accessed: Jan. 30, 2026].
- [4] M. R. Al Saidat, S. Y. Yerima, and K. Shaalan, "Advancements of SMS Spam Detection: A Comprehensive Survey of NLP and ML Techniques," *Procedia Computer Science*, vol. 244, pp. 248–259, 2024, doi: 10.1016/j.procs.2024.10.198.
- [5] H. Jain and M. Rokade, "An Analysis of SMS Spam Detection using Machine Learning Model," in *Proc. 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, IEEE, 2022, pp. 151–156. [Online]. Available: <https://ieeexplore.ieee.org/document/9913532>. [Accessed: Jan. 30, 2026].
- [6] D. A. Anggraini, M. Ihksan, and Suhardi, "Implementation of the Naïve Bayes Algorithm in the SMS Spam Filtering System," *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 6, no. 2, pp. 840–848, Apr. 2024, doi: 10.47709/cnahpc.v6i2.3875.
- [7] J. C. Obi, "A Comparative Study of Several Classification Metrics and Their Performances on Data," *World Journal of Advanced Engineering Technology and Sciences*, vol. 8, no. 1, pp. 308–314, 2023, doi: 10.30574/wjaets.2023.8.1.0054.
- [8] A. I. Kadhim, "Survey on supervised machine learning techniques for automatic text classification," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 273–292, 2019, doi: 10.1007/s10462-018-09677-1.
- [9] N. J. Saputra, "Analysis of SMS Spam Detection Using TF-IDF: A Study on SMS Spam Collection Dataset," *Jurnal Sosial dan Teknologi (SOSTECH)*, vol. 4, no. 4, pp. 214–217, Apr. 2024.

- [10] T. Almeida and J. Hidalgo, "SMS Spam Collection," UCI Machine Learning Repository, 2012. doi: 10.24432/C5CC84. [Online]. Available: <https://archive.ics.uci.edu/dataset/228/sms+spam+collection>. [Accessed: Feb. 03, 2026].
- [11] tinu10kumar, "SMS SPAM DATASET (10286 rows)," Kaggle Datasets. [Online]. Available: <https://www.kaggle.com/datasets/tinu10kumar/sms-spam-dataset> [Accessed: Feb. 03, 2026].
- [12] S. Mishra and D. Soni, "SMS Phishing Dataset for Machine Learning and Pattern Recognition," Mendeley Data, vol. 1, Jun. 20, 2022, doi: 10.17632/f45bkkt8pr.1. [Online]. Available: <https://data.mendeley.com/datasets/f45bkkt8pr/1>. [Accessed: Feb. 03, 2026]

## APPENDICES

### Appendix A – Code Snippets

#### Appendix A.1 – Text Preprocessing and TF-IDF Feature Extraction

This appendix provides representative code snippets illustrating the preprocessing and TF-IDF feature extraction stage described in Chapter 5. These steps were applied consistently across all datasets prior to model training.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(
    lowercase=True,
    stop_words='english'
)
X_tfidf = vectorizer.fit_transform(messages)
```

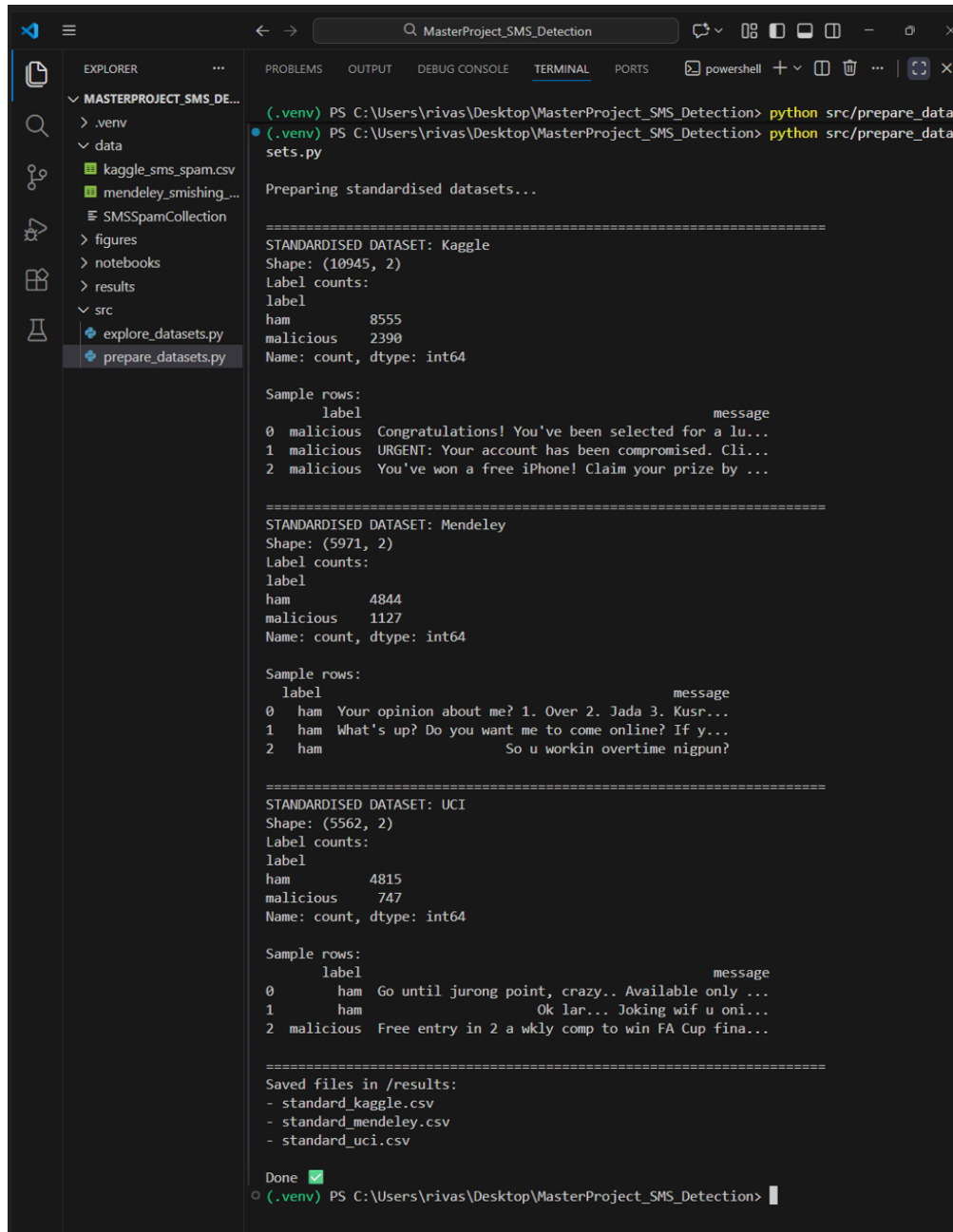
#### Appendix A.2 – Model Training and Evaluation

The following snippet illustrates the supervised training and evaluation process used for both Naive Bayes and Random Forest classifiers, as discussed in Chapter 5 and evaluated in Chapter 6.

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
X_train, X_test, y_train, y_test = train_test_split(
    X_tfidf, labels, test_size=0.2, random_state=42
)
model = MultinomialNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

## Appendix B – Dataset Standardisation and Class Harmonisation

This figure shows the preprocessing and standardisation process applied to the three datasets used in this study. Each dataset was transformed into a unified structure with consistent label naming (*ham* and *malicious*) and a common feature format. The figure also illustrates the class distributions after harmonisation and confirms that all standardised datasets were successfully saved and used in subsequent experimental evaluations.



```
MasterProject_SMS_Detection
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection> python src/prepare_data
sets.py
Preparing standardised datasets...

=====
STANDARDISED DATASET: Kaggle
Shape: (10945, 2)
Label counts:
label
ham      8555
malicious 2390
Name: count, dtype: int64

Sample rows:
label      message
0  malicious  Congratulations! You've been selected for a lu...
1  malicious  URGENT: Your account has been compromised. Cli...
2  malicious  You've won a free iPhone! Claim your prize by ...

=====
STANDARDISED DATASET: Mendeley
Shape: (5971, 2)
Label counts:
label
ham      4844
malicious 1127
Name: count, dtype: int64

Sample rows:
label      message
0  ham  Your opinion about me? 1. Over 2. Jada 3. Kusr...
1  ham  What's up? Do you want me to come online? If y...
2  ham  So u workin overtime nigpun?

=====
STANDARDISED DATASET: UCI
Shape: (5562, 2)
Label counts:
label
ham      4815
malicious 747
Name: count, dtype: int64

Sample rows:
label      message
0  ham  Go until jurong point, crazy.. Available only ...
1  ham  Ok lar... Joking wif u oni...
2  malicious  Free entry in 2 a wkly comp to win FA Cup fina...

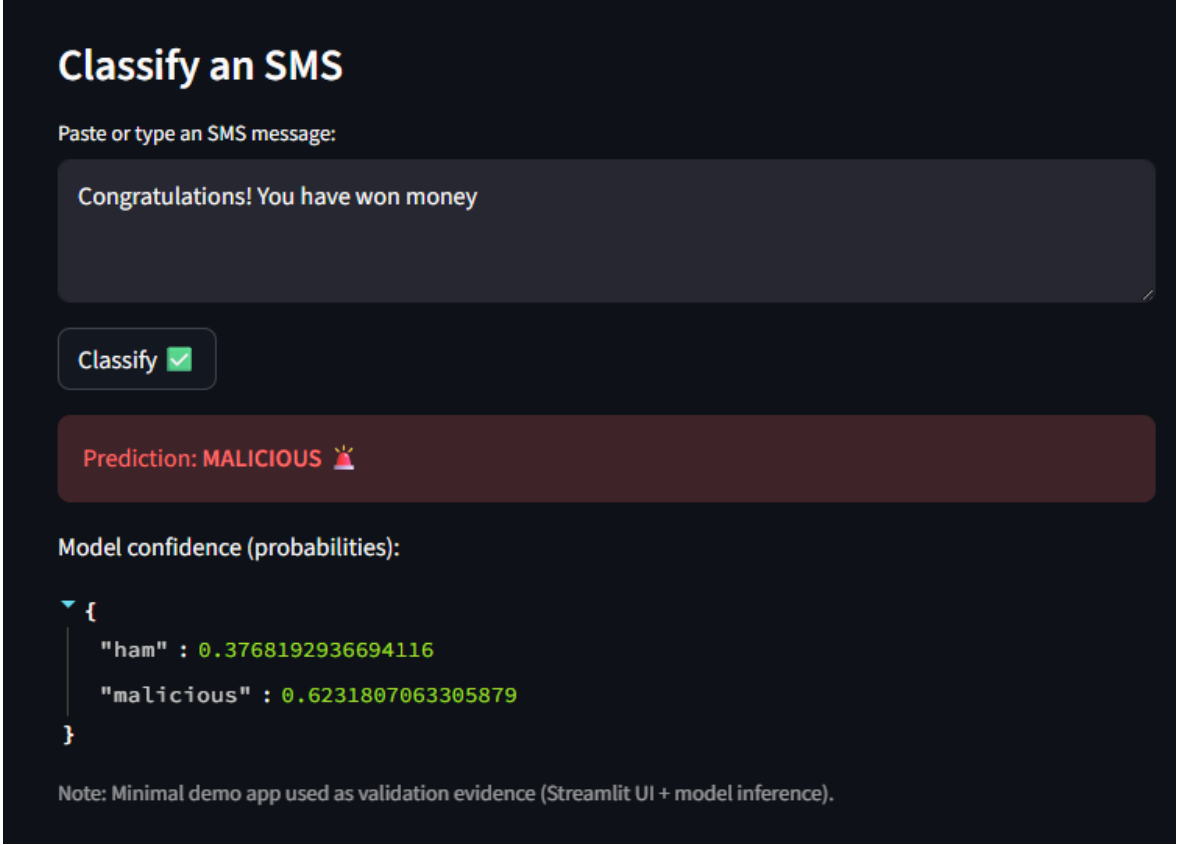
=====
Saved files in /results:
- standard_kaggle.csv
- standard_mendeley.csv
- standard_uci.csv

Done ✓
(.venv) PS C:\Users\rivas\Desktop\MasterProject_SMS_Detection>
```

Figure 7 Dataset standardisation and class harmonisation across Kaggle, Mendeley and UCI datasets.

## Appendix C – Dataset Samples

This appendix provides visual evidence of the functional validation of the proposed SMS classification system through a lightweight Streamlit-based interface. The interface allows users to input an SMS message and receive a real-time prediction (*ham* or *malicious*), along with class probability estimates. This implementation serves as proof of concept and validates the practical applicability of the trained machine learning models.



**Classify an SMS**

Paste or type an SMS message:

Congratulations! You have won money

Classify ✓

Prediction: MALICIOUS 🚨

Model confidence (probabilities):

```
{
  "ham" : 0.3768192936694116
  "malicious" : 0.6231807063305879
}
```

Note: Minimal demo app used as validation evidence (Streamlit UI + model inference).

Figure 8 SMS spam and smishing detection system validation using a Streamlit-based interface.