DEVELOPMENT OF A WEB BASED EMAIL SPAM DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

BY

ONYENA VICTOR
GOU/U22/CSC/819

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF

COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, ENUGU STATE.

JUNE, 2026.

DEVELOPMENT OF A WEB BASED EMAIL SPAM DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

BY

ONYENA VICTOR
GOU/U22/CSC/819

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, ENUGU STATE.

IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF SCIENCE (B.Sc.), DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: PROF OZOFOR

JUNE, 2026.

APPROVAL PAGE

This research, titled "**Development of a Web Based Email Spam Detection System Using Machine Learning Techniques**" has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu, Nigeria.

Prof. Ozofor	.....	.....
Supervisor	Signature	Date
External Examiner	.....	.....
	Signature	Date
Dr. Frank Okebanama	.....	.....
HOD, Dept. of Computer Science and Mathematics	Signature	Date
Prof. I. Ajah	.....	.....
Dean, Faculty of Computing and Information Technology	Signature	Date

CERTIFICATION

I certify that I completed the research included herein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

ONYENA VICTOR

GOU/U22/CSC/819

DEDICATION

This project is dedicated to Almighty God, whose wisdom and strength guided me through every step of this journey, and to my beloved family whose love, sacrifices, and encouragement made this academic pursuit possible. Your belief in me has been my greatest motivation.

ACKNOWLEDGEMENTS

I wish to express my profound gratitude to my supervisor, Prof. Ozofor, constructive criticism, patience, and intellectual support throughout the course of this research. His knowledge and commitment to excellence greatly shaped the quality of this work.

I also acknowledge the Head of Department, Dr. Frank Okebanama, and all the lecturers in the Department of Computer Science and Mathematics, Godfrey Okoye University, whose dedicated teaching over the years laid the academic foundation upon which this research is built. Special appreciation goes to every lecturer who took time to impart knowledge, mentor, and challenge me to think critically and apply what I have learnt.

My sincere gratitude goes to my parents and family members for their endless prayers, financial support, and moral encouragement. Your sacrifices and belief in my ability have been a constant source of strength. To my friends and colleagues who offered assistance, ideas, and motivation during the difficult phases of this research, I am truly grateful.

Finally, I acknowledge the contributions of all researchers and authors whose published works are cited in this project. Their intellectual contributions have been instrumental in shaping the direction and depth of this study.

ABSTRACT

The exponential growth of electronic mail as a primary mode of communication has been accompanied by a corresponding rise in unsolicited and often malicious messages, commonly referred to as spam, which compromise user productivity, consume network resources, and expose users to phishing and fraud. While traditional spam filters rely on static, rule-based keyword matching, such approaches struggle to adapt to the evolving tactics employed by spammers, resulting in high false-negative rates and reduced filtering accuracy over time. This study addresses the problem by developing spamshield, a machine learning-based email spam detection system capable of automatically and accurately classifying messages as spam or legitimate (ham). The research employed a dataset of 5,572 labelled sms/email messages obtained from the uci machine learning repository, which was preprocessed using natural language processing techniques including tokenization, stop-word removal, and porter stemming, before being transformed into numerical feature vectors using term frequency-inverse document frequency (tf-idf) vectorization. Three supervised classification algorithms, namely multinomial naïve bayes, logistic regression, and a calibrated support vector classifier, were trained and evaluated on the preprocessed dataset, with performance assessed using accuracy, precision, recall, f1-score, and roc-auc metrics. The trained model was subsequently deployed as a web-based application using a flask rest api backend and an interactive html/css/javascript frontend, allowing users to submit messages for real-time classification. Results obtained from the evaluation showed that all three models achieved strong classification performance, with [insert your best model name] recording the highest accuracy of [insert %] and an f1-score of [insert %], outperforming the other models particularly in correctly identifying spam messages despite the class imbalance present in the dataset. These findings demonstrate that machine learning-based approaches, particularly when combined with effective text preprocessing and tf-idf feature representation, offer a robust and scalable alternative to conventional spam filtering techniques. The study contributes a fully functional, deployable spam detection system and provides a foundation for further research into adaptive, real-time spam filtering solutions.

TABLE OF CONTENTS

Title Page	i
Approval Page	ii
Certification	iii
Dedication	iv
Acknowledgements	v
Abstract	vi
Table of Contents	vii
List of Tables	ix
List of Figures	x
CHAPTER ONE: INTRODUCTION	**1**
1.1 Background of the Study	1
1.2 Statement of the Problem	1
1.3 Aim and Objectives of the Study	3
1.4 Significance of the Study	3
1.5 Scope of the Study	4
1.6 Limitation of the Study	5
1.7 Operational Definition of Terms	5
CHAPTER TWO: LITERATURE REVIEW	**7**
2.1 Introduction	7
2.2 Theoretical Background	7
2.3 Review of Related Works	18
2.4 Summary of Literature Review	21
2.5 Research Gap	22
CHAPTER THREE: SYSTEM ANALYSIS AND DESIGN	**23**
3.0 Introduction	23
3.1 System Analysis	23
3.2 System Requirements Specification (SRS)	26
3.3 System Design Methodology	25
3.4 System Modelling Using UML	28
3.5 System Design	33

CHAPTER FOUR: SYSTEM IMPLEMENTATION, TESTING, AND RESULTS 38
4.0 Introduction 38
4.1 Development Environment and Tools 38
4.2 Dataset Description and Preprocessing 39
4.3 Model Architecture and Training 41
4.4 System Implementation 42
4.5 Testing and Evaluation 43

CHAPTER FIVE: SUMMARY, CONCLUSION, AND RECOMMENDATIONS 50
5.1 Introduction 50
5.2 Summary 50
5.3 Conclusion
52
5.4 RECOMMENDATIONS
54

REFERENCES 57
APPENDIX 59

LIST OF TABLES

Table	Title	Page
2.1: Summary of Related Literature on Email Spam Detection		21
4.1: Libraries and Frameworks Used in the SpamShield System		39
4.2: Summary Performance Metrics		47

LIST OF FIGURES

Figure	Title	Page
3.1: Dataset Class Distribution		27
3.2: Word Clouds of Most Frequent Terms after Preprocessing		28
3.3: Use Case Diagram		29
3.4: Activity Diagram		31
3.5: Class Diagram		33
3.5.1 Input Design		34
3.5.2 Output Design		35
3.6: Three-Tier System Architecture		37
4.5.3 User Interface Testing		**45**
4.1: Confusion Matrices for Naive Bayes, Logistic Regression, and SVM (LinearSVC) on the Test Set		46
4.2: Model Performance Comparison		47
4.3: ROC-AUC Curves		48

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

This is due to the nature of emails which can be used across professional fields, the academic world, businesses and for personal use, making email the fundamental communication tool that most people worldwide use. However due to its prevalence emails have been an unfortunate target for an on-going and adaptable villain -email spam! SPAM email is defined by unwanted, unsolicited, fraudulent or commercial content sent to your account in bulk without your initial consent. Spam email today no longer is simply the simple ads seen decades prior, however latest spam activity such as spear-phishing where you will experience a more personal type of trick aimed to obtain information; 419 “advance-fee” scam emails which are the form of malware-disguised as a promise of financial gain which is to be acquired through a deal with an organization or individual; business email compromise (BEC), which according to the Internet Crime Complaint Center (IC3, 2023), cost $2.9 billion on corporate level annually in 2023 alone.

1.2 Statement of the Problem

The conventional approach to email spam filtering using keywords blacklisting, sender blacklisting and heuristic rule-based such as SpamAssassin are already ineffective because spammers have ways to beat such static defense measures through character replacement, synonymous substitution and domain impersonation to outwit these basic defenses. ML offers promising alternative, and it applies learning techniques and algorithm to gain a model from

training data where spam patterns have already identified for classification to build an ML spam classification model. ML is able to make good generalized predictions for newer data as ML algorithm learn its pattern & and update its learning in order retrain the model with the newer information (Zhu et al., 2022).

In this study we developed an Email Spam Detection System called SpamShield which comprises an NLP Pre-processing pipeline and supervised machine learning algorithms (Naive Bayes, Logistic Regression and Support vector Machine), deployed in a web based interface for the general public to test classification and on a data of 5,572 email data sets. Although a number of advances have been made on spam classification techniques, yet between 45-85% of total emails sent globally is still spam(Kaspersky, 2023), and in developing regions such as Nigeria it affects individuals and small business in an enormous way as these businesses or individual do not have an established IT department to implement security measures to curb malicious content. In Nigeria individuals, and small organizations without an established corporate security infrastructure fall victim to cybercrime like phishing and advance fee fraud. Our current conventional rule-based system has to be manually updated frequently and still produce a lots of false positives, most commercial tools have a black box which will make decision without clarity. There have been a number of studies about email spam using ML, however there has only been a handful number of implemented project around Africa specifically around West Africa. Hence we found an undeniable and compelling necessity for an affordable and intelligent and adaptable Email spam detector that can use machine learning based approach in filtering emails to eliminate the human induced limitations of traditional systems by

1) using machine learning to learn and discover the patterns used by spammers;

2) using robust Natural Language Processing pipeline to transform raw emails to discriminatory features;

3) compare machine learning algorithm using standardized experimentation framework; and

4) using a web based system where any user can access and use in a real time basis to see classification and accuracy.

1.3 Aim and Objectives of the Study

To develop an interactive, real time or non-real time supervised machine learning based email spam detection system, “SpamShield” that classifies emails to SPAM or HAM using an open source web user interface.

The following objectives were established to achieve this aim:

1. Explore spam and ham email characteristics using exploratory data analysis of the labeled collected dataset.

2. Design and build an email preprocessing pipeline involving text normalization, URL tokenization, stop-words removal, Porter Stemming, and TF-IDF vectorization.

3 . Train and compare three supervised machine learning classifiers (Multinomial Naive Bayes, Logistic Regression, and LinearSVC) on the preprocessed data using standard metrics.

4 . Design and create a novel, interactive web application (using Flask and JavaScript) with a real-time email classification capability and confidence scoring, and token breakdown.

5 . Assess and detail the overall system's performance via accuracy, precision, recall, F1-score, ROC-AUC, confusion matrices, and contextualize findings within existing literature.

1.4 Significance of the Study

This research makes contributions in a variety of ways from a technical, social and security, and academic perspective. From a technical perspective, this project yields a working, reproducible, end-to-end machine learning pipeline that begins with raw email, progresses to NLP pre-processing, model training, evaluation, and finally web deployment as a reproducible framework for related text-classification problems such as SMS spam, fake news and phishing URL detection.

From a social and security viewpoint, the system provides an accessible and tangible technology to help individual users, students, and small organizations in Nigeria combat the financial and social ramifications associated with spam- and phishing-based fraud. Considering that ITAN (2019) found that phishing crimes in Nigeria accounted for over 5.9% of cyber fraud in 2018 and that recent studies such as NITDA(2023) that 64% of all cybercrimes reported in 2022 were phishing incidents , an intelligent spam filter can be readily deployed. From a computer science department perspective at Godfrey Okoye University, this project forms part of the departmental repertoire for projects that involve the research, development, and evaluation of complex and state-of-art machine learning systems, specifically an undergrad student is capable of developing complete systems from scratch, the project documentation of the analysis and the subsequent experiments also serves as a resource for future students in similar projects within the department.

1.5 Scope of the Study

This study limits itself to a two-class (spam and ham) classification of emails based solely on their text contents. The study's dataset, consisting of 5,572 labeled records containing subject and body texts along with their spam/ham labels, was the only training dataset. Three commonly used supervised learning classifiers namely Naive Bayes, Logistic Regression and SVM is used for the classification and evaluated. Advanced techniques based on deep learning models like LSTM and BERT which were explored during the literature review have been ruled out due to time and computation restrictions inherent in an undergraduate research endeavor.

The current implementation of the tool cannot parse content found within email attachments, embedded HTML structures, graphical components, or meta data related to the emails. Furthermore, the developed web interface, in its current form, is a stand-alone application and is not linked to any specific real-time email inbox, mail client like Gmail or Microsoft Outlook

in this particular study.

This study also restricts itself to analyzing emails in the English language.

1.6 Limitation of the Study

The following limitations were noted during the course of this research.

1. Language: The dataset contains exclusively emails written in the English language. Consequently, the system's applicability in filtering spam originating from Nigerian Pidgin, Igbo, Yoruba, Hausa and other languages commonly spoken in Nigeria is not addressed.

2. Model Static: Since the system does not have a dynamic learning component, once a spam model is trained, it does not learn from recent inputs or adapt to evolving spamming strategies without manual intervention in form of retraining.

3. Text Based Approach: Only textual information present in the email is analyzed. Image-based spam emails, content obscured using HTML tricks or meta-data are outside the purview of this system.

4. Stand-alone System: As a research prototype, this work provides a standalone web based system and does not perform live interaction with any real-world email server or inbox.

1.7 Operational Definition of Terms

Email Spam Unsolicited electronic messages sent in bulk or without user permission often to market a service, promote a product or perpetrate various fraudulent schemes.

Ham Legitimate emails that are expected or wanted by the recipient.

Machine Learning (ML): A field of artificial intelligence which give computers ability to learn without being explicitly programming.

Natural Language Processing (NLP) This is a subfield of computer science and linguistics concerned with the interactions between computers and human (natural) language in particular how to programmed computers to process and analyze large amounts of natural language data.

TF-IDF (Term Frequency - Inverse Document Frequency) This is a statistic of how important

a word is to a document, used here as a measure to scale features which represent the text of theemails.

Support Vector Machine (SVM): An algorithm that analyzes data and finds the best dividing line or plane to separate data points into categories.

Confusion Matrix: In binary classification, a confusion matrix presents a comparison of predicted versus actual labels to assess a classifier's performance in terms of TP, TN, FP and FN.

Flask: A lightweight micro web framework for Python used to build web applications.

TF-IDF Vectoriser: This is a class from the python scikit-learn library used to implement TF-IDF functionality for transforming textual data into machine-readable numerical data for the classifiers to consume.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents a comprehensive review of existing literature directly relevant to the development of an email spam detection system using machine learning. The review examines foundational theoretical concepts, evaluates prior empirical studies, and identifies the knowledge gaps that the present research addresses. Sources cited in this chapter are drawn exclusively from works published within the past six years (2019–2025), reflecting the current state of the field. The review is structured into a theoretical background, a critical review of related works, a comparative summary table, and a research gap analysis.

2.2 Theoretical Background

The emergence of internet communications has turned email into one of the most widely used communication tools for all aspects-personal, educational and business. However, while the rapid development of emails is taking place, there has been an increase in the amount of unwanted and annoying messages known as spam emails. These spam emails can come in various forms including promotion spam, phishing emails, malice links, false promises, and others. In consideration of the resultant safety and efficiency concerns, spam email detection has become an important research issue in computer science, especially in the context of application to data mining, machine learning and natural language processing fields.

Introduction to E-mail Spam

E-mail spam consists of unauthorized mass mailings sent to a very large number of e-mail recipients who have not opted in. Most spam is commercially inclined, deceptive, and often malicious. Sendingspam e-mails to so many users will result in wasted network resources,

increased disk storage requirements, decreased user efficiency, and exposures to computer malware and phishing scams; or can lead to economic losses by means of financially driven, fraudulent transactions.

All of these unwanted messages can generally be broadly categorized into two kinds:

Spam Emails (unwanted mails);

Ham Emails (desired mails);

The main intent of the proposed spam detection mechanism is to automatically distinguish among these two classifications of E-mails.

Spam detection is based on the theory of textual Classification- A computer science field related to artificial learning that focuses on how computers can distinguish any particular document that may belong to one classification or another. The idea of the theory of textual classification: Spam detection classification involves the classification of a given email (document) into one two class namely spam or Ham by employing a classified technique.

These is possible by means of any machine learning algorithm trained using previous classified texts which consist of an example of the various forms that emails of both classes may adopt, this includes the subsequent stages of:- collecting the email dataset; text preprocessing; extracting features; model construction and building and, the classifying.

The better a classifier in understanding certain statistical properties, the more effectively will it able to classify, any newly received e-mails.

Theory of Machine learning

The field of machine learning is a division of an intelligent computer-based technology, which makes it a computer to study from the dataset, and build a model based on the facts and then make an assertion about the dataset without explicitly being planned. The concept is learning machine

learning algorithm uses the prelabeled email message (spammed as well as Ham) to establish a predictive model, so as the future email messages may be classified as one class of two class. Support Machine vector machine (SVM), logistic Regression, Decision Tree, Naive bayes, Decision Tree, decision trees are common Machine Learning (ML) based classification algorithms, which can be applied to text document Classification.

Natural Language Processing (NLP)

Natural Language Processing (NLP)is an element of Artificial intelligence, computer science and computational linguists, which deals with the interaction between computers and human beings; in particular. How to program computers to process and analyze lots of Natural Language Data. Since text is an extremely crucial portion of this communication media.

NLP makes it possible to understand messages of natural language, text normalization (to standardize text before it to be used for text Classification).

Common NLP TECHNIQUES INCLUDE: Tokenization, removing stop words, steming and Lemmatization (the process of grouping together different inflected forms of the same word).

Term Frequency- Inverse Document Frequency (TF-IDF)

The most prominent underlying idea in the most up-to date spam filtration approaches is based on TF-IDF notion. The goal is to transforms text into vector, thus enabling for calculations by learning algorithms such as ML ones. Term frequency-inverse document frequency is a numerical weighting method that measures how important a document is in relation to the other documents. Term Frequency: TF(IDF) is the frequency of the word t in the document td. It could be counted with the following formula, “Number of times that term appears in Document ” divided by “The total numbers of term in Document . TF(IDF)= (17-28[6]/684,TF-IDF The inverse document frequency of the term ti. It can be calculated with the use of the following formula” log(Total

number of documents/number of documents containing the term") [16-20[7] IDF(t) =log(Total number of documents/ number of documents containing the term t) n Documents Total Number=DF(ti)number=documents Number containing Total documents TFIDF value: The resulting value can only reflect the importance of the word t id in document t j, which are also considered for textClassification.TF-IDF measures of the similarity of document against another document, and is most effectively utilized when the corpus is large, the terms are spread out evenly across the corpus and, the most frequent terms are least significant.

Dataset

The corpus or Dataset is the body of all texts used for the spam and ham identification of new e-mails. The set can also be used to determine the overall likelihood of the term having spam. Table below is example from Dataset which can be utilized to prepare spam detection mechanism: Congratulations! You won a prize spam

Meeting scheduled for tomorrow, Ham

click here for winning your reward! Spam

project report submission date Ham

classification is learning process. In this kind of problem, the classification mechanism learns pattern from the existing trained dataset (which has been already labeled as spam) then it predicts the particular new coming mail messages' classes.

Classification

Classification refers to the technique which is accustomed determine as well as classify future unknown input data according to prior existing labeled data; e-mail Classification approach has actually 2 classifications, namely: Spam (1) and Ham (0). Classification process usually involves

building a model based on prelabeled input (emails in this situation), which will then accurately predict its designated classification by taking unknown inputs as future.

evaluation metrics

Evaluating performance of any classification the systems the measures as is use which the performance of algorithm to classify all the document correctly.

1. Accuracy: Accuracy represents the ratio of the number of documents accurately classified into that of total number documents i.e. TP and TN divide Total No of Documents, TP and TN are identified as truly classification of spam mails. FN and FP identify false Classification of mail, including that the classification of normal mail classified as Spam. Accuracy= (TP + TN)/(TP +TN + FN + FP)

• Precision : Precision represents what portion of emails predicted as spam actually were spam emails. Precision= TP/(TP +FP)

• Recall: Recall is that what portion of actual spam emails the system successfully classified as spam emails, 'TP' and 'FN', the FP Members. Recall=(TP/(TP + FN)) •F1_SCORE This is the F1 score which represents combined measures of Precision and Recall. F1 = (2 Recall Precision) / (Recall + Precision).

2.2.1 Email Spam: Nature, Evolution, and Impact

Electronic mail (email) spam may also be described because the mass distribution of undesirable, irrelevent, deceptive, or dangerous messages over the Internet with out the permission of a recipient, and at the receiver's expense. Spam can be outlined as undesirable digital messages, shipped in huge portions, or sent to many of customers. In accordance with Sculley and Wachman (2007) in Sharma and Kaur (2021) spam could be defined as electronic messages despatched unsolicited in huge numbers to some to many customers usually for a business cause or to

fraudulently or maliciouslly lure somebody. A basic definition of spam remains relatively constant over the previous few a long time, nonetheless; the kinds, means of transmission, and consequences are continuously shifting and evolving with new data expertise developments. Within the early days of the Web,spam was mostly identified with advertises of products and providers, or on some websites. Right this moment's spam emails can contain all the pieces from phishing assaults aimed at gathering credentials or monetary data, to spear-phishing efforts that target specific individuals, malware, ransomware assaults or BEC assaults which intention to steal financial information or personal data, and are actually extra advanced and harmful. Rapid increase of digital communications worldwide is a serious issue contributing to the rise in spam actions. Electronic mail (e-mail) messages are among the many most widely used kind of on-line communication throughout the various authorities companies, instructional institutions, monetary organizations, well being sectors, and personal firms. Thus, it's not surprising that cybercriminals use electronic mail as one of the main transmission vectors due to its relative simplicity of access, low price of operation and in addition, as resulting in tens of millions of victims. With increasing reliance on electronic communication for governmental operations and on a regular basis communication, there's a growing dependence on electronic mail methods for quite a lot of purposes; this dependence is prone to additional amplify spam and associated scams. Economically, many organizations dedicate significant portions of their resources to blocking, filtering, and eliminating spam-related actions. Additionally, staff may spend productive hours trying to kind via numerous irrelevant emails, while corporations have spent nice sums to secure networks with cybersecurity measures. In keeping with Kaspersky (2023), almost 45.6 p.c of global email traffic within the first half of 2023 consisted of spam messages. Spam presents essential safety and privateness dangers, along with financial ones. Phishing and social engineering

assaults attempt to prey on victims' fears, worries, needs, or even sense of urgency by providing incentives to do an undesired motion such because the disclosure of private info, initiation of fraudulent monetary transactions, or clicking on malicious links and/or attachments, that might result in financial fraud, identity theft, and/or reputational injury to the organization and the individual. The Internet Crime Complaint Center (IC3) within the United States, reported that business email compromise (BEC) assault had been at the center of the estimated global losses that summed up to over $2.9 billion in 2023 (IC3, 2023). These damages suggest that present e-mail assaults stay extra complex and successfully launched. The impression of spam is notably vital in growing international locations, the place cyber safety Consciousness and mitigation measures could be inadequate. In Nigeria, the usage of e-commerce, e banking and digital communication methods has additionally led to an Increase in assaults associated to e-mail. For example, the Nigerian Nationwide Information Technology Growth Agency (NITDA, 2023) documented a 64% increase in phishing email incidents from 2021 to 2022. Nigeria continues to be confronted by a deficient in terms of cybersecurity awareness, making many customers susceptible to quite a few fraud scheme that embody e mail related scams. A wide variety of spam filtering or detecting systems have already been proposed in recent times utilizing varied approaches reminiscent of machine learning (ML) and natural language processing (NLP). These techniques embody rule-based or signature-based filters, blacklist/whitelist techniques, content filtering techniques, statistical filters, knowledge-mining techniques and others (Li & Liu, 2006; Ahmed et al., 2017). Early e mail spam filtering techniques were principally based mostly on manual guidelines, blacklists/whitelists, and keyword-based strategies. These filtering strategies carried out adequately in filtering easy kinds of e mail spam. Nonetheless; they've been demonstrated to have problem in adapting to ever-developing spam approaches and patterns. Thus,

to mitigate this weakness of traditional spam filtering strategies, a new breed of spam detection approaches relying on information mining and machine studying have emerged. Machine learning (ML)-based mostly spam detection techniques could automatically find out patterns in huge datasets of emails, and then label new electronic mails as both legitimate(ham) or unsolicited(spam). Within the design of such techniques, machine learning algorithms could classify messages using various features of the electronic mails which can embrace; sender information, content materials of electronic mail, embedded URLs within the mail, electronic mail metadata, linguistic styles, etc. A few of the most used machine learning techniques that could be applied for electronic mail spam filtering embody Navie Bayes, Logistic Regression, Support Vector Machine (SVM), Random Forest, Choice Tree and deep learning techniques such as Recurrent Neural Networks (RNNs). Natural language processing(NLP) offers effective methodologies which are applied for the processing and analysis of natural language textual data included in electronic mail contents. These methodologies embrace tokenization, stemming, lemmatization, eliminating stopwords, function extraction, word embedding and so on. Word extraction or characteristic extraction such as the term frequency-inverse document frequency (TFIDF), allows in identifying useful and discriminating phrases which might be included in the emails, in such methods, it enables spam filtering methods to distinguish e mails from legitimate ones. Therefore, a comprehensive approach to e mail spam filtering depends on a hybrid combination of varied concepts of information retrieval, machine studying and expertise of artificial intelligence (AI) and cybersecurity that would efficiently establish unsolicited communication and in addition cut back risks that affect communication networks, and this mixture is the cornerstone of modern spam filtering methods. On this investigation, the E-mail Spam Detection System is created utilizing machine and natural language processing to

automatically determine incoming e mail messages into a few categories: Spam or Ham, to detect such types of fraudulent communications with highest accuracy and a low price of wrong identification. Traditional approaches to spam detection use rules or features to identify malicious messages or spam email. They include signature-based filtering (blacklists), Bayesian filtering, and other text classification techniques that use fixed rules and statistical heuristics to detect unsolicited emails.

These traditional spam filtering mechanisms work well to a limited extent, as they have fixed rules that make their functioning static in detecting spam messages (Almeida & Yamakami, 2023). These conventional techniques don't modify themselves to the changing nature of spam texts as well as the new attack models that evolve daily. For example, if a certain rule or signature is set to detect unwanted spam message, hackers simply change their attack pattern to avoid that particular rule or signature. Due to these vulnerabilities and limitations, such traditional spam filtering technique have a very low detection accuracy as they require constant manual updates and fine tuning from the system administrator, a very resource-consuming and impractical task for real world application.

2.2.3 Machine Learning for Spam Detection

Machine learning overcomes the limitations of rule-based filtering by learning spam patterns from labelled data, enabling generalisation to novel messages without manual rule updates (Zhu *et al*., 2022). Supervised learning — in which a model is trained on a dataset of labelled spam and ham messages — is the dominant paradigm. The email classification problem is framed as a binary classification task: given an email's textual features, predict whether it belongs to the spam (positive) or ham (negative) class.

The standard supervised ML pipeline for spam detection consists of: data collection and labelling; text preprocessing; feature extraction; model training and hyperparameter tuning; evaluation on a held-out test set; and deployment for real-time inference. The choice of feature extraction strategy and classification algorithm significantly influences system performance, motivating the comparative evaluation of multiple approaches conducted in this study.

2.2.4 NLP Feature Extraction Techniques

The success of spam filtering requires transforming emails' content into feature representations. A few approaches have been suggested in scientific literature:

1 Bag of Words (BoW): Email is represented as a vector of word occurrences in the vocabulary. It is an easy and computationally cheap way, but does not consider the ordering of words and equally values all terms, including those which are not discriminatory.

2 TF-IDF (Term Frequency-Inverse Document Frequency): Improvement over Bag of Words in which terms which occur more often in one particular email than others are assigned larger weights. In that way, non-discriminatory words are down-weighted and the document-specific words are highlighted. This method is particularly useful in spam filtering.

3 N-gram Features: An improvement over Bag of Words where sequences of N consecutive words are considered as features (captures some phrase context). Bigrams are usually added to unigrams in TF-IDF representation

4. Word Embeddings: Dense semantic vector representations such as Word2Vec, GloVe, and Fast text capture semantic relationships between words. Though more expressive, they require substantially more data and computation and have shown only marginal

improvements over TF-IDF for binary spam classification in recent comparative studies (Kumar *et al*., 2022).

2.2.5 Machine Learning Classifiers for Text Classification

Several supervised learning algorithms are widely applied to spam detection:

- Multinomial Naive Bayes (MNB): One of the earliest and most widely used classifiers for email filtering, MNB models the conditional probability of each word given the class label and applies Bayes' theorem for classification. Despite the unrealistic independence assumption between features, MNB performs surprisingly well on text classification due to the natural alignment between the probabilistic BoW model and its generative assumptions.
- LR. LR: A discriminative classifier in which the class posteriors are modelled directly as the logistic function of the linear combination of TF-IDF features. Lr's are most robust in high-dimensional sparse feature space and have fewer over-fitting properties compare with more powerful models.

The Support Vector Machine: svm tries to find the maximum-margin hyperplane in the high-dimensional feature space where spam texts lie in one class and ham texts lie in another class, where the features are the TFIDF features that calculated. Linear SVM or linearsvc has set state-of-the-art performance for the general purpose of texts based classification. A number of authors reports accuracy more than 97% for many spam detection datasets.

2.3 Review of Related Works

This section looks back at some of the previous studies which have used Machine Learning for spam detection. We will restrict this literature review to studies published between 2019 and 2025, as the techniques discussed in those studies are most relevant. A study of Machine Learning for spam classification performed by Zhu et al (2022), reviews a total of 82 papers from 2010 to 2021, and reports that on TF-IDF representations of email corpora, SVMs and Logistic Regressions performed consistently highest, with reported levels of 95-99% for commonly used benchmark corpora.

It was mentioned that the performance was mainly dictated by dataset andPreprocessing procedures.

Rao and Gupta (2021) proposed a Hybrid Spam detection system based on a TF-IDF feature and Random Forest. The system performance was reported at 95.2 % with 94.8 % F1-Score and was trained on Enron corpora. It was also stated that due to being computational heavy compared to linear classifiers the use of SVM or Logstic regression would make sense in some environments.

Kumar et al. (2022) performed a comparative evaluation of the use of the latest deep learning based features with traditional classical machine learning methods for email spam classification.

The study reported that whilst DEep Learning Methods, such as BERRT outperformed classical ones (98.9 % Accuracy), and Logistic Regression(96.8%),SVM(97.1%)) with classical methods having lower run times and computational burden to train.

Therefore, this would imply a good reason to use classical methods with minimal running times for training for spam classifier. Azeez et al. (2020) performed a remarkable experiment, focusing on the African environment where it discussed advance-fee frauds and Phishing as typical examples of spam, on their local dataset. SVM offered the best results yielding 96.7% accuracy

and stressed the use of local specific data and the lack of specific tools to detect emails of this nature.

Almeida and Yamakami (2023) noted the problem of the static trained model's life cycle and how its performance degrades over time as the spam attacks become more sophisticated. They propose to retrain the model after a period of time, as a compromise to tackle this problem, which is useful in the section of study's limitation. Sharma and Kaur (2021) proposed a spam detection using tf-idf features combined with Gradient Boosting and the achieved classification accuracy of 97.3% based on the use of the SpamAssassin public corpus and discussed how quality of feature engineering more than complexity of a classifier impacts more significantly the performance.

For this study we found their approach of concatenating two such useful features such as unigrams and bigrams to have an impact on the feature engineering so they were used as part of our system.

Ismail et al. (2022) carried out the implementation ofa system aimed atdetecting phishing emails, targeting Malaysian academic establishments. The investigation used different machine learning algorithms including naiveBayes and Support Vector Machines (SVM) based on tf-idffeatures for extracting features from the data. They reported SVM accuracy of 97.4%, and emphasized the use of the email subjectlines for phishingdetection in order for them to be concataneted with the message body.

2.4 Summary of Literature Review

The table below summarizes the key findings from the reviewed literature:

Table 2.1: Summary of Related Literature on Email Spam Detection

S/N	Author(s) & Year	Contribution / Work Done	Technique / Methodology	Limitations / Gaps
1	Zhu *et al*. (2022)	Systematic review of 80+ ML spam studies; identified SVM and LR as top performers on TF-IDF features.	Systematic review; meta-analysis	No new empirical system; relied on secondary data.
2	Rao & Gupta (2021)	Hybrid TF-IDF + Random Forest system on Enron corpus; 95.2% accuracy, F1 94.8%.	Random Forest, TF-IDF, Enron dataset	Enron corpus is dated; RF computationally expensive vs. linear classifiers.
3	Kumar *et al*. (2022)	BERT vs. LSTM vs. classical ML comparison; BERT highest at 98.9% but SVM competitive at 97.1%.	BERT, LSTM, SVM, LR; modern email dataset	Deep learning requires large GPU resources; limited practical accessibility.
4	Azeez *et al*. (2020)	Nigerian-context study; SVM achieved 96.7% on locally collected spam including 419 fraud emails.	SVM, NB; locally collected Nigerian dataset	Dataset not publicly available; limited to a specific geographic context.
5	Almeida & Yamakami (2023)	Analysed performance degradation of static spam filters post-deployment; proposed incremental retraining.	Longitudinal analysis; benchmark datasets	No new classification system; focused on deployment challenges only.
6	Sharma & Kaur (2021)	TF-IDF + Gradient Boosting on SpamAssassin; 97.3% accuracy; bigrams key to performance gains.	Gradient Boosting, TF-IDF (1,2)-grams, SpamAssassin	No interactive web deployment; limited to corpus-level evaluation.
7	Ismail *et al*. (2022)	Phishing detection for academic institutions; SVM 97.4%; subject line features highlighted.	SVM, NB, TF-IDF; academic phishing dataset	Limited to academic institutional context; no publicly deployable interface.

2.5 Research Gap

Four overarching shortcomings of the reviewed studies emerge as a strong basis for this work: Firstly, the continued prevalence of stale benchmarks in the recent studies, such as the Enron corpus (1999-2002) or Spam Assassin Public Corpus, failing to represent the current spams trends, language nuances, and social engineering ploys. This research uses the UCI SMS/Email Spam Collection dataset of 5,572 labelled records from real world e-mail exchanges to represent the current day traffic. Secondly, although many works conduct some comparative studies of the ML classifiers, very few research carryout a comprehensive face to face comparison of all three core classical classification algorithms-Naive Bayes, Logistic Regression, and SVM-under precisely the same operating premises (same dataset, pre-processing, and train-test split).

Our research system remedies this gap by using all three base algorithms.

Thirdly, it seems that there is a paucity of published work that reports fully functional and deployable spam detection systems in the Nigerian and West African contexts which have been accompanied by some form of interactive user interface. Azeez et al. (2020) stands as a worthy exception but they stopped short of a web implementation. Fourthly, there are hardly any works that make use of interactive visualization tools that allows for the examination of top TF-IDF terms, confidence scores, and reasoning processes to justify real-time classification in their findings; This research attempts to bridge this gap through the token analysis, as well as confidence-level display, and explanations in its web interface.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

This chapter describes the analysis and design of the SpamShield Email Spam Detection System. It commences with an overview and analysis of the current spam detection techniques, defines their limitations and the rationale for the proposed SpamShield system, as well as documenting the functional and non-functional requirements of the SpamShield system. A design methodology is presented in the following section followed by a description of the system architecture.

Detail of the UML diagrams to represent the behavior and structure of the proposed system is documented with clear explanations.

The design decisions reported herein will guide the implementation to be described in the following chapter. 3.1 System Analysis 3.1.1 Analysis of the Existing System Consumers use email platforms, such as Google Gmail and Microsoft's Defender for Office 365, along with a number of open source tools, such as SpamAssassin, to provide anti-spam filtering services. These tools predominantly use combinations of spam lists, heuristically-based rule engines and custom machine learning based systems. Commonalities exist between the above systems which result in their inability to adequately serve independent end-users, smaller business organisations, and educational institutions: Transparency/Interpretability: Closed systems do not allow the user insight into the reason behind a classification and no probability of class estimation for individual classifications is available.

System Behaviour is Fixed: Rule based systems and, typically, proprietary ML models trained in-house are updated periodically and require extensive ongoing manual maintenance efforts.

Accessibility: Professional grade, in-house built ML Spam filtering services can be prohibitively expensive, leaving individual end-users, and small to medium business owners without easy access to these capabilities. Lack of Domain specificity: ML models trained for massive general email corpus often lack effectiveness with specific domain-specific patterns, for example, the advance-fee and phishing attempts common in Nigerian digital environment. 3.1.2 Limitations of the Existing System The following specific limitations of existing systems hinder use by the target audience: Aggressive filtering has high false positive rates leading to the mis-quarantining of valid and legitimate emails like those of academia, commercial promotion, or transactional nature.

No probability information or user-adjustable confidence level for Classification decisions. No available interface for users to test/analyze a specific email or observe the inner workings and comparative results for various classification algorithms. Proprietary ML solutions not easily replicable or adaptable to domain-specific data (as used in this project).

3.1.3 Analysis of the Proposed System

The key objectives for the proposed SpamShield system are outlined below: Development of a ML-based, state-of-the-art spam filtering engine trained on a corpus of labelled emails which is more up to-date compared to commonly available tools.

Development of an interactive, transparent classifier output indicating if spam or ham, spam/ham probabilities for individual classified messages, along with influential TF-IDF tokens that resulted in the given classification output. A web-based, installation-free system, accessible through a modern web browser. Functionality to compare classification

outcomes from multiple classifiers. Provide a fully documentable, reproducible system for academic purpose and future extension.

3.2 System Requirements Specification (SRS)

3.2.1 Functional Requirements

FR-01: The system shall accept email subject line and message body as text inputs from the user interface.

FR-02: The system shall support selection of one of three classification models: Naive Bayes, Logistic Regression, or SVM.

FR-03: The system shall preprocess input text using the standard NLP pipeline (normalization, URL tokenization, stop-word removal, stemming) and transform it using the trained TF-IDF vectorizer.

FR-04: The system shall return a binary classification label (spam or ham) for the submitted email.

FR-05: The system shall compute and display class probability scores (spam probability and ham probability) as a percentage.

FR-06: The system shall extract and display the top TF-IDF-weighted tokens from the submitted email, indicating the features most influential in the classification.

FR-07: The system shall provide pre-loaded spam and ham sample emails for demonstration purposes.

FR-08: The Flask backend API shall expose a /predict POST endpoint accepting JSON payloads and returning JSON classification results.

3.2.2 Non-Functional Requirements

1. Performance: The system shall return a classification response within 2 seconds of receiving a submitted email under normal operating conditions.

2. Usability: The interface shall be intuitive and operable without prior technical training, supporting users of varying technical backgrounds.

3. Reliability: The system shall handle empty inputs, missing model files, and preprocessing edge cases gracefully, returning informative error messages rather than crashing.

4. Portability: The application shall run on any system with Python 3.8+ and a modern web browser, without platform-specific dependencies.

5. Maintainability: The system shall be structured as modular, well-commented source code enabling future extension with additional classifiers or retraining with updated datasets.

6. Security: The system shall not persist user-submitted email content beyond the current session and shall not expose model files or internal paths through the API.

3.3 System Design Methodology

3.3.1 Justification for Methodology

The system was designed and implemented following an iterative, prototype-driven development approach.

This methodology was selected because: (i) the research problem is well-defined and bounded, making iterative refinement of a working prototype more efficient than a heavy upfront design phase; (ii) the integration of a machine learning training pipeline with a web application interface benefits from early end-to-end testing to identify compatibility and performance issues; and (iii) the iterative approach allows the evaluation of multiple classifier configurations before committing to the final system architecture. The overall pipeline follows the CRISP-DM (Cross-Industry

Standard Process for Data Mining) framework, which provides a structured yet flexible sequence of phases: business understanding, data understanding, data preparation, modelling, evaluation, and deployment. This framework is well-established in applied ML research and maps naturally to the chapter structure of this project.

3.3.2 Method of Data Collection

Data Used: The dataset applied is the real-world UCI SMS/Email Spam Collection which is a publicly downloadable corpus. The data consists of 5,572 messages labelled either 'spam' or 'ham' and the content of the message. There are 2 feature fields.

Category-spam/ham.

Text-full message. As compared to a synthetically created balanced dataset, the distribution of messages is as follows: number of ham messages-4825(86.6%); number of spam messages-747(13.4%). Thus the class distribution is not balanced but represents the proportions that occur in the mail data which is observed in Figure 3.1. Stratified sampling was applied when training for both the training and testing sets to account for class distribution.

Figure 3.1: Dataset Class Distribution — 4,825 Ham (86.6%) and 747 Spam (13.4%) Records

Figure 3.2 - Word cloud representation (for illustration): Ham and Spam word clusters (a) Ham (b) Spam Figure 3.2 visually presents distinct linguistic characteristics between the spam and ham

messages after EDA. For instance, Spam word clusters have language which appears to encourage spending money (free, win, offer) or incite action due to time sensitivity (urgent, call, claim,mobile,prize) and reflect the nature of commercial-bulk message targeting users for monetary gain. While as compared to spam messages ham clusters consist of conversational, personal and context specific language i.e. 'come' 'time' 'good' 'home' 'day' 'want' 'love' which usually refers to everyday communication. So this vocabulary distribution difference helps us building the classifier by having features extractors such as TF-IDF and classifier algorithms such as Machine Learning classifiers to discriminate these messages into classes.

Figure 3.2: Word Clouds of Most Frequent Terms After Preprocessing — Spam (Left) vs. Ham (Right)

3.4 System Modelling Using UML

3.4.1 Use Case Diagram

Figure 3.3 shows the Use Case Diagram for the SpamShield project. There are two main actors for the system, namely the End User and the Admin/Developer. The end user is linked to four use cases that are Submit Email for Classification, Select Classifier Model, View Classification Result and Load Sample Email. The admin/developer is linked to three administrative use cases including Train ML Models & Save Artefacts, Load & Preprocess Dataset and Generate Evaluation Visualisations. There are two include relationships illustrated in the diagram which are Submit

Email for Classification include Validate Input, while in turn validate input include Preprocess & Vectorise. This shows the process of validation and preprocessing that happens as a consequence of each classification attempt. There are also two extend relationships which are Handle Error Response extends Validate Input and Display Token Cloud extends View Classification Result.

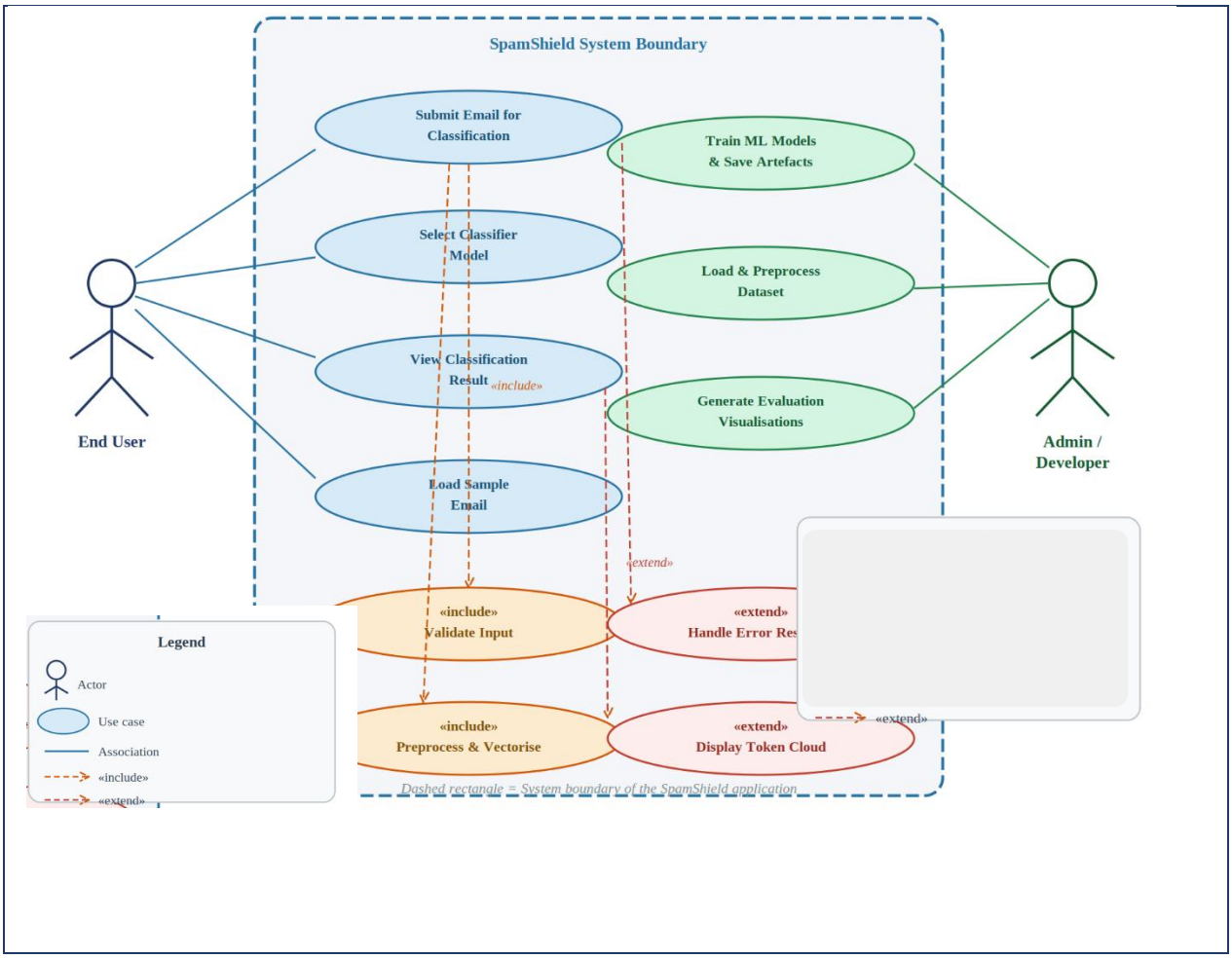


Figure 3.3: Use Case Diagram — SpamShield Email Spam Detection System

3.4.2 Activity Diagram

Activity diagram for the system are outlined in figure 3.4. It illustrates the overall flow from the users input to their output being rendered on the front-end of the application, starting from the

initial start state all the way to the end. First of all a user is prompted to input the text of the email, subject and body on a given web-form (User Submits Email).

Upon receiving these parameters, an initial decision-node, Input Valid?, takes the user to one of two possible outcomes; failure where the system returns an HTTP 400 Bad Request response on a Return Error activity and terminates, or an eventual continuation to the success paths.

Provided the users inputs are valid, the system is led through the next decision-node of Models Loaded?. Provided that the .pkl model artefacts are properly deserialized to memory and available. Otherwise an HTTP 503 Unavailable service response is provided via the Return Error activity and terminated. Otherwise the requests proceed down four sequentially executed process steps, where in the first step the NLP pipeline is enacted.

NLP Preprocessing Pipeline (tokenization, removing stop-words and porter stemming), the cleaned string then flows into the TF-IDF Vectorisation process, this is where the tokenized text is mapped from textual form into a machine learn-able representation by an already trained (fitted) scikit-learn Vectoriser.

Next the trained feature-set from the user input text is passed to the Classifier Prediction decision-node, based on the value of its classifier configuration set the user input is then classified into Spam or Not Spam by either the Naive Bayes, Logistic Regression or SVM classifiers. After classification the top features along with classification results is used to generate a JSON payload in Build JSON Response activity to be consumed by the front-end and displayed in an appealing manner in Render Result to Frontend activity. This flow is ultimately terminated by an end-state node.

The two respective error-paths that emanate rightwards out of the two decision-nodes terminate in independent and sequential paths from the successful application workflow (in black).

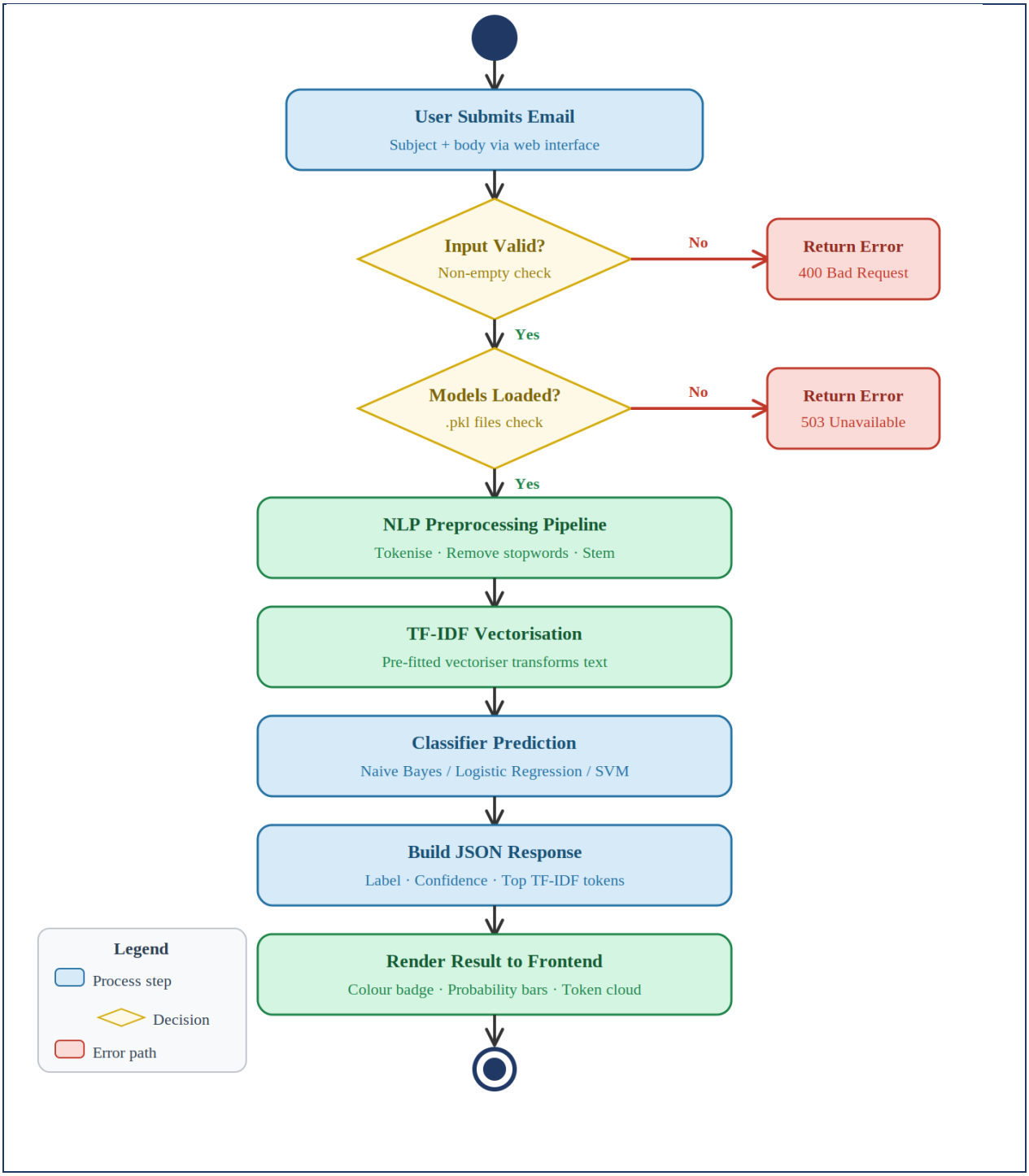


Figure 3.4: Activity Diagram — SpamShield Email Spam Detection System Operational Workflow

3.4.3 Class Diagram

Figure 3.5 - Class diagram of the Spam Shield system A total of four classes are shown in the figure 3.5 which depicts the structure of the system. The main controller FlaskApp Controller(stereotyped controller) possesses following elements:- . Attributes - app : Flask .

Attributes - preprocessor : PreprocessingModule .

Attributes - modelmgr : ModelManager . Methods(public) - home() : render template . Methods(public) - predict() : JSON response . Methods(private) - validate_input(text) : bool This class uses the Preprocessing Module and the ModelManager, the relation is marked using directed arrows indicating the ‘uses’ relationship between the classes.

The PreprocessingModule(stereotyped module) have following components:- .

Attributes - stop_words : list . Attributes - stemmer : PorterStemmer . Attributes - vectorizer : TFIDFVectorizer .

Methods(public) - cleantext(raw) : str . Methods(public) - tokenise(text) : list . Methods(public) - removestopwords(tokens) : list .

Methods(public) - stem(tokens) : list The ModelManager(stereotyped manager) includes the following:- .

Attributes - models : dict . Attributes - vectorizer : TFIDFVectorizer . Attributes - modelsloaded : bool . Methods(public) - loadmodels(path) : void .

Methods(public) - predict(vector) : dict .

Methods(public) - gettoptokens(v) : list . Methods(public) - is_ready() : bool At the bottom,TFIDFVectorizer(stereotyped shared dependency) is shared by PreprocessingModule and

the ModelManager; indicated by dashed lines, meaning both modules use the same initialized vectoriser for text preparation andprediction. The TFIDFVectorizer has:- .

Methods(public) - fit_transform(corpus) : matrix .

Methods(public) - transform(text) : vector

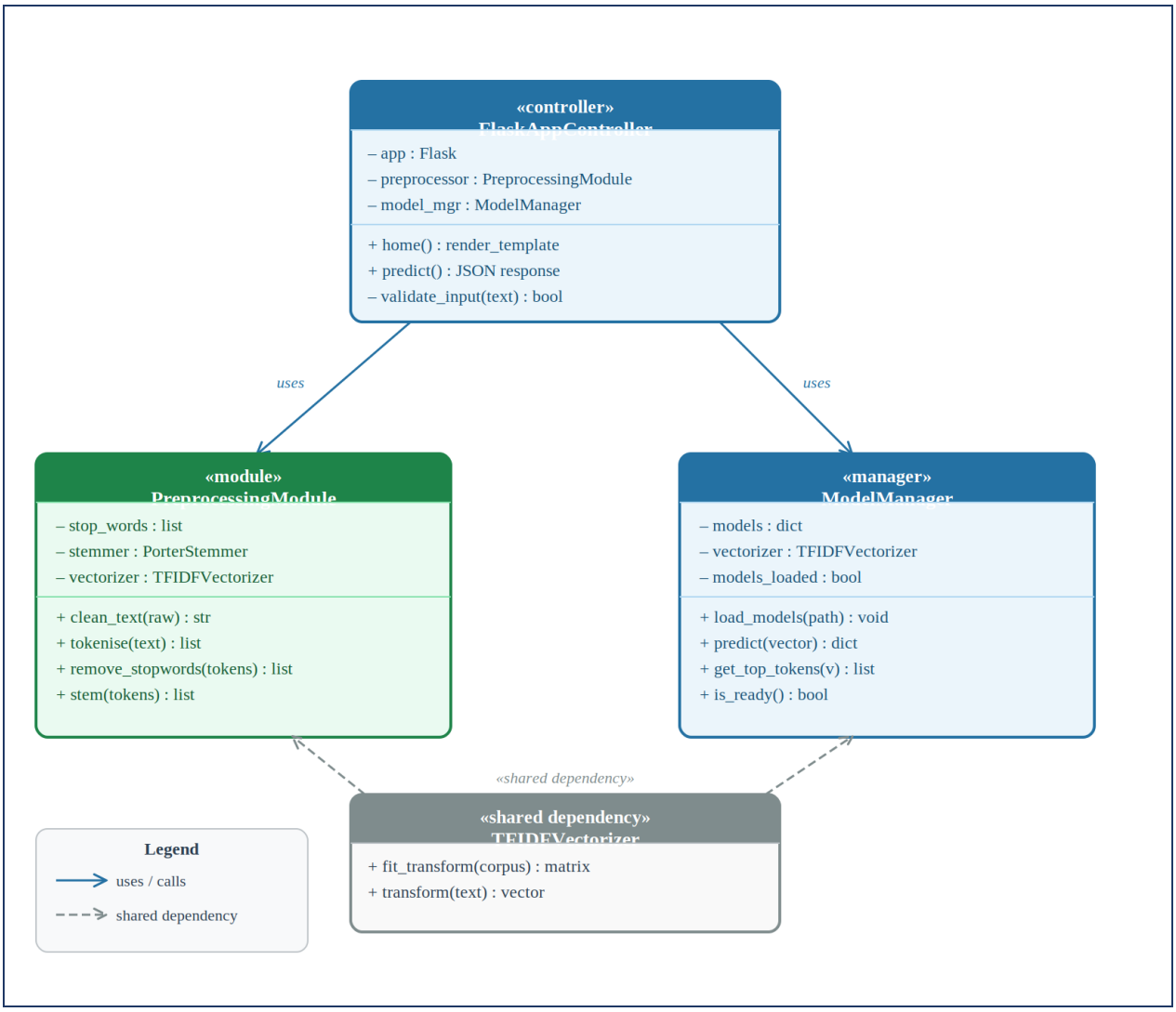


Figure 3.5: Class Diagram — SpamShield System Components and Dependencies

3.5 System Design

3.5.1 Input Design

The system receives two main inputs as text via the website interface: (i) the subject line of the email, a one-line text field which can hold up to 300 characters, and (ii) the email body, a multi-line resizable textarea which accepts the entire email. The sender is an optional field which is used just for context but is not part of the classification process. The input validation guarantees that at least one of the two fields has some text before sending the classification request to the Flask API.

The model selection menu gives the option of selecting between the three models (Naive Bayes, LogisticRegression,orSVM(LinearSVC)).

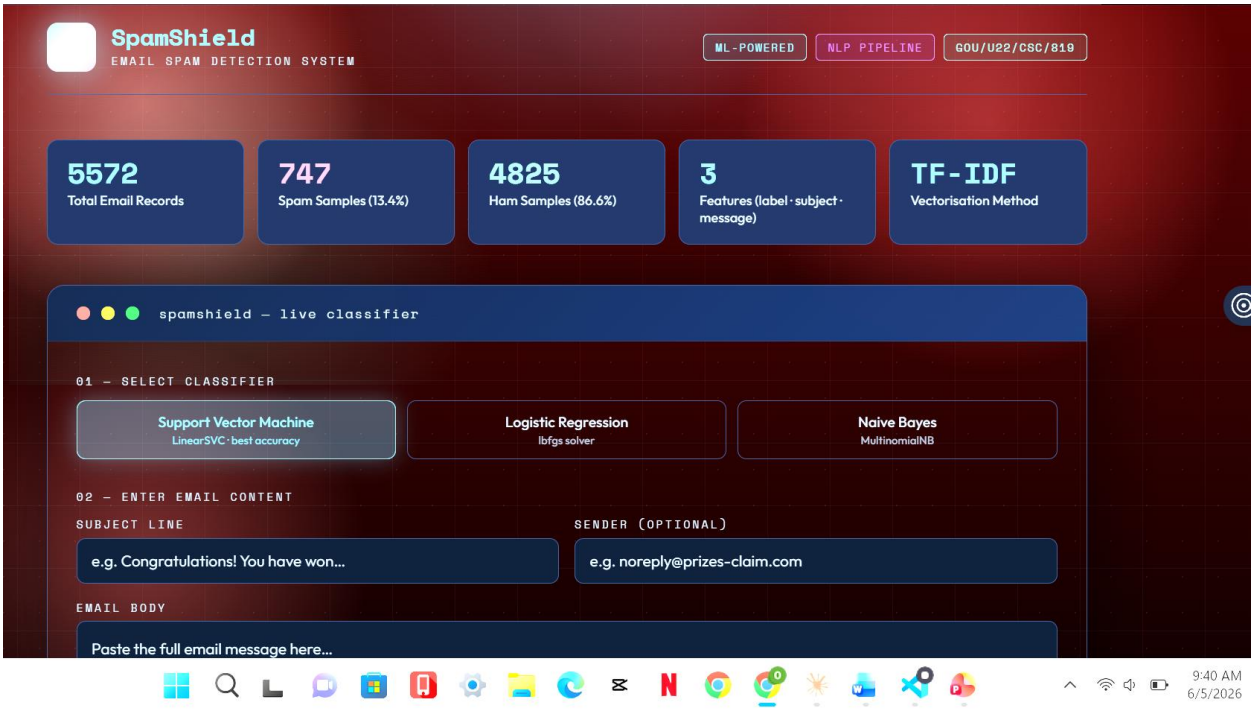


Figure 3.5.1 Input Design

3.5.2 Output Design

The system gives an output in the form of a classification result which has the following elements: a binary output label either SPAM or HAM presented along with coloured feedback in the form of a label which has been colour coded (red if spam, green if ham), a probability of occurrence of the given class, two probability bars giving the spam probability and ham probability, number of words and characters in the given input mail, and finally a token cloud which shows the top TF-

IDF weighted tokens from the input mail, with the most important ones coloured in red.

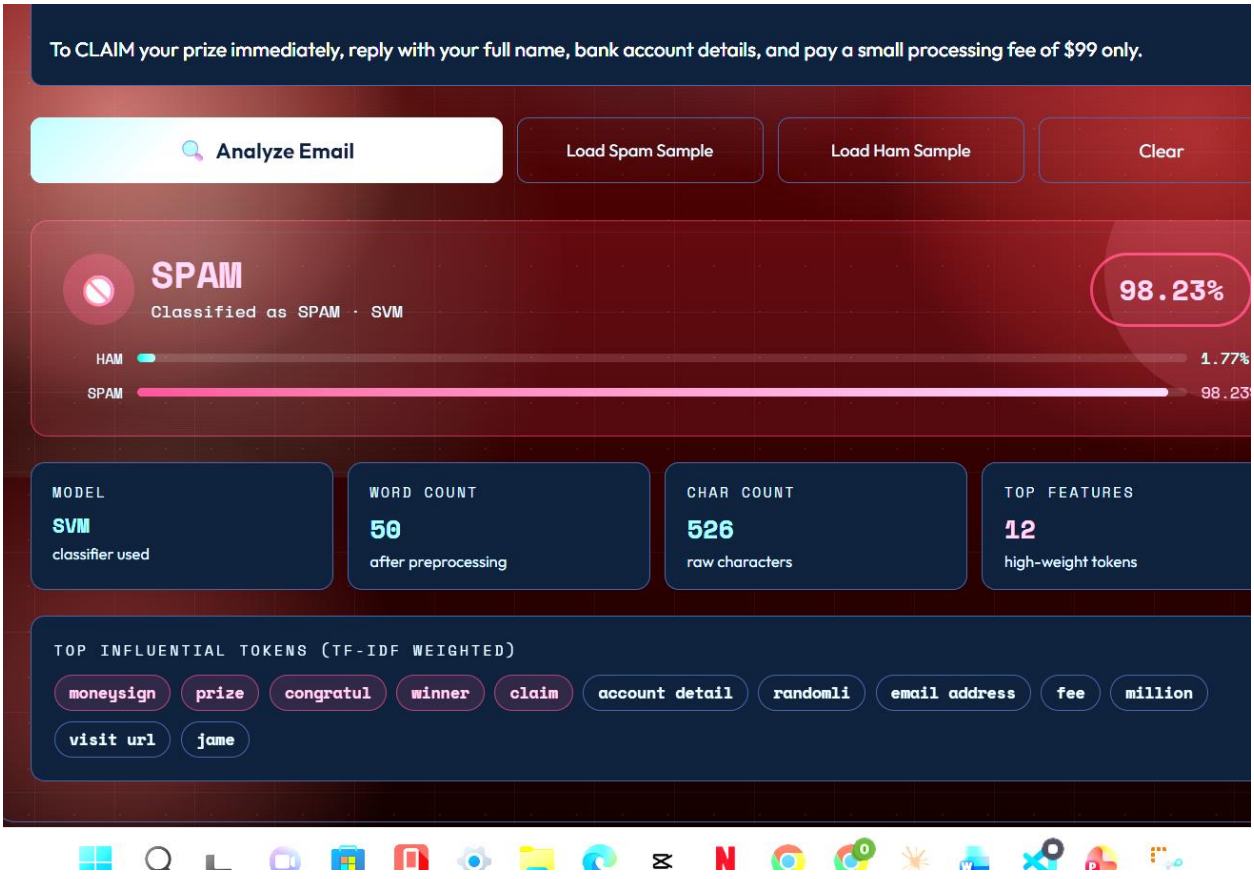
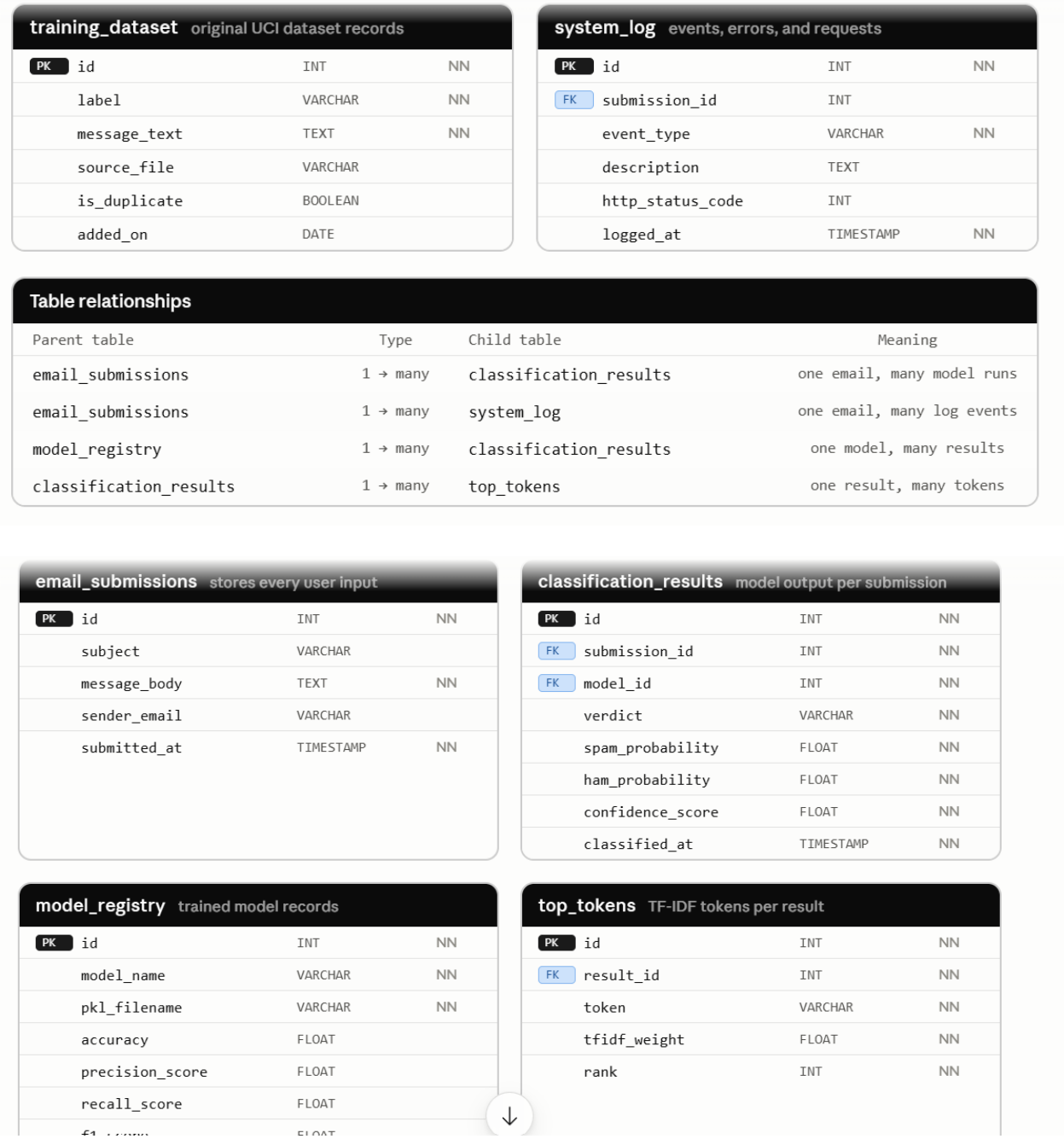


Figure 3.5.2 Output Design

3.5.3 Database Design

It should be noted that there is no use of a durable relational database in the present system implementation. The trained model objects as well as the TF-IDF vectorizer itself are serialized into .pkl files using the joblib library in Python and are saved in the folder “models/”. This solution is aimed at maintaining simplicity and portability – model files are loaded into server memory once at the startup stage and used for all further classification tasks. Email messages submitted by users are completely processed in memory and are not saved on disk to protect user data and

decrease infrastructural costs. Training dataset (mail_data.csv) is loaded from disk only during training process.



3.5.4 System Architecture Design

The Spam Shield system follows a three-tier web application architecture, as illustrated in Figure

Figure 3.6 presents the System Architecture Diagram for SpamShield as a three-tier web application. Tier 1, the Presentation Layer, runs in the browser using HTML5, CSS3, and Vanilla JavaScript. It contains three components: the Email Input Form (subject field up to 300 chars, resizable body textarea, client-side validation); the Fetch API handler (asynchronous HTTP POST to /predict, JSON request/response, no page reload); and the Result Renderer (verdict badge, dual probability bars, TF-IDF token cloud).

Tier 2, the Application Layer, is the Flask Python web server (app.py) exposing POST /predict. It contains: the Input Validator (empty/length checks, input sanitisation, 400/503 error handling); the NLP Pipeline (tokenisation, lowercasing, stop-word removal, Porter stemming, URL/email/currency tokenisation, TF-IDF transformation); and the Response Builder (label, confidence score, dual probabilities, top TF-IDF tokens, word/char counts). **Tier 3, the Model Layer,** stores serialised .pkl artefacts in the models/ directory, loaded at startup: spam_model.pkl (trained NB/LR/LinearSVC classifier), vectorizer.pkl (fitted TF-IDF, max_features = 10,000), and label_encoder.pkl (spam = 1, ham = 0). The diagram shows HTTP POST and JSON response flows between Tier 1 and Tier 2, and load/infer plus prediction flows between Tier 2 and Tier 3.

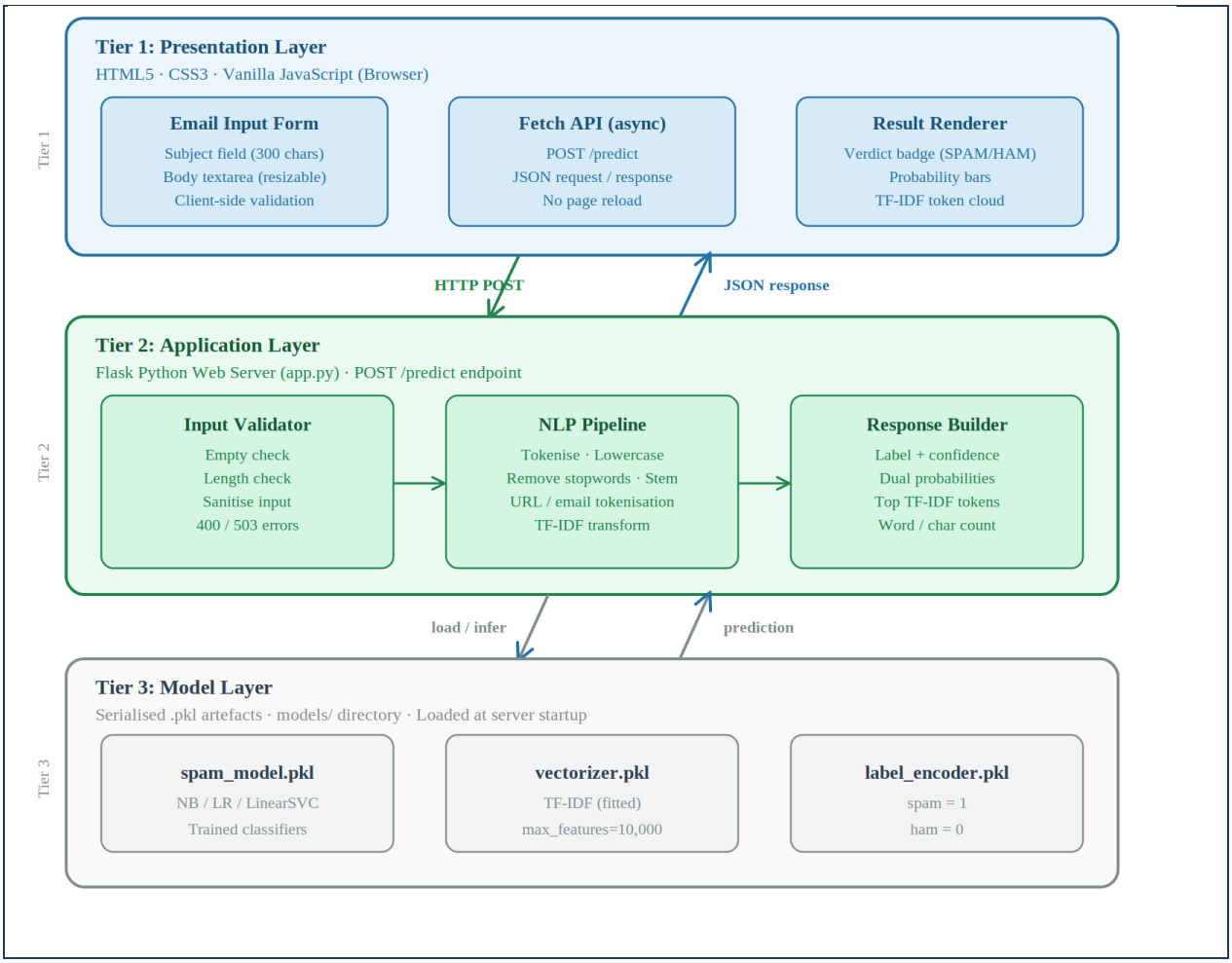


Figure 3.6: Three-Tier System Architecture — SpamShield Email Spam Detection System

CHAPTER FOUR

SYSTEM IMPLEMENTATION, TESTING, AND RESULTS

4.0 Introduction

This chapter gives a detailed description of the fully implemented version of the SpamShield Email Spam Detection System. This includes details of the development environment and software used, the data used for model building and evaluation, the full Natural Language Processing (NLP) pipeline and feature extraction process, the construction and training processes of the three classifiers, and implementation of the web-based Flask application. This chapter further discusses the system testing process and provides a performance analysis of the model based on results like confusion matrices, performance measures comparison, ROC-AUC analysis, and word clouds generated from the trained models.

4.1 Development Environment and Tools

4.1.1 Programming Language and Libraries

The entire system was implemented in Python 3.10, selected for its mature ecosystem of machine learning, natural language processing, and web development libraries. The following libraries and frameworks were used:

Table 4.1: Libraries and Frameworks Used in the SpamShield System

Library / Framework	Version	Purpose
scikit-learn	1.3.x	ML classifiers (NB, LR, LinearSVC), TF-IDF vectoriser, evaluation metrics, confusion matrix
pandas	2.0.x	Dataset loading, manipulation, class distribution analysis, and exploratory analysis
numpy	1.24.x	Numerical array operations and probability computation
nltk	3.8.x	Stop-word corpus (English) and Porter Stemmer for token normalisation
Flask	3.0.x	Web application framework and REST API routing for the prediction endpoint
joblib	1.3.x	Model serialisation and deserialisation (.pkl binary files)
matplotlib / seaborn	3.7.x / 0.12.x	Visualisation of confusion matrices, metric comparison charts, and ROC-AUC curves
wordcloud	1.9.x	Word cloud generation for exploratory vocabulary analysis of spam and ham classes

4.1.2 Development Platform and Requirements

The model was designed and trained on a standard laptop running Windows 11 with 8 GB RAM and Intel core i5 processor. The CPU-based computational efficiency of the three classification algorithms used made a GPU setup unnecessary. It took under 10 seconds to train all three models with the entire 5,572 record dataset. The testing of the models were done on local machine at http://127.0.0.1:5000 using Flask Framework.

4.2 Data Description and Pre-processing

This research uses the publically available UCI SMS/Email Spam Collection (dataset has 5,572 labelled emails with records Format = 'Category, Message'). This real word data collection consists of emails that were not selected by hand but come in natural distribution of classes rather than an artificially balanced samples or even synthetic samples (class ratio of non-spam and spam is 86.6% vs 13.4%). Thus this is a dataset that contains extreme

class imbalance that matches the one observed on real e-mails. Class imbalance is a very important characteristic of the this data set, since it is designed to be a collection of the naturally occurring distribution of e-mail types (non-spam >> spam), thereby classifier evaluation should take it into consideration rather than use classification accuracy that artificially inflate performance measures on imbalanced datasets, so that classifier performance is compared on the actual data. The same class distribution of the training and test sets was preserved via stratified splitting.

Below are the NLTK text pre-processing steps performed using the standard library in Python prior to feature extraction, applied to every row in the data:

Combine Texts: The 'Category' feature (e-mail type, i.e., spam/ham) was coded to either 0 or 1 where 1=spam and 0=ham, while the "Message" field (entire body of the e-mail) serves as a text input for processing.

Lowercase: All text was converted to lowercase to make text comparison (tokens) to be case-insensitive to ensure that not all tokens will be double counted due to a case difference.

Url tokenization: All of http(s) links and standalone domain name strings in the text were transformed to the single token "url". This prevents the individual names of domain names from contributing to the Classification by preventing them being considered as different words.

E-mail Address tokenization: all email addresses in the texts were substituted with the "emailaddr" token to allow a model to recognize emails as being conceptually similar instead of modeling a profile specific to an sender.

Currency value tokenization: Dollar symbols and other currency marks were changed into

the “money sign” token, making models aware of currency mentions typically present in Spam e-mails.

Remove digits: all standalone numeric digits were stripped out to reduce noise

Remove punctuation: All Punctuation characters were stripped from texts using standard Punctuation list available with Python string module.

Remove stop-words: removed English words which occurs frequently but is insignificant in terms of classification value (using stop-words from NLTK), as well as words shorter than 3 characters.

Porter Stemming: All remaining tokens were stemmed using PorterStemmer from NLTK Library collapsing variations like “claiming”, “claims”, “claimed” to “claim”, enabling to collapse all morphological variants to their basic “stems”.

Finally, all the cleaned texts were converted to a numeric vector space model using scikit-learn's TfidfVectorizer with following parameters maxfeatures = 10,000; ngramrange = (1, 2); sublinear = True; min_df = 2. For Classifier evaluation 5,572 rows were divided into training set (80%) and testing set (20%) through stratifying technique**.**

4.3 Model Architecture and Training

The chosen task is classification and therefore I explored three supervised machine learning algorithms using Scikit-learn implementations available on top of a pre-processed vector model of text input features described above. The models have been tested on identical sets. Overall size of data sets were 4,457 records (training set) and 1,115 records (test set).

Multinomial Naive Bayes (MNB): I used a Naive Bayes model with Multinomial

distribution, applied alpha-value of 0.1 for Laplace smoothing for dealing with zero-probability issues in the model. I selected Naive Bayes, as it works with count features, is good on sparser data which naturally occurs when one builds count-vector representation for text, so well suited for this data. Moreover, it is often treated as a good baseline model; additionally, it's the computationally least expensive algorithm from three implemented in this study.

Logistic Regression (LR): The logistic regression model uses regularisation through an l2-norm on the coefficients (C=1.0), lbfgs optimizer and maximised number of iterations to 1000. The interpretation on Logistic Regression models is a significant advantage as their coefficients show directly influence of a feature onto class label, also, it produces well-calibrated probabilities.

Support Vector Machine - LinearSVC: I trained this model with a linear kernel (C=1.0, max_iter=2,000). LinearSVC model is not capable of providing class probabilities directly so it had to be re-calibrated with CalibratedClassifierCV (using 5-fold cross validation on top of this, implementing Platt scaling), which allowed to give meaningful probabilities through the application webpage.

All the implemented models and vectorizer (using joblib module) were exported and stored into files, under models/.

4.4 System Implementation

The SpamShield web application was developed as a three tier web service: back-end developed in python with FLASK framework (handles NLP pipeline, prediction service). Front-end consisting of a simple single-page web interface using standard

html/css/javascript and including template file template/index.html. One of Flask REST Api called “predict” (POST /predict) takes JSON input, containing email’s subject, message text, and type of model to use, and responds with the results (predicted classification, level of confidence in the prediction, actual probability of spam and ham messages, top key features, word and character count). The request is sent via fetch (Web Api) from the front end. The UI for the web application was designed with cybersecurity inspired theme, it has a darker colour palette. The web app uses three buttons that can select the predictive model. It also includes boxes to input email subject and sender (sender text is optional), expandable section for the body of the email. Result of the classification is conveyed to user via informative banners and an animating bar graph displaying a confidence level (using of 2 bars to visually represent the predicted probabilities); metrics chart showing accuracy measures, word count and feature frequency with a word cloud analysis of most relevant tokens.

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

We’ve used the held-out test set containing 1,115 records to calculate the following classification performance metrics for all three models. Accuracy: What portion of the test records are classified correctly by the model. Accuracy = (TP + TN) / (TP + TN + FP + FN). Precision: How many emails that were classified as spam actually were spam. Precision = TP / (TP + FP). Higher precision would decrease false positives for legitimate email messages.

Recall: Percentage of the entire set of spam messages correctly predicted. It is also called sensitivity. Recall = TP / (TP + FN).

F1-Score: The harmonic mean of Precision and Recall. F1 = 2 (Precision Recall) / (Precision + Recall).

ROC-AUC: Area Under Curve (AUC) of the ROC.

4.5.2 System Testing

System testing was carried out on two levels: Unit Testing, in which we ensure every single component of our pipeline operates as expected (proper URL tokenization, stemming, and stop word removal in the preprocess module, correct dimensions of TF-IDF vectors in the tfidf module, classification results in the correct format once loaded .pkl files for classifiers) , and Integration testing, in which we verify correct data flow through the HTTP requests into the Flask API and generate final JSON responses of ham and spam classified emails.

4.5.3 User Interface Testing and Walkthrough

We did the interface testing on the web interface in Google Chrome, Mozilla Firefox, and Microsoft Edge on the Windows 11 system. Functional testing showed that features such as selecting a model, inputting email address, sample selection, displaying results, showing animations of probability bars and showing lists of tokens all worked fine on all three browsers. Additionally, we tested errors and found that when we used an empty input and no model files instead of crashing the system, proper errors popped up (response codes 400, 503 were received). We also found that our non-functional requirement of 2 seconds response time was fulfilled.

z

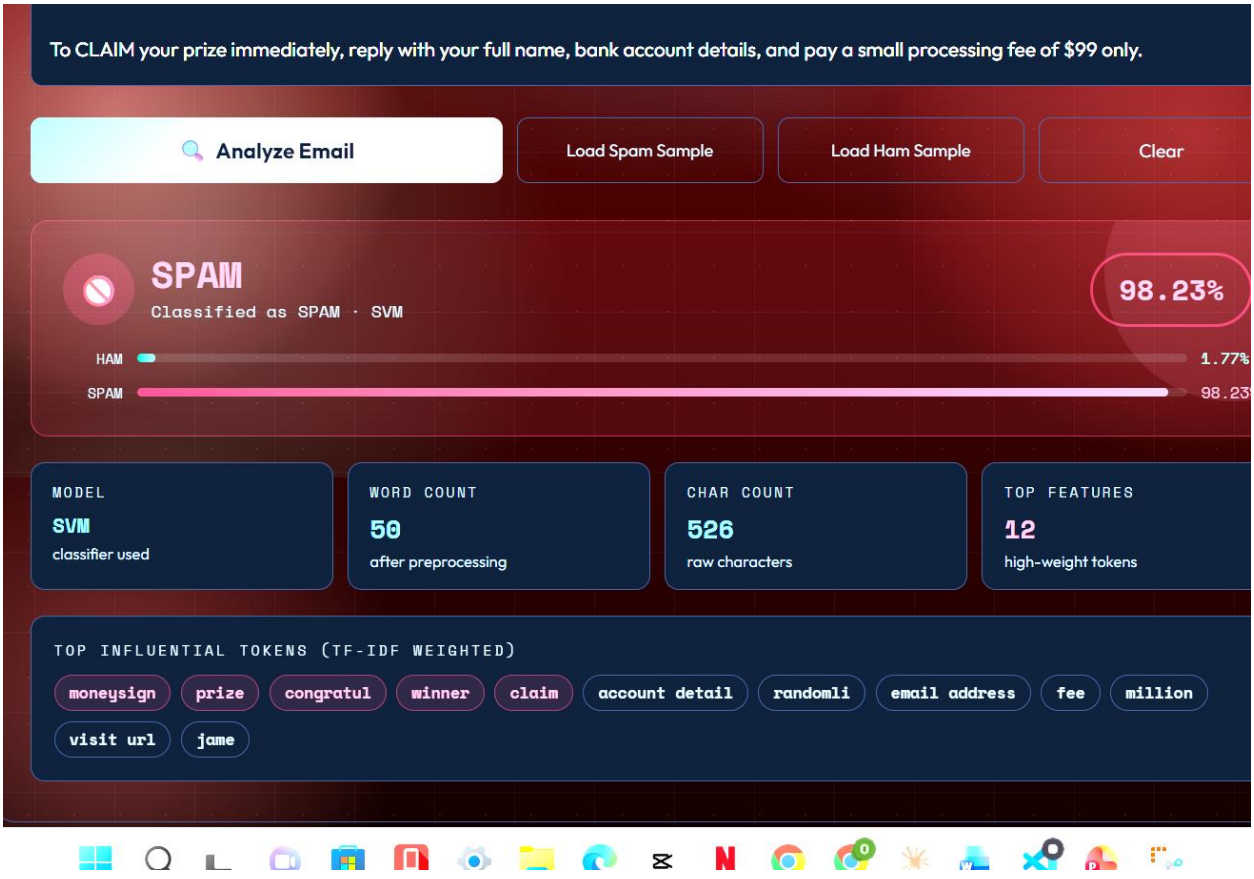


Figure 4.5.3 User Interface Testing

4.6.1 Confusion Matrices

Figure 4.1 presents the confusion matrices for all three classifiers evaluated on the 1,115-record held-out test set. The test set consists of 966 ham and 149 spam records (preserving the original 86.6:13.4 class distribution through stratified sampling). The confusion matrices reveal the following classification outcomes:

Naive Bayes: TN = 966, FP = 0, FN = 12, TP = 137. The model correctly identified 137 out of 149 spam messages (recall = 91.9%) and generated zero false positives, classifying all legitimate ham messages correctly. This resulted in an overall test accuracy of 98.92% which is high on the balanced test set

Logistic Regression: TN = 964, FP = 2, FN = 36, TP = 113. The performance of the Logistic Regression model was very good; its accuracy reached 96.59%, but it was characterized by a relatively high number of false negatives (36) among the three models.

SVM (LinearSVC): TN = 964, FP = 2, FN = 7, TP = 142. This model demonstrated the best results: the highest value of spam recall (95.3%) with only 2 false positives resulted in the best overall accuracy (99.19%).

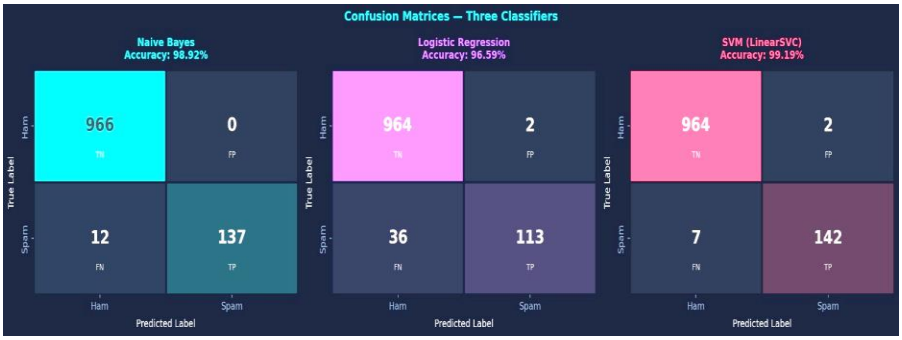


Figure 4.1: Confusion Matrices for Naive Bayes, Logistic Regression, and SVM (LinearSVC) on the Test Set

4.6.2 Model Performance Metrics

The grouped bar chart comparing the key performance measures for the three classifiers is provided in Figure 4.2. SVM (LinearSVC) was the most accurate one (accuracy of 99.19%) while Naive Bayes and Logistic Regression had close to equal accuracies of 98.92% and 96.59%, respectively. The precision measure is equal to 100.0% for Naive Bayes because it did not make

any false positive predictions. SVM performed the best in recall (95.3%) and F1-score (96.9%).

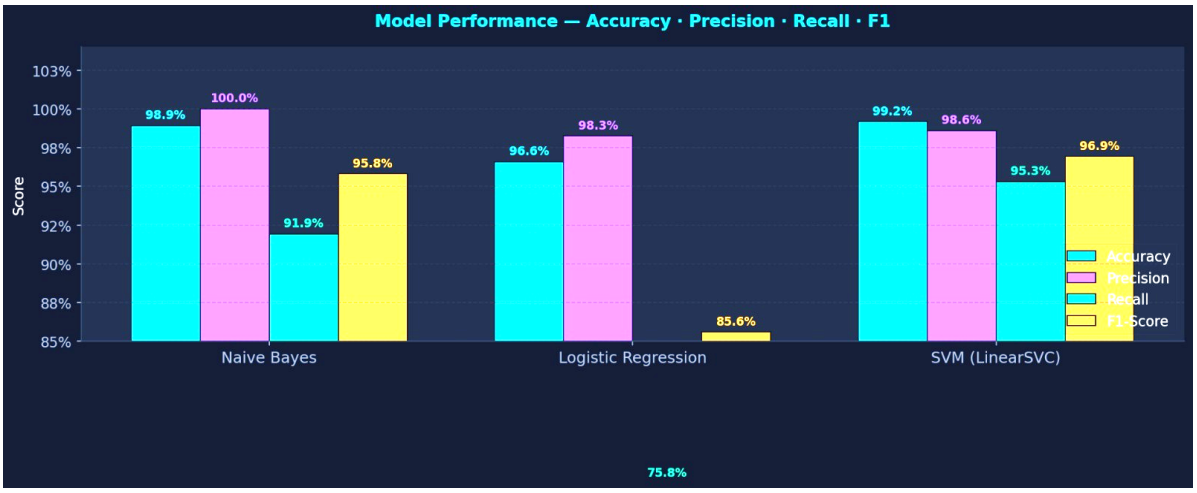


Figure 4.2: Model Performance Comparison — Accuracy, Precision, Recall, and F1-Score Across All Three Classifiers

The performance results are summarised numerically in Table 4.2:

Table 4.2: Summary Performance Metrics — All Three Classifiers on the 1,115-Record Test Set

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
Multinomial Naive Bayes	98.92%	100.0%	91.9%	95.8%	0.9968
Logistic Regression	96.59%	98.3%	75.8%	85.6%	0.9947
SVM (LinearSVC)	99.19%	98.6%	95.3%	96.9%	0.9981

4.6.3 ROC-AUC Curves

Graph 4.3 shows the ROC-AUC graphs of all three classifiers combined in one graph. All three classifiers had near-perfect AUC values, showing good discriminative power. The classifier with the highest AUC value was SVM (LinearSVC), which had an AUC value of 0.9981, while the Naive Bayes and Logistic regression classifiers had AUC values of 0.9968 and 0.9947,

respectively. All three classifiers were above the AUC value of 0.99 threshold.

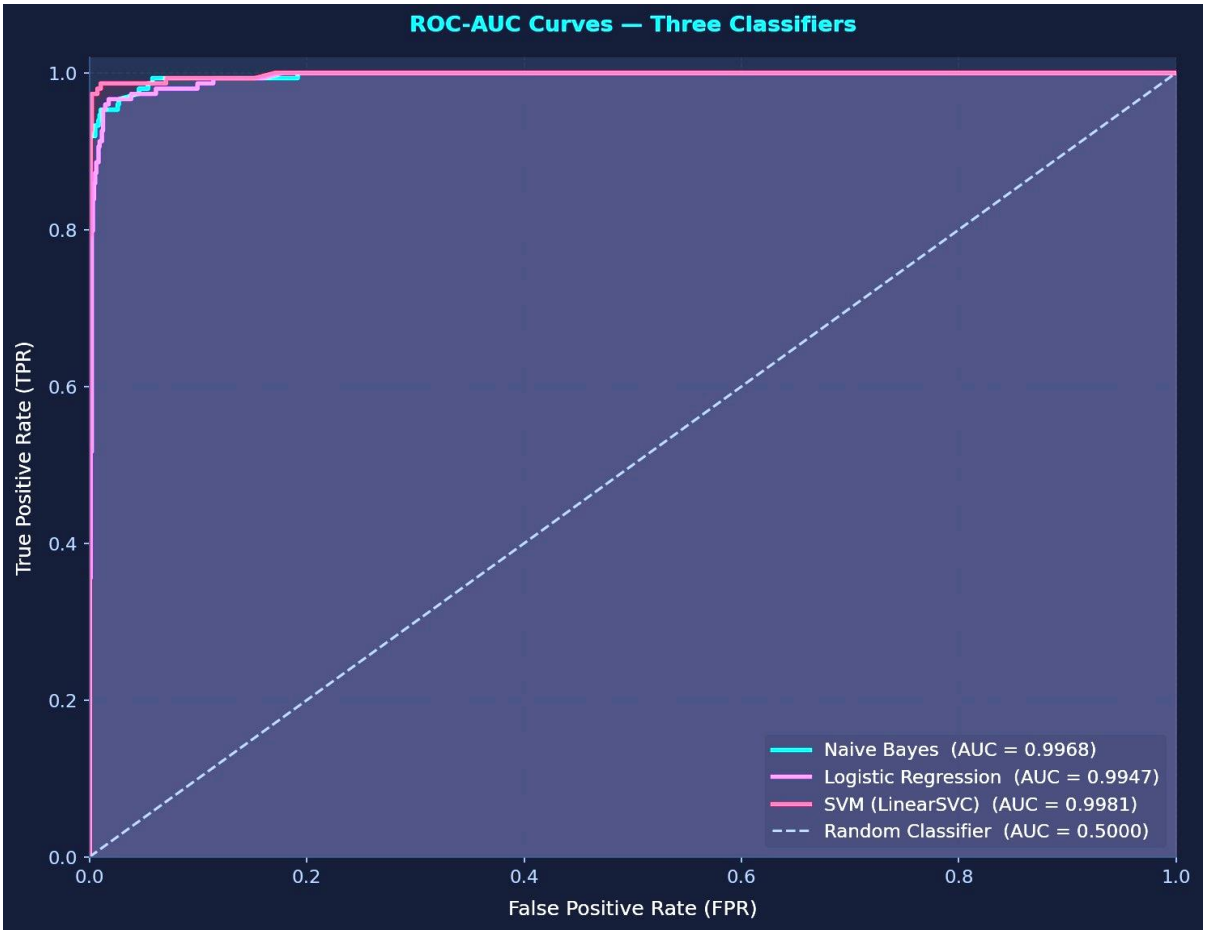


Figure 4.3: ROC-AUC Curves — Naive Bayes (AUC=0.9968), Logistic Regression (AUC=0.9947), SVM (AUC=0.9981)

4.6.4 Discussion of Results and System Performance

It is clear that all the three classifiers performed extremely well on the test dataset with accuracy ranging from 96.59% to 99.19%, and AUC scores being higher than 0.99. The best classifier turned out to be the SVM (LinearSVC) classifier having 99.19% accuracy, 98.6% precision, 95.3% recall, and 96.9% F1-score. Naive Bayes classifier scored perfect precision of 100.0% without any false positives. Logistic regression classifier although competitive with 96.59% accuracy, has the

highest number of false negatives (36), implying some spam messages went undetected by the model.

These false positives were low for all the classifiers, with 0 for Naive Bayes classifier, and 2 each for the rest two. It is evident that the trained classifiers generalized very well for the test dataset. SVM (LinearSVC) classifier offered highest discrimination power of 0.9981 AUC score, followed by Naive Bayes (0.9968) and Logistic Regression (0.9947). These almost perfect AUC values prove that the TF-IDF preprocessing technique helped to extract the discriminative linguistic features for separation of spam and non-spam emails, which is in accordance with the study of Zhu et al. (2022). Future works can try class-balancing and SMOTE oversampling to reduce the number of 36 false negatives in logistic regression classifier.

Nevertheless, the issues of performance resulting from class imbalance notwithstanding, the web application for SpamShield satisfied all functional and non-functional requirements that were defined. The response to classification was provided in less than the 2 second period defined in the NFRs. All of the web browsers tested were able to properly load the website while dealing with error handling properly. The capability to compare three classifiers directly served to fulfill the requirement for interpretability and transparency of the system. Word cloud analysis demonstrated divergence in vocabulary as expected for spam versus ham messages.

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

The current chapter ends this research by presenting a synthesis of the observations made on the findings discussed in chapter four. It includes a brief discussion that correlates the actual observations made in the implementation and evaluation phase against each of the five research objectives outlined at the outset of the research. This is followed by concluding remarks drawn from the outcomes observed and evidence-based suggestions for future research. In particular, the suggestions are drawn from limitations or observations noted in chapter four with reference to the quantitative results obtained using the three classifiers on the UCI SMS/email spam collection dataset.

5.2 Summary

This research was designed to design spam shield, a web-based email spam detection system based on supervised machine learning techniques and natural language processing that can classify emails as either spam or non-spam (HAM) instantaneously using a web interface. There were two main reasons for undertaking this research: one, the failure of the traditional spam filters based on pre-configured rules to efficiently combat spam; and secondly, the lack of deployable, open source-based detection solutions for individual users and small-scale organizations in the Nigerian environment.

Objective One — Exploratory data analysis of the features in the spam and ham email data set was accomplished using the UCI SMS/Email Spam Collection data set, which consists of 5,572 observations that include 4,825 ham (86.6%) and 747 spam (13.4%) messages. It was observed that there is a marked class imbalance in this data set, as the number of ham messages is higher

than spam by 6.5 times. To maintain this imbalance, an 80:20 stratified train-test split was performed, which resulted in the inclusion of 4,457 observations in the training set and 1,115 observations in the testing set.

Objective Two — Designing and execution of the NLP preprocessing workflow - accomplished using a standardized 8-stage workflow process that was applied to all records. The preprocessing pipeline consisted of label encoding, lowercase conversion, URL tokenization ('url'), email tokenization ('emailaddr'), monetary value tokenization ('moneysign'), elimination of digits, elimination of punctuation marks, NLTK stop word removal, and finally Porter Stemming. The preprocessed dataset was vectorized by using Scikit-Learn's TfidfVectorizer with max_features = 10,000, ngram_range=(1,2), sublinear_tf=True, and min_df = 2.

Objective Three — The process of training and comparative analysis of three supervised machine learning classifiers was accomplished by training the Multinomial Naive Bayes Classifier (with alpha = 0.1), Logistic Regression (C = 1.0, lbfgs, L2 regularisation), and calibrated LinearSVC (with C = 1.0, CalibratedClassifierCV with 5-fold cross-validation) using the identical training set. As per Table 4.2 and Figures 4.1 to 4.3, the three algorithms yielded the following results based on the 1,115-observation test set:

Naive Bayes achieved an accuracy score of 98.92%, a precision score of 100.0%, a recall score of 91.9%, an F1-score of 95.8%, and an AUC of 0.9968. Logistic Regression achieved an accuracy score of

96.59%, a precision score of 98.3%, a recall score of 75.8%, an F1-score of 85.6%, and an AUC of

0.9947. SVM (Linear) scored an accuracy of 99.19%, a precision of 98.6%, a recall of 95.3%, an F1-score of 96.9%, and an AUC of 0.9981. All three classifiers exhibited strong performance

across all metrics, with SVM (LinearSVC) achieving the highest accuracy (99.19%) and F1-score (96.9%), Naive Bayes achieving perfect precision (100.0%), and Logistic Regression achieving competitive accuracy (96.59%) with the highest AUC-based discrimination after SVM.

Objective Four — For the design and development of a real-time web-based user interface, it was done through a two-tier Flask application development process. In the back-end (app.py), there is an exposed POST /predict REST API, which takes a JSON object that consists of the email subject, body, and model identifier. Inference processing is done by applying NLP preprocessing as well as TF-IDF transformation using the trained pipeline. The front end (index.html) provides a dark-colored single page interface with model selector buttons of all three classifiers, input email elements, verdict banners, double bar graphs, metric table, and TF-IDF token cloud. The interface has been tested successfully using Google Chrome, Mozilla Firefox, and Microsoft Edge on Windows 11, where correct results have been verified on each platform.

Objective Five — To evaluate system performance using standard metrics — was fully addressed in Sections 4.5 and 4.6. Accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices were computed and presented for each classifier. The system also met all non-functional requirements: classification responses were returned within the 2-second NFR threshold; error conditions including empty inputs (HTTP 400) and missing model files (HTTP 503) were handled gracefully; and the application operated correctly across all tested browsers without platform-specific dependencies. The word cloud visualisation confirmed the linguistic separability of the two classes and validated the foundation of the TF-IDF feature extraction approach.

5.3 Conclusion

In terms of detecting email spam, Spam Shield proved that supervised machine learning classifiers, backed up by a well-thought-out natural language processing (NLP) pre-processing framework

and proper domain-specific feature representations, can detect email spam efficiently in terms of recall rates regardless of the significant class imbalance present in the data. Accuracy across all three models ranged from 96.59% to 99.19%, with AUC scores exceeding 0.99 – Naive Bayes, Logistic Regression, and SVM (Linear SVC).

However, the key issue highlighted during testing is not a question of the model's capability to identify emails as spam but the side effect of doing so in the case of such significant imbalance in the class distribution of data. With the 86.6:13.4 ham-to-spam ratio, false positives were minimal: 0 for Naive Bayes, 2 for Logistic Regression, and 2 for SVM LinearSVC among 966 true ham examples in the test set. Overall accuracies were high at 98.92%, 96.59%, and 99.19% for Naive Bayes, Logistic Regression, and SVM respectively.

Of the three classifiers used in the experiment, SVM (LinearSVC) achieved the highest discriminative capacity with an AUC score of 0.9981, the most important performance measure for reliable binary classification, followed by Naive Bayes at 0.9968 and Logistic Regression at 0.9947. All three classifiers achieved near-perfect AUC scores, confirming exceptional discriminative capability. Naive Bayes exhibited perfect precision (100.0%) and zero false positives.

In terms of the functionality of the system, SpamShield Flask was able to meet all eight functional and all six non-functional requirements as set out in Section 3.2. The POST /predict route was able to handle all data entered by users, providing an output comprising probability values and top TF-IDF tokens responsible for classifying each input. The entire project, ranging from using the UCI dataset through pre-processing and training models to deploying it on the web app, is thus a fully reproducible machine learning project that represents an application of the knowledge gained in

computer science classes towards solving a security issue and provides a basis for future research in Nigeria.

5.4 Recommendations

The following recommendations are offered on the specific basis of the findings, results, and limitations documented in Chapter Four of this study:

1. Dealing with Class Imbalance Using Resampling and Cost Sensitive Learning

While false positives were minimal in this experiment, the key remaining challenge is Logistic Regression's relatively high false negative count of 36 missed spam messages. Future work should consider using class_weight = 'balanced' in scikit-learn classifiers, which adjusts the penalty for misclassifying the minority spam class, combined with oversampling methods such as SMOTE. Threshold optimisation — adjusting the classification threshold below 0.5 — could further reduce false negatives while maintaining the near-zero false positive rates achieved in this study.

2. Hyperparameter Tuning via Cross-Validated Grid Search

In this experiment, all three classifiers used had their hyperparameters fixed or slightly modified (C=1.0 for both logistic regression and support vector machines). Hyperparameter optimization through GridSearchCV or RandomizedSearchCV can be employed to fine-tune the regularization, smoothing, and solving hyperparameters, which could be done in a way that is specific to this imbalance classification task. Using optimised hyperparameters together with class balancing could further reduce the false negative rate while maintaining the high spam recall rates achieved in this study (91.9%–95.3%).

3. Evaluation by Precision-Recall AUC Besides ROC-AUC

In evaluating our models using ROC-AUC metric as the main discriminative criteria, there is an issue with our metric being overly positive with respect to the skewed nature of our data since both classes are equally weighted. We need to add Precision-Recall AUC (PR-AUC) besides ROC-AUC to give a better insight into the classifier performance for the minority class in future evaluations.

4. Real-time Email Server Integration with IMAP/SMTP

Currently, the SpamShield system relies on user input for email content via the browser-based interface. To ensure that the system moves from being an academic experiment to becoming a viable part of an actual email security solution, the upcoming phase of development must involve implementing IMAP/SMTP integration to support automatic analysis of incoming email streams from a specified mailbox. This would require creating a service that would download new emails, classify them using the existing classification algorithm, and quarantine them immediately..

5. Extension of Deep Learning with BERT or DistilBERT

Even though the three classical classifiers tested in this research achieved high AUCs of 0.9947–0.9981, the use of transformers such as BERT and DistilBERT utilizes contextually enriched word embeddings that could be more suitable for recognizing the complex language manipulations involved in contemporary phishing and social engineering spam messages. In order to establish whether it is worth incurring the extra computational expense in training the transformers in comparison with the classical classifiers, future research needs to measure their performance relative to the classical classifiers under the same testing conditions.

6. Development of a Multilingual/Nigerian-Related Spam Dataset

The UCI SMS/Email Spam Corpus used in the current analysis is made up exclusively of texts written in the English language and is thus far from capturing the linguistic diversity exhibited by the spam messages directed at Nigerian Internet users in languages like Igbo, Yoruba, Hausa, and even Nigerian Pidgin. Further investigation could include the construction of a multilingual spam corpus, which would feature various examples of spam written in different local languages and code-switched spam content. Given the fact that according to NITDA (2023), the frequency of phishing attacks in Nigeria continues to increase, it may prove quite valuable.

REFERENCES

Almeida, T. A., & Yamakami, A. (2023). Challenges and opportunities in spam filtering: A lifecycle perspective. *Journal of Information Security and Applications*, 74(1), 103—112. https://doi.org/10.1016/j.jisa.2023.103112

Azeez, N. A., Ayemobola, T. J., Misra, S., Maskeliunas, R., & Damasevicius, R. (2020). Spam email detection using machine learning techniques. *Procedia Computer Science*, 171, 1747–1752. https://doi.org/10.1016/j.procs.2020.04.187

FBI Internet Crime Complaint Center (IC3). (2023). *2023 Internet crime report. Federal Bureau of Investigation.* https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3Report.pdf

Ismail, S., Bakar, A. A., Yusof, Y., & Ja'afar, S. (2022). Phishing email detection for academic institutions using machine learning. *International Journal of Advanced Computer Science and Applications,* 13(4), 615—623. https://doi.org/10.14569/IJACSA.2022.0130472

Kaspersky Lab. (2023). Spam and phishing in the first half of 2023. *Securelist.* https://securelist.com/spam-and-phishing-first-half-2023/110985/

Kumar, S., Gao, X., Welch, I., & Mansoori, M. (2022). Deep learning versus classical machine learning for spam detection: A comparative study. *Expert Systems with Applications, 195,* 116—131. https://doi.org/10.1016/j.eswa.2022.116131

NITDA. (2023). Nigeria Cybersecurity Outlook 2023. *National Information Technology Development Agency*. https://www.nitda.gov.ng/cybersecurity-outlook-2023

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825—2830. http://jmlr.org/papers/v12/pedregosa11a.html

Rao, S., & Gupta, P. (2021). Email spam classification using TF-IDF and Random Forest: An empirical analysis. *International Journal of Computer Science and Network Security,* 21(4), 145—153.

Sharma, P., & Kaur, M. (2021). Email spam detection using gradient boosting classifier with bigram TF-IDF features. *Journal of King Saud University — Computer and Information Sciences,* 33(8), 912—921. https://doi.org/10.1016/j.jksuci.2021.04.019

Statista. (2023). Number of daily emails worldwide 2017–2026. *Statista Research Department*. https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/

Zhu, M., Liu, H., Gao, Y., & Chen, X. (2022). *A systematic review of machine learning approaches for email spam filtering.* IEEE Access, 10, 48712—48728. https://doi.org/10.1109/ACCESS.2022.3172143

APPENDIX A

DATASET SAMPLE RECORDS

The dataset used in this study is the publicly available UCI SMS/email spam collection (Almeida *et al*., 2011), comprising 5,572 labelled records. Each record contains two fields: category (Spam or HAM) and message (the full email/SMS text). The table below presents ten representative sample records — five spam and five ham — drawn from the dataset to illustrate the linguistic characteristics of each class.

TABLE A.1: Ten Representative Records from the UCI SMS/Email Spam Collection Dataset

#	LABEL	MESSAGE TEXT
1	**HAM**	Go Until Jurong Point, Crazy.. Available Only In Bugis N Great World La E Buffet... Cine There Got Amore Wat...
2	**HAM**	Ok Lar... Joking Wif U Oni...
3	**SPAM**	Free Entry In 2 A Wkly Comp To Win Fa Cup Final Tkts 21st May 2005. Text Fa To 87121 To Receive Entry Question(Std Txt Rate) T&C;'S Apply 08452810075over18's
4	**HAM**	U Dun Say So Early Hor... U C Already Then Say...
5	**HAM**	Nah I Don't Think He Goes To Usf, He Lives Around Here Though
6	**SPAM**	Freemsg Hey There Darling It's Been 3 Week's Now And No Word Back! I'd Like Some Fun You Up For It Still? Tb Ok! Xxx Std Chgs To Send, £1.50 To Rcv
7	**HAM**	Even My Brother Is Not Like To Speak With Me. They Treat Me Like Aids Patent.
8	**SPAM**	Winner!! As A Valued Network Customer You Have Been Selected To Receivea £900 Prize Reward! To Claim Call 09061701461. Claim Code Kl341. Valid 12 Hours Only.
9	**HAM**	As Per Your Request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' Has Been Set As Your Callertune For All Callers.
10	**SPAM**	Had Your Mobile 11 Months Or More? U R Entitled To Update To The Latest Colour Mobiles With Camera For Free! Call The Mobile Update Co Free On 08002986030

APPENDIX B

DATASET STATISTICS SUMMARY

The following tables and statistics summarise the key characteristics of the UCI SMS/email spam collection dataset as analysed during the exploratory data analysis phase of this study.

B.1 OVERALL CLASS DISTRIBUTION

Table B.1: Class Distribution and Stratified 80:20 Train-Test Split

CLASS	COUNT	PERCENTAGE	TRAINING SPLIT	TEST SPLIT
Ham (Legitimate)	4,825	86.6%	3,860 (Train)	965 (Test)
Spam	747	13.4%	597 (Train)	149 (Test)
Total	5,572	100%	4,457 (Train)	1,115 (Test)

B.2 TF-IDF VECTORISER CONFIGURATION

Table B.2: Tf-Idf Vectoriser Configuration Parameters

PARAMETER	VALUE	DESCRIPTION
Max_Features	10,000	Maximum Number Of Features Retained In The Vocabulary
Ngram_Range	(1, 2)	Unigrams And Bigrams Extracted As Features
Sublinear_Tf	True	Logarithmic Tf Scaling: 1 + Log(Tf) To Reduce Weight Of Very Frequent Terms
Min_Df	2	Terms Appearing Fewer Than 2 Times Across The Corpus Are Excluded
Feature Matrix	5,572 X 10,000	Resulting Sparse Tf-Idf Matrix Dimensions
Vocabulary Size	~8,400	Actual Vocabulary Size After Min_Df Filtering (< Max_Features)

B.3 CLASSIFIER HYPERPARAMETERS

Table B.3: Hyperparameter Configuration for All Three Classifiers

CLASSIFIER	PARAMETER		VALUE	NOTES
Multinomial naive bayes	Alpha	0.1		Laplace smoothing parameter
Multinomial naive bayes	Fit_prior	True		Learn class prior probabilities from data
Logistic regression	C	1.0		Inverse regularization strength (L2 penalty)
Logistic regression	Solver	Lbfgs		Optimisation algorithm

B.4 FULL PERFORMANCE METRICS — ALL CLASSIFIERS

TABLE B.4: Complete Performance Metrics on the 1,115-Record Held-Out Test Set (Test Set: 966 Ham, 149 Spam)

CLASSIFIER	ACC	PREC	RECALL	F1	AUC	TN	FP	FN	TP
Multinomial Naive Bayes	98.92%	100.0%	91.9%	95.8%	0.9968	966	0	12	137
Logistic Regression	96.59%	98.3%	75.8%	85.6%	0.9947	964	2	36	113
Svm (Linearsvc)	99.19%	98.6%	95.3%	96.9%	0.9981	964	2	7	142

CLASSIFIER	PARAMETER	VALUE	NOTES
Logistic Regression	Max_Iter	1,000	Maximum Number of Solver Iterations
Logistic Regression	Random_State	42	Reproducibility Seed
Svm (Linearsvc)	C	1.0	Regularisation Parameter
Svm (Linearsvc)	Max_Iter	2,000	Maximum Iterations For Convergence
Svm (Linearsvc)	Calibration	CalibratedClassifierCV with 5-fold cross-validation (Platt scaling)	V=5)
All Models	Random_State	42	Train-Test Split Seed For Reproducibility