# A paper with text and images Description automatically generated

# Confidentiality Page

A close-up of a paper

Description automatically generated

# Library page

A close-up of a document

Description automatically generated

Acknowledgement  
I am grateful to Asia Pacific University of Technology and Innovation for providing a foundation for my academic path. I would like to express my deepest gratitude to the faculty of Computing, Engineering, and Technology for their advice and expertise during my research. My heartfelt gratitude goes to Dr. Preethi Subramanian, my supervisor, for her important input and persistent support, as well as Ms. Farhana Illiani Binti Hassan for her collaborative efforts. I also appreciate the support of my family, friends, and classmates. This initiative is the conclusion of these individuals' and institutions' collaborative efforts, and I am pleased to have been a part of it.

# Abstract

This paper provides a thorough investigation of the creation of a spam email categorization system using machine learning methods. The project uses a Multinomial Naive Bayes model trained on a dataset of spam and ham emails. The model performs well using preprocessing techniques like text tokenization, stemming, and TFIDF vectorization. The Streamlit-powered web application offers a simple interface for predicting email classification. While the research effectively handles the spam detection issue, it discusses certain drawbacks, such as dataset imbalance and model training's static nature. Future enhancements should include resolving dataset imbalances, investigating advanced models, and integrating real-time training for increased flexibility. The overall aim of this study is to provide an effective and accessible solution to the ongoing issue of email spam, with possible applications in user security and experience inside email communication systems.

Table of Contents

[1](#_Toc162530984)

[Confidentiality Page 2](#_Toc162530985)

[Library page 3](#_Toc162530986)

[Acknowledgement 4](#_Toc162530987)

[Abstract 5](#_Toc162530988)

[CHAPTER 1: Introduction 11](#_Toc162530989)

[Introduction 11](#_Toc162530990)

[Problem Background 12](#_Toc162530991)

[1.2.1 Inadequate spam detection: 13](#_Toc162530992)

[1.2.2 Developing spamming techniques: 13](#_Toc162530993)

[1.2.3 Communication overload: 13](#_Toc162530994)

[1.2.4 Waste of time: 13](#_Toc162530995)

[1.2.5 Bad User Experience 13](#_Toc162530996)

[1.2.6 Phishing attacks and security risks: 14](#_Toc162530997)

[1.2.7 Legal and regulatory compliance: 14](#_Toc162530998)

[1.3 Project Aim 14](#_Toc162530999)

[1.4 Project Objectives 14](#_Toc162531000)

[1.5 Scope 15](#_Toc162531001)

[1.5.1 Tasks to be Executed 15](#_Toc162531002)

[1.5.2 Possible Constraints of the Project 16](#_Toc162531003)

[1.5.3 Future application of the project which will not be part of the Project. 17](#_Toc162531004)

[1.6 Potential Benefits 18](#_Toc162531005)

[1.6.1 Tangible benefits 18](#_Toc162531006)

[1.6.2 Intangible benefits 19](#_Toc162531007)

[1.6.3 Target Users 19](#_Toc162531008)

[1.8 Project Plan 21](#_Toc162531009)

[CHAPTER 2: LITERATURE REVIEW 21](#_Toc162531010)

[2.1 Introduction 21](#_Toc162531011)

[2.2 Domain Research 22](#_Toc162531012)

[2.2.1 Natural Language processing (NLP) 22](#_Toc162531013)

[2.2.2.2 Relevant research papers on Machine Learning/Deep learning techniques for email/text classification 36](#_Toc162531014)

[2.3 Similar Systems 42](#_Toc162531015)

[2.3.1 Gmail 42](#_Toc162531016)

[2.3.2 Outlook 43](#_Toc162531017)

[2.3.3 Yahoo Mail 45](#_Toc162531018)

[2.4 Technical Research 46](#_Toc162531019)

[2.4.1 Programming language Chosen 46](#_Toc162531020)

[2.4.2 IDE (Interactive Development Environment) Chosen 50](#_Toc162531021)

[2.4.3 Libraries chosen / Tools chosen 51](#_Toc162531022)

[2.4.4 Database Management System 52](#_Toc162531023)

[2.4.5 Operating system 52](#_Toc162531024)

[2.4.6 Chapter Summary 53](#_Toc162531025)

[Chapter 3: Methodology 53](#_Toc162531026)

[3.1 Introduction 53](#_Toc162531027)

[3.2 Methodology 53](#_Toc162531028)

[CHAPTER 4: DESIGN AND IMPLEMENTATION 58](#_Toc162531029)

[4.1 Introduction 58](#_Toc162531030)

[4.2 Data Collection and Data Understanding 59](#_Toc162531031)

[4.2.1 Data Collection 59](#_Toc162531032)

[4.2.2 Data Understanding 59](#_Toc162531033)

[4.2.3 Data Cleaning 61](#_Toc162531034)

[4.2.4 Exploratory Data Analysis 66](#_Toc162531035)

[4.3 Model Building 76](#_Toc162531036)

[4.3.1 Pre-Model Building steps 77](#_Toc162531037)

[4.3.2 Using Count Vectorization 78](#_Toc162531038)

[4.3.3 Using TFIDF Vectorization 83](#_Toc162531039)

[4.3.4 Employing Grid Search to find the best hyperparameters for each model and tuning them with the hyperparameters. 87](#_Toc162531040)

[5.0 Results and Discussion 91](#_Toc162531041)

[5.1 Introduction 91](#_Toc162531042)

[5.2 Model Evaluation 91](#_Toc162531043)

[5.3 Model Deployment 93](#_Toc162531044)

[5.3.1 Web-App 93](#_Toc162531045)

[5.3.2 Web-App Deployment 97](#_Toc162531046)

[6.0 Conclusion 102](#_Toc162531047)

[6.1 Critical Evaluation 102](#_Toc162531048)

[6.2 Limitation of the project 103](#_Toc162531049)

[6.3 Recommendations 103](#_Toc162531050)

[Appendices 105](#_Toc162531051)

[References 105](#_Toc162531052)

[Project Proposal Form 110](#_Toc162531053)

[Ethics Form 118](#_Toc162531057)

[Gant Chart 122](#_Toc162531058)

[Poster 123](#_Toc162531059)

[Sample Code 124](#_Toc162531060)

[Google Collab Notebook 124](#_Toc162531061)

[Pycharm 125](#_Toc162531062)

# List of Figures

[Figure 1: Visual Represenation of SVM 28](#_Toc162531153)

[Figure 2: Visual Representation of Naive Bayes 29](#_Toc162531154)

[Figure 3: Mathematical Represenation of NBM 29](#_Toc162531155)

[Figure 4: Mathematical Represenation of NBM 30](#_Toc162531156)

[Figure 5: Mathematical Represenation of NBM 30](#_Toc162531157)

[Figure 6: Mathematical Represenation of NBM 30](#_Toc162531158)

[Figure 7: Mathematical Represenation of NBM 31](#_Toc162531159)

[Figure 8: Visual Representation of Decision Tree 32](#_Toc162531160)

[Figure 9: Visual Representation of Random Forest 33](#_Toc162531161)

[Figure 10: Visual Representation of Neural Network 33](#_Toc162531162)

[Figure 11: Visual Representation of RNN 35](#_Toc162531163)

[Figure 12: Visual Representation of CRISP-DM 53](#_Toc162531164)

[Figure 13: Data import in Google Collab 59](#_Toc162531165)

[Figure 14: Data view and information 59](#_Toc162531166)

[Figure 15: Check for missing values 60](#_Toc162531167)

[Figure 16: Check and Drop Duplicate values 61](#_Toc162531168)

[Figure 17: Change variables into 0s and 1s for Label, URL, EMAIL and PHONE columns 62](#_Toc162531169)

[Figure 18: Detect all the languages of each variable in the TEXT column 63](#_Toc162531170)

[Figure 19: Remove all but English rows in the TEXT column 63](#_Toc162531171)

[Figure 20: Text preprocessing 64](#_Toc162531172)

[Figure 21: Distribution of Spam and Ham Emails 66](#_Toc162531173)

[Figure 22: Code to see visualize top 20 most used spam and ham messages 67](#_Toc162531174)

[Figure 23: visualization of top 20 most used spam and ham messages 68](#_Toc162531175)

[Figure 24: Word cloud of Spam and Ham text 69](#_Toc162531176)

[Figure 25: Word Cloud Result 69](#_Toc162531177)

[Figure 26: Average Word length between Spam and Ham Messages 70](#_Toc162531178)

[Figure 27: Code for Sentiment analysis of spam and ham text using sentiment score 71](#_Toc162531179)

[Figure 28: Sentiment Analysis result 72](#_Toc162531180)

[Figure 29: code for Chi-square statistics and heat maps of EMAIL, URL and PHONE columns against Label Column (Spam/Ham) 73](#_Toc162531181)

[Figure 30: Chi-square statistics and heat maps of EMAIL, URL and PHONE columns against Label Column (Spam/Ham) 74](#_Toc162531182)

[Figure 31: Distribution of Emails, URLs and Phone Numbers in Spam vs Ham 75](#_Toc162531183)

[Figure 32: Split the data into train and test 76](#_Toc162531184)

[Figure 33: vectorization 77](#_Toc162531185)

[Figure 34: Applied SMOTE 77](#_Toc162531186)

[Figure 35: Using Count Vectorization 77](#_Toc162531187)

[Figure 36: SVM Count Vectorization 78](#_Toc162531188)

[Figure 37: Naive Bayes using Count Vectorization 79](#_Toc162531189)

[Figure 38: Decision Tree using Count Vectorization 80](#_Toc162531190)

[Figure 39: Random Forest using Count Vectorization 81](#_Toc162531191)

[Figure 40: Using TFIDF Vectorization 82](#_Toc162531192)

[Figure 41: SVM using TFIDF Vectorization 82](#_Toc162531193)

[Figure 42: Naive Bayes using TFIDF Vectorization 83](#_Toc162531194)

[Figure 43: Decision Tree using TFIDF Vectorization 84](#_Toc162531195)

[Figure 44: Random Forest using TFIDF 85](#_Toc162531196)

[Figure 45: Grid Search for SVM 86](#_Toc162531197)

[Figure 46: Applied hypermeters for SVM 86](#_Toc162531198)

[Figure 47: Grid Search for Naive Bayes 87](#_Toc162531199)

[Figure 48: Applied hypermeters for Naive Bayes 87](#_Toc162531200)

[Figure 49: Grid Search for Decision Tree 88](#_Toc162531201)

[Figure 50: Applied hypermeters for Decision Tree 88](#_Toc162531202)

[Figure 51: Grid Search for Random Forest 89](#_Toc162531203)

[Figure 52: Applied hypermeters for Random Forest 89](#_Toc162531204)

[Figure 54: Script Ran to organize and display result of each model across different vectorization techniques and iterations. 91](#_Toc162531205)

[Figure 55: Saving the Model and Vectorizer using the Pickel Tool 92](#_Toc162531206)

[Figure 56: Import and Download Libraries and Tools 93](#_Toc162531207)

[Figure 57: Code snippet of text preprocessing function for user input 93](#_Toc162531208)

[Figure 58: Use pickle to load the vectorizer and the model 94](#_Toc162531209)

[Figure 59: Added title for the web-app and a text area for the user to input data 94](#_Toc162531210)

[Figure 60: Functions to occur if the predict button is clicked on 94](#_Toc162531211)

[Figure 61: Code to run the web-app pre deployment 95](#_Toc162531212)

[Figure 62: View of the web App 95](#_Toc162531213)

[Figure 63: Testing ham message 95](#_Toc162531214)

[Figure 64: Testing Spam Message 96](#_Toc162531215)

[Figure 65: Contents of the requirements.txt file 96](#_Toc162531216)

[Figure 66: Used the git add and commit to push code to repository. 97](#_Toc162531217)

[Figure 67: Create a new web-service 97](#_Toc162531218)

[Figure 68: Build and deploy from Git Repository selected. 98](#_Toc162531219)

[Figure 69: Select desired Git Repo from the list 98](#_Toc162531220)

[Figure 70: Unique Name Given 99](#_Toc162531221)

[Figure 71: Adding Start Command 99](#_Toc162531222)

[Figure 72: Create web service button in Render. 100](#_Toc162531223)

[Figure 73: Web App link of the deployed app in Render 100](#_Toc162531224)

[Figure 74: Live App 101](#_Toc162531225)

# CHAPTER 1: Introduction

## Introduction

In the present digital era, email communication has become an essential part of both one’s personal and professional lives. Spam, sometimes referred to as unsolicited or unwelcome emails, has become a more common problem as email usage has grown. Spam emails not only clog up our inboxes but also pose major risks including phishing scams, malware spread, and fraudulent schemes. To solve this issue, a robust spam email classification model must be developed.

Spam email, as defined by Cisco, is bulk unsolicited and unwanted junk email that is sent to any recipient on the list. Spam is frequently sent with a hidden agenda. It may be sent in massive volumes by networks of compromised computers known as botnets ("Cisco," n.d.). Spam emails hurt people, organizations, and businesses across all industries, regardless of their size or industry. They pose a severe risk to data security, productivity, and user experience, affecting financial institutions, e-commerce platforms, healthcare providers, educational institutions, and more. In 2020, the Radicati Group anticipated that 306 billion spam emails will be sent every day, accounting for more than 56% of all emails sent worldwide (Radicati Group, 2020).

This project suggests conducting research on creating a machine learning-based categorization algorithm to effectively address the spam email issue. The model is to be trained on a big dataset of labeled emails in the specified approach to differentiate between authentic and spam emails. The computer can anticipate emails that haven't been sent yet and learn trends by employing artificial intelligence and natural language processing techniques. It can also recognize characteristics specific to spam emails.

# Problem Background

In today's digital communication ecosystem, spam emails have developed as a ubiquitous and diverse threat, affecting individuals, organizations, and overall email communication efficiency. This section discusses numerous issue statements related to spam emails, offering insight on the limitations of existing spam detection tools as well as the developing approaches used by spammers. Spam has far-reaching consequences that go beyond discomfort. They include lost productivity, financial losses, and even possible cybersecurity dangers. The statements are as follows:

### 1.2.1 Inadequate spam detection:

Current spam email filtering methods frequently have trouble classifying incoming emails as real or spam. Due to having to manually go through their emails, consumers experience a substantial number of false positives and false negatives, which is inconvenient for them.

### 1.2.2 Developing spamming techniques:

Spammers are always coming up with new ways to get past standard spam filters, making it difficult to keep up with their development. This calls for the creation of more advanced and adaptable classification techniques.

### 1.2.3 Communication overload:

Spam shuts down communication channels and generates traffic that must be paid for by the service provider, the user, or in the case of a firm, the employer. Alexander Ivanov, the president of the Russian Association of Networks and Services, estimated that three years ago, spam cost Internet service providers $55 million in losses. This sum just includes transportation costs. Additionally, there are mail servers that take in and handle spam; these servers require highly paid professionals to manage them. As a result, maintaining the infrastructure has significant expenditures ("Damage caused by spam," n.d.).

### 1.2.4 Waste of time:

Any spam that makes it into a user's inbox must be manually deleted by the receiver. A person who reads 10–20 emails each day can get 160–180 spam messages in addition to their work mail. Accordingly, they will lose out on their valuable working time by spending 5–6 hours per month merely eliminating spam ("Damage caused by spam," n.d.).

### 1.2.5 Bad User Experience

Users' experiences with email might be negatively impacted by an overabundance of spam emails. Additionally, the abundance of spam emails can make users less trusting in email communication, which might result in a decrease in the usage of email as a communication medium. There can also be the situation of the loss of a crucial email that is unintentionally deleted in addition to the abundance of junk. ("Damage caused by spam," n.d.).

### 1.2.6 Phishing attacks and security risks:

Spam emails are frequently used as a distribution channel for harmful practices like these, in which victims are duped into providing personal data. Users run the risk of falling prey to these security dangers since it is difficult to effectively detect and filter such emails.

### 1.2.7 Legal and regulatory compliance:

In some circumstances, businesses may be required by law to put in place efficient spam email filtering systems in order to abide with data protection laws. If you don't, you risk legal repercussions and reputational harm (TechTarget. ,2023, July 14).

## 1.3 Project Aim

The project's goal is to provide a precise and effective method for classifying incoming emails as either ham (non-spam) or spam automatically. The goal of the project is to employ machine learning techniques and algorithms to build a strong classification model that can recognize and filter out undesirable spam emails without interfering with legitimate emails' ability to reach users' inboxes.

The project also intends to address the rising problem of spam emails and their detrimental effects on consumers, companies, and email service providers by offering a web app which can accept any new text messages and predict the legitimacy of the message by utilizing the created predictive model.

## 1.4 Project Objectives

The project's goals for classifying spam emails are as follows:

* To train and test the classification model, gather and curate a varied and representative sample of emails, including both spam and valid messages.
* Utilize data preparation techniques to clean and convert the raw email data, including tokenizing the text, eliminating irrelevant information, and extracting pertinent characteristics.
* Implement and train machine learning algorithms: A labeled dataset will be used to explore and train viable algorithms for the spam email classification model. This technique will allow the model to distinguish between spam and genuine emails in a quick and accurate manner.
* Evaluate model’s performance: The trained classification model's performance should be assessed using a different test dataset. Calculate and examine measures like accuracy, precision, recall, and F1-score to evaluate how well the system classifies emails.
* Fine tune model: Enhance the model's accuracy and effectiveness in identifying spam emails while reducing false positives and false negatives by fine-tuning and optimizing the classification model in light of the assessment findings.
* Select the top model and deploy it in the system/web-app.
* The system/web-app will allow the user to enters a fresh body of text which it will classify as either spam or a legitimate email.

Overall, completing these goals will result in the creation and effective deployment of a system for accurately classifying spam emails, which will have a positive impact on productivity, security, and user experience with email.

## 1.5 Scope

### 1.5.1 Tasks to be Executed

The features and results that the suggested system is meant to achieve are all included in the scope of the spam email categorization project. These points serve as the concrete end products of the development process and are crucial parts of the completed project. The suggested system's functionality consists of the following:

* **Spam Email Classification Model:** The main output is a machine learning model that has been properly trained and optimized and can categorize incoming emails correctly as either spam or valid. When identifying spam emails, the model should have high accuracy, precision, recall, and F1-score while reducing false positives and false negatives.
* **User Interface:** To allow users to engage with the system effortlessly, a user-friendly and intuitive interface will be built. The UI should prompt the users to type or paste text in a text field. The text should be analyzed and classified in the back end. Once evaluated, the UI should prompt the user whether the text is spam or legitimate along with the accuracy of the classification.
* **Scalability for future improvements:** To be scalable, the spam email classification system should be built to manage a large volume of email traffic and a rising number of users without affecting performance. Scalability may be done by refining the system's architecture and algorithms to process and categorize emails effectively, allowing it to scale up to perform in real-time in an email service. Furthermore, the system should be able to smoothly integrate upgrades and changes, allowing it to adapt to developing spamming strategies and user needs.

### 1.5.2 Possible Constraints of the Project

Creating a strong spam email categorization system poses a number of issues that must be handled during the project's execution. These difficulties need careful analysis and imaginative solutions to assure the system's performance. Among the most significant problems are:

* **Finding a thorough dataset**: Finding a relevant and sufficient dataset is one of the most difficult aspects of the spam email categorization endeavor. This is critical since the classification model's success is strongly dependent on the quality and representativeness of the data used for training.
* **Data Imbalance**: There may be an imbalance in the amount of spam and valid emails in the dataset used to train the classification algorithm. Because of this mismatch, the algorithm may prefer the majority class while struggling to effectively categorize the minority class (Mazumder, 2022).

In a paper conducted by Mangena Madhavan and peers, some of the possible challenges to be faced or they faced were mentioned. Noise, overfitting, missing values, and varied types of data are some of the issues mentioned that email spam filtering faces (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).

* **Noise**: Noise is interference that impacts the accuracy of feature measurements, such as shadows, bad lighting, or typos in spam letters (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).
* **Overfitting**: It happens when there are too many characteristics and too few observations, causing the classifier to become complicated and unable to correctly categorize basic patterns (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).
* **Missing values**: Missing values in the dataset make distinguishing between classes harder (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).
* **Email Variability**: Because emails come in a variety of forms and languages, it might be difficult to extract key characteristics consistently. To properly identify spam trends, the system must manage the variety in email structures, including headers, attachments, and formatting (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).

Other issues were also mentioned by Mengana Madhavan and team in their paper. One of the issues is the usage of storage space on servers, which increases prices and may necessitate the purchase of more storage. Spam emails are readily missed or mistakenly deleted, disrupting regular email communication within a business. Spam filters are used to limit the amount of unsolicited emails as well as to defend against harmful files such as viruses or ransomware.

* **Deployment of chosen model**: The student has no prior experience of deployment of model to a system. Deploying a predictive model has technological complexities that might be daunting for someone with no prior knowledge. To overcome these challenges, the student must seek advice, use online learning resources, and progressively build skill in installing machine learning models.

### 1.5.3 Future application of the project which will not be part of the Project.

The developer wishes to deploy the model as a real time spam classifier which will be able to detect incoming messages in the inbox and send it to the spam folder. This approach will be taken into consideration but is still a question mark as the student has no prior knowledge of deployment of machine learning model into a Realtime email service. Implementation of this will require the student to the service must function swiftly, process several emails at once, and not slow down the service. False predictions must also be addressed.

Furthermore, email platforms such as Outlook, Gmail, and Yahoo have implemented several types of filters to distinguish between spam and valid communications. However, these filters may occasionally wrongly reject genuine emails, resulting in a 20% loss of legitimate emails. To analyze the risk level of each email received, many frameworks and approaches are employed, such as spam limits, sender security procedures, blacklists, and whitelists (Madhavan, Pande, Umekar, Mahore, & Kalyankar, 2021).

## 1.6 Potential Benefits

The following project possesses both tangible and intangible benefits to it. Benefits that are tangible can be measured and quantified. They are employed to evaluate a task's worth. Usually, its value is fiduciary. The benefit's worth varies according to a person's ability level. However, because they are subjective, intangible advantages are far more difficult to quantify. The rewards that are intangible come from how someone feels about their project or task (Capozzi, 2017). The intangible and tangible benefits for the projects are as follows:

### 1.6.1 Tangible benefits

* Reduced spam: The research could result in the creation of a spam filter that is more effective, which will lessen the quantity of spam people get. Users would save time and effort by doing this, and there would be less chance that they would fall victim to fraud or malware infection. The time saved can be put to use on activities that will produce more.
* Security: Effective spam email classification assists in locating and excluding fraudulent emails and dangerous material, lowering the risk of intrusions and securing users' sensitive data.
* Cost savings: By effectively screening and deleting spam emails before they enter users' inboxes, organizations may reduce expenses related to storage space, network traffic, and server resources.

### 1.6.2 Intangible benefits

* Improved user experience: Users' overall email experiences will be enhanced by a dependable and effective spam email categorization system. Less spam will be sent to users, which will result in a cleaner inbox and a more efficient system for communicating.
* Increased trust in email communication: Improved user confidence in the validity of their incoming emails will result from the reduction of spam email flood. In order to establish successful communication and build enduring bonds with clients, partners, and stakeholders, trust is essential.
* Reduced irritation and stress: Users frequently experience irritation and stress as a result of the predominance of spam emails. The suggested solution will lessen this annoyance by successfully filtering out spam, resulting in a more enjoyable and stress-free email experience.
* Enhanced problem-solving abilities: The project would put the project worker in the position of having to tackle a complicated problem. One would be able to improve their problem-solving abilities, which would be beneficial for their future careers.

In conclusion, tangible advantages are real and quantifiable, frequently having an immediate financial impact, but intangible benefits are more ephemeral and tied to general experiences and perceptions, having long-term effects that may be difficult to accurately measure. When assessing the accomplishment and usefulness of a project or campaign, tangible and intangible rewards are both crucial factors to take into account. These potential advantages, both tangible and intangible, emphasize the need to create a precise and effective spam email categorization system. They respond to the demands and worries of individuals, groups, and society at large, eventually enhancing email security, productivity, and usability.

### 1.6.3 Target Users

In 2023, there will be 4.37 billion email users globally, predicts a research that has been studying email usage numbers since 1993. More than half of the world's population is represented by this, a rise of 2.7% from the year before. According to analysts, this number will continue to increase over the next several years, climbing 2.5% to 4.48 billion in 2024 and another 2.5% to 4.59 billion the following year. Global email usage is expected to reach 4.73 billion people by 2026. The number of individuals using email globally is expected to increase by 11.1% in only four years. (Lin, n.d).

The initiative to classify spam emails aims to reach a wide variety of people who often communicate via email. The following categories apply to the key target users:

* **Individual Users**: The spam email classification system has a sizable focus on individual users, such as students, workers, and owners of personal email accounts. These users frequently get a large number of emails, so an effective spam filtering system can help them clear out their inboxes and spot any security risks.
* **Small and Medium-sized Businesses**: For a variety of company activities, such as customer contacts, sales, and internal cooperation, Small/Medium businesses often rely extensively on email communication. The suggested spam email classification system can help businesses manage their email more effectively by cutting down on time spent sifting through junk and ensuring that crucial communications are not missed.
* **Large Corporations and Enterprises**: Due to the daily flood of emails that larger firms receive, they are more susceptible to spam assaults and phishing scams. Protecting sensitive data and sustaining efficient email operations need the implementation of a strong spam email categorization system.
* **Educational Institutions**: Educational institutions, such schools, colleges, and universities, deal with a significant volume of email traffic relating to administrative duties, student communication, and academic issues. These organizations may make sure that crucial emails reach their intended recipients without being blocked by spam by incorporating the spam email categorization system.
* **Government agencies and NGOs**: Email correspondence between government agencies and nonprofit organizations frequently involves sensitive and secret content. Sensitive information may be protected and dangerous data breaches can be avoided with the use of an effective spam email categorization system.
* **Email service providers:** Email service providers may incorporate the spam email categorization system into their platforms to give their consumers access to improved email security and filtering features. By integrating their services, they may increase user interest in their offerings and boost client happiness. Currently email service providers like Microsoft and Google have Spam filtering in their service but it’s accuracy varries and people are finding newer techniques to send spam filters to users.

In conclusion, the project's target consumers for classifying spam emails include people, small and large enterprises, educational institutions, governmental and non-governmental organizations, and email service providers. The project seeks to improve email communication, increase productivity, and provide a safe and effective email experience for all stakeholders by adapting to the demands of these varied users.

## 1.8 Project Plan

A Gant Chart is to be found in the Appendices Section of the document.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

The following chapter digs into a thorough examination of scholarly papers, respectable journals, conference proceedings, and books pertinent to the issue of spam email categorization. This chapter serves as a key foundation for the research endeavor, offering a complete awareness of existing knowledge and developments in the topic. The chapter seeks to uncover essential concepts, strategies, and approaches used in the domain of spam email categorization by evaluating the current corpus of research. Furthermore, it aims to indicate any gaps or areas that require further examination in order to improve the efficiency of spam filtering systems. The study intends to improve on current information and contribute to the establishment of an accurate and reliable spam email categorization system through this detailed review. The Literature Review chapter is an important element in developing the research technique and guiding decision-making throughout the project's implementation.

## 2.2 Domain Research

When undertaking spam email categorization domain research, one should seek for academic papers, research articles, conference proceedings, and journal publications that focus on the following two essential areas. They are as follows:

* Natural Language Processing (NLP) Methodologies: Papers that examine the use of natural language processing (NLP) approaches in spam email categorization, such as text preparation, feature extraction, and semantic analysis. To increase categorization accuracy, NLP approaches can assist extract useful information from email text.
* Machine learning Techniques: Papers examining the use of machine learning methods such as Support Vector Machines (SVM), Naive Bayes, Decision Trees, Random Forests, Neural Networks, and Deep Learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). various publications detail how various approaches have been utilized for spam email categorization, as well as how their performance has been evaluated.

The student found some academic papers related to the above-mentioned areas.

### 2.2.1 Natural Language processing (NLP)

A paper conducted by (Khurana, Koli, Khatter, & Singh, 2023), discusses Natural Language Processing (NLP) topics covered include essential NLP and NLG terminology, the history, applications, and current advancements in the area of NLP, datasets, techniques, and evaluation metrics used in NLP, and the obstacles involved in NLP. The study also addresses relevant work done in the current literature, including their conclusions, as well as some of the most important applications and initiatives in NLP.

The paper also discusses that Natural Language Understanding (NLU) and Natural Language Generation (NLG) are the two branches of NLP. NLU refers to the process of teaching robots to understand and interpret human language. It entails determining the meaning, intent, and context of a particular piece of text or speech. NLU's major purpose is to convert unstructured linguistic data into structured forms that machines can process. The paper also emphasizes the different terminologies for NLU. Some of the commonly used terminologies for NLU are as follows:

* **Phonology:** Phonology is a branch of linguistics that studies sound in language, especially how they are arranged and employed in various languages. It is concerned with the abstract and conceptual representations of speech sounds, as well as the rules governing their systematic structure and utilization within a certain language.
* **Morphology**: The study and examination of the inner structure and creation of words in a language is referred to as morphology. It is an important part of language processing because it deals with the laws and patterns that control how words are generated from smaller components known as morphemes.
* **Lexical**: The term "lexical" in Natural Language Understanding (NLU) refers to anything linked to words or a language's lexicon. It is a research and analysis of individual words, what they mean, and their interactions within sentences and broader language systems.

Natural Language Generation (NLG) is an important aspect of Natural Language Processing (NLP) that entails creating significant phrases, sentences, and paragraphs from an internal model. It is divided into four stages: establishing goals, planning how to attain them by assessing the circumstances and available communicative sources, and implementing the plans as text. Natural Language Generation is the inverse of Natural Language Understanding.

An NLG system includes a speaker or application, as well as a generator or software, to create text. The speaker portrays the situation, starts the procedure, and saves vital information. The generator turns the speaker's intentions into fluent sentences that are situationally suitable.

The NLG process includes tasks such as content selection, which involves selecting and including information in the set; textual organization, which involves structuring the information according to grammar and language relations; and the use of linguistic assets, such as words, idioms, and syntactic constructs, to support the realization of the information as text or voice output (Khurana, Koli, Khatter, & Singh, 2023).

In another by paper by (Priya, Nandhini, & Gnanasekaran, 2021), it discusses various applications of Natural Language Processing. According to the paper, Natural Language Processing (NLP) has several applications in a variety of industries. One of the most common applications is machine translation, in which NLP assists in bridging language barriers on the internet by translating technical documentation and support resources. Automatic Summarization is another critical application that helps with information overload by generating high-level summaries of documents and identifying emotional meanings within data. NLP is also widely utilized in Sentiment Analysis, allowing firms to watch social media and identify user opinions to aid in product development. NLP increases the completeness and quality of Electronic Health Records, assists in phenotyping patients, identifies probable healthcare delivery problems, and adds to predictive analysis for detecting high-risk patients in the field of health care. Text mining is another important application that uses NLP to extract valuable information from text, whether it is for text recognition, customer service, tailored bots, or sentiment analysis.

In addition, NLP has a significant influence in the field of education, where it parses and summarizes arguments, offers feedback to writers to help them improve their writing, and aids in the development of automated writing evaluation systems for students. Agriculture benefits from NLP through digital agriculture and precision farming, which employ data-intensive ways to increase agricultural output while minimizing environmental effect. NLP improves comprehension of the dynamic crop-soil-weather interactions in agricultural operations by evaluating data from multiple sensors, allowing for faster and more precise decision-making in the agricultural sector. Overall, the adaptability and utility of NLP are obvious in its numerous applications, where it plays an important part in current technology and addresses issues in a variety of areas (Priya, Nandhini, & Gnanasekaran, 2021).

Another paper by (Shaik et al., 2022) explores the influence of Artificial Intelligence (AI) on education, with an emphasis on Natural Language Processing (NLP) applications in the education area. The authors investigate how AI, including machine learning, deep learning, and natural language processing (NLP), might transform different elements of education, including customized learning, teaching techniques, learning management systems, and student feedback analysis.

The study also goes into the difficulties encountered while using Natural Language Processing (NLP) techniques in the educational arena. One of the most significant difficulties is connected to domain-specific language. NLP techniques must comprehend the essential aspects of the instructional setting in order to successfully categorize academic datasets or student feedback. In the educational environment, however, there is an abundance of student feedback collected from numerous sources such as surveys, questionnaires, and feedback portals, making it difficult to interpret without domain-specific knowledge. NLP approaches may fail to reveal the underlying semantic meaning of text without sufficient training and awareness of the respective domains. In response to this problem, academics in higher education have suggested domain-specific NLP models geared to subjects such as computer science or information technology. These algorithms extract tech-related abilities and develop customized course recommendation systems using techniques such as named entity recognition (NER).  
Sarcasm decoding is another key difficulty in NLP for teaching. Understanding sarcasm is essential for sentiment annotation and opinion analysis because it allows you to comprehend student ideas and perceptions about course structures and educational infrastructure. Various ways to automate sarcasm detection have been investigated in research. Rule-based techniques detect sarcasm based on key signs of sarcasm gathered as evidence, whereas statistical approaches define sarcasm based on attributes such as emotion lexicons, word embeddings, and the frequency of unusual terms. For automatic sarcasm detection, deep learning methods such as recurrent neural networks (RNNs) and long short-term memory (LSTM) models have been used.

Natural language ambiguity is also a common difficulty in NLP activities. Ambiguity can be caused by structural, syntactic, or lexical variables, and it frequently depends on context and user perception when reading a document. Emoticons and unusual characters present still another NLP issue, particularly in student feedback that expresses emotions. Deep learning models and feature extraction approaches have been investigated for analyzing cross-cultural responses, sentiment polarity, and emotions from emoticon-validated tweets. Aspect-based sentiment analysis, a relatively untapped topic in education, meets the requirement for eliciting thoughts on specific parts of feedback, extending beyond overall document-level sentiment analysis. In order to enhance classification performance when dealing with data imbalance, a typical issue in AI and NLP, strategies such as transfer learning, sampling methods, and text augmentation have been researched, particularly in resource-intensive education-related tasks. These efforts in NLP research provide prospective options for improving the analysis of student input and successfully improving educational systems (Shaik et al., 2022).

Based on the above-mentioned information, it can be concluded that the acomplishment of this project will require Natural Language Understanding. Natural Language Understanding (NLU) is concerned with the processing and comprehension of human language, which includes tasks such as classification of texts, sentiment analysis, named entity identification, and language translation. NLU approaches are employed in the context of spam email classification to evaluate the content of emails and extract key elements that aid in determining whether an email is spam or not. These criteria might include the inclusion of specific terms, patterns, or traits that are frequent in spam emails.

#### 2.2.1.1 Pipeline for Natural language processing:

Understanding the Natural Language Processing (NLP) pipeline is critical for finishing the spam email classification project since NLP contains a number of interrelated activities and techniques that are important for efficiently processing and analyzing text data. The steps for data processing are as follows:

1. **Data preprocessing**: Assuming a relevant dataset is found, the very first step to building a predictive model for text classification is data preprocessing. Eliminating punctuation, changing text to lowercase, and dealing with unusual characters or symbols may fall under this category. The steps of data processing in natural language processing are as follows:
   1. **Tokenization**: Tokenization divides the text into individual words or subwords known as tokens. Tokenization is required to break down the text into smaller components, making further processing easier.
   2. **Lowercasing**: This step is used for modifying all words into lowercase.
   3. **Stopword removal**: Stopwords are words that appear often in the language but have little or no significance. Removing stopwords reduces data dimensionality and concentrates on more informative words.
   4. **Normalization**: Normalization involves reducing words to their basic or root form. To do this, techniques such as stemming or lemmatization are utilized, which aids in decreasing word variants and combining similar terms.

d1. **Stemming**: Stemming is the process of reducing words to their basic or root form. The purpose of stemming is to reduce diverse versions of a word to a common base form so that they may be considered as the same word, even if they contain distinct inflections or suffixes.

d2: **Lemmatization**: Lemmatization's purpose is similar to stemming's, but it provides more accurate and understandable base forms by taking the word's context and dictionary meaning into account.

Lemmatization, as opposed to stemming, which uses simple principles to remove suffixes or prefixes from words to produce the root form, is based on a lexical and morphological examination of the word. Lemmatization considers the part-of-speech (POS) tag of the word and guarantees that the resultant lemma is a legitimate term contained in the language's dictionary.

e. **POS Tagging**: POS tagging is the process of assigning grammatical labels (e.g., noun, verb, adjective) to each word in the text. POS tags can help you grasp the sentence's grammatical structure and context.

f. **Named Entity Recognition (NER):** NER is a task that recognizes and categorizes named entities in text, such as the names of people, organizations, locations, and dates.

g. **Feature Extraction**: Feature extraction is the process of translating processed text into numerical representations that machine learning algorithms may use. Bag-of-words, TF-IDF, and word embeddings are typical techniques for this purpose.

1. **Machine Learning or Deep Learning Models**: Depending on the NLP goal, the processed text data is input into machine learning or deep learning models, such as text categorization, sentiment analysis, or language translation.
2. **Model Evaluation**: After the model provides predictions, post-processing processes are frequently conducted to enhance the output or transform it into a human-readable format. The accuracy and efficacy of the model are assessed using relevant measures.
3. **Deployment**: Deployment: The final stage is to put the trained NLP model to use in applications. Setting up the model in a production environment, making it available for usage, and integrating it into the target application, such as a web service, mobile app, or chatbot, are all part of this process. Deployment requires resource management, guaranteeing scalability, monitoring model performance, and dealing with changes or model refreshes as needed (Hasith, 2019).

#### 2.2.2.1 Investigation into various machine learning techniques

* **Support Vector Machines (SVM)**

A diagram of a graph

Description automatically generated

Figure 1: Visual Represenation of SVM

Support Vector Machine is a sophisticated supervised machine learning technique that is frequently utilized. SVM is mostly used for classification jobs, but it may also be utilized to solve regression issues. SVM is a classification technique that uses a hyperplane to divide data points into multiple groups while reducing misclassification error. The fundamental purpose of SVM is to find a hyperplane in a high-dimensional space that optimally separates distinct classes of data points. In a binary classification scenario, this hyperplane serves as a decision border, ideally separating data points from one class from those from the other.

The key notions for SVM are as follows:   
a. Hyperplane: A hyperplane is a simple straight line in 2D space that may be represented by the equation: w \* x + b = 0, where 'w' is the vector of weights (typical to the hyperplane) and 'b' is the bias factor. The hyperplane produces a hyperplane in higher dimensions.

b. Support Vector: Support Vectors are the data points nearest to the decision border (hyperplane). They are the crucial data points that define the hyperplane's location and orientation.

c. Margin: The margin is the distance between the support vectors of distinct classes and the decision border. SVM tries to maximize this margin since it increases confidence in the model's ability to make generalizations to previously unknown data (“Support Vector Machine (SVM) algorithm*”,* n.d*).*

* **Naive Bayes**

A graph and diagram of a function

Description automatically generated with medium confidence

Figure 2: Visual Representation of Naive Bayes

Naive Bayes is a straightforward and commonly used machine learning technique, notably for text categorization and natural language processing. Despite its simplicity, it may be very successful in some instances, particularly when working with high-dimensional data such as text. It is termed 'Nave' because it demands a strict independence condition between input variables. As a result, Simple Bayes or Independence Bayes are more appropriate names (S, 2020).

The purpose of the Nave Bayes Classifier is to compute conditional probability:

A black text on a white background

Description automatically generated

Figure 3: Mathematical Represenation of NBM

Ck is the probability of each of the K potential outcomes or classes.

Let x equal (x1,x2,...,xn). Using the Bayesian theorem, it can be obtained:

A black text on a white background

Description automatically generated

Figure 4: Mathematical Represenation of NBM

The joint probability is denoted as:

A math equations on a white background

Description automatically generated

Figure 5: Mathematical Represenation of NBM

Assuming all x characteristics are mutually independent, it can be obtained that:

A mathematical equation with a number of letters

Description automatically generated with medium confidence

Figure 6: Mathematical Represenation of NBM

As a result, the final formula for the Nave Bayes Classifier may be expressed as:

A math equations on a white background

Description automatically generated

Figure 7: Mathematical Represenation of NBM

(S, 2020).

* **Decision Trees**

A decision tree is a non-parametric controlled learning approach that may be used for classification as well as regression applications. It has a tree structure that is layered and consists of a root node, branches, internal nodes, and leaf nodes. By undertaking a greedy search to determine the best split points inside a tree, decision tree learning applies a divide and conquer technique. This dividing procedure is then repeated top-down and recursively until all or the majority of entries have been categorized under particular class labels. The decision tree's complexity determines whether or not all data points are classed as homogeneous sets. Smaller trees can more readily achieve pure leaf nodes, or data points in a single class (“what is decision tree”, n.d).

A diagram of a root node

Description automatically generated

Figure 8: Visual Representation of Decision Tree

The figure above depicts a decision tree that begins with a root node and has no incoming branches. The root node's outgoing branches then feed into the internal nodes, also known as decision nodes. Both node types evaluate the available characteristics to generate homogeneous subsets, which are designated as leaf nodes or terminal nodes. The leaf nodes reflect all of the dataset's conceivable outcomes (“what is decision tree”, n.d).

* **Random Forests**

A random forest is a machine learning approach for solving regression and classification issues. It makes use of ensemble learning, a technique that combines several classifiers to solve complicated problems (Mbaabu, 2020).

Random Forest is a famous and widely used method among Data Scientists. It constructs decision trees from several samples and uses the largest number of votes for classification and average for regression. To create predictions or classifications, it employs an ensemble of several decision trees. The random forest technique produces a more accurate result by mixing the outputs of various trees (R, 2023).

A diagram of a question

Description automatically generated

Figure 9: Visual Representation of Random Forest

An article by Sruthi ER explains Random Forest with an analogy. The figure above explains the analogy. A student called X wants to pick a degree after finishing his 10+2, but he is unsure which course to take depending on his skill set. As a result, he decides to confer with a variety of individuals, including his relatives, instructors, parents, degree students, and working people. He asks them a variety of questions, such as why he should take that course, career prospects with that course, course pricing, and so on. Finally, after talking with several individuals about the course, he chooses to take the course recommended by the majority of people (R, 2023).

* **Neural Networks**

A diagram of a network

Description automatically generated

Figure 10: Visual Representation of Neural Network

Neural networks, commonly referred to as artificial neural networks (ANNs) or simulated neural networks (SNNs), are an area of machine learning that provide the foundation of deep learning techniques. Their name and form are inspired by the how the human brain performs, and they replicate the way real neurons communicate with one another (“What are neural networks?”, n.d).

ANNs are made up of numerous layers of nodes, including an input layer, one or more hidden layers, and an output layer. These nodes, referred to as artificial neurons, are linked together by weights and thresholds. When a node's output reaches a predefined threshold, the node activates and transfers data to the next tier. However, if the output goes below the threshold, no data is sent to the next layer (“What are neural networks?”, n.d)

To improve their accuracy through learning, neural networks rely on training data. Once tuned for accuracy, these algorithms become powerful tools in computer science and artificial intelligence, enabling quick data categorization and grouping. For tasks such as speech recognition and picture identification, neural networks may now finish the process in minutes rather than hours. Google's search algorithm is a well-known example of a neural network (“What are neural networks?”, n.d).

Neural networks can be categorized into various types, each serving distinct purposes. They are:

* Feedforward neural networks/multi-layer perceptrons (MLPs)

They are made up of three layers: an input layer, a couple of hidden layers, and an output layer. Although they are commonly referred to as MLPs (Multi-Layer Perceptrons), they are constructed with sigmoid neurons rather than perceptrons due to the fact that real-world issues are generally nonlinear. These models are data-driven and serve as the foundation for a variety of applications such as computer vision, natural language processing, and other forms of neural networks (“What are neural networks?”, n.d).

* **Convolutional Neural Networks (CNN)**

Similar to feedforward networks, convolutional neural networks (CNNs) are created expressly for applications like pattern recognition, computer vision, and image recognition. To find and analyze patterns inside photos, they mainly rely on concepts from linear algebra, particularly matrix multiplication (“What are neural networks?”, n.d).

* **Recurrent Neural Networks (RNN)**

Recurrent connections, which create feedback loops, are what give recurrent neural networks (RNNs) their name. When working with time-series data, these learning algorithms are extremely helpful since they enable the prediction of future events, such as forecasting stock market movements or predicting sales (“What are neural networks?”, n.d).

A blue circle with lines connecting

Description automatically generated

Figure 11: Visual Representation of RNN

### 2.2.2.2 Relevant research papers on Machine Learning/Deep learning techniques for email/text classification

Jhaveri et al.'s (2022) work "A review on Machine Learning strategies for real-world engineering applications" gives an overview of several machine learning strategies and their applications in real-world engineering scenarios. The authors examine the growing relevance of machine learning in the digital era, when vast volumes of unstructured, semi-structured, and structured data are generated by numerous sources, including digital recordings. They emphasize that by extracting insights from this data, intelligent applications may be constructed, leading to the improvement of systems and decision-making processes.

The paper also investigates the performance and properties of machine learning algorithms, as well as how they are impacted by the data's type. As fundamental components of data-driven systems, many machine learning domains such as reinforcement learning, association rule learning, dimensionality reduction, feature engineering, data clustering, regression, and classification analysis are explored. Furthermore, the authors define Deep Learning as a technique derived from artificial neural networks that plays an important role in intelligent data analysis.

The necessity of picking the optimal machine learning algorithm for certain applications is also emphasized in the study, since the choice might have a considerable influence on the outcomes. The authors argue for a thorough grasp of the applicability and principles of machine learning algorithms in a variety of real-world engineering applications (Jhaveri et al., 2022).

Another paper by (Sharma, Sharma, & Jindal, 2021) goes in depth into machine learning and its application. In the paper, it talks about how machines are taught in the realm of computer vision to process, analyze, and detect visual input using methods such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Nave Bayes. With the COVID-19 epidemic, technologies such as face recognition and iris scans have gained significance since they comply to contactless standards, making them useful for Aadhar cards, banking, and event security. Furthermore, computer vision assists in security applications by allowing algorithms to detect persons' faces for access control and automated attendance systems, replacing traditional methods such as keys and identity cards.

Handwritten recognition software has made document digitization more efficient for companies that deal with large amounts of handwritten papers, such as colleges and testing centers. Speech recognition, also known as speech-to-text, is essential in healthcare, the military, automobile systems, and voice interfaces, enhancing accessibility and user experience. Machine learning is useful in the healthcare industry because it allows for real-time monitoring and detection of many factors, allowing for accurate medical records and statistical analysis.

Furthermore, Machine learning makes predictions based on previous data, which is useful for a variety of applications such as stock trading, scientific research, and marketing campaigns. Furthermore, machine learning is widely used in the banking and finance industries, where it aids in fraud detection by identifying patterns in customer transactions and credit ratings. Machine learning also has been useful in diagnosing patients, forecasting viral propagation, identifying viable treatments, and projecting future pandemics for COVID-19 applications. Researchers have created prediction models to track reported instances in various locations, and deep learning techniques such as Convolutional Neural Networks (CNN) have been used to better manage COVID-19 data (Sharma, Sharma, & Jindal, 2021).

Another paper by (Taye, 2023) talks about machine learning and deep learning and how they are different. According to the paper, machine learning is a kind of artificial intelligence that allows computers to automatically learn and improve from experience. It entails creating algorithms that learn patterns and anticipate outcomes based on data without requiring explicit programming for each activity. Deep learning, on the other hand, is an area of machine learning that depends on artificial neural networks. Deep learning is training numerous layers of deep neural networks to learn from enormous quantities of data and make complicated judgments. Because of its capacity to autonomously acquire hierarchical data representations, it is frequently regarded as the most successful kind of self-training. While machine learning comprises a wide range of learning techniques, deep learning focuses on building deep neural networks for problems with complex patterns. Traditional machine learning may need feature engineering by domain experts to identify and design appropriate features, but deep learning frequently does not necessitate this step since neural networks acquire relevant features automatically during training. Deep learning has demonstrated higher performance in tests like picture and speech recognition, indicating that it is well-suited for complicated applications.

Based on the paper mentioned above, it can be concluded that Deep learning is not absolutely necessary for spam email categorization. Spam email categorization is a popular job in machine learning and may be achieved efficiently using a variety of algorithms and approaches, including classic machine learning methods. Typically, supervised learning methods such as Naive Bayes, Support Vector Machines (SVM), Logistic Regression, Random Forest, or even basic Decision Trees can be used to classify spam emails. These algorithms may be trained on labeled datasets including spam and non-spam email instances, allowing them to learn to differentiate between the two groups. Deep learning, particularly deep neural networks, is a potent method that may also be used to classify spam emails. However, for smaller jobs like spam filtering, it may not always be essential.

In a paper called, "Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification”, by (Pranckevičius & Marcinkevičius, 2017), the researchers carried out an experiment in which they compared the effectiveness of several machine learning algorithms in categorizing text reviews. They utilized an Amazon dataset of product reviews, with each review paired with an overall rating and the review text itself. The objective was to categorize the reviews based on their content and attitude into predetermined groupings.

Five machine learning classifiers were used in the experiment: Naive Bayes, Random Forest, Decision Tree, Support Vector Machines (SVM), and Logistic Regression. The process included data extraction and data preparation which involved preprocessing of review texts via tokenization, stop word removal, lowercase conversion, and stemming. Following that, bags of words were generated by employing several n-gram models (unigrams, bigrams, and trigrams) for expressing the reviews as numerical feature vectors. The classifiers were then trained and assessed for accuracy in categorizing the reviews using 10-fold cross-validation.

According to the results, Logistic Regression had the highest classification accuracy, ranging from 32.43% to 58.50%. The accuracy scores for Naive Bayes, Random Forest, and Support Vector Machines were close, ranging from 33% to 45%, with Naive Bayes doing somewhat better. Decision Tree, on the other hand, has the lowest accuracy rates, ranging from 24.10% to 34.58% (Pranckevičius & Marcinkevičius, 2017).

In another paper by (Renuka, Hamsapriya, Chakkaravarthi, & Surya, 2011), conducted a thorough experiment to examine the performance of several machine learning algorithms in the context of spam email categorization. a thorough experiment to examine the performance of several machine learning algorithms in the context of spam email categorization. Spam email has become a major problem due to the increased usage of the internet and email over the last decade, necessitating the development of effective spam filtering algorithms. The researchers gathered a dataset of 4601 email messages, both spam and legal, and identified around 57 parameters for each email, such as word frequency, character frequency, and structural features.

The researchers used three distinct types of machine learning algorithms on the dataset. The first method utilized was the Multilayer Perceptron (MLP) classifier, which is a multi-layer artificial neural network model. The MLP classifier regularly outperformed other classifiers and took less time to create. The J48 classifier was used as the second method, which constructs decision trees based on information entropy and normalized information gain. Although it was significantly less efficient than the MLP classifier, it demonstrated high prediction accuracy. The Naive Bayes classifier, a probabilistic classifier based on Bayes' theorem and strong independence assumptions, was used in the experiment as the third technique. When compared to the other two methods, the Naive Bayes classifier originally showed poorer prediction accuracy. However, the researchers improved its performance by using the Filtered Bayesian Learning (FBL) approach. The FBL technique efficiently removed dependent characteristics from the dataset, resulting in a more accurate representation of the data and enhanced prediction accuracy for the Naive Bayes classifier.

These algorithms' performance was compared using parameters like precision, recall, and prediction accuracy. The researchers used a 10-fold cross-validation procedure on both the training and test datasets to provide a credible assessment. In terms of prediction accuracy and efficiency, the findings showed that the MLP classifier beat the other two methods. Despite this, the J48 and Naive Bayes classifiers performed well in spam email categorization. The huge increase in the Naive Bayes classifier's performance after using the FBL method, which decreased the number of dependent characteristics and improved prediction accuracy, was especially impressive (Renuka, Hamsapriya, Chakkaravarthi, & Surya, 2011).

In a different paper by (Lakshmi et al., 2010), an experiment was conducted to assess several machine learning methods for spam email categorization. The dataset utilized was compiled over two months from various email IDs and comprised around 20 spam email properties. The primary objective was to develop prediction algorithms capable of distinguishing between spam and non-spam (ham) communications.

Several machine learning methods were used in the experiment, including J48 (Decision Tree), MLP (Multilayer Perceptron), Simple Logistic, NB (Naive Bayes), and LDA (Linear Discriminant Analysis). These algorithms were trained on the dataset to generate models that could distinguish between spam and non-spam emails.

Each classifier's performance was assessed using a variety of parameters, including computation time, the number of successfully categorized instances, the number of mistakenly classified instances, prediction accuracy, and error rate. These assessment measures revealed information about each algorithm's ability to effectively anticipate spam emails.

MLP consistently outperformed the other classifiers in terms of error rate and prediction accuracy, according to the results. While certain classifiers, such as J48 and Simple Logistic, performed well, MLP emerged as the most accurate and efficient classifier across all software tools used in the experiment, which included WEKA and RapidMiner (Lakshmi et al., 2010).

Another paper discusses a study focused on applying machine learning techniques to classify spam and ham emails by (Iqbal & Khan, 2022). The major goal was to choose the optimal characteristics for increased classification accuracy. The researchers ran experiments on the spambase email dataset from the University of California, Irvine (UCI) and used a variety of machine learning classifiers, including K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Artificial Neural Network (ANN), Logistic Regression (LR), and Radial Basis Function (RBF).

There were various steps to the suggested approach. To begin, data was collected using the UCI spambase dataset, which had 4601 instances with the class labels 1 (spam) and 0 (ham). Second, preprocessing was performed to normalize properties that had a large range of values. Third, the researchers used the Point-Biserial correlation approach to identify important characteristics that were related to class labels.

## 2.3 Similar Systems

### 2.3.1 Gmail

The upgraded anti-spam filter in Gmail has gotten more sophisticated, causing issues for organizations who use email marketing efforts. Due to the speedier identification of spam-related phrases in subject lines and content, many emails are now being sent to the spam folder instead of reaching consumers' inboxes. Furthermore, machine learning is used to detect senders with poor reputations, which aids in the filtering process. The anti-spam filter in Gmail is designed to keep unwanted or potentially hazardous emails from reaching users' inboxes. Natural language processing, machine learning, statistical analysis, and artificial intelligence algorithms are among the tactics and technologies used. Aside from checking email subject lines and content for spam-related phrases, the filter evaluates sender reputation, email domains, and IP addresses when deciding whether an email should be banned or sent to the user's mailbox (Mailpro, 2023).

Some of techniques used by Google for spam filtering are:

* **Use of natural language processing:** Gmail's anti-spam filter, NLP, analyzes email text to discover patterns and features that are typically linked with spam. The filter can highlight potentially hazardous emails by recognizing the structure of text and detecting odd or suspicious terms (Mailpro, 2023).
* **Allowing emails from recognizable domains and Ip addresses:** Gmail's anti-spam filter considers the reputation of the email sender's domain and IP address to avoid incorrectly labeling innocent emails as spam. If an email is sent from a well-known and respectable domain or IP address, it is more likely to be trusted and delivered to the recipient's mailbox (Mailpro, 2023).
* Blacklisting spam email adresses to prevent future spam: Gmail's anti-spam filter keeps a list of known spam email addresses up to date. When an email is discovered from one of these blacklisted addresses, it is automatically banned, preventing it from reaching users' inboxes. This blacklisting method is a proactive spam-fighting tool that stops subsequent emails from the same sources from generating problems for users (Mailpro, 2023)

### 2.3.2 Outlook

Like other email service providers, Outlook also uses machine learning and AI for filtering spam emails. Some of them are:

* **Machine Learning and AI Algorithms**: Outlook analyzes incoming emails using machine learning and artificial intelligence (AI) algorithms to identify spam trends. These algorithms learn in real time from a massive collection of known spam and valid emails, allowing the filter to adapt to new spam strategies and increase its accuracy over time. Outlook, like other email providers, use Bayesian filtering to determine the likelihood that an email is spam based on the content and characteristics of past spam and non-spam emails. The filter determines the chance of an email being spam and takes necessary actions depending on that likelihood.
* **Content and header analysis**: Outlook searches the subject line, text, and attachments of incoming emails for spam-related keywords, phrases, and patterns. Emails with dubious content are more likely to be marked as spam. The email header contains information about the sender, receiver, and the path traveled by the email. Outlook analyses header data to detect abnormalities or suspect trends, such as emails sent from fake addresses or services with a bad reputation.
* **URL and Link Analysis:** The spam filter examines URLs and links in emails to identify known dangerous domains that are frequently used in phishing attempts.
* **Sender Reputation**: Outlook may keep a list of known spammers and their sending habits. Emails from senders that have a history of spamming may be tagged as spam more frequently.
* **Collaborative Filtering**: Outlook may employ collaborative filtering in some circumstances, in which information from several users is merged to improve the efficacy of the spam filter (OpenAI, 2021).

Microsoft states that their junk email filter does not prevent junk email messages from being delivered, but it does the next best thing by moving suspected spam to the Junk Email folder. The Junk Email Filter is enabled by default, with the protection level set to No Automatic Filtering in Outlook mail provided Microsoft. The aggressiveness of the filter might be enhanced by altering the degree of protection it provides. Each incoming communication is assessed using a variety of criteria, including the message's timestamp and content (“Overview of the Junk Email Filter”, n.d).

Microsoft provides users with some control when it comes to filtering spam messages. While the Junk Email Filter evaluates incoming messages automatically for the user, Junk Email Filter Lists allow the user to modify the definition of spam. The user may add people, email addresses, and domains to these lists, telling the filter not to investigate communications coming from trustworthy sources. Furthermore, the user may use these lists to restrict messages from unknown or untrusted email addresses and domains (“Overview of the Junk Email Filter”, n.d).

Some of the spam filtering techniques implemented by Outlook are as follows:

* **Safe sender list:** The Safe Senders list assures that the email addresses and domain names on it are never labeled as junk email, regardless of the message's content. Users can add their Contacts and other correspondents to this list. It is crucial to remember, however, that safe domains are not immediately recognized by default in Exchange Online or Exchange Online Protection. Blocked domains, blocked sender addresses, and safe sender addresses are the only ones that are acknowledged. All names and addresses in the global address list (GAL) are automatically considered safe for users having a Microsoft Exchange Server account. It is critical to understand that there is a limit to the number of entries that may be added to the Safe Sender list, which is set at 1024 (“Overview of the Junk Email Filter”, n.d).
* **Safe recipients list:** Users who are members of mailing lists or distribution lists can add the list sender to the Safe Recipients List. By doing so, all communications sent to these specified email addresses or domain names will always be immune from being labeled as garbage, regardless of their content (“Overview of the Junk Email Filter”, n.d).
* **Blocked Senders list:** Users may quickly and easily block communications from certain senders by adding their email addresses or domain names to the Blocked Senders List. When a person or email address is added to this list, Outlook automatically forwards any incoming messages from that source to the Junk Email folder. It is vital to remember that any communications sent from persons or domain names on this list will always be considered as garbage, regardless of their content (“Overview of the Junk Email Filter”, n.d).
* **Blocked Top-Level Domains list:** Users can use the Blocked Top-Level Domains List to avoid unsolicited email communications from specified countries/regions. By include country/region codes such as CA [Canada], US [United States], and MX [Mexico] in the list, emails ending in.ca,.us, and.mx will be prevented from reaching the user's inbox (“Overview of the Junk Email Filter”, n.d).
* **Blocked Encodings list:** To block unwanted email messages that appear in another character set or alphabet, you can add encodings to the Blocked Encodings List (“Overview of the Junk Email Filter”, n.d).

### 2.3.3 Yahoo Mail

To safeguard users from unwanted and potentially hazardous emails, Yahoo Mail employs a variety of spam filtering algorithms in the backend. Some of the various approaches utilized are:

* **Machine Learning**: Yahoo Mail uses machine learning techniques to improve its spam filtering capabilities over time. Machine learning algorithms can react to new spam patterns and trends, increasing the filter's effectiveness over time. Yahoo uses Bayesian filtering method. This method analyzes the words and patterns in emails using probability-based algorithms. To understand which traits are commonly linked with each category, the filter is trained using a huge sample of known spam and non-spam emails. As new emails come, the filter determines their likelihood of being spam based on their resemblance to previously detected spam patterns.
* **Analysis of the Email Header**: The email header provides information about the sender, receiver, and the path the email took. Yahoo Mail analyses this header data to detect unusual trends, such as emails sent from known spam-sending IP addresses or websites with a bad reputation.
* **Content Analysis**: The subject line, text, and attachments of the email are analyzed for spam-related keywords, phrases, or HTML code typically used in spam communications.
* **Sender Reputation**: Yahoo Mail keeps a list of known spammers and their sending habits. Emails from senders who have a history of sending spam may be blocked out (OpenAI, 2021).

## 2.4 Technical Research

The technological features of the spam email categorization effort are examined in Chapter 3. The chapter goes over the many options for programming languages, IDEs, libraries/tools, database management systems, operating systems, web servers, and web browsers. These choices were carefully scrutinized to ensure the spam email categorization system's efficiency and usefulness.

### 2.4.1 Programming language Chosen

To choose the best programming language for the project, a thorough examination of the advantages and disadvantages of two significant competitors, Python and R, has been conducted. This examination dives into numerous aspects of these languages, from their essential properties to their use in machine learning and natural language processing. This research seeks to determine the preferable choice between Python and R for the upcoming project by focusing on criteria such as adaptability, simplicity of use, performance, and security.

* **Python:** According to python.org, Python is an object-oriented, high-level language for programming with dynamic semantics that is interpreted. Its high-level built-in data structures, together with dynamic typing and dynamic binding, make it particularly appealing for usage as a scripting or glue language to connect existing components together. Python's concise, easy-to-learn syntax prioritizes readability, lowering software maintenance costs. Python has support for modules and packages, which promotes program modularity and code reuse. The Python interpreter and substantial standard library are free to use and distribute in source or binary form for all major platforms (“What is python? executive summary”, n.d).

An article explains why Python is used widely by developers for Machine learning and Artificial Intelligence. Some of the common reasons are as follows:

1. **It offers a large ecosystem of libraries**: Python has a large selection of modules and frameworks that help to streamline the development process and increase productivity. Among the well-known libraries are NumPy for scientific computations, SciPy for advanced calculations, and scikit for data mining and analysis. These libraries operate in tandem with powerful frameworks like as TensorFlow, CNTK, and Apache Spark to offer important tools for machine and deep learning applications. This wealth of resources considerably speeds up development and improves the capabilities of these initiatives (Ogoti, 2021).
2. **The simplicity it provides**: Python code retains its conciseness and clarity, giving newbies to machine and deep learning initiatives a considerable edge. The language's simple syntax speeds up application development, exceeding many other programming languages. Furthermore, it permits algorithm testing without requiring full implementation. Code readability is critical for collaborative coding, allowing several contributors to work effortlessly on complex projects. Python's popularity as a platform means that competent engineers are readily available for team integration. As a result, new developers may quickly learn Python's fundamentals and contribute to project progress (Ogoti, 2021).
3. **Visualization tools:** Python has a large number of libraries, some of which have powerful visualization features. The capacity to show data in an intelligible manner is important in the fields of AI, machine learning, and deep learning. As a result, Python appears as an excellent choice for implementing this capability. Data scientists may use libraries like Matplotlib to create charts, histograms, and plots, which improves data representation and visualization. Furthermore, Python's support for several APIs increases the efficacy of the visualization technique (Ogoti, 2021).
4. **Flexibility in integration:** Python projects have the flexibility to seamlessly integrate with systems developed using diverse programming languages. This facilitates effortless amalgamation with other AI projects written in alternative languages. Moreover, Python's extensibility and portability enable the execution of tasks spanning multiple languages. This adaptability streamlines the process of training machine learning models, presenting data scientists and developers with a convenient framework (Ogoti, 2021).

Python has some drawbacks too. An article touches upon some of them. They are as follows:

1. **Speed**: Python's performance is limited by its interpreted nature, which requires code to be performed sequentially. As a result, Python's execution is typically slower than those of other programming languages. Unless the project's emphasis is entirely on quick execution, this speed constraint is inconsequential. If ideal speed is not a requirement, the benefits of Python are strong enough to offset the performance limits (“Pros and cons of python in machine learning”, 2021).
2. **Multi-threading:** Threading is difficult in Python because of the Global Interpreter Lock (GIL), a mutex that limits concurrent execution to a single thread at a time. As a result, the performance of multi-threaded CPU-bound operations may suffer, potentially resulting in slower execution than single-threaded counterparts. This risk can be alleviated by using multiprocessing techniques rather than multi-threaded apps (“Pros and cons of python in machine learning”, 2021).
3. **Not good for mobile applications:** Python's interaction with mobile environments is less straightforward, making it less suitable for mobile computing. Python is a lesser-known language in the mobile setting, with no official support on platforms such as Android and iOS. Python, on the other hand, can be used to create mobile applications with a little more effort. Various Python libraries make it possible to build for both the Android and iOS platforms (“Pros and cons of python in machine learning”, 2021).
4. **Limitations in design**: Python's design incorporates several constraints. The dynamic typing feature of the language reduces the need to define variable types during coding. It follows the notion of duck-typing, in which an object's type is only significant at runtime, eliminating the need for explicit type declaration. According to this concept, if an object resembles a duck, it is considered as one. This strategy can lead to runtime mistakes, which programmers can easily identify during the development phase (“Pros and cons of python in machine learning”, 2021).

* **R programming language:** According to r-projects.org, R is both a language and an environment for statistical computation and graphics. It is derived from the GNU project and is similar to the S language and environment, which was initially designed at Bell Laboratories by John Chambers and colleagues. R can be seen as a different interpretation of S, having substantial differences while yet accommodating a major percentage of S code without modification. R's features include a wide range of statistical techniques and graphical tools (including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and more), as well as strong extensibility. The S programming language is frequently used to advance statistical methods, and R provides an Open-Source route for active participation in this research arena (“What is R?”, n.d).

Some of the benefits of using R programming language are:

1. **Open source**: An open-source language allows users to interact with it without the need for licenses or payments. R, being an open-source language, encourages contributions to its development. Individuals may improve R by improving current packages, developing new ones, and resolving issues in the language's ecosystem (“R advantages and disadvantages – javatpoint", n.d).
2. **Platform independence**: As a cross-platform programming language, R demonstrates platform independence, allowing its code to run flawlessly across several operating systems. This feature enables developers to create applications for numerous competing platforms with a single program development. R runs flawlessly on Windows, Linux, and Mac OS systems (“R advantages and disadvantages – javatpoint", n.d).
3. **Machine learning operations**: R serves as a versatile tool in the field of machine learning operations, accommodating diverse tasks such as classification and regression through its array of packages and neural network development capabilities, all of which have earned it esteem among the global community of accomplished data scientists. The language shines in data manipulation, with exceptional assistance provided by packages such as dplyr and readr, which adeptly turn unstructured data into a cohesive format. Furthermore, R differentiates itself by its ability to generate high-quality plots and graphs, which is accomplished with libraries such as ggplot2 and plotly. These materials promote visually appealing and artistically beautiful graphical representations, solidifying R's unique place among programming languages (“R advantages and disadvantages – javatpoint", n.d).

Similar to python, R also has some drawbacks to it. Some of them are as follows:

1. **Data Handling**: R distinguishes itself from other programming languages in terms of data management by keeping objects within the computer's physical memory. When compared to Python, one obvious distinction is R's somewhat greater memory utilization. R requires the aggregation of whole datasets in memory, which may not be optimum for dealing with large datasets, particularly in Big Data scenarios (“R advantages and disadvantages – javatpoint", n.d).
2. **Security**: R lacks fundamental security capabilities, distinguishing it from most traditional programming languages such as Python, where security is a key part. This shortcoming places numerous constraints on R, most notably its viability for incorporation into web applications, a space where effective security measures are critical (“R advantages and disadvantages – javatpoint", n.d).
3. **Difficult learning curve:** R demonstrates to be a sophisticated programming language with a severe learning curve, which may provide issues for people with no prior programming knowledge or expertise, inhibiting their ability to understand R successfully (“R advantages and disadvantages – javatpoint", n.d).

Following a thorough evaluation of the features and capabilities of both Python and R programming languages, Python is the ideal choice for the project. Python is a powerful and practical alternative because to its varied environment, vast libraries, simplicity of learning, and cross-platform interoperability. Python's ability to connect with a wide range of systems, together with its extensive support for machine learning and data visualization, makes it a great fit for the project's objectives. While R has its advantages, its limitations in areas like as security and data handling make Python the more well-rounded and beneficial choice. Thus, based on the extensive investigation, the decision to use Python as the project's specified programming language is well-founded and promising.

### 2.4.2 IDE (Interactive Development Environment) Chosen

The IDE chosen for the completion of this project is PyCharm. PyCharm is a flexible platform created by JetBrains that serves as a full IDE for Python development. It is widely used in Python application development, with significant industry companies such as Twitter, Facebook, Amazon, and Pinterest using it as their primary Python IDE. The program supports both Python v2.x and Python v3.x versions and is compatible with Windows, Linux, and Mac OS environments. PyCharm provides a collection of modules and packages within its framework, facilitating the Python software development process while minimizing time and effort. Furthermore, it supports customization, allowing developers to adjust the environment to their own requirements (“What is PyCharm?: What is Pycharm used for?”, 2023).

PyCharm provides a plethora of essential features for an enhanced Python programming experience. Its clever code editor makes high-quality code authoring easier by using color schemes, mistake detection, and autocompletion. Navigating code components, classes, and files with ease speeds up editing and debugging, while refactoring tools enable efficient structural upgrades. PyCharm extends its capabilities to numerous web technologies, including web app building, live editing, and HTML, CSS, and JavaScript previewing. It is a faithful ally for well-known Python web frameworks such as Django, providing developers with autocompletion, debugging tools, and faster code. Furthermore, PyCharm's support for Python's scientific libraries, such as Matplotlib and NumPy, is crucial for data science and machine learning activities, aided by interactive graphs and integrations with tools such as IPython and Pytest, which stimulate inventive solutions (“What is PyCharm?: What is Pycharm used for?”, 2023).

### 2.4.3 Libraries chosen / Tools chosen

The success of the NLP spam email categorization project is dependent on the strategic integration of a number of critical libraries and tools. The project will use a complete toolset and numerous key tools/libraries. Furthermore, new libraries may be added during the implementation phase to enhance the project's analytical capabilities. As the project progresses, the combined power of these libraries and tools will pave the way for efficient data preparation, model training, assessment, and visualization, ultimately leading to a robust and accurate spam email classifier. Some of the libraries/tools include:

* **Pandas**: Pandas is a Python package which is a powerful data manipulation and analysis tool. It allows for the study, cleaning, investigation, and modification of datasets, and its name alludes to "Panel Data" and "Python Data Analysis." Pandas, created by Wes McKinney in 2008, allows for the examination of large amounts of data, allowing for educated conclusions based on statistical principles. It specializes in converting jumbled datasets into cohesive, understandable representations, which is critical in data science projects. Pandas allows users to extract useful information and improve their knowledge of the underlying patterns in data by providing insights into correlations, averages, and extremes. This greatly contributes to the discipline of data science (“Pandas introduction”, n.d).
* **NLTK**: NLTK is a well-known framework for developing Python applications for human language data processing. It provides user-friendly interfaces to more than 50 corpora and lexical resources, including WordNet. This comprehensive toolset includes text processing packages for applications like as categorization, tokenization, stemming, tagging, parsing, and semantic reasoning. Furthermore, NLTK enables easy interaction with powerful NLP packages, which are supported by an active discussion community. Tokenization, stemming, and stopword removal will be performed by NLTK, which are crucial components in NLP undertakings (“Documentation”, n.d).
* **scikit-learn**: Scikit-Learn (Sklearn) emerges as a very valuable and robust Python machine learning package. Its toolbox includes a wide range of efficient resources for tasks including classification, regression, clustering, and dimensionality reduction, all accessible via a unified Python interface. The library is mostly written in Python and is built other libraries such as NumPy, SciPy, and Matplotlib (“Scikit learn – introduction", n.d).
* **Google Collab**: Google Colab (short for Google Colaboratory) is a cloud-based Jupyter notebook platform that is commonly used to teach machine learning and other disciplines. It allows users to write text explanations and Python code using a web interface, making it accessible and portable ("Google Research," n.d.).

### 2.4.4 Database Management System

Neither a Database or a Database Management System (DBMS) was required for this project.

### 2.4.5 Operating system

Windows 11 was chosen as the project's operating system. This choice is based on its compliance with the project's software needs and the student's experience with the Windows environment. Windows 11 has a user-friendly interface and offers a large selection of tools and programs that are critical to the project's success.

### 2.4.6 Chapter Summary

The following chapter was conducted to allow the developer to undergo substantial background research for his chosen topic. The chapter first focused on domain research which investigated machine learning and natural language processing and how they can be utilized to create a working model for filtering spam emails. The chapter digs deeper into various machine learning models that can be utilized for spam filtering. The chapter also emphasized the pipeline to be utilized for the completion of the project. Furthermore, the model also discussed on research papers conducted by individuals where machine learning and NLP was utilized for performing text or spam email classification. The chapter also delves into similar systems related to the student’s topic. This includes widely used email services like Gmail, Outlook, and Yahoo mail and how they work in the backend. Lastly, the chapters investigate the best programming language, IDE, tools and libraries to utilize for the completion of this project.

# Chapter 3: Methodology

## 3.1 Introduction

The methods used to address the problem of spam email classification using natural language processing method is described in this chapter. The technique includes several steps, such as data collection, pre-processing, transformation, modeling, and the selection of relevant assessment measures. These steps are critical in the development of an effective and resilient spam email categorization system.

## 3.2 Methodology

CRISP DM methodology was selected for the completion of this project. CRISM DM stands for CRoss Industry Standard Process for Data Mining. It serves as a basic process model for data science projects. It was introduced in 1999 with the goal of establishing standard data mining techniques across multiple industries, and it has since established as the dominant strategy for managing data mining, analytics, and data science projects. Data science teams are more likely to obtain optimal results when they use a flexible interpretation of the CRISP-DM framework while including team-oriented agile project management approaches (Hotz, 2023).

Diagram of a diagram of data

Description automatically generated

Figure 12: Visual Representation of CRISP-DM

CRISP DM methodology has 6 phases, and they are follows:

* Business understanding: The Business Understanding phase focuses on comprehending the project's objectives and requirements. The tasks for this phase are as follows:

1. **Clarifying business objectives**: As stressed in the CRISP-DM Guide, it is critical to get a thorough understanding of the client's business goals. As a result, defining criteria for assessing company performance is critical.
2. **Evaluating the context**: This task includes determining resource availability, outlining project prerequisites, assessing risks and contingencies, and doing a full cost-benefit analysis.
3. **Defining data mining objectives:** In addition to creating commercial goals, the technological characteristics of success in the data mining domain should be outlined.
4. **Creating a project plan:** Making a project plan involves selecting appropriate technologies and tools and developing precise plans for each project phase (Hotz, 2023).

This is the phase where the student had to identify a problem and come up with a solution. In this case, spam emails being the problem and a classification model to detect spam emails being the solution. In this phase, the aim and objectives of the project were mentioned. This is also the phase where a project plan was implemented by the student’s university with phase 1 being intensive research on the project and preliminary data analysis before model building.

* **Data Understanding:** The following stage is the Data Understanding stage. Building on the foundation built by Business Understanding, this phase moves the emphasis to identifying, acquiring, and examining datasets that contribute to the attainment of project objectives. This phase consists of four tasks, which are as follows:

1. **Initial data collection:** Important data is gathered and, if necessary, imported into the specified analytic tool.
2. **Data description:** The data is examined, and its surface properties, such as data format, record count, and field characteristics, are painstakingly documented.
3. **Data exploration:** Data exploration begins with a deeper dig into the data. Intricate linkages within the data are revealed and studied through querying and visualization.
4. **Data quality verification:** This task involves determining the cleanliness or impurity of the data. Any data quality concerns that are discovered are meticulously recorded (Hotz, 2023).

The Data Understanding phase concentrated on locating, obtaining, and analyzing datasets relevant to the project's goals for the student. The dataset's surface attributes were recorded. Data exploration including data visualization dug deeper into the data to find subtle relationships, while data quality verification guaranteed the data's cleanliness and dependability.

* The Data Preparation phase, sometimes known as "data munging," is critical, accounting for around 80% of the project's scope—a generally accepted rule of thumb. This step prepares the final dataset(s) for modeling and consists of five separate tasks, and they are as follows:

1. **Data selection:** Carefully pick relevant datasets, properly documenting the reasoning behind each inclusion or removal.
2. **Data cleansing:** While this is frequently the most time-consuming job, it is critical to maintaining a high-quality dataset. It protects against the "garbage-in, garbage-out" situation. Rectifying, imputing, or deleting erroneous values is a common approach in this work.
3. **Data construction:** The creation of new qualities that aid in the analytical process. One famous example is the calculation of an individual's body mass index using height and weight information.
4. **Data Integration**: Data integration is the process of combining data from several sources to create new datasets that give complete insights.
5. **Data formatting**: Data is customized to satisfy unique needs. This work is shown by the translation of string-based numeric values into their numerical equivalents, allowing mathematical operations (Hotz, 2023).

During this phase, extensive data preparation was conducted on the email text data to prepare it for further analysis and model creation, including tokenization, stop word removal, lemmatization, and duplication management.

* **Modelling:** What is usually regarded as the most captivating aspect of data science work frequently coincides with the project's shortest period. Various models are routinely formulated and tested at this step using various modeling methodologies. This phase has four separate tasks, which are as follows:

1. **Algorithm selection:** The algorithms to be tested (such as regression or neural networks) are chosen.
2. **Test design:** Depending on the modeling technique used, the dataset may be divided into training, test, and validation sets.
3. **Model development:** This task involves tweaking the model by deriving, testing, and building upon multiple models until a model that meets the necessary requirements is developed.
4. **Model evaluation:** As numerous models compete for dominance, the data scientist must assess model findings in the context of domain expertise, set success criteria, and the established test design (Hotz, 2023).

Various modeling approaches will be used to create and test numerous models throughout the Modelling phase to find the best model for classifying spam emails. To create a model that fits the project's needs, algorithms will be chosen, tests will be designed, models will be developed.

* **Evaluation:** In contrast to the technical model assessment performed during the Modeling phase's Assess Model task, the Evaluation phase takes a broader approach by analyzing the alignment between the chosen model and the overarching business objectives. This stage consists of three separate tasks, and they are follows:

1. **Analyze the outcomes:** Do the models fit the requirements for corporate success? Which one(s) should be accepted for the business?
2. **Process evaluation:** A thorough assessment of the completed job is performed. Were any details overlooked? Were all steps taken with care? The findings are presented, and any remedial steps are implemented.
3. **Determination of next step:** Next actions are determined based on the outcomes of the previous activities. The path of action for the next phase is mapped. The options include continuing with deployment, incremental refining, or the start of new initiatives (Hotz, 2023).

The Evaluation phase of the spam email classification project will take a broader approach by assessing the alignment between the chosen model and the overall goal of effectively identifying spam emails. The model assessment results will be rigorously examined to evaluate whether the models match the success criteria set for successful spam email categorization. The process will be thoroughly examined to find any areas for improvement in the model. The student will decide the next steps for the spam email categorization project based on the evaluation findings. This will involve model deployment decisions if the chosen model performs well and achieves the desired objectives.

* **Deployment**: A model is of no use if it cannot be utilized by users. This phase involves deploying the model into a working system to allow users to interact with it. The phase involves 4 tasks, and they are follows:

1. **Plan the model's deployment**: Create and describe a plan for deploying the model.
2. **Create a monitoring and maintenance plan**: Create a comprehensive plan for constant monitoring and maintenance to avoid operational issues throughout the active or post-project phases of the model.
3. **Produce a decisive report:** The project team prepares a complete synthesis of the attempt, which may include a final presentation of data mining results.
4. **Reflective Review:** Conduct a reflective review of the project, looking into its accomplishments, areas for improvement, and plans for future improvement (Hotz, 2023).

The best functional spam email categorization model will be used to create a web app in python. The app will be able to take in input from the user, utilize the chosen model to predict the legitimacy of the input. If the app works without any complications, it will be deployed in an online web-app hosting service like Heroku or Streamlit Sharing.

# CHAPTER 4: DESIGN AND IMPLEMENTATION

## 4.1 Introduction

Chapter 4 focuses on converting conceptual concepts into real models for a successful spam email categorization system. The chapter begins with a comprehensive examination of data through the phases of collection and comprehension, offering light on the dataset's origins and complexities. Following then, the focus shifts to data pre-processing, which involves careful processes to improve and harmonize the dataset for later model creation. The next section in which several model-building approaches are used to create a complex spam email categorization model. Each item in this section gives insights into approaches and outcomes, with the goal of identifying the most successful strategy that is consistent with the project's objectives.

Overall, this chapter acts as a link between theoretical conception and actual implementation, helping the project get closer to its aim of improving email security, efficiency, and user experience.

## 4.2 Data Collection and Data Understanding

### 4.2.1 Data Collection

The following dataset, retrieved from Mendeley Data, consists of tagged text messages gathered for SMS Phishing research. It contains 5971 text messages classified as legitimate (ham), spam, or smishing. It consists of 489 spam messages, 638 smishing messages, and 4844 ham messages. This dataset comprises raw message content, which may be used as labeled data in Deep Learning or to extract additional properties. The collection comprises extracted characteristics from harmful communications, which may be used to classify messages as malicious or normal. This dataset also contains Python code that is used to extract characteristics. The data was acquired by translating photos from the Internet to text using Python programming. Attributes were chosen based on their significance (Mishra & Soni, 2022).

Dataset URL: <https://data.mendeley.com/datasets/f45bkkt8pr/1>

### 4.2.2 Data Understanding

Google Collab was utilized to start a new notebook and import the dataset.

* **Data Import**

A screen shot of a computer

Description automatically generated

Figure 13: Data import in Google Collab

Number and Pandas libraries were loaded in the notebook. The pandas library was used to load the dataset, in a variable called data.

* **Data View and Initial Information**

A screenshot of a chat

Description automatically generated

Figure 14: Data view and information

The variable data was utilized to view 5 random rows of the dataset in a data frame in Collab using the sample function. The shape function was called to see the number of rows and columns the dataset contains.

Based on the results, it could be seen that the dataset consists of 5971 rows and 5 columns, with each row representing a message. The columns contain "LABEL" to categorize messages as "ham," "smishing," or "spam." "TEXT" for message content, including markers for URLs, emails, and phone numbers. The "ham" communications are most likely valid, however "smishing" or "spam" ones might be harmful. This dataset provided an extensive resource for text analysis and classification application trying to be built. Valuable insights were gained by diving into message content, identifying trends, and using labeled data for classification modeling.

### 4.2.3 Data Cleaning

Before getting into identifying trends and patterns in the dataset, it was crucial to clean the dataset to get it ready for Exploratory Data Analysis (EDA). This included tasks like checking for missing values or duplicate values all the way to text preprocessing. The steps taken are as follows:

* **Check for missing values**

**A screenshot of a computer

Description automatically generated**

Figure 15: Check for missing values

When the dataset was checked for missing values, it was discovered that none of the columns had any. This showed that the dataset was comprehensive in terms of data availability, with no null values needing imputation.

* **Check for Duplicate values**

**A screenshot of a computer program

Description automatically generated**

Figure 16: Check and Drop Duplicate values

After checking for duplicate rows in the dataset, it was discovered that there are 17 duplicate rows. To resolve this issue, duplicates were eliminated from the DataFrame.After deleting duplicate entries, the dataset now had 5954 rows and 5 columns. This procedure helped to clean the data and ensure that each message entry in the dataset is unique.

* **Change** **variables into 0s and 1s for Label, URL, EMAIL and PHONE columns**

It was decided to change the variables into 0s and 1s for Label, URL, EMAIL and PHONE columns in order to standardizes the representation of labels and binary variables in the dataset, which improved consistency and simplified future analysis and modeling operations.

A screenshot of a computer

Description automatically generated

Figure 17: Change variables into 0s and 1s for Label, URL, EMAIL and PHONE columns

In the Label Column, the variables Spam and Smishing were set to be identified as 1 and ham variable as 0. Similarly, 0 for No and 1 for Yes variables in the URL, EMAIL and PHONE columns were also changed.

According to the result it could be noticed the data frame contained numerical values (0 ,1 ) for all categorical value variables previously seen on the data frame.

* **Drop Messages which are not English**

It was noticed that some of the messages in the dataset were not in English. Some were written in English but using English to spell another language. Thus, it was decided to drop any variables in the TEXT column which are not in English. This improved the dataset's applicability for English-centric text analysis and modeling tasks, laying the foundation for a more precise and targeted approach.

A screenshot of a computer program

Description automatically generated

Figure 18: Detect all the languages of each variable in the TEXT column

A screen shot of a computer program

Description automatically generated

Figure 19: Remove all but English rows in the TEXT column

The dataset, which originally had 5971 rows and 5 columns, was enhanced with a 'language' column that indicated the identified language for each text item. The distribution of languages indicated English ('en') as the main language, with other languages appearing in lower numbers. Non-English rows were detected and deleted using systematic filtering, leaving a revised dataset of 5466 rows and 6 columns.

* **Text Preprocessing**

**A screenshot of a computer program

Description automatically generated**

Figure 20: Text preprocessing

The developer created a function called 'preprocess\_text' to improve the quality of the dataset's 'email' column. This single function was also created considering that, when model building is completed, this function can be used for model deployment. This is because the input data from users will also need to go through the same preprocessing for the created model to interpret and classify the input. The function included many crucial steps:

* **Lowercasing**: Changes all text to lowercase to ensure consistency.
* **Tokenization**: Breaks down the text into individual words for further examination.
* **Special Character Removal**: Removes non-alphanumeric characters and leaves just words.
* **Stopwords/Punctuation Removal**: Removes frequent English stopwords and punctuation marks to concentrate on significant words.
* **Stemming**: Use the Porter Stemmer to reduce words to their root form.

The function is then applied to the 'TEXT' column via the 'apply' method, and the modified text is saved in a new column called 'preprocessed\_text' within the data frame. This meant that this newly created column will be the column used for model building. Also, it also ensured that if for some reason, the preprocessing function resulted in faulty data, the email column can be used again to create another preprocessed version of the column.

### 4.2.4 Exploratory Data Analysis

* **Distribution of Spam and Ham emails**

The distribution of spam and ham emails after filtering the sample yields interesting information. Following the data processing step, it was found that there are 1111 spam messages and 4355 non-spam messages in the preprocessed dataset. This distribution was visualized as a pie chart, with spam messages in red and non-spam messages in green. The graphic gives a clear picture of the percentage of spam and non-spam emails in the dataset, 20% of the dataset being spam while 80% of the dataset being ham. This shows there is a big class imbalance in the dataset.

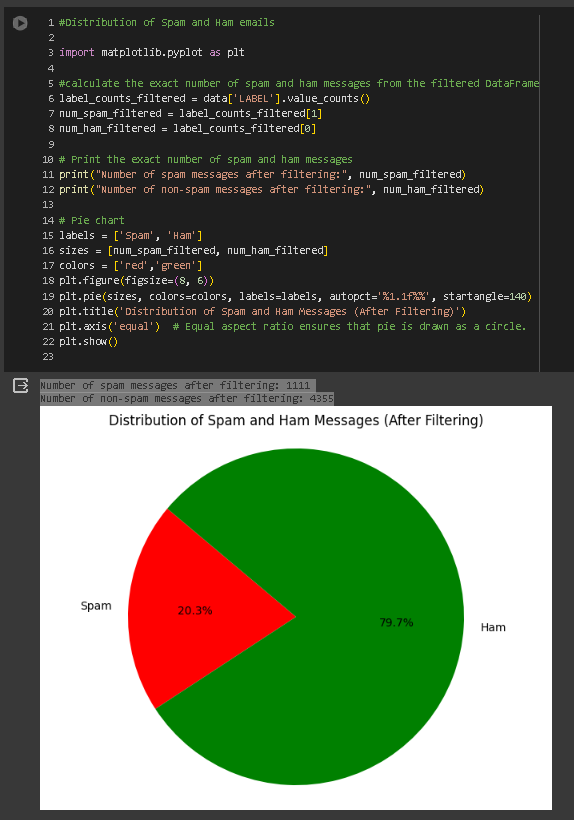


Figure 21: Distribution of Spam and Ham Emails

* **Top 20 most commonly used words for Spam and Ham text**

**A computer screen shot of a program

Description automatically generated**

Figure 22: Code to see visualize top 20 most used spam and ham messages

It was decided to see the top 20 most frequent words used in spam and ham messages. It was done by separating spam and ham messages from the dataset, tokenizing the words inside the preprocessed text column, and then determining the frequency of each word occurrence. Using the Counter module, the script effectively found the most common terms in spam and ham messages.

A screenshot of a graph

Description automatically generated

Figure 23: visualization of top 20 most used spam and ham messages

Based on the screenshot above, it was noticed that ham emails had distribution of words being used everyday by humans on a regular basis. This included words, ranking by frequency, u (you), im, go, get, come and more. On the other hand, spam messages displayed a pattern of having more phishing or baiting words like call. free, claim, prize and more.

* **Word Cloud of Spam and Ham messages**

**A computer screen shot of text

Description automatically generated**

Figure 24: Word cloud of Spam and Ham text

It was decided to generate word clouds for spam and ham messages. These word clouds visually represented the most common words in both spam and ham messages after preprocessing.

A screenshot of a phone

Description automatically generated

Figure 25: Word Cloud Result

Similar to the top 20 most frequent spam and ham words, the screenshot above also showed similar results. Ham emails had a distribution of terms that humans use on a daily basis. This comprised terms such as u (you), im, go, get, and come. Spam messages, on the other hand, displayed more phishing or baiting phrases such as call, free, claim, reward, and so on.

* **Visualization of Average Word length between Spam and Ham Messages**

**A screenshot of a computer

Description automatically generated**

Figure 26: Average Word length between Spam and Ham Messages

It was determined to identify the average word length between spam and ham communications by tokenizing the preprocessed text, computing the length of each word, and then calculating the average word length for each message type and plotting it on a graph. The research revealed disparities in word lengths between spam and ham communications, providing insights into the linguistic features of each group. Based on the result, it was noticed that Spam messages had a slightly higher average words length of 5.3 as expected. Ham messages contrarily had an average word length of 4.3. This displayed that spam messages are likely to contain more words in the message compared to real messages.

* **Sentiment analysis of spam and ham text using sentiment score**

It was decided to use sentiment ratings to analyze spam and ham text messages. The approach comprised of assigning sentiment scores to each preprocessed text message using NLTK's VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analyzer. The sentiment scores were computed using the compound score, which indicated the overall sentiment intensity of the text. The results were then plotted using a histogram.

**A computer screen shot of a program

Description automatically generated**

Figure 27: Code for Sentiment analysis of spam and ham text using sentiment score

A screen shot of a graph

Description automatically generated

Figure 28: Sentiment Analysis result

Based on the result, it was noticed that Ham messages had a wide distribution of words, distributed across both positive and negative sentiment scores, although the majority was at a neutral sentiment score. This was expected as ham messages can vary emotionally from sad to angry to neutral to happy. Similarly, Ham messages displayed a more positive sentiment as majority of them being either neutral or on the happier side though there were some outliers on the negative end. This also displayed that spam messages are likely to have more positive sentiment as these messages were constructed to persuade and bait people.

* **Chi-square statistics and heat maps of EMAIL, URL and PHONE columns against Label Column (Spam/Ham)**

It was decided to look at the relationship between the existence of certain elements, notably the EMAIL, URL, and PHONE columns, and the classification of spam/ham messages. To do this, chi-square statistics and contingency tables were used for each characteristic (EMAIL, URL, and PHONE) against the label column (spam/ham). Heatmaps were used to depict the contingency tables, which helped to visualize the relationships.

A screenshot of a computer program

Description automatically generated

Figure 29: code for Chi-square statistics and heat maps of EMAIL, URL and PHONE columns against Label Column (Spam/Ham)

A screenshot of a computer

Description automatically generated

Figure 30: Chi-square statistics and heat maps of EMAIL, URL and PHONE columns against Label Column (Spam/Ham)

The chi-square statistics result and heatmaps in the screenshot above showed that the chi-square statistics for URL vs. LABEL were 754.4590244258243, with a p-value of 4.303808368396881e-166, indicated a substantial relationship between URLs and spam messages. Similarly, the chi-square values for Phone vs. LABEL were 3760.345240056755 (p-value = 0.0), indicated a high correlation between phone numbers and spam messages. In contrast, the chi-square statistics for Email vs. LABEL were 60.660379473657315, with a p-value of 6.7821525047237105e-15, indicated a limited correlation between emails and spam messages.

* **Distribution of Emails, URLs and Phone Numbers in Spam vs Ham**

**A screenshot of a computer screen

Description automatically generated**

Figure 31: Distribution of Emails, URLs and Phone Numbers in Spam vs Ham

The distribution of characteristics such as emails, URLs, and phone numbers in spam and ham communications was determined to be investigated. By mapping these traits to binary indicators (No/Yes), the prevalence of these items in spam and ham messages may be seen. Using count graphs, it became clear how these characteristics differed between spam and ham messages. For example, based on the results, it was discovered that messages with URLs, phone numbers, or emails were the majority of spam messages. Although many messages did not contain emails, phone numbers, or URs, those that did were more likely to be spam messages. This showed that spam messages typically contain URLs, emails, or phone numbers.

## 4.3 Model Building

During the research conducted by the student, it was found that 4 models work really well in performing classification tasks. They are as follows:

* + - 1. Support Vector Machine
      2. Naive Bayes
      3. Decision Tree Classifier
      4. Random Forest Classifier

Throughout the model building phase, these 4 models have been tweaked and tuned to select the best model. Model Building was concluded in 4 steps:

Pre-Model Building Steps

Using Count Vectorization

Using TFIDF vectorization

Employing Grid Search to find the best hyperparameters for each model and tuning them with the hyperparameters.

### 4.3.1 Pre-Model Building steps

As mentioned above, several critical pre-modeling activities were conducted before model development to assure the classification task's efficacy and accuracy.

#### 4.3.1.1 Split the Data

A screen shot of a computer program

Description automatically generated

Figure 32: Split the data into train and test

The first step was to divide the dataset into training and testing sets using the train\_test\_split function, using 80% for training and 20% for testing. This division is critical for assessing the model's performance on unseen data.

#### 4.3.1.2 Vectorization

A screenshot of a computer code

Description automatically generated

Figure 33: vectorization

The text data was then vectorized, converting it into numerical attributes that machine learning algorithms can handle. This transformation helped the models to efficiently comprehend and evaluate text.

#### 4.3.1.3 SMOTE

A screen shot of a computer

Description automatically generated

Figure 34: Applied SMOTE

To address the imbalance in the dataset, which has 20% spam messages and 80% ham communications, the Synthetic Minority Over-sampling Technique (SMOTE) was used to balance the data distribution. SMOTE provides synthetic minority class samples to address class imbalances and improve model performance (Wang et al., 2021).

### 4.3.2 Using Count Vectorization

A screenshot of a computer program

Description automatically generated

Figure 35: Using Count Vectorization

#### 4.3.2.1 Support Vector Machine

A screenshot of a computer

Description automatically generated

Figure 36: SVM Count Vectorization

The Support Vector Machine (SVM) model achieved an accuracy of 77.06% during the investigation. Its precision for spam detection was discovered to be 47%, with a recall of 99% and an F1-score of 64. Despite attaining a high recall, indicating its efficacy in detecting spam, the accuracy score suggests a poorer capacity to effectively categorize spam messages, meaning a larger false positive rate. The confusion matrix further demonstrated SVM's performance, with 620 true negatives, 249 false positives, 2 false negatives, and 223 true positives.

#### 4.3.2.2 Naive Bayes

A screenshot of a computer program

Description automatically generated

Figure 37: Naive Bayes using Count Vectorization

Naive Bayes performed well throughout the analysis, obtaining an accuracy of 96.53%. The precision for spam detection was impressive at 91%, with a recall of 92% and an F1-score of 92%. These metrics demonstrated the model's ability to properly categorize both spam and non-spam communications. The confusion matrix confirms this discovery, with 849 true negatives, 20 false positives, 18 false negatives, and 207 true positives. This balanced performance demonstrates Naive Bayes' strong ability to distinguish between spam and non-spam messages.

#### 4.3.2.3 Decision Tree Calssifeier

A screenshot of a computer program

Description automatically generated

Figure 38: Decision Tree using Count Vectorization

Decision Trees produced an accuracy of 36.84%. Although it had a great recall of 100% for spam messages, its accuracy was somewhat low, at 25%. As a result, the model had a significant false positive rate, showing a tendency to misclassify non-spam messages as spam. This discovery is supported by the confusion matrix, which indicated 179 true negatives, 690 false positives, 1 false negative, and 224 true positives. Despite its capacity to detect spam communications, Decision Trees' poor accuracy highlights its limitations in reliably identifying non-spam messages.

#### 4.3.2.4 Random Forest Calssifeier

A screenshot of a computer program

Description automatically generated

Figure 39: Random Forest using Count Vectorization

Random Forests, like Decision Trees, achieved a high recall of 100% for spam messages during the analysis, but had a low accuracy of 24%. This disparity between accuracy and recall demonstrated the model's difficulty in correctly categorizing non-spam communications. Thus, Random Forests revealed a poor overall accuracy of 36.20%. This finding is verified by the confusion matrix, which shows 172 true negatives, 697 false positives, one false negative, and 224 true positives. Despite its ability to detect spam communications, Random Forests struggled to retain accuracy, resulting in mediocre performance in reliably classifying non-spam messages.

#### 4.3.2.5 Conclusion

When count vectorization was used for model building, Naive Bayes outperformed the other models, with good accuracy and balanced precision-recall scores. SVM showed potential with a high recall but needed additional refinement in accuracy. Decision Trees and Random Forests performed poorly when compared to the other models, because of their low accuracy ratings.

### 4.3.3 Using TFIDF Vectorization

A screenshot of a computer program

Description automatically generated

Figure 40: Using TFIDF Vectorization

Before model building, all the pre-model building steps were similar except the vectorization. Instead of Count Vectorization, TFIDF vectorization was implemented.

#### 4.3.3.1 Support Vector Machine

A screenshot of a computer program

Description automatically generated

Figure 41: SVM using TFIDF Vectorization

SVM obtained 96.16% accuracy using TF-IDF vectorization. The precision for spam detection increased to 96%, with a recall of 85% and an F1 score of 90%. The confusion matrix identified 861 true negatives, 8 false positives, 34 false negatives, and 191 true positives. This demonstrated SVM model's improved ability to accurately categorize both spam and non-spam communications.

#### 4.3.3.2 Naïve Bayes

A screenshot of a computer

Description automatically generated

Figure 42: Naive Bayes using TFIDF Vectorization

Naive Bayes performed well with TF-IDF vectorization, obtaining an accuracy of 96.16%. The precision for spam identification maintained at 96%, with 85% recall and a 90% F1-score. The confusion matrix showed 843 true negatives, 26 false positives, 33 false negatives, and 192 real positives, demonstrating Naive Bayes' ability to correctly categorize spam and non-spam communications.

#### 4.3.3.3 Decision Tree Classier

A screenshot of a computer program

Description automatically generated

Figure 43: Decision Tree using TFIDF Vectorization

Using TF-IDF vectorization, Decision Trees attained an accuracy of 94.61%. The accuracy of spam detection was 88%, with a recall of 85% and an F1-score of 87%. The confusion matrix indicated 843 true negatives, 26 false positives, 33 false negatives, and 192 true positives. Although Decision Trees performed well, they had somewhat lower precision than SVMs and Naive Bayes.

#### 4.3.3.4 Random Forest Classifier

A screenshot of a computer program

Description automatically generated

Figure 44: Random Forest using TFIDF

Random Forests achieved an accuracy of 96.43% using TF-IDF vectorization. The precision of spam detection increased to 99%, with a recall of 84% and an F1-score of 91%. The confusion matrix yielded 867 true negatives, 2 false positives, 37 false negatives, and 188 true positives, demonstrating Random Forests' superior ability to effectively categorize spam and non-spam communications when compared to other models.

#### 4.3.3.5 Conclusion

Overall, the models performed better with TF-IDF vectorization than Count Vectorization. Naive Bayes and Random Forests were the most accurate and precise for spam identification, whereas Decision Trees had somewhat lower precision but still performed quite well. Further improvements and ensemble strategies could improve the performance of these models in spam classification tasks.

### 4.3.4 Employing Grid Search to find the best hyperparameters for each model and tuning them with the hyperparameters.

For the next set of models, it was decided to run a grid search for each of the 4 models and find their best hyperparameters and build the models again using the found hyperparameters. The pre-model building steps were the same, TFIDF being the vectorizer.

#### 4.3.4.1 Support Vector Machine

A screenshot of a computer program

Description automatically generated

Figure 45: Grid Search for SVM

A screenshot of a computer program

Description automatically generated

Figure 46: Applied hypermeters for SVM

The tuned SVM model had an accuracy of 95.16 percent. It had an accuracy of 94% for non-spam messages and 99% for spam messages. The recall rate for non-spam messages was 100%, however for spam messages it was 77%. The F1-scores for spam and non-spam communications were 87% and 97%, respectively. These findings showed that hyperparameter adjustment enhanced the model's capacity to reliably categorize both spam and non-spam communications.

#### 4.3.4.2 Naïve Bayes

A screenshot of a computer program

Description automatically generated

Figure 47: Grid Search for Naive Bayes

A screenshot of a computer program

Description automatically generated

Figure 48: Applied hypermeters for Naive Bayes

The tuned Naive Bayes model has an accuracy of 97.17%. It had an accuracy of 98% for non-spam messages and 92% for spam messages. The recall rate for non-spam communications was 98%, while for spam messages it was 94%. The F1-scores for spam and non-spam communications were 93% and 98%, respectively. These findings showed that the model's accuracy in categorizing both categories of messages improved following hyperparameter adjustment.

#### 4.3.4.3 Decision Tree Classier

A screenshot of a computer program

Description automatically generated

Figure 49: Grid Search for Decision Tree

A screenshot of a computer

Description automatically generated

Figure 50: Applied hypermeters for Decision Tree

The tuned Decision Tree model achieved an accuracy rate of 94.52%. It had an accuracy of 96% for non-spam messages and 88% for spam messages. The recall rate for non-spam communications was 97%, whereas for spam messages it was 85%. The F1-scores for spam and non-spam communications were 86% and 97%, respectively. This demonstrated the model's enhanced ability to effectively categorize both spam and non-spam messages after hyperparameter adjustment.

#### 4.3.4.4 Random Forest Classifier

A screen shot of a computer program

Description automatically generated

Figure 51: Grid Search for Random Forest

A screenshot of a computer program

Description automatically generated

Figure 52: Applied hypermeters for Random Forest

The tuned Random Forest model had an accuracy of 96.71 percent. It achieved an accuracy of 96% for non-spam messages and 99% for spam messages. The recall rate for non-spam messages was 100%, however for spam messages it was 85%. The F1-scores for spam and non-spam messages were 91% and 98%, respectively. These findings highlight the model's improved ability to effectively categorize spam and non-spam messages following hyperparameter adjustment.

#### 4.3.4.5 Conclusion

Overall, all models' performance increased dramatically following hyperparameter adjustment, resulting in greater accuracy, precision, recall, and F1 scores. In terms of accuracy and balanced precision-recall scores, Naive Bayes outperformed all other models.

# 5.0 Results and Discussion

## 5.1 Introduction

This chapter provided a full description of the results and comments that arose from the model assessments for the spam email classification task. The major goal was to understand the performance of several machine learning models, their comparisons, and the decision-making process that resulted in the selection of the best model. The debate covered the subtleties of each model's merits and limitations, with an emphasis on crucial metrics like accuracy and precision. In addition, this part delves into the complexities of model deployment, providing insights into the chosen model's real-world applicability and impact. The following subsections gave a full breakdown of the evaluation results, comparative analyses, and the factors that influenced the ultimate model deployment choice.

## 5.2 Model Evaluation

As mentioned earlier, during the model-building phase, the four models (SVM, Naive Bayes, Decision Tree, and Random Forest) were modified and tuned to choose the optimal model. A script was run to arrange the outputs of each model after each step, allowing for a detailed study and selection of the best model.

A screenshot of a computer

Description automatically generated

Figure 54: Script Ran to organize and display result of each model across different vectorization techniques and iterations.

The screenshot above showed that Naive Bayes consistently outperformed other models in terms of precision, recall, and accuracy across different vectorization procedures and iterations. Notably, after hyperparameter tuning, the Naive Bayes model emerged as the best option, with the highest precision, recall, and accuracy of all models. This decision was based on its balanced performance in reducing false positives and false negatives, resulting in greater overall accuracy in spam classification tasks. Followed by Naive Bayes, SVM using TFIDF vectorization, showed significant improvement in precision, recall, and accuracy also demonstrating its ability to effectively classify both spam and non-spam messages. In Conclusion, the Naive Bayes model, tuned hyperparameters from grid search, proved to be the most reliable and successful model for deployment.

## 5.3 Model Deployment

### 5.3.1 Web-App

* **Dumped Model and Vectorizer:**

After the optimal model was chosen, it was decided to save the Naïve Bayes model and TFIDF vectorizer using the pickle tool.

A screenshot of a computer program

Description automatically generated

Figure 55: Saving the Model and Vectorizer using the Pickel Tool

A Web-App Implementation in PyCharm using python virtual environment in Streamlit was created. A new file called app.py was added to the project along with the exported model and vectorizer attained from Collab Notebook. The steps used to build the web app are as follows:

* **Imported and Downloaded Libraries and Tools**

**A screenshot of a computer screen

Description automatically generated**

Figure 56: Import and Download Libraries and Tools

Imported the essential libraries, such as Streamlit for web application development, Pickle for loading pre-trained models and vectorizers, and NLTK for natural language processing workloads. Downloaded NLTK resources required for tokenization and stopwords removal. PorterStemmer library was also initialized in a variable called porter\_stemmer.

* **Function for Preprocessing input data**

A screen shot of a computer program

Description automatically generated

Figure 57: Code snippet of text preprocessing function for user input

The same text preprocessing function, used for text preprocessing of the ‘text’ column of the dataset in Google Collab, was also defined in this project to preprocess the input data.

* **Loaded the model and Vectorizer**

A computer code with text

Description automatically generated with medium confidence

Figure 58: Use pickle to load the vectorizer and the model

* **Added a title for the web-app and a text area for the user to input data**

A screen shot of a computer

Description automatically generated

Figure 59: Added title for the web-app and a text area for the user to input data

A title for the web-app saying “Spam Email Classifier” was added along with a user text area for user to key in data.

* **Added a button called Predict and designed the code the perform necessary actions (preprocess, vectorize user input) if the button is pressed.**

A screenshot of a computer program

Description automatically generated

Figure 60: Functions to occur if the predict button is clicked on

The web-app included a button for triggering the prediction. When the button is hit, the input text is preprocessed, vectorized, and sent to the pre-trained model for prediction. The model's prediction is used to present the outcome as either "Spam:(" or "Ham:)".

* **Tested the Web-App**

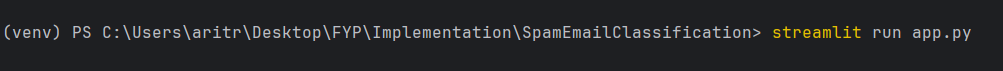


Figure 61: Code to run the web-app pre deployment

To test the app, the function called streamlit run app.py was used to run in the terminal. Soon after a few seconds, a new tab in the web browser was opened with the web app.

A screenshot of a computer

Description automatically generated

Figure 62: View of the web App

* **Testing ham message**

**A screenshot of a computer

Description automatically generated**

Figure 63: Testing ham message

* **Testing Spam Message**

**A screenshot of a computer

Description automatically generated**

Figure 64: Testing Spam Message

### 5.3.2 Web-App Deployment

Render a free cloud-app hosting service was chosen for deployment. To deploy the app in render the following steps were taken:

**1. Created a new file called requirements.txt and added it to the project folder.**

A screenshot of a computer

Description automatically generated

Figure 65: Contents of the requirements.txt file

**2. Used Git to connect and push code to the developer’s GitHub repository**



Figure 66: Used the git add and commit to push code to repository.

1. **Set up app in Render**

In the Render-Website, on the top right, the plus icon was clicked and webservice was selected.

A screenshot of a web service

Description automatically generated

Figure 67: Create a new web-service

1. **Build and Deploy from Git Repository was selected as the option to proceed.**

**A screenshot of a web service

Description automatically generated**

Figure 68: Build and deploy from Git Repository selected.

1. **Connect to the repository from the list of repositories.**

**A screenshot of a computer

Description automatically generated**

Figure 69: Select desired Git Repo from the list

1. **Unique name keyed in for the URL and start command box was set**

**A screenshot of a computer

Description automatically generated**

Figure 70: Unique Name Given

1. **Start Command filled in.**

**A screenshot of a computer

Description automatically generated**

Figure 71: Adding Start Command

The rest of the fields were left untouched and the create web service button at the end was clicked.

1. **Create the Web Service**

A screenshot of a web service

Description automatically generated

Figure 72: Create web service button in Render.

The app took about 15 minutes to deploy and finally a link was generated to be used.

A screenshot of a computer

Description automatically generated

Figure 73: Web App link of the deployed app in Render

1. **Running the app through the link**

**A screenshot of a computer

Description automatically generated**

Figure 74: Live App

# 6.0 Conclusion

## 6.1 Critical Evaluation

The entire project's principal aim of creating a spam email classifier using machine learning techniques had been achieved. The method effectively employs a Multinomial Naive Bayes model trained on a dataset of spam and ham emails. The use of a TFIDF Vectorizer improves the model's grasp of the text data. The online application designed using Streamlit has an easy-to-use interface for guessing if an email is spam or not. The achievement was attained due to the successful integration of preprocessing steps, model training, and web application development to provide a functioning and practical spam classifier.

This initiative benefits the community and companies by providing a real answer to the ongoing issue of email spam. An accurate spam classifier has useful applications in email filtering systems, improving user experience and security. In sectors where email communication is critical, the installation may lessen the likelihood of users falling victim to phishing or other criminal actions. Furthermore, the project's open-source nature allows others to learn from the code and perhaps alter or enhance it to meet their own requirements.

**Strengths of the project:**

* **Effective Model Selection:** Using a Multinomial Naive Bayes model for text classification is suitable, especially given the dataset and purpose.
* **Streamlined User Interface**: The Streamlit web application offers consumers a straightforward and easy way to interact with the spam classifier.
* **Preprocessing Techniques:** The addition of text preparation techniques such as lowercasing, tokenization, and stemming improves the model's capacity to interpret and categorize email text correctly.
* **Model Persistence:** The use of Pickle for model and vectorizer serialization allows for easy deployment and reuse of trained components

## 6.2 Limitation of the project

* **Imbalanced Dataset:** Since the dataset was imbalanced with more ham emails than spam emails, techniques like SMOTE had to be utilized to balance the dataset. A balance dataset with equal or approximately equal records of spam and ham messages could result in the chance of the models ‘classifying more accurately with higher precision.
* **Narrow Scope:** This project focuses on email categorization using a Multinomial Naive Bayes model. While effective for this challenge, the method may not be appropriate for more sophisticated NLP tasks or varied text data.
* **No integration with any email-service:**  The ideal approach for the project was to build the model and try to implement the model in an email service so the process of judging emails and sending them either to inbox or spam folders could be displayed. This could not be accomplished because of time constraints and required knowledge.

## 6.3 Recommendations

* **Balanced Dataset**

To increase spam detection performance, a balanced sample with equal proportions of spam and ham emails could have been utilized. Since the dataset was imbalanced, SMOTE had to be applied to data before model building.

* **Advance Models**

To identify subtle patterns in email language, further advances might require researching other models like deep learning architectures, in addition to the Multinomial Naive Bayes model. Since deep learning models like CNN and ANN require a large dataset to perform well and the dataset at hand was not particularly, these deep learning models were not utilized.

* **Email Service Integration**

If the developer had a chance to further improve the model, integrating the model into a real time email service like Google or Outlook would be the ideal option to go forward with. This can be achieved with time and further knowledge about email integration.

# Appendices

## References

CampusX. (2022, September 19). Machine Learning Course | Lecture 2 - Types of Machine Learning [Video]. YouTube. https://www.youtube.com/watch?v=YncZ0WwxyzU&t=790s&ab\_channel=CampusX

Capozzi, C. (2017, September 26). The financial & non-financial theories of motivation. Bizfluent. https://bizfluent.com/info-7962820-financial-nonfinancial-theories-motivation.html

Cisco. (n.d.). What is spam? Cisco. Retrieved from https://www.cisco.com/c/en/us/products/security/email-security/what-is-spam.html

Driscoll, M. (2023, July 21). Jupyter Notebook: An introduction. Real Python. https://realpython.com/jupyter-notebook-introduction/

Google Research. (n.d.). Welcome To Colaboratory. https://research.google.com/colaboratory/

Harshith. (2019, November 21). Text preprocessing in Natural Language Processing using python. Medium. https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8

Hotz, N. (2023, January 19). What is CRISP DM?. Data Science Process Alliance. https://www.datascience-pm.com/crisp-dm-2/

Iqbal, K., & Khan, M. S. (2022). Email classification analysis using machine learning techniques. Applied Computing and Informatics. doi:10.1108/aci-01-2022-0012

Jhaveri, R. H., Revathi, A., Ramana, K., Raut, R., & Dhanaraj, R. K. (2022). A review on Machine Learning strategies for real-world engineering applications. Mobile Information Systems, 2022, 1–26. doi:10.1155/2022/1833507

Kaspersky Lab. (n.d.). Damage caused by spam. Kaspersky IT Encyclopedia. Retrieved July 17, 2023, from https://encyclopedia.kaspersky.com/knowledge/damage-caused-by-spam/

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 82(3), 3713–3744. doi:10.1007/s11042-022-13428-4

Lakshmi, R. D., et al. (2010). Supervised Learning Approach for Spam Classification. International Journal on Computer Science and Engineering, 2(8), 2760-2766. https://www.researchgate.net/publication/50235003\_Supervised\_Learning\_Approach\_for\_Spam\_Classification\_Analysis\_using\_Data\_Mining\_Tools

Lin, Y. (n.d.). How many people use email in 2023? [Feb 2023 update]. Oberlo. <https://www.oberlo.com/statistics/how-many-people-use-email#:~:text=According%20to%20a%20study%20that,half%20of%20the%20global%20population>.

Lutkevich, B. (2023, February 7). What is tokenization?. Security. <https://www.techtarget.com/searchsecurity/definition/tokenization>

Mbaabu, O. (2020, December 11). Introduction to random forest in machine learning. Introduction to Random Forest in Machine Learning. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/#:~:text=A%20random%20forest%20is%20a%20machine%20learning%20technique%20that’s%20used,consists%20of%20many%20decision%20trees.

Madhavan, M. V., Pande, S., Umekar, P., Mahore, T., & Kalyankar, D. (2021). Comparative analysis of detection of email spam with the aid of machine learning approaches. IOP Conference Series. Materials Science and Engineering, 1022(1), 012113. doi:10.1088/1757-899x/1022/1/012113

Mailpro (2023) ‘Everything you need to know about Gmail’s spam filters and how to avoid them’, Mailpro Email Marketing. Available at: https://www.mailpro.com/blog/gmail-spam-filter (Accessed: 01 August 2023).

Mazumder, S. (2022, December 1). 5 techniques to handle imbalanced data for a classification problem. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

Microsoft Support. (n.d.). Overview of the Junk Email Filter. Retrieved from https://support.microsoft.com/en-us/office/overview-of-the-junk-email-filter-5ae3ea8e-cf41-4fa0-b02a-3b96e21de089#:~:text=How%20the%20Junk%20Email%20Filter,level%20is%20set%20to%20Low.

Mishra, S., & Soni, D. (2022). SMS Phishing Dataset for Machine Learning and Pattern Recognition (Version 1) [Data set]. Mendeley Data. https://doi.org/10.17632/f45bkkt8pr.1

OpenAI. (2021). ChatGPT (Version 3.5) [Computer software]. Retrieved from https://openai.com

Pandas introduction (no date) W3 schools. Available at: https://www.w3schools.com/python/pandas/pandas\_intro.asp (Accessed: 03 August 2023).

Preprocessing steps in Natural Language Processing (NLP). Educative. (n.d.). https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp

Priya B, Nandhini J.M, & Gnanasekaran T. (2021). An analysis of the applications of natural Language Processing in various sectors. In Advances in Parallel Computing. Advances in Parallel Computing. doi:10.3233/apc210109

R advantages and disadvantages - javatpoint (no date) www.javatpoint.com. Available at: https://www.javatpoint.com/r-advantages-and-disadvantages (Accessed: 03 August 2023).

Radicati Group. (2020). Email Statistics Report, 2020-2024. Retrieved from https://www.radicati.com/wp/wp-content/uploads/2020/01/Email\_Statistics\_Report,\_2020-2024\_Executive\_Summary.pdf

Robert. (2021, March 22). Pros and cons of python in machine learning. Zarantech. https://www.zarantech.com/blog/pros-and-cons-of-python-in-machine-learning/

Shaik, T., Tao, X., Li, Y., Dann, C., McDonald, J., Redmond, P., & Galligan, L. (2022). A review of the trends and challenges in adopting natural language processing methods for education feedback analysis. IEEE Access: Practical Innovations, Open Solutions, 10, 56720–56739. doi:10.1109/access.2022.3177752

Sharma, N., Sharma, R., & Jindal, N. (2021). Machine learning and deep learning applications-A vision. Global Transitions Proceedings, 2(1), 24–28. doi:10.1016/j.gltp.2021.01.004

Support Vector Machine (SVM) algorithm - javatpoint. www.javatpoint.com. (n.d.). https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

Taye, M. M. (2023). Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. Computers, 12(5), 91. doi:10.3390/computers12050091

TechTarget. (2023, July 14). Privacy compliance. Retrieved from https://www.techtarget.com/searchcio/definition/privacy-compliance

Tennis, J. (2020). The importance of text preprocessing in NLP and machine learning. Medium. <https://towardsdatascience.com/the-importance-of-text-preprocessing-in-nlp-and-machine-learning-58bc8c0a5a>

Tetko, I. V., Livingstone, D. J., & Luik, A. I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. Journal of Chemical Information and Modeling, 35(5), 826-833.

Tutorialspoint. (n.d.). Scikit learn - introduction. https://www.tutorialspoint.com/scikit\_learn/scikit\_learn\_introduction.htm

University of California, Irvine School of Information and Computer Sciences. (n.d.). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

## Project Proposal Form

A document with text and numbers

Description automatically generated with medium confidence

## A close-up of a document Description automatically generated

## A paper with text on it Description automatically generatedA paper with text on it Description automatically generated

## A close-up of a document Description automatically generatedA close-up of a document Description automatically generatedA close-up of a document Description automatically generatedA screenshot of a computer Description automatically generatedA close-up of a document Description automatically generated

## Ethics Form

A close-up of a form

Description automatically generated

A close-up of a form

Description automatically generated

A close-up of a document

Description automatically generatedA close-up of a document

Description automatically generated

## Gant Chart

A screenshot of a computer screen

Description automatically generated

## Poster

**A poster of a project

Description automatically generated with medium confidence**

## Sample Code

### Google Collab Notebook

A screenshot of a computer

Description automatically generated

### Pycharm

A screen shot of a computer

Description automatically generated