SYNOPSIS/EXECUTIVE SUMMARY

This project proposes a spam detection system that utilizes advanced machine learning techniques to accurately identify and filter out spam emails.

The key steps involved in this project include:

1. **Data Loading:** Start by loading the datasets and performing initial checks to verify their integrity, such as checking for duplicates and inconsistencies.
2. **Data Cleaning:** Clean and format the raw datasets to ensure consistency, handle missing values, and prepare the data for analysis.
3. **Exploratory Data Analysis (EDA):** A detailed analysis will be conducted to understand the structure of the data, identify patterns, and gain insights that will inform the model-building process.
4. **Text Preprocessing:** For the text data, we will carry out preprocessing tasks such as normalization, tokenization, and vectorization to prepare the data for model training.
5. **Model Training:** Using the preprocessed data, we will build and train machine learning models aimed at detecting phishing attempts across different types of data.

The goal is to develop a spam detection system that accurately identifies and filters out spam emails, protecting users from harmful activities and improving their email experience.

PROBLEM DEFINITION

- Common Problems with Old Information Systems:

As technology evolves and spam techniques become more sophisticated, older information systems designed for spam detection may encounter several challenges:

1. **Outdated Algorithms and Techniques:** Older systems may rely on outdated machine learning algorithms or feature extraction techniques that are no longer effective against modern spam tactics.
2. **Limited Feature Extraction:** Older systems may not be able to extract the full range of features that are relevant for detecting current spam types, such as social engineering techniques or language models.
3. **Inability to Handle New Spam Variants:** As spammers continuously adapt their techniques, older systems may struggle to detect new types of spam, leading to false negatives and reduced effectiveness.
4. **Scalability Issues:** Older systems may not be able to handle the increasing volume of emails and the complexity of modern spam detection techniques, leading to performance degradation.
5. **Integration Challenges:** Integrating older systems with newer technologies or platforms can be difficult and time-consuming, hindering their ability to leverage the latest advancements in spam detection.
6. **Lack of Real-time Updates:** Older systems may not be able to receive timely updates to their machine learning models or feature extraction techniques, limiting their ability to adapt to evolving spam trends.

7. **Security Vulnerabilities:** Older systems may have security vulnerabilities that can be exploited by spammers to bypass detection or compromise user data.

➢ Proposed System Improvements:

A new system can address these problems by: -

- **Utilize Advanced Machine Learning Algorithms:** Employ state-of-the-art algorithms like deep learning, recurrent neural networks, and natural language processing techniques to improve accuracy and adaptability.
- **Expand Feature Extraction:** Extract a wider range of features, including contextual information, social engineering indicators, and language model-based features to capture the nuances of modern spam.
- **Ensure Scalability:** Design the system to handle large volumes of emails and scale efficiently to accommodate future growth.
- **Integrate Seamlessly with Existing Systems:** Ensure compatibility with existing email infrastructure and platforms to minimize disruption.
- **Prioritize Security:** Implement robust security measures to protect user data and prevent unauthorized access.
- **Continuous Monitoring and Improvement:** Regularly evaluate the system's performance and make necessary adjustments to address new challenges and improve accuracy.

➢ Objectives of the Project:

A spam detection system aims to achieve the following objectives:

1. **Reduce Spam:** Minimize the amount of spam that reaches users' inboxes, protecting them from unwanted and potentially harmful content.
2. **Improve User Experience:** Enhance the overall email experience by providing a cleaner and more secure inbox.
3. **Protect User Privacy:** Safeguard users from phishing attacks and other malicious activities that can compromise their personal information.
4. **Increase Productivity:** Reduce the time and effort users spend dealing with spam, allowing them to focus on more productive tasks.
5. **Ensure Compliance:** Comply with relevant regulations and industry standards related to spam filtering and privacy.
6. **Continuously Improve:** Develop a system that can adapt to evolving spam techniques and provide ongoing improvements in accuracy and effectiveness.

LITERATURE REVIEW

Spam detection has been a significant research area due to the increasing prevalence of unsolicited and often malicious emails. Researchers have explored various techniques to effectively identify and filter spam messages.

- Traditional Techniques

  - **Rule-based filtering:** This approach involves creating rules based on specific patterns or characteristics of spam emails, such as the presence of certain keywords, unusual email addresses, or excessive capitalization.
  - **Bayesian filtering:** This probabilistic approach calculates the probability of an email being spam or ham based on the frequency of words and phrases.
  - **Keyword-based filtering:** This technique identifies spam based on the presence of specific keywords or phrases that are commonly found in spam emails.
  - **Bag-of-Words:** This technique represents emails as vectors of word frequencies.
  - **TF-IDF:** Term Frequency-Inverse Document Frequency weights words based on their importance in the document and the corpus.
  - **N-grams:** Sequences of words or characters are used to capture contextual information.

- Challenges and Future Directions:
  - **Evolving Spam Techniques:** Spammers continuously adapt their tactics, making it challenging to keep spam detection systems up-to-date.
  - **False Positives and Negatives:** Balancing the trade-off between detecting spam (recall) and avoiding false positives (precision) is crucial.
  - **Computational Efficiency:** Spam detection systems must be efficient to handle large volumes of emails in real-time.
  - **Integration with Other Security Measures:** Combining spam detection with other security measures, such as phishing detection and malware analysis, can provide more comprehensive protection.
  - **User Experience:** The system should be easy to use and provide clear feedback to users.

- Conclusion

Spam detection is a complex problem that requires a combination of techniques to achieve effective results. While traditional methods have been used successfully, machine learning and deep learning approaches have shown promise in addressing the challenges of modern spam. Future research should focus on developing more robust and adaptable spam detection systems that can keep pace with evolving spam tactics.

METHODOLOGY

1. Data Collection and Preprocessing

- **Dataset Acquisition:** Gather a large and diverse dataset of emails, including both legitimate (ham) and spam emails.
- **Data Cleaning:** Remove irrelevant information such as headers, footers, and HTML tags.
- **Feature Extraction:** Extract relevant features from the emails, such as:
    - **Text-based features:** Word frequency, n-grams, TF-IDF scores
    - **Email header features:** Sender address, recipient address, subject line, date
    - **Image and attachment analysis:** If applicable, analyze images and attachments for suspicious patterns

2. Machine Learning Model Selection and Training

- **Model Selection:** Choose a suitable machine learning algorithm based on the dataset characteristics and desired performance. Common choices include:
    - Naive Bayes
    - Support Vector Machines (SVM)
    - Random Forest
    - Deep learning models (e.g., recurrent neural networks, convolutional neural networks)
- **Training:** Train the selected model on the labeled dataset to learn the patterns and characteristics of spam and ham emails.

3. Model Evaluation

- **Cross-Validation:** Use cross-validation techniques to evaluate the model's performance on different subsets of the data.
- **Metrics:** Calculate metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.

4. Filtering and Deployment

- **Bayesian Filtering:** Use Bayesian techniques to update the model's parameters based on user feedback.
- **Deployment:** Deploy the spam detection system in a production environment.

5. Continuous Improvement

- **Monitoring:** Monitor the system's performance and identify areas for improvement.
- **Model Updates:** Regularly update the machine learning model to adapt to new spam techniques.

By following this methodology, we can develop a robust and effective spam detection system that protects users from harmful emails.

TOOLS

For this project, I propose using a combination of the following tools and platforms:

1. Python:
   - Programming Language: Python is a popular choice for data analysis and machine learning due to its readability, versatility, and extensive libraries.
   - Libraries:
     NumPy: For numerical computations and array operations.
     Pandas: For data manipulation and analysis.
     Scikit-learn: For machine learning algorithms and tools.
     Matplotlib: For data visualization.
     Seaborn: For statistical data visualization.
     NLTK (Natural Language Toolkit): For natural language processing tasks, including text cleaning, tokenization, and feature extraction.

2. Kaggle: A platform for data science competitions and collaboration, where you can find datasets and benchmark your models.

3. Jupyter Notebook: Jupyter Notebook provides an interactive environment for writing and executing Python code, making it ideal for data exploration, analysis, andvisualization.

4. Cloud Platform:

Depending on the project's scale and requirements, a cloud platform like Google Cloud Platform, Amazon Web Services, or Microsoft Azure can provide scalable computing resources and cloud-based tools.

RESEARCH GAP

Despite significant advancements in spam detection, several research gaps remain:

1. **Evolving Spam Techniques:** Spammers constantly adapt their techniques to evade detection. Research is needed to develop techniques that can effectively identify and mitigate new spam variants.
2. **Social Engineering Spam:** Phishing attacks and other social engineering techniques are becoming increasingly sophisticated. Developing methods to detect and prevent these types of attacks is a critical area of research.
3. **Deepfakes and Synthetic Content:** The emergence of deepfake technology poses new challenges for spam detection. Research is required to develop techniques that can accurately identify and filter out synthetic content.

4. **Cross-Platform Spam:** Spam can originate from various platforms, such as email, social media, and messaging apps. Research is needed to develop unified approaches that can detect spam across different platforms.
5. **Privacy and Ethical Considerations:** Ensuring the privacy and security of user data while effectively detecting spam is a complex challenge. Research is required to develop privacy-preserving techniques and address ethical concerns.

Addressing these research gaps will be crucial for developing more effective and robust spam detection systems that can keep pace with the evolving landscape of spam threats.

REFERENCES

- Journal articles
- Books
- Research papers
- Kaggle dataset: https://www.kaggle.com/c/titanic
- Scikit-learn documentation: https://scikit-learn.org/stable/
- Pandas documentation: https://pandas.pydata.org/docs/
- Matplotlib documentation: https://matplotlib.org/stable/index.html

CHAPTER 1: INTRODUCTION

More than 4.5 billion people in our technological age find it convenient to use the Internet for their convenience, making it a necessary component of our everyday life. Without the Internet, it would have been hard to do anything — whether it be for entertainment, study, e-commerce, social media connections, or just about everything else one could think of. Emails also developed alongside the internet, and Internet users regard emails as a reliable form of communication. Over time, email services have become a powerful tool for communicating a variety of information. The email system is one of the most widely used and effective forms of communication. Email's ease-of-use and speedy communication capabilities are what have made it so popular. The "Internet," the ultimate source of knowledge, does, however, also have certain immoral features. It's known as internet spam. Spams come in many forms, but in this research the authors only address email spam. Due to the surge in e-mail use, spam assaults on Internet users are also increasing. Spam may be sent from anywhere on the planet by anybody who has access to the Internet and nefarious intentions.

Email spam is a collection of promotional text or images that are sent with the aim of stealing money, promoting goods or websites, engaging in phishing, or spreading viruses. Whether intentionally or unintentionally, if you click on this spam, your computer might become infected with a virus, you can waste network resources, and you might waste time. These emails are distributed to a significant number of recipients in bulk. The main motivations for email spam include information theft, money-making, and sending multiple copies of the same message, all of which not only have a negative financial impact on a business but also upset recipients. Spam emails generate a lot of unnecessary data, which decreases the network's capacity and efficacy in addition to aggravating the users. Spam is a major problem that has to be addressed, which is why spam filtering is essential. An email's body and subject line are always the same in every message. By using content filtering, spam may be located. The method of spam detection in emails is dependent on the words that have been used in it, whether the words are pointing out that the letter is spam or not. For instance, phrases used in service or product recommendations. There are two different approaches that may be used to identify spam in email: knowledge engineering and machine learning (ML). A technique based on networks called knowledge 1 engineering measures whether an email is spam or not by analysing its IP address and network address in conjunction with roughly stated criteria. Although this method has provided remarkably precise results, updating the rules takes time and is not always convenient for users. In this project spam detection is done using the ML approach.Since there are no predefined rules using ML, it is more effective than Knowledge Engineering. It uses a technique called Natural Language Processing (NLP), a crucial area of artificial intelligence. NLP focuses on assessing, extracting, and retrieving useful information from text data and gleaning text-based insights that resemble human language. The suggested efficient spam mail detection offers a comparison of the most important machine learning models for spam detection. In computer terminology, spam refers to unwelcome material. It is typically used to represent spam messages, and it is now also used to denote spam phone calls sent by SMS and Instant Messenger (IM).

Unwanted, unsolicited email advertising a product for sale is known as spam. Email spam is frequently referred to by the terms spam emails, unsolicited bulk email (UBE), or unsolicited commercial email (UCE). While occasionally also promoting phone or other sales channels, spam

typically pushes online transactions. Spammers are people that specialise in sending spam. Companies pay spammers to send emails on their behalf. To transmit these messages, spammers have created a variety of computer tools and methods. Additionally, spammers operate their own web shops and sell their products there.

By and large, emails from reputable sources are ignored when using the term "spam email," regardless of whether the content is objectionable. One example would be the never-ending list of jokes sent by friends. Despite the fact that they have some common characteristics with spam in general, email infections, games, and other malware (short for malicious code) are not typically classified as spam. Antispam federation in particular regularly refers to messages that are not spam as "ham." Because spam is emotive, some recipients may view a message as spam while others may view it as an invitation. The majority of the time, spammers are paid to promote pornographic websites, products, and organisations; they are skilled at sending spam messages. There are a few well-known spammers who are in charge of a sizable portion of the spam and have shunned legal action. Individual website administrators can send their own spam, but spammers have extensive email lists and excellent tools for avoiding spam conduits and avoiding detection. Present-day showcase companies are being taken advantage of by spammers that have found a gap in the market.

According to a public statement made by a broadband skilled professional, the majority of spam messages are currently delivered from "Trojan" PCs. Owners or users of Trojan computers have been tricked into running programmes that allow spammers to send spam from a computer without knowing who the client is. Security flaws in the operating system, the client's operating system, a software, or an email client are routinely exploited by Trojan programming. The PC introduces a distraction programme while browsing a malicious website. Their PC may become the source of thousands of spam messages each day if they have obscure clients. The speculation that prompted this examination emerged from the requirement for a choice to stack shedding in streaming conditions. Some exploration has been finished on computational sewing, which is the fine-grained evacuation of calculations trying to recapture computer processor cycles. While computational shedding is viable in specific circumstances a more broad methodology is expected to build its feedback information throughput when the ongoing processing framework is under load. The proposed arrangement is called Algorithmic Transformation. The speculation states: "There is practicality to carry out transformation/closure estimation calculations in the ongoing computation framework under load, where the estimations performed are adequately perplexing and options with trade effectiveness and recuperated costs are accessible'. To examine this theory, executing various spam identification models to assess their presentation spam discovery has been chosen. Spam email squeezes into the stream climate like mail waiters should manage erratic email rates as they happen, and discovery strategies are adequately intricate.

More than 4.5 billion people in this period of growth believe it desirable to use the Internet for their benefit, making it an essential part of our daily routines. Be it for obtaining anything, just diverting attention, making an internet purchase, interacting with others through electronic entertainment, or pretty much anything else that could be envisioned, any of these would have been endless without the Internet. Messages similarly emerged with the web; what's more Web purchasers view messages as a reliable technique for correspondence. Email organisations have framed throughout the span of the years into an extraordinary gadget for exchanging numerous

sorts of information. One of the most popular and capable methodologies for correspondence is the email structure. The commonsense and expedient correspondence capacities of email make it so popular. In any case, the "Internet", a conclusive wellspring of information, moreover has explicit tricky points of view. It is called Web spam.

Despite the fact that there are many different types of spam, email spam is the focus of this project. Due to the widespread use of email, spam assaults against Web clients are also on the rise. Spam may be sent from anywhere on earth by anybody with access to the Internet and bad intentions. Email spam is a collection of strange words or content sent with the goal of committing phishing, advertising goods or websites, stealing money, or transmitting viruses. Whether on purpose or by accident, if you click on this spam, your computer might become infected, your association's resources could be wasted, and you could also lose time. Numerous people get these communications that are extensively disseminated.
The key motivations driving email spam are information robbery, cash endlessly making various copies of a comparative message, all of which not simply financially influence an affiliation yet moreover disturbs recipients. As well as disturbing the clients, spam messages produce a lot of unfortunate data that diminishes the association's capacity and suitability. Spams are a troublesome issue to be settled and thus spam filtering transforms into a need. Each email has a comparable plan, consisting of a title and a body. It is practical to recognize spam by filtering .

The method of identifying spam in communications depends on the words used, and whether such phrases indicate that the message is spam or not. For instance, phrases seen in flimsy ads or concepts for organisations. In order to identify email spam, two different methods can be used: data planning and a machine learning (ML) strategy. Data planning is an association-based method that evaluates an email's IP address, network address, and general sets of presented rules to determine if it is spam or not. This method has provided very precise results, but it has also been time-consuming and unsupportive for all clients to resurrect regulations.

In this project, spam acknowledgment is done using the ML approach. ML procedure is more effective than the Data Planning strategy since it incorporates no course of action of rules. A development like Ordinary Language Taking care of (NLP), which is a huge subfield of Man-made cognizance, is used. NLP manages isolating, removing, and recovering important information from text data as well as gathering linguistic bits that resemble human speech from the text. The suggested practical evidence for identifying spam mail compares the vast artificial intelligence models on spam regions.

CHAPTER 2: Objective and Scope of the Project

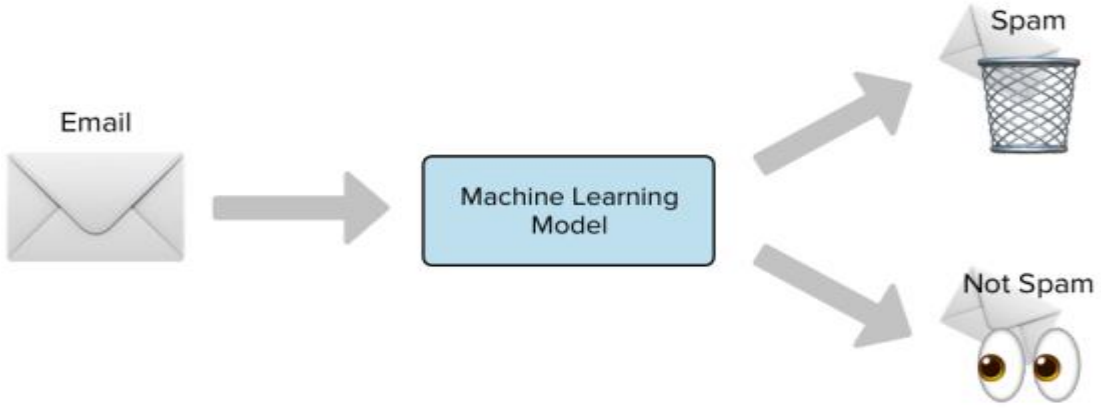
Objective

The primary objective of this project is to create an effective, efficient, and adaptive system for detecting and filtering email spam. Spam emails, often unsolicited, irrelevant, or malicious, represent a significant threat to both individual users and organizations. By implementing an advanced spam detection system, this project aims to address challenges such as inbox clutter, reduced productivity, and heightened security risks.

Furthermore the objective of this project is to analyse the impact and issues of spam messages on email structure. To make an effective email arrangement process that is consolidated with the fitting email framework and arranging the approaching messages as spam and authentic sends. The proposed framework contrasts and the various kinds of existing characterization ways to deal with perceiving the powerful and strong classifier. Moreover, the proposed framework centres around clinical web entry email upkeep administration by utilising a viable classifier.

More modern projects, like Bayesian channels and other heuristic channels, distinguish spam messages by perceiving dubious word examples or word recurrence. They do this by learning the client's inclinations in light of the messages set apart as spam. The spam programming then, at that point, makes rules and applies them to future messages that focus on the client's inbox.

For instance, at whatever point clients mark messages from a particular source as spam, the Bayesian channel perceives the example and consequently moves future messages from that shipper to the spam envelope. ISPs apply spam channels to both inbound and outbound messages. In any case, little to medium undertakings as a rule centre around inbound channels to safeguard their organisation. There are likewise various spam 7 separating arrangements accessible. They can be facilitated in the cloud, facilitated on servers or coordinated into email programming, like Microsoft Outlook. Figure below shows the basic workflow of the project.



The overarching goal is to develop a **machine learning-based spam detection system** capable of automatically identifying and filtering out spam emails, ensuring that legitimate emails reach users

without unnecessary delays. To achieve this, the system must address the following specific objectives:

1. **Accurate Classification**:
    a. Design a system that can distinguish between spam and legitimate emails with high precision and recall.
    b. Reduce misclassification rates to minimize false positives (legitimate emails marked as spam) and false negatives (spam emails marked as legitimate).
2. **Adaptability to Evolving Spam Techniques**:
    a. Develop an adaptive model that can learn from new patterns and trends in spam tactics.
    b. Continuously update itself using real-time data and feedback to remain effective against increasingly sophisticated spammers.
3. **Real-Time Filtering**:
    a. Implement a scalable solution that can process and classify emails in real-time, regardless of the volume of incoming messages.
    b. Ensure the system performs efficiently under heavy loads, suitable for both individual users and large-scale enterprises.
4. **Security and Protection**:
    a. Safeguard users from phishing attacks, malware distribution, and other malicious activities often carried out through spam emails.
    b. Protect sensitive information and reduce risks associated with spam, such as identity theft and data breaches.
5. **User-Centric Design**:
    a. Create a system that is user-friendly, integrates seamlessly with existing email platforms, and provides customizable filtering options.
    b. Offer feedback mechanisms to allow users to improve the system's accuracy by marking emails as spam or not spam.
6. **Cost-Effectiveness**:
    a. Design the solution to be resource-efficient, using minimal computational and storage resources while maintaining high accuracy and reliability.

Scope of the Project

The scope of this project encompasses all stages of spam detection system development, from data collection to deployment. It includes identifying key challenges in email spam detection, exploring state-of-the-art machine learning techniques, and implementing a robust and scalable solution. The key elements of the project's scope are outlined below:

1. **Data Collection and Preprocessing**
    - The project will utilize publicly available datasets, such as the **Enron Email Dataset** or **SpamAssassin Public Corpus**, to train and evaluate the system. These datasets provide labeled examples of spam and non-spam emails, essential for supervised learning.
    - Data preprocessing steps, including cleaning, tokenization, removal of stop words, stemming, and normalization, will be performed to prepare email content for analysis.
2. **Feature Engineering**
    - The system will extract relevant features from email data, such as:
        - **Content-Based Features**: Frequency of certain words, presence of suspicious phrases, and overall text structure.
        - **Metadata-Based Features**: Sender information, email headers, and timestamps.
        - **Behavioral Features**: Number of recipients, links included, and attachment types.
    - Natural Language Processing (NLP) techniques, including word embeddings and n-gram analysis, will be applied to enhance text feature extraction.
3. **Machine Learning Model Development**
    - A variety of machine learning algorithms will be explored and compared for spam detection, including:
        - **Traditional Models**: Naive Bayes, Logistic Regression, and Support Vector Machines (SVM).
        - **Advanced Models**: Random Forest, Gradient Boosting, and Neural Networks (e.g., Convolutional Neural Networks, Recurrent Neural Networks).
    - The models will be trained and evaluated on metrics such as accuracy, precision, recall, and F1-score to identify the most effective solution.
4. **Implementation of the System**
    - The system will be developed using programming languages such as Python, leveraging libraries like Scikit-learn, TensorFlow, and PyTorch.
    - Integration with email platforms will allow real-time email classification, ensuring smooth operation in practical use cases.
5. **Evaluation and Optimization**
    - Extensive testing will be performed using a separate validation dataset to assess the system's performance.
    - Optimization techniques, such as hyperparameter tuning and cross-validation, will be employed to improve the system’s accuracy and efficiency.
    -

6. **Deployment**
    - The spam detection system will be deployed as a web-based service, an API, or integrated directly into email servers.
    - The solution will be designed to scale for individual users and organizations, ensuring compatibility with different email services.
7. **Security and Maintenance**
    - Security measures will be implemented to prevent adversarial attacks on the system, such as attempts to bypass the filter.
    - Regular updates and retraining of the model will ensure the system stays effective against new spam patterns.

Applications and Use Cases

The project will cater to a wide range of users and organizations, including:

1. **Email Service Providers**: Enhance spam filtering in platforms like Gmail, Outlook, and Yahoo.
2. **Businesses and Enterprises**: Protect corporate email systems from phishing attacks and productivity loss due to spam.
3. **Individual Users**: Offer improved spam filtering for personal email accounts, ensuring a smooth communication experience.
4. **E-Commerce Platforms**: Safeguard customers and merchants from phishing scams and other malicious spam.

Significance of the Project

This project will significantly contribute to addressing the challenges posed by email spam. It aims to provide a solution that is not only accurate and efficient but also adaptable to the ever-changing tactics of spammers. By doing so, it enhances user productivity, ensures the safety of sensitive information, and fosters a more secure digital communication environment.

The combination of machine learning, real-time processing capabilities, and user-friendly design ensures that the spam detection system will meet current needs while being scalable and adaptable for future advancements in email technology and spam techniques.

CHAPTER 3: Theoretical Background and Definition of the Problem

Theoretical Background

Email has become an indispensable tool in modern communication, enabling individuals and organizations to share information efficiently. Its ubiquitous presence spans professional correspondence, personal interactions, and business operations. However, the convenience and accessibility of email have also made it a prime target for misuse, primarily through the proliferation of spam emails.

Spam emails, also referred to as unsolicited bulk messages, are typically sent to a large number of recipients without their consent. While initially used for marketing and advertisements, spam has evolved into a significant threat encompassing phishing attacks, malware dissemination, and fraudulent schemes aimed at stealing sensitive information or causing disruption. This misuse has necessitated the development of robust spam detection systems to mitigate the challenges associated with spam emails.

The concept of spam detection emerged alongside the growth of email systems. Early approaches relied on simple rule-based methods, where emails containing certain keywords or phrases were flagged as spam. However, as spammers became more sophisticated, these traditional systems proved inadequate. Spammers began employing techniques such as obfuscation, where text was deliberately misspelled, or using images instead of text to avoid detection. Moreover, they started dynamically changing content and using techniques like domain spoofing, making it increasingly difficult for static systems to keep up.

Challenges in Spam Detection

The process of detecting spam emails involves distinguishing between legitimate (ham) emails and spam emails. This task is fraught with several challenges, making it an ongoing area of research and development. Key challenges include:

1. **High Volume of Email** :
   With billions of emails sent daily, spam accounts for a significant proportion of this traffic. Managing and filtering such a massive volume in real-time is a complex task that requires efficient processing capabilities.
2. **Evolving Spam Techniques**:
   Spammers continuously adapt their methods to bypass detection systems. These tactics include:
   - **Misspelled Words**: Altering text to evade keyword detection (e.g., “fr33” instead of “free”).
   - **Image-Based Spam**: Embedding spam content within images to avoid text analysis.
   - **Link Obfuscation**: Using shortened URLs or redirect chains to disguise malicious links.

- **Dynamic Content**: Regularly altering email content to prevent detection by static filters.

3. **Balancing False Positives and False Negatives**:
   - **False Positives**: Legitimate emails mistakenly flagged as spam can disrupt communication and undermine user trust in the system.
   - **False Negatives**: Failing to detect spam emails allows them to reach the user's inbox, exposing them to potential threats. Achieving a balance between these two metrics is critical for any spam detection system.
4. **Security and Privacy Risks**:
   Spam emails are often carriers of significant security risks, such as:
   - **Phishing Attacks**: Emails designed to trick users into providing sensitive information, such as login credentials or financial details.
   - **Malware Distribution**: Emails containing malicious attachments or links that install harmful software.
   - **Identity Theft**: Using deceptive emails to impersonate trusted entities and steal personal information.
5. **Resource Constraints**:
   Spam detection systems must process large volumes of emails efficiently, requiring computational and storage resources. Ensuring scalability while maintaining performance is a significant challenge.
6. **Integration with Existing Systems**:
   The spam detection solution must integrate seamlessly with existing email platforms, such as Gmail, Outlook, or enterprise email systems, without disrupting user experience.
7. **User Feedback and Personalization**:
   User preferences for spam classification can vary. Incorporating feedback and providing customizable filtering options are essential to enhance system accuracy and user satisfaction.

Definition of the Problem

The problem of spam detection can be summarized as the need to develop an automated system capable of identifying and filtering spam emails accurately and efficiently, while ensuring legitimate emails are not affected. The system must be robust, adaptable, and scalable to address the following challenges:

1. **Content Analysis**:
   Spam emails often employ deceptive language or hidden intent. Advanced text analysis techniques are required to differentiate between genuine and malicious content.
2. **Behavioral Patterns**:
   Spam emails often exhibit specific patterns, such as being sent in bulk, containing links to dubious domains, or using uncharacteristic email headers. Recognizing and leveraging these patterns is crucial for effective detection.
3. **Adaptability to New Techniques**:
   The system must adapt to evolving spam tactics, learning from new data to remain effective against changing trends. This adaptability requires the use of dynamic and self-learning models.

4. **Real-Time Filtering**:
   Email filtering must occur in real-time to prevent spam from reaching users' inboxes without causing noticeable delays in email delivery.
5. **Scalability**:
   The system must handle the growing volume of global email traffic, ensuring that performance remains unaffected even under high loads.
6. **Minimizing False Positives**:
   Incorrectly labeling legitimate emails as spam can disrupt communication and damage user trust. Ensuring minimal false positives is critical for the system's reliability.
7. **User-Centric Approach**:
   The system should offer options for users to customize spam filtering according to their preferences, allowing for a more tailored and effective email management experience.

Role of Machine Learning in Addressing the Problem

Machine learning (ML) has emerged as a powerful approach to solving the spam detection problem. Unlike rule-based systems, ML models can analyze large datasets, learn patterns, and adapt to new scenarios without manual intervention. The following techniques are central to ML-based spam detection:

1. **Text Classification**:
   - Algorithms like Naive Bayes, Logistic Regression, and Support Vector Machines (SVM) are commonly used for classifying email content as spam or legitimate.
   - Deep learning models such as Recurrent Neural Networks (RNNs) and Transformer-based models like BERT are used for more complex text analysis.
2. **Feature Engineering**:
   - Extracting relevant features from email data, including:
     - **Text Features**: Word frequency, n-grams, and sentiment analysis.
     - **Metadata Features**: Sender reputation, email headers, and timestamps.
     - **Behavioral Features**: Number of recipients, inclusion of links or attachments, etc.
3. **Natural Language Processing (NLP)**:
   NLP techniques enable the analysis of email content, helping detect suspicious phrases, sentiment, or context that might indicate spam.
4. **Ensemble Learning**:
   Combining multiple models or classifiers to improve detection accuracy and reduce misclassification rates.
5. **Adaptability and Learning**:
   Machine learning models can be retrained periodically to incorporate new spam patterns, ensuring their relevance and effectiveness over time.

By leveraging advanced machine learning techniques, this project aims to address the challenges associated with spam detection comprehensively.

CHAPTER 4: System Analysis & Design vis-a-vis User Requirements

System Analysis

The system analysis phase is a cornerstone in the development of an effective spam detection system. It serves as the critical first step in ensuring the system aligns with user needs and addresses the challenges inherent in managing and filtering spam emails. This phase focuses on a thorough understanding of the problem, defining both the functional and non-functional requirements, and identifying potential constraints that may impact the development and deployment of the system. By doing so, the system analysis phase establishes a clear vision of the objectives and sets a solid foundation for designing a solution that is not only functional but also scalable, reliable, and efficient.

A robust system analysis process begins with a detailed examination of the existing challenges in spam detection. These include the ever-evolving nature of spam tactics, the need for real-time processing, and the balancing act between minimizing false positives (incorrectly flagging legitimate emails as spam) and false negatives (failing to identify actual spam). Understanding these challenges ensures that the proposed solution can effectively mitigate risks and deliver a high-performance spam detection system.

Additionally, system analysis involves defining the requirements in a structured manner. Functional requirements focus on the core capabilities of the system, such as accurate spam filtering, customizable rules, and real-time processing. Non-functional requirements emphasize the quality attributes, including scalability to handle large email volumes, security to thwart adversarial attacks, and seamless integration with existing email platforms. These requirements collectively guide the design and development efforts, ensuring that the system meets user expectations and performs effectively in real-world scenarios.

Moreover, the system analysis phase includes identifying constraints, such as computational resources, network limitations, and the diversity of email formats and languages. Recognizing these constraints early in the development process helps in crafting a realistic and achievable implementation plan.

The importance of this phase cannot be overstated, as it lays the groundwork for all subsequent stages of the project. By carefully analyzing user needs, system objectives, and potential obstacles, the system analysis phase ensures that the spam detection system is designed to address the complexities of modern email communication while delivering a secure, efficient, and user-friendly experience.

Understanding User Requirements

The effectiveness and success of a spam detection system hinge on its ability to meet the diverse needs of its users. Users interact with email systems in varying capacities, ranging from casual individual users to large-scale organizational email networks. Hence, it is essential for the spam

detection system to be versatile, efficient, and robust, capable of addressing different scenarios while maintaining user satisfaction. To achieve this, user requirements must be identified, analyzed, and categorized into two main types: **functional requirements** and **non-functional requirements**. Each category plays a distinct role in shaping the development and performance of the system.

Functional Requirements

Functional requirements describe the specific tasks and functions the system must perform to fulfill its purpose. These requirements are user-centric and define the "what" of the system's operations. For a spam detection system, functional requirements ensure the system actively addresses the issues posed by spam emails while providing tools to customize, monitor, and improve user interaction. Key functional requirements include:

1. **Spam Detection**:
    - The primary function of the system is to accurately identify and filter spam emails from legitimate emails.
    - It must employ advanced algorithms and techniques to adapt to evolving spam tactics such as obfuscation, phishing, or malicious attachments.
2. **Real-Time Processing**:
    - To maintain seamless email communication, the system must analyze and classify incoming emails in real-time without noticeable delays.
    - Prompt identification ensures that spam emails do not linger in the inbox and legitimate emails are delivered immediately.
3. **Customizable Filtering**:
    - Users have unique preferences for what constitutes spam. The system should allow customization, enabling users to:
        - Block specific email addresses or domains.
        - Create rules based on keywords or phrases.
        - Adjust spam detection sensitivity according to individual needs.
4. **False Positive Management**:
    - The system should minimize false positives (legitimate emails incorrectly flagged as spam).
    - It should provide tools for users to review and recover legitimate emails from spam folders or quarantine areas.
5. **Quarantine and Recovery**:
    - Suspicious emails should not be deleted outright but quarantined for review.
    - Users must have the ability to access quarantined emails and mark any misclassified messages as legitimate.
6. **Support for Attachments and Links**:
    - Modern spam emails often include harmful attachments or malicious links.
    - The system should scan and analyze these elements to detect malware, phishing attempts, or fraudulent content.
7. **Multilingual Support**:
    - Spam emails can be composed in various languages.

- The system must be capable of processing and accurately classifying emails in multiple languages, ensuring comprehensive coverage for global users.

Non-Functional Requirements

Non-functional requirements define the qualities and constraints of the system, focusing on how it performs its tasks rather than what it does. These requirements ensure the system operates reliably and efficiently under varying conditions. For a spam detection system, non-functional requirements include:

1. **Accuracy**:
   - The system should maintain high precision and recall rates, ensuring it can identify spam emails effectively while minimizing false positives and negatives.
2. **Scalability**:
   - As email traffic grows, the system must scale to handle increasing volumes of emails without a drop in performance.
   - Scalability is particularly important for large organizations and email service providers managing millions of emails daily.
3. **Efficiency**:
   - The system must be capable of processing emails quickly, ideally within milliseconds, to ensure real-time classification and uninterrupted email communication.
4. **Security**:
   - The system should be robust against adversarial attacks, such as attempts to bypass spam filters or exploit vulnerabilities in detection algorithms.
   - It must also protect sensitive user data during email processing.
5. **Integration**:
   - The spam detection system should integrate seamlessly with existing email platforms (e.g., Gmail, Outlook, Yahoo Mail) and be compatible with various devices, including desktops, smartphones, and tablets.
6. **User Experience**:
   - The system should offer an intuitive interface for users to manage spam settings, review quarantined emails, and monitor detection performance.
   - It should also provide clear and actionable feedback, such as why an email was flagged as spam.
7. **Adaptability**:
   - The system must be capable of learning and adapting to new spam patterns and techniques. This requires the incorporation of machine learning models or rule-based updates that can evolve with changing spam tactics.

Understanding user requirements is fundamental to developing a spam detection system that is both effective and user-friendly. By addressing functional requirements, the system ensures it performs the necessary tasks of spam detection and management. By meeting non-functional requirements, the system guarantees reliability, efficiency, and scalability across diverse usage scenarios. Together, these requirements form the blueprint for a spam detection system that delivers a seamless, secure, and efficient email experience.

Stakeholder Analysis

A robust spam detection system must address the diverse needs of its stakeholders, who play a crucial role in defining the system's features, performance, and success. Each stakeholder group has unique priorities, expectations, and challenges, making it essential to design the system with their requirements in mind. Below is an expanded analysis of the key stakeholders:

1. End Users

End users, including individuals and small groups, are the primary beneficiaries of the spam detection system. Their needs revolve around convenience, reliability, and a seamless email experience.

Key Requirements:

- **Accurate Spam Filtering:** Users expect the system to accurately differentiate between spam and legitimate emails to minimize disruptions.
- **Customization Options:** The ability to set personalized spam filters, block specific senders, and adjust sensitivity levels is highly valued.
- **Ease of Use:** Users prefer a simple interface that allows them to manage spam settings without requiring technical expertise.
- **Quick Recovery of Legitimate Emails:** Users need an accessible way to review and recover emails incorrectly classified as spam to ensure no critical communications are lost.

Challenges for End Users:

- Annoyance caused by persistent spam that clutters inboxes.
- Risks of falling victim to phishing, malware, or fraudulent schemes due to undetected spam.
- Frustration from false positives (legitimate emails flagged as spam) affecting productivity and communication.

2. Organizations

Businesses and enterprises depend on email for internal and external communication, often managing high volumes of email traffic. The impact of spam on organizations extends beyond inconvenience, posing risks to operational efficiency and cybersecurity.

Key Requirements:

- **Enterprise-Grade Protection:** Systems should provide robust spam detection capable of protecting employees and sensitive corporate data.
- **Productivity Maintenance:** Spam-free inboxes ensure employees can focus on their tasks without distractions.
- **Data Security:** Effective filtering of phishing attempts and malware-laden emails is critical to prevent data breaches and financial losses.

- **Integration with Corporate Email Systems:** Seamless deployment across the organization's email platforms is essential for consistent protection.

Challenges for Organizations:

- Potential loss of sensitive information or financial damage due to phishing attacks.
- Disruption of workflows caused by spam and false positives.
- The need to manage spam detection at scale for hundreds or thousands of users simultaneously.

3. Email Service Providers

Platforms like Gmail, Yahoo, and Outlook are at the forefront of email management, handling billions of emails daily. Their reputation and user satisfaction depend significantly on the effectiveness of their spam detection systems.

Key Requirements:

- **Scalability:** The system must process massive volumes of emails efficiently without performance degradation.
- **User Trust:** Reliable spam filtering helps maintain the trust and loyalty of users.
- **Reduced Server Load:** Filtering spam at the source reduces the burden on email servers, improving overall system performance.
- **Reputation Protection:** Providers must ensure their platforms are not associated with undetected spam or security risks.

Challenges for Email Service Providers:

- Continuously adapting to new spam techniques without compromising system performance.
- Balancing strict spam detection with user complaints about false positives.
- Maintaining high accuracy in multilingual and global email traffic.

4. System Administrators

System administrators are responsible for deploying, maintaining, and updating spam detection systems. Their role ensures the system remains effective against emerging threats and operates smoothly in dynamic environments.

Key Requirements:

- **Monitoring Tools:** Administrators need dashboards to track system performance and spam detection accuracy.
- **Feedback Mechanisms:** Handling user feedback on spam classification to improve the system's effectiveness.

- **Regular Updates:** Administrators require tools to apply updates and patches, ensuring the system evolves alongside spam tactics.
- **Support for Troubleshooting:** Effective documentation and support channels for resolving system issues quickly.

Challenges for System Administrators:

- Staying ahead of sophisticated spam techniques and adversarial attacks.
- Managing user complaints and requests related to spam classification.
- Balancing security and performance during updates and maintenance.

Problem Statement Based on Analysis

Spam emails remain a persistent and growing challenge for individuals, organizations, and email service providers. These unsolicited messages disrupt productivity, pose cybersecurity threats, and reduce system efficiency. Traditional spam detection methods, such as rule-based filters, are proving inadequate against the increasingly sophisticated techniques employed by modern spammers. Key issues include:

1. **Ineffective Classification:** Traditional methods struggle to distinguish between spam and legitimate emails, leading to high rates of false positives (incorrectly flagged legitimate emails) and false negatives (undetected spam).
2. **Adaptation to New Tactics:** Spammers constantly evolve their methods, using obfuscation, phishing techniques, and adversarial attacks that bypass outdated filters.
3. **Scalability Challenges:** As email volumes grow exponentially, traditional systems face difficulties processing traffic efficiently without delays.
4. **Integration Limitations:** Legacy systems often fail to integrate seamlessly with modern email platforms, resulting in user dissatisfaction.

Conclusion of Analysis

This analysis highlights the critical need for a next-generation spam detection system capable of addressing the functional and non-functional requirements of diverse stakeholders. By leveraging advanced technologies and prioritizing accuracy, scalability, and user experience, the proposed system aims to transform email communication into a more secure, efficient, and productive medium for individuals and organizations alike.

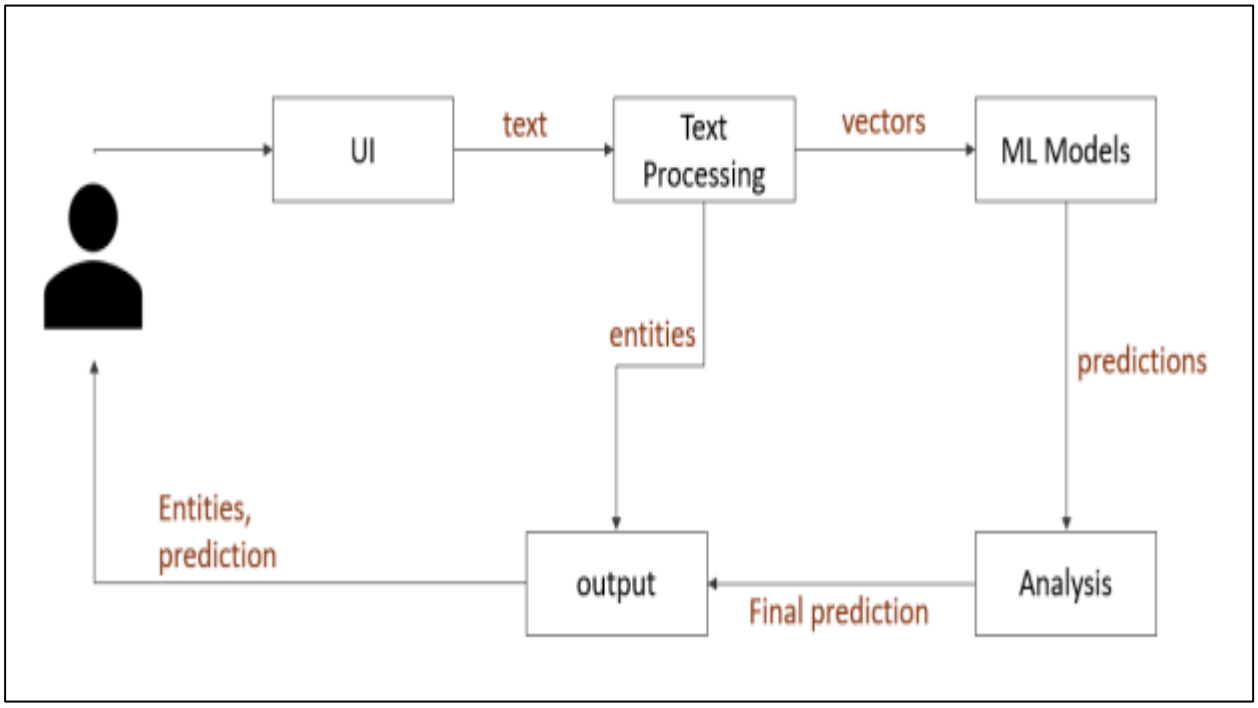
System Design for Spam Detection System

The **system design phase** is a crucial step in the development of a spam detection system. It involves creating a detailed blueprint that defines how the system will function, addressing both functional and non-functional requirements identified during the analysis phase. This blueprint outlines the system architecture, its components, the flow of data, and how users interact with the system.

Goals of the System Design:

- **Blueprint Creation:** Define the overall structure of the system, ensuring each component fulfills its intended purpose.
- **Efficiency and Scalability:** Design a solution capable of handling large volumes of emails without compromising performance.
- **User-Friendliness:** Ensure the system is intuitive and easy to use for both technical and non-technical users.
- **Accuracy and Adaptability:** Prioritize accurate spam classification and design the system to adapt to evolving spam patterns.

System Architecture:



This diagram can be interpreted as the architecture of an **email spam detection system**, which classifies incoming emails as either *spam* or *not spam*. Below is an expanded step-by-step explanation of how each component in the architecture functions in the context of spam detection:

1. User Interface (UI):

- **Purpose:**
  The UI serves as the entry point where users interact with the spam detection system. Users might manually submit an email for classification or link their email accounts for automated processing.
- **Responsibilities:**
  - Accept user inputs, such as email content (subject, body, attachments, and metadata like sender information).
  - Allow users to review the classification results.
  - Enable users to provide feedback (e.g., marking an email as spam or not spam if incorrectly classified).
- **Example Use Case:**
  - A web-based dashboard where users paste an email or upload an `.eml` file for spam analysis.
  - An integrated email client feature that flags incoming emails as spam directly in the inbox.

2. Text Processing Module:

- **Purpose:**
  This module processes the raw input email data and extracts meaningful features that can be used by the spam detection model. It transforms unstructured text into structured numerical representations.
- **Responsibilities:**
  - **Data Cleaning:**
    - Remove unnecessary elements such as HTML tags, special characters, or excessive whitespace.
    - Normalize the text by converting it to lowercase.
  - **Tokenization:**
    Split the email text into smaller components (e.g., words, phrases, or tokens).
  - **Feature Extraction:**
    Extract relevant features for spam detection, including:
    - **Content-based features:** Frequencies of specific keywords (e.g., "win," "offer," "free"), unusual formatting (e.g., excessive capitalization or punctuation), or presence of links and attachments.
    - **Metadata-based features:** Sender email domain, IP address, or timestamps.
    - **Statistical features:** Word frequencies, email length, or character distribution.
    - Techniques like TF-IDF, Bag of Words, or embeddings (e.g., Word2Vec, GloVe, BERT) can be used to convert text into numeric vectors.
  - **Entity Extraction:**
    Identify suspicious entities, such as:
    - Blacklisted domains or IP addresses.
    - Common spam patterns (e.g., "Congratulations, you've won!").

- **Output:**
  - **Vectors:** Structured numerical data representing the email content.
  - **Entities:** Relevant extracted information, such as flagged keywords, domains, or links.

3. Machine Learning Models (ML Models):

- **Purpose:**
  This is the core decision-making component of the system, responsible for classifying emails as spam or not spam based on the processed data.
- **Responsibilities:**
  - Use the vectorized data from the **Text Processing module** as input.
  - Apply a trained machine learning model to analyze the features and make predictions.
- **Types of Algorithms:**
  - **Traditional ML Models:**
    - Logistic Regression
    - Naive Bayes (commonly used for spam detection due to its efficiency with text data)
    - Support Vector Machines (SVM)
  - **Advanced Models:**
    - Deep learning approaches like LSTMs (for sequential text analysis) or transformers (e.g., BERT) for complex understanding of email content.
- **Training:**
  - The model is trained on a labeled dataset of emails categorized as spam or not spam.
  - The model learns patterns, such as keywords, metadata indicators, or formatting styles, that differentiate spam from legitimate emails.
- **Output:**
  - **Predictions:** A classification label (e.g., "spam" or "not spam").
  - **Confidence Scores:** A numerical value indicating the likelihood of the classification (e.g., 90% probability of being spam).

4. Analysis Module:

- **Purpose:**
  Refines the predictions from the machine learning model and ensures the results are actionable and interpretable.
- **Responsibilities:**
  - Validate the prediction made by the ML model.
  - Apply thresholds or additional rules for classification:
    - Example: Emails with a confidence score above 85% are classified as spam.
  - Combine outputs from multiple models if ensemble methods are used (e.g., averaging predictions from different models).
  - Incorporate external checks, such as:
    - Comparing sender domains or links with a blacklist of known spammers.

        - Checking for phishing patterns in URLs.
- **Output:**
    - **Final Prediction:** A refined and accurate classification (spam or not spam) with supporting details for further processing

5. Output Module:

- **Purpose:**
  Displays the results of the spam detection process to the user in an understandable and actionable format.
- **Responsibilities:**
    - Present the classification result (e.g., "This email is spam").
    - Provide additional insights, such as:
        - Extracted suspicious keywords or flagged entities.
        - The confidence score or probability of the email being spam.
        - Recommendations (e.g., "Move this email to the spam folder" or "Flag this sender as suspicious").
    - Allow users to provide feedback, such as marking the email as misclassified.
- **Example Output:**
    - "This email is classified as spam with a 92% confidence score."
    - "Keywords detected: 'lottery,' 'free,' 'urgent.'"
    - "Sender domain flagged as suspicious: example-spam.com."

Feedback Loop:

- **Purpose:**
  The feedback loop allows users to correct the system when an email is misclassified, enabling continuous improvement of the model.
- **How It Works:**
    - Users can mark an incorrectly classified email as spam or not spam.
    - The system logs this feedback to update and retrain the machine learning models.
- **Benefit:**
  This mechanism ensures the system evolves and adapts to new spam patterns, maintaining high accuracy over time.

Example of System Workflow:

1. **Input through UI:**
   A user submits an email via the interface for spam detection.
2. **Processing the Email:**
   The email text and metadata are cleaned, tokenized, and transformed into numerical features by the **Text Processing module**.
3. **Prediction by ML Models:**
   The features are passed to the machine learning model, which predicts the email as spam with a confidence score (e.g., 90%).

4. **Analysis:**
   The prediction is refined by checking against blacklists and applying rules. The email is confirmed to be spam.
5. **Output to User:**
   The system displays the result, providing additional insights like flagged keywords, suspicious links, or sender details.

Advantages of This Architecture for Spam Detection

The architecture of the spam detection system is carefully designed to ensure reliability, efficiency, and adaptability. Below is an expanded explanation of the key advantages:

1. Modularity

- **Definition:**
  The system is built with clearly defined components, such as the UI, Text Processing, ML Models, Analysis, and Output, each handling specific tasks.
- **Advantages of Modularity:**
  - **Ease of Maintenance:**
    Individual components can be updated or debugged without impacting the rest of the system. For instance, if a new feature extraction technique becomes available, it can be incorporated into the Text Processing module without altering the ML models or UI.
  - **Flexibility:**
    Allows for seamless integration of new technologies. For example, a newer, more accurate machine learning model can replace the current one without modifying other modules.
  - **Testing and Debugging:**
    Since components operate independently, errors can be isolated and resolved more efficiently.
  - **Customizability:**
    Components can be customized for specific use cases. For instance, the Analysis module can be adapted to focus more on phishing detection in financial institutions.

2. Scalability

- **Definition:**
  The architecture is designed to efficiently handle a growing number of emails and users without compromising performance.
- **Advantages of Scalability:**
  - **Large-Scale Email Processing:**
    By leveraging distributed systems and parallel processing, the system can process thousands (or even millions) of emails simultaneously.

- **Cloud Integration:**
  The system can utilize cloud-based infrastructure to dynamically allocate resources based on demand, ensuring smooth operation even during peak loads.
- **Efficient Resource Utilization:**
  The use of optimized machine learning models ensures that computational resources are used effectively, enabling faster processing.
- **Adaptation to Growth:**
  As the user base grows or spam detection requirements evolve, the system can scale horizontally (by adding more servers) or vertically (by upgrading hardware or software).

3. Accuracy

- **Definition:**
  The combination of machine learning predictions and rule-based analysis enhances the system's ability to correctly classify emails as spam or not spam.
- **Advantages of High Accuracy:**
  - **Reduced False Positives and Negatives:**
    By leveraging advanced machine learning algorithms alongside heuristic rules, the system minimizes the chances of marking legitimate emails as spam (false positives) or missing actual spam (false negatives).
  - **Hybrid Approach:**
    Machine learning excels at identifying complex patterns, while rule-based methods can address specific spam behaviors (e.g., flagged keywords or blacklisted domains). This synergy leads to better overall performance.
  - **Continuous Learning:**
    The machine learning models are trained on large, diverse datasets, ensuring they generalize well across different types of spam emails, including those with subtle patterns.
  - **Explainability:**
    The Analysis module ensures that the system's decisions are interpretable, providing users with insights into why an email was classified as spam or not spam.

4. Feedback-Driven Improvement

- **Definition:**
  The architecture incorporates a feedback loop, enabling users to correct classification errors and help the system improve over time.
- **Advantages of Feedback Integration:**
  - **Dynamic Learning:**
    Feedback data is used to retrain or fine-tune the machine learning models, ensuring the system adapts to new spam tactics and evolving patterns.
  - **Personalization:**
    Feedback allows the system to learn user-specific preferences. For example, emails from certain senders may be consistently marked as "not spam" by a user, and the system can adjust its behavior accordingly.

- **Improved Accuracy Over Time:**
  As more feedback is collected, the system becomes increasingly robust, identifying nuanced spam behaviors and reducing misclassifications.
- **User Engagement:**
  Users feel empowered to contribute to the system's effectiveness, fostering trust and satisfaction.
- **Rapid Adaptation to Evolving Threats:**
  Spammers frequently change their tactics. By incorporating real-time user feedback, the system stays up-to-date and resilient against emerging threats.

5. Other Key Advantages

A. Efficiency

- The architecture ensures that resources are allocated intelligently. For example, complex ML models are only applied after initial preprocessing steps, saving computational costs for simpler cases.

B. Security

- The system includes multiple layers of validation, such as blacklist checks and phishing detection, enhancing its ability to protect users from malicious emails.

C. Cross-Platform Integration

- The modular design allows the system to be integrated into various platforms, such as email clients, web applications, or mobile apps.

D. Cost-Effectiveness

- By optimizing resource usage and focusing computational efforts where they're needed most, the system minimizes operational costs.

CHAPTER 5: SYSTEM PLANNING (PERTCHART)

A **PERT (Program Evaluation and Review Technique)** chart is a visual project management tool used to plan and organize tasks involved in completing a project. It helps project managers identify the tasks that need to be done, their sequence, and the time required to complete each task. PERT charts are especially useful for large, complex projects where tasks are interdependent, and they help ensure that no critical steps are missed.

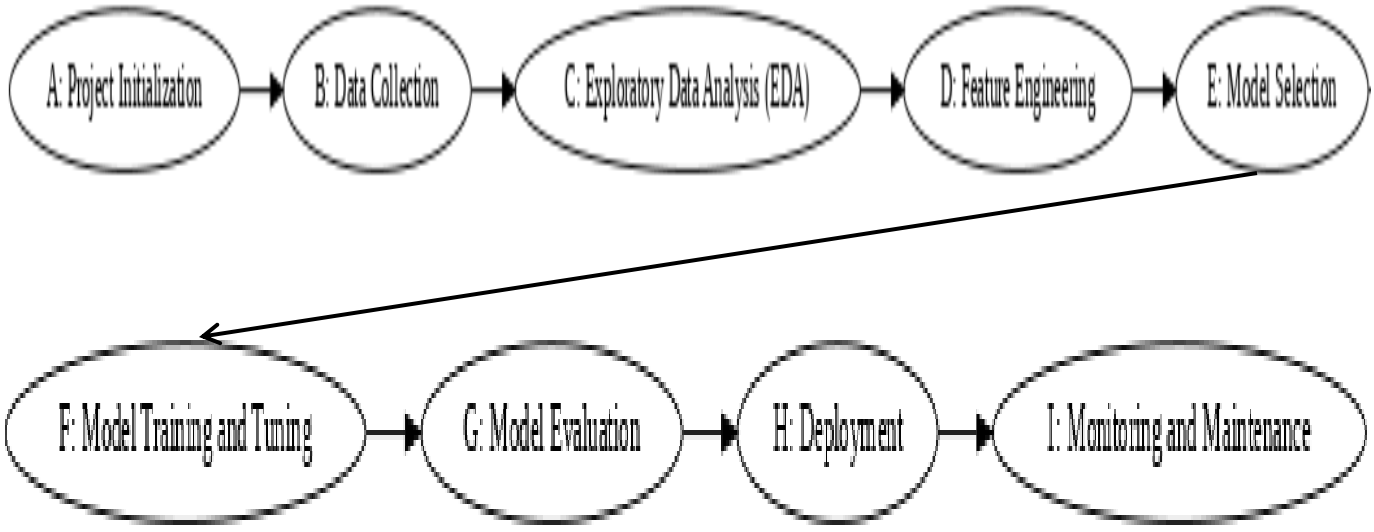
Tasks and Dependencies

1. **Project Initialization**
    - Define goals and metrics.
    - Assign team roles.
    - Estimate resources and timelines.
2. **Data Collection**
    - Gather public datasets.
    - Scrape/clean private email datasets (if allowed).
    - Data cleaning and preprocessing.
3. **Exploratory Data Analysis (EDA)**
    - Analyze dataset properties (class imbalance, missing values, patterns).
    - Visualize data distributions and correlations.
4. **Feature Engineering**
    - Tokenization, stopword removal, stemming/lemmatization.
    - Generate features (e.g., word frequency, n-grams, TF-IDF).
    - Consider metadata (e.g., sender domain, links, attachments).
5. **Model Selection**
    - Choose algorithms: Logistic Regression, Naïve Bayes, SVM, Random Forest, Neural Networks.
    - Compare performance using baseline models.
6. **Model Training and Tuning**
    - Split data into training, validation, and test sets.
    - Hyperparameter tuning (e.g., grid search, random search).
    - Perform cross-validation.
7. **Model Evaluation**
    - Measure performance: accuracy, precision, recall, F1 score.
    - Analyze false positives/negatives for insights.
8. **Deployment**
    - Integrate the model with an email system.
    - Deploy on cloud or on-premise servers.
9. **Monitoring and Maintenance**
    - Track metrics for real-world performance.
    - Update the model with new data periodically.

PERT Chart

Here is a tabular version of the tasks with dependencies for creating a PERT chart diagram:

Task ID	Task Name	Duration	Dependencies
A	Project Initialization	1 week	None
B	Data Collection	2 weeks	A
C	EDA	1 week	B
D	Feature Engineering	2 weeks	C
E	Model Selection	1 week	D
F	Model Training and Tuning	3 weeks	E
G	Model Evaluation	1 week	F
H	Deployment	2 weeks	G
I	Monitoring and Maintenance	Ongoing	H



CHAPTER 6: METHODOLOGY ADOPTED; SYSTEM IMPLEMENTATION & DETAILS OF HARDWARE & SOFTWARE USED SYSTEM MAINTENANCE &EVALUATION

Methodology adopted

Spam email resembles some other sort of PC information. As a representation of a location, its digital components are brought together to create a document or information item with design and presence. The suggested method for spam discovery uses several AI computations. The state technique is used to apply AI models, and after almost breaking down the impacts of the models, the best and most sophisticated model for spam discovery is selected. There are many existing procedures that attempt to forestall or restrict the extension of enormous measures of spam or spontaneous messages. The procedures accessible generally spin around the utilisation of spam channels. To determine if an email message is spam or not, spam channels or spam location general processes review different parts of the email message. Spam identification techniques might be named in light of several email message components. Provides processes for finding spam by its composition and methods for finding spam by its content.

As a rule, the greater part strategies applied to the issue of spam location are compelling, yet they assume a significant part in limiting spam content-based separating. Its positive outcome constrained spammers to routinely change their techniques, conduct and stunt their messages to keep away from these sorts of channels. Spam discovery strategy is - Beginning Based Method: Beginning or address based channels are techniques which consider using network information to perceive whether or not an email message is spam. The email address and the IP address are the primary bits of association information used. There are very few head groupings of starting Based channels like boycotts.

Content Based Spam Discovery Procedures: Content-put together channels are based with respect to inspecting the substance of messages. These substance put together channels are based with respect to physically made rules, likewise called heuristic channels, or these channels are picked up utilising AI calculations. These channels attempt to decipher the text regarding its substance and pursue choices in light of it spread among Web clients, from individual clients on their PCs to enormous business ones sewing. The outcome of content channels to identify spam is perfect to such an extent that spammers have accomplished an ever increasing number of refined assaults that intend to sidestep them and arrive at clients' post boxes.

There are different famous substance based channels, for example, Supervised machine learning, Bayesian Classifier, Support Vector Machines (SVM) and Artificial Neural Network (ANN).

● Supervised Machine Learning: Rule-based channels utilise a bunch of rules for the words remembered for the entire message to check whether the message is spam or not. In this methodology, an examination is made between each email message and a bunch of rules to decide if a message is spam or ham. The standard set contains rules with various loads alloted to each

standard. Toward the start, each email message has a score of nothing. Then, at that point, the email is investigated for the presence of any standard, if any. On the off chance that a standard is found in the message, the weight rules are added to the last email score. Toward the end, in the event that the last score is found to surpass some edge esteem, the email is proclaimed as spam. The impediment of the standard based spam location procedure is that a bunch of rules is exceptionally huge and static, which causes lower execution. Spammers can undoubtedly sidestep these channels with a straightforward word disarray, for instance "Deal" could be changed to S*A*L*E, bypassing the channels. The rigidity of the standard based approach is another significant hindrance. A standard based spam channel isn't wise since there is no self-learning capacity accessible in the channel.

● Bayesian Channels: Bayesian channels are the most developed type of content-based sifting, these channels utilise the laws of likelihood to figure out which messages are real and which are spam. Bayesian channels are too notable AI approaches [19]. To distinguish each message as spam or real, at first, the end client must "train" the Bayesian channel physically to obstruct spam successfully. At long last, the channel takes words and expressions tracked down in genuine messages and adds them to the rundown; that also utilises a similar strategy with words tracked down in spam. Conclude which messages will be named spam messages, the substance of the email is examined with a Bayesian channel and afterward the message is contrasted with its two word records to compute the likelihood that a message is spam. For instance, if "free" seems multiple times in the spam list, yet just multiple times in the ham (real) messages, then there is a 95% opportunity that an approaching email containing "free" is spam or spam messages. Since the Bayesian channel is continually fabricating its statement list in light of messages that a gets, hypothetically turning out to be more successful the more it is utilised. In any case, since the Bayesian channel technique requires preparation before it functions admirably, we will require persistence and you'll most likely need to physically erase a couple of spam messages, basically the initial time.

● Support Vector Machines: Support Vector Machines (SVM) have had outcome in being utilised as text classifiers reports. SVM has prodded significant examination into its utilisation in spam separating. SVMs are the centre strategies, the fundamental thought of which is to embed information checking text reports into a vector space where calculation and straight polynomial maths can be performed. SVM attempts to make a direct division between two classes in v vector space. The separating line characterises the limit on the left of which all articles are PINK and to the right of which all items are BLUE. Any new item (white circle) tumbling to one side is stamped, for example named BLUE (or delegated PINK if it could deceive the left of the isolating line).

SYSTEM IMPLEMENTATION

Analytical System Development

Computers may now learn without being explicitly customised thanks to the research of machine learning. The most amazing innovation that has ever been discussed is probably machine learning. As implied by the name, it grants the machine the ability to learn, which makes it more like people.

Today, ML is efficiently used, possibly in many unexpected places. Machine learning makes it simpler to process large amounts of data. Although it typically provides faster and more accurate findings to identify dangerous content, it does not cost more money or time to train its models for a high degree of performance. The ability to handle massive volumes of data can be improved by combining machine learning, AI, and cognitive computing. There are various ways to illustrate machine learning. supervising machine learning Supervised machine learning techniques are one class of machine learning models that require labelled data.

The expanding subject of information science includes machine learning significantly. Calculations are performed using quantifiable procedures to make characterizations or forecasts and to highlight significant experiences in information mining initiatives. Therefore, internal apps and organisations use this information to inform decision-making, ideally changing important development metrics. Information researchers will become more in-demand as massive data continues to grow and expand. They will be expected to assist in identifying the most important business questions and providing the data necessary to address them. While computerised reasoning (man-made intelligence) is the expansive study of copying human capacities, AI is a particular subset of simulated intelligence that prepares a machine how to learn. Also deep learning is being applied to find the spam and LSTM model is being recreated and used.

Computational System Development

1. Supervised Machine Learning

As the name suggests, supervised machine learning requires administration. It suggests that we train the machines using the "marked" dataset throughout the supervised machine learning process, and based on the configuration, the computer estimates the outcome. According to the marked information in this instance, some of the data sources are now planned to the outcome. What's more, we can say that we ask the machine to predict the outcome using the test dataset after feeding it training data, comparing results, and then asking it to do so. We should figure out managed learning with a model. Assume we have an information dataset of felines and canine pictures. In this way, first, We will provide the computer with the information it needs to understand the images, such as the canine and feline tail's size and shape, the state of the eyes, variety, level (canines are taller, felines are more modest), and so on.

After finishing preparing, we input the image of a feline and request that the machine distinguish the item and foresee the result. Currently, the machine is fully prepared, so it will carefully examine all of the article's distinguishing features, such as level, shape, variety, eyes, ears, tail,

and so on, and determine that it is a feline. As a result, it will be classified as a feline. In supervised machine learning, this is the process the machine follows to identify the items.

The information variable (x) and the result variable have to be planned as the primary goals of the controlled learning technique (y). Hazard Evaluation, Misrepresentation Discovery, Spam Sifting, etc. are a few real-world examples of managed learning applications. Supervised. Machine Learning can be grouped into two kinds of issues, which are given underneath:

I. Classification

II. Regression

I. *Classification*

Classification calculations are utilised to tackle the grouping issues in which the result variable is absolute, for example, "Yes" or “No”, Day or Night, Red or Blue, and so on. The characterization calculations foresee the classifications that are already in the dataset. A few true instances of order calculations are Spam Location, Email separating, and so on.

Some famous classification calculations are given beneath:

● Random Forest Algorithm

● Decision Tree Algorithm

● Logistic Regression Algorithm

● Support Vector Machine Algorithm

II. *Regression*

To address relapse problems where there is a direct correlation between information and result components, regression methods are used. These are used to predict things that have an ongoing effect, such as market trends, expected climatic changes, and so forth. Problems can be solved using this type of instruction. To differentiate spam messages, we have taken the lead in developing AI models. Supervised learning is an idea where the dataset is parted into two parts:

1) Preparing information

2) Testing information.

Benefits:

● Sciencedirect learning works with the named dataset so we can have a precise thought regarding the classes of articles.

● These calculations are useful in anticipating the result based on related knowledge.

Hindrances:

● These calculations can't address complex assignments.

● It might foresee some unacceptable result assuming the test information is not the same as the preparation information.

● It demands loads of computational investment to prepare the calculation.

● Utilizations of Managed Learning

A few normal utilizations of Managed Learning are given underneath:

● Picture Division: Managed Learning calculations are utilised in picture division. In this cycle, picture characterization is performed on various picture information with pre-characterized marks.

● Clinical Analysis: Directed calculations are additionally utilised in the clinical field for conclusion purposes. It is finished by involving clinical pictures and past marked information with names for illness conditions. With such an interaction, the machine can distinguish sickness for the new patients.

● Extortion Recognition - Regulated Learning order calculations are utilised for distinguishing misrepresentation exchanges, extortion clients, and so on. It is finished by utilising noteworthy information to distinguish the examples that can prompt conceivable misrepresentation.

● Spam identification - In spam location and sifting, arrangement calculations are utilised. These calculations characterise an email as spam or not spam. The spam messages are shipped off the spam envelope.

● Discourse Acknowledgment - Managed learning calculations are additionally utilised in discourse acknowledgment. The calculation is prepared with voice information, and different recognizable pieces of proof should be possible utilising something very similar, for example, voice-enacted passwords, voice orders, and so on.

2. Unsupervised Machine Learning

Unsupervised machine learning differs from managed learning in that it does not call for supervision, as suggested by its name. In other words, in unassisted AI, the computer prepares itself with the unlabeled information and predicts the outcome independently.

Unsupervised machine learning uses input that is neither sorted nor labelled to build models, and they follow that data virtually unsupervised. The basic goal of the solo learning calculation is to

compile or categorise the unsorted dataset according to analogies, examples, and contrasts. Machines are instructed to search the information dataset for the hidden examples. Assume there are a tonne of images of natural products, and we feed them into the AI model as a guide so that we may understand it even more vitally. To find examples and classifications of the articles is the machine's task because the pictures are completely opaque to the model.

Thus, presently the machine will find its examples and contrasts, like variety distinction, shape contrast, and foresee the result when it is tried with the test dataset.

Unsupervised Machine Learning can be additionally arranged into two kinds, which are given underneath:

I. Clustering

II. Association

I. *Clustering*

When we need to identify the intrinsic gatherings from the data, we use the bunching approach. It is a technique for grouping objects together so that the ones that resemble each other the most remain in one group and have little to no similarity to the items in other groups. Putting together a group of clients based on their purchasing habits is an example of a bunching calculation.

A portion of the well known grouping calculations are given beneath:

- K-Means Grouping calculation
- Mean-shift calculation
- DBSCAN[15] Calculation
- Head Part Examination
- Autonomous Part Examination

II. *Association*

Using an individual learning method called association rule learning, one can uncover surprising relationships between many variables within a sizable dataset. The main purpose of this learning calculation is to identify the dependencies between various informational elements and to steer those elements in the right directions so that the maximum advantage may be produced. This calculation is mainly used in market bin analysis, web usage mining, consistent creation, etc. A few well known calculations of Affiliation rule learning are A Priori Calculation, Eclat, FP-development calculation.

Pros:

● These calculations can be utilised for muddled errands contrasted with the administered ones in light of the fact that these calculations work on the unlabeled dataset.

● Solo calculations are ideal for different undertakings as getting the unlabeled dataset is simpler when contrasted with the named dataset.

Cons:

● As the dataset is unnamed and the computations are not built up with the exact outcome in mind previously, the result of a solo calculation may be less accurate.

● Working with unassisted learning is more challenging since it uses a dataset that is unlabeled without any outcome planning.

Utilizations of Unsupervised Learning:

● Network assessment: In report network assessment of text information for academic publications, unsupervised learning is used to discern between literary theft and copyright.

● Suggestion Frameworks: Proposal frameworks generally utilise unaided learning methods for building suggestion applications for various web applications and internet business sites.

● Oddity Recognition: Peculiarity discovery is a famous utilisation of unaided realising, which can distinguish strange pieces of information inside the dataset. Finding deceitful transactions is utilised.

● Solitary Worth Deterioration: Solitary Worth Decay or SVD is utilised to separate specific data from the information base. For instance, extricating data of every client situated at a specific area.

3. Semi-Supervised Learning

Semi-Supervised Learning is a type of machine learning that lies among Directed and Unaided AI. It addresses the moderate ground between Administered (With Marked preparing information) and Unaided learning (with no named preparing information) calculations and utilises the blend of named and unlabeled data sets during the preparation time frame.

The concept of semi-supervised learning is put out to combat the drawbacks of supervised learning and unassisted learning calculations. The principal point of semi-administered learning is to actually utilise every one of the accessible information, as opposed to just marked information like in directed learning. At first, comparable information is bunched alongside an unaided learning calculation, and further, it assists with marking the unlabeled information into named information. It is on the grounds that marked information is similarly more costly than unlabeled

information. We can envision these calculations with a model. Regulated learning is where an understudy is under the management of an educator at home and school. Further, assuming that understudy is self-examining a similar idea with next to no assistance from the teacher, it goes under solo learning. Under semi-directed learning, the understudy needs to amend himself subsequent to dissecting a similar idea under the direction of an educator at school.

Benefits:

- Understanding the algorithm is basic and simple.
- It is profoundly proficient.
- Addressing disadvantages of Directed and Unaided Learning algorithms is utilised.

Weakness:

- Emphasess results may not be steady.
- We can't make a difference between these calculations to organise level information.
- Exactness is low.

4. Reinforcement Learning

Support learning deals with a criticism based process, in which a man-made intelligence specialist (A product part) naturally investigates its encompassing by hitting and trial, making a move, gaining from encounters, and working on its presentation. Specialists get compensated for every great activity and get rebuffed for every horrendous act; thus the objective of supporting learning specialists is to boost the prizes. In support realising, there is no marked information like administered learning, and specialists gain from their encounters as it were.

As a person, the support educational experience is comparable to how a toddler learns new things through encounters in his everyday life. Playing a game in which the climate is the game, a specialist's actions at each step characterise states, and the specialist's goal is to get a high score is an example of support learning in action. Experts receive criticism regarding rewards and discipline.

Because of its approach to working, support learning is utilised in various fields, for example, Game hypothesis, Activity Exploration, Data hypothesis, multi-specialist frameworks.

A support learning issue can be formalised utilising Markov Choice Process(MDP). In MDP, the specialist continually connects with the climate and performs activities; at each activity, the climate answers and creates another state.

Classes of Reinforcement Learning:

● Encouraging feedback Learning: Uplifting feedback learning determines expanding the inclination that the expected way of behaving would happen again by adding something. It improves the strength of the way of behaving of the specialist and decidedly influences it.

● Negative Support Learning: The exact opposite of positive RL is how negative support learning operates. By avoiding the bad situation, it increases the likelihood that the specific behaviour would occur again.

Genuine Use instances of Support Learning:

● Computer games: RL calculations are famous in gaming applications. Acquiring godlike performance is utilised. A few famous games that utilise RL calculations are AlphaGO and AlphaGO Zero.

● Asset The executives: The "Asset The executives with Profound Support Learning" paper told that the best way to involve RL in PC is to consequently learn and plan assets to trust that various positions all together will limit normal work stoppage.

● Mechanical technology: Advanced mechanical applications frequently use RL. In the contemporary and assembly world, robots are used, and help learning only serves to increase their impressiveness. Many companies have a goal of creating intelligent robots using AI innovation.

● Text Mining : Text-mining, one of the extraordinary utilizations of NLP, is currently being executed with the assistance of Support Advancing by the Salesforce organisation.

Pros:

● It helps in tackling complex certifiable issues which are hard to be settled by broad methods.

● The learning model of RL is like the learning of people; subsequently most exact outcomes can be found.

● Helps in accomplishing long haul results.

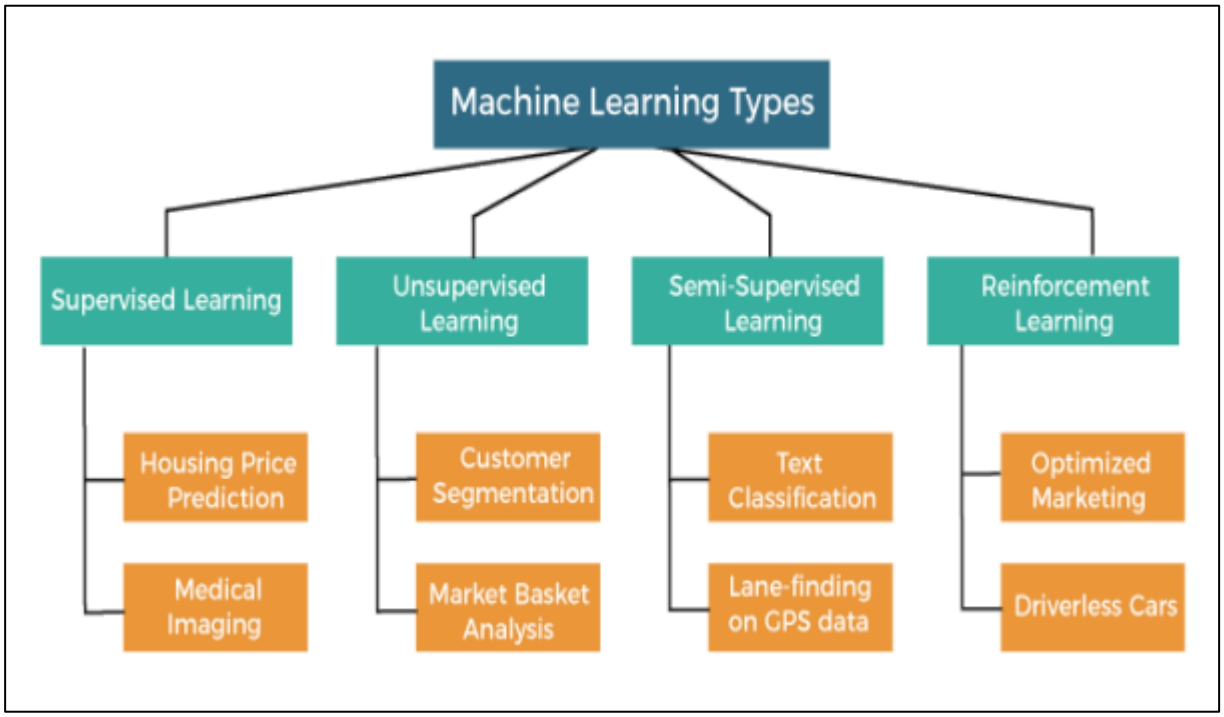
Cons:

● RL calculations are not liked for straightforward issues.

● RL calculations require gigantic information and calculations.

An excess of support learning can prompt an over-burden of states which can debilitate the outcomes. Despite the fact that Semi-regulated learning is the centre ground among directed and unaided learning and works on the information that comprises a couple of names, it for the most part comprises unlabeled information. As names are expensive, yet for corporate purposes, they

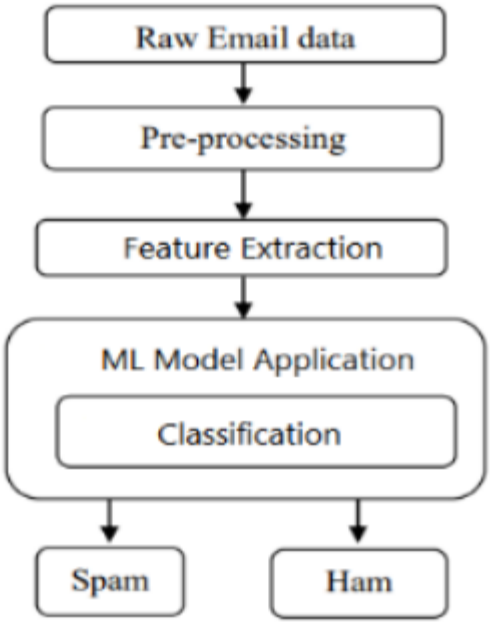
might not have many marks. It is totally not the same as directed and solo advancing as they depend on the presence and nonattendance of marks. Figure below shows the Machine Learning types and its classification.

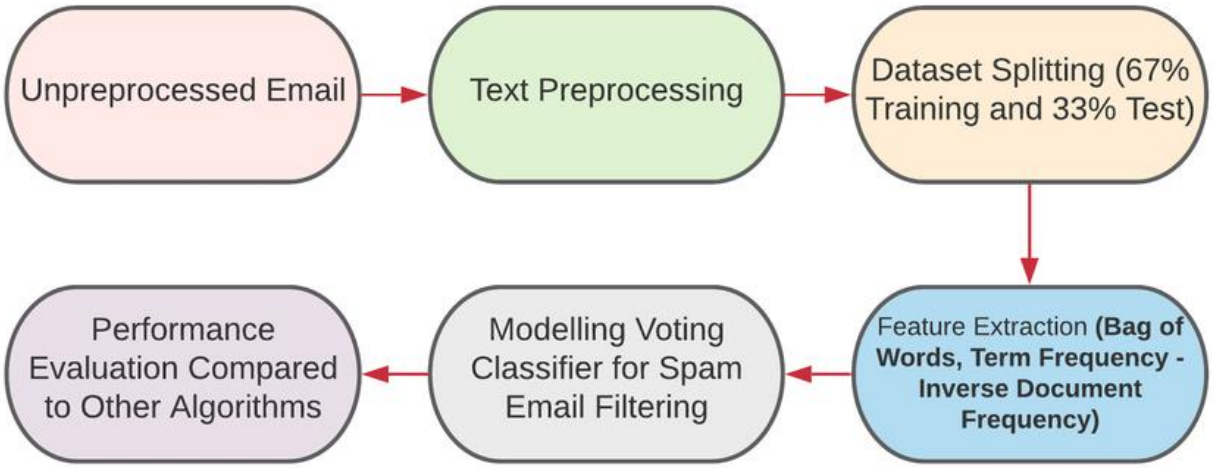


Design and Development

Spam email is only another kind of digital data. Its digital bits are organised into a file, or data object, which has existence and structure since a description is present elsewhere. The recommended strategy for spam identification is illustrated in Fig 4.1 and employs a variety of machine learning techniques. Following the application of machine learning models in accordance with the state approach, the models' outputs are compared.

Under this section the step wise procedure taken up in the project is explained. All the details about the steps taken up from beginning till the end of the project are explained, right from collecting data and making datasets, EDA on the data, Machine Learning models used and all the related steps are discussed. Figure below shows the steps followed





Dataset Description

As the digital environment has grown and technology has evolved, the spreading of spam email messages has become more prevalent, particularly among youth and mischievous people. There are several databases that incorporate various types of spam email messages. We have utilized a preexisting benchmark dataset namely ‘spam collection dataset’ as we are focusing on recognizing spam messages from different emails. This dataset contains 5574 emails with proper labeling. Labeling categorizes all the existing emails into two groups- ‘Spam’ and‘Ham’. Amongst the 5574 emails, 87% are tagged as ’Ham’(0) and the remaining 13% emails are tagged as ’Spam’(1).

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
820	ham	Good afternoon starshine! How's my boytoy? Doe...	NaN	NaN	NaN
2093	spam	Final Chance! Claim ur å£150 worth of discount...	NaN	NaN	NaN
1612	spam	RT-KIng Pro Video Club>> Need help? info@ringt...	NaN	NaN	NaN
559	ham	Aiyo... U always c our ex one... I dunno abt m...	NaN	NaN	NaN
3581	ham	You are right. Meanwhile how's project twins c...	NaN	NaN	NaN

To execute the machine learning models, the following procedures are taken into consideration for the dataset: data cleaning, exploratory data analysis (EDA), data pre-processing, model development, and model evaluation

Data Cleaning

Data cleaning is the process of removing unneeded, redundant, null values, erroneous, or insufficient data from a dataset. Although the results and algorithms may appear to be precise, inaccurate data makes them unreliable. The data cleaning process varies for each dataset. Unwanted observations in the dataset were removed. A few options exist for dealing with missing data. First, throw away observations with missing values, though doing so will also throw away some data. We also have the option of inputting missing numbers based on additional observations.

Steps involved in data cleaning:

● Removing unwanted data: Erasing duplicate, repetitive, or unnecessary data from your dataset is part of removing undesirable data. Copy perceptions are ones that most frequently appear during the information gathering process, while unessential perceptions are those that don't actually match the specific problem you're trying to solve.

Repetitive perceptions drastically alter proficiency since the information is repeated and can either contribute to the correct side or the wrong side, producing unreliable results.

● Any type of information that is of no use to us and can be extracted directly is considered an immaterial perception.

● Fixing Underlying mistakes: Fundamental errors are those that occur during estimating, the transfer of information, or other comparison situations. Grammatical problems in element names, identical quantities with different names, incorrectly labelled classes, such as separate classes that should really be something very similar, and inconsistent capitalization are examples of underlying flaws.

For instance, the model will regard America and America as various classes or values, however they address a similar worth or red, yellow, and red-yellow as various classes or properties, however one class can be remembered for the other two classes. In this way, these are a few primary mistakes that make our model wasteful and give low quality outcomes.

● Overseeing Undesirable anomalies: Anomalies can bring on some issues with particular sorts of models. For instance, direct relapse models are less vigorous to exceptions than choice tree models. By and large, we shouldn't eliminate exceptions until we have a genuine motivation to eliminate them. Once in a while, 39 eliminating them further develops execution, at times not. In this way, one high priority is a valid justification to eliminate the exception, for example, dubious estimations that are probably not going to be important for genuine information.

● Taking care of missing information: Missing information is a beguilingly precarious issue in AI. We can't simply overlook or eliminate the missing perception. They should be dealt with cautiously as they can be a sign of something significant. The two most well known ways of managing missing information are:

● Dropping perceptions with missing qualities: The way that the worth was missing might be educational in itself. Furthermore, in reality, you frequently need to make expectations on new information regardless of whether a portion of the highlights are absent!

● Ascribing the missing qualities from past perceptions: Once more, "missingness" is quite often educational in itself, and you ought to let your calculation know if a worth was absent. Regardless of whether you fabricate a model to credit your qualities, you're not adding any genuine data. You're simply supporting the examples previously given by different highlights.

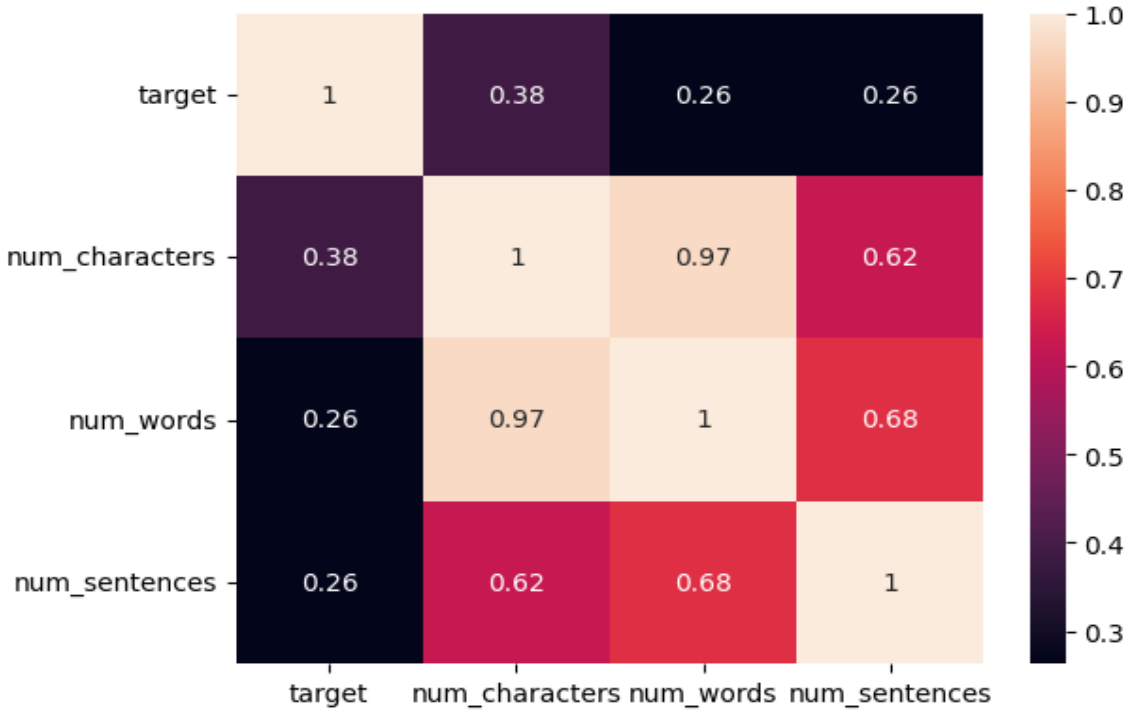
● Missing information resembles missing an interconnecting piece. Assuming you drop it, that resembles imagining the riddle opening isn't there. Assuming that you credit it, that is like attempting to crush in a piece from elsewhere in the riddle. Along these lines, missing information is generally an instructive and a sign of something significant. Furthermore, we should know about our calculation of missing information by hailing it. By utilising this procedure of hailing and filling, you are basically permitting the calculation to appraise the ideal steady for missingness, rather than simply filling it in with the mean.

Fig below shows the data after cleaning

	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

Exploratory Data Study (EDA):

It is the process of characterising data using statistical and visual methods in order to highlight essential components for additional research. Calculations are done to determine the characters, words, and sentences. The fraction of ham and spam is plotted using the character, phrase, and word counts from the dataset. Once the data has been tokenized, stop words, punctuation, and other special characters are removed. On the provided dataset, the stemming procedure which reduces inflected or derived words to their root or base form is applied. Numerous machine learning methods, including logistic regression, decision trees, support vector machines, Naive Bayes, and k-NN, are used to build the model. While the model is trained using 80% of the dataset, only 20% is used to test the applied models. The accuracy of the models is then calculated and contrasted. In Fig. below, the confusion matrix is displayed.



SORTS OF EXPLORATORY DATA ANALYSIS:

● Univariate Non-graphical

● Multivariate Non-graphical

● Univariate graphical

● Multivariate graphical

1. *Univariate Non-graphical*: this is the most straightforward type of information investigation as during this we utilise only one variable to explore the data. The standard objective of univariate non-graphical EDA is to know the fundamental example of appropriation/information and mention observable facts about the populace. Anomaly discovery is furthermore important for the examination. The qualities of populace circulation include:

● Central Tendency: A common or central quality must be the focal tendency or region of conveyance. Measurements with names like mean, middle, and in some circumstances mode—mode being the main normal—are typically beneficial proportions of focal tendency. The middle may be preferred for slanted conveyance or in situations where there is concern over exceptions.

● Spread: Spread indicates how far we should look to get the data values in relation to the centre. There are two useful proportions of spread: the quality deviation and the difference. The difference is found on the variance, which is the mean of the squares of the singular deviations.

● Skewness and kurtosis: In addition, the skewness and kurtosis of the dispersion are two advantageous univariate features. In contrast to a typical distribution, kurtosis and lopsidedness may have a more distinct peakedness proportion, or "skewness."

2. *Multivariate Non-graphical*: Multivariate non-graphical EDA method normally wants to show the association between at least two factors inside the kind of either cross-classification or measurements.

An addition to an arrangement called cross-classification is quite beneficial for absolute information. Cross-classification for two factors resembles creating a two-way table where the column headings represent the amounts of the other two factors and the segment headings represent the amounts of the one variable. After that, all subjects who share the same set of levels are added to the counts. We measure the quantitative factors separately for each level of each unaffected variable and one quantitative variable, then consider the 43 conclusions regarding how much the unaffected factor contributes to each level of the quantitative factor. Contrasting the means is a spur of the moment rendition of ANOVA and looking at medians might be a strong variant of one-way ANOVA.

3. *Univariate graphical*: Non-graphical strategies are quantitative and objective, they do not give the total image of the information; in this manner, graphical techniques are more include a level of emotional examination, likewise are required. Normal kinds of univariate designs are:

● Histogram: The principal fundamental chart is a histogram, which might be a barplot during which each bar addresses the recurrence (count) or extent (count/all out count) of cases for different qualities. Histograms are one of the most straightforward approaches to gain some significant knowledge about your information, including focal inclination, spread, methodology, shape and exceptions rapidly.

● Stem-and-leaf plots: A simple substitute for a histogram might be stem-and-leaf plots. It shows all information values and hence the state of the conveyance.

● Box Plots: Another exceptionally helpful univariate graphical method is the boxplot. Boxplots are great at introducing data about focal propensity and show hearty proportions of area and spread additionally as giving data about balance and anomalies, in spite of the fact that they will be deluding about angles like multimodality. One among the most straightforward purposes of boxplots is inside the kind of next to each other boxplots.

● Quantile-ordinary plots: a definitive univariate graphical EDA method is the most perplexing. It's known as the quantile-typical or QN plot or all the more by and large the quantile or QQ plot. it's wont to perceive how well a particular example follows a particular hypothetical dissemination. It permits recognition of non-ordinariness and finding of skewness and kurtosis

4. *Multivariate graphical*: Multivariate graphical information utilises illustrations to show connections between at least two arrangements of information. The sole one utilised normally might be a gathered barplot with each gathering addressing one degree of 1 of the factors and each bar inside a noisy group addressing how much the contrary variable.

Other normal kinds of multivariate illustrations are:

● Scatterplot: The basic graphical EDA process for two quantitative components is the scatter plot, with one variable on the x-pivot and one on the y-hub and, 44 consequently, the point for each case in your dataset.

● Run outline: It is a line graph with data plotted over a long period of time.

● Heat map: It is a graphical representation of data where values are represented by variation.

● Multivariate outline: It depicts the relationships between the various ingredients and the reaction graphically.

● Bubble graph: an information perception shows different circles (rises) in a two-layered plot.

Basically the continued use of fitting EDA before additional examination of information. Playing out any steps are important to turn out to be more familiar with information, check for clear errors, find out about factor appropriations, and learn about connections between factors. EDA is definitely not a precise science-It is vital!

Data Preprocessing:

Preparing the raw data and making it appropriate for an AI model is known as information preparation. When creating an AI model, it is the first and most important stage. It isn't typically the case that we reveal all facts and design details when developing an AI project. Additionally, keep in mind that before engaging in any action involving the use of information, it must first be cleaned and organised. We thus employ information 45 preparation activities for this. True information is typically riddled with complaints, lacking attributes, and occasionally in an unsuitable arrangement that makes it difficult for AI models to use it in a straightforward manner. Information pretreatment is necessary to clean the data and make it appropriate for an AI model, which also increases the productivity and precision of an ML model.

It includes underneath steps:

1. Getting the dataset

2. Bringing in libraries

3. Bringing in datasets

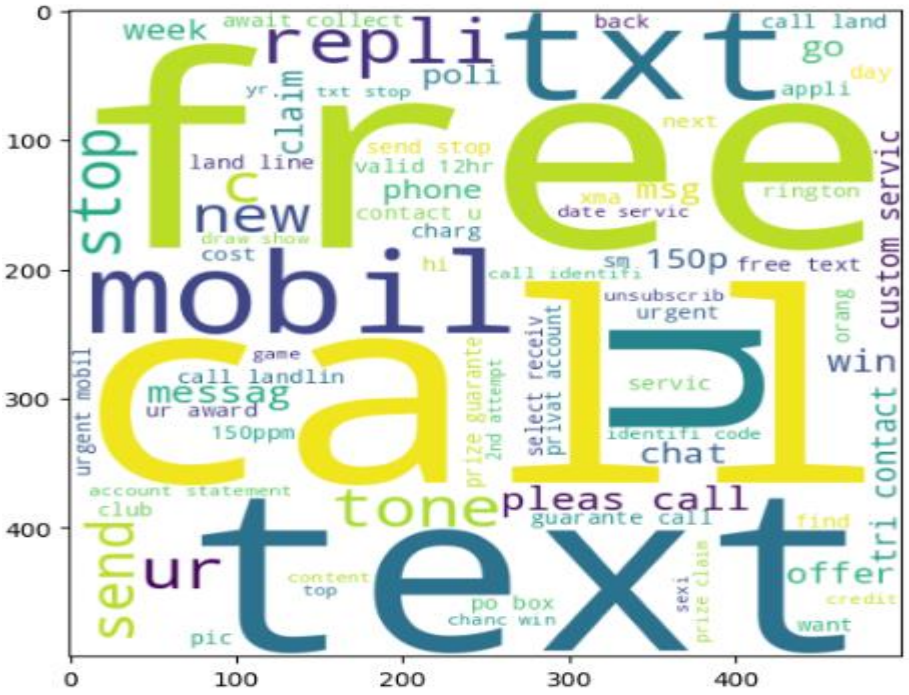
4. Tracking down Missing Information

5. Encoding All out Information

6. Parting dataset into preparing and test set

Feature Extraction

Due to the huge number of words, concepts, and phrases in text categorization, learning from a large amount of data is difficult. As a result, the entire operation becomes computationally expensive. Furthermore, any classification model's accuracy and performance are affected by irrelevant and repetitious features. Hence, to keep the classification process simple, precise, and less repetitive it is better to work with only the significant features extracted from data. We have focused on two feature extraction methods of all the available techniques for our work: "Bag of Words" (BoW) and "Term Frequency-Inverse Document Frequency" (TF-IDF). 'Bag of words' is a simple and flexible way of text representation that describes the frequency with which words appear in a document. We only keep records of word occurrences and don't pay attention to grammatical subtleties or word arrangement. Since any information about the sequence and structure in the document is removed, it is referred to as a "bag" of words. Here the number of appearances of each term in the document is represented as key-value pairs in a count vector format. Whereas, in the TF-IDF method, term frequency is multiplied by the inverse document frequency. Counting the total occurrences of a word in a document can be used to determine term frequency and the IDF is determined by dividing the total document number by the sum of documents in the corpus that contain the term. It's helpful for decreasing the weight of terms that appear most commonly in a group of documents. Finally, multiplying the log of this value with term frequency, we get the final product of TF-IDF.



Bag of Words(spam)

Bag of Words(ham)

MODEL Preparation AND TESTING Stage

The model was constructed with known information and tested with obscure information to predict the accuracy and other execution measures, as was seen through the examination, which also showed the use of controlled learning techniques. K-Overlap cross approval was used to produce reliable results. This approach has certain drawbacks. For instance, it's possible that the test data will contain just spam messages, or that the preparation set will contain the majority of spam messages. This was resolved using defined k-overlap cross approval, which isolates the data while attempting to include a respectable amount of ham and spam in the circulated set. In order to deal with attempts and work on the accuracy of ML models, boundary tuning was ultimately directed with the Scikit-Learn and bio-enlivened calculations. This gives a stage to contrast the Scikit-learn library and the bio-roused calculation.

Hardware and Software Used

The development and deployment of an email spam detection system is a multifaceted process that relies on a combination of hardware and software components. These components are carefully selected to ensure optimal performance, scalability, and user satisfaction. A well-designed spam detection system must handle large volumes of data, process emails in real-time, and provide accurate and reliable results. Furthermore, it should be adaptable to evolving spam techniques and user requirements, necessitating robust hardware and software infrastructure.

Below provides a comprehensive overview of the hardware and software utilized in the creation and deployment of a spam detection system, covering everything from the technical specifications of the hardware to the tools and frameworks that drive its functionality. Understanding the role and selection criteria of each component is critical to building a system that meets the functional and non-functional requirements of its stakeholders.

Purpose of the System

The primary purpose of the email spam detection system is to filter out unwanted, unsolicited, and potentially harmful email messages while allowing legitimate emails to pass through. This involves:

- Handling diverse datasets to train the system for accurate spam detection.
- Running real-time algorithms to classify incoming emails with minimal delays.
- Protecting users and organizations from phishing attacks, malware, and unnecessary email clutter.

To achieve these goals, the system's hardware and software must support extensive data processing, advanced machine learning techniques, and seamless deployment in a variety of environments.

Importance of Hardware and Software Components

Hardware

The hardware forms the backbone of the system, ensuring that it has the computational power to process large volumes of email data efficiently. The hardware requirements vary depending on the system's scale:

- **Development Stage:** Moderate specifications for building and testing the system locally.
- **Deployment Stage:** High-performance servers or cloud-based solutions to handle real-world traffic.

Software

The software stack provides the tools and frameworks necessary to implement, train, evaluate, and deploy the spam detection system. This includes:

- **Programming languages and libraries:** To handle natural language processing (NLP), machine learning, and data manipulation tasks.
- **Development environments:** To streamline coding, testing, and debugging.
- **Deployment tools:** To ensure that the system integrates seamlessly with email platforms and can be scaled efficiently.

Key Objectives for Hardware and Software

1. **Efficiency:** Handle high volumes of email data with minimal processing time.
2. **Scalability:** Adapt to increasing user demands without compromising performance.
3. **Accuracy:** Provide reliable spam detection results with minimal false positives and negatives.
4. **Security:** Ensure robust protection against adversarial attacks and system breaches.
5. **User-Friendly Features:** Allow easy configuration and customization by end-users and administrators.

Role of Hardware

Development Hardware

Development hardware is essential for building and testing the system:

- **Processor:** A modern multi-core CPU, such as Intel Core i5/i7 or AMD Ryzen, ensures quick execution of algorithms and efficient data processing.
- **RAM:** A minimum of 8GB of RAM allows for smooth handling of small to medium-sized datasets. For larger datasets, 16GB or more is recommended.
- **Storage:** An SSD ensures faster data access and reduces the time required to load libraries and datasets.
- **Networking:** A stable internet connection facilitates downloading of datasets, libraries, and cloud-based resources.

Deployment Hardware

Deployment requires robust infrastructure to manage high traffic and multiple user requests:

- **Servers:** Cloud-based servers (AWS, Azure, or Google Cloud) or on-premises servers with enterprise-grade specifications ensure reliability and scalability.
- **Networking:** High-speed internet connectivity ensures real-time email classification without delays.

Role of Software

Operating System

- **Development:** Platforms like Windows, macOS, or Linux (Ubuntu) offer flexibility and compatibility with a wide range of tools.
- **Deployment:** Linux-based servers (e.g., Ubuntu Server, CentOS) provide stability, security, and performance.

Programming Languages

Python is the primary language due to its simplicity and extensive library support. Its frameworks for machine learning and NLP, such as Scikit-learn and NLTK, make it ideal for spam detection systems.

Python Tools

SCIKIT-LEARN: The Python programming language is integrated with the SCIKIT-LEARN (SKLearn) learning environment. There are a ton of directed computations available in the library that will work well for this project. The library provides high-level execution to get ready using "Fit" techniques and "anticipate" from an assessor (Classifier). Additionally, it provides for the cross approval to be performed, including selection, highlight extraction, and boundary tuning.

KERAS: A programming interface called KERAS supports brain organisations. For a quick and easy approach, the programming interface supports further in-depth learning calculations. In order to handle the models concurrently, it provides computer chip and GPU running capabilities. The brain network may learn from and advance through online educational tasks. Their assistant demonstrates how to improve the exhibition using GPU and how to work with RNN calculations and other sophisticated learning calculations.

TensorFlow: Tensorflow is a start to finish ML stage that is created by Google. The engineering allows a client to run the program on different computer processors and it likewise approaches GPUs. The site likewise gives a learning stage to the two novices and specialists. TensorFlow can likewise be consolidated with Keras to perform profound learning tests . 35

PYTHON Stages:

JUPYTER NOTEBOOK: This is an open source device that gives a Python system. This is like 'Spyder' IDE, with the exception that this device allows a client to run the source code by means of an internet browser. Boa constrictor system likewise offers 'Jupyter' to be used by the client through the nearby server. Alongside the work area based stages, other web-based stages that offer extra help are: Google Collaboratory and Kaggle. The two stages are the top ML and DL based that additionally offers TPU (Tensor Handling Unit) alongside Central processor and GPU.

System Maintenance & Evaluation

After developing and deploying an email spam detection system, the focus shifts to maintaining its performance, ensuring reliability, and continuously improving its efficiency. **System Maintenance** and **Evaluation** are critical stages to ensure the system adapts to evolving spam patterns, remains secure, and continues to meet user expectations. These activities involve monitoring the system, addressing issues, updating models, and evaluating overall effectiveness through performance metrics.

System Maintenance

1. Regular Updates

- **Model Updates:** Spam patterns change frequently, requiring retraining of machine learning models with updated datasets to capture new tactics.
- **Rule Updates:** For systems with rule-based components, these rules should be reviewed and updated periodically to reflect emerging spam trends.
- **Software Updates:** Regularly updating libraries, frameworks, and system dependencies ensures compatibility, performance, and security.

2. Performance Monitoring

- Monitor metrics such as detection accuracy, false positives, false negatives, and processing time.
- Use logging mechanisms to track system behavior and identify anomalies in real-time.
- Analyze user feedback to pinpoint recurring issues or areas for improvement.

3. Security Management

- Protect the system against adversarial attacks, such as spammers using obfuscation techniques to bypass detection.
- Employ encryption for data transmission and implement robust authentication mechanisms.
- Regularly patch vulnerabilities in the system to prevent breaches.

4. Backup and Recovery

- Maintain regular backups of the system's configuration, training data, and model parameters to ensure quick recovery in case of failure.
- Implement disaster recovery protocols for seamless restoration after unexpected system crashes.

5. Scalability Adjustments

- Upgrade hardware or allocate additional cloud resources to handle increased email traffic or user growth.

System Evaluation

1. User Feedback

- Collect feedback from users to understand how well the system meets their expectations.
- Identify complaints about misclassified emails (either spam or legitimate).
- Use this feedback to fine-tune model parameters or update filtering rules.

2. Testing Against New Datasets

- Evaluate the system's performance on fresh, unseen datasets to ensure it generalizes well and maintains accuracy against new spam techniques.
- Conduct stress testing to evaluate system performance under high email traffic.

3. Benchmarking

Compare the system's performance with industry standards or alternative spam detection systems. This helps to identify areas of improvement and competitive advantages.

4. System Audits

Conduct regular audits of the system, covering:

- Code quality and compliance with best practices.
- Adherence to security standards and data privacy regulations.
- Effectiveness of implemented algorithms and rules.

Challenges in Maintenance and Evaluation

- **Evolving Spam Techniques:** Spammers constantly adapt their methods, requiring continuous updates to the detection system.
- **False Positives and Negatives:** Striking a balance between strict filtering and allowing legitimate emails through can be challenging.
- **Performance vs. Cost:** High accuracy and scalability may demand additional computational resources, increasing costs.
- **Real-Time Processing:** Ensuring low latency while maintaining accuracy remains a critical challenge.

CHAPTER 7: DETAILED LIFE CYCLE OF THE PROJECT

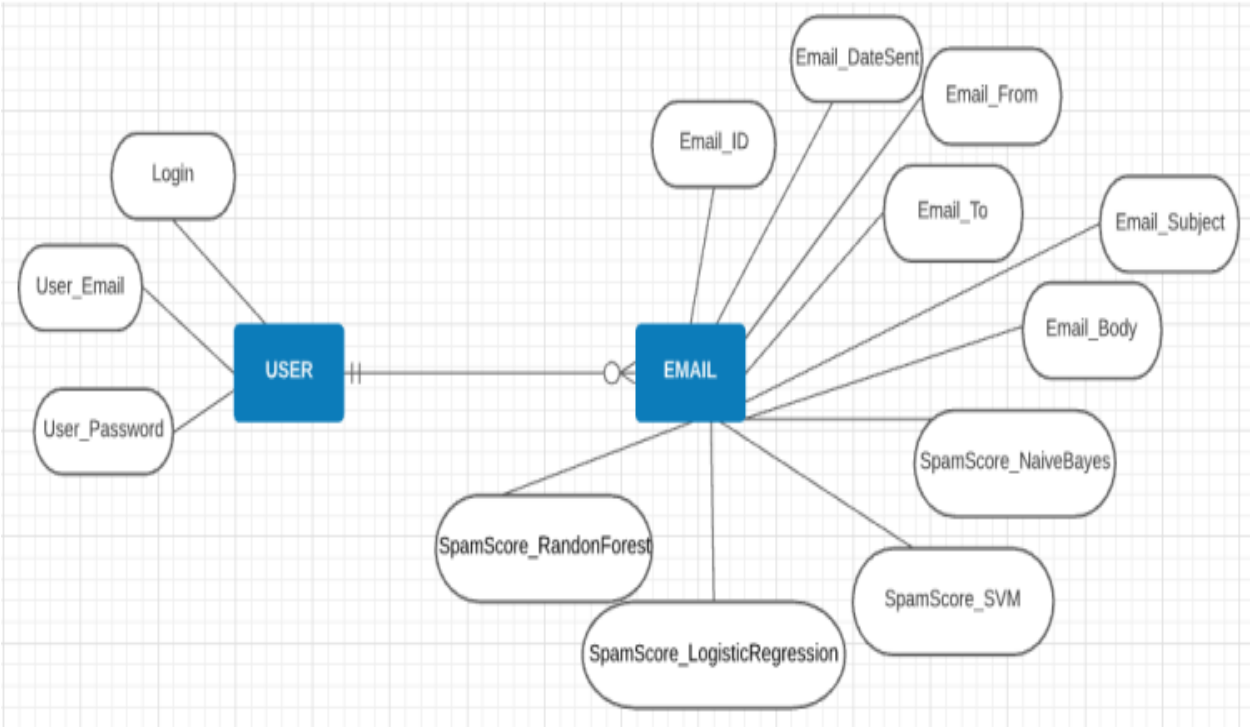
7.1 ERD, DFD

Entity Relationship Diagram

The **ER (Entity-Relationship) Diagram** illustrates the relationship between the two primary entities involved in a typical email spam detection system: **User** and **Email**.

The relationship between the **User** and **Email** entities is central to the spam detection system. A **User** can receive or send many **Emails**, forming a one-to-many relationship. Conversely, each **Email** belongs to exactly one **User**, creating a dependency that ensures emails are correctly associated with their intended recipients or senders.

This diagram is fundamental for understanding how data flows within the system. It helps in designing the database structure that supports functionalities like filtering spam emails, analyzing user-specific patterns to improve spam detection accuracy, and maintaining logs for user activity. Furthermore, the relationship between the entities also plays a role in implementing features like personalized spam filters, where the system learns from a user's interactions (e.g., marking emails as spam or ham) to enhance detection.

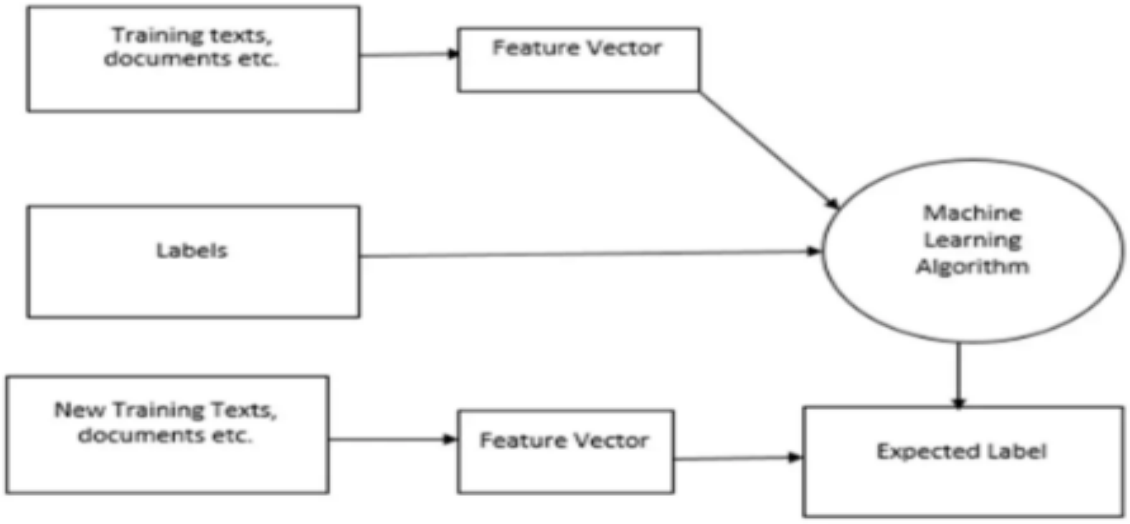


Data Flow Diagram (DFD)

A **Data Flow Diagram (DFD)** is a visual representation used to illustrate how data flows through a system, showing the movement of information between processes, data stores, and external entities. It provides a clear and structured way to analyze and document the flow of data within a business or information system, helping to understand how inputs are transformed into outputs. DFDs are composed of several key components: **processes** (which represent operations or actions that transform data), **data flows** (which show the direction of data movement), **data stores** (which represent repositories where data is stored), and **external entities** (such as users or other systems that interact with the system).

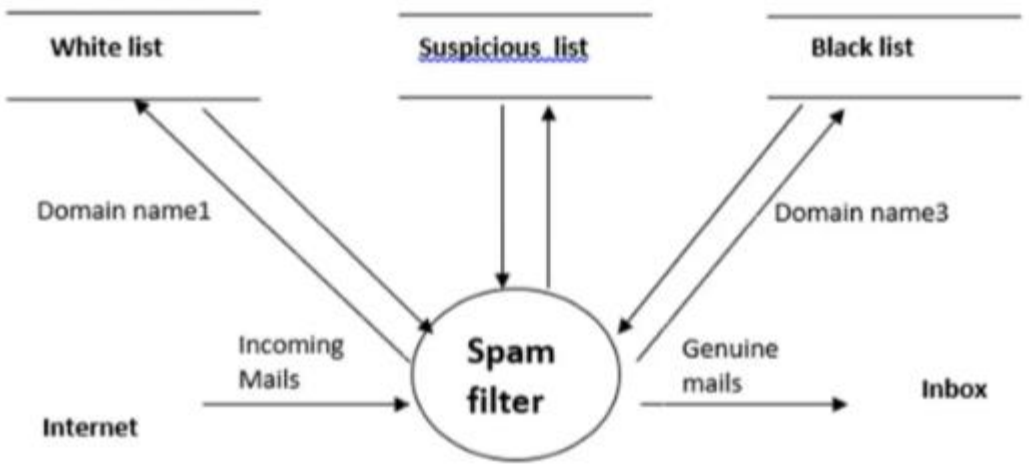
DFDs are typically created at different **levels** to provide varying degrees of detail.

The **Level 0 DFD** (also known as the **Context Diagram**) provides a high-level overview of the entire system, showing the system as a single process and its interactions with external entities.



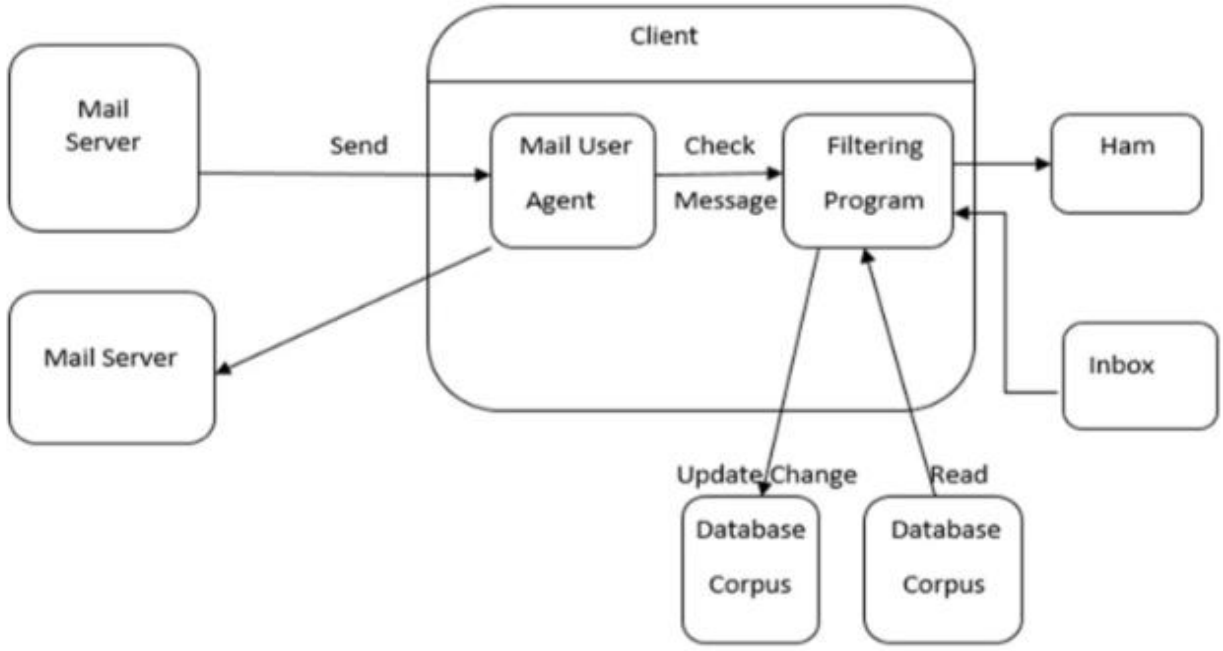
DFD Level 0

The **Level 1 DFD** breaks down the single process into its major sub-processes, providing more detailed insight into the system's functions.



DFD Level 1

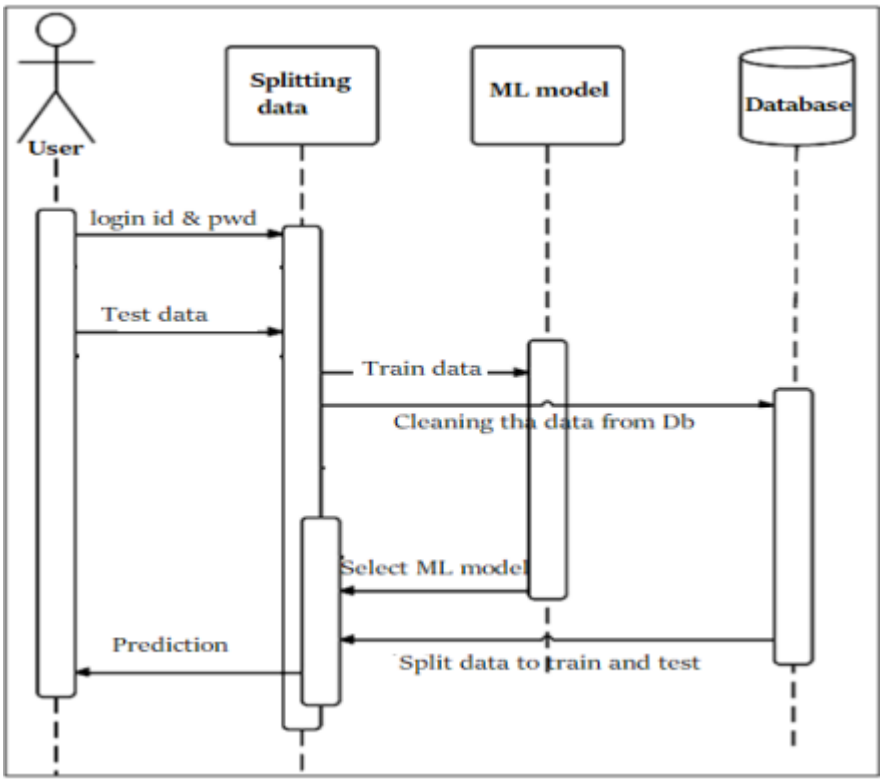
Further levels, such as **Level 2** and beyond, continue to decompose the sub-processes into even finer details, offering a step-by-step breakdown of individual operations.



DFD Level 2

Sequence Diagram

A Sequence Diagram is a type of interaction diagram that models the flow of messages between different objects or entities in a system, arranged in the order in which they occur over time. It provides a detailed view of how objects interact in a particular scenario, emphasizing the sequence of actions or messages exchanged between them. In a sequence diagram, objects or entities are represented as vertical lifelines, and the messages exchanged between them are depicted as horizontal arrows that indicate communication or method calls. The diagram also illustrates the order in which these messages occur, with time progressing from top to bottom. Sequence diagrams are particularly useful for detailing the dynamic behavior of a system, as they show how the system components collaborate to complete a specific process or task, capturing the flow of control in a time-ordered manner. These diagrams typically focus on a single scenario or use case, where each interaction between objects is represented as a message being passed between them. This helps stakeholders and developers understand the detailed sequence of operations, dependencies between objects, and potential timing constraints or performance considerations in the system's behavior. Sequence diagrams are invaluable for both system design and testing, as they provide a clear and structured way to visualize object interactions and ensure that the system's behavior matches the intended functionality.



7.2 Input and Output Screen Design

The developed model, which achieves the best accuracy on most datasets, was used to create a web application that provides an interface for user interaction. In this application, a user enters an email message, and the machine learning model predicts whether the email is spam or not. While I initially considered using a cloud platform like Heroku, for this project, I opted to run the application on a local server instead.

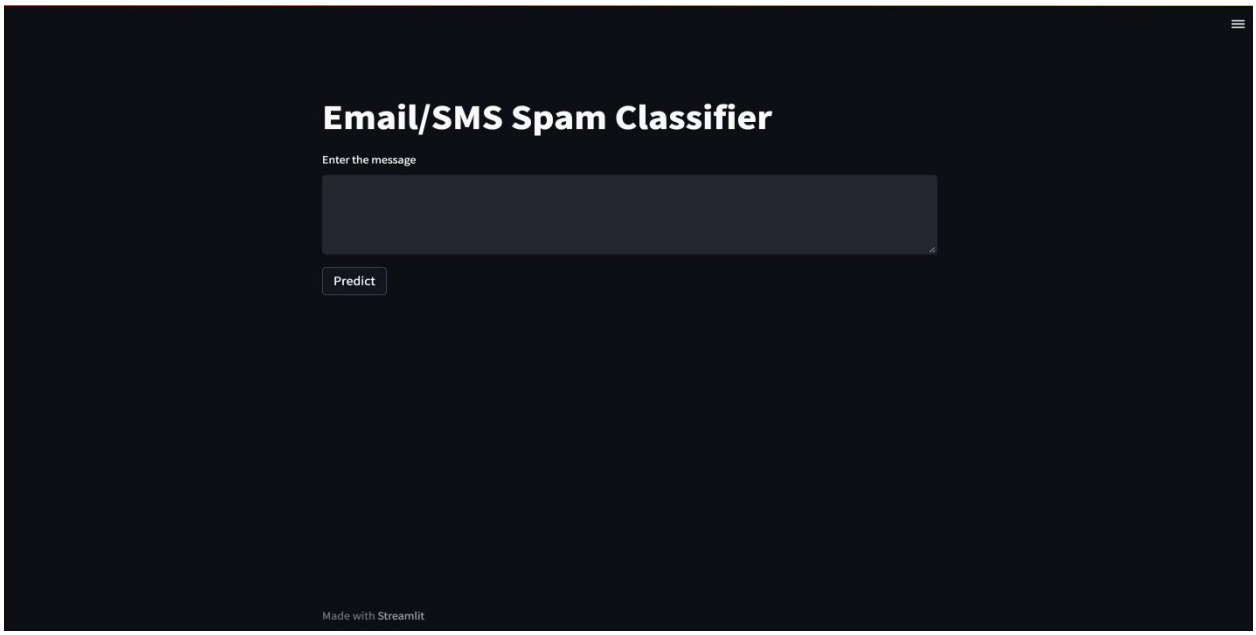
To set up the application, in pycharm I created my own virtual environment. Install all the dependencies required using pip or conda. To build my website I downloaded and setup streamlit. **Streamlit** is an open-source Python library used to create interactive web applications for machine learning and data science projects with minimal effort. It is designed to allow developers and data scientists to quickly build and share custom web applications, especially those that involve data visualization, machine learning models, or other computational tasks.

Streamlit simplifies the process of turning Python scripts into interactive web apps, where users can interact with data, input parameters, and see real-time results, all through an easy-to-use interface. It is known for its simplicity and ease of use, as it does not require any knowledge of web development technologies such as HTML, CSS, or JavaScript.

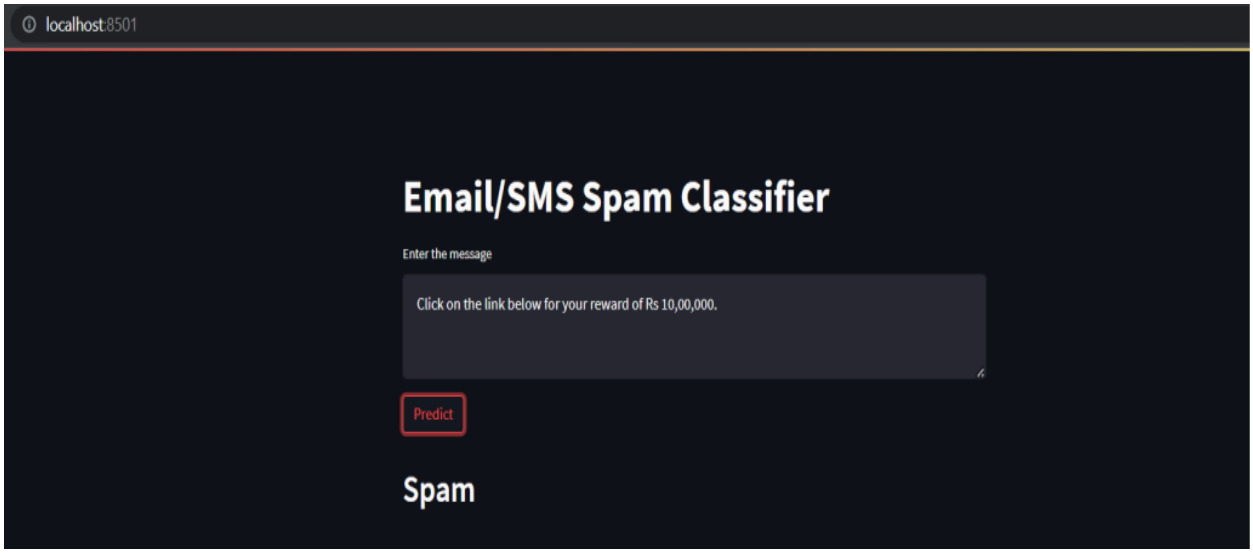
Key features of Streamlit include:

1. **Rapid Development**: You can create and deploy applications with only a few lines of code. Streamlit automatically updates the interface as you change the Python code, making it easy to see live updates without manual refreshing.
2. **Integration with Python Libraries**: Streamlit integrates seamlessly with popular Python libraries like **Pandas**, **Matplotlib**, **Plotly**, **TensorFlow**, **PyTorch**, and **Scikit-learn**, enabling you to build applications that showcase data analysis, machine learning models, and visualizations.
3. **Interactivity**: Streamlit supports interactive elements like sliders, buttons, text input fields, and file upload components, allowing users to customize their input and explore different outcomes or visualizations dynamically.
4. **Deployment**: Once the app is built, it can be easily shared by deploying it to a server, or you can use services like **Streamlit Cloud** (formerly Streamlit Sharing) to host the app online for free or for paid plans with more resources.
5. **Customizable UI**: Though it is simple to use, Streamlit offers customization options to make applications visually appealing and user-friendly.

I tested it thoroughly on my local machine. Once I was satisfied with its performance, I proceeded to run the application on the local server. For database management, I used a local database to store application data, ensuring that everything was functioning as expected. After configuring the environment and making necessary adjustments, I accessed the application through a web browser to verify its functionality. By running the application on a local server, I was able to ensure it worked seamlessly without the need for cloud deployment, allowing me to efficiently test and demonstrate the spam detection model.



Input Screen



Output Screen

7.3 Process Involved

An email spam detection system is designed to identify and filter out unwanted, unsolicited, or malicious email messages (known as spam) while allowing legitimate emails (known as ham) to pass through. The effectiveness of such a system is critical to maintaining an efficient and secure email communication environment. The system typically involves several key processes to achieve accurate spam classification. Let's expand on these processes and explore how they work together to detect and filter spam:

1. Data Collection

- The first crucial step in building a spam detection system is gathering a **large and representative dataset** of emails. The dataset is typically divided into two main categories:
    - **Spam**: Unsolicited emails, often sent for advertising, phishing attempts, or spreading malware.
    - **Ham**: Legitimate emails from trusted sources, such as personal, business, or transactional emails.
- **Labeled Data**: A labeled dataset where each email is manually tagged as "spam" or "ham" is essential for training machine learning models. Public datasets, such as the **SpamAssassin corpus**, **Enron corpus**, or **LingSpam dataset**, are commonly used for this purpose.
- **Feature Identification**: The system must identify key features within each email that could help distinguish between spam and non-spam. These features include the **email subject**, **body content**, **headers**, **sender information**, **attachments**, and **metadata**.

2. Preprocessing

Preprocessing is an essential step to clean and structure raw email data, making it suitable for analysis by machine learning algorithms.

- **Text Cleaning**: Emails often contain noise, such as **HTML tags**, **special characters**, or **irrelevant content** (e.g., email signatures, advertisements, or unnecessary formatting). These need to be removed or simplified for effective analysis.
- **Normalization**: Converting the text to a standard form, such as converting all text to **lowercase**, helps avoid distinguishing between the same word written in different cases (e.g., "Free" vs "free").
- **Tokenization**: This process involves splitting email content into smaller, more manageable units called **tokens** (usually words or phrases). Tokenization allows the system to handle and analyze text more effectively.
- **Stop-word Removal**: Common words like "the," "is," and "on" do not contribute much meaning in spam detection, so they are typically removed to focus on the more important, distinctive words.
- **Stemming/Lemmatization**: These techniques reduce words to their base forms (e.g., "running" becomes "run"), ensuring that different forms of the same word are treated as the same feature.

- **Handling Misspellings and Variations**: Spammers often try to bypass filters by using unusual spelling or symbol replacements (e.g., “fr33” instead of “free”). The system may need to correct or account for such variations.

3. Feature Extraction

Once the data is cleaned and tokenized, the next step is to extract features that will help the model distinguish between spam and ham.

- **Bag of Words (BoW)**: One of the simplest approaches is to represent each email as a vector of word occurrences. The **Bag of Words** model creates a list of all unique words in the corpus and counts the frequency of each word in the email. This approach helps the model analyze the importance of each word in relation to the classification task.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: This technique refines the basic Bag of Words approach by weighing each word's importance based on how often it occurs in the email relative to how rare it is across the entire dataset. Rare words with high frequency in a document are considered more important.
- **N-grams**: To capture context and relationships between words, n-grams (e.g., **bigrams** or **trigrams**) are used. N-grams are sequences of N consecutive words (e.g., "free offer" as a bigram). This is particularly useful for detecting certain patterns or phrases commonly found in spam.
- **Domain-specific Features**: In addition to textual features, other elements such as the **sender’s email domain**, **IP address**, **email attachments**, and **URLs** in the body of the email can provide strong indicators of spam. For instance, email addresses from suspicious domains or messages with executable file attachments may be flagged.

4. Model Training

The next step involves training a machine learning model on the extracted features to distinguish between spam and ham.

- **Supervised Learning**: Spam detection is typically treated as a supervised learning problem, where the model is trained on a labeled dataset (emails that are already classified as spam or ham).
- **Classification Algorithms**:
    - **Naive Bayes**: One of the simplest and most commonly used models for spam detection. The Naive Bayes classifier is based on **Bayes’ Theorem** and makes predictions based on the probability of an email being spam, given the observed features.
    - **Support Vector Machine (SVM)**: This model creates a hyperplane to separate spam emails from non-spam emails by learning the best possible boundary in a high-dimensional feature space.
    - **Logistic Regression**: Another commonly used classification model that estimates the probability of an email being spam or not based on the features.
    - **Random Forests**: An ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting.

    - **Deep Learning**: For large-scale datasets or complex patterns, deep learning models, such as **neural networks**, may be used. These models can capture sophisticated patterns in the data, especially when dealing with a large volume of email content.
- **Training**: The model is trained using the labeled dataset, adjusting its parameters to minimize classification errors (i.e., false positives and false negatives). During this phase, the system learns to recognize the distinguishing characteristics of spam and ham.

5. Model Evaluation

After the model is trained, it is evaluated on a separate test set of emails to measure its performance.

- **Accuracy**: The overall proportion of correct classifications, i.e., the number of spam and non-spam emails correctly identified by the system.
- **Precision**: The percentage of emails predicted as spam that are actually spam. High precision ensures that legitimate emails are not mistakenly marked as spam.
- **Recall**: The percentage of actual spam emails that are correctly classified as spam. High recall ensures that most spam emails are detected, reducing the risk of spam emails slipping through.
- **F1-Score**: The harmonic mean of precision and recall, which provides a balance between both metrics. It's especially useful when dealing with imbalanced datasets (where the number of ham emails is much greater than spam emails).
- **Confusion Matrix**: A confusion matrix is used to visualize the performance of the model by displaying the true positives, false positives, true negatives, and false negatives. This helps identify specific areas for improvement (e.g., if the model is classifying too many legitimate emails as spam).

6. Spam Classification

Once the model is trained and evaluated, it is deployed to classify incoming emails in real-time.

- **Real-time Classification**: As new emails arrive, they are processed by the spam detection system. The system analyzes the content of the email, extracts the relevant features, and classifies the email as either spam or ham.
- **Filtering**: If an email is classified as spam, it is moved to the **spam folder** (or junk folder) to prevent it from cluttering the user's inbox. This filtering process helps ensure that users only see legitimate emails in their inbox.
- **Adaptive Filtering**: Some spam detection systems allow users to mark emails as "spam" or "not spam." This feedback is valuable for further training the model, allowing the system to adapt and improve over time.

7. Post-Processing

- **User Feedback**: In many modern spam detection systems, users can provide feedback by marking legitimate emails that were mistakenly flagged as spam or by moving spam emails

that were not detected. This feedback helps refine the system by correcting misclassifications.

- **Continuous Learning**: Some spam detection systems incorporate feedback loops where the model is periodically retrained using updated datasets, including newly labeled emails. This ensures that the system stays up-to-date and continues to perform well as spam tactics evolve.

By combining these processes — from data collection and preprocessing to model evaluation and post-processing — email spam detection systems become increasingly effective at identifying and filtering out unwanted emails. The goal is to provide users with a seamless and secure email experience while minimizing false positives and ensuring that spam messages are accurately detected.

7.4 Methodology used testing

In testing an **email spam detection system**, various methodologies and techniques are employed to evaluate the system's performance, ensure its reliability, and assess its ability to correctly classify emails as either **spam** or **ham** (legitimate email). These testing methodologies typically focus on how well the model generalizes to unseen data, how robust it is to evolving spam tactics, and how accurately it can distinguish between the two classes. Below are some of the common methodologies used in testing an email spam detection system:

1. Holdout Testing (Train-Test Split)

- **Description**: This is one of the most basic methods for evaluating a spam detection model. In this approach, the entire dataset is divided into two sets:
    - **Training Set**: A portion of the data is used to train the model.
    - **Test Set**: A separate portion of the data is used to evaluate the model's performance.
- **Procedure**:
    1. The dataset is randomly split into two subsets (e.g., 80% for training and 20% for testing).
    2. The model is trained using the training set.
    3. The model is then tested on the test set, which has never been seen by the model.

2. Cross-Validation

- **Description**: Cross-validation is a more robust method than holdout testing. It involves splitting the dataset into multiple folds and training/testing the model multiple times, each time using a different fold as the test set and the remaining folds as the training set.
- **Procedure**:
    1. The dataset is split into **k folds** (typically 5 or 10).
    2. For each iteration, one fold is held out for testing, and the remaining folds are used to train the model.
    3. The process is repeated for each fold, and the model's performance is averaged over all the iterations.
- **Types**:
    - **K-fold Cross-Validation**: The dataset is divided into **k** equal-sized folds, and each fold is used as a test set once.
    - **Stratified K-fold Cross-Validation**: Ensures that the class distribution (spam vs. ham) is similar across each fold, which is particularly important if the dataset is imbalanced.

3. Confusion Matrix

- **Description**: The confusion matrix is a critical tool used to assess the performance of a spam detection model by breaking down the classification results into four categories:
    - **True Positives (TP)**: The number of spam emails correctly classified as spam.

    - **False Positives (FP)**: The number of legitimate emails incorrectly classified as spam (also known as Type I error).
    - **True Negatives (TN)**: The number of legitimate emails correctly classified as ham.
    - **False Negatives (FN)**: The number of spam emails incorrectly classified as ham (also known as Type II error).
- **Usage**: By analyzing the confusion matrix, one can derive performance metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.
    - **Accuracy**: $\frac{TP+TN}{TP+FP+TN+FN}$
    - **Precision**: $\frac{TP}{TP+FP}$
    - **Recall (Sensitivity)**: $\frac{TP}{TP+FN}$
    - **F1-Score**: The harmonic mean of precision and recall:

      $\frac{2\times Precision\times Recall}{Precision+Recall}$

4. Precision-Recall Curve

- **Description**: The **precision-recall curve** is a graphical representation of a model's performance across different thresholds. It plots **precision** against **recall** for various classification thresholds (from 0 to 1). The curve shows the trade-off between precision (how many of the predicted spam emails are actually spam) and recall (how many of the actual spam emails are detected).
- **Procedure**:
    1. The model predicts the probability of an email being spam.
    2. Different thresholds are set (e.g., 0.1, 0.2, 0.3,... 0.9), and for each threshold, the model's performance is measured in terms of precision and recall.
    3. The curve is plotted with precision on the y-axis and recall on the x-axis.

5. Cross-Validation with Hyperparameter Tuning

- **Description**: Hyperparameters are settings or configurations that are not learned from the data (e.g., the regularization strength in SVMs or the number of trees in a Random Forest). Tuning hyperparameters is crucial to optimizing the model's performance.
- **Procedure**:
    1. Use **k-fold cross-validation** to estimate the model's performance.
    2. Simultaneously, apply hyperparameter tuning methods like **Grid Search** or **Random Search** to explore various configurations of the model.
    3. The goal is to find the optimal combination of hyperparameters that results in the best model performance (e.g., maximizing accuracy, precision, recall, or F1-score).

6. A/B Testing and Real-World Testing

- **Description**: A/B testing involves deploying two or more versions of the spam detection system to see how different models or configurations perform in real-world conditions.
- **Procedure**:
    1. Split a sample of the user base into groups that are exposed to different versions of the system.
    2. Measure how each version performs in terms of user satisfaction (e.g., false positives, false negatives).

By using a combination of these methodologies, an email spam detection system can be rigorously tested for accuracy, robustness, and adaptability. These testing approaches ensure that the system effectively identifies spam while minimizing false positives and adapting to new and evolving spam tactics.

CHAPTER 8: CODING AND SCREENSHOTS OF THE PROJECT

Uploading and reading data

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files spam.csv
• spam.csv(text/csv) - 503663 bytes, last modified: 11/16/2024 - 100% done
Saving spam.csv to spam.csv
```

```
[ ] import numpy as np
    import pandas as pd
```

```
[ ] df = pd.read_csv('spam.csv', encoding='ISO-8859-1')
    df.sample(5)
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
820	ham	Good afternoon starshine! How's my boytoy? Doe...	NaN	NaN	NaN
2093	spam	Final Chance! Claim ur å£150 worth of discount...	NaN	NaN	NaN
1612	spam	RT-KIng Pro Video Club>> Need help? info@ringt...	NaN	NaN	NaN
559	ham	Aiyo... U always c our ex one... I dunno abt m...	NaN	NaN	NaN
3581	ham	You are right. Meanwhile how's project twins c...	NaN	NaN	NaN

```
[ ] df.shape
```

```
(5572, 5)
```

Data Cleaning

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

```
[ ] # drop last 3 cols
    df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

```
[ ] df.sample(5)
```

	v1	v2
1550	ham	He says hi and to get your ass back to south t...
4073	ham	A lot of this sickness thing going round. Take...
4496	ham	Ok
3161	ham	I can't describe how lucky you are that I'm ac...
5389	ham	Ok.ok ok..then..whats ur todays plan

```
# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

	target	text
3282	ham	Hey tmr maybe can meet you at yck
4707	ham	Did you say bold, then torch later. Or one tor...
3181	ham	My Parents, My Kidz, My Friends n My Colleague...
1825	ham	Wat makes some people dearer is not just de ha...
2737	ham	Hi Chachi tried calling u now unable to reach ...

```
[ ] from sklearn.preprocessing import LabelEncoder
    encoder = LabelEncoder()
```

```
    df['target'] = encoder.fit_transform(df['target'])
```

```
[ ] df.head()
```

	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
[ ] # missing values
    df.isnull().sum()
```

	0
target	0
text	0

dtype: int64

```
[ ] # check for duplicate values
    df.duplicated().sum()
```

```
403
```

```
[ ] # remove duplicates
    df = df.drop_duplicates(keep='first')
```

```
[ ] df.duplicated().sum()
```

```
0
```

```
[ ] df.shape
```

```
(5169, 2)
```

Exploratory Data Analysis

```
df.head()
```

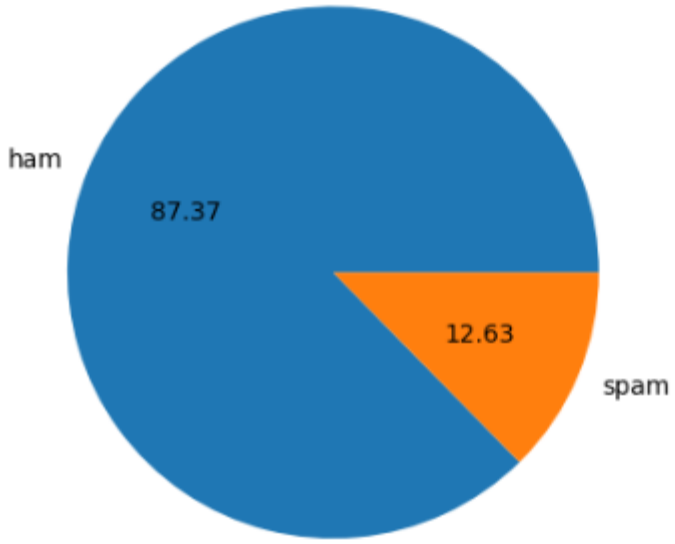
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
[ ] df['target'].value_counts()
```

target	count
0	4516
1	653

dtype: int64

```
[ ] import matplotlib.pyplot as plt
    plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
    plt.show()
```




```
import nltk
!pip install nltk
nltk.download('punkt_tab')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
[ ] df['num_characters'] = df['text'].apply(len)
```

```
[ ] df.head()
```

	target	text	num_characters
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

```
[ ] # num of words
    df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

```
[ ] df.head()
```

	target	text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

```
[ ] df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

```
[ ] df.head()
```

	target	text	num_characters	num_words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

```
[ ] df[['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

```
[ ] # ham
    df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

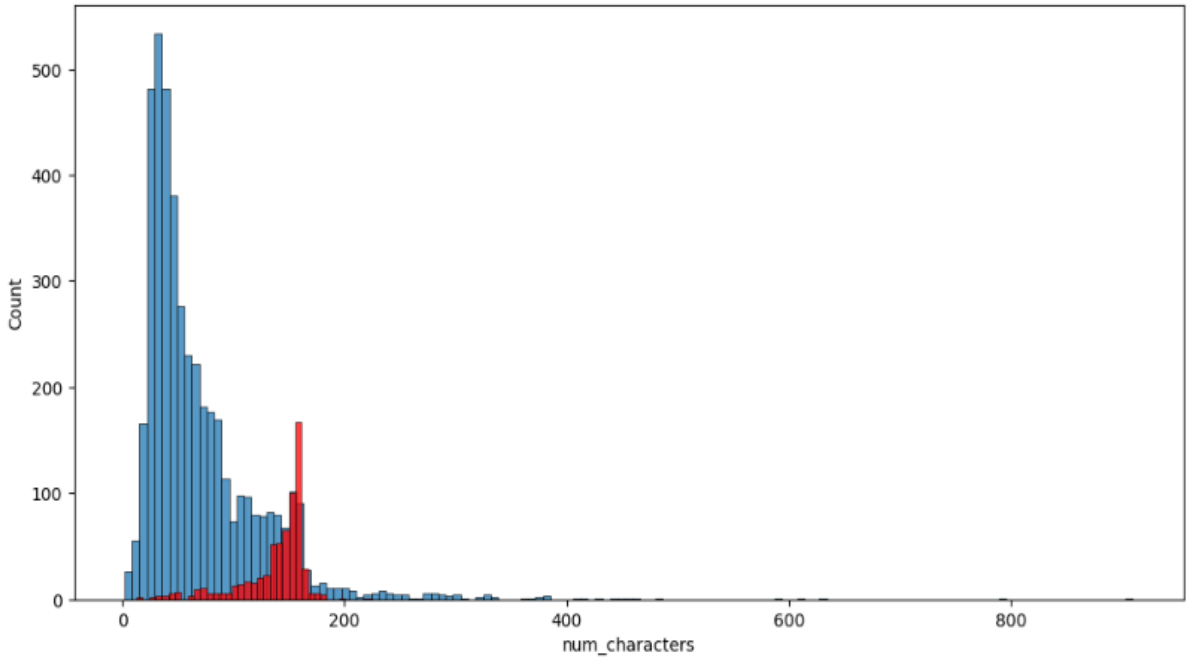
```
[ ] #spam
    df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

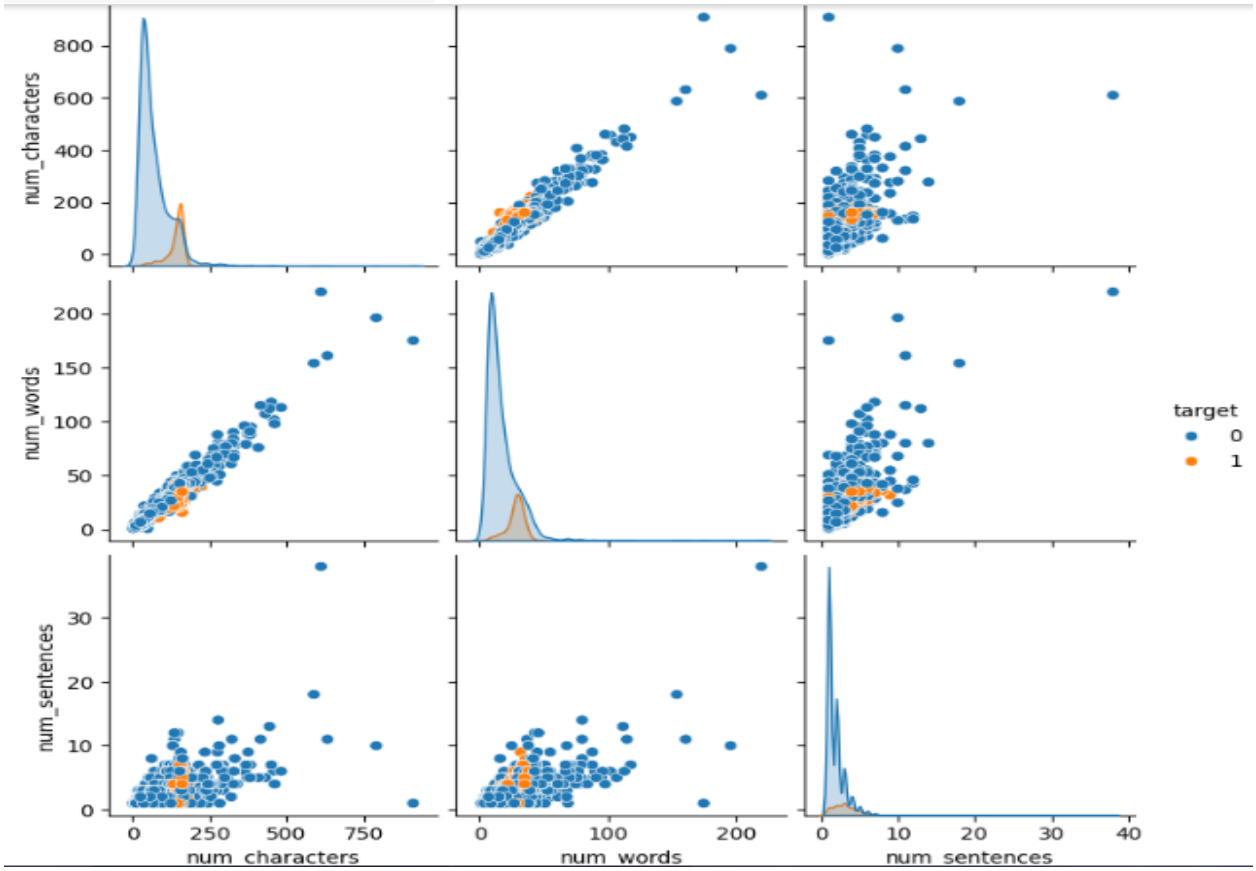
```
import seaborn as sns
```

```
[ ] plt.figure(figsize=(12,6))
    sns.histplot(df[df['target'] == 0]['num_characters'])
    sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

```
<Axes: xlabel='num_characters', ylabel='Count'>
```

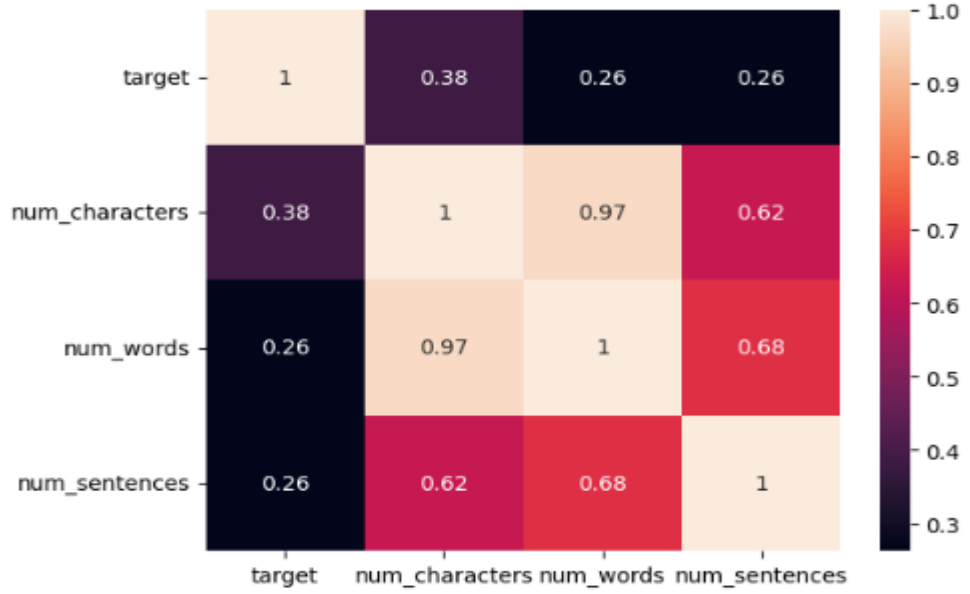


```
[ ]  sns.pairplot(df,hue='target')
```




```
[ ]  numeric_df = df.select_dtypes(include=['float64', 'int64'])
     sns.heatmap(numeric_df.corr(), annot=True)
```

```
<Axes: >
```



```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

```
# Removing stop words and punctuations
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords.words('english')
len(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
179
```

```
#now for punctuation
import string
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
# stemming
from nltk.stem.porter import PorterStemmer
ps =PorterStemmer()
```

```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.")
```

```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

```
df['text'][10]
```

```
'I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.'
```

```
ps.stem('loving')
```

```
'love'
```

```
df['transformed_text'] = df['text'].apply(transform_text)
```

```
df.head()
```

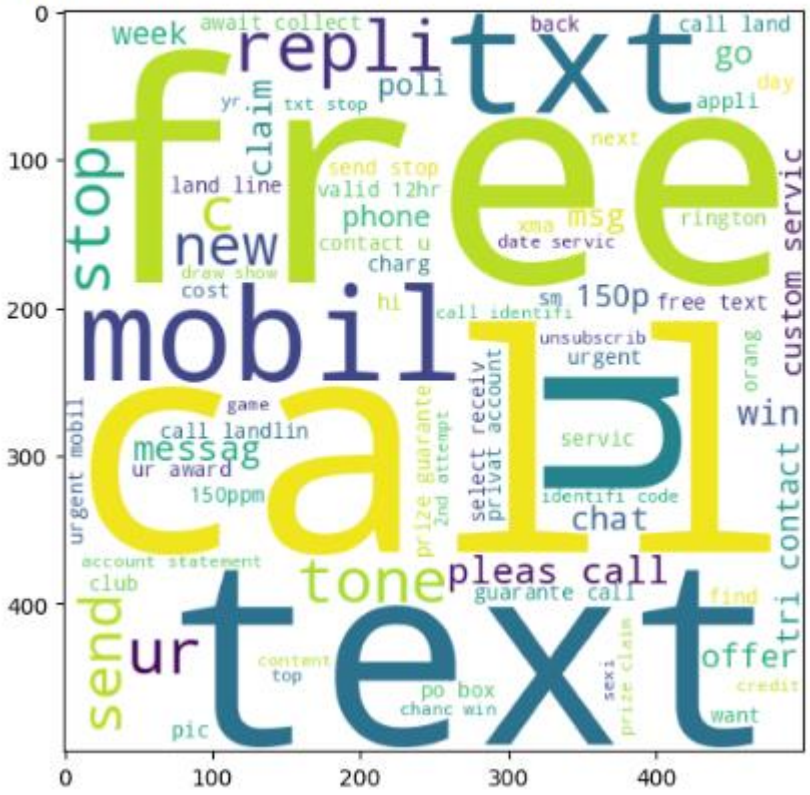
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

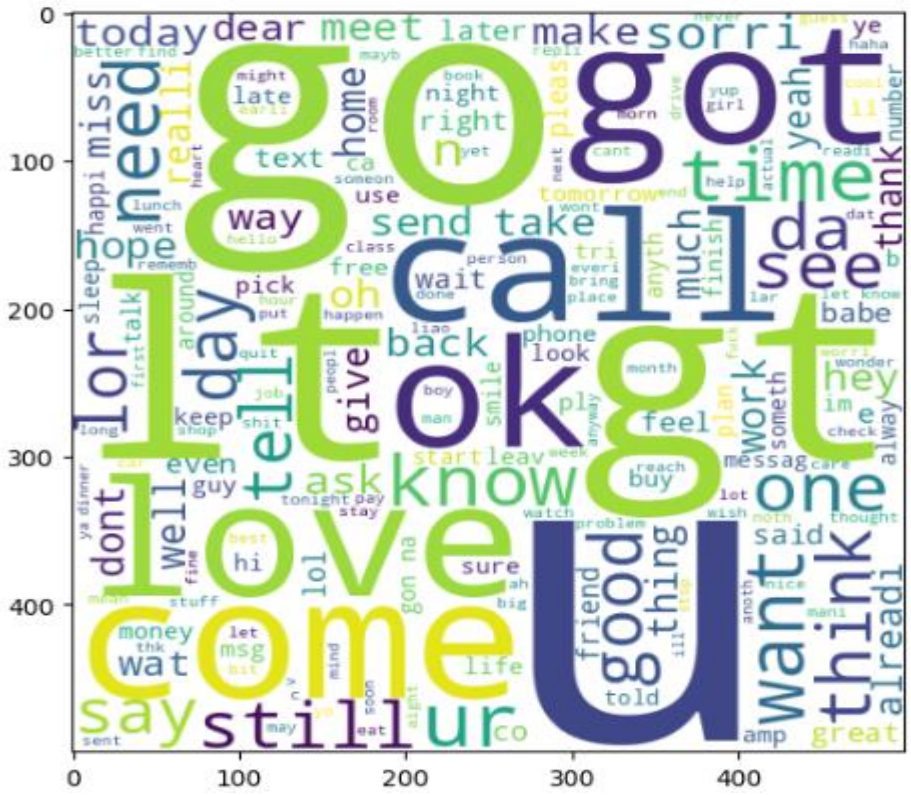
```
<matplotlib.image.AxesImage at 0x7ac7b22d6440>
```



```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x7ac7b1cc9270>
```




```
df.head()
```

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

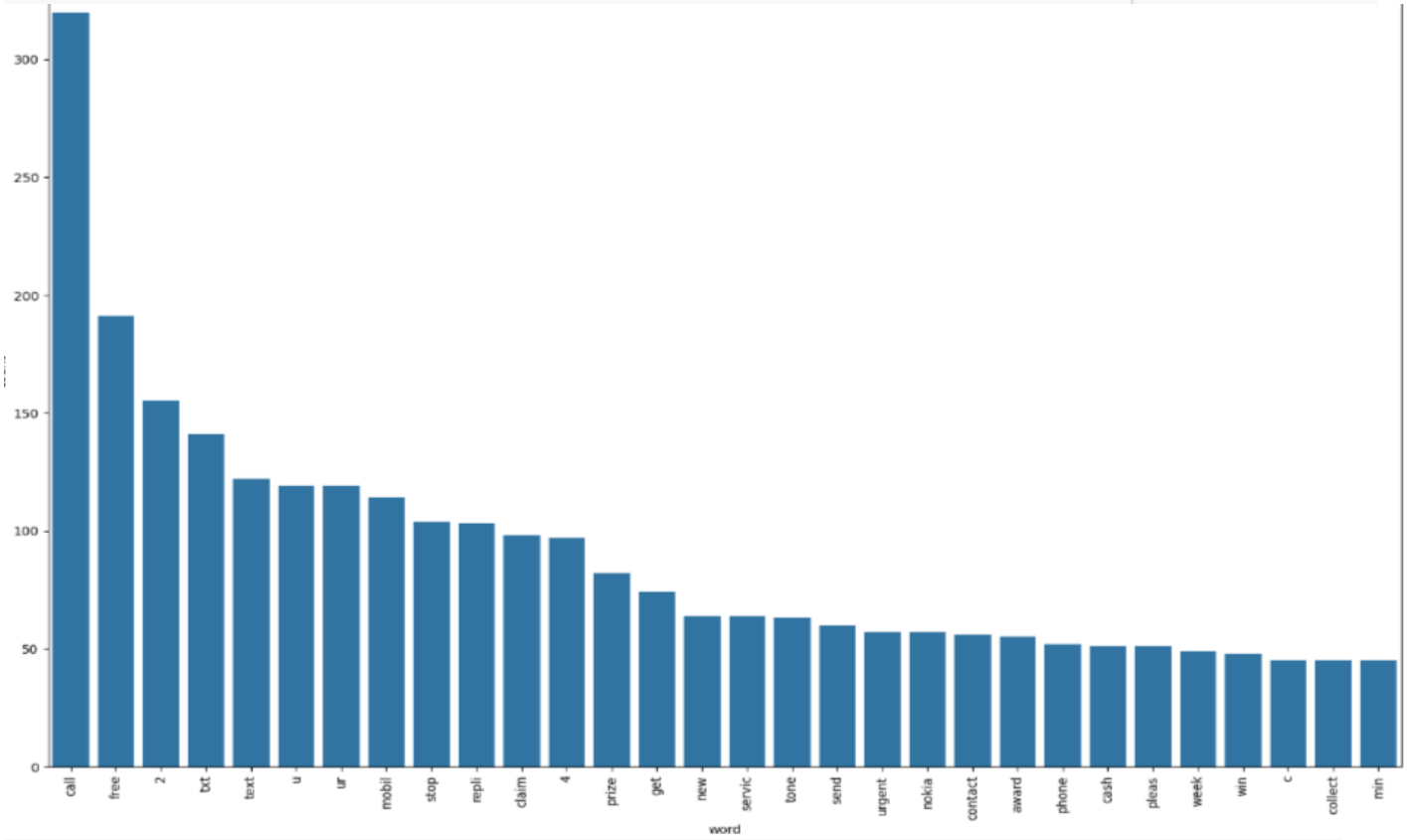
```
len(spam_corpus)
```

```
9939
```

```
df.reset_index(drop=True, inplace=True)
```

```
most_common_words = pd.DataFrame(Counter(spam_corpus).most_common(30), columns=['word', 'count'])

plt.figure(figsize=(18, 12))
sns.barplot(x='word', y='count', data=most_common_words)
plt.xticks(rotation='vertical')
plt.show()
```



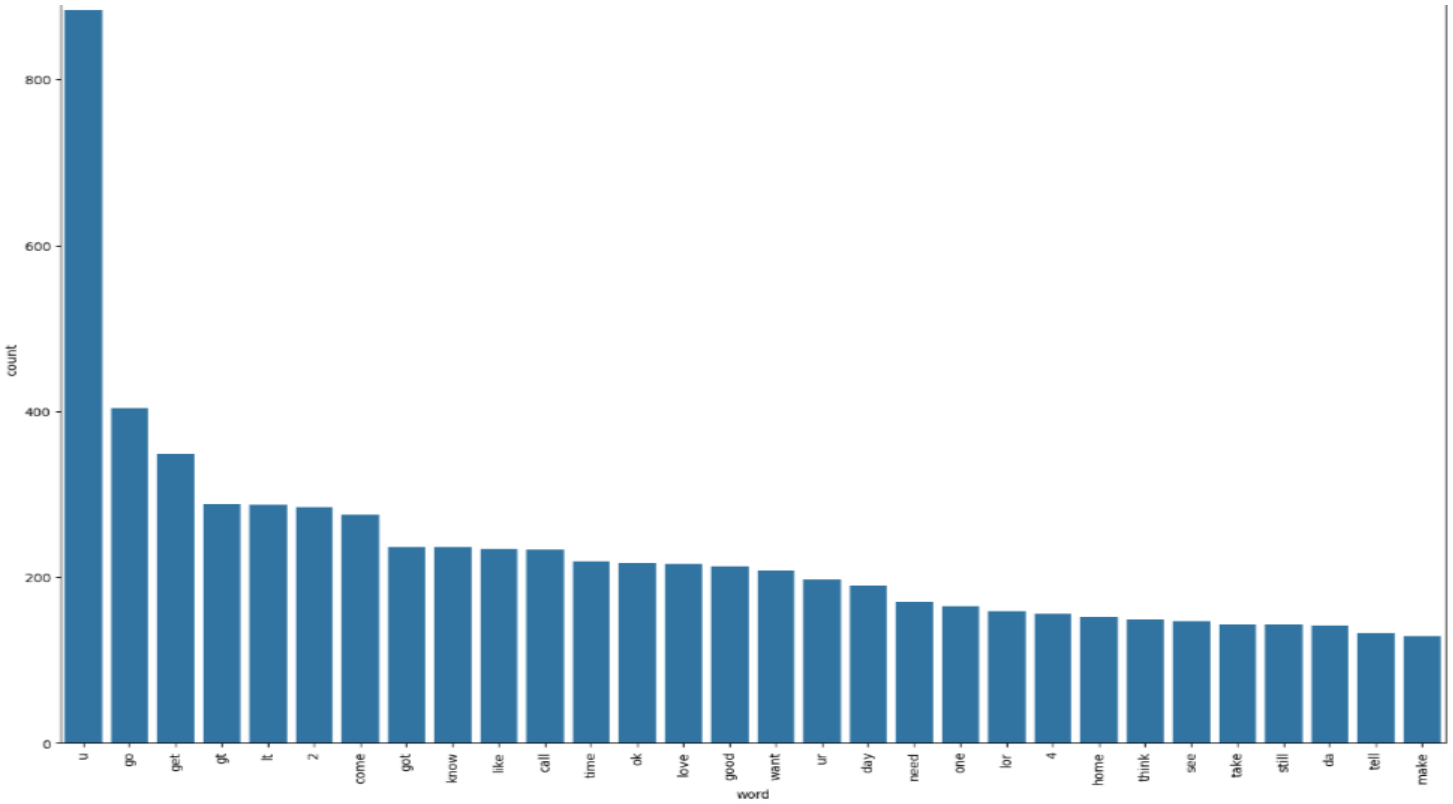

```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

```
len(ham_corpus)
```

```
35404
```

```
ham_common_words = pd.DataFrame(Counter(ham_corpus).most_common(30), columns=['word', 'count'])

# Plot the barplot
plt.figure(figsize=(18, 12))
sns.barplot(x='word', y='count', data=ham_common_words)
plt.xticks(rotation='vertical')
plt.show()
```




```
# Text Vectorization
# using Bag of Words
df.head()
```

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

4. Model Building

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

```
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

```
# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
```

```
X.shape
```

```
(5169, 3000)
```

```
y = df['target'].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

```
# tfidf --> MNB
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

```
def train_classifier(clf,X_train,y_train,X_test,y_tes
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9758220502901354, 0.9747899159663865)
```

```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  DT
Accuracy -  0.9332688588007737
Precision -  0.8415841584158416
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
```

```
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9584139264990329
Precision -  0.8682170542635659
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9468085106382979
Precision -  0.9191919191919192
For  xgb
Accuracy -  0.9671179883945842
Precision -  0.9262295081967213
```

```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',ascending=False)
```

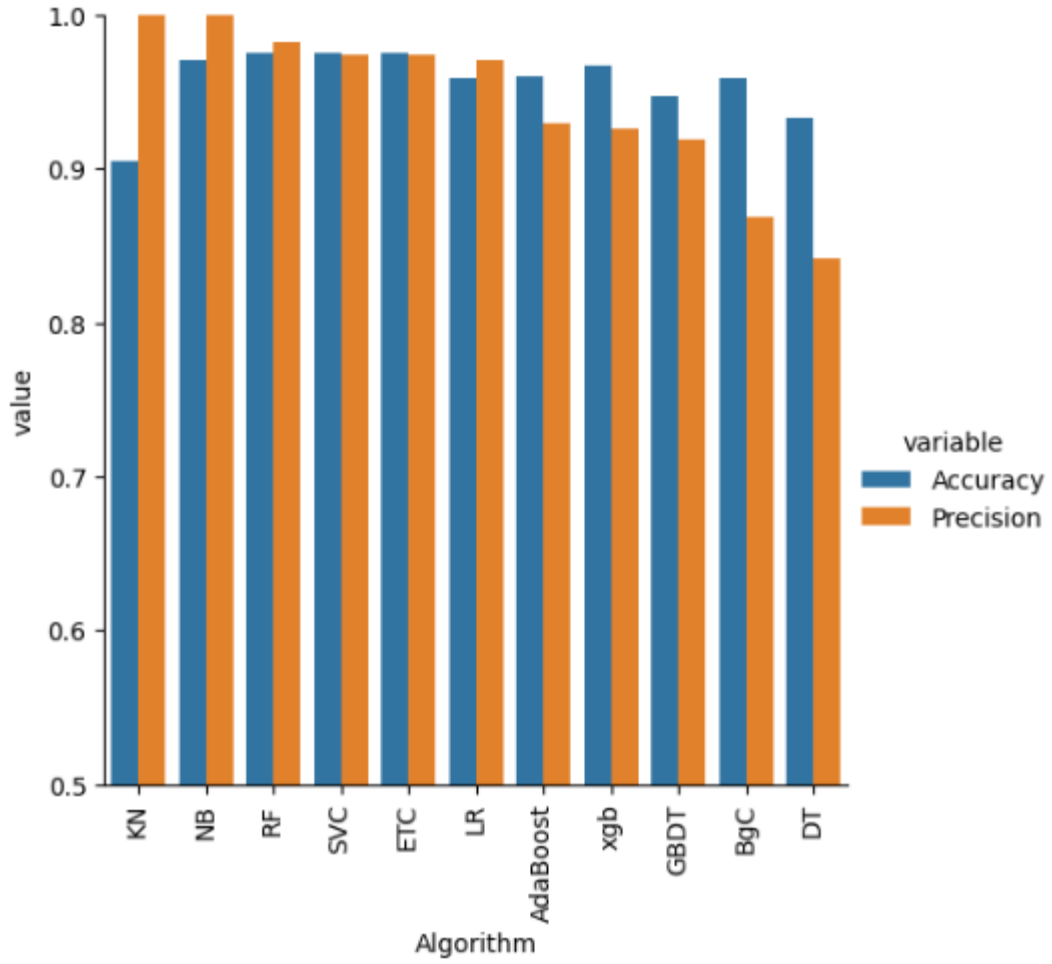
```
performance_df
```

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
10	xgb	0.967118	0.926230
9	GBDT	0.946809	0.919192
7	BgC	0.958414	0.868217
3	DT	0.933269	0.841584

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

	Algorithm	variable	value
0	KN	Accuracy	0.905222
1	NB	Accuracy	0.970986
2	RF	Accuracy	0.975822
3	SVC	Accuracy	0.975822
4	ETC	Accuracy	0.974855
5	LR	Accuracy	0.958414
6	AdaBoost	Accuracy	0.960348
7	xgb	Accuracy	0.967118
8	GBDT	Accuracy	0.946809
9	BgC	Accuracy	0.958414
10	DT	Accuracy	0.933269
11	KN	Precision	1.000000
12	NB	Precision	1.000000
13	RF	Precision	0.982906
14	SVC	Precision	0.974790
15	ETC	Precision	0.974576
16	LR	Precision	0.970297
17	AdaBoost	Precision	0.929204
18	xgb	Precision	0.926230
19	GBDT	Precision	0.919192
20	BgC	Precision	0.868217
21	DT	Precision	0.841584

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```




```
# model improve
# 1. Change the max_features parameter of TfIdf
```

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,
                        'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000',ascending=False)
```

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,
                        'Precision_scaling':precision_scores}).sort_values('Precision_scaling',ascending=False)
```

```
new_df = performance_df.merge(temp_df,on='Algorithm')
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,
                        'Precision_num_chars':precision_scores}).sort_values('Precision_num_chars',ascending=False)
```

```
new_df_scaled.merge(temp_df,on='Algorithm')
```

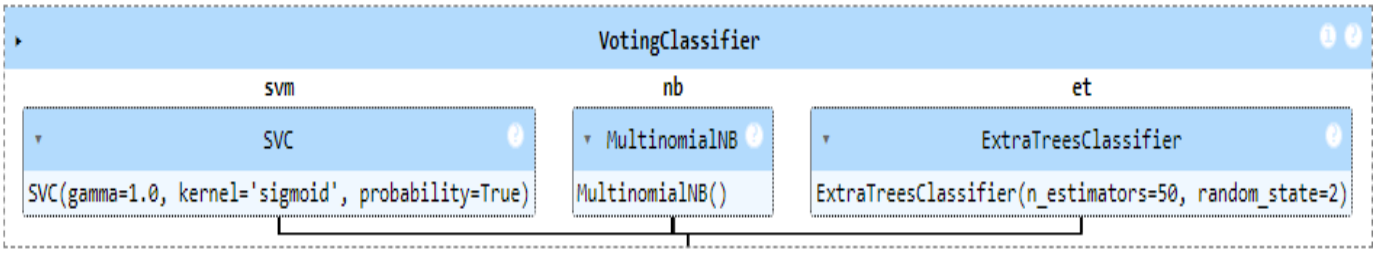
	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y	Accuracy_num_chars	Precision_num_chars
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
2	RF	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576
5	LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297
6	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204
7	xgb	0.967118	0.926230	0.967118	0.926230	0.967118	0.926230	0.967118	0.926230
8	GBDT	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192
9	BgC	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217
10	DT	0.933269	0.841584	0.933269	0.841584	0.933269	0.841584	0.933269	0.841584

```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier
```

```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
```

```
voting.fit(X_train,y_train)
```




```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```


```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```


```
from sklearn.ensemble import StackingClassifier
```

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9806576402321083
Precision 0.946969696969697
```

5. Deployment

```
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()
```

```
    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)

tfidf = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model.pkl','rb'))

st.title("Email/SMS Spam Classifier")

input_sms = st.text_area("Enter the message")

if st.button('Predict'):
    # 1. preprocess
    transformed_sms = transform_text(input_sms)
    # 2. vectorize
    vector_input = tfidf.transform([transformed_sms])
    # 3. predict
    result = model.predict(vector_input)[0]
    # 4. Display
    if result == 1:
        st.header("Spam")
    else:
        st.header("Not Spam")
```

CHAPTER 9: CONCLUSION AND FUTURE SCOPE

CONCLUSION

Spam is typically pointless and occasionally dangerous. Such communications are spam if you get them, and spam emails are spam if they appear in your inbox. Spammers are continually changing their strategies to bypass spam filters. The algorithm must continually be modified to capture the majority of spams, which is a significant effort that most services lack. Most free mail services don't do it, but Gmail and a few other commercial mail checking services do. In this study, we examined ML methods and how they were used to spam filtering. For the purpose of classifying communications as spam or ham, a study of the top calculations in use is provided. Examined were the attempts made by several analysts to use ML classifiers to address the spam problem. It was examined how systems for spotting spam messages have evolved over time to help users avoid channels. The raw, unstructured nature of acquired email data is the first state. It is cleansed before EDA is applied to it to extract the data's insights. The authors pre-process the data based on the EDA results. Stop words are removed from the data after stemming and pre-processing. Then, crucial information is obtained using word tokenization. The dimensionality of the data and characteristics is decreased during the pre-processing stage. Then, machine learning models are used, and their accuracy levels are compared to obtain the best effective spam detection method. Python software is used to assess the accuracy and precision of various machine learning methods. In all of the many data circumstances, the suggested RNN model was able to attain the highest accuracy of any.

Future Scope

The research from this investigation can be expanded upon further recipient-related characteristics that can be added from organisation databases, as well as file level Metadata elements like document path location, author names, and so forth. Additionally, it can broaden multi-class results that connect to a particular recipient. This method is quite helpful for corporate email messaging processes (for instance, a medical email web portal, where a message may belong to more than two folders, and where the strategy of folding processes sends the incoming message to the multiple folder with a specified weighing scheme which will help in classification with more accuracy.
Delivering useful emails to the recipient while separating junk emails is the goal of spam detection. Every email service provider already includes spam detection, but it is not always accurate; occasionally, it labels useful emails as spam. This study suggests a more effective method for categorising emails using comparative analysis, in which different machine learning models are used to analyse the same dataset and the accuracy of each model is determined. In terms of future work, it is possible to create a website that will be open to all users and allow users to quickly identify spam or junk mail. They merely need to type their email address into the provided text field, and it will identify them properly.

Applications

1. It Smoothes out Inboxes

The typical office labourer gets about 121 messages each day, a big part of which are assessed to be spam. In any case, even at 60 messages per day, it is not difficult to lose significant correspondences to the sheer number that are coming in. This is one of the mystery advantages of spam sifting that individuals have hardly any familiarity with: it just smoothes out your inbox. With less trash coming into your inbox, you can really go through your messages all the more successfully and keep in contact with the people who matter.

2. Safeguard Against Malware

More astute spam gets into more inboxes, which makes it bound to be opened and bound to actually hurt. With spam separating, you can keep steady over the many spam strategies that are being utilised today so you can guarantee that your email inboxes stay liberated from unsafe messages.

3. Keeps User Consistent

Numerous little and medium measured organisations are missing out on significant clients today in light of the fact that their network safety isn't satisfactory. Spam separating is a significant piece of any network safety plan, and it assists you with remaining consistent with the desires and requests of organisations and offices that are worried about their data. Without appropriate spam sifting, you could accidentally place spyware in your messages and break security conventions. The outcome could be a deficiency of business, notoriety, and eventually pay.

4. Protects Against Monetary Frauds

Consistently, somebody succumbs to a phishing trick, a specific sort of spam-based conspiracy where somebody thinks they are receiving a genuine email and winds up unveiling charge card data. Now and then it is an individual Visa, at times it is an organisation charge card. In the two examples, the outcome is losing important time and cash to a trick. Spam sifting is likewise extraordinarily reasonable, making it a modest yet incredibly viable method for protecting yourself. Inboxes are powerful devices for correspondence, not a spot where anybody can get into and begin hitting you with futile or risky messages. For that reason spam sifting is a particularly significant part of current organisations. Instead of depending on obsolete, free spam separating administrations, pick Securence spam sifting. With authorised security conventions, it can assist your business with conveying all the more successfully while keeping malware out of your inboxes, for short of what you might think.

REFERENCES

[1] C. Yang, R. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers", In: Recent Advances in Intrusion Detection, Springer Berlin/Heidelberg, pp.318-337, 2011.

[2] S. Kumar, and S. Arumugam, "A Probabilistic Neural Network Based Classification of Spam Mails Using Particle Swarm Optimization Feature Selection", Middle-East Journal of Scientific Research, Vol.23, No.5, pp.874-879, 2015.

[3] N. P. DíAz, D. R. OrdáS, F. F. Riverola, and J. R. MéNdez, "SDAI: An integral evaluation methodology for content-based spam filtering models", Expert Systems with Applications, Vol.39, No.16, pp.12487-12500, 2012.

[4] A. K. Sharma, S. K. Prajapat, and M. Aslam, "A Comparative Study Between Naive Bayes and Neural Network (MLP) Classifier for Spam Email Detection", In: IJCA Proceedings on National Seminar on Recent Advances in Wireless Networks and Communications. Foundation of Computer Science (FCS), pp.12- 16, 2014.

[5] W. Ma, D. Tran, and D. Sharma, "A novel spam email detection system based on negative selection", In: Proc. of Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT 09, Seoul, Korea, pp.987-992, 2009.

[6] https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fspam-detection-with-logistic-regression-23e3709e522&psig=AOvVaw2X9gZ_Rwql7wqWrKNDQibe&ust=1668795984348000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCMCW3tHrtfsCFQAAAAAdAAAAABAD.

[7] https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.javatpoint.com%2Ftypes-of-machine-learning&psig=AOvVaw1xJK1kN-JXPxx4yJjPKjz6&ust=1668796135528000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCMCEiZnstfsCFQAAAAAdAAAAABAD