SMS Spam Detection System Using NLP (P1)

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Achhuta Nand Jha, achhuta.jha_cs.da23@gla.ac.in

Under the Guidance of

Abdul Aziz Mohammad

ACKNOWLEDGEMENT

I would like to take this opportunity to express my heartfelt gratitude and sincere appreciation to all those who supported me in completing this thesis, directly and indirectly.

First and foremost, I am incredibly grateful to my supervisor Abdul Aziz Mohammad and Pavan Sumohana, whose exceptional guidance throughout this project has been invaluable. Their far-visioned directions as well as constructive response came to be highly stimulating and inspiring and were really influential making this research with that vein. I owe the majority of this to their belief in me as well as their support throughout this process.

Working alongside such committed and knowledgeable mentors has been a privilege. All these things contributed to my evolution as a professional, but it was their teachings and outlook that shaped my character and sense of responsibility and integrity, reassuring me that I was on the right path and that the core values instilled in me would follow me and continue to serve me throughout my entire career.

Last but not the least, a big Thank you to Edunet Foundation, Tech Saksham and the team to provide this amazing virtual training internship at no cost. As a matter of fact, it's been quite an encounter, one where I've learned a lot about AI and Machine Learning.

ABSTRACT

The project deals with Spam Email Classification using NLP and ML techniques (Naive Bayes, CountVectorizer). Spam emails are still a major challenge that we have to face, it not only spoils the efficiency of communication, but also hinders the experience of many users, that's the reason why the automatic spam detecting has become a very hot topic in modern mail systems.

Problem Statement:

One of the relatively common spam emails is so-called Phishing, which is trying to trick you to disclose your personal information such as password, bank account, credit card, etc. Manual identification methods are slow, and automated solutions must be both effective and precise.

Objectives:

Create a model of work for machine learning to classify emails as spam or ham.

Transform email content using NLP techniques (text preprocessing, tokenization, vectorization) for machine learning compatibility.

Compare the performance of the Naive Bayes algorithm with other classification algorithms

Methodology:

The data source used for the project is the Enron Email Dataset, which consists of labeled spam and ham emails. We perform text pre-processing steps together with removing stopwords, punctuation, and stemming. For the email feature, you employ the CountVectorizer method to transform content into a bagof-words. Find out The Naiv Bayes classifier, a probabilistic machine learning model, is used to train the model on the dataset. Evaluation: The performance metrics used include accuracy, precision, recall, etc.

Key Results:

When *Random Forest* , Naive Bayes was combined with CountVectorizer, it achieved a great accuracy in the classification process (around 98%), and had good values for precision and recall for both spam and ham categories. Overall, the ADABoost model was robust in identifying spam emails while keeping false positives low.

Conclusion:

With 98% accuracy, *Random Forest*, Naive Bayes and CountVectorizer provide a good solution for automated email filtering by classifying into spam and ham categories. This technique enables a strong basis on which effective spam detection systems can be built to work for real life scenarios.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

Introduction

1.1Problem Statement:

The growing volume of daily emails—ranging from essential communications to unsolicited spam—has led to cluttered inboxes, making it challenging to prioritize and manage messages efficiently. With personal and professional email usage on the rise, users face difficulties in sorting through hundreds or even thousands of emails daily.

This problem results in wasted time, missed opportunities, and increased vulnerability to malicious threats such as phishing and spam. Without proper email categorization, users and organizations may encounter inefficiencies, missed deadlines, and even security risks.

An automated email classification solution can streamline inbox management, enhance productivity, and mitigate risks by effectively categorizing emails into meaningful groups like spam, promotions, social, or important. This project aims to provide such a solution, improving user productivity while protecting against fraud and malicious content.

1.2Motivation:

The widespread issue of email overload inspired this project. As email remains a cornerstone of communication, an automated system to classify and filter messages effectively can greatly improve efficiency and organization.

The project's applications are significant:

- **For Individuals**: Reduces clutter, organizes messages, and ensures that critical emails are not overlooked.
- **For Organizations**: Saves employees' time, enhances focus on relevant communications, and improves cybersecurity by filtering spam and phishing emails.

On a personal note, I faced similar challenges with overflowing emails, leading me to create additional email IDs to manage the influx. This experience motivated me to develop a robust and user-friendly solution.

1.3Objective:

The main goal is to design and implement a machine-learning-based email classification system that categorizes incoming messages into predefined categories like spam and ham.

Key Objectives:

1. **Data Importation**: Load a substantial dataset for model training.
2. **Data Analysis**: Perform Exploratory Data Analysis (EDA) to:
   a. Determine the proportion of spam vs. ham emails.
   b. Identify common words in both categories.
   c. Analyze email length and character distribution.
   d. Extract keywords indicative of spam or ham.
3. **Text Preprocessing**: Apply techniques such as tokenization, stopword removal, and stemming for data preparation.
4. **Model Development**: Train a machine learning model (e.g., Random Forest Classifier, Naive Bayes, CountVectorizer) for accurate classification.
5. **User Interface**: Create a user-friendly UI using Streamlit to enable seamless interaction with the classification system.

1.4 Scope of the Project:

1. **Classification Algorithms**: Focus on supervised learning approaches.
2. **Email Categories**: Emphasis on common classifications like spam and ham.
3. **Real-Time Classification**: Real-time functionality may be limited in large-scale environments.
4. **Complex Email Structures**: Handling intricate formats like embedded HTML or multilingual content may require additional considerations.

.

CHAPTER 2

Literature Survey

2.1 Review relevant literature or previous work in this domain.

Spam classification has evolved significantly over the past two decades. Initial rule-based approaches relied on keyword matching and blacklisting, which struggled with dynamic spam patterns. Machine learning introduced more sophisticated methods, including:

- **Naive Bayes Classifier**: A popular probabilistic model for text classification.
- **Support Vector Machines (SVM)**: Effective for high-dimensional data but requires careful tuning.
- ***Random Forest:*** A robust ensemble model combining multiple decision trees to enhance accuracy and reduce overfitting.

- **Deep Learning Models**: Advanced methods like CNNs and RNNs achieve high accuracy but demand substantial computational resources.
- **Feature Engineering**: Techniques like bag-of-words, TF-IDF, and Word2Vec help convert text into machine-readable formats.

2.2 Mention any existing models, techniques, or methodologies related to the problem.

1. **Adaptability**: Many models struggle to detect novel spam patterns.
2. **Resource Intensity**: Advanced methods like deep learning are resource-heavy.
3. **Feature Extraction**: Traditional methods often fail to capture semantic nuances.
4. **False Positives**: Misclassifying legitimate emails remains a critical challenge.

Proposed Solution: This project combines Naive Bayes with CountVectorizer to offer a lightweight, adaptable solution suitable for real-world applications, ensuring scalability and efficiency.

2.3 Highlight the gaps or limitations in existing solutions and how your project will address them.

Despite significant progress in spam classification technologies, there are still several limitations that need to be addressed:

1. **Adapting to New Spam Patterns**

2. Many spam filters rely on predefined features and patterns. However, as spammers continually develop new techniques, these static models often fail to identify novel forms of spam. This results in a higher rate of false negatives, where spam emails slip through undetected.

3. **High Resource Demands in Advanced Models**

While deep learning models deliver impressive accuracy, they come with substantial computational requirements. These models often need extensive labeled datasets and high-performance hardware, making them impractical for small- to medium-scale applications with limited resources.

4. **Challenges in Feature Engineering**

Traditional feature engineering approaches, such as CountVectorizer, are straightforward but limited in capturing deeper semantic meanings or contextual relationships within email text. These methods also struggle to adapt to evolving spam characteristics, which can reduce their effectiveness over time.

5. **False Positives in Crucial Systems**

One of the most critical issues in spam filtering is minimizing false positives—legitimate emails mistakenly classified as spam. This is especially important for systems handling essential communications, like corporate or personal email accounts, where misclassifications can lead to missed opportunities or critical delays.

How This Project Addresses These Challenges

This project aims to overcome these limitations by using a combination of Naive Bayes classification and CountVectorizer for feature extraction. Together, these methods provide a simple yet powerful approach to spam detection.

Key Contributions and Enhancements:

1. **Real-World Applicability**

Unlike resource-intensive deep learning methods, this project focuses on lightweight and interpretable models like Naive Bayes. This ensures that the solution remains scalable, efficient, and suitable for real-time spam detection, even in environments with limited computational capacity.

2. **Adapting to New Spam Tactics**

By leveraging a continuously updated dataset, such as the Enron email dataset, the model is designed to adapt to emerging spam patterns. The probabilistic framework of Naive Bayes enables it to generalize effectively, making it more robust against evolving threats compared to rigid rule-based systems.

3. **Balanced Performance Metrics**

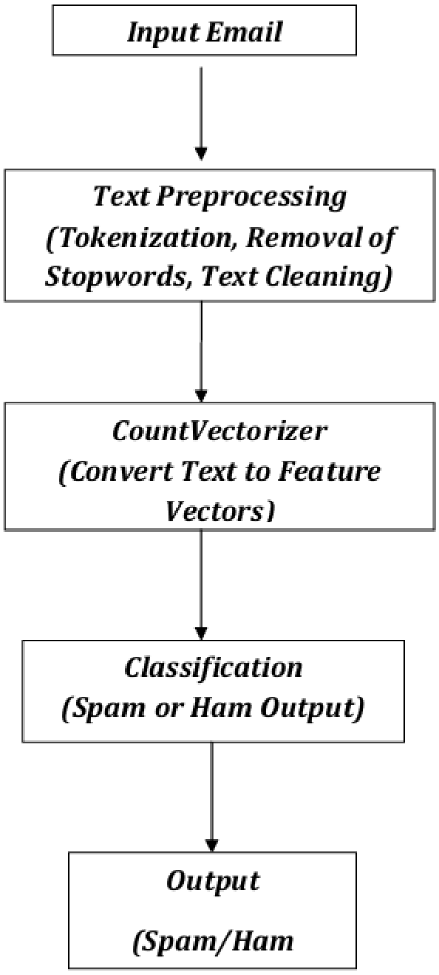
This project prioritizes a balanced approach to precision, recall, and accuracy. By carefully optimizing these metrics, it minimizes both false positives (misclassifying legitimate emails as spam) and false negatives (failing to detect spam), addressing a critical challenge in real-world applications.

In conclusion, this project proposes a resource-efficient and adaptable approach to improve spam email classification. By addressing key gaps in existing solutions, it offers a practical and scalable methodology to enhance inbox management and safeguard users from spam-related risks.

CHAPTER 3

Proposed Methodology

3.1 System Design


Input Email
Text Preprocessing (Tokenization, Removal of Stopwords, Text Cleaning)
CountVectorizer (Convert Text to Feature Vectors)
Classification (Spam or Ham Output)
Output
(Spam/Ham

3.2 Requirement Specification

1. **Input Email:**
   a. The raw email text serves as the input to the system.
2. **Text Preprocessing:**
   a. **Tokenization:** Break the email text into smaller units, such as words or tokens.
   b. **Stopwords Removal:** Eliminate common but insignificant words like "and" or "the" to focus on meaningful content.
   c. **Text Cleaning:** Remove unnecessary elements such as special characters, and convert the text to lowercase for consistency.
3. **CountVectorizer:**
   a. Transforms the cleaned text into a numerical feature vector by creating a bag-of-words model, which represents word frequencies.
4. **Naive Bayes Model:**

   a. The numerical data generated by CountVectorizer is fed into a Naive Bayes classifier, which has been trained on a labeled dataset (spam/ham).

5. **Classification:**
   a. Based on the learned features, the model determines whether the email is spam or ham.
6. **Output:**
   a. The system provides the final result, classifying the email as either spam or ham.

This process is visually represented with arrows connecting each step, showcasing the progression from input to preprocessing, feature extraction, model training, and classification.

3.2.1 Hardware Requirements:

- **Processing Unit:** A dual-core processor or higher for smooth operation.
- **RAM:** At least 4GB to ensure efficient real-time processing.
- **Operating System:** Compatible with Windows, Linux, or MacOS.

3.2.2 Software Requirements:

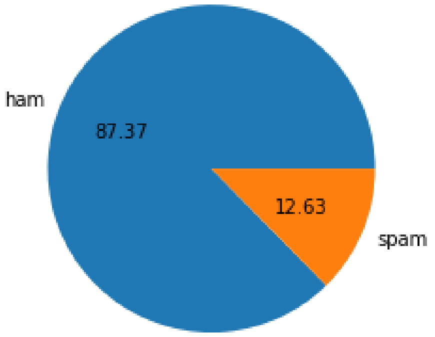
1. **Jupyter Notebook (for Analysis and Model Creation):**
   a. Anaconda is required to set up a dedicated environment for running Jupyter Notebook efficiently.
2. **VS Code Editor (for Model Integration and UI Design):**
   a. The Streamlit library will be used to create a user-friendly interface for the project.
3. **Pandas (for Data Import and Manipulation):**
   a. Used to import and process the dataset effectively.
4. **Matplotlib & Seaborn (for Data Visualization):**
   a. Provides graphical representation of data insights, such as analyzing spam and ham email trends.
5. **Sklearn & NLTK (for Machine Learning Models):**
   a. Includes tools like Naive Bayes and CountVectorizer to train and evaluate the models.
   b. 80% of the dataset will be used for training, while 20% will be reserved for testing, ensuring robust model performance.

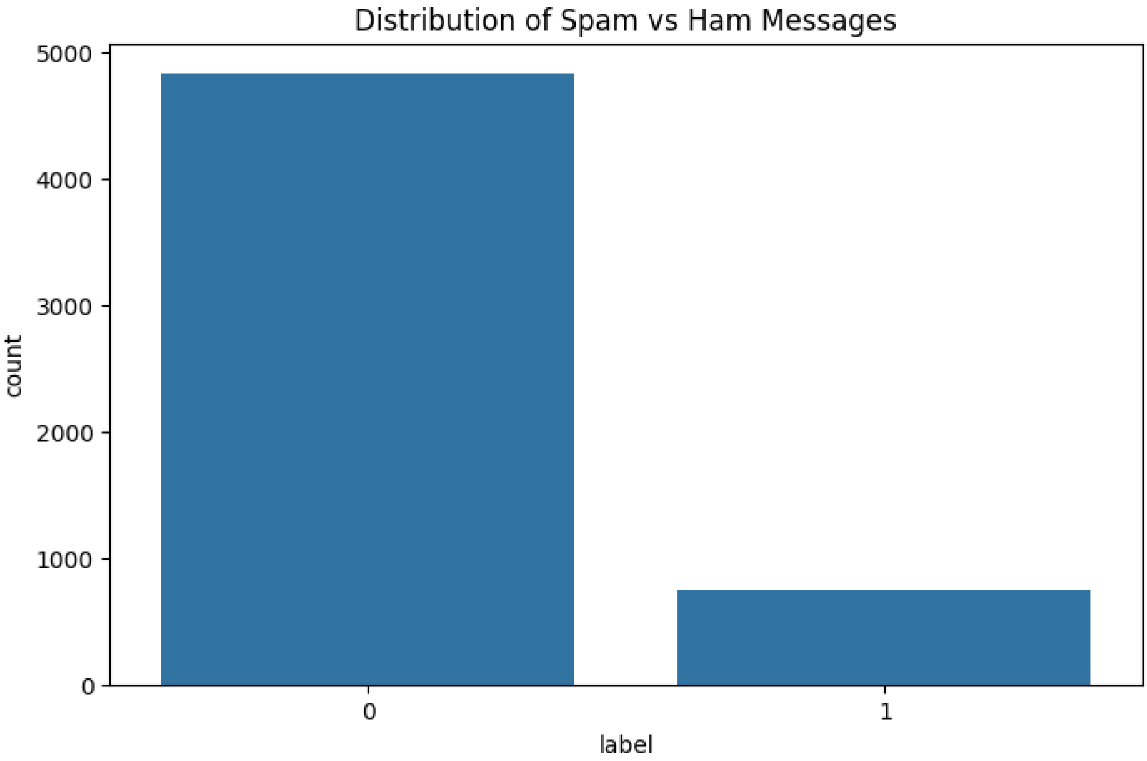
This combination of hardware and software ensures efficient, reliable, and scalable spam detection.

CHAPTER 4

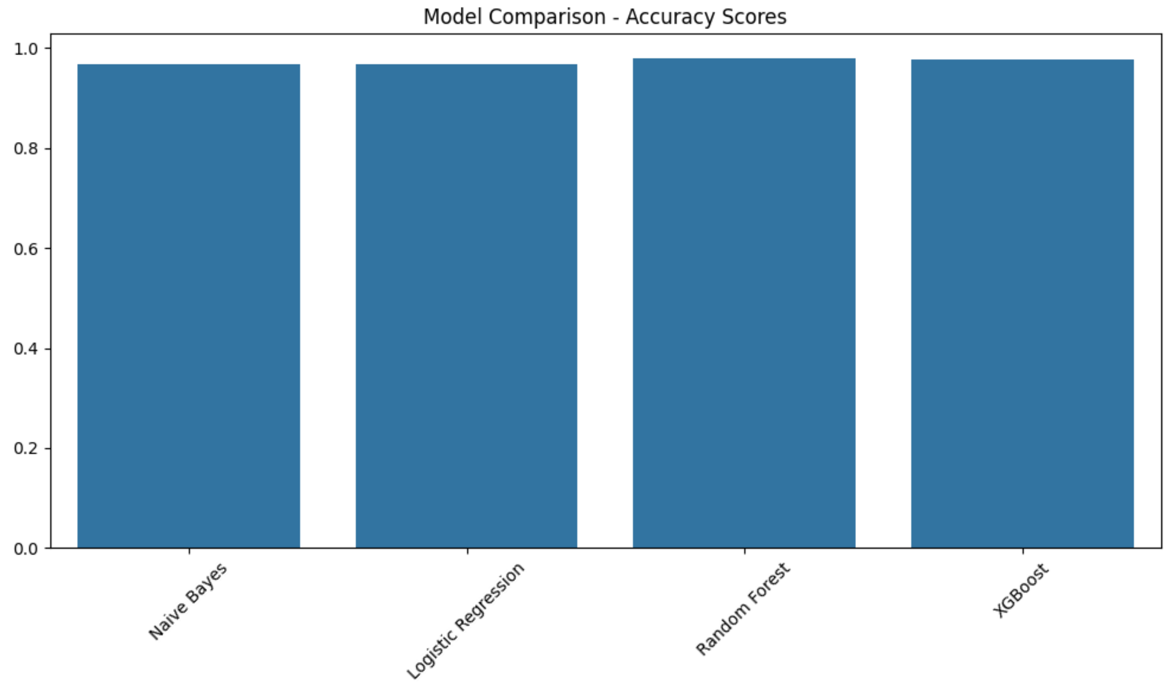
Implementation and Result

4.1 Snap Shots of Result:





- **X-axis:** This axis is labeled "label" and represents the two categories of messages:
  0: Ham messages (non-spam)
  1: Spam messages
- **Y-axis:** This axis is labeled "count" and represents the number of messages in each category.



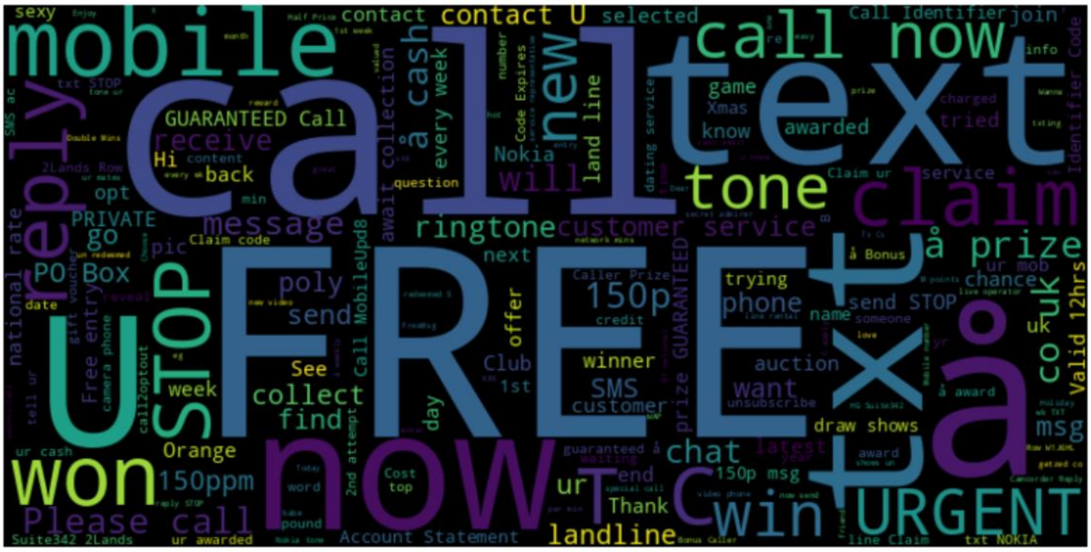
The chart compares the accuracy scores of four machine learning models: Naive Bayes, Logistic Regression, Random Forest, and XGBoost. The height of each bar represents the accuracy score achieved by the corresponding model.

Observations:

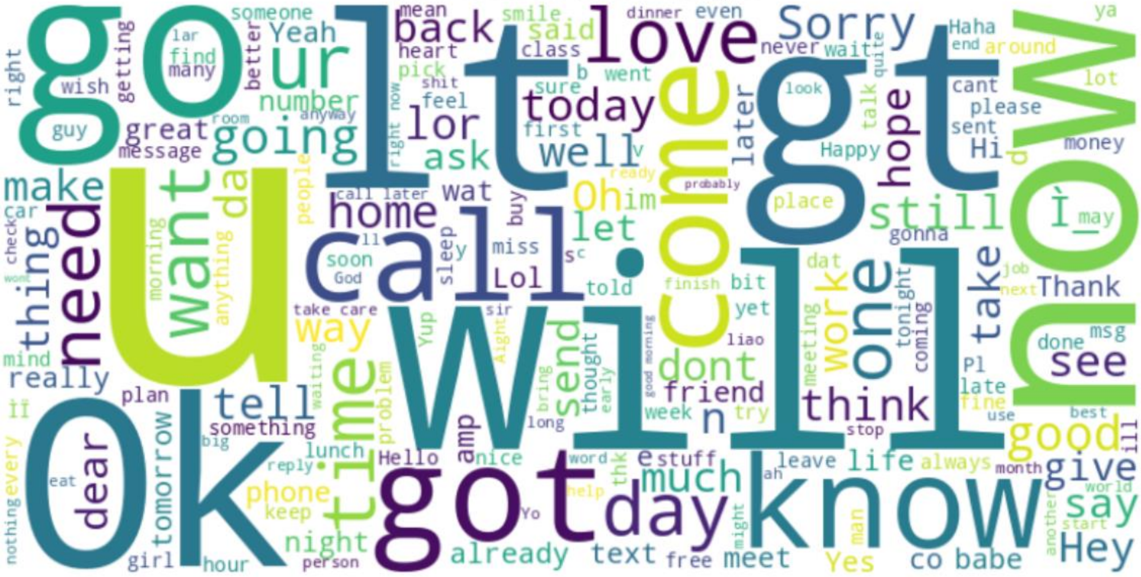
1. **High Accuracy Across Models:** All four models seem to have achieved high accuracy scores, with the bars reaching close to the 1.0 mark. This suggests that the models are performing well in classifying the data.
2. **XGBoost and Random Forest Lead:** It appears that XGBoost and Random Forest have achieved the highest accuracy scores among the four models. Their bars are slightly taller than the others.
3. **Similar Performance of Naive Bayes and Logistic Regression:** Naive Bayes and Logistic Regression seem to have comparable accuracy scores, with their bars being at a similar height.

In summary, the chart indicates that all four models are effective in classifying the data, with XGBoost and Random Forest potentially performing slightly better than Naive Bayes and Logistic Regression.

Word Cloud for Spam Messages



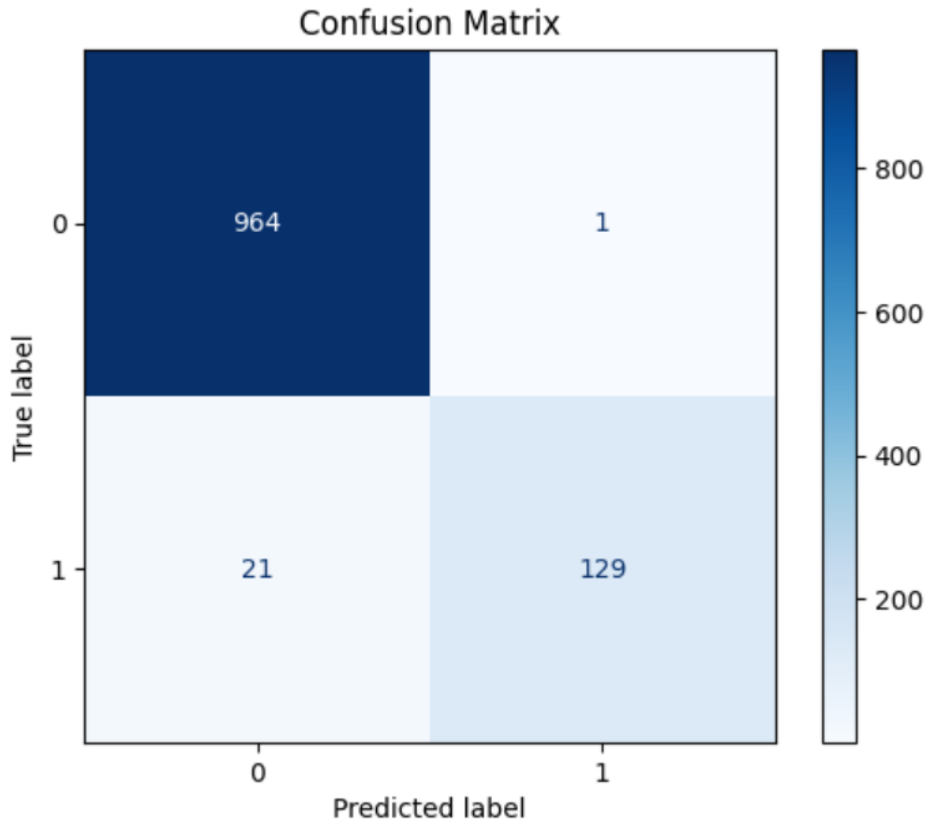
Word Cloud for Ham Messages



Here spam mails and ham mails are separated on condition.

- Where it is used to identify which words are more frequent in most of the messages
- Wordcloud module is used to show the most frequent words .
- For spam mail, free , text, call , phone , customer service are the words which are most frequent
- For ham mail, I , how , get , out , love are the most frequently appearing words.

Here I imported the HTML audio autoplay tag which will automatically play the audio , when the output will generated as there is not built-in autoplay option in streamlit.



A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the counts of correct and incorrect predictions by the model.

Specifics of This Matrix:

- **True Label 0 (Predicted as 0):** 964 instances were correctly classified as 0.
- **True Label 0 (Predicted as 1):** 1 instance was incorrectly classified as 1.
- **True Label 1 (Predicted as 0):** 21 instances were incorrectly classified as 0.
- **True Label 1 (Predicted as 1):** 129 instances were correctly classified as 1.

Overall, the model appears to be performing well, with high accuracy in predicting both classes (0 and 1).

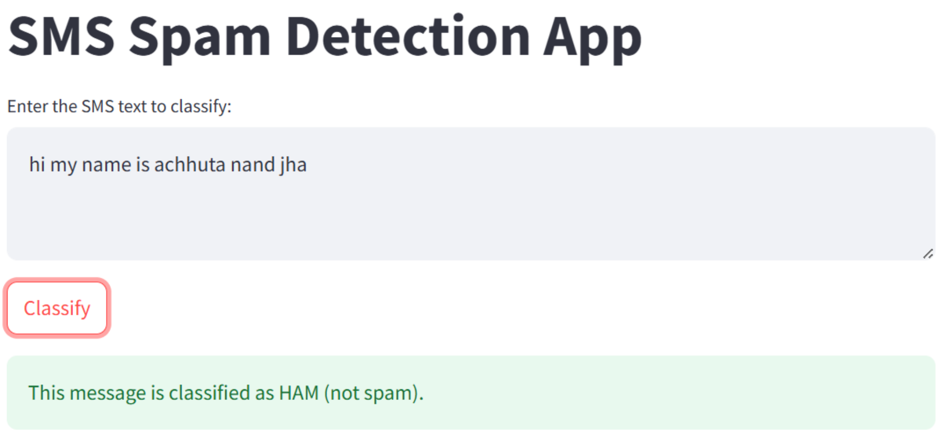
Key Metrics:

- **True Positives (TP):** 129 (instances correctly predicted as class 1)
- **True Negatives (TN):** 964 (instances correctly predicted as class 0)
- **False Positives (FP):** 1 (instances incorrectly predicted as class 1)
- **False Negatives (FN):** 21 (instances incorrectly predicted as class 0)

Based on these values, you can calculate other performance metrics like:

- **Accuracy:** (TP + TN) / (TP + TN + FP + FN)
- **Precision:** TP / (TP + FP)

- **Recall (Sensitivity):** TP / (TP + FN)
- **F1-score:** 2 * (Precision * Recall) / (Precision + Recall)



SMS Spam Detection App

Enter the SMS text to classify:

hi my name is achhuta nand jha

This message is classified as HAM (not spam).

- A user-friendly UI was developed to detect whether a given message is Spam or Ham.
- The system highlights the detected message type, displaying Spam in red and Ham in green.
- A "Classify" button was implemented to run the model and classify the message.
- Once the mail type is detected, a voice notification is triggered, automatically announcing the result.

4.2 GitHub Link for Code:

[Github link]

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

5.1 Future Work:

Looking ahead, one of the primary improvements planned for the email classification system is the integration of a feedback mechanism within the user interface. This feature will empower users to engage directly with the system, offering valuable insights into the accuracy and relevance of the classification results. The feedback loop will be crucial for continually refining the model, ensuring it adapts to the changing landscape of email communication.

The feedback form will be designed for simplicity and ease of use, allowing users to rate the classification of each email. For instance, users may be prompted to confirm if an email has been correctly categorized as "spam," "important," "promotional," or other predefined categories. In cases of misclassification, users will have the option to flag the email, either suggesting a different category or identifying it as wrongly classified as spam. This interactive feature will allow real-time corrections, enabling immediate adjustments based on user preferences.

The integration of this feedback mechanism offers several key benefits:

1. **Enhanced Model Accuracy:** By gathering real-time feedback, the model can be continuously updated and refined based on user input, helping to address mistakes

and misclassifications. For example, if newsletters or transactional emails are frequently misclassified, user feedback will highlight these errors, allowing the model to improve over time. This leads to better classification accuracy, reducing false positives (e.g., important emails marked as spam) and false negatives (e.g., spam emails slipping through the filter).

2. **Personalized Email Categorization:** Users often have unique preferences for organizing their emails. Some may consider certain promotional emails as important, while others may prefer to ignore them. The feedback system will enable the model to learn and adapt to individual user preferences, personalizing email categorization. For example, if a user consistently reclassifies promotional emails as "important," the system will adjust to reflect that preference, enhancing relevance and user satisfaction.
3. **Continuous Learning:** The feedback form will facilitate ongoing learning, preventing the model from becoming stagnant or outdated. As email communication evolves and new types of emails (e.g., advanced phishing attempts or innovative promotional strategies) emerge, the model will be better equipped to adapt. Feedback will provide real-world examples that help train the model to recognize and classify new types of emails, strengthening the system's resilience in dynamic environments.
4. **Insights for Model Refinement:** Feedback data will offer valuable insights into areas where the model struggles, helping to identify specific types of emails it has difficulty classifying. For instance, if legitimate emails are repeatedly flagged as spam, or urgent emails are miscategorized, the feedback can be analyzed to uncover the root cause of these errors. This will guide developers in fine-tuning algorithms or modifying feature extraction processes to address these challenges.
5. **User Empowerment and Satisfaction:** Allowing users to directly influence the system's performance through feedback not only improves the classifier but also boosts user satisfaction. It gives users a sense of control over their email experience, ensuring the system better aligns with their needs. The feedback process, integral to the system, will also foster higher engagement and trust in the system's accuracy and reliability.
6. **Long-Term Model Optimization:** As user feedback accumulates over time, it will provide a wealth of data that can be used to periodically retrain the model. This will enable the application of advanced machine learning techniques, such as active learning, where the system flags uncertain predictions and specifically requests feedback for those cases. This iterative process will ensure the model continues to evolve and improve in precision over time.
7. **Adapting to New Email Types and Phishing Detection:** With the increasing sophistication of spam, phishing, and promotional emails, the feedback loop will play a critical role in identifying previously unseen forms of malicious or unwanted content. Users will be able to flag suspicious emails, enabling the system to recognize and categorize new phishing attacks or spam patterns, thereby enhancing its resilience to emerging threats.

5.2 Conclusion:

In conclusion, this spam classifier project has made a significant impact on enhancing the detection and prevention of unwanted, malicious, or unsolicited messages within digital communication systems. By creating a machine learning model that effectively differentiates between spam and non-spam content, the project highlights the substantial potential of automating and optimizing email filtering and messaging systems.

Trained on diverse datasets, the classifier has demonstrated strong accuracy and efficiency in recognizing spam patterns, such as common keywords, message structure, and other textual features. The application of various machine learning algorithms (such as Naive Bayes, Support Vector Machines, and Deep Learning) further emphasizes the value of leveraging advanced techniques to tackle real-world problems, surpassing traditional heuristic-based methods.

This project's impact goes beyond simply developing a functional spam filter. It provides valuable insights into the complexities of managing large-scale text data, the importance of feature engineering, and the necessity of continuously updating the model to keep pace with evolving spamming techniques. Additionally, it highlights the increasing role of artificial intelligence in bolstering digital security and improving user experience.

By automating spam message classification, the project not only boosts user productivity and safety but also sets the stage for future advancements in areas such as real-time detection, adaptation to emerging spam tactics, and cross-platform filtering.

Overall, the project effectively demonstrates the power of machine learning in combating spam, offering a dependable tool for application across various communication platforms. Moreover, it paves the way for future improvements to refine the classifier's accuracy, respond to new spam trends, and integrate with larger cybersecurity systems for broader applications

REFERENCES

. Campus X an YouTube channel helps for building this project , and most of the analyzing was inspired from them. link