Email Spam Detection

Mohamed Ilham
ilhmask360@gmail.com

***Abstract*— Email spam classification is a critical task in today's digital world, where the amount of spam emails has increased dramatically. In this project, we propose to use machine learning (ML) and natural language processing (NLP) techniques to classify email messages as either spam or legitimate. The dataset will consist of a large number of email messages with their corresponding labels (spam/ham. We will use the Naive Bayes algorithm to develop the spam classifier. The project's outcome will be an efficient spam classifier that can accurately identify and filter spam emails, improving email security and productivity**.

Keywords—Email, Spam/ham, Machine Learning, Natural Language Processing

I. INTRODUCTION

In today's digital age, email continues to be one of the most crucial communication tools for both personal and professional purposes. Despite the advent of various communication platforms, email remains a primary medium for exchanging information, sharing documents, and conducting business transactions. However, the widespread use of email has also led to the proliferation of spam emails—unsolicited and often malicious messages that pose significant challenges to both users and email service providers. These spam emails clutter inboxes, consume valuable time, and serve as vectors for phishing attacks and malware distribution. As a result, they present substantial financial and data security threats, undermining the overall efficiency and safety of email systems.

The growing prevalence of spam emails necessitates the development of effective solutions to combat this issue and protect users from its detrimental effects. This project is centered on creating a robust email spam classification system by leveraging the power of machine learning (ML) and natural language processing (NLP) techniques. The primary goal is to develop a model capable of accurately distinguishing between spam and legitimate (ham) emails, thereby enhancing the user experience and fortifying defenses against potential threats.

To achieve this, our proposed solution involves a comprehensive and systematic approach to text data preprocessing. The process begins with tokenization, which involves breaking down the email text into individual tokens or words. This is followed by stop word removal, where common words that do not contribute significant meaning to the text, such as "and," "the," and "is," are filtered out. Next, stemming is applied to reduce words to their root forms, enabling the model to recognize different variations of the same word. Feature extraction is then conducted to transform the preprocessed text into meaningful numerical representations that can be utilized by ML algorithms.

The Naive Bayes classifier is the chosen algorithm for this project due to its simplicity and effectiveness in text classification tasks. Naive Bayes is a probabilistic classifier based on Bayes' theorem, which assumes that the presence of a particular feature in an email is independent of the presence of any other feature. Despite this naive assumption, the Naive Bayes classifier has proven to be highly effective for spam detection, making it an ideal choice for our model. By training the classifier on a labeled dataset of spam and ham emails, we aim to develop a model that can accurately classify new, unseen emails.

The significance of this project extends beyond the creation of an accurate spam classifier. It contributes to the broader efforts of enhancing email security and productivity. An effective spam filtering system can significantly reduce the volume of unwanted emails, allowing users to focus on their important communications without the distraction and risks posed by spam. This not only improves individual productivity but also enhances organizational efficiency and data security.

In summary, this project seeks to address the pressing issue of spam emails through the development of a robust spam classification system utilizing ML and NLP techniques. By transforming raw email content into meaningful representations and employing the Naive Bayes classifier, we aim to create an accurate and reliable spam detection model. Ultimately, our efforts will contribute to improving email security, safeguarding user data, and enhancing the overall email experience for users worldwide.

II. RELATED WORK

In In this section, we review several notable works in this area, highlighting their methodologies, findings, and limitations.
Efficient Spam Email Classification using Machine Learning Algorithms (Pallavi N & Jayarekha, 2023): Pallavi N and Jayarekha presents a study on email spam detection utilizing various machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM) and Decision Trees. Overall, the paper offers a comprehensive analysis of ML-based email spam classification. Challenges may arise from concentrating on a particular subset of machine learning algorithms, which could restrict its relevance to broader contexts[1]

Spam Detection System Using Supervised ML(Abhila & Delphin, 2021): Abhila and Delphin present a proposed spam detection system which utilizes the Naive Bayes method, a mining approach, for identifying spam and ham messages in an inbox. The system employs a series of steps, including data collection, pre-processing, feature extraction, training, and testing, to classify messages accurately. Overall, while Naïve Bayes classifiers offer simplicity and ease of implementation, they may exhibit limitations in terms of model complexity, feature handling, adaptability to evolving spamming techniques.[2]

A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms (Mansoor & Muhana, 2021): Mansoor and Muhana summarize and emphasize the importance of machine learning algorithms in enhancing email spam classification. They outline the challenges associated with traditional rule-based filtering methods and the potential of machine learning algorithms to address these challenges. The authors highlight the need for further research to develop more robust and adaptive spam detection systems capable of mitigating evolving spamming tactics.[3]

ML Approaches to Detect Email Spam Anomaly (Narendra Kumar, 2022): Narendra Kumar emphasizes the importance of machine learning in email spam detection. It involves preprocessing the data along with removing duplicates and punctuation. The Naive Bayes classifier is used for its accuracy and efficiency, especially in pattern matching with regular expressions. Advantages of the proposed model include its ability to predict spam using classifiers based on email content rather than domain names or other criteria. It addresses the challenge of testing emails with a restricted corpus by leveraging machine learning techniques for effective spam filtering.[4]

III. Top Level Architecture

This image is a simplified diagram of the steps involved in using machine learning to Classify spam emails from legitimate ones and minimizing the impact of spam.

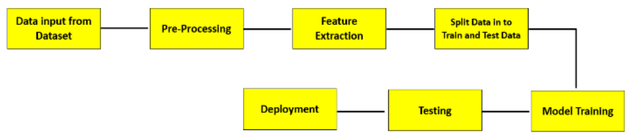


Figure 1Top Level Architecture

A. Data Collection

Data plays a pivotal role in the domains of prediction and classification, with the accuracy of models often directly correlating to the volume and quality of the data used. In this project, I have sourced the dataset from Kaggle, a renowned platform for data science and machine learning resources. This particular dataset is well-suited for our spam classification task as it contains 5,200 rows, providing a substantial amount of data to train and test our model. Each row consists of two columns: 'Label' and 'Text'. The 'Label' column indicates whether an email is 'ham' (legitimate) or 'spam', while the 'Text' column contains the actual email content. By leveraging this dataset, we can build a robust machine learning model that accurately distinguishes between spam and legitimate emails, thereby enhancing the overall effectiveness of our spam detection system.

B. Data Analysis

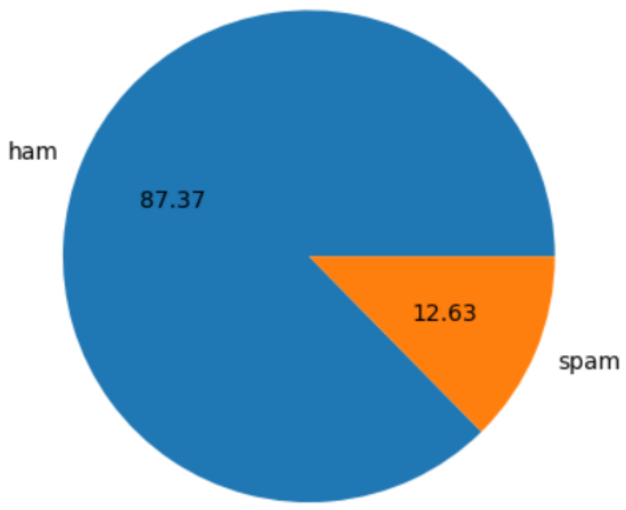


Figure 2Dataset overview

We have 4516 ham and 653 spam. 87.95% of the data are ham emails and just 12.36% is spam so the problem is present here is that the model will learn with ham much more than spam

C. Data Pre-Processing

Data preprocessing is a critical step in preparing raw text data for machine learning tasks such as email spam detection. Effective preprocessing enhances the performance of models by ensuring the input data is clean and consistent. This document outlines the methodology for text cleaning, stopword removal, and stemming.

- Text Cleaning: Text cleaning involves several sub-steps to ensure the data is uniform and devoid of noise. First, the text is converted to lowercase to avoid the model differentiating between uppercase and lowercase characters. URLs, which often introduce irrelevant information, are removed next. Punctuation is stripped away as it generally does not contribute to the semantic content of the text and can interfere with tokenization. Newline characters are also removed to streamline the text. Digits are excluded as they are often irrelevant for spam detection. Multiple spaces are normalized to single spaces to maintain the structure of the text. Finally, single characters, which are usually residuals from punctuation removal, are eliminated as they typically do not add meaningful information to the analysis.

- Stopword Removal: Stopwords are common words such as "and", "the", and "is" that do not contribute significant meaning to the text. Removing these words helps reduce noise and the dimensionality of the data, allowing the model to focus on more relevant terms that contribute to the task at hand.

- Stemming: Stemming reduces words to their base or root form, for example, transforming "running" to "run". This process helps in minimizing the dimensionality of the text data by grouping different forms of the same word together, thus simplifying the analysis and improving the efficiency of the model.

D. *Feature Extraction*

Feature extraction is a fundamental step in text processing, transforming raw text data into numerical features that can be used by machine learning models. In the context of email spam detection, we use the Bag of Words (BOW) model with CountVectorizer to convert text into a matrix of token counts. This document outlines the methodology for feature extraction using BOW.

Bag of Words (BOW) Model: The Bag of Words (BOW) model is a popular method for text representation in natural language processing. It involves creating a vocabulary of all the unique words present in the text data and representing each document as a vector of word occurrence counts. CountVectorizer, a tool provided by the scikit-learn library in Python, facilitates this process by converting a collection of text documents into a matrix of token counts. This matrix can then be used as input features for machine learning algorithms.

E. *Split Data in to Train and Test Data*

The data splitting process involves defining the feature matrix and target labels from the preprocessed dataset. The feature matrix is derived from the previously created document-term matrix, and the target labels are extracted from the 'label' column of the dataset. To achieve an effective division of the data, we utilize the train_test_split function from the scikit-learn library, which randomly splits the data into training and testing sets based on specified parameters.

In this methodology, we allocate 10% of the data for testing by specifying an appropriate test size. The random state parameter is set to a fixed value to ensure reproducibility, controlling the shuffling process so that the same train-test split is obtained every time the code is run. The result is four subsets: the feature matrix and target labels for both the training set and the testing set.

F. *Model training*

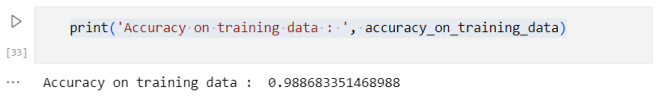
Model training and evaluation are critical stages in developing a machine learning model. In this email spam detection project, we employ a Naive Bayes classifier due to its simplicity and effectiveness for text classification tasks. This document outlines the methodology for training the Naive Bayes model, evaluating its performance on both training and test data, and interpreting the results using various metrics.

- Model Selection: The Naive Bayes classifier was chosen for this email spam detection project due to several key reasons. First, Naive Bayes is particularly effective for text classification tasks, as it assumes independence between features, which is a reasonable approximation for word occurrences in text. It is computationally efficient, making it suitable for large datasets. Additionally, Naive Bayes has been proven to perform well in spam detection tasks, offering robust performance with relatively simple implementation. Its probabilistic nature also provides clear insights into the decision-making process, allowing for straightforward interpretation and analysis.

- Model Training: To begin with, we initialize the Naive Bayes model using the MultinomialNB class from the scikit-learn library. The Multinomial Naive Bayes classifier is particularly suitable for discrete features such as word counts in text classification. Once the model is initialized, we train it using the training dataset. The fitting process involves feeding the feature matrix and corresponding labels from the training set into the model. This enables the model to learn the relationship between the features and the target labels.

G. *Model Evaluation*

Accuracy on Training Data: After training the model, we evaluate its performance on the training data by making predictions and calculating the accuracy. The accuracy is the proportion of correctly predicted instances out of the total instances in the training set. This metric helps assess how well the model has learned from the training data. Accuracy on training data: 98%

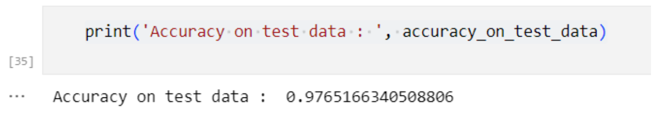


```
[33] print('Accuracy on training data : ', accuracy_on_training_data)
```

```
Accuracy on training data :  0.988683351468988
```

Figure 3Accuracy on Training Data

Accuracy on Test Data: To evaluate the model's generalization ability, we make predictions on the test data and calculate the accuracy. The test accuracy provides an estimate of how well the model performs on unseen data, indicating its effectiveness in real-world scenarios. Accuracy on test data: 97%



```
[35] print('Accuracy on test data : ', accuracy_on_test_data)
```

```
Accuracy on test data :  0.9765166340508806
```

Figure 4Accuracy on test data

Confusion Matrix: To gain deeper insights into the model's performance, we compute the confusion matrix. The confusion matrix is a table that summarizes the number of true positives, true negatives, false positives, and false negatives. By plotting the confusion matrix as a heatmap, we can visualize the model's prediction accuracy across different classes (spam and ham in this case).

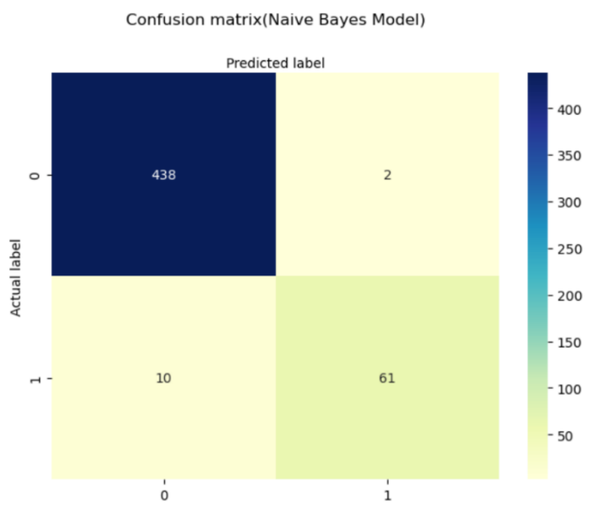
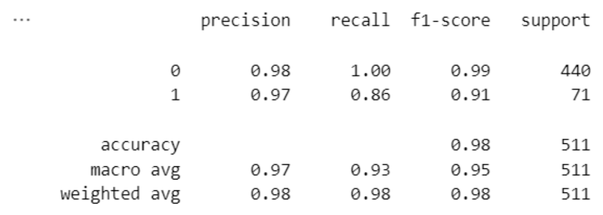


Figure 5confusion matrix for Naive Bayes

Classification Report: The classification report provides additional performance metrics, including precision, recall, and F1 score. Precision indicates the proportion of positive predictions that are correct, recall measures the proportion of actual positive cases that are correctly identified, and the F1 score is the harmonic mean of precision and recall. These metrics offer a comprehensive evaluation of the model's performance.

- Precision — What percent of your predictions were correct? (Precision = TP/(TP + FP))
- Recall — What percent of the positive cases did you catch? (Recall = TP/(TP+FN))
- F1 score — What percent of positive predictions were correct? (F1 Score = 2*(Recall * Precision) / (Recall + Precision))



```
...
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       440
           1       0.97      0.86      0.91        71

    accuracy                           0.98       511
   macro avg       0.97      0.93      0.95       511
weighted avg       0.98      0.98      0.98       511
```

Figure 6Model classification report

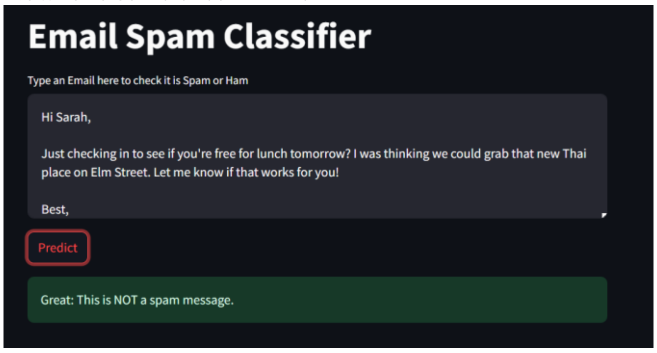
H. Deploymont

deploying an email spam detection model using Streamlit, a popular open-source framework for creating interactive web applications. The model utilizes machine learning (ML) techniques to classify emails as either spam or ham (non-spam). The deployment process involves loading the pre-trained model and vectorizer, creating a user interface, and integrating the prediction functionality.

IV. Outcome

Streamlit Interface

Now let's some check ham email



Email Spam Classifier

Type an Email here to check it is Spam or Ham

Hi Sarah,

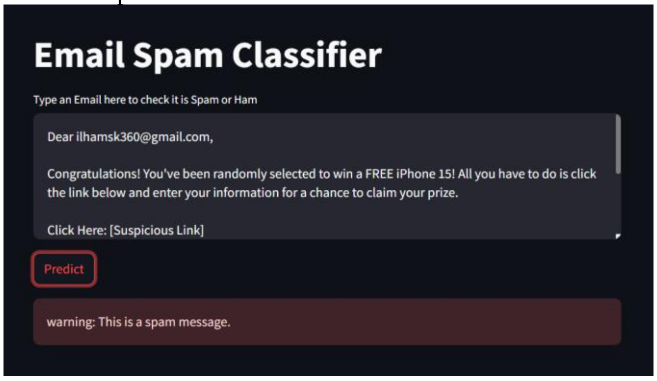
Just checking in to see if you're free for lunch tomorrow? I was thinking we could grab that new Thai place on Elm Street. Let me know if that works for you!

Best,

Great: This is NOT a spam message.

Figure 7 ham mail outcome

If mail is spam its show



Email Spam Classifier

Type an Email here to check it is Spam or Ham

Dear ilhamsk360@gmail.com,

Congratulations! You've been randomly selected to win a FREE iPhone 15! All you have to do is click the link below and enter your information for a chance to claim your prize.

Click Here: [Suspicious Link]

warning: This is a spam message.

Figure 8Spam mail outcome

V. Limitations of the Project

This project has certain limitation

- Data collection (Imbalanced Dataset)
- This can only predict and classify spam but not block it.
- Analysis can be tricky for some alphanumeric messages and it may struggle with entity detection.

VI. Descussion

The email spam detection project successfully met its aim and objectives, achieving a high accuracy rate of 98%. The project demonstrated the efficacy of the Naive Bayes classifier, supported by rigorous data preprocessing and feature extraction techniques. The successful deployment using Streamlit provided an interactive tool for practical spam detection. The insights gained and the methodology employed in this project set a solid foundation for future enhancements and applications in the domain of email spam detection.

References

[1] P. N and P. Jayarekha, "Efficient Spam Email Classification Using Machine Learning Algorithms," 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 2023.

[2] T. Toma, S. Hassan and M. Arifuzzaman, "An Analysis of Supervised Machine Learning Algorithms for Spam Email Detection," 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021.

[3] A. B, D. P. M, K. M, M. N. Joseph and D. R, "Spam Detection System Using Supervised ML," 2021 International Conferenceon System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 2021.

[4] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti and M. Alazab, "A Comprehensive Survey for Intelligent Spam Email Detection," in IEEE Access, vol. 7, pp. 168261-168295, 2019.

[5] W. Hijawi, H. Faris, J. Alqatawna, A. M. Al-Zoubi and I. Aljarah, "Improving email spam detection using content based feature engineering approach," 2017.

[6] A. Reddy et al., "Email Spam Detection Using Machine Learning," International Journal of Fisheries and Aquatic Research, vol. 10, no. 1, pp. 2658-2664, 2023.

[7] D. E. G. Bassi, J. S. Chiroma, H. A. Shafi'i, M. A. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," Heliyon, vol. 4, no. 6, p. e00625, 2018.

[8] C. Panem et al., "Machine-Learning-Based Spam Mail Detector," SN Computer Science, vol. 4, no. 8, pp. 1-16, 2023.

[9] R. Alanazi, "Analysis of e-Mail Spam Detection Using a Novel Machine Learning-Based Hybrid Bagging Technique," Computational Intelligence and Neuroscience, vol. 2022, Article ID 9381222, 2022.