

**EMAIL SPAM AND MALWARE FILTERING**

**21AI702 APPLICATIONS OF MACHINE LEARNING**

**REPORT**

**BY**

**B.C.SAI VYSHNAVI**

**CB.SC.P2AIE23002**

**MASTER OF TECHNOLOGY**

**IN**

**ARTIFICIAL INTELLIGENCE**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AMRITA SCHOOL OF COMPUTING**

**AMRITA VISHWA VIDYAPEETHAM**

**COIMBATORE- 641 112**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **TABLE OF CONTENTS** | **PAGE NO.** |
| **1.** | **ABSTRACT** | **3** |
| **2.** | **INTRODUCTION** | **4** |
| **3.** | **PROBLEM STATEMENT** | **5** |
| **4.** | **MOTIVATION** | **6** |
| **5.** | **OBJECTIVE** | **7** |
| **6.** | **LITERATURE SURVEY** | **8** |
| **7.** | **SYSTEM ARCHITECTURE** | **12** |
| **8.** | **DATASET DESCRIPTION & PREPROCESSING** | **15** |
| **9.** | **ALGORITHM** | **20** |
| **10.** | **IMPLEMENTATION** | **23** |
| **11.** | **RESULTS** | **35** |
| **12.** | **CONCLUSION** | **43** |
| **13.** | **REFERENCES** | **44** |

**ABSTRACT**

Email spam and malware pose significant threats to digital communication and cybersecurity. Effective filtering mechanisms are essential to safeguard users from these malicious activities. This study explores the implementation of Naive Bayes and Support Vector Machine (SVM) algorithms for email spam and malware detection. The Naive Bayes algorithm, with its probabilistic approach, efficiently handles the uncertainty and variability in spam characteristics. SVM, a robust classification technique, excels in high-dimensional spaces and complex boundary delineation. By integrating both methods, we aim to enhance detection accuracy and reduce false positives. Our experimental results, derived from diverse email datasets, demonstrate that the hybrid model leverages the strengths of both algorithms, offering improved performance compared to standalone implementations. This hybrid approach provides a reliable, scalable solution for real-time email filtering, contributing to a more secure digital communication environment.

**INTRODUCTION**

Email is a key way people and organizations communicate, but it often gets flooded with spam and malware. Spam emails can include annoying advertisements, scams, and phishing attempts, while malware can harm your device and steal information. This makes it crucial to have effective ways to filter out these unwanted and harmful emails.

Machine learning offers powerful tools to tackle this problem. Two popular algorithms are Naive Bayes and Support Vector Machine (SVM). Naive Bayes is good at quickly and efficiently identifying spam because it uses probabilities to make decisions. SVM is excellent at distinguishing between spam and legitimate emails, even when the data is complex.

This study investigates the application of Naive Bayes and SVM algorithms for filtering email spam and malware. By leveraging the probabilistic reasoning of Naive Bayes and the robust classification capabilities of SVM, we aim to develop a hybrid model that enhances detection accuracy and minimizes false positives. Through comprehensive experimentation on diverse email datasets, we evaluate the performance of these algorithms both individually and in combination.

The findings of this research contribute to the field of cybersecurity by providing an effective, scalable solution for real-time email filtering. By integrating Naive Bayes and SVM, It offers a sophisticated approach that adapts to evolving threats, ensuring a safer and more reliable email communication environment.

**PROBLEM STATEMENT**

The increasing prevalence of spam and malware in email communication poses significant risks to users and organizations, including data breaches, financial loss, and compromised security. Traditional email filtering methods struggle to keep pace with the evolving tactics used by spammers and cybercriminals, leading to inadequate protection and high rates of false positives and negatives. There is a pressing need for more effective, adaptive filtering mechanisms that can accurately distinguish between legitimate and malicious emails.

This study aims to address the problem of email spam and malware detection by investigating the combined use of Naive Bayes and Support Vector Machine (SVM) algorithms. Naive Bayes offers simplicity and efficiency in handling text-based data, while SVM provides robust classification capabilities in high-dimensional spaces. By integrating these two methods, we seek to enhance the accuracy and reliability of email filtering systems, thereby reducing the incidence of spam and malware reaching end users.

The specific problem to be addressed is the integration of Naive Bayes and SVM algorithms improve the detection and filtering of email spam and malware, compared to their standalone implementations.The goal is to develop a hybrid model that leverages the strengths of both algorithms to deliver superior performance in real-time email filtering applications.

**MOTIVATION**

The problem of email spam and malware poses significant challenges, necessitating effective filtering systems to ensure secure communication. Naive Bayes (NB) classifiers offer a simple yet effective solution, excelling in text classification by leveraging probabilistic principles to handle large vocabularies and provide real-time, accurate spam detection. Their ability to adapt to new data ensures ongoing relevance. Support Vector Machines (SVMs) enhance this further by handling high-dimensional data and excelling in binary classification, achieving high precision and low false positives. SVMs' robustness and adaptability to evolving threats make them ideal for detecting sophisticated spam and malware, providing a powerful, complementary approach to email filtering.

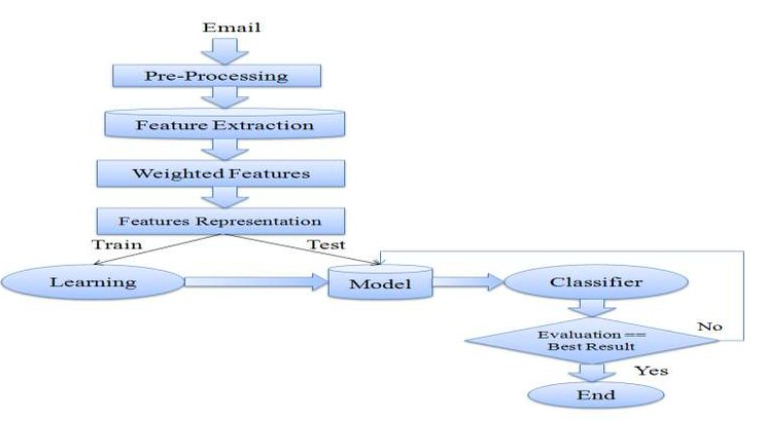
**OBJECTIVE**

The objective of email spam and malware filtering is to create a robust, adaptive, and efficient system that accurately identifies harmful emails while minimizing false positives. This system aims to enhance detection accuracy, ensure adaptability to evolving threats, and maintain computational efficiency for real-time processing. It should be scalable to handle increasing email volumes, user-friendly with easy configuration, and seamlessly integrable with existing email infrastructure. By combining machine learning approaches like Naive Bayes and Support Vector Machines, the system will provide comprehensive protection, improving email security and reliability.

**LITERATURE REVIEW**

|  |  |  |
| --- | --- | --- |
| NAME | AUTHOR | SUMMARY |
| **Machine learning for email spam filtering: review, approaches and open research problems.** | Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). | In this paper explores various algorithms such as decision trees, random forests, and deep learning models. The authors provide a critical analysis of each method's performance and identify unresolved issues and potential research directions, emphasizing the need for more robust, adaptive, and scalable spam filters. |
| **Applicability of machine learning in spam and phishing email filtering: review and approaches.** | Gangavarapu, T., Jaidhar, C.D. &notepad Chanduka, B.(2020). | In this paper The authors conduct a thorough review of machine learning techniques used for filtering spam and phishing emails. They explore various algorithms and methodologies, highlighting their applicability and effectiveness in combating email threats. This review offers insights into the state-of-the-art approaches and identifies future research directions to improve the detection and prevention of spam and phishing attacks. |
| **A Spam Email Detection Mechanism for English Language Text Emails Using Deep Learning Approach,** | S. Kaddoura, O. Alfandi and N. Dahmani,(2020). | In this paper the author focused on English language text emails, the authors leverage deep learning models to effectively classify emails as spam or legitimate. Their approach contributes to the development of more sophisticated and accurate spam filtering systems, addressing the evolving nature of email-based threats. |
| **An anti-spam detection model for emails of multi-natural language.** | Mohammed, Mazin Abed, Salama A. Mostafa, Omar Ibrahim Obaid, Subhi RM Zeebaree, Mohd Khanapi Abd Ghani, Aida Mustapha, Mohd Farhan Md Fudzee et al.(2019). | In this paper the authors propose an anti-spam detection model tailored for emails written in multiple natural languages. Their model aims to effectively identify and filter spam emails across diverse linguistic contexts, enhancing email security for users worldwide. This research contributes to the development of language-agnostic spam detection solutions, addressing the challenges posed by multilingual communication in email systems. |
| **PCSF: privacy-preserving content-based spam filter** | Kim, Intae, Willy Susilo, Joonsang Baek, Jongkil Kim, and Yang-Wai Chow,(2023). | In this paper the work introduces PCSF, a privacy-preserving content-based spam filter that aims to maintain user privacy while effectively filtering spam emails. By leveraging cryptographic techniques, PCSF ensures that email content remains confidential during the filtering process, addressing privacy concerns associated with traditional spam filtering approaches. This innovative solution contributes to enhancing email security while safeguarding user privacy in the digital age. |
| **Machine learning techniques for spam detection in email and IoT platforms: analysis and research challenges.** | Ahmed, Naeem, Rashid Amin, Hamza Aldabbas, Deepika Koundal, Bader Alouffi, and Tariq Shah,(2022). | In this paper investigates machine learning techniques for spam detection across email and Internet of Things (IoT) platforms, analyzing their effectiveness and identifying research challenges. By exploring the application of machine learning in diverse environments, including IoT, the authors shed light on the evolving landscape of spam detection and highlight avenues for future research to address emerging threats in email and IoT ecosystems. |
| **Efficient clustering of emails into spam and ham: The foundational study of a comprehensive unsupervised framework.** | Karim, Asif, Sami Azam, Bharanidharan Shanmugam, and Krishnan Kannoorpatti, (2020). | In this paper the author proposes an efficient unsupervised framework for clustering emails into spam and ham categories. By leveraging clustering techniques, their framework facilitates the automatic categorization of emails without the need for labeled training data. This foundational study lays the groundwork for developing unsupervised spam filtering systems, offering a promising approach to email security that minimizes manual intervention and improves scalability. |
| **A comparative approach to Naïve Bayes classifier and support vector machine for email spam classification.** | Ma, Thae Ma, Kunihito Yamamori, and Aye Thida,(2020). | Their study compares the performance of Naïve Bayes and Support Vector Machine (SVM) classifiers for email spam classification. By conducting a comparative analysis, the authors provide insights into the strengths and weaknesses of each approach, helping researchers and practitioners make informed decisions when selecting classification algorithms for email spam detection. |
| **Evaluation of machine learning techniques for email spam classification.** | Jazzar, Mahmoud, Rasheed F. Yousef, and Derar Eleyan,(2021). | Their research assesses the effectiveness of various machine learning techniques in classifying email spam, providing valuable insights into the performance of different algorithms. By conducting an empirical evaluation, the authors contribute to advancing the understanding of machine learning-based spam detection methods, informing the development of more accurate and efficient email filtering systems. |
| **E-mail spam classification via machine learning and natural language processing.** | Junnarkar, Akash, Siddhant Adhikari, Jainam Fagania, Priya Chimurkar, and Deepak Karia,(*2021).* | Their paper explores the application of machine learning and natural language processing techniques in email spam classification. By combining these methodologies, the authors aim to enhance the accuracy and robustness of spam filtering systems, addressing the complex nature of email-based threats in today's digital communication landscape. |

**SYSTEM ARCHITECTURE**



**Email**

The input to the system is an email that needs to be classified.

**Pre-Processing**

This step involves cleaning and preparing the email data for further processing. Common tasks include removing stop words, normalizing text (e.g., lowercasing), removing punctuation, and handling special characters or formatting issues.

**Feature Extraction**

After pre-processing, the relevant features are extracted from the email. Features can be individual words (unigrams), pairs of words (bigrams), frequencies of certain words, presence of certain keywords, etc. This step transforms the raw text data into a structured format that can be used for model training.

**Weighted Features**

Once features are extracted, weights are assigned to them. This could involve techniques like Term Frequency-Inverse Document Frequency (TF-IDF), which gives higher weights to words that are more informative for distinguishing between classes. This step helps in emphasizing the importance of certain features over others.

**Features Representation**

The weighted features are then represented in a way suitable for model input, typically as vectors. This structured representation allows machine learning algorithms to process the data.

**Learning**

In this step, a machine learning model is trained using the training data. The model learns to map the input features to the correct email classes (e.g., spam or not spam). This involves choosing an appropriate algorithm and optimizing its parameters.

**Model**

The trained model is the output of the learning step. It encapsulates the knowledge gained during training and is ready to classify new emails based on their features.

**Classifier**

The classifier uses the trained model to predict the class of new, unseen emails. It takes the test data (emails that were not used in training) and assigns them to the appropriate category based on the model's predictions.

**Evaluation**

The performance of the classifier is evaluated. This involves measuring metrics such as accuracy, precision, recall, and F1-score on the test data to determine how well the model is performing.

**Evaluation Best Result**

The system checks if the evaluation results meet the desired criteria or the best possible performance. If the evaluation indicates optimal performance, the process ends.

**No/Yes Decision**

If the evaluation does not meet the criteria (No), the process may involve going back to earlier steps to refine the model. This could include adjusting features, re-training the model with different parameters or using different algorithms.

If the evaluation is satisfactory (Yes), the process ends, and the model is considered ready for deployment.

**End**

The final step where the process concludes, and the trained model can be used in a production environment for classifying new emails in real-time.

**DATASET DESCRIPTION AND PREPROCESSING**

The Dataset was downloaded from KAGGLE, SPAM-EMAIL CLASSIFICATION. The dataset consists of a large collection of email messages, each labeled as either "spam" or "ham" (non-spam). These emails come from various sources to ensure a wide range of spam and legitimate email content are represented.Each email is tagged with a label indicating whether it is spam or ham. These labels are used to train and test the algorithm.

* The used dataset is a CSV file.
* It contains 5573 individual emails.
* Each email has classified by Ham or Spam.

**Dataset Statistics**

Total Emails: 5,573

Spam Emails: 3,225

Ham Emails: 2,348

Emails with Malware: 850 (subset of spam emails)

**Training and Test Set Split**

To effectively train and evaluate our model, the dataset is split into a training set and a test set. This split ensures that the model can be trained on a diverse set of examples and then tested on unseen data to evaluate its performance.

**Training Set:**

Total Emails: 3,901 (70% of total)

Spam Emails: 2,258

Ham Emails: 1,643

Emails with Malware: 595

**Test Set:**

Total Emails: 1,672 (30% of total)

Spam Emails: 967

Ham Emails: 705

Emails with Malware: 255

**PREPROCESSING**

Data preprocessing for email spam and malware filtering involves several steps to prepare the raw email data for machine learning model training. Here is a general preprocessing steps:

1.Data Collection

* Gather a large dataset of emails that are labeled as spam and ham. You can collect these emails from various sources or use pre-existing labeled datasets.

2.Data Cleaning

* Remove any irrelevant or noisy data such as HTML tags, special characters, and non-textual content like images or attachments.
* Handle missing values if any, by removing corresponding samples.

3.Tokenization

* Break down the text of each email into smaller units called tokens.This step involves splitting the text into individual words or phrases.

4.Normalization:

* Convert all text to lowercase to ensure consistency in token representation.
* Remove punctuation and other non-alphanumeric characters.

5.Stopword Removal

* Eliminate common words that do not carry significant meaning for classification (example : and , the , is).

6.Stemming or Lemmatization

* Reduce words to their root form to consolidate similar words

7.Feature Extraction:

* TF-IDF (Term Frequency-Inverse Document Frequency): Assign weights to words based on their frequency and importance across all emails.

8.Handling Imbalanced Classes

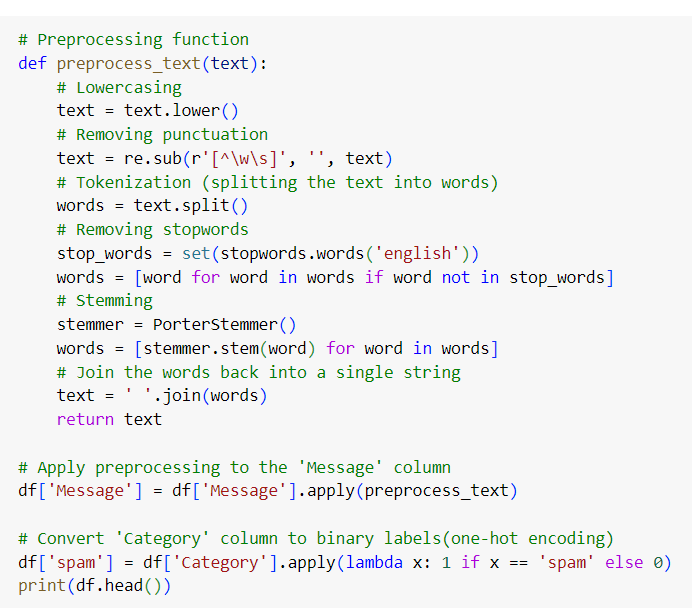
* Address any class imbalances by applying techniques such as oversampling, undersampling, or using algorithms that are robust to class imbalance.

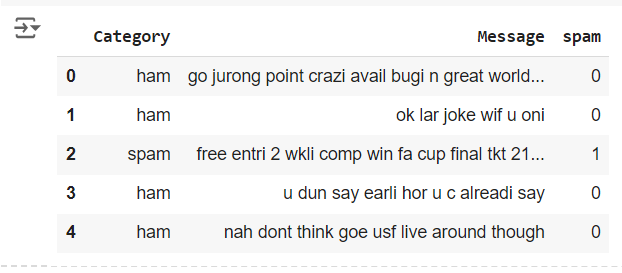
9.Data Splitting

* Split the dataset into training, validation, and test sets to evaluate the performance of the model accurately.

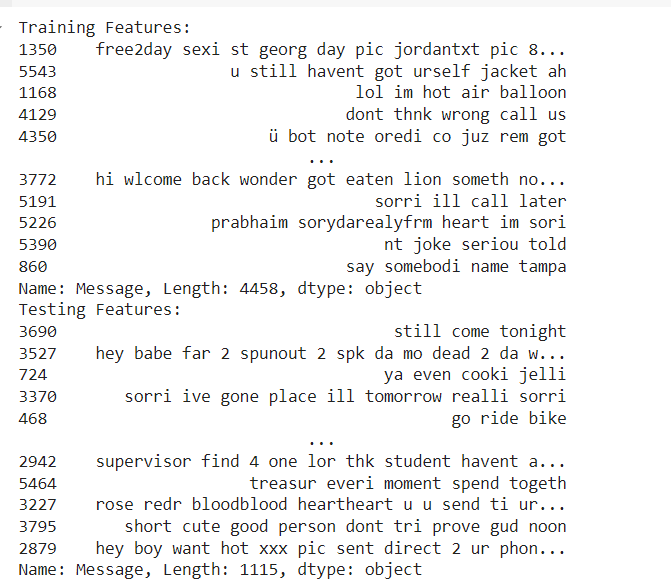
10.Save Preprocessed Data

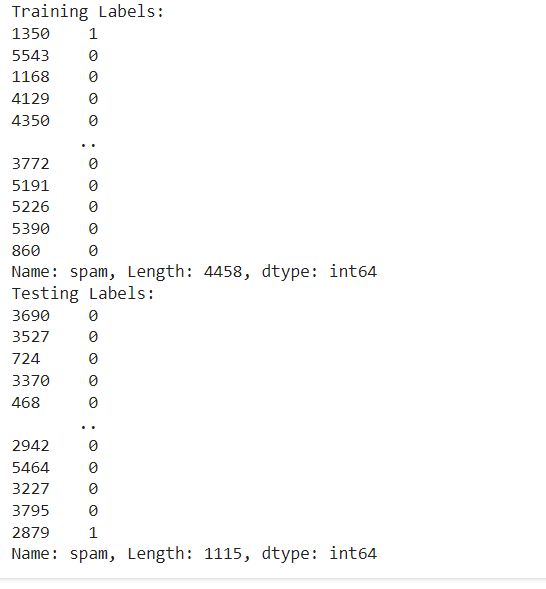
* Save the preprocessed data in a format suitable for model training, such as CSV or numpy arrys, for future use.





**Training and Testing Split**





**ALGORITHMS**

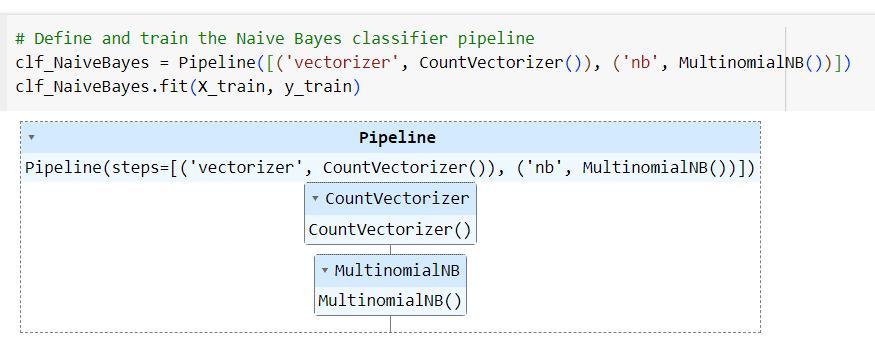
**Naive Bayes**

Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence between predictors. In the context of email spam and malware filtering, it is highly effective due to its simplicity, speed, and robustness with small datasets.

Naive Bayes works by first breaking down each email into features, typically words or tokens extracted from the email's content. During the training phase, the algorithm calculates the probabilities of each word appearing in spam versus non-spam emails using a labeled training dataset. When classifying a new email, the Naive Bayes classifier evaluates the likelihood that the email is spam or non-spam based on the presence and frequencies of these words. The email is then classified as spam or non-spam based on the higher calculated probability, leveraging the probabilistic framework to make efficient and accurate predictions.

Naive Bayes offers several advantages in email spam and malware filtering. Its simplicity makes it easy to implement and interpret, providing straightforward probabilistic insights into classification decisions. The algorithm is also highly efficient, capable of training and making predictions quickly, even with large datasets. Furthermore, Naive Bayes performs well with small to medium-sized datasets and is robust in handling noisy data, maintaining its effectiveness even when the data is imperfect. These qualities make it a valuable tool for quickly and accurately filtering spam and malware in email systems.

* fit(X\_train, y\_train): This method fits the pipeline to the training data. It applies each transformation step to the training data and trains the final estimator (classifier) on the transformed data.



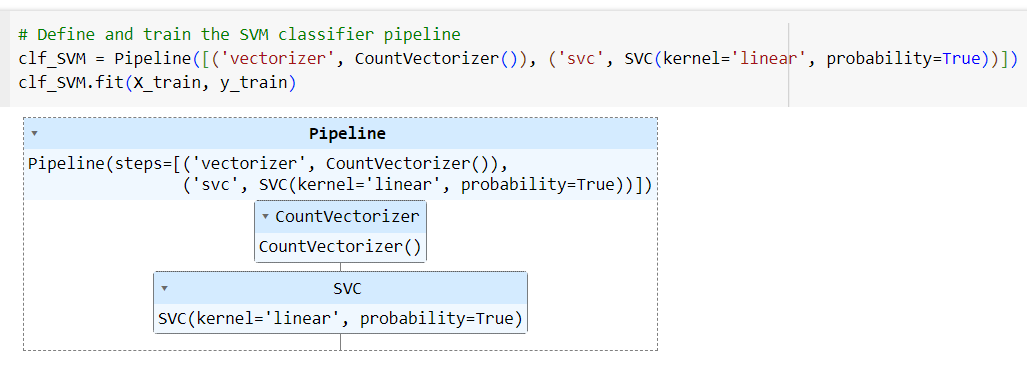
**Support Vector Machine(SVM)**

Support Vector Machine (SVM) is a powerful, supervised machine learning algorithm used for classification and regression tasks. It is particularly known for its effectiveness in high-dimensional spaces and when there is a clear margin of separation between classes.

SVM works by first converting emails into feature vectors, similar to the process used in Naive Bayes. During the training phase, SVM identifies the optimal hyperplane that separates spam and non-spam emails within the feature space, maximizing the margin between the closest points of the two classes, known as support vectors. When classifying a new email, SVM determines which side of the hyperplane the email falls on, thereby classifying it as either spam or non-spam. This approach ensures a clear and effective separation between the two classes, enhancing the accuracy of the classification.

SVM offers several advantages in the context of email spam and malware filtering. It is highly effective in high-dimensional spaces, making it suitable for datasets with many features. Its robustness is a significant benefit, as it performs well with both linear and non-linear data by employing the kernel trick to map data into higher dimensions where it can be more easily separated. Additionally, SVM excels at generalization, meaning it can accurately classify new, unseen data by maintaining a clear margin of separation between classes. These characteristics make SVM a powerful and versatile tool for distinguishing between spam and legitimate emails.

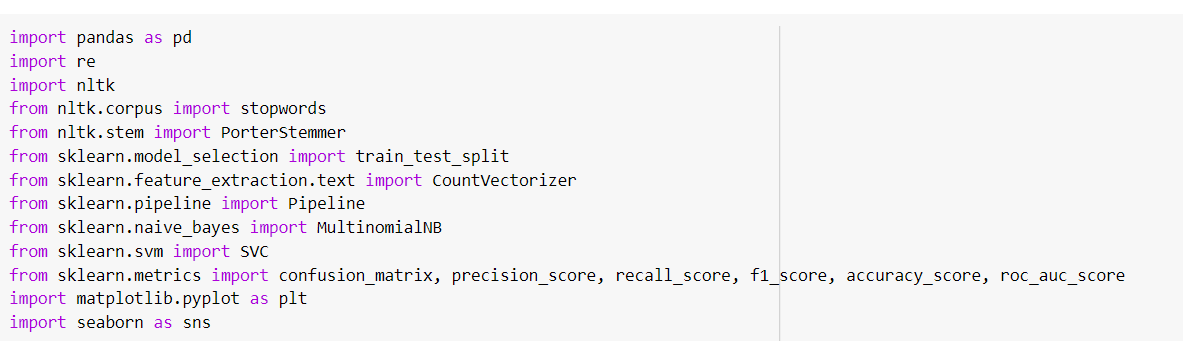
* fit(X\_train, y\_train): This method trains the SVM pipeline on the training data, similar to the Naive Bayes pipeline. It applies vectorization to the text data and trains the SVM classifier on the transformed features.



**IMPLEMENTATION**

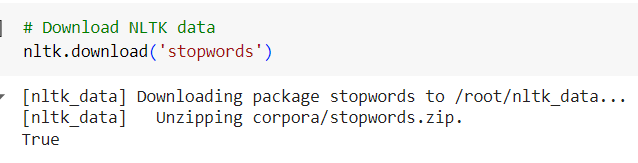
**1.import libraries**

This imports necessary libraries for data manipulation, text preprocessing, machine learning, model evaluation, and visualization.



**2. Downloading NLTK Data**

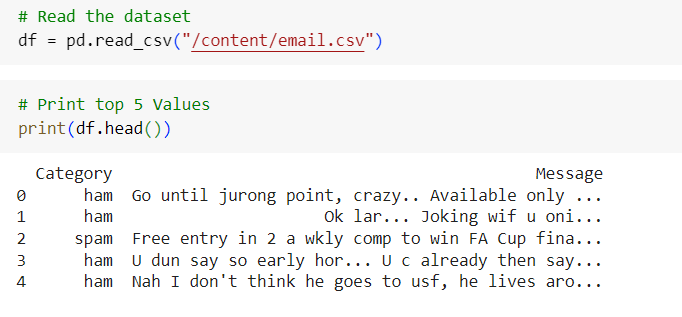
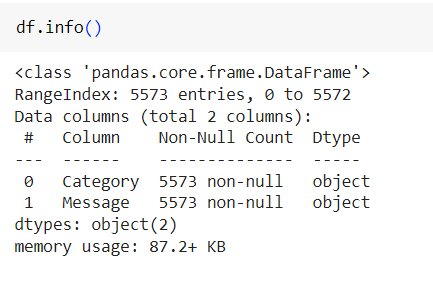
Downloads the stopwords dataset from the NLTK library, which is used to remove common words like "and", "the", etc., from the email texts.



**3. Reading the Dataset and Print top 5 values**

Loads the email dataset from a CSV file and displays the first few records and dataset information.

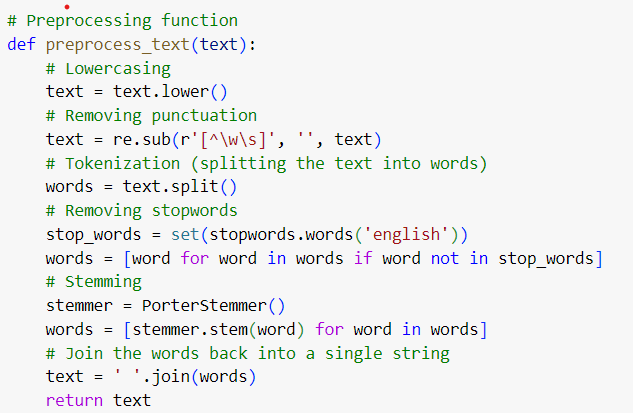
Reads the dataset named "email.csv" into a pandas DataFrame object called df.

**4. Preprocessing Function**

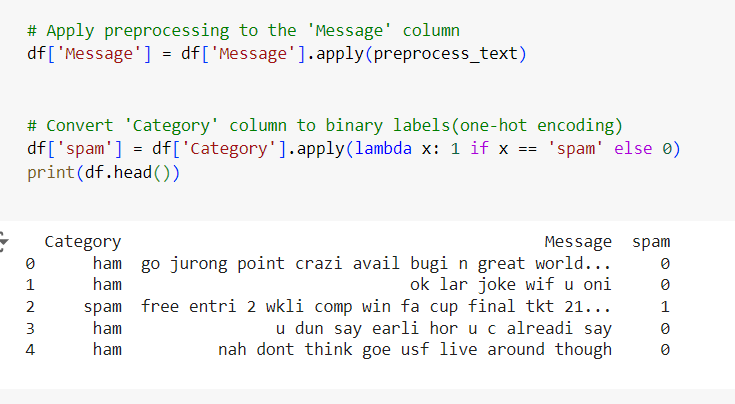
Defines a function to preprocess the email text by:

* Converting to lowercase
* Removing punctuation
* Tokenizing (splitting into words)
* Removing stopwords
* Stemming (reducing words to their root form)



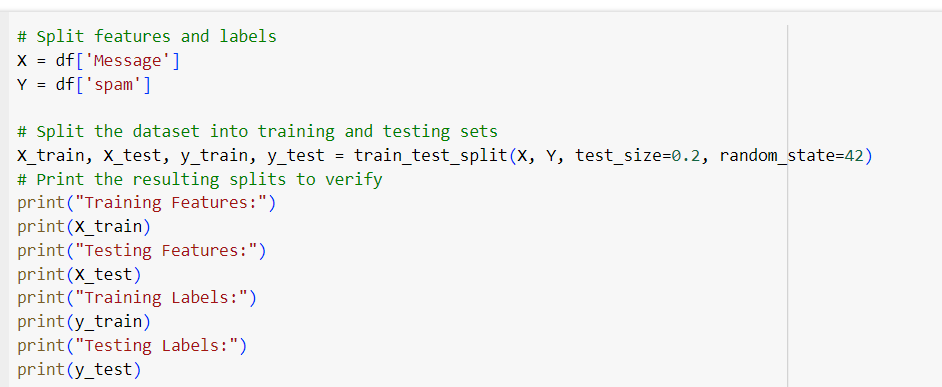
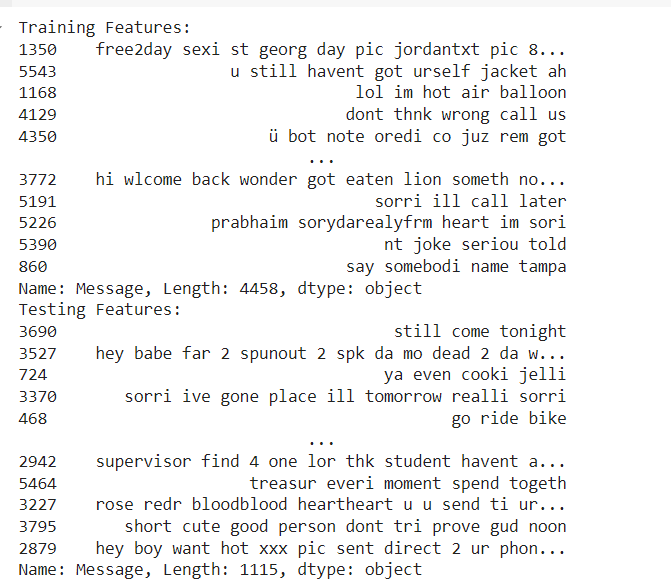
**5. Applying Preprocessing and One-Hot Encoding for Labels**

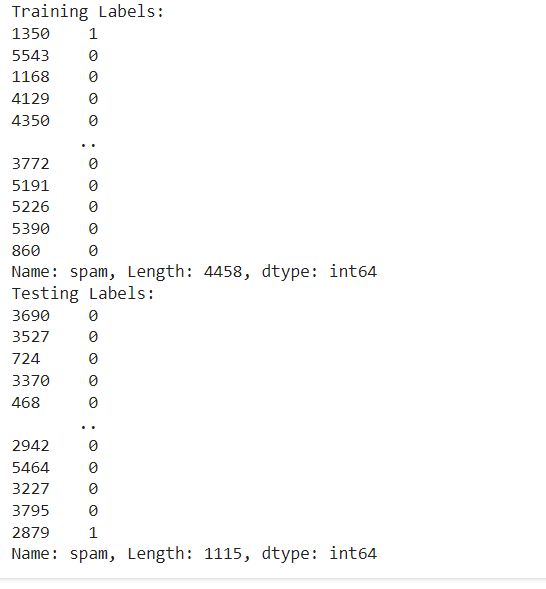
Applies the preprocessing function to the 'Message' column in the dataset. Converts the 'Category' column to binary labels: 1 for spam and 0 for not spam (which could include malware in this context).



**6. Splitting the Dataset**

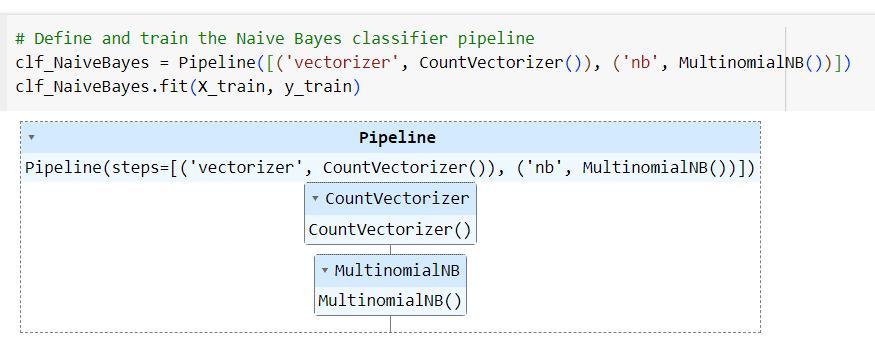
Splits the data into training and testing sets. 80% of the data is used for training, and 20% is used for testing.



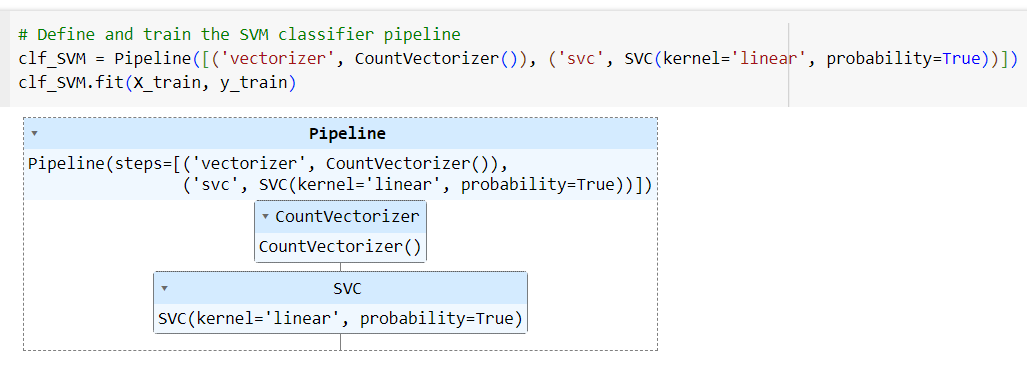
**7. Training Naive Bayes Classifier**

Creates a pipeline with a CountVectorizer (to convert text to a matrix of token counts) and a MultinomialNB classifier, then fits the model on the training data.



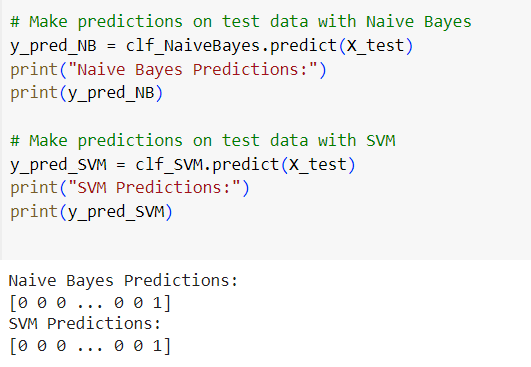
**8. Training SVM Classifier**

Creates a pipeline with a CountVectorizer and an SVM classifier with a linear kernel, then fits the model on the training data.



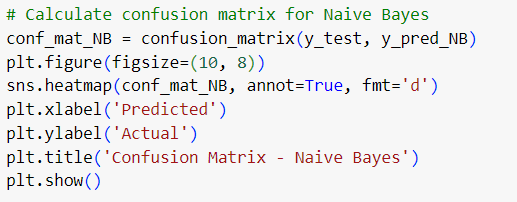
**9. Making Predictions**

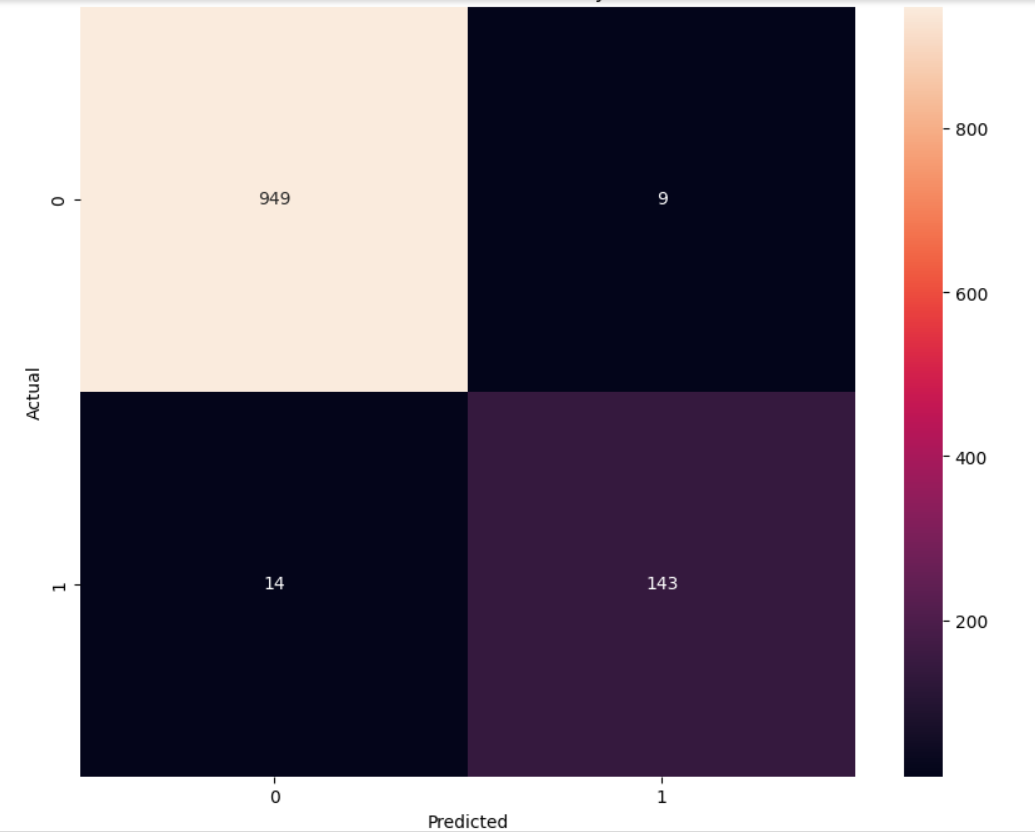
Makes predictions on the test data using both classifiers.



**10. Confusion Matrix and Metrics for Naive Bayes**

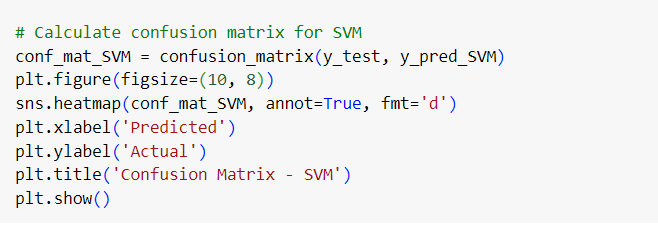
Calculates and visualizes the confusion matrix for the Naive Bayes classifier.

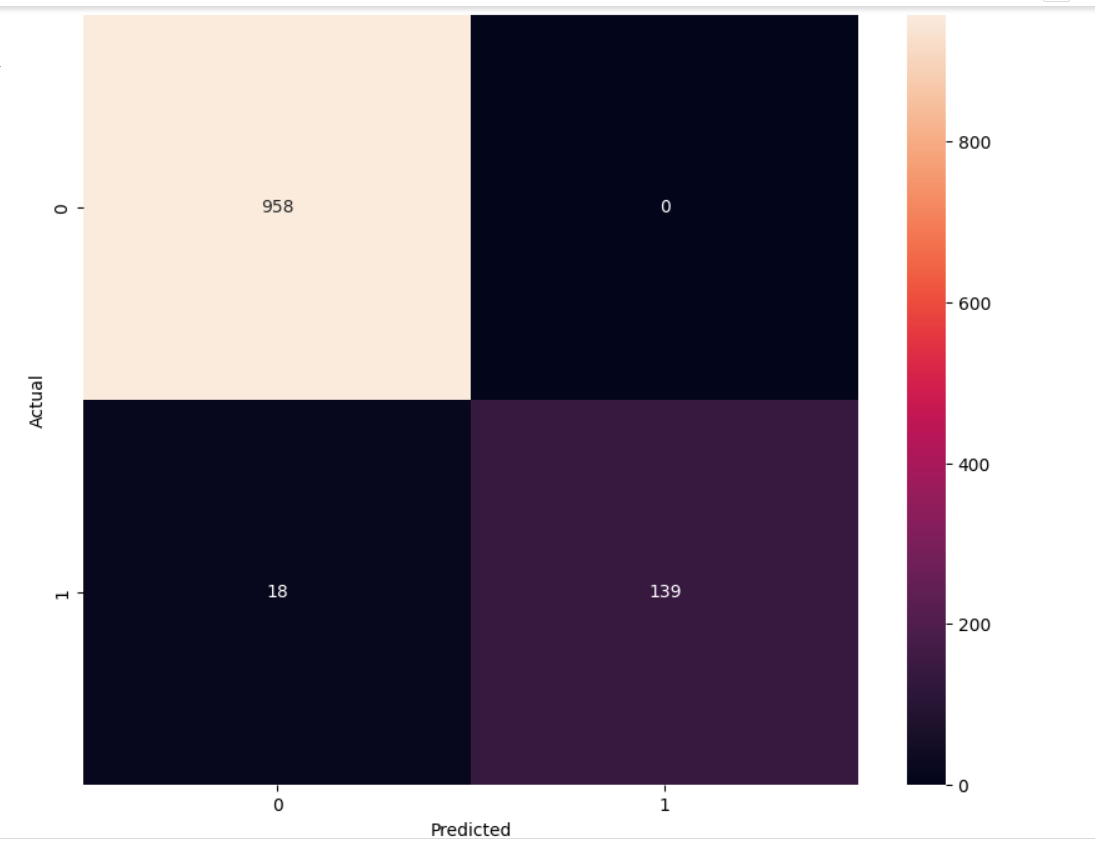




**11. Confusion Matrix and Metrics for SVM**

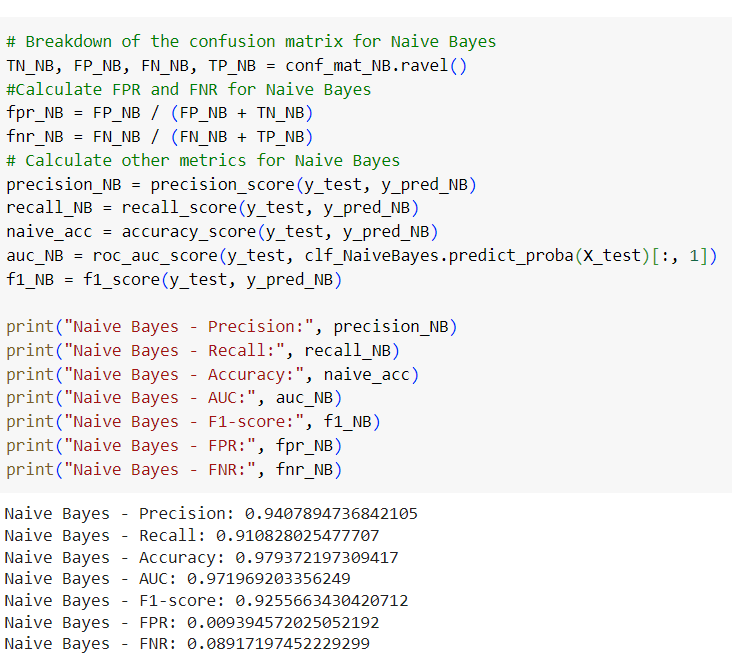
Calculates and visualizes the confusion matrix for the SVM classifier.





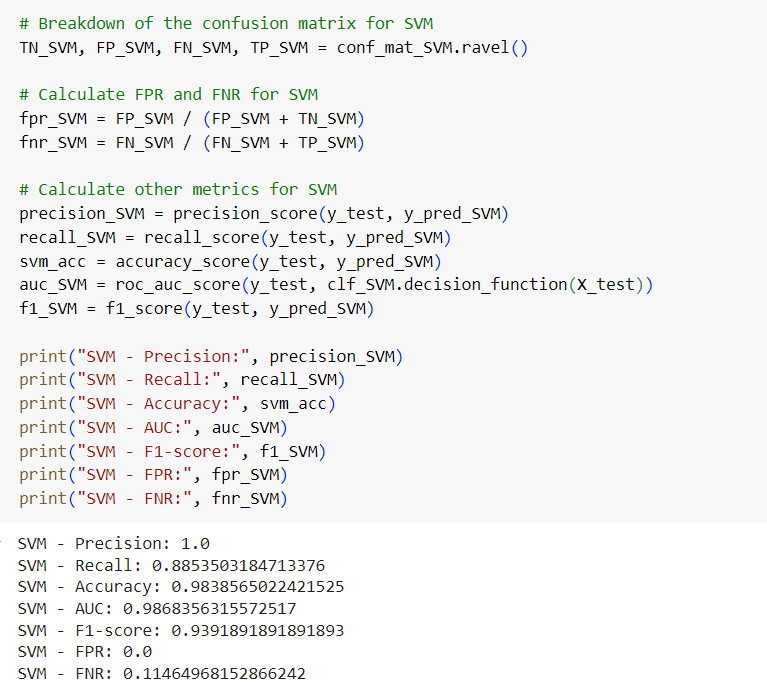
**12. Detailed Metrics Calculation for Naive bayes**

Calculates various evaluation metrics for the Naïve bayes classifier, including precision, recall, accuracy, AUC-ROC, F1-score, false positive rate (FPR), and false negative rate (FNR).



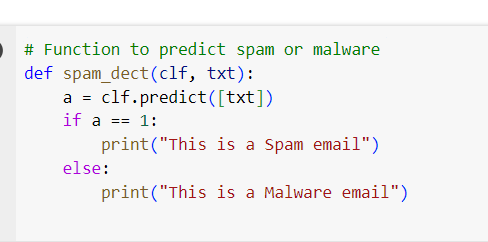
**13. Detailed Metrics Calculation for SVM**

Calculates various evaluation metrics for the SVM classifier, including precision, recall, accuracy, AUC-ROC, F1-score, false positive rate (FPR), and false negative rate (FNR).



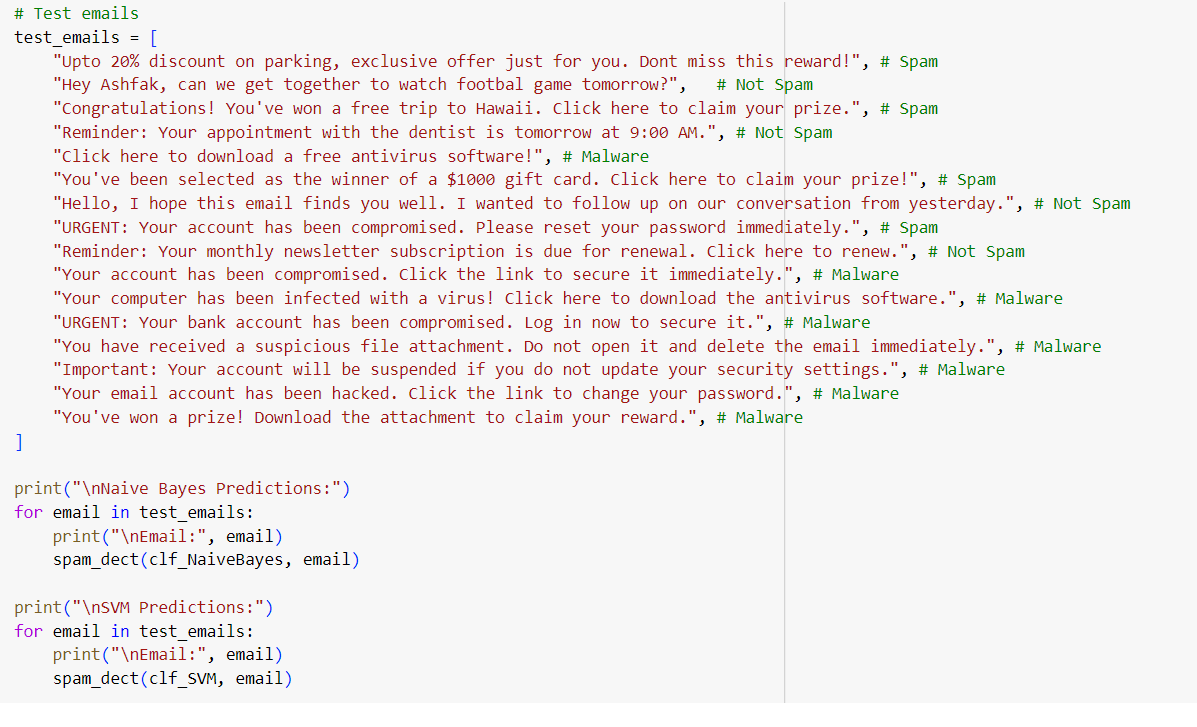
**14. Function to Predict Spam or Malware**

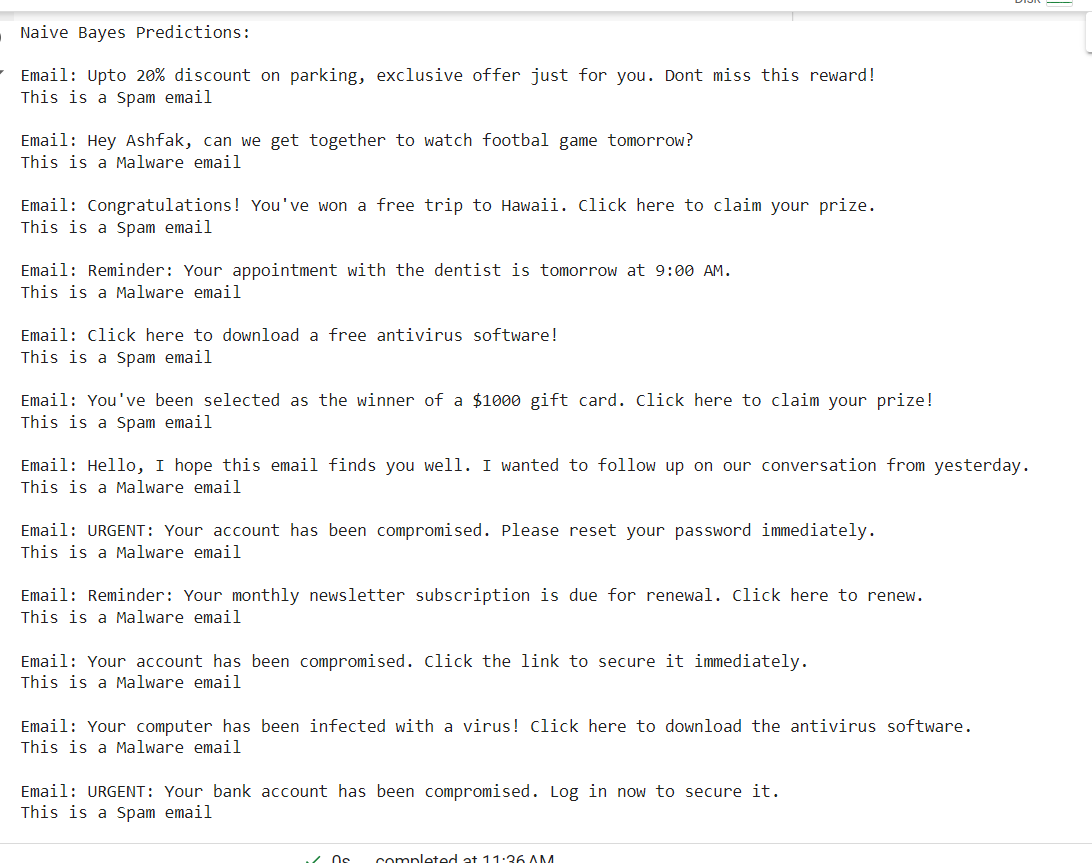
Defines a function to classify new emails as either spam or malware using the trained classifier.

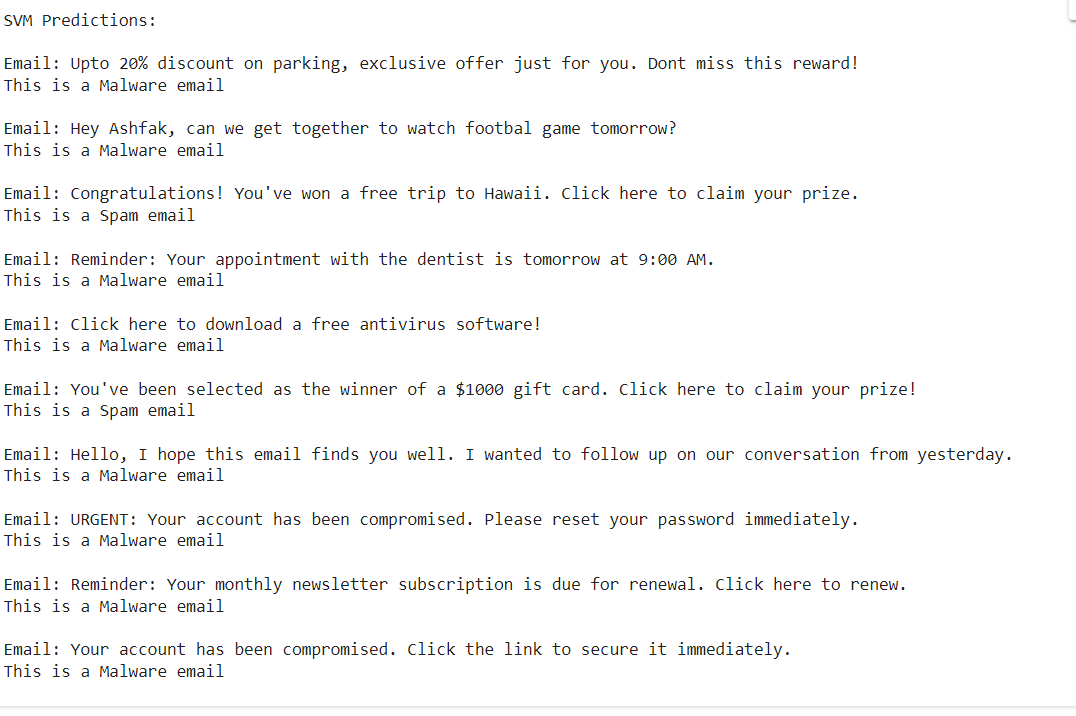


**15. Testing with Sample Emails**

Tests the classifiers on a set of sample emails to predict whether each email is spam or malware.





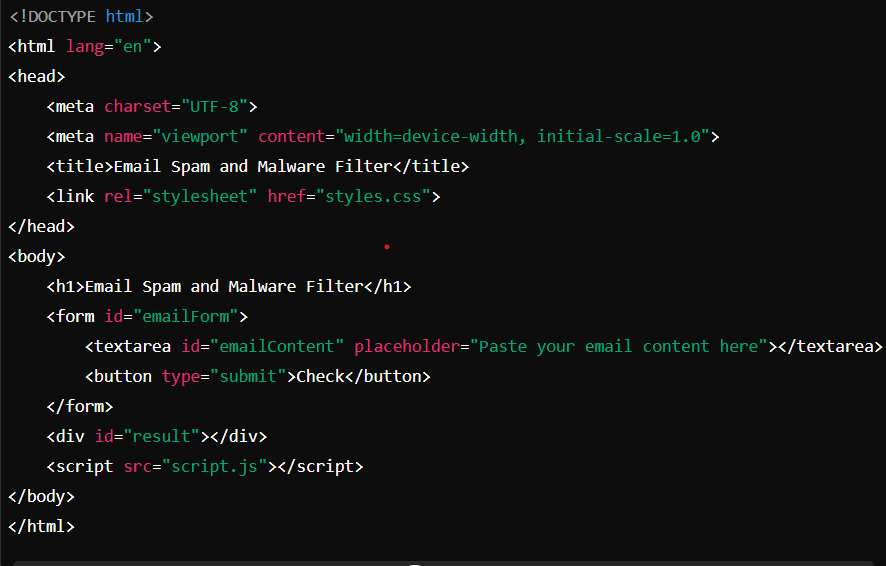


**RESULTS**

**Frontend**

**Create HTML File**

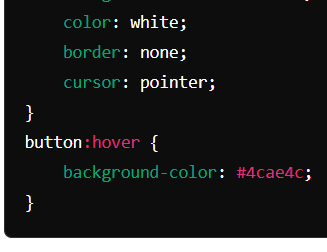
**index.html**



**Create CSS File**

**Styles.css:**





**Create JavaScript File**

**script.js:**



**Backend**

**Create Python Backend with Flask**

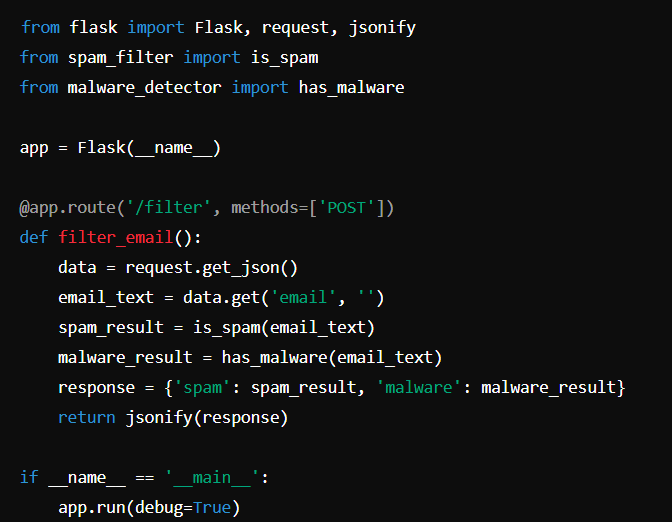
**Install Flask**

* Open Command Prompt.
* Navigate to the project folder Install Flask by running



**Create Main Application File**

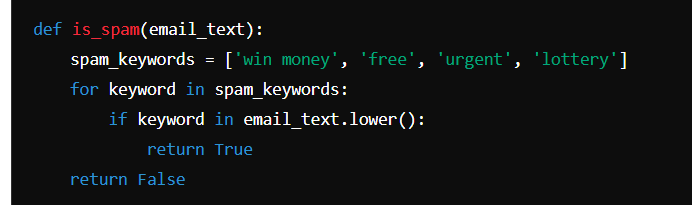
**app.py:**



Flask is a lightweight and flexible web framework for Python that is well-suited for developing web applications and APIs. It can be used to create a web service that filters emails for spam and malware by leveraging machine learning models.

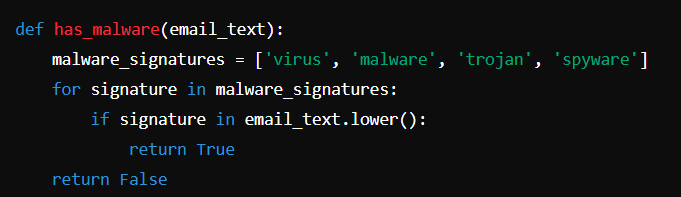
**Create Spam Filter Logic**

**spam\_filter.py:**



**Create Malware Detector Logic**

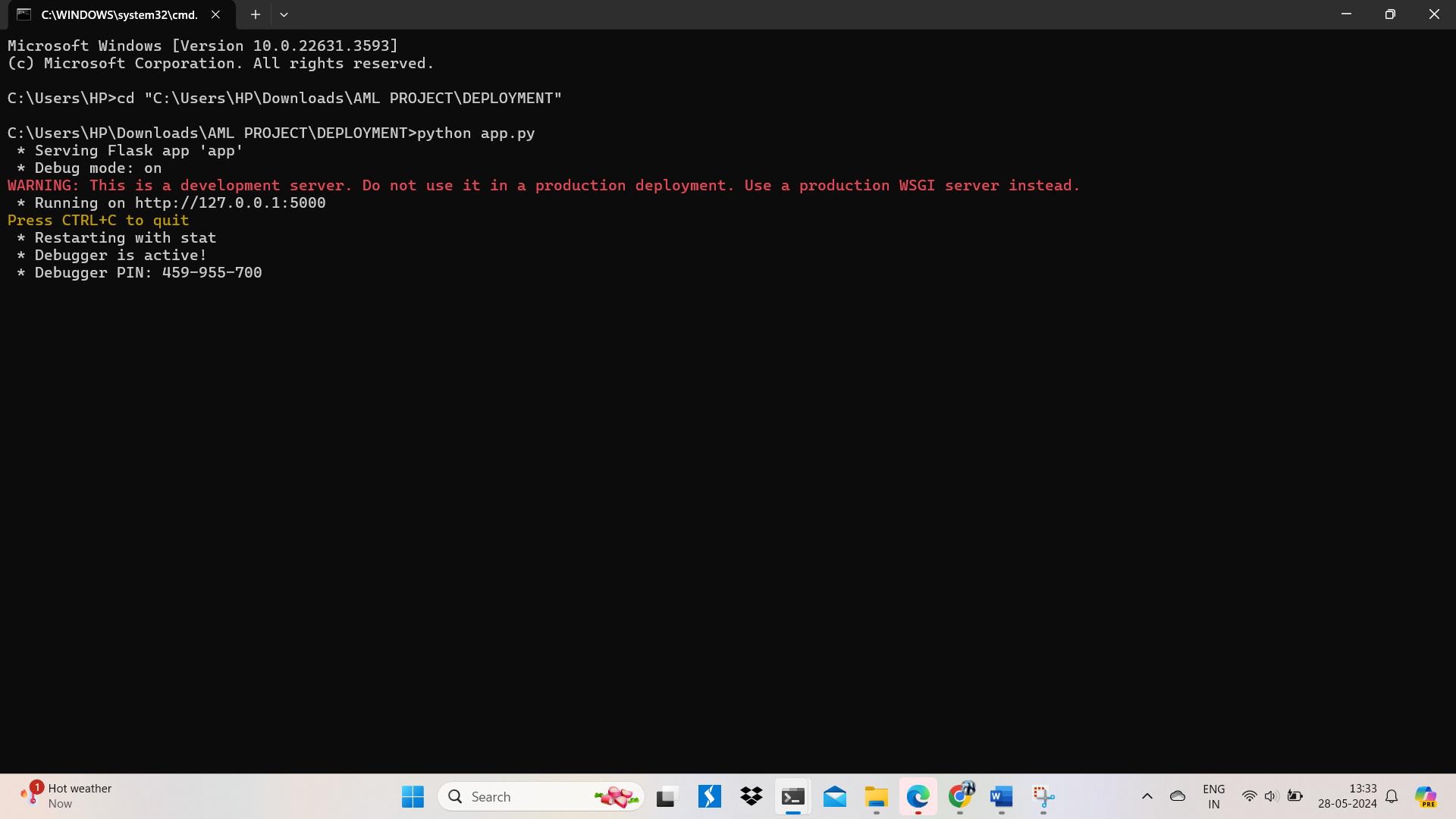
**malware\_detector.py:**



**Run the Flask Application**

* Open Command Prompt .
* Navigate to the project folder cd "C:\Users\HP\Downloads\AML PROJECT\DEPLOYMENT"
* Run the Flask app:



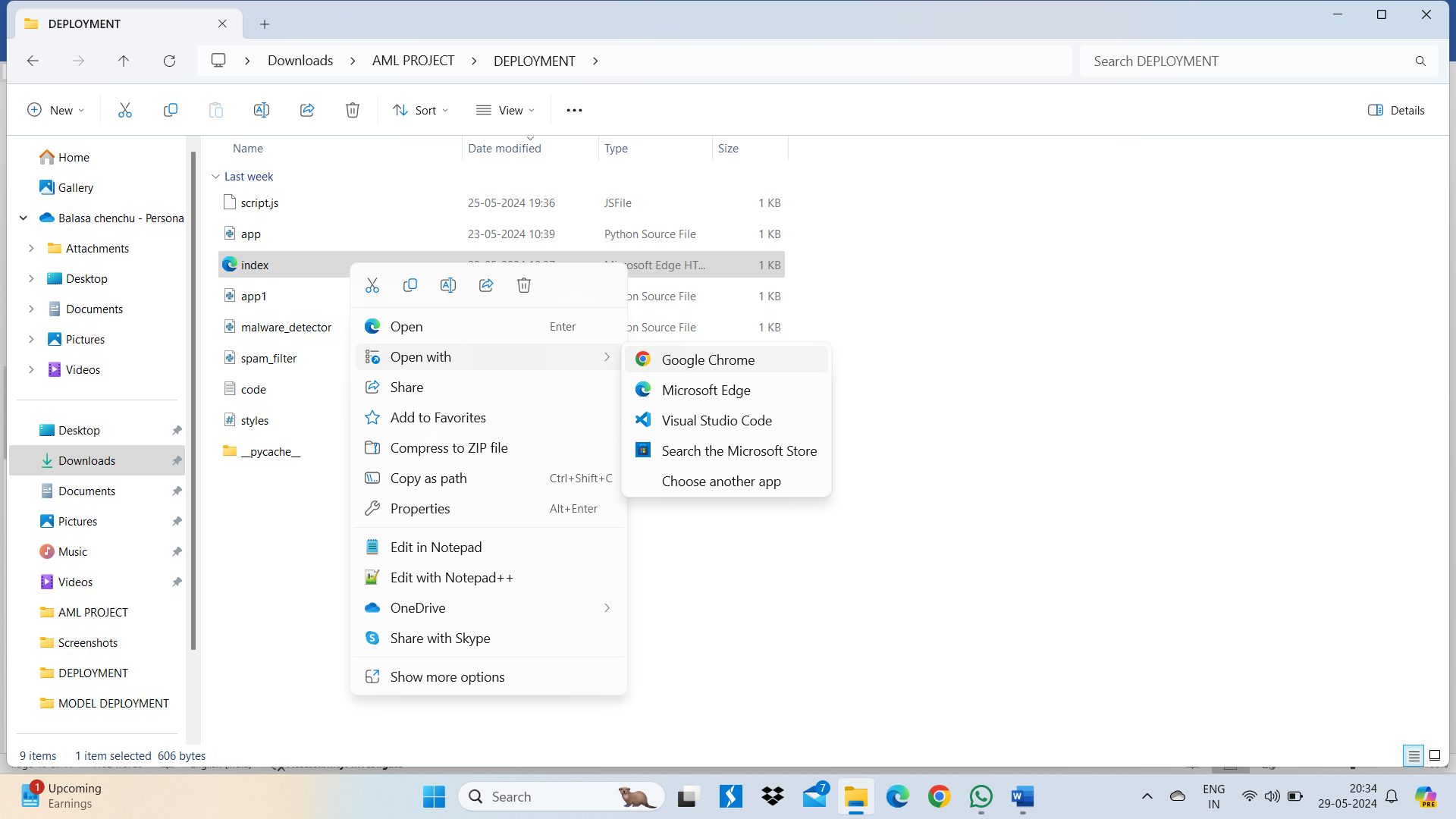


After giving python app.py in command prompt. Flask application runs successfully, and it will be able to access at http://127.0.0.1:5000 in the web browser.

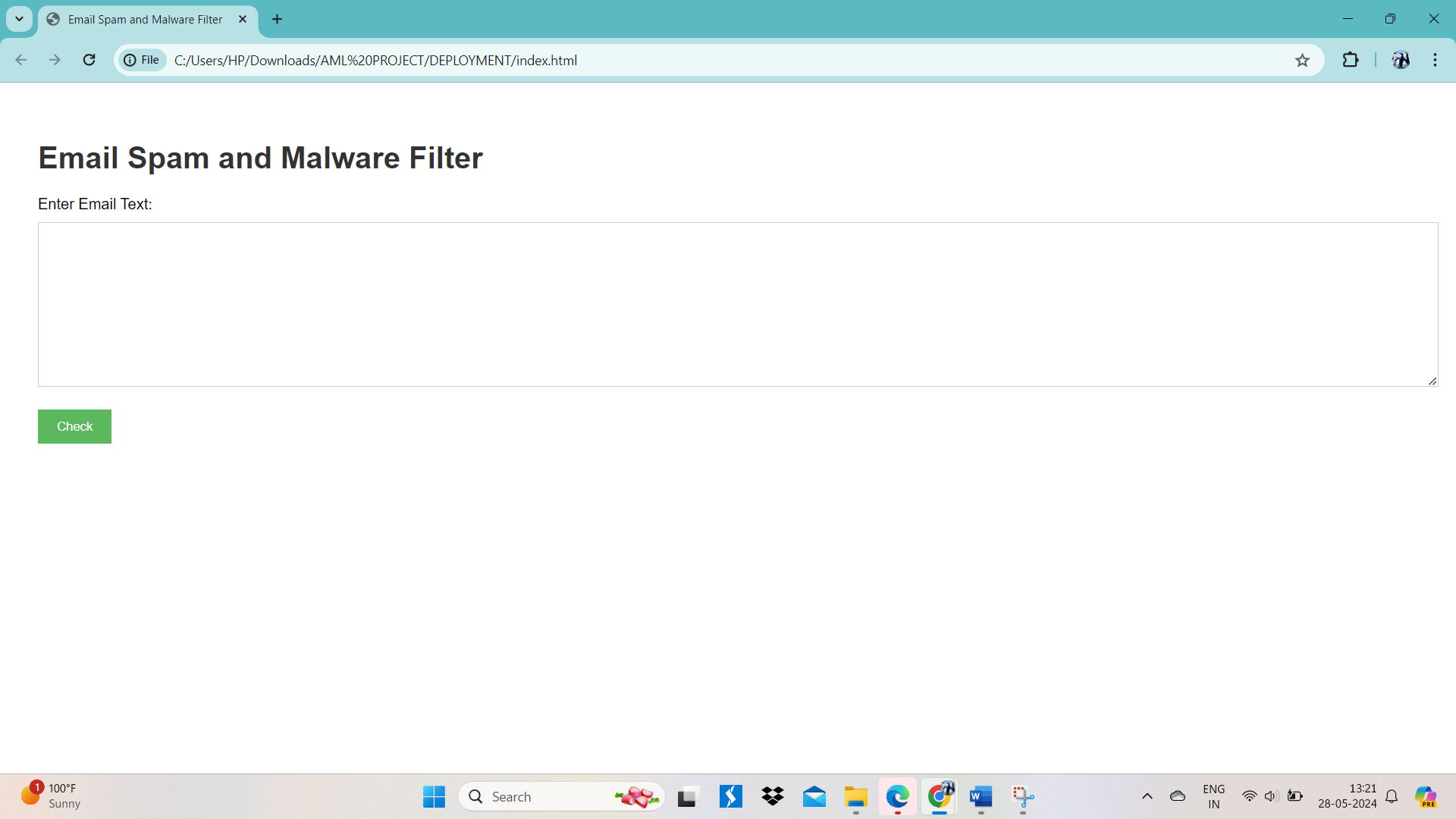
**Test Application**

Open index.html in web browser.

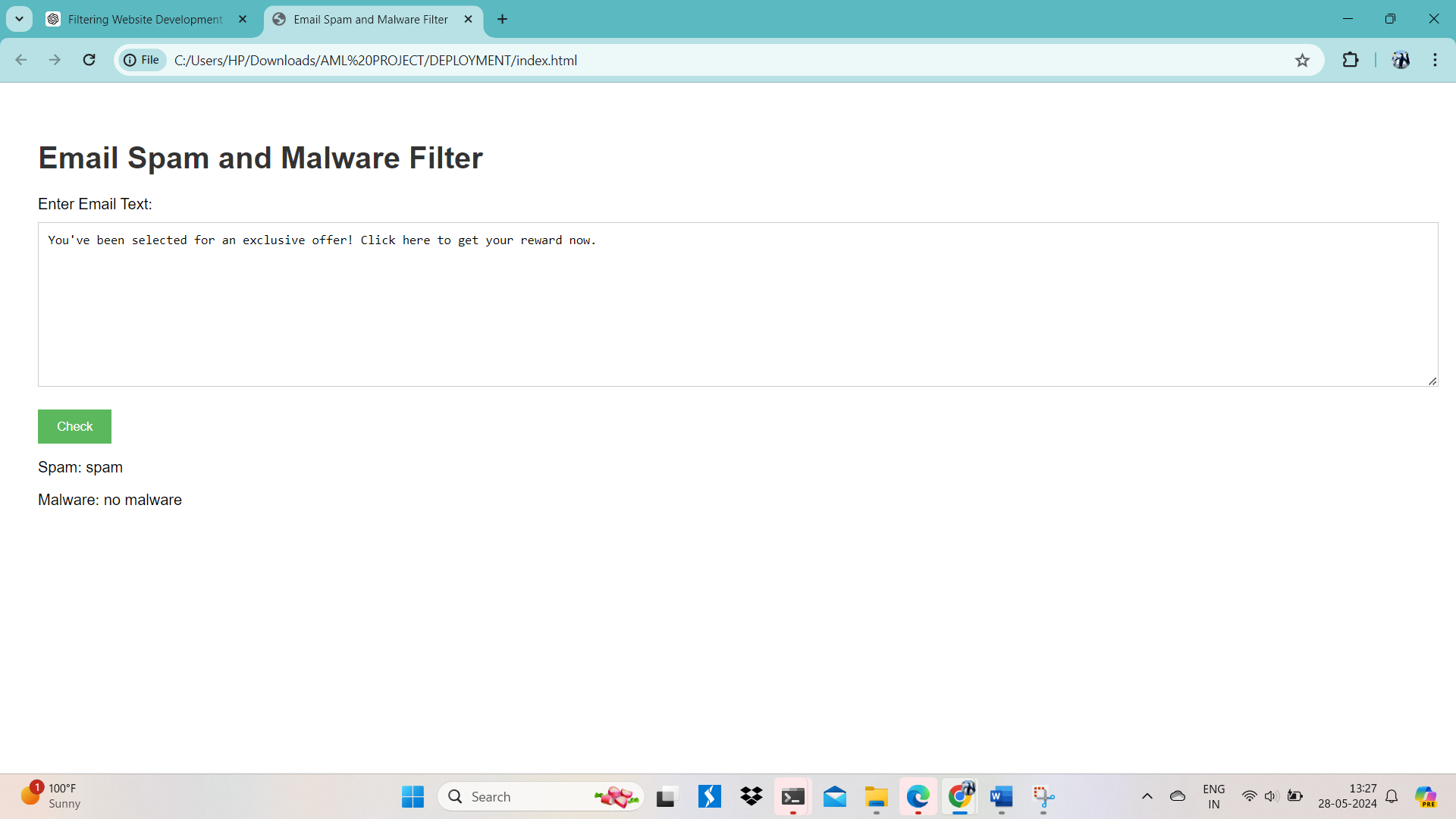
By right-clicking the index.html file in File browser and selecting "Open with" and the preferred web browser.



After clicking on that it directly takes me to the html page where I created website for email spam and malware filtering.



After going into the website.We can see empty text area in that empty text box.Enter some text into the textarea and click "Check".

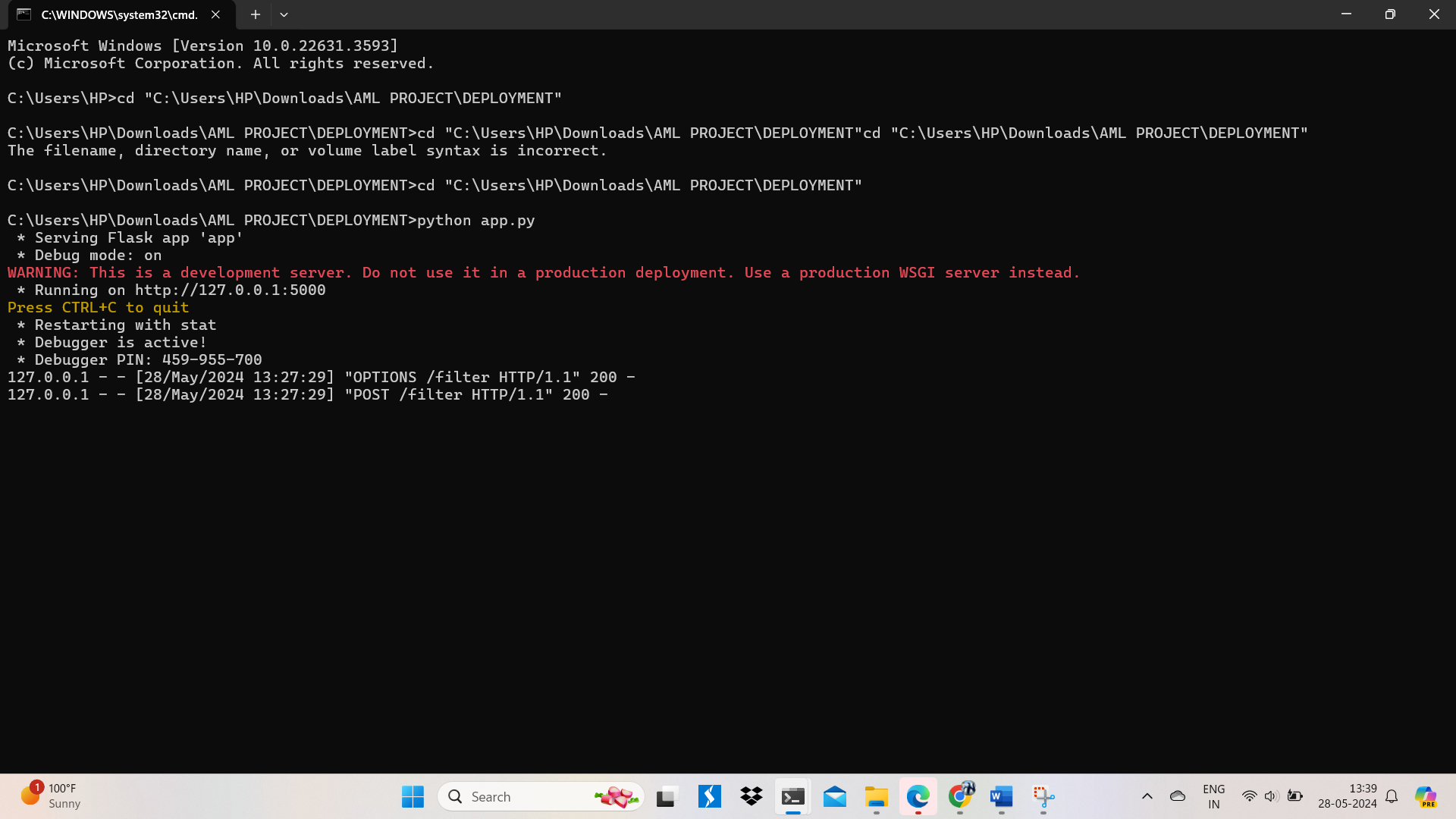


After entering the text into text area click the check button.It displays the result as shown in the above figure.

JavaScript should send the text to the Flask backend, which will check for spam and malware, and return the result to be displayed on the page.

**Verify Flask Server Log**

Check the Flask server log in terminal for any incoming POST requests and the responses. When we submit the form, we can see log entries like this as shown in the below figure.



**CONCLUSION**

In conclusion, both Naive Bayes and Support Vector Machine (SVM) classifiers demonstrate effectiveness in filtering email spam and malware. Naive Bayes, known for its simplicity and efficiency, performs well in classifying emails based on their content features. On the other hand, SVM, with its ability to handle high-dimensional data and find complex decision boundaries, shows promising results in distinguishing between spam and legitimate emails, as well as identifying potential malware threats. However, the choice between these classifiers may depend on specific requirements such as computational resources, dataset characteristics, and the need for interpretability. Overall, leveraging machine learning algorithms like Naive Bayes and SVM significantly enhances email security by accurately detecting and mitigating spam and malware threats, thereby safeguarding users privacy and minimizing potential risks associated with malicious email content.For Naïve bayes the accuracy is 97% and for SVM the accuracy is 98%.By comparing both algorithms the best model for Email Spam and Malware Filtering is SVM.

**REFERENCES**

1. Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, 5(6).
2. Gangavarapu, T., Jaidhar, C.D. & Chanduka, B. Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artif Intell Rev* **53**, 5019–5081 (2020). <https://doi.org/10.1007/s10462-020-09814-9.>
3. S. Kaddoura, O. Alfandi and N. Dahmani, "A Spam Email Detection Mechanism for English Language Text Emails Using Deep Learning Approach," 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 2020, pp. 193-198, doi: 10.1109/WETICE49692.2020.00045.
4. Mohammed, Mazin Abed, Salama A. Mostafa, Omar Ibrahim Obaid, Subhi RM Zeebaree, Mohd Khanapi Abd Ghani, Aida Mustapha, Mohd Farhan Md Fudzee et al. "An anti-spam detection model for emails of multi-natural language." Journal of Southwest Jiaotong University 54, no. 3 (2019).
5. Kim, Intae, Willy Susilo, Joonsang Baek, Jongkil Kim, and Yang-Wai Chow. "PCSF: privacy-preserving content-based spam filter." IEEE Transactions on Information Forensics and Security (2023).
6. Ahmed, Naeem, Rashid Amin, Hamza Aldabbas, Deepika Koundal, Bader Alouffi, and Tariq Shah. "Machine learning techniques for spam detection in email and IoT platforms: analysis and research challenges." Security and Communication Networks 2022 (2022): 1-19.
7. Karim, Asif, Sami Azam, Bharanidharan Shanmugam, and Krishnan Kannoorpatti. "Efficient clustering of emails into spam and ham: The foundational study of a comprehensive unsupervised framework." IEEE Access 8 (2020): 154759-154788.
8. Ma, Thae Ma, Kunihito Yamamori, and Aye Thida. "A comparative approach to Naïve Bayes classifier and support vector machine for email spam classification." In 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), pp. 324-326. IEEE, 2020.
9. Jazzar, Mahmoud, Rasheed F. Yousef, and Derar Eleyan. "Evaluation of machine learning techniques for email spam classification." International Journal of Education and Management Engineering 4 (2021): 35-42.
10. Junnarkar, Akash, Siddhant Adhikari, Jainam Fagania, Priya Chimurkar, and Deepak Karia. "E-mail spam classification via machine learning and natural language processing." In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 693-699. IEEE, 2021.