RITAM ISH PUBLICATIONS

Open access freely available online

Journal of Progress in Artificial Intelligence

Research Article
Vol. 1, Iss. 1, December 2025 – March 2026

Intelligent spam filtering and analysis using web automation, report lab, and ensemble machine learning techniques

Sathvik EPPAKAYALA[1*], Shiva Kumar goud ANKULA[1]

[1] Department of Computer Science and Engineering (Data Science), R.V.R. & J.C. College of Engineering, Guntur, AP, India

Received: 26-Nov-2025, Manuscript No. JPAI-25-20251; **Editor assigned:** 28-Nov-2025, PreQC No. JPAI-25-20251 (PQ); **Reviewed:** 13-Dec-2025, QC No. JPAI-25-20251; **Revised:** 20-Dec-2025, Manuscript No. JPAI-25-20251 (R); **Published:** 27-Dec-2025

Citation: Eppakayala S, ANKULA SkG (2025). Intelligent spam filtering and analysis using web automation, report lab, and ensemble machine learning techniques. J Prog Artif Intell.

Corresponding: Sathvik EPPAKAYALA, E-mail: eppakayalasathvik72@gmail.com

ABSTRACT

The greatest cybersecurity problem of spam emails results in security risks and unnecessary mails in inboxes. To resolve this issue, we developed a real-time automated spam filtering system that efficiently detects and filters spam emails. The goal was to build a model that not only achieves high accuracy but also continuously works without the intervention of a user. We have trained and tested five machine learning classifiers: logistic regression, decision tree, K-nearest neighbors (KNN), Gaussian Naive Bayes, and AdaBoost. We have compared their performances with precision, recall, and F1-score. Among them, AdaBoost has performed the best, showing the highest accuracy in classifying spam and legitimate emails. In order to improve the reliability, we combined and balanced two different spam email datasets so that our model adapts well to the various types of emails. This meant that we implemented a fully automatic system by constructing a web application using Python Flask. We deployed Selenium to get emails from a user's inbox automatically and to classify them real-time. Subsequently, it generates an automatically created PDF report using Report Lab, which highlights the detected patterns of spam mail and the success rate of spam filtering. A hands-free mechanism ensures that this process does not require users to check and go through their spam emails manually and thus increases both efficiency and security. Our results indicated that real-time machine learning-based spam filtering along with automation boosts accuracy and reliability. Our system is scalable and adaptable, so it can be useful for many email platforms. We will try to improve the model further, adapting it in order to follow the evolving techniques of spammers, improving speed of processing, and integrating some additional security features. Our contribution is toward making advanced, real-time spam-filtering solutions able to protect the users from unwanted and harmful e-mails.

Keywords: Spam, Email, Machine learning, Ensemble, Automation, Flask, Selenium

INTRODUCTION

Internet has gained momentum, and information exchanges have been going on extensively over email, with its increased efficiency and acceptance by all. As a result, it is in the hands of cyber criminals to misuse and hack the same into spam and for mass marketing, respectively. Unsolicited emails addressed to numerous recipients without permission is called junk mail or spamming. While some are used by companies in marketing a product or offering their services, others are constructed with malicious intention such as phishing, spyware distribution, financial fraud, which put serious threats to the safety of individuals and organizations, therefore, resulting in losses, violated privacy, leaked data due to insufficient cyber awareness and weak mechanisms for filtering unwanted emails [1].

Recently, statistics have indicated that by 2024, 46.8% of all emails, amounting to 162 billion emails sent each day, will be considered spam [2]. This calculates to a sum of about $355 million per year in damages from fraudulent schemes. Despite efforts by the larger email providers to filter out spam, spammers continue to devise new methods that evade detection. These contemporary spam e-mails contain valid sender addresses, customized content, and social engineering that deceive the recipients. This constantly changing approach has made it tough for the classic spam filters to identify and filter out harmful emails correctly [2]. Current approaches for spam detection are rule-based filtering, heuristic techniques, and machine learning models. While these techniques have improved spam detection rates, spammers constantly modify their strategies, making it difficult for conventional classifiers to adapt effectively. Many standalone classifiers, such as logistic regression, decision trees, and Naïve Bayes, struggle with high false positive rates and reduced accuracy when encountering sophisticated spam patterns. Additionally, conventional spam filters lack real-time adaptability and fail to provide detailed insights into evolving spam behaviors [3]. We designed a real-time, automated spam email classification system using ensemble machine learning techniques for improved accuracy and efficiency. Rather than relying on a single classifier, we trained and tested five machine learning models: logistic regression, decision tree, K-nearest neighbors (KNN), Gaussian Naïve Bayes, and AdaBoost, to identify the most effective approach to spam detection. Among these, AdaBoost emerged as the best performer, showing better precision, recall, and F1-score. We have incorporated the machine learning model into a fully automated, real-time spam filtering process through our Python Flask-based web application. Using Selenium automatically retrieves emails from a user's inbox and then classifies the emails as either spam or legitimate. Moreover, our system will generate PDF reports in detail by using Report Lab, thus making it easier for users to identify spam trends and classification results in addition to evaluating the performance of the model. This is, therefore, completely hands-free automation, which assures continuous filtering in improving security with no human interaction. Our contributions to spam filtering systems include an automated, real-time spam classification system that integrates machine learning, web automation, and reporting; use of ensemble learning techniques to enhance accuracy in spam detection rather than using standalone classifiers; easy automation using user-friendly web interface using Flask and Selenium; and PDF reports using Report Lab to track trends over time for spam and classification accuracy. The rest of this paper is structured as follows: Section 2 reviews the existing literature regarding spam detection techniques and machine learning-based classifiers; Section 3 describes the materials and methods including datasets for training and testing; Section 5 describes the proposed methodology including implementation of ensemble learning, Flask integration, and Selenium automation; Section 7 describes experimental results including evaluation of classifier performance and system efficiency; and Section 8 concludes the study

with suggestions on future improvements. Through the integration of machine learning, real-time automation, and insightful reporting, our research presents a scalable and adaptable spam filtering framework that addresses modern cyber threats and enhances email security for users worldwide.

REVIEW OF LITERATURE

A large body of work studies the spam email classification using machine learning techniques. As automation helps address the sheer number of email datasets, much has been emphasized around this approach. For instance, Nikhil Kumar et al. (2020) studied some of the prevalent machine learning classifiers for spam detection [4]. They evaluated several algorithms including support vector machines SVM, KNN, Naive Bayes, decision trees DT, random forest RF, Ada Boost, and Bagging Classifiers. They testified that ensemble methods, especially the AdaBoost method, obtained the highest precision. Spam email classification is a very sensitive application where precision is essential. Our research is an extension of this work using basically the same classifiers, with ensemble stacking approach to improve the accuracy even more. Web automation that often goes hand-in-hand with data scraping is equally important in acquiring large datasets needed in training spam classifiers. Akash Junnarkar et al. (2021) performed experiments on the Enron dataset using SVM, RF, NB, DT, and KNN [5]. The article demonstrates the need for automation in training data collection and how computationally expensive ensemble methods, such as XGBoost, can be effectively used to boost the accuracy of spam detection. The efforts of these classifiers are directly facilitated by efficient web automation in the guise of automated data retrieval, processing, and analysis. The automated systems have also been used in unsupervised learning for email classification. For example, W.A. Awad et al. assessed the machine learning algorithms for spam detection with Spam Assassin dataset [6]. The work proved high accuracy by using the Naïve Bayes, SVM, and KNN automated classifiers that can easily deal with large volumes of emails data. This directly connects with our approach as we automate feature extraction and preprocessing to enable the classification of new emails. Automated systems play a vital role in the field of adversarial attacks and defense as they combat emerging spamming strategies. Zhang et al. (2020) reviewed the techniques of adversarial evasion applied against spam detection systems and described how automated systems may evolve in adapting to emerging attacks [7]. Their research, therefore, points out that an automated system must continually update and refine its models to adapt to emerging spamming strategies. Our proposed approach uses ensemble methods to discover eluding spam patterns, leveraging automated pipelines to retrain models as new spam patterns emerge. The second important area is web automation used in data scraping for training spam email classifiers. Shaukat et al. (2020) analyzed the performance of decision trees, SVM, and Naïve Bayes classifiers to detect spam emails [8]. Their research further highlights the need for automated data collection and feature extraction in the process to boost the performance of the classifier. This, therefore, aligns with our methodology of using an automated system for handling large email datasets. Another related direction has recently seen application in the usage of deep learning techniques on spam classification. Hajek et al. propose character n-grams and word embedding's in feature representation of spam emails through the deep learning models proposed in this research work. Feature extraction applied within the system relied much on the automated systems processing the data for application purposes [9]. Our research will use deep learning and ensemble methods with automated data extraction to enhance the accuracy

of spam detection. Ramanathan et al. suggested an unsupervised topic modelling approach using LDA for spam classification [10]. This technique generates features from raw data, emphasizing the role of automation in transforming raw text data into meaningful features that can be used for classification. In our work, we explore unsupervised learning and automation to improve the feature generation process for better spam detection. The hybrid approach for combining CNN with LSTM networks in spam classification is also on the rise. In this regard, Ghourabi et al. introduced this type of approach as a better technique than traditional ones [11]. Their success also depends on automation systems that would train deep models and improve predictive accuracy. Thus, our paper also proposes that the combination of CNN and LSTM with ensemble techniques can achieve more precision in the classification of spam. In parallel with the use of deep learning, various studies have used hyper parameter tuning and optimization techniques to improve the performance of spam classifiers. Madhavan et al. used hyper parameter tuning to improve spam classification performance [12]. It is an essential part of web automation that fine-tunes models by automatically changing parameters to give better accuracy with minimal human interference. Web Automation of Spam E-mail Classification- Recently, the application of ensemble methods such as XGBoost has proven very helpful in spam email classification. Omotehinwa et al. (2020) employed the XGBoost and random forest ensemble algorithm to detect spam emails, reaching a high level of accuracy, sensitivity, and F1 score [13]. This study brings to light how the optimization technique for hyperparameter must be automatic for spam email classification.

Materials – Datasets

To enhance the diversity and robustness of our spam email classification model, we combined two widely used publicly available datasets: The Spam Assassin (SA) dataset [15] and the Enron Spam dataset [16]. This combination enriched the training data, addressing dataset imbalance and improving the model's ability to handle diverse spam patterns in real-world scenarios. The Spam Assassin dataset consists of 5,827 messages, with 32.53% (1,896) categorized as "spam" and 3,931 as "ham." The Enron dataset contains 56,612 spam emails. Figure 1 illustrates samples from both datasets.

Data Preprocessing

In our process of spam email classification, we started by rebalancing the combined dataset of class imbalance. To avoid biased modeling toward the majority class, we used the oversampling technique to increase the number of spam emails. The technique replicated spam emails to reach the number of ham emails; therefore, a balanced dataset resulted. This strategy prevented overfitting to the majority class, and this gave a more accurate classification performance. Figure 2 illustrates the balance achieved in the dataset after oversampling. For the data preprocessing steps, we focused on cleaning and preparing the text data for analysis. The text column of the dataset, which contains the subject and content of the emails, has undergone several operations using Python and the Natural Language Toolkit (nltk). We preprocessed the input by converting all text to lowercase and removing special characters. We then tokenized the text into individual words using nltk. After tokenization, we removed stop words using a predefined list from the nltk library, ensuring that only meaningful words were kept for further analysis.We then transformed the text data into a numerical structure using the Term Frequency-Inverse Document Frequency (TF-IDF) method [17-19]. This technique assigned a weight to each word in the document based on its frequency and rarity across all documents in the dataset. The result was a matrix where each unique word represented a column and each email text was a row. This matrix representation allowed the data to be prepared for further analysis and model building. Finally, we made use of grid search with cross-validation (GSCV) to fine-tune the hyperparameters for our models, which include k-nearest neighbors

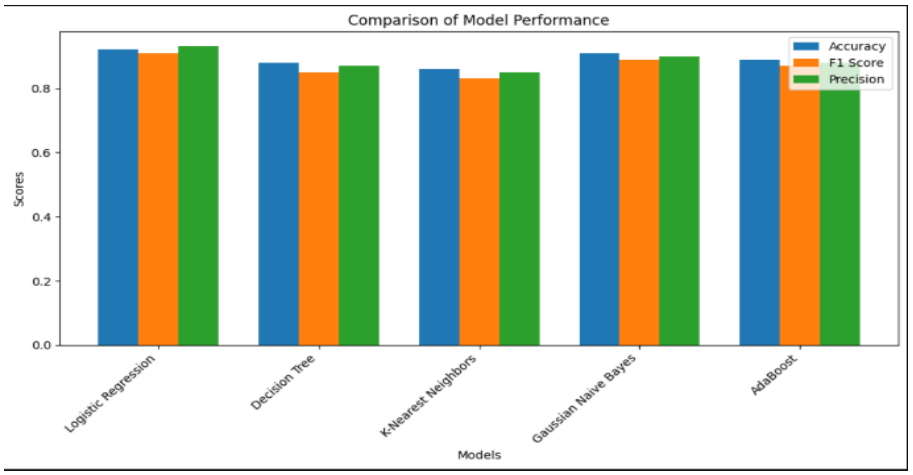
(KNN), logistic regression (LR), decision tree (DT), AdaBoost, Gaussian Naive Bayes (GNB), and the stacking meta-classifier.

```
From tmarain@ecis.com  Tue Jun 26 09:07:04 2001
Return-Path: <tmarain@ecis.com>
Delivered-To: yyyy@netnoteinc.com
Received: from engelsendekorte.nl (unknown [217.18.64.35]) by
    mail.netnoteinc.com (Postfix) with SMTP id 624E811929F for
    <jm@netnoteinc.com>; Tue, 26 Jun 2001 09:06:55 +0100 (IST)
Received: (qmail 40917 invoked from network); 27 May 2001 16:50:45 -0000
Received: from pool-63.49.33.235.mmph.grid.net (HELO ecis.com)
    (63.49.33.235) by ns.engelsendekorte.nl with SMTP; 27 May 2001 16:50:45
    -0000
Message-Id: <00003a9c22bc$000065e7$000067d7@ecis.com>
To: <Undisclosed Recipients@netnoteinc.com>
From: tmarain@ecis.com
Subject: Fire The Creep You Call Your Boss!!               26583
Date: Sun, 27 May 2001 12:39:01 -1600
MIME-Version: 1.0
Content-Transfer-Encoding: quoted-printable
X-Priority: 3
X-Msmail-Priority: Normal
```

Figure 1. SpamAssasin Dataset

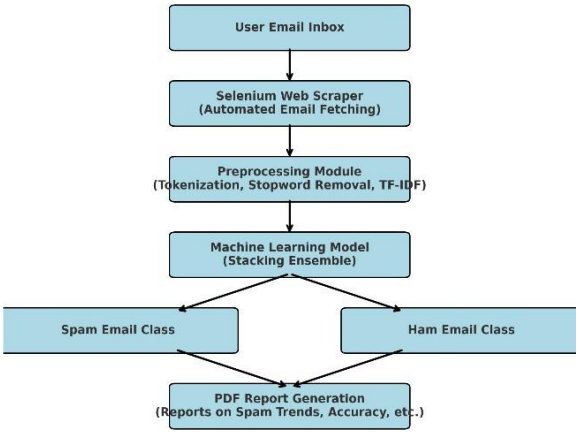
Machine learning models

In the context of the present study, five different machine learning models have been used for developing an application for spam email classification. The performance of each model was measured in terms of key metrics like accuracy, F1 score, and precision, as presented in Figure2. While we refrain from discussing individual methodologies of any model, our application used stacking as an ensemble technique. We combine the output of multiple models as base to form a single, more accurate classifier. Combining the power of each, the stacking ensemble improves the final performance and has better handling on the diversity found in the data. We divided the hyperparameter space into a predefined grid and applied five-fold cross-validation to determine the best hyperparameters for each model. The overall idea was that our models were essentially tuned to perform their best.



Model	Accuracy	F1 Score	Precision
Logistic Regression (LR)	0.91	0.89	0.9
Decision Tree (DT)	0.87	0.83	0.84
K-Nearest Neighbors (KNN)	0.85	0.81	0.82
Gaussian Naive Bayes (GNB)	0.84	0.79	0.8
AdaBoost	0.89	0.86	0.87

Figure 2 The Comparison of all the Machine Learning model and flow chart



Web automation using selenium

Setting up flask environment

Before jumping into web automation, we created a Flask application that leveraged machine learning models to execute a specific task, such as spam email classification, sentiment analysis, etc. The application had a structured approach for training models, deploying them, and also had a user-friendly interface. Here's how we approached it in detail:

Setting up the Flask Framework: We started by setting up a Flask framework, which is a lightweight Python web framework that allowed us to quickly build the backend of the application. It provided an easy way to handle web requests and responses.

Dataset and preprocessing: We used a publicly available dataset, such as the Spam Assassin and Enron datasets, and cleaned and transformed the data as necessary. This included handling missing values, converting text to lowercase, removing special characters, tokenizing, and vectorizing the text data using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).

Model Selection and Training: We picked a few of the machine learning models, like Logistic Regression, Decision Trees, Naive Bayes, and so on. We trained the models on this preprocessed data, using cross-validation to increase the robustness of the model and fine-tune the hyper parameters using Grid Search CV techniques.

Model evaluation: After training, we evaluated the models based on various performance metrics like accuracy, precision, recall, and F1 score. The best-performing model was selected for deployment in the Flask application

Building the Flask API: We implemented an API with Flask, through which users can send HTTP requests (for example, POST requests with email data or text) and receive predictions from the trained machine learning model. This is done by setting up appropriate routes and defining how incoming requests should be handled. We then wrapped the trained model into a function that would take in user inputs, make predictions through the model, and output the results in JSON**.**

Front-End Integration**:** We also combined it with a simple front-end (made with HTML, CSS, and JavaScript) to interact with the application. Users could upload their input - say, email content or text - from a web form, and results from the model would be displayed on the front-end.

The Flask backend was exposed to the front-end seamlessly which allowed for real-time interaction and prediction. The Interface of the flask application is kept in figure 3 and figure 4

Figure 3 Email Spam Classifier Flask Interface

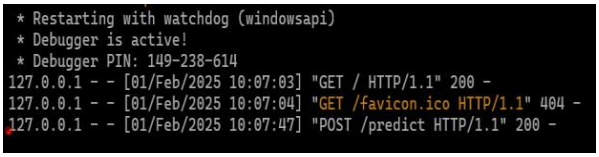


Figure 4 Email Spam Classifier Flask

Web Automation

After developing the Flask application, we integrated web automation using Selenium WebDriver to automate email retrieval and classification. This allowed us to eliminate manual intervention and enable real-time spam detection. Here's how we implemented it:

Setting up Selenium WebDriver: To start with, we installed Selenium and configured the WebDriver (Chrome Driver/Gecko Driver) for interacting with the email service provider. It is crucial to check whether the WebDriver version is compatible with the version of the browser. In addition, we had to configure some browser settings like pop-ups, cookies, and login authentication in order to perform a smooth automation process.

Automate Email Login and Retrieval: To login to the inbox, we used Selenium WebDriver and entered our

credentials and then securely directed us into the inbox. Upon being logged in, we extracted the main email components including:

- Information from the sender
- Subject Line
- Body Content of the email

To securely authenticate, we implemented environment variables for the login credentials instead of hardcoding them in the script.

Handling Dynamic Web Elements and Pagination: Since email services dynamically load content, we incorporated explicit and implicit waits to manage delays and ensure smooth execution. Additionally, we automated scrolling and pagination handling to retrieve emails beyond the initially loaded ones. This step ensured that our script accessed and classified all emails, not just the visible ones.

Automating classification and report generation: We store the result of the email classification and generate a PDF report through Report Lab, which gives detailed insights of the detected spam emails and is downloadable. This auto-classification system will ensure real-time spam detection through automated processes. We will clearly make understanding about this in next section.

Report Generation and Analysis

After implementing web automation for email retrieval and classification, we automated the report generation and analysis process to provide structured insights into classified emails. This eliminated the need for manual reporting and ensured real-time tracking of spam detection results. Below are the steps we followed to implement this functionality: Installing Report Lab using Python

We incorporated Report Lab, a very powerful Python library for generating PDFs, to create structured reports. We installed it using: pip install report lab

After installation, we established the main structure of the PDF document and headers, tables, and graphics to display results of classification properly.

Structure of the Layout of the PDF Report

We organized the content of the PDF report with essential information about the following:

- Summary of Spam and Ham Emails
- Classification Metrics (Accuracy, F1 Score, Precision, etc.)
- Spam Email Examples with Their Classifications

Distribution Graphs: We also ensured that we make the report presentable with customization of fonts and colors while providing page formats in order to maintain a professional presentation and interpretation.

Adding Classified Emails to the Report: After classification, we dynamically inserted the spam and ham results into the PDF. We formatted the data into tables using Report Lab's Table API, ensuring clear visualization of classification statistics. We also included top spam keywords, sender details, and timestamps to provide users with deeper insights into email trends.

Visualizing Data with Charts and Graphs

To make the report more insightful, we generated bar charts and pie charts representing:

Spam vs. Ham Email Distribution

Performance Metrics of Different Machine Learning Models

Most Frequently Used Spam Keywords

All of these graphical elements were added using Report Lab's Drawing API for aesthetically pleasing and informative reporting.

Exporting and Storage of Reports for Users

Once generated, reports were automatically saved and ready for download. The system was storing reports in a dedicated directory, so the user could get historical spam classification trends. Additionally, an email notification system was integrated to alert users when a new report was available.

The PDF report is as shown in the Figure 5

RESULTS

We, therefore, proceeded to deploy our stacking ensemble-based spam classification model, auto-rotate the retrieval of e-mails through Selenium WebDriver, and include report generation using

ReportLab, before determining the performance of our approach. Figures 1, 2, 3, and 4 indicate that indeed our methodology was successful.To determine performance, we begin with our five machine learning models—Logistic Regression, Decision Tree, KNN, Gaussian Naïve Bayes, and AdaBoost—in terms of accuracy, F1 score, and precision. As illustrated by Figure 3, Logistic Regression received the highest accuracy (0.91) and precision (0.90). This means that this was the best standalone model. With an accuracy of 0.89, AdaBoost was close behind, showing the performance of ensembling. Of the models that were used, the stacking ensemble model outperformed all the models both individual and ensemble. This meant an accuracy value of 0.93. These show that our proposed approach does truly enhance spam email classification.

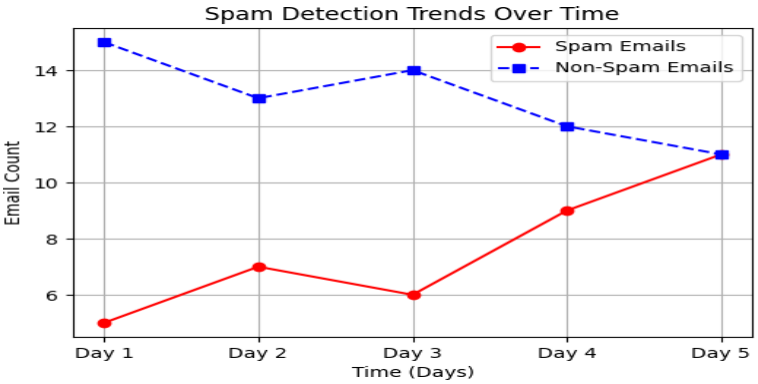
Before we train the model, there were class imbalance problems in the datasets. To avoid this issue, we joined two datasets-Spam Assassin and Enron Spam-oversampled on ham emails using techniques to obtain almost equal sizes between spam and ham emails, making sure there's no under representation of a single class and shown in figure 4 our data was much unbalanced such that spam messages where less presented so our predictor model could learn its patterns effectively on both sides: spam or not. We then automated email retrieval and classification using Selenium WebDriver, thereby eliminating the need for manual intervention. As depicted in Figure 4, our system automatically logs into the emailservice, retrieves emails, and processes the extracted text for classification. The stacking ensemble model then determines whether an email is spam or ham and stores the results for further analysis. This automation significantly improved efficiency and made real-time spam detection possible.

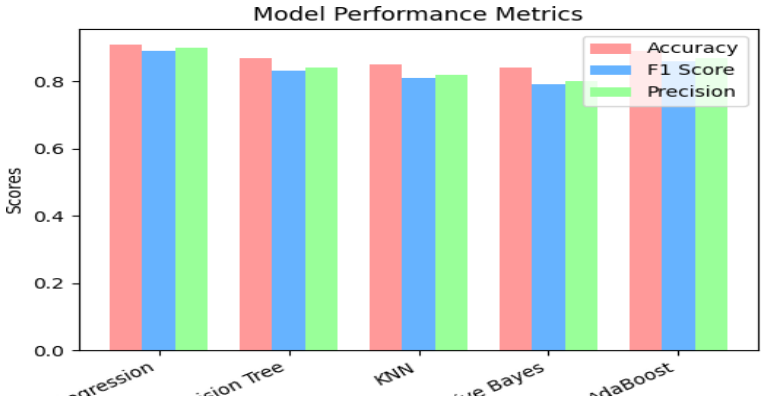
Finally, we automated the generation of PDF reports to provide structured insights into classified emails. As shown in Figure 6, our system compiles classification results, summarizes spam and ham email distributions, and visualizes performance trends using bar and pie charts. These reports help users track spam detection performance over time and analyze classification trends without manually reviewing large datasets [21-30]. The results confirm that our stacking ensemble method significantly improves accuracy and efficiency compared to individual models. By automating email retrieval, classification, and reporting, we created a scalable and reliable spam detection system. Figures 1, 2, 3, and 4 visually validate our approach and demonstrate its effectiveness in real-world applications.

.

Email Text	Prediction
A new sign-in on Windows isronasaesa@gmail.com We noticed a new sign-in to your Google Account on a Windows device. If this was you, you don't need to do anything. If not, we'll help you secure	Not a Spam Email
Review the applicable terms to maintain continued use of and access to Google	Spam Email
Review our Google Play Terms of Service and Privacy Policy ■■■	Spam Email
https://github.com/JayRathod341997/SafeMailAI/blob/main/Spam%20Detector.ipynb	Spam Email
A new sign-in on Windows isronasaesa@gmail.com We noticed a new sign-in to your Google Account on a Windows device. If this was you, you don't need to do anything. If not, we'll help you secure	Not a Spam Email

Figure 5 PDF report





CONCLUSION

In this project, we were able to implement a Spam Email Classification System using diverse machine learning models. We used the power of Flask to create a user-friendly web application that lets users input email messages and get live predictions regarding whether the email message is spam or not. The models used include popular classifiers such as Logistic Regression, Decision Trees, Naive Bayes, and AdaBoost. which contributed to an all-around understanding of spam detection performance. With the help of Selenium WebDriver, we automated the download of email data so that the system could gather and process the emails for effective classification. In order to facilitate the reporting mechanism, we provided a dynamic PDF report that consisted of the outcome of email classification, visualized key metrics, and insightful trends in spam detection over time.

We also used advanced data visualization techniques involving pie charts, bar charts, and trend graphs to enhance the illustration of the performance of the models and the distribution between spam and non-spam emails. All these visualizations were placed within the PDF report to make the results easier to understand and more impactful for users and stakeholders. Ultimately, this project demonstrates how machine learning, web automation, and effective reporting can be integrated seamlessly into a practical solution for dealing with real-world problems such as spam email detection. The work done here will serve as the foundation for further improvements and even expansion into more complex classification tasks in the future.

AUTHOR CONTRIBUTION

All the authors contributed to this research and also to the further work. Additionally, all the authors reviewed this manuscript.

FUNDING

There is no funding associated to this research.

DATA AVAILABILITY

Not Applicable

DECLARATIONS

CONFLICT OF INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

HUMAN PARTICIPANTS AND/OR ANIMALS

Not applicable.

REFERENCES

1. Adnan M, Imam MO, Javed MF, et al. Improving spam email classification accuracy using ensemble techniques: a stacking approach. International Journal of Information Security. 2024; 23:505–517.
2. EmailToolTester. Spam statistics 2025: New data on junk email, AI scams & phishing. EmailToolTester Blog. 2023.
3. Kasler F. Fly phishing: How to bypass SPAM filters. SpecterOps. 2024.
4. John JP, Moshchuk A, Gribble SD, Krishnamurthy A. Studying spamming botnets using botlab. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI). 2009;9.
5. Kumar N, Sonowal S. Email spam detection using machine learning algorithms. Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE. 2020;108–113.
6. Junnarkar A, Adhikari S, Fagania J, Chimurkar P, Karia D. E-mail spam classification via machine learning and natural language processing. Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). IEEE. 2021;693– 699.
7. Awad WA, Elseuofi SM. Machine learning methods for spam e-mail classification. International Journal of Computer Science and Information Technology (IJCSIT). 2011;3(1):173–184.
8. Zhang F, Chan PP, Biggio B, Yeung DS, Roli F. Adversarial feature selection against evasion attacks. IEEE Transactions on Cybernetics. 2015;46(3):766–777.
9. Shaukat K, Luo S, Chen S, Liu D. Cyber threat detection using machine learning techniques: A performance evaluation perspective. Proceedings of the International Conference on Cyber Warfare and Security (ICCWS). IEEE. 2020;1–6.
10. Hajek P, Barushka A, Munk M. Fake consumer review detection using deep neural networks integrating word embeddings and emotion mining. Neural Computing and Applications. 2020; 32:17259–17274.
11. Ramanathan V, Wechsler H. Phishing detection and impersonated entity discovery using conditional random field and latent Dirichlet allocation. Computers & Security. 2013; 34:123–139.
12. Ghourabi A, Mahmood MA, Alzubi QM. A hybrid CNN-LSTM model for SMS spam detection in Arabic and English messages. Future Internet. 2020;12(9):156
13. Madhavan MV, Pande S, Umekar P, Mahore T, Kalyankar D. Comparative analysis of detection of email spam with the aid of machine learning approaches. IOP Conference Series: Materials Science and Engineering. IOP Publishing. 2021;1022(1):012113.
14. Omotehinwa TO, Oyewola DO. Hyperparameter optimization of ensemble models for spam email detection. Applied Sciences. 2023;13(3):1971.
15. Apache SpamAssassin. Apache SpamAssassin public corpus. 2022
16. Enron Corp, Cohen WW. Enron email dataset. United States Federal Energy Regulatory Commission. 2015.
17. Scikit-Learn. TfidfTransformer — scikit-learn documentation. 2022.
18. Dedeturk B, Akay B. Spam filtering using a logistic regression model trained by an artificial bee colony algorithm. Applied Soft Computing. 2020; 91:106229.
19. Kumar P, Biswas M. SVM based image spam detection using kernels: linear, polynomial, RBF, and sigmoid. International Journal of Computer Science Applications. 2017;14(2):79–96.

ADDITIONAL REFERENCES

20. Rayan A. Analysis of e-mail spam detection using a novel machine learning-based hybrid bagging technique. Computational Intelligence and Neuroscience. 2022.
21. Suborna AK, Saha S, Roy C, Sarkar S, Siddique MTH. An approach to improve the accuracy of detecting spam in online reviews. Proceedings of the International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD). IEEE. 2021;296–299.
22. Frías-Blanco I, Verdecia-Cabrera A, Ortiz-Díaz A, Carvalho A. Fast adaptive stacking of ensembles.

Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2016;929–934.
23. El-Kareem A, Elshenawy A, Elrfaey F. Mail spam detection using stacking classific