IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Guardians of the Inbox
Constructing an Email Spam Classifier for Seamless Deployment

Mahinoor Sindagiri, Prof. D. A. Kulkarni, Dr. M. M. Math

Student, Principal KLS GIT Belagavi, Head of Computer Network Department

Master of Technology. Computer Science and Engineering Department,
KLS Gogte Institute of Technology, Belgaum, India

Abstract : In today's dynamic digital landscape, the proliferation of email communication has revolutionized connectivity and collaboration. However, the persistent challenge of email spam disrupts productivity, clogs inboxes, and poses security threats. In response, the fusion of Machine Learning (ML) and DevOps principles has emerged as a compelling strategy. This project explores the development and deployment of an Email Spam Classifier through ML, enriched by DevOps principles.
The dataset is pre-processed using various methods, including converting plain text into features, stemming and tokenization. Naive Bayes, a classifier utilized in supervised machine learning techniques. for the purpose of detecting spam with the utmost precision score. The approach demonstrates the potential of machine learning in identifying spam emails and enhancing the effectiveness of existing screening algorithms.
This project presents an integrated approach to combat email spam, utilizing ML algorithms and DevOps practices. The Email Spam Classifier demonstrates efficient identification, while the deployment process showcases streamlined automation and collaboration, fostering a comprehensive solution to the challenge of email spam.

IndexTerms **- Naive Bayes, Machine learning techniques, Support Vector Machine algorithm, Classifiers, DevOps, Jfrog, Gitlab.**

I. INTRODUCTION

INTRODUCTION

Over the past 10-year periods of time, internet utilization has expanded significantly and has continued to do so. Therefore, it is accurate to state that the Internet is progressively becoming a part of daily life. E-mail has developed into a potent instrument for idea and information transmission, and internet usage is predicted to continue increasing. A few of the numerous advantages associated with email has over other physical means are low prices, a small-time delay during transmission, and data security. However, a few difficulties prevent emails from being used effectively. A notable instance is spam email [3]. Numerous endeavors have been made to recognize and effectively filter out unwanted and unsolicited email communications on the client side in order to tackle the problem of spam. Numerous Machine Learning (ML) approaches have been used in earlier studies to solve the issue, including Bayesian classifiers like Naive Bayes, C4.5, and SVM, among others [2]. Bayesian classifiers were successful in these methods, leading to their widespread application in several filtering software programs. However, virtually all methods discover and learn how the feature set is distributed between unsolicited and legitimate communications.

In the present day, numerous diverse forms of spam email, such as ads with the intent of generating money or selling something, urban legends with the intent of spreading hoaxes or falsehoods, and so on. Furthermore, HTML emails may contain a web bug, which is a visual in an email message meant to track who is reading the message. As a result, even when we apply existing screening algorithms, some spam emails are classified as non-spam.

Spam and Non-Spam: As defined by Wikipedia, "spam" refers to the practice of utilizing electronic messaging systems to distribute unsolicited bulk messages, which may encompass mass advertising, malicious links, and similar content. "Unsolicited" pertains to messages received from sources without prior requests. Therefore, emails can be categorized as spam when the recipient is unfamiliar with the sender. Often, individuals are unaware that they have subscribed to such mailing lists, particularly when they have engaged with free services, downloaded software, or performed software updates.[8] This terminology was coined by Spam. Bayes in the vicinity of 2001 and is characterized as "emails that deviate from general preferences and are not classified as legitimate communication."

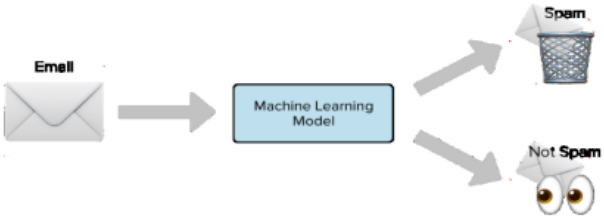

Figure 1. Detecting Spam and Non-Spam mails

The inherent significance of this project is multifold, transcending the realms of both Machine Learning and DevOps. By conjunctively harnessing these disciplines, this endeavor portrays a holistic approach to tackling the pervasive conundrum of email spam. The ML-based classifier, honed to precision through iterative learning, enables the accurate identification of spam. Concurrently, the embrace of DevOps principles ensures that deployment is fluid, continuous, and fortified against potential hitches, assuring a proactive response to the dynamic landscape of spam tactics.

In the subsequent sections of this paper, an intricate dissection of the techniques implemented in the creation of the Email Spam Classifier ensues. This is seamlessly interwoven with an exposition on the intricate embrace of DevOps paradigms, unearthing the seamless union of these disciplines. By portraying the successful confluence of ML and DevOps through the deployment of the Email Spam Classifier, this report aims to serve as a beacon, illuminating the path toward a harmonious marriage of cutting-edge technology and operational efficacy to combat the multifaceted specter of email spam.

LITERATURE SURVEY

The literature review demonstrates an extensive exploration of email spam classification techniques, spanning from conventional methods like Naïve Bayes and Support Vector Machines to advanced approaches like deep learning and ensemble methods. The research predominantly focuses on enhancing classifier performance, discussing strategies for adept feature engineering, efficient text preprocessing, and innovative dataset augmentation.

Research by A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab has been useful in identifying email spam by employing methods from the realm of machine learning. [ii] They give a concentrated literature assessment of in the fields of artificial intelligence in the fields of machine learning and artificial intelligence algorithms for identifying spam in emails. K. Agarwal and T. Kumar [4]. Mohamad & Selamat (2015) [v] and Harisinghaney et al. (2014) [5] both used the "image and collections of textual or written data set for the detection of e-mail spam using various methods." With dataset experimentation, Harisinghaney et al. (2014) [iv] used the KNN algorithm, Nave Bayes, and the Reverse DBSCAN method. OCR library" [iii] is used for text recognition, although it does not perform adequately. Mohamad and Selamat (2015)
D. Puniskis [6] employed the neural network approach in his study to classify spam. His method makes use of characteristics that are descriptive parts of the evasion patterns utilized by spammers rather than depending on the context of the message or the quantity of keywords in it. The data used is a corpus of 2788 legitimate and 1812 spam communications that were collected over a period of months. The result shows that ANN is efficient, but it shouldn't be utilized as a stand-alone spam eradication technique.
Four distinct classifiers—Neural Network, SVM classifier, Nave Bayesian Classifier, and J48 classifier—were used in [7] to categorize email data. varied data sizes and varied feature sizes were used in the experiment. If it is ultimately determined to be spam, the categorization result should be "1", otherwise, it must be "0". This study demonstrates how a straightforward J48 classifier that creates a binary tree may be effective for datasets that is capable to categorize as binary trees.

2.1 Existing System

Several techniques have been employed in the current email spam classification systems. Rule-based filtering involves creating a set of rules to flag potential spam based on keywords, sender addresses, and other characteristics. Bayesian filtering assigns probabilities to words in emails and classifies emails as spam or non-spam based on these probabilities. Machine learning methods like Naive Bayes, Support Vector Machines (SVM), and Neural Networks analyse email content, metadata, and structural features to make accurate predictions.

2.2 Proposed System

The proposed solution for email spam classification aims to overcome the limitations of existing systems by adopting the Multinomial Naive Bayes algorithm. This well-established and effective machine learning technique is widely recognized for its success in various natural language processing applications, making it a fitting choice for tackling the email spam classification challenge.

The Multinomial Naive Bayes algorithm functions on the principle of conditional probability, capitalizing on the occurrence of words and their frequencies within emails to differentiate between spam and legitimate messages. Through supervised learning on a labelled dataset containing both spam and non-spam emails, the algorithm constructs a probabilistic model. This model assigns probabilities to words and computes the probability of an email being classified as spam, based on the observed word frequencies.

The selection of the Multinomial Naive Bayes algorithm is based on its inherent simplicity, efficiency, and effectiveness—particularly in handling text-based datasets. By harnessing this algorithm and seamlessly integrating it with sophisticated feature engineering, the proposed system strives to achieve a notable level of accuracy in distinguishing between spam and legitimate emails. This endeavour is envisioned to significantly elevate user experience and bolster overall security measures.

2.3 Exploring Deployment Strategies Through a DevOps Lens

Within the realm of email spam classification, the effective deployment of the classifier stands as a critical endeavour. The implementation of DevOps methodologies presents a streamlined pathway for this deployment journey. DevOps, characterized by its fusion of development and operational practices, underscores collaboration and automation as cornerstones for achieving continuous delivery and deployment.

Numerous scholarly sources underscore the utilization of DevOps toolchains, including GitLab, JFrog Artifactory, and Jenkins, to facilitate the seamless deployment of applications. GitLab functions as a repository for version-controlled source code, facilitating efficient collaboration and comprehensive change tracking. Meanwhile, JFrog Artifactory assumes the role of an artifact repository, assuring dependable storage and version control for vital build artifacts, such as trained models and code components.
At the heart of the deployment process, Jenkins, a widely recognized automation server, plays a pivotal role. Its capabilities extend to automating essential tasks such as build compilation, rigorous testing, and the precise deployment of the email spam classifier. The integration of Jenkins and GitLab through access tokens and SSH keys fosters heightened security and ensures fluid communication between these integral platforms.

Augmenting this framework is the deployment of the email spam classifier onto an Apache server, thereby solidifying the operational flow. The process entails the meticulous configuration of the server environment, encompassing the installation of requisite dependencies and the optimization of performance parameters.

In summation, the assimilation of DevOps practices into the deployment of an email spam classifier yields enhancements across efficiency, collaboration, and security dimensions. Through the strategic utilization of tools such as GitLab, JFrog Artifactory, Jenkins, and the Apache server, complemented by the protective embrace of the Django-secured Streamlit app, a holistic and robust deployment process manifests.

DATASET DESCRIPTION

For the purpose of this research, the initial data gathered has been sourced from the online platform Kaggle. This data will serve as the foundation for training machine learning models. The original dataset was presented in the English language and acquired in the format of comma-separated values (CSV).
The email spam classifier focuses on either header, subject, and content of the email. In this project, we are focusing mainly on the content of the email.

This model utilizes E-Mail-Datasets from several Internet-Sources, such as Kaggle and sklearn, as well as certain Datasets produced by the author.

A Kaggle spam email collection is used for training our model, while another email dataset set is utilized to obtain results. The "spam.csv" data set has 5574 rows and two columns. (Text, Target). The target feature consists of two classes ham and spam, the column name is target.

The classes are labelled for each document within the collection of data and represent our target feature with a binary string-type alphabet of {ham; spam}. Classes are additionally linked to integer 0 (ham) and 1 (spam).

METHODOLOGY

4.1 Data Preprocessing

The electronic mails within the training data collection are initially provided in plain text format. To effectively process these Emails, we must transform the plain text into features that can adequately represent the content. These features will subsequently enable us to apply a learning algorithm to analyse the emails.
To achieve this, several pre-processing steps are conducted. The plain text files are converted into files wherein each line contains a single word. In this particular project, we approach emails as a compilation of individual words.

4.2 Stop Words

Certain English vocabulary items tend to occur frequently across all documents, yet they hold limited significance in terms of representation. These words are commonly referred to as "stop words," and removing them does not compromise the overall meaning. Examples of such words include "the," "a," and "for," among others. Moreover, there exist domain-specific stop words relevant to the context of emails, such as "Mon," "Tue," "email," "sender," and "from."
Therefore, it is advantageous to eliminate these stop words from all files within the dataset.

4.3 Stemming

The subsequent task to be executed involves stemming. It is employed to identify the base form of a word, thus substituting all words with respective stem, thereby decreasing the number of words to be taken into account when representing a text or content. In the project, we use the Python Porter Stemmer algorithm.

4.4 Tokenization

Tokenization refers to the process of dividing a sequence of text, whether it's a manuscript or any other form of content, into segments such as phrases, symbols, words, or other expressive components known as tokens. These tokens serve as the fundamental units used for subsequent processing tasks, like content mining and parsing. Tokenization holds significance in both semantics, where it aids in segmenting content, and in lexical analysis within the fields of computer science and engineering.
The concept of a "word" can sometimes be challenging to precisely define, especially as tokenization operates at the word level. Often, the criteria for creating tokens are based on simple rules. For example, tokens can be separated by whitespace characters, such as "line breaks" or "spaces," as well as by "punctuation characters."

In this process, any contiguous sequence of alphabetic characters or numbers is considered a single token. The inclusion of white spaces and punctuation marks in the resulting token lists can vary.
Numbers and alphabetic characters both consist of a single token that is made up of all nearby strings of characters. In the resultant lists of tokens, white spaces along with punctuation could be or might not be present.

4.5 Naive Bayes

Starting from the year 1998, Naive Bayes has been harnessed within the realm of supervised machine learning to discern and label spam messages [3]. This method predominantly hinges on its capacity to differentiate between distinct entities grounded on predetermined attributes. Naive Bayes captures the essence of a word or an event that transpired within a preceding context and calculates the probability of that word or event reoccurring in subsequent instances. For instance, when a word emerges in a spam email but is absent in legitimate correspondence, the algorithm is inclined to classify it as a spam indicator.

$$P(c/x) = (P(x/c)\ P(c)) / (P(x)),$$

$$P(x) = \sum_y P(x/c)\ P(c).$$

In this context, X represents a collection of feature vectors, while C symbolizes a variable denoting different possible outcome. The notation P(c/x) refers to the probability of an event occurring in the future, P(x/c). P(c) signifies the preliminary probability, and P(x) represents the evidence grounded in feature variables.

4.5.1 Naïve Bayes Algorithm

a) Input:
Training dataset with labelled examples.
New, unseen example to be classified.

b) Preprocessing:
Clean and preprocess the text data (if applicable). Convert text data into numerical features (e.g., Bag of-Words or TF-IDF representation).

c) Training:
Calculate class priors: P(Class) = Count of examples in Class / Total number of examples.
For each feature:Calculate likelihoods: P(Feature|Class) = Count of Feature occurrences in Class / Total count of Features in Class.

d) Classification:
For a new example: For each class: Calculate posterior probability using Bayes' theorem:
P(Class Features) = P(Class) * P(Feature1|Class) * P(Feature2|Class) * P(FeatureN|Class).
Choose the class with the highest posterior probability as the predicted class.

IMPLEMENTATION

The model is implemented using the Visual Studio Code platform, and the training dataset in this module is taken from the "Kaggle" website. To improve machine performance, the entered dataset is first examined for duplication and null values. The dataset is subsequently split into two more compact datasets, say the "train dataset" and "test dataset," with a ratio of 70:30. The "train" and "test" datasets are then provided as text-processing input parameters. Punctuation marks and terms on the list of stop words is extracted out during text processing and replaced with clean words. The term "Feature Transform" is then passed using these clear phrases. In features transform, the clean keywords that the text-processing provided are then utilized for "fit" and "transform" to build the machine's vocabulary. Additionally, the collection of data is subjected to "hyperparameter tuning" to ascertain the best classifier settings derived from the dataset. The characteristics and state of the model that was trained are retained for use in testing new data in the future. Machines are trained with the values gleaned from above using classifiers from Python's sklearn package.

5.1 Strengthening Streamlit App Security via Integration with Django

Streamlit and Django Integration: Integrate the Streamlit app, which serves as the classifier's user interface, with Django. Django provides robust security features and user authentication.
User Authentication: Utilize Django's user authentication system to control access to the Streamlit app. This prevents unauthorized access and ensures data privacy.

5.2 Setting Up the Environment

GitLab Setup: Create a GitLab repository to host the email spam classifier's source code. Utilize GitLab's version control and collaborative features to manage the development process effectively.
JFrog Artifactory: Configure JFrog Artifactory to act as a repository for storing build artifacts, such as trained models and classifier code. This enables seamless artifact management and versioning.

5.3 Developing the Email Spam Classifier

Python Programming: Implement the email spam classifier using appropriate machine learning libraries like Scikit-learn or TensorFlow. Train the model using a relevant dataset, fine-tune its parameters, and ensure high accuracy.
Git Workflow: Utilize Git's branching and merging features to maintain a clean development workflow. Regularly commit and push changes to the GitLab repository.

5.4 Jenkins Automation

Jenkins Integration: Set up Jenkins to automate the deployment process. Configure Jenkins jobs to trigger on code changes in the GitLab repository.

Access Tokens: Establish GitLab integration with Jenkins using access tokens. Generate a GitLab access token and use it in Jenkins for secure communication between the two platforms.

SSH Key Integration: Implement SSH key authentication for secure access to GitLab repositories. Jenkins can use the SSH key to securely pull code and initiate builds.

5.5 Deploying on Apache Server

Server Configuration: Prepare the Apache server to host the email spam classifier. Install the required dependencies and configure the server to handle incoming requests.

Continuous Deployment: Configure Jenkins to automatically deploy new versions of the classifier to the Apache server. This ensures that the latest version is always available to users.

II. RESULTS AND DISCUSSION

To ensure higher accuracy, model underwent training through many classifiers, which were then checked and compared. The user will receive each classifier's assessed results.

Comparison chart - accuracy and precision

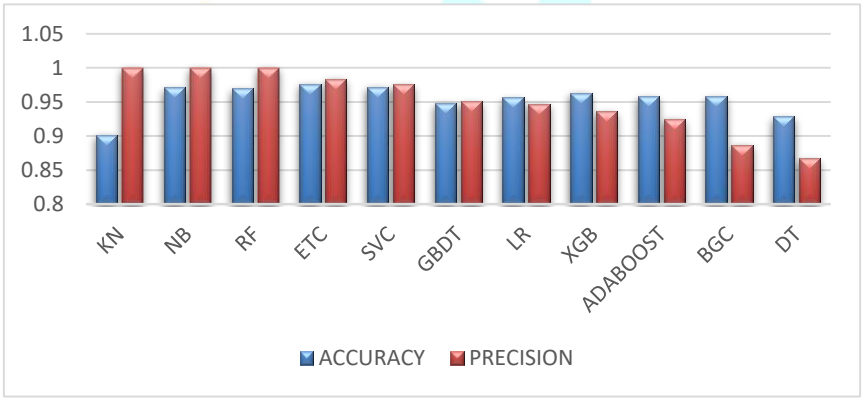

Figure 2. Algorithms accuracy and precision in sorted manner

From the Above Comparison chart, we understand that the Naïve bayes (NB), Random Forest (RF) and K-Neighbour Algorithm has highest precision value (precision refers to the excellence of a positive prediction rendered by the model) and decision tree (DT) has lowest precision and accuracy. Blue bar indicates the Accuracy and orange bar indicate the Precision.

Comparison Chart - Involving various range of attributes and learning algorithms.

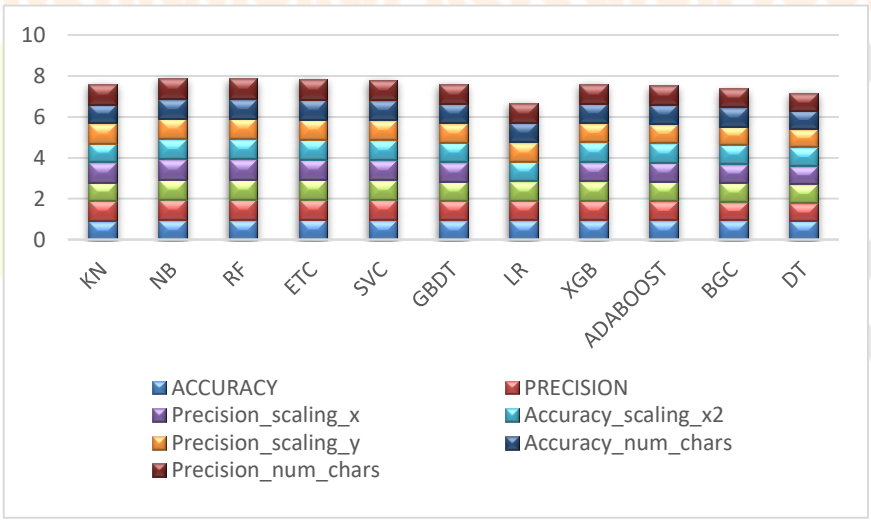

Figure 3. Experimental Results

From trials involving various range of attributes and learning algorithms, we examine the classification accuracy of the constructed models on the testing dataset. Subsequently, we proceed to compare the outcomes.

CONCLUSION

The pervasiveness of email communication in the ever-changing digital ecosystem has transformed the way we communicate and cooperate. Nonetheless, the ongoing annoyance of spam emails has disrupted productivity, clogged inboxes, and

subjected users to dangers related to security. As a result, a combination of ML (machine learning) and DevOps approaches has emerged as a possible option. This project began with a thorough examination of designing and implementing an Email Spam Classification via the lens of ML, bolstered by DevOps concepts.

The primary goal of this project encompassed two pivotal aspects: constructing a robust Email Spam Classifier using a diverse range of ML algorithms to adeptly discern between legitimate messages and spam, and demonstrating the practical application of DevOps practices in orchestrating the seamless deployment and ongoing maintenance of the classifier. This harmonious synergy aspired to achieve not only precise spam detection but also the seamless integration of ML advancements with operational workflows, resulting in a dynamic and adaptive solution.

The Naïve Bayes method gives the highest classification accuracy no matter how many attributes are used and which method is used. We also see that on average the accuracy improves as the number of attributes increase. It is possible that the accuracy may increase more than 97.10 % if we train the model by further increase the number of attributes.

By seamlessly integrating technological advancements with operational efficiency, this project not only offered a robust email spam classification system but also illuminated a path toward a harmonious blend of innovation and practicality in addressing the multifaceted issue of email spam.

REFERENCES

[1] Witten, Frank, E., and I. (2000). "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations." Published by Morgan Kaufmann.

[2] M. Embrechts, B. Szymanski, K. Sternickel, T. Naenna, and R. Bragaspathi, 2003. Utilization of Machine Learning for Classification of Magnetocardiograms, Proc. IEEE Conference on System, Man and Cybernetics, Washington DC.

[3] C. Pu and S. Webb. 2006. Observed trends in spam construction techniques: An Illustrative Case Study of spam evolution. In Proceedings of the 3rd Conf. on EMail and Anti-Spam.

[4] Agarwal, K., & Kumar, T. (2018). "Detection of Email Spam Using an Integrated Approach of Naïve Bayes and Particle Swarm Optimization." In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 685-690). Madurai, India.

[5] Harisinghaney, A., Dixit, A., Gupta, S., & Arora, A. (2014). "Classification of Spam Emails using KNN, Naïve Bayes, and Reverse DBSCAN Algorithm for Text and Image-Based Content." In Proceedings of the 2014 International Conference on Optimization,Reliability,and Information Technology (ICROIT) (pp. 153-155). IEEE.

[6] D. Puniškis, R. Laurutis, R. Dirmeikis, An Artificial Neural Nets for Spam e- mail Recognition, electronics and electrical engineering ISSN 1392 – 1215 2006. Nr. 5(69)

[7] Bekker, S 2003, Spam to Cost U.S. Companies $10 Billion in 2003, ENT News, viewed May 11 2005,

[8] Youn and Dennis McLeod, “"A Comparative Analysis of Email Classification Techniques", Seongwook Los Angeles” , CA 90089, USA, 2006.

[9] IEEE 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)

[10] L. Bass, I. Weber, and L. Zhu, DevOps: A software architect’s perspective. Addison-Wesley Professional, 2015.

[11] G. Kim, J. Humble, P. Debois, and J. Willis, The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution, 2016

[12] 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) :: An Empirical Taxonomy of DevOps in Practice, Ruth W. Macarthy School of Science, Engineering and Environment University of Salford Manchester, UK r.w.macarthy@edu.salford.ac.uk and Julian M. Bass School of Science, Engineering and Environment University of Salford Manchester, UK j.bass@salford.ac.uk.