Email/SMS Spam Classifier Using Machine Learning Algorithms

Ms. Nidhi Malik[1], Lakshay Dabas[2], Akshay Rathee[3], Krish Panwar[4] , Himanshu[5]
[1] Assistant Professor [2,3,4,5] Student CSE Department
HMR Institute of Technology and Management, GSSIPU, Delhi 110036
akshayrathee902@gmail.com

Abstract

These days, email spam has grown to be a significant issue due to the rapid increase in internet users. They are being used for immoral and unlawful purposes, practices, fraud, and phishing. sending harmful links in spam emails that have the potential to compromise both your system and ours. It is very simple for spammers to create a phony profile and email account; they pose as real people in their spam emails and target those who are unaware of these scams. Therefore, it is necessary to identify spam emails that are fraudulent. This project will do so by employing machine learning techniques. This paper will go over machine learning algorithms and apply each one to our data sets, choosing the one with the highest precision and accuracy for email spam detection.

Keywords: Support Vector Machine (SVM), Naive Bayes, Linear Regression , spam detection, tokenization, stop words removal, machine learning for text classification, error minimization, linear regression for feature weights, regression line, Naive Bayes classifier for spam, text classification with Naive Bayes

Introduction

Email spam, or electronic mail spam, refers to the practice of sending unsolicited emails or advertisements to multiple recipients. Unsolicited emails imply that the recipient has not given consent to receive these messages. The prevalence of spam emails has been increasing over the last decade. Spam has become a significant nuisance on the internet. It leads to wasted storage, time, and efficiency in communications. While automatic email filtering is often considered the most effective way to identify spam, spammers have become adept at circumventing these filtering systems. In the past, it was typically possible to manually block most spam based on specific email addresses. A machine learning approach will be utilized for identifying spam. Key methods used for filtering junk mail include “text analysis, white and blacklists of domain names, and community-based strategies.” Text analysis of email content is a widely adopted technique for identifying spam. Numerous solutions are available for both server-side and client-side deployment.

Naive Bayes is among the most recognized algorithms utilized in these processes. However, filtering out emails primarily based on content analysis can be challenging due to the issue of false positives. Generally, neither users nor organizations want any legitimate messages to be lost. The blacklist approach has been one of the earliest methods used for spam filtering. This method involves accepting all emails except those from explicitly banned domains or email addresses. As new domains emerge that fall into the category of spamming, this method no longer functions as effectively.

The whitelist approach involves accepting emails from openly whitelisted domain names or addresses while placing others in a much lower priority queue, which is only addressed after the sender responds to a verification request sent through the "spam filtering system.".

Spam and Ham: As defined by Wikipedia, “spam refers to the use of electronic messaging systems to send unsolicited bulk communications, particularly mass advertisements, harmful links, and similar content.” The term “unsolicited” signifies that these messages are not requested by the recipient. Therefore, if the sender is unknown, the email may be spam. People often fail to recognize that they inadvertently subscribed to these mailings when they install free services, software, or during software updates. The term “ham” was introduced by Spam Bayesian filters around 2001 and is described as “emails that are not wanted but do not qualify as spam.”

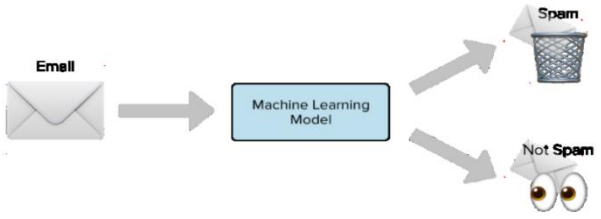

Fig.1. Classification into Spam and non-spam

Literature Review

Several studies have applied machine learning techniques to detect email spam, including work by A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab. They provide a targeted literature overview of artificial intelligence and machine learning approaches for identifying email spam. K. Agarwal and T. Kumar, along with Harisinghaney et al. (2014) and Mohamad & Selamat (2015), utilized both image and text datasets for email spam detection using various techniques. In their 2014 study, Harisinghaney et al. implemented the K-Nearest Neighbor (KNN) algorithm, Naïve Bayes, and the Reverse DBSCAN algorithm through experimentation on their dataset. For the text recognition component, they employed an OCR library, although this particular OCR showed subpar performance. Mohamad & Selamat (2015) adopted a hybrid feature selection method based on TF-IDF (Term Frequency-Inverse Document Frequency) alongside Rough pure mathematics.[1][2]

Data Set

This model utilizes email datasets sourced from various online platforms such as Kaggle and sklearn, along with some datasets created independently. A spam email dataset from

Kaggle is employed to train our model, while a different email dataset is utilized to obtain results. The “spam.csv” dataset consists of 5,573 entries and 2 columns, whereas the other datasets comprise 574, 1,001, and 956 records of email data in text format. Further research focuses on how well different machine learning algorithms and data preprocessing methods detect spam. For example, X. Zhang et al. (2018) compared models like ensemble techniques, Decision Trees, and Support Vector Machines (SVM) and found that ensemble methods increased classification accuracy on large datasets. The use of Convolutional Neural Networks (CNN) to classify spam based on textual and image-based features was also investigated by Y. Shen et al. (2019), who showed enhanced performance on multi-modal spam datasets.[3][4]

In recent years, deep learning methodologies have gained significant popularity. For instance, F. Li and M. Wu (2020) explored the application of Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks to effectively capture the sequential characteristics of textual data in spam detection, particularly focusing on emails with complex language structures. They incorporated pre-trained word embeddings to improve the performance of their models and discovered that LSTM networks yielded higher recall rates when applied to imbalanced datasets.Data Augmentation and Preprocessing Numerous studies have highlighted the critical role of data preprocessing in spam detection systems. Textual data often necessitates the application of techniques such as tokenization, stop-word elimination, and lemmatization to ensure that relevant features are utilized. Notably, Chen and Lee (2019) implemented data augmentation strategies, including synonym replacement and random insertion, to enhance their dataset and bolster model resilience.[5][6][7]

The primary dataset referenced in various studies is "spam.csv" (Kaggle), which comprises 5,573 messages categorized as spam or ham. Other frequently utilized datasets include the "Enron" dataset, which offers a comprehensive email corpus, and the "Ling-Spam" corpus, which is used for traditional text-based spam detection. Each of these datasets has played a vital role in the advancement of robust and generalizable spam detection models.[8][9]

Methodology

This section provides an in-depth overview of key reinforcement learning (RL) techniques applied in the training of autonomous agents, particularly Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) (This technique is considered but not provided expected results), along with a brief overview of RL and imitation learning. These methods are foundational within Unity’s ML-Agents toolkit, enabling robust training environments for agents through either reinforcement or learning by imitation.

Data Preprocessing

In the realm of data analysis, one often encounters extensive datasets characterized by a significant number of rows and columns. However, it is important to recognize that data can exist in various formats, including images, audio, video files, and structured tables. Machines do not inherently comprehend images, videos, or textual data; they interpret information solely in binary form, represented as 1s and 0s.

The steps involved in Data Preprocessing are as follows:

Data Cleaning: This phase involves tasks such as addressing "missing values," "smoothing out noisy data," "identifying or eliminating outliers," and "resolving inconsistencies."

Data Integration: This step entails the amalgamation of multiple databases, information files, or datasets.

Data Transformation: This process includes aggregation and normalization to adjust data to a specific scale.

Data Reduction: This aspect focuses on generating a concise summary of the dataset that retains the ability to yield equivalent analytical results despite its reduced size.

1. Stop Words: Stop words refer to common English words that do not significantly contribute to the meaning of a sentence. These words can be disregarded without losing the overall sense of the statement. For instance, when searching for a query such as "How to make a veg cheese sandwich," the search engine will look for web pages containing the terms "how," "to," "make," "a," "veg," "cheese," and "sandwich." The search engine prioritizes pages that include the terms "how," "to," and "a," which are frequently used in the English language. If these three words are omitted, the search engine can focus on retrieving pages that contain the keywords "veg," "cheese," and "sandwich," thereby yielding more relevant results.

2. Tokenization: Tokenization is the procedure of dividing a stream of text into phrases, symbols, words, or any other meaningful elements referred to as tokens. The resulting tokens are then utilized for further processing, such as text mining and parsing. Tokenization plays a crucial role in both semantics, where it is known as text segmentation, and in lexical analysis within computer science and engineering. Defining the term "word" can sometimes be challenging, as tokenization occurs at the word level. Often, a token relies on simple heuristics; for example, tokens are separated by whitespace characters, such as line breaks or spaces, or by punctuation marks. Each contiguous string of alphabetic characters constitutes a single token, as do numbers. Whitespace and punctuation may or may not be included in the final list of tokens.

3. Bag of Words: The Bag of Words (BOW) model is a technique for extracting features from text documents, which can subsequently be used to train machine learning algorithms. This method generates a vocabulary comprising all unique words found across the documents in the training set.

CLASSIC CLASSIFIERS

Classification represents a method of data analysis aimed at identifying models that characterize significant data categories. A classifier, or model, is developed to predict class labels, such as determining whether "a loan application is risky or safe." The process of data classification consists of two primary phases:

- the learning phase, which involves the construction of the classification model, and

- the classification phase.

1. NAÏVE BAYES:

The Naïve Bayes classifier was first employed in 1998 for the purpose of spam detection. This algorithm is utilized in supervised learning contexts. The Bayesian classifier operates on dependent events and assesses the likelihood of future occurrences based on previously observed events. The foundation of Naïve Bayes lies in Bayes' theorem, which presumes that the features are independent of one another. This technique is known as the Naïve Bayes classifier. The classification of spam emails can effectively utilize word probability, which plays a crucial role in this process. If a particular word appears frequently in spam emails but rarely in legitimate ones, the email in question is likely to be classified as spam. The Naive Bayes classifier algorithm has emerged as a leading technique for email filtering. To achieve optimal performance, the model is trained using the Naïve Bayes filter. This algorithm consistently calculates the probability of each class, selecting the class with the highest probability as the output. Naive Bayes is known for delivering accurate results and is widely applied in various domains, including spam filtering.

2. SUPPORT VECTOR MACHINE:

The Support Vector Machine (SVM) is a widely recognized supervised learning algorithm utilized for classification tasks within machine learning methodologies. The foundation of the Support Vector model is based on the concept of decision points. The primary objective of the Support Vector Machine algorithm is to establish a decision boundary or line. The output of the Support Vector Machine algorithm is a hyperplane that classifies new data samples. In a two-dimensional space, this hyperplane serves as a line that divides the plane into two distinct regions, with each class occupying one side.

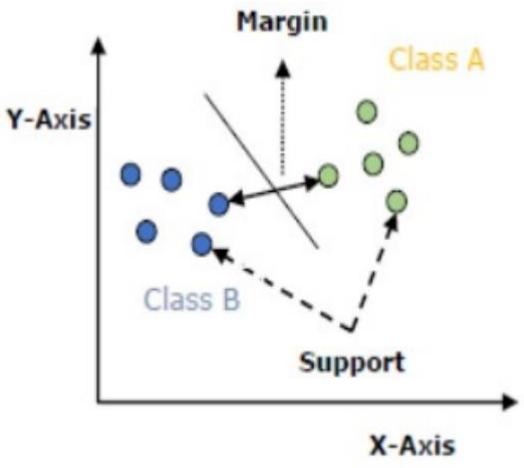

Fig.2 Support Vector Machine

3. DECISION TREE:

Decision tree induction involves the process of constructing a decision tree from training tuples that are labelled with class information. A decision tree is structured like a flowchart, where:

- An internal node or non-leaf node represents a test on a specific attribute.

- A branch indicates the outcome of that test.

- A leaf node contains a class label.

- The top node is referred to as the root node.

The construction of decision tree classifiers does not require any specialized domain knowledge or parameter adjustments suitable for knowledge examination. This method effectively manages multidimensional data. The learning and classification processes associated with decision tree induction are both straightforward and efficient. Key attribute selection events are employed to determine the attribute that best divides the tuples into distinct classes. During the construction of the decision tree, many branches may reflect noise, and anomalies present in the training data. Tree pruning aims to identify and eliminate such branches to enhance the classifier's accuracy on unseen data.

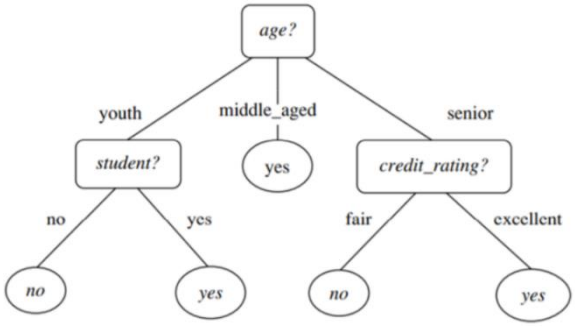

Fig.3. Decision Tree Structure

4. K- NEAREST NEIGBOUR:

K-nearest neighbours is a supervised classification algorithm that utilizes a set of data points and vectors categorized into various classes to determine the classification of new sample points. This algorithm is characterized as a "lazy" algorithm, meaning it primarily relies on memorization of the training data rather than actively learning from it. Consequently, it does not independently make decisions. The K-nearest neighbours algorithm classifies new points by employing a similarity measure, such as Euclidean distance, to identify its nearest neighbours. The formula for calculating the Euclidean distance is given by $dist.((x, y), (a, b)) = \sqrt{((x - a)^2 + (y - b)^2)}$.

ENSEMBLE LEARNING METHODS

Ensemble methods in machine learning refer to techniques that combine multiple base models to generate a predictive model aimed at reducing variance through bagging, bias through boosting, and enhancing predictions via stacking. There are two primary types of ensemble methods: Sequential, where base classifiers are developed in a sequence, and Parallel, where base classifiers are constructed simultaneously.

1. RANDOM FOREST CLASSIFIER

The random forest classifier is an ensemble tree-based model that comprises various types of decision trees, each differing in shape and size. It employs random sampling of the training data during the tree-building process and utilizes random

subsets of input features when making splits at each node. This randomness reduces the correlation among decision trees, thereby improving the generalization error of the ensemble, as the features of the trees will not appear identical.

2. BAGGING

The bagging classifier is an ensemble technique that trains base classifiers on random subsets of the original dataset and subsequently combines their individual predictions through voting or averaging to produce a final outcome. Bagging is a combination of bootstrapping and aggregating. The bootstrapping process helps to reduce the variance of the classifier and mitigates overfitting by resampling the training data while maintaining the same cardinality as the original dataset. High variance is detrimental to model performance. Bagging proves to be an effective approach, particularly with limited data, as it allows for estimation through the aggregation of scores.

3. BOOSTING AND ADABOOST CLASSIFIER

Boosting is an ensemble method designed to construct a robust classifier by utilizing several weak classifiers. The process involves creating a model from the training dataset, followed by the development of additional models that aim to correct the errors of the preceding model. This iterative process continues until the training set is accurately predicted. AdaBoost, or Adaptive Boosting, is the first successful boosting algorithm specifically developed for binary classification. The concept of boosting is characterized by its ability to enhance model performance through successive iterations.

IV. ALGORITHMS

1.1. Input the dataset or file intended for training or testing.

1.2. Verify the dataset for compatible encoding.

1.2.1. If the encoding is among those supported, proceed to step 1.4.

1.2.2. If the encoding is not supported, move to step 1.3.

1.3. Convert the encoding of the uploaded file to one of the supported formats. Then attempt to read it again.

1.4. Choose whether to “Train,” “Test,” or “Compare” the models utilizing the dataset.

1.4.1. If “Train” is chosen, advance to step 1.5.

1.4.2. If “Test” is chosen, proceed to step 1.6.

1.4.3. If “Compare” is chosen, go to step 1.7.

1.5. If “Train” is selected:

1.5.1. Determine which classifier will be trained with the provided dataset.

1.5.2. Inspect for duplicate entries and NAN values.

1.5.3. Identify values through Hyperparameter Tuning.

1.5.4. Prepare the text for feature transformation.

1.5.5. Execute the model training.

1.5.6. Store the model and features, and present the results.

1.5.7. Choose which classifier to evaluate using the provided dataset.

1.5.8. Check for duplicates and NAN values again.

1.5.9. Retrieve the model and features saved during the training phase.

1.5.10. Utilize the retrieved values to test the dataset.

1.5.11. Present the results.

1.6. If “Compare” is selected:

1.6.1. Evaluate all classifiers using the provided dataset.

1.6.2. Display the results of the classifiers.

A. Implementation

The Visual Studio Code platform serves as the environment for executing the model, utilizing a dataset sourced from the "Kaggle" website as the training dataset. Initially, the dataset undergoes a thorough examination to identify and eliminate any duplicates and null values, thereby enhancing the machine's performance. Subsequently, the dataset is divided into two subsets, referred to as the "train dataset" and the "test dataset," in a ratio of 70:30. These subsets are then utilized as parameters for text processing. During the text processing phase, punctuation marks and words identified as stop words are removed, resulting in a collection of clean words. These clean words are subsequently employed in the "Feature Transform" stage, where they are utilized for fitting and transforming to establish a vocabulary for the machine. Additionally, the dataset is subjected to "hyperparameter tuning" to determine the optimal values for the classifier based on the dataset characteristics. Once the values from the hyperparameter tuning process are obtained, the machine is trained using these values along with a specified random state. The state of the trained model and its features are preserved for future applications involving unseen data. The classifiers from the sklearn module in Python are employed to train the machines using the previously acquired values.

RESULT

Our model has undergone training with various classifiers to evaluate and compare results for enhanced accuracy. Each classifier provides its assessed outcomes to the user. Once all classifiers have delivered their results, the user can juxtapose these findings to determine whether the data is classified as "spam" or "ham." The results from each classifier will be presented in graphical and tabular formats for improved comprehension. The dataset utilized for training was sourced from the "Kaggle" website, specifically the dataset titled "spam.csv." To assess the performance of the trained model, a separate CSV file containing unseen data, referred to as "emails.csv," has been created. After completing the text editing process, the document is prepared for the template. Duplicate the template file using the Save As command and adhere to the naming conventions specified by your conference for your paper's title. In the newly generated file, select all content and import your prepared text file. You are now set to format your paper; proceed with the scrolling.

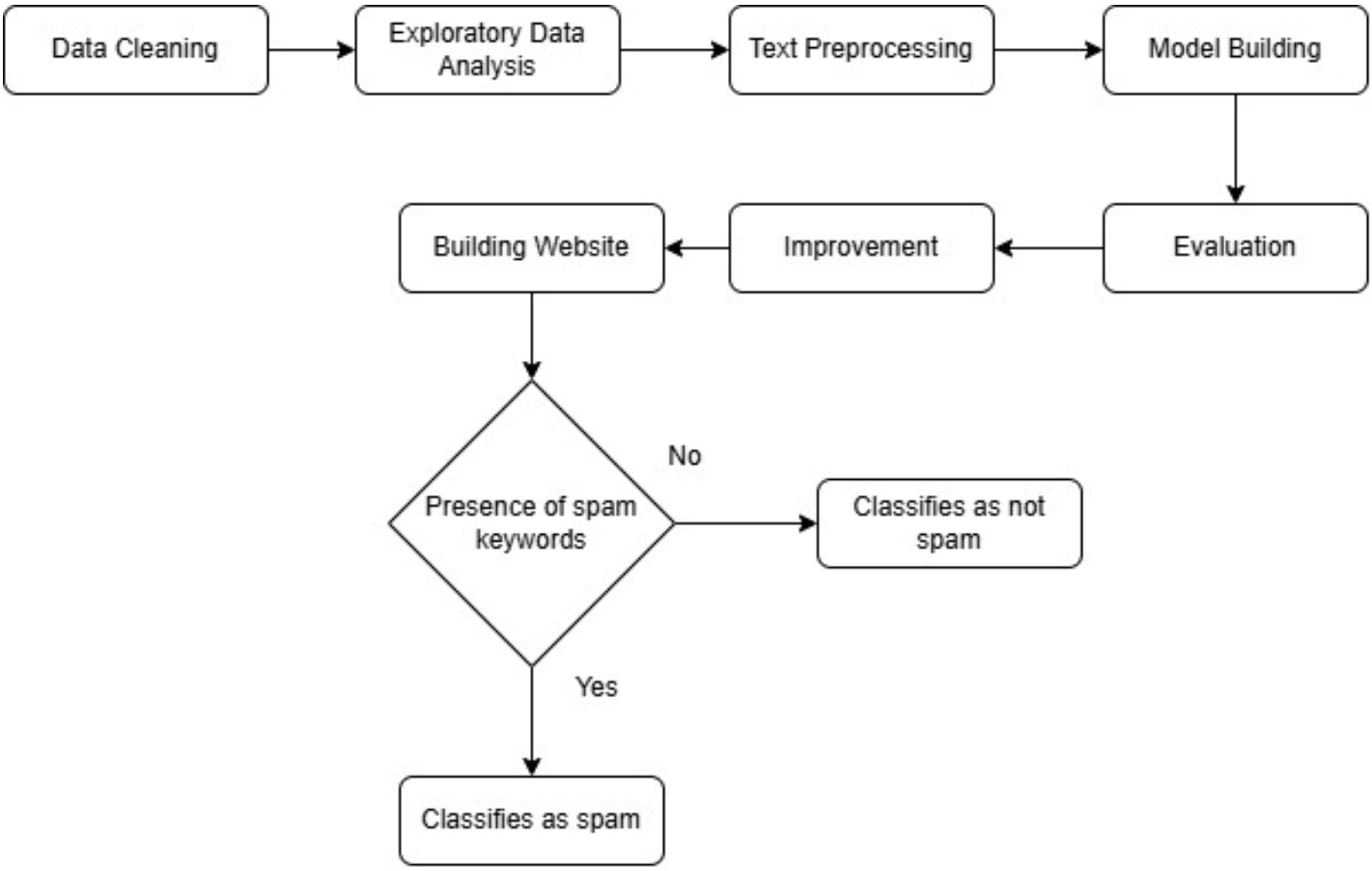

Fig. 4 Flowchart

TABLE I. COMPARISION TABLE

	Classifiers	*Score 1*	*Score 2*	*Score 3*	*Score 4*
1	Support Vector Classifier	0.81	0.92	0.95	0.92
2	K-Nearest Neighbour	0.92	0.88	0.87	0.88
3	Naïve Bayes	0.87	0.98	0.98	0.98
4	Decision Tree	0.94	0.95	0.93	0.95
5	Random Forest	0.90	0.92	0.92	0.92
6	AdaBoost Classifier	0.95	0.94	0.95	0.94
7	Bagging Classifier	0.94	0.94	0.95	0.94

a. score 1: using default parameters
b. score 2: using hyperparameter tuning
c. score 3: using stemmer and hyperparameter tuning
d. score 4: using length, stemmer and hyperparameter tuning

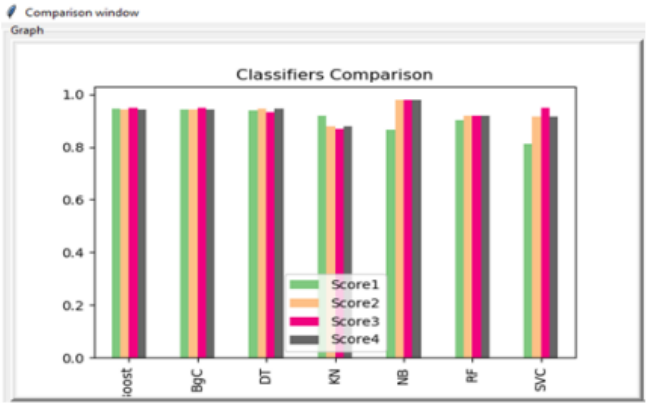

Fig.5 Comparison of all algorithms

Conclusion

The findings indicate that the Multinomial Naïve Bayes model yields the most favorable results; however, it is constrained by the assumption of class-conditional independence, which can lead to the misclassification of certain instances. In contrast, ensemble methods have demonstrated their effectiveness by employing multiple classifiers for the purpose of class prediction. Given the substantial volume of emails exchanged today, our project faces challenges as it can only evaluate emails based on a limited corpus. Consequently, our spam detection system is adept at filtering emails based on their content rather than relying on domain names or other criteria. This results in a restricted scope of email analysis. There exists significant potential for enhancement in our project, and the following improvements could be implemented:

1. Spam filtering could be enhanced by utilizing trusted and verified domain names.

2. The classification of spam emails is crucial for effectively categorizing emails and distinguishing between spam and non-spam messages.

3. This approach can be adopted by larger organizations to identify legitimate emails that align with their desired communications.

References

[1] Shubhangi Suryawanshi, Anurag Goswami, and Pramod Patil. (2019). An empirical comparison of various machine learning and ensemble classifiers for email spam detection. 10.1109/IACC48062.2019.8971582. 69–74.

[2] Alazab, M., Shanmugam, B., Krishnan, K., Karim, A., and Azam, S. (2019). An Extensive Analysis for Perceptive Spam Email Identification. [08907831] IEEE Access, 7, 168261-168295. 10.1109/ACCESS.2019.2954791 https://doi.org

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

[4] Anuja Arora, Saurabh Gupta, Aman Dixit, Harisinghaney, and Anirudh. "Text and image-based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm." In the 2014 International Conference on Optimization, Reliability, and Information Technology (ICROIT), pages 153–155. IEEE (2014)

[5] ELseuofi, W.A., Awad, & S.M. (2011). Machine Learning Techniques for Classifying Spam Emails. 3. 10.5121/ijcsit.2011.3112 International Journal of Computer Science & Information Technology.

[6] Ameen and Kaya, A. K., "Deep learning for spam detection in online social networks," 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 2018, pp. 1-4.

[7] Hatipoglu, T., Selvi, I.H., Boran, S., & Diren, D.D. (2019). Root Cause Identification in the Multivariate Manufacturing Process Using an Ensemble Machine Learning Approach.

[8] Atif Hasan Rahman, Abida Sanjana Shemonti, and Tasnim Kabir.IEEE 18th International Conference on Publication Principles, "Notice of Violation: Species Identification Using Partial DNA Sequence: A Machine Learning Approach," 2018

[9] "An evaluation on the efficiency of hybrid feature selection in spam email classification" by Mohamad, Masurah, and Ali Selamat. International Conference on Computer, Communications, and Control Technology (I4CT), 2015, pp. 227–231. IEEE, 2015

[10] "E-mail Spam Detection and Classification Using SVM and Feature Extraction" in the International Journal of Advance Research, Ideas, and Innovation by Shradhanjali and Prof. Toran Verma. Ali Selamat, Masurah, and Mohammed. Spam Detection and Classification" in the International Journal of Advance Research, Ideas, and Innovation