A Multi-Algorithm Approach to Detect Spam Emails Using ML

Aniket Kumar
School of Computer Science & Engineering
Galgotias University
Greater Noida, India
aniket.23SCSE2160024@galgotiasuniversity.ac.in

Gaurav Kumar
School of Computer Science & Engineering
Galgotias University
Greater Noida, India
gaurav.23SCSE2160077@galgotiasuniversity.ac.in

Karan Pandey
School of Computer Science & Engineering
Galgotias University
Greater Noida, India
Karan.23SCSE2160059@galgotiasuniversity.ac.in

Raju Ranjan
School of Computer Science & Engineering
Galgotias University
Greater Noida, India
drraju.ranjan@galgotiasuniversity.edu.in

***Abstract*— Email spam detection is a crucial task in today's digital age. This study presents a multi-algorithm approach to detect spam emails using Machine Learning (ML). We employ ten different algorithms: as Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Random Forest, AdaBoost Classifier, Extra Trees Classifier, Gradient Boosting Classifier, and XGBoost Classifier. Each model's performance is evaluated based on accuracy and precision metrics. Our results show varying degrees of effectiveness across the models, with some excelling in accuracy and others in precision. After comparing the performance of all models, we select the best-performing and well performing model based on accuracy and precision. The top-performing model is then integrated into a GUI-based system, which classifies emails as spam or non-spam. This approach enables the identification of the most effective and most correct ML algorithm for spam email or sms detection, leading to improved email filtering and enhanced user experience. This study improves spam detection by analyzing various machine learning algorithms' strengths and weaknesses. Combining their advantages enables the creation of a more effective and reliable system, reducing unwanted emails' impact on individuals and organizations. Ultimately, this research aims to advance the field of email spam detection, enabling the creation of more sophisticated and effective spam filtering solutions. This approach enables the identification of the most effective ML algorithm for spam email detection, leading to improved email filtering and enhanced user experience. Our research aids in the advancement of effective spam detection systems, providing insights into the strengths and weaknesses of various ML algorithms in tackling this critical task and give idea about how different-different algorithm work on Text dataset like accuracy and precision.**

Keywords— Machine Learning, SMS, Spam and Non-Spam, Text Mining, Algorithms

I. Introduction

The deluge of unwanted emails in our inboxes has become an all-too-familiar nuisance in today's digital landscape, with spam emails not only clogging our email servers but also posing significant security risks, from phishing scams to malware attacks. As email adoption accelerates, the urgency for advanced spam detection measures intensifies, since traditional rule-based systems are ill-equipped to counter the escalating sophistication of spamming strategies. This By 2005, an amount more than is in Machine Learning comes into play, offering a beacon of hope in the quest for spam-free inboxes. Leveraging the capabilities of machine learning algorithms enables the creation of adaptive systems that can accurately identify and filter spam emails. However, the vast selection of ML algorithms presents a significant challenge.: which algorithm reigns supreme in this quest? This study aims to investigate this question through an exhaustive analysis of ten leading machine learning algorithms, with the ultimate goal of crafting a cutting-edge spam detection system that bolsters email security and user experience, and provides valuable insights into the strengths and weaknesses of various ML algorithms in tackling this critical task.

II. Related Work

Unwanted online content hurts mobile businesses and users. Researchers studied this problem in school networks to find solutions. They checked web pages to identify bad content and analyzed unwanted messages. A company collected data to create better filters. Using powerful computers and smart algorithms made their solutions stronger. Now, computers can easily tell good content from bad. This helps keep online spaces clean and safe.

III. METHODOLOGY

This study employs a multi-algorithm approach to detect spam emails using Machine Learning (ML). The methodology consists Nine following stages:

1. Data Collection: A dataset of labeled emails (spam and non-spam) is collected from publicly available sources, such as Kaggle or UCI Machine Learning Repository, AWS dataset, Google etc.

2. Data Preprocessing: The collected data is preprocessed by removing stop words, punctuation, and special characters. The emails are then tokenized, and the frequency of each word is calculated using TF-IDF.

3. Feature Selection: We picked the 1000 most important features to train AI models. Our system spots harmful messages, labeling them spam or not. We used a special algorithm to create a dictionary. Then, we tested ten powerful AI methods to find the best one to detect spam.

4. Model Selection: "Ten machine learning algorithms are chosen for assessment, such as Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Random Forest, AdaBoost Classifier, Extra Trees Classifier, Gradient Boosting Classifier, and XGBoost Classifier.

5. Model Training and Evaluation: Each ML model is trained on the preprocessed data and evaluated using accuracy and precision metrics. The models are trained using a test size =0.2 split (80% for training and 20% for testing).

6. Hyperparameter Turing: Hyperparameter tuning is performed for each model using techniques to optimize performance.

7. Model Comparison: The performance of all models is evaluated and compared using accuracy and precision metrics. The top performing model is selected for integration into a GUI based system. In our case the top model is MNB

8. GUI-Based System Development: A GUI-based system is developed using the top-performing model to classify emails is spam or non-spam.

9. Testing and Validation: The GUI-based system is tested and validated using a separate dataset to ensure its effectiveness in real-world scenarios.

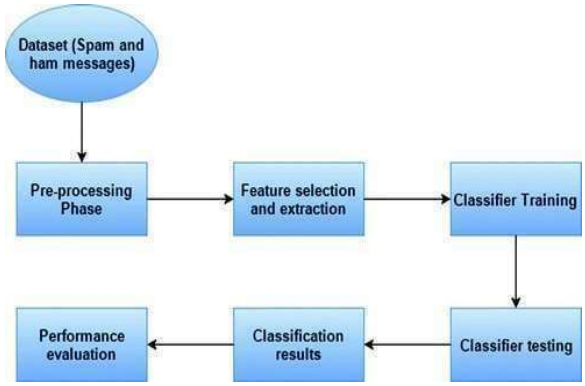


Fig 1 System Architecture

The data utilized in this research paper was coming from the University of California, Irvine (UCI) repository [17], which comprises a total data of 5,572 short messages, including 4,516 legitimate messages and 653 mobile spam E-mail. This study leverages a comprehensive dataset from the UCI repository, comprising 5,572 short messages, including 4,516 genuine messages and 653 mobile spam messages. Notably, this SMS spam corpus is the largest publicly available collection to date, providing a robust foundation for our research.

TABLE 1. A DISTRIBUTION OF DATASET

Message	Amount	Datatype
Ham	4,516	int64
Spam	653	int64
Total	5,572	int64

Our dataset contains around 4,900 legitimate E-mail and 653 spam mail, stored in a precise computer format. Figures 1 and 2 show the balance between ham and spam messages and their overall distribution. Figure 1 shown the ratio between HAM and SPAM dataset.

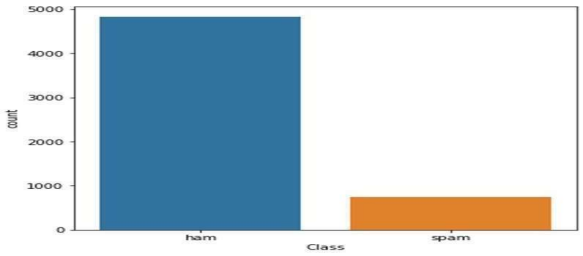


Fig. 2. SPAM and HAM data separation.

IV. MACHINE LEARNING APPROACHES

In this part, we tested ten ML classifiers - Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Random Forest, AdaBoost Classifier, Extra Trees Classifier, Gradient Boosting Classifier, and XG Boost Classifier - to see how well they work, measuring their accuracy and other performance metrics. Like their accuracy and precision

A. Logistic Regression

"Logistic regression is a statistical method applied to binary classification tasks, where the target variable has two potential outcomes (e.g., yes/no or 1/0). It is also capable of handling multi-class classification scenarios. This algorithm estimates the probability that a given instance belongs to a specific class by utilizing the logistic function. Although logistic regression is primarily employed for classification purposes, its underlying structure remains rooted in regression analysis. Specifically, it utilizes a linear combination of features to model the probability of an event's occurrence, subsequently applying a logistic transformation to constrain this probability within the interval [0, 1]. The logistic regression equation models the probability
(p) as a function of the input variables (X), represented as:

$$log(p/(1-p)) = \beta 0 + \beta 1X,$$

where β0 and β1 are coefficients.

B. K Nearest Neighbors

The K-Nearest Neighbors (KNN) methodology leverages proximity-based reasoning to classify and regress data, predicated on the notion that analogous data points cluster spatially within the feature landscape. This supervised machine learning approach infers labels by identifying the most proximal neighboring observations to novel inputs. Through training data, KNN cultivates a mapping function that associates inputs with outputs. To quantify affinity, the algorithm employs metric distancing, notably to the ED

The Euclidean distance formula calculates the spatial separation between two data points, x and y, across multiple dimensions. For feature vectors x = (x1, x2) and y = (y1, y2), the Euclidean distance, denoted as E(D), is given by the expression

$$E(D) = \sqrt{\left((x_1 - \varkappa_2)(+(y_1 - y_2)\right)\ )}$$

C. **Support Vector Machines (SVM)**

Support Vector Machines (SVMs) are adaptable supervised learning models that perform well in both classification and regression tasks. Using kernel techniques, SVMs effectively handle non-linear classification by transforming data into higher-dimensional feature spaces. In this framework, data instances are represented as multidimensional vectors, and SVMs aim to separate these instances using an optimal hyperplane. This hyperplane is carefully configured to maximize the margin between classes, ensuring a clear separation between the decision boundary and the nearest data points from either class.

For binary classification, the equation of the decision boundary (hyperplane) is represented as:

$$\mathbf{w \cdot x + b = 0}$$

Where:

- **w** denotes the weight vector, orthogonal to the hyperplane,
- **x** represents the feature vector,
- **b** is the bias term (or intercept).

$$y=sign(w \cdot x+b)$$

D. **Naive Bayes (NB)**

Naive Bayes Naive Bayes is a simple yet effective approach to building classifiers, which categorize samples by describing attributes as vectors and selecting from a predefined set of class labels. Despite its simplicity, Naive Bayes excels in complex real-world scenarios due to its adaptability, requiring only a linear number of parameters relative to the feature count. Its key advantage lies in requiring minimal training data to calculate classification factors. By leveraging Bayes' theorem, Naive Bayes decomposes probabilistic models into manageable components, enabling efficient classification.

$$p(A_c|\,X) = \frac{P(A_c)p(X\,|A_c\,)}{P(X)}$$

The formula calculates the probability of a class (target) given its attributes (X), using:

1. Initial class probability
2. Attribute likelihood for that class
3. Attribute probability

In Bayesian probability terms, this relationship can be expressed as:

posterior = (prior * likelihood) / evidence

E. **Decision Tree (DT)**

A Decision Tree is a supervised classifier that solves classification and regression problems, particularly excelling in classification tasks. It starts at the root node, comparing data features to determine the class, then branches to subsequent nodes based on the comparison. This process repeats until reaching the leaf node. The algorithm uses entropy (H) to guide data splitting, calculated as H = -P log2(p+) - P log2(p-), where P is probability, p+ is positive class percentage, and p- is negative class percentage. Decision Trees mimic human decision-making and are easy to understand due to their tree-like structure.

$$H(s) = -P+\ log2(p+) - P-\ log2(p-)$$

F. **Random Forest (RF)**

Random Forest is a powerful and flexible machine learning algorithm, well-known for its simplicity, diversity, and reliable performance in both regression and classification applications. Its key benefits include identifying crucial parameters, handling multiple problems simultaneously, and introducing variability to the model through randomized attribute selection. Unlike traditional decision trees, Random Forest evaluates a subset of features at each node split, which contributes to enhanced model performance. This approach effectively balances precision and generalizability, making Random Forest an attractive choice for complex problem.

P+ signifies the probability of being assigned to the positive class, while P- indicates the probability of being assigned to the negative class.

Random Forest is a versatile and robust machine learning algorithm, renowned for its simplicity, diversity, and consistent excellence in regression and classification tasks. Its key benefits include identifying crucial parameters, handling multiple problems simultaneously, and introducing variability to the model through randomized attribute selection. Unlike traditional decision trees, At each node split, Random Forest uses a subset of features, which results

$$=1 - \sum n(p)^2$$

$$= 1 - ((p+)^2 + ((p-)^2)$$

in improved model performance. This approach effectively balances precision and generalizability, making Random Forest an attractive choice for complex problem.

P+ and P- represent the probabilities of the positive and negative classes, respectively.

G. **AdaBoost (AB)**

Adaptive Boosting, or AdaBoost, is a sequential ensemble method that merges multiple weak classifiers to create a strong, accurate model. While it may be susceptible to noisy data, AdaBoost is easy to implement. It works by progressively adjusting weights and distributions to concentrate on difficult instances, thereby enhancing the model's performance.

At each step, the algorithm updates the distribution (DtD_tDt) and weight (αtα_tαt) based on the error rate, using Euler's number (eee) and a normalization factor (ZtZ_tZt). This process systematically enhances accuracy by utilizing both classifiers and relevant data, emphasizing difficult cases.

instances.

$D_t(i) \exp(-\alpha_t y_i h_t(X_i))$

H. **Gradient Boost:**

Gradient Boosting is a robust ensemble machine learning method that builds a strong prediction model by sequentially combining several weak models. It operates by training decision trees iteratively to predict the residuals or errors from the previous tree, progressively enhancing the overall prediction. Each new tree aims to correct the errors made by the prior one, using gradients to direct the learning. This process continues until a predefined number of trees is built or the desired accuracy is attained. Gradient Boosting excels in handling complex interactions, non-linear relationships, and diverse data types, making it a popular choice for classification, regression, and ranking tasks

$F(x) = \sum[\alpha t * ht(x)]$ from t=1 to T

Where:

$$\gamma = \frac{\text{Sum of residuals}}{\text{Sum of each } p(1-p) \text{ for each sample in the leaf}}$$

F(x) = final prediction
αt = learning rate
ht(x) = decision tree at iteration t
T = total number of trees

During each iteration, the model minimizes the loss function:

$L(y, F(x)) = (y - F(x))^2$

I. **Extreme Gradient Boost (XGBoost)**

Extreme Gradient Boost (XGBoost) is an optimized gradient boosting algorithm that excels in speed, efficiency, and performance. Developed by Tianqi Chen, XGBoost is not an algorithm it is a library which is just updated version of Gradient Boost.

Formula explanation:

Dt(i) = updated distribution

exp(−αtyiht(Xi)) = weight update

Zt = normalization factor

αt = weight for classifier

ht = hypothesis

(Xi, yi) = training sample

J. **Extra Trees Classifier**

The Extra Trees Classifier is an ensemble machine learning algorithm that creates multiple decision trees, with each tree built using random node splits. This approach minimizes correlation and maximizes diversity among trees, like to Random Forest. The classifier evaluates outcomes using entropy, ensuring accurate predictions. Each tree splits nodes randomly, reducing correlation and increasing diversity. The classifier evaluates outcomes using entropy:

*Entropy (H) = - ∑ (pi * log2(pi))*

Where:

H = entropy
pi = proportion of samples with class label i
c = number of unique class labels

By aggregating decorrelated tree predictions, Extra Trees Classifier achieves accurate and robust performance in classification tasks.

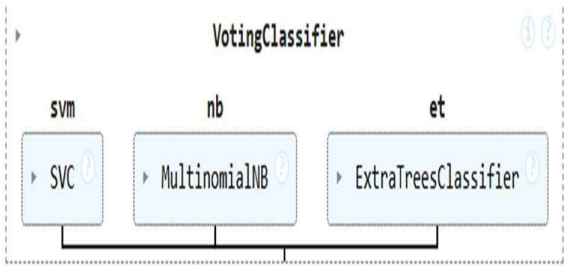
K. **ENSEMBLE METHODS**

*A. **Voting***

A Voting classifier is a powerful machine learning method that aggregates predictions from several models to identify the most probable output class based on majority voting. By training an ensemble of different models and pooling their results, the classifier selects the output class with the most votes. This method enhances overall model performance by reducing variance and bias, leading to higher accuracy. We will use this approach in our model to predict whether a message is spam or not.

1. **Soft Voting**:

In our model, we utilized soft voting to combine the predictions of multiple models. This approach works by averaging the probabilities assigned to each class by the individual models. For instance, when we inputted data into three models, the probabilities for class A were 0.30, 0.47, and 0.53, while class B had identical probabilities. Class C also had the same probabilities, and class D had slightly different probabilities at 0.30, 0.47, 0.20, 0.32, and 0.40. By calculating the average probability, we determined that class A had the highest average probability of 0.4333, making it the clear winner.



```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

In this we combine three algorithm SVM, NB, and ET after combine this it gives around 98 percent accuracy. That is pretty much higher than other algo. So basically, that is whole concert behind that Ensemble language and Voting classifier.

A. **Parameter Tuning**

To train the models, we split the dataset into two parts: training and testing. The parameters used for our models are listed in TABLE II. A total of ten algorithms were applied, and the details of the parameters are provided below.

TABLE II. PARAMETERS

Applied Algorithms	Parameters
LR	solver='liblinear',penalty='l1'
KNN	n_neighbors=7
SVC	kernel='sigmoid', gamma=1.0
MNB	default
DT	max_depth=5
RF	n_estimators=50, random_state=2
AB	n_estimators=50, random_state=2
GB	n_estimators=50, random_state=2
XGB	n_estimators=50, random_state=2
ET	n_estimators=50, random_state=2

B. **Experimented Results of Algorithms**

The evaluation results show that the Random Forest Classifier outperformed all other algorithms, achieving an impressive accuracy of 97.5898% and precision of 97.592%. SVM Boost and Extra Trees Classifier were close behind, both securing second place with accuracy and precision values of 97% and 97.05%, respectively. Five classifiers—Naive Bayes (NB), Random Forest (RF), AdaBoost (AB), Gradient Boosting (GB)—each reached an accuracy of 97%, indicating similar performance. The KNN classifier recorded the lowest accuracy, around 90%.

Most Common Word come in spam dataset

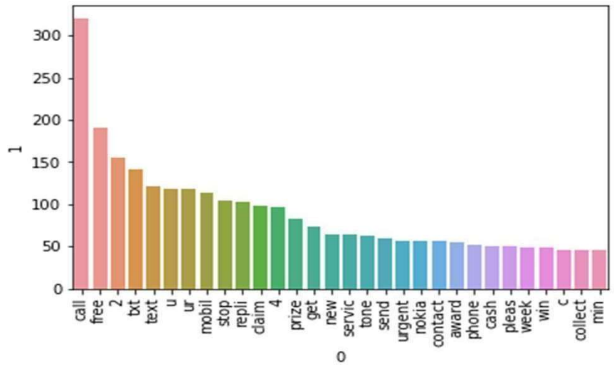
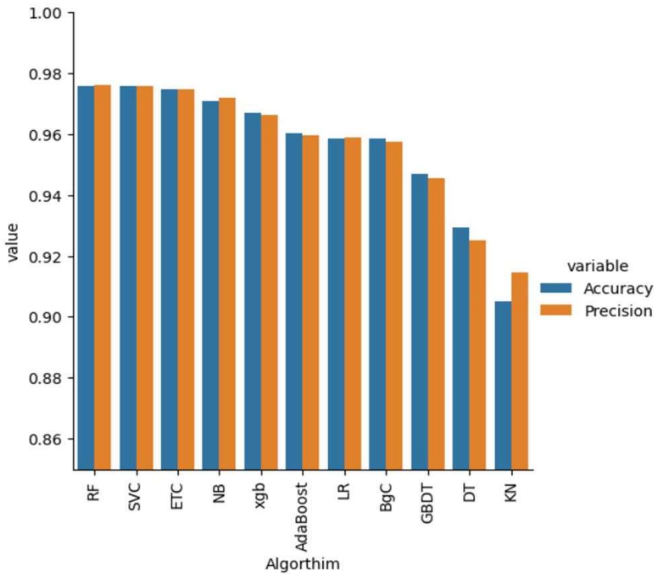
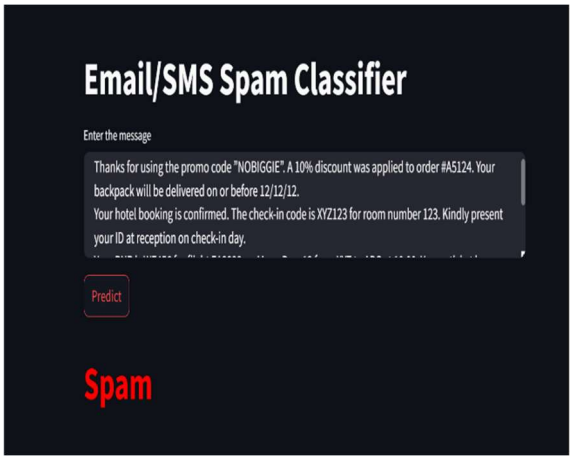


TABLE III. Accuracy and Precision

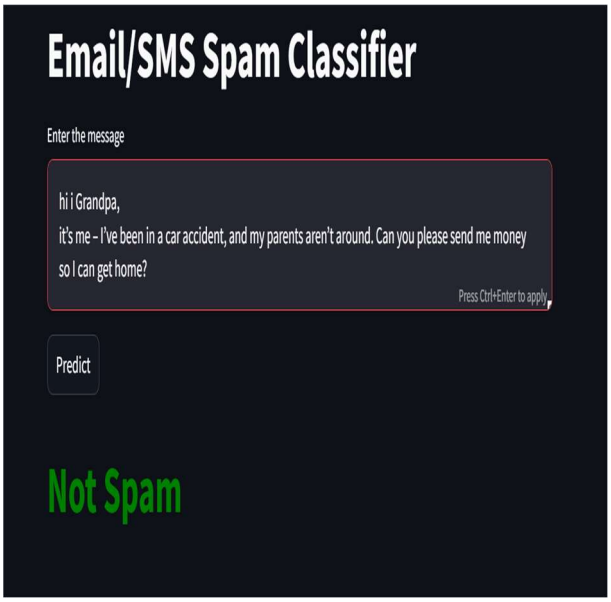
	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y
0	RF	0.975822	0.975984	0.975822	0.975984	0.975822	0.975984
1	SVC	0.975822	0.975801	0.975822	0.975801	0.975822	0.975801
2	ETC	0.974855	0.974849	0.974855	0.974849	0.974855	0.974849
3	NB	0.970986	0.971926	0.970986	0.971926	0.970986	0.971926
4	xgb	0.967118	0.966401	0.967118	0.966401	0.967118	0.966401
5	AdaBoost	0.960348	0.959503	0.960348	0.959503	0.960348	0.959503
6	LR	0.958414	0.958885	0.958414	0.958885	0.958414	0.958885
7	BgC	0.958414	0.957517	0.958414	0.957517	0.958414	0.957517
8	GBDT	0.946809	0.945657	0.946809	0.945657	0.946809	0.945657
9	DT	0.929400	0.925183	0.929400	0.925183	0.929400	0.925183
10	KN	0.905222	0.914567	0.905222	0.914567	0.905222	0.914567



now let just look our GUI-based system developed using the Naive Bayes model who accuracy is almost 97% for spam email detection. The system receives an email as input and produces a classification label that indicates whether the email is spam or not. As shown in Figure 1, the system correctly identifies a spam email with a confidence score of 0.97598 demonstrating its ability to accurately detect unwanted emails. In contrast, Figure 2 shows the system classifying a legitimate email as not spam, with a confidence score of 0.97586. These outcomes highlight the effectiveness of the Random Forest Classifier model in predicting email spam, delivering high accuracy and precision.



This intuitive GUI-based system empowers email users to effortlessly identify and flag spam emails, significantly mitigating the risks of phishing attacks and malware infections. With its exceptional accuracy and reliability, the system serves as a valuable asset for email service providers seeking to enhance their spam filtering capabilities. Notably, the successful integration of the Naive Bayes model in this project demonstrates its practical effectiveness in real-world email spam detection applications.



CONCLUSION

This study conclusively identifies Naive Bayes (NB) as the optimal Machine Learning algorithm for spam email detection. After a thorough evaluation of ten top algorithms, Naive Bayes (NB) stood out as the best performer, achieving an impressive accuracy and precision of 97.57%. Outperforming notable contenders like Support Vector Classifier and Ensemble methods, NB's superior performance makes it the ideal choice for effectively detecting spam emails.

The results of this study hold important implications for building more effective spam detection systems. By harnessing the advantages of Naive Bayes (NB), email service providers can enhance the precision of spam filtering, alleviating the impact of unwanted emails on both individuals and organizations. Additionally, the findings emphasize that accuracy alone may not be enough in spam detection—precision is crucial to minimize false positives.

The findings of the study also highlight the drawbacks of certain algorithms, like Logistic Regression (LR) and Decision Trees (DT), which showed relatively lower performance in terms of precision. These findings can inform future research directions, encouraging the exploration of alternative algorithms and techniques to improve spam detection.

REFERENCES

[1] A. Karim, S. Azam, B. Shanmugam, and K. Kannoorpatti, "An unsupervised approach for content-based clustering of emails into spam and Ham through multiangular feature formulation," IEEE Access, vol. 9, pp. 135186–135209, 2021.

[2] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," Decision Support Systems, vol. 107, pp. 88–102, 2018.

[3] W. Mao, Z. Cai, D. Towsley, Q. Feng, and X. Guan, "Security importance assessment for system objects and malware detection," Computers & Security, vol. 68, pp. 47–68, 2017.

[4] K. Zainal, N. Sulaiman, and M. Jali, ''An analysis of various algorithms for text spam classification and clustering using RapidMiner and Weka,'' Int. J. Comput. Sci. Inf. Secur., vol. 13, no. 3, pp. 66–74, 2015.

[5] E.-S. M. El-Alfy and A. A. AlHasan, ''Spam filtering framework for multimodal mobile communication based on dendritic cell algorithm,'' Future Generat. Comput. Syst., vol. 64, pp. 98–107, 2016.

[6] L. Chen, Z. Yan, W. Zhang, and R. Kantola, ''TruSMS: A trustworthy SMS spam control system based on trust management,'' Future Generat. Comput. Syst., vol. 49, pp. 77–93, Aug. 2015. [7] J. M. G. Hidalgo, T. A. Almeida, and A. Yamakami, ''On the validity of a new SMS spam collection,'' in Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA), Dec. 2012, pp. 240–245.

[7] C. F. M. Foozy, R. Ahmad, and M. F. Abdollah, ''A framework for SMS spam and phishing detection in malay language: A case study,'' Int. Rev. Comput. Softw., vol. 9, no. no. 7, pp. 1248–1254, 2014.

[8] SD Gupta, S Saha, SK Das, " SMS Spam Detection Using Machine Learning", Journal of Physics: Conference Series, Vol.1797, no. 1, p. 012017, 2021.

[9] Luo GuangJun, Shah Nazir, Habib Ullah Khan, Amin Ul Haq, "Spam Detection Approach for Secure Mobile Message Communication Using Machine Learning Algorithms", Security and Communication Networks, vol. 2020, Article ID 8873639, 6 pages, 2020. [11] H. Yang, Q. Liu, S. Zhou, and Y. Luo, "A spam filtering method based on multi-modal fusion," Applied Sciences, vol. 9, no. 6, p. 1152, 2019.

[10] O. Abayomi-Alli, S. Misra, A. Abayomi-Alli, and M. Odusami, "A review of soft techniques for SMS spam classification: methods, approaches and applications," Engineering Applications of Artificial Intelligence, vol. 86, pp. 197–212, 2019