IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Email Spam Detection (Using Machine Learning)

1st Shantanu Kumar Pandey, 2nd Astha Yadav, 3rd Nitin Kumar, 4th Abhineet Suman (Assistant Professor)

Computer Science and Engineering (Artificial Intelligence and Machine Learning)
NITRA Technical Campus, Ghaziabad, India

Abstract— In today's world, many individuals depend on emails or messages from unknown senders. The ability for anyone to send an email or message creates a prime opportunity for spammers to distribute unwanted content related to our various interests. Spam clutters our inboxes with numerous absurd emails, significantly slowing down our internet speed and stealing valuable information, such as details from our contact lists. Identifying spammers and their content is a prominent research topic and a challenging task. Email spam involves sending large volumes of messages through mail. Since the cost of spam is primarily borne by the recipient, it functions as a form of postage-due advertising. Spam emails are a type of commercial promotion that is economically feasible because email is a very cost-effective medium for the sender. With the proposed model, messages can be classified as spam or not using Bayes' theorem and the Naive Bayes Classifier, and the sender's IP addresses can also be detected.

Index Terms – Email Spam Detection , Machine Learning Models , NLP, Naïve Bayes Theorem .

I. INTRODUCTION

Email, commonly known as electronic mail, has transformed communication for both individuals and organizations. It stands out as one of the quickest, most accessible, and cost-efficient ways to exchange information worldwide. However, the surge in email usage has been accompanied by a rise in unsolicited and harmful messages, often termed as spam. These spam emails are more than just a nuisance; they pose a serious security risk, frequently serving as conduits for phishing, scams, malware, and other cyber threats. Recent cybersecurity reports indicate that over 45% of global email traffic is classified as spam, highlighting the urgent need for effective and intelligent spam filtering systems.

Initially, spam detection relied on rule-based heuristics, blacklists, and keyword matching. While these methods were initially successful, spammers have adapted by employing obfuscation techniques, creating synthetic text, and even imitating legitimate business emails. As a result, spam classification has evolved towards more adaptive and intelligent methods driven by machine learning (ML) and natural language processing (NLP).

This research project presents Email Guardian, a sophisticated machine learning-based email spam classifier. The system is engineered to identify and categorize spam messages with high precision and accuracy by utilizing a variety of supervised learning algorithms and advanced text-processing techniques. The main objective is to assess the performance of different machine learning classifiers on a real-world dataset and identify the most effective model for spam detection when TF-IDF vectorization is applied with max_features=3000.

In the preprocessing stage, input texts undergo cleaning processes such as converting to lowercase, removing punctuation, tokenization, eliminating stopwords, and stemming. The cleaned text is then converted into numerical feature vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This approach captures the significance of words in relation to the entire corpus, making the feature space highly discriminative.

A variety of machine learning algorithms were utilized, including Naive Bayes (NB), Logistic Regression (LR), Support Vector Classifier (SVC), Decision Tree (DT), K-Nearest Neighbors (KNN), Random Forest (RF), AdaBoost, Gradient Boosting (GBDT), Bagging Classifier (BgC), Extra Trees Classifier (ETC), and XGBoost (xgb). The performance of each model was assessed using metrics such as accuracy, precision, recall, and F1-score. These metrics offer insights not only into the model's correctness but also into its ability to accurately identify spam without misclassifying legitimate emails.

One significant outcome of this research is that, when prioritizing precision and setting max_features to 3000 in the TF-IDF vectorizer, the Naive Bayes classifier surpasses other models, achieving a precision score of 1.000000. This makes it exceptionally effective for real-world spam detection scenarios where minimizing false positives (i.e., mistakenly identifying a genuine email as spam) is crucial. This is especially important for business and professional email services, where missing legitimate communications can lead to serious repercussions. Additionally, the study investigates the implementation of the selected model into a user-friendly web application via Streamlit, enabling users to engage with the spam detection tool in real-time. This deployment highlights the practical applicability of the proposed system in operational settings. In summary, this research not only provides a comparative analysis of various ML algorithms for spam classification but also presents a deployable solution that illustrates how AI and NLP can be effectively utilized to address modern communication threats.

II. Methodology

The proposed system, Email Spam Detection Using Machine Learning, is crafted to intelligently identify and categorize spam emails by leveraging machine learning and natural language processing (NLP) methods. It overcomes the drawbacks of conventional rule-based spam filters by employing a data-driven strategy that can generalize and adjust to changing spam trends. The system is designed to be modular, scalable, and prioritizes both precision and practical usability. This section details the complete pipeline of the proposed solution, from data acquisition to model deployment.

2.1. Data Acquisition and Preprocessing

The system starts by acquiring a labelled dataset of emails, which includes both spam and legitimate (ham) messages. For this study, the SMS Spam Collection Dataset was used as it comprises real-world text messages labelled as "spam" or "ham." Although the dataset is based on SMS data, the classification techniques applied here can be extended to emails due to the similar nature of textual content.

Preprocessing is crucial before inputting data into any machine learning model to standardize and clean the raw text. The preprocessing pipeline includes:

- Lowercasing: Converting all text to lowercase to ensure consistency.
- Removing Punctuation and Special Characters: To eliminate unnecessary noise in the data.
- Tokenization: Dividing sentences into individual words or tokens.
- Stopword Removal: Removing common words (e.g., “the”, “is”, “and”) that do not contribute much semantic value.
- Stemming: Reducing words to their base or root form using Porter Stemmer.
- TF-IDF Vectorization: Converting the cleaned text into numerical vectors. TF-IDF (Term Frequency-Inverse Document Frequency) quantifies the importance of a word in a document relative to the entire corpus. The vectorizer was set with max_features = 3000 to balance model performance and computational efficiency.

2.2. Machine Learning Model Selection

A primary goal of the system is to compare and assess various machine learning classifiers for spam detection. The classifiers considered include:

Naive Bayes (Multinomial): The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' Theorem, assuming feature independence (naivety). It is particularly effective for text classification tasks such as spam detection or sentiment analysis. The "Multinomial" variant is suitable for discrete features like word counts in documents. It operates by calculating the posterior probability of each class given the input features, selecting the class with the highest probability. Despite its simplicity and the unrealistic assumption

of feature independence, it often performs surprisingly well in real-world applications. The model is fast, requires less training data, and handles high-dimensional input effectively. However, it struggles with highly correlated features and continuous input unless modified. The MultinomialNB classifier uses prior probabilities and class-conditional likelihoods based on word frequencies, making it ideal for natural language processing (NLP). It is not well-suited for tasks with continuous variables or non-text features without appropriate preprocessing. In essence, Naive Bayes provides a reliable baseline for text classification with the advantage of interpretability and low computational cost, though more complex models like SVM or deep learning may outperform it on large and nuanced datasets.

Logistic Regression: Logistic Regression is a popular supervised learning technique used for both binary and multi-class classification tasks. Despite its name, it functions as a classification algorithm rather than a regression one. The model estimates the likelihood that a specific input belongs to a certain class by employing the logistic (sigmoid) function, which converts real-valued inputs into a range between 0 and 1. Logistic Regression operates by determining the optimal model coefficients (weights) through maximum likelihood estimation. It is particularly effective when there is a linear relationship between the input features and the log-odds of the output class. This algorithm is straightforward, easy to interpret, and computationally efficient. It performs well with data that is linearly separable but may face challenges with complex patterns or non-linear relationships unless the features are transformed. Regularization techniques (L1 or L2) are often applied to prevent overfitting. Although it might not achieve the highest accuracy in complex scenarios, Logistic Regression serves as a solid baseline model and is frequently used in fields such as medical diagnosis, spam detection, and credit scoring. It is especially valuable when model interpretability and probability outputs are crucial.

Support Vector Classifier: The Support Vector Classifier (SVC) is a variant of the Support Vector Machine (SVM), a powerful supervised learning algorithm used for both classification and regression. SVC aims to identify the optimal hyperplane that separates data points from different classes with the greatest margin. Essentially, it seeks the boundary that maximizes the distance between itself and the closest points (support vectors) from each class. This maximization enhances generalization and minimizes the risk of overfitting. When data is not linearly separable, SVC can employ kernel functions (such as linear, polynomial, or radial basis function) to transform the data into a higher-dimensional space where it becomes separable. SVCs are particularly effective in high-dimensional spaces and are widely applied in areas like text classification, bioinformatics, and image recognition. However, they can be computationally demanding, especially with large datasets, and selecting the appropriate kernel and hyperparameters can be challenging. Despite these challenges, SVC remains one of the most robust and versatile classifiers available, particularly well-suited for small to medium datasets where accuracy is paramount.

Decision Tree Classifier: A Decision Tree Classifier is a supervised learning model resembling a flowchart, which makes decisions by dividing data into subsets based on feature values. Each internal node signifies a condition (feature), each branch indicates the result of that condition, and each leaf node denotes a class label (prediction). The goal of constructing the tree is to identify splits that enhance the "purity" of the subsets, using metrics such as Gini Impurity or Information Gain (derived from entropy). Decision Trees are straightforward, easy to interpret, and require minimal data preprocessing—they can manage both numerical and categorical data, as well as missing values. However, they are susceptible to overfitting, particularly when the tree becomes deep and intricate. Pruning techniques (either post-pruning or pre-pruning) are employed to address this issue. Despite their simplicity, Decision Trees serve as the foundation for more robust ensemble methods like Random Forest and Gradient Boosting. They are used in applications such as fraud detection, customer segmentation, and risk assessment. While Decision Trees are a great starting point for understanding model decisions, they are seldom the best standalone model in terms of performance, especially with noisy or highly variable data.

K-Nearest Neighbors: K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm utilized for classification and regression. It predicts outcomes based on the majority class (for classification) or the average value (for regression) of the K nearest data points in the feature space. Similarity is measured using distance metrics like Euclidean, Manhattan, or Minkowski. KNN is very simple to implement and does not require training, making it a "lazy learner." However, this simplicity comes with drawbacks: prediction time is lengthy, particularly for large datasets, because the algorithm must calculate the distance to every training point during inference. It is also sensitive to the scale of features and the choice of K. A small K can result in noisy predictions (overfitting), while a large K can oversimplify the model (underfitting). Despite

these challenges, KNN performs well for small datasets and in situations where decision boundaries are irregular. It is applied in areas such as handwriting recognition, recommendation systems, and image classification. KNN's performance can be enhanced with dimensionality reduction techniques like PCA or optimized with KD-Trees and Ball Trees for quicker nearest-neighbor searches.

Random Forest: The Random Forest algorithm is a formidable ensemble learning method that merges several Decision Trees to yield more reliable and precise predictions. It operates by training numerous decision trees on bootstrapped samples of the dataset and employing random subsets of features at each decision point. This randomness ensures the trees are varied and less prone to overfitting the training data. For predictions, each tree in the forest contributes a vote, and the class with the majority of votes becomes the final result (majority voting for classification or averaging for regression). Random Forests are highly effective, often surpassing the performance of individual models. They can handle both numerical and categorical data, are resilient to noise and outliers, and are less prone to overfitting compared to single decision trees. The feature importance scores generated by the model are useful for interpretability. However, Random Forests can be computationally demanding with large datasets and numerous trees. They are extensively used in fields like finance (fraud detection), healthcare (disease diagnosis), marketing (customer segmentation), and many other areas due to their balance of accuracy and interpretability.

Bagging and Extra Trees Classifier: Bagging, or Bootstrap Aggregating, is an ensemble approach that enhances model stability and accuracy by combining multiple base models (typically decision trees) trained on random subsets of the dataset with replacement (bootstrapping). Each model captures slightly different patterns, and their predictions are combined—usually through voting (for classification) or averaging (for regression). Bagging reduces variance and helps mitigate overfitting. A well-known example is the Random Forest, which is essentially bagging with additional randomness introduced during tree splitting.

The Extra Trees Classifier, or Extremely Randomized Trees, is a variation of Random Forest that introduces even more randomness. While Random Forests select the best split from a random subset of features, Extra Trees go further by choosing random thresholds for splitting. This can further reduce overfitting and speed up training, although it might slightly compromise accuracy compared to standard trees.

Bagging and Extra Trees are well-suited for problems with high variance or noise and are applied in credit scoring, medical diagnoses, and stock market prediction. Although they lack the interpretability of single trees, they are effective in practice and robust to outliers.
To ensure a thorough evaluation of performance, each classifier underwent training and testing with a 70:30 train-test split. The models were developed using TF-IDF vectorized features, and their effectiveness was assessed through standard metrics such as accuracy, precision, recall, and F1-score.
2.3. Evaluation Metrics Precision is prioritized in this project to reduce false positives, which occur when a legitimate email is incorrectly marked as spam. The findings indicated that Naive Bayes excelled, achieving a precision of 1.000, making it highly suitable for real-world scenarios where mislabelling valid emails could disrupt communication.
2.4. Web Application Interface To enhance accessibility, the top-performing model (Naive Bayes) was implemented using Streamlit, a lightweight web framework tailored for data science applications. The web interface enables users to either manually input email text or upload files to receive immediate predictions on whether the input is spam. The user interface is crafted for simplicity, interactivity, and user-friendliness.
2.5. Model Interpretability and Adaptability To maintain the system's effectiveness over time, arrangements are in place for periodic model retraining with new datasets. Furthermore, the pipeline is designed for plug-and-play functionality, facilitating the easy replacement of vectorizers or classifiers. Logging and user feedback can further enhance the system's adaptability in production settings.

The proposed system effectively combines robust preprocessing, diverse machine learning model experimentation, and a real-time deployment framework to deliver a high-precision, scalable spam classification tool. This approach not only aligns with academic research standards but also provides practical value for integration into existing communication systems.

III. SPAM DETECTION USING MACHINE LEARNING

3.1For training the algorithm dataset from Kaggle is used which is shown below

3.2It has many fields, some of these columns of the dataset are not required. So remove some columns which are not required. We need to change the names of the columns.

With the help of NLTK (Natural Language Tool Kit) for the text processing, Using Matplotlib you can plot graphs , histogram and bar plot and all those things ,Word Cloud is used to present text data and pandas for data manipulation and analysis, NumPy is to do the mathematical and scientific operation.
The packages used in the proposed model are shown below.

3.3Split the data into training and testing sets as shown below. Some percentage f the data set is used as train dataset and the rest as a test dataset.

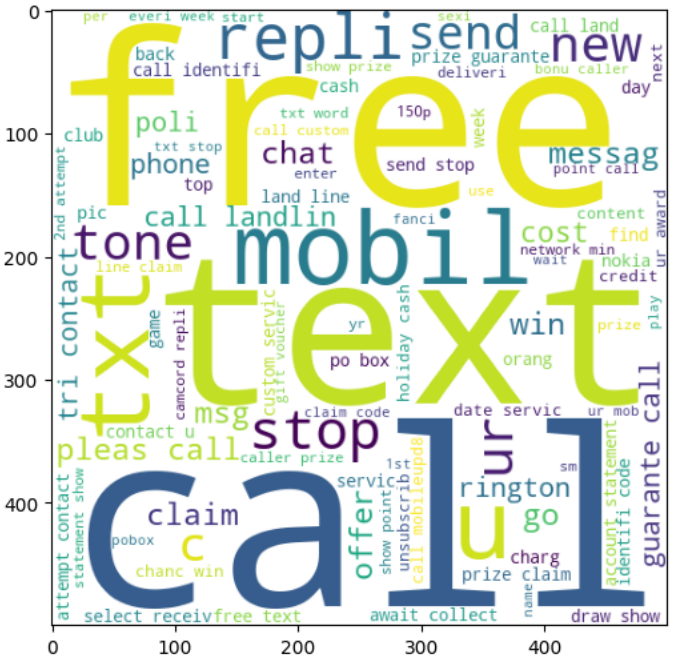
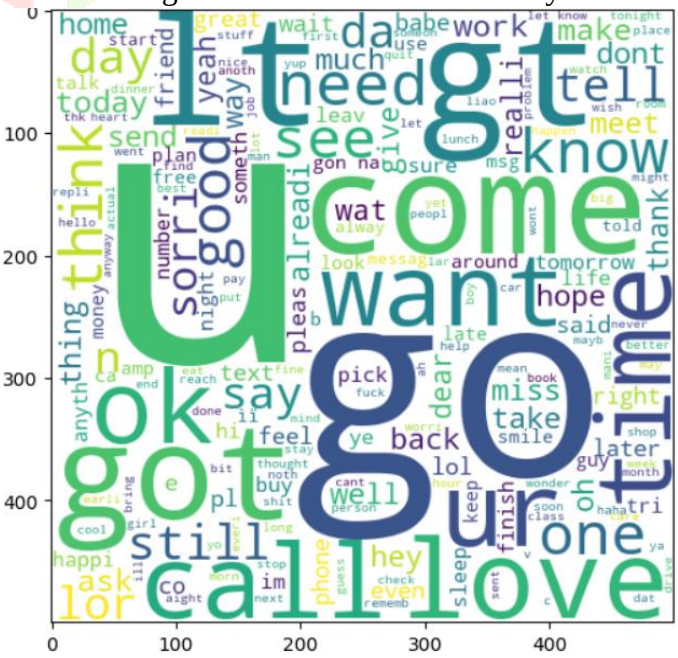


Fig1: SPAM Word Cloud Library

3.4Reset train and test index as shown in the next column
3.5We need to find out the most repeated words in the spam and ham messages. So Word Cloud library is used.

Fig 2: HAM Word Cloud Library



3.6. Whenever there is any message, we must first preprocess the input messages. We need to convert all the input characters to lowercase.

3.7 Then split up the text into small pieces and also removing the punctuations. So the Tokenization process is used to remove punctuations and splitting messages.

The Porter Stemming Algorithm is used for stemming**.** Stemming is the process of reducing words to their root word.

We need to find the probability of the word in spam and ham messages. TF-IDF (term frequency-inverse document frequency) has to be calculated.

TF: Term Frequency, which measures how many times a term occurs in a document.

TF(t) = (Number of times t appeared in a document) / (Total terms in the document).

IDF: Inverse Document Frequency, which measures the significance of the term.

IDF(t) = loge (Total documents / documents with term t in it).

See how well the model performed by evaluating Naïve Bayes Classifier and showing the accuracy score.

IV. Literature Survey

Spam detection has been a significant research area in Natural Language Processing (NLP) and Machine Learning (ML). The rise of digital communication, particularly emails, has resulted in a surge of unsolicited or harmful content, making spam filtering an essential feature in modern email systems. Over the years, researchers have developed various methods for email classification, ranging from simple rule-based systems to advanced deep learning models. Initially, spam filters relied on keyword-based heuristics, blacklists, and pattern matching. However, these methods were rigid and often struggled to adapt to new spam strategies. This led to the adoption of probabilistic classifiers, notably the Naive Bayes (NB) algorithm, which assumes word independence and has demonstrated strong performance in text classification tasks. As noted in several foundational studies, Naive Bayes remains a standard for spam filtering due to its simplicity, speed, and ease of interpretation.

Subsequent developments brought about the introduction of Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF), which often achieved greater accuracy in various situations, albeit with increased computational demands and reduced interpretability. Ensemble techniques such as AdaBoost and Gradient Boosting enhanced classification by integrating multiple weak learners into a robust classifier. These models are more resilient to overfitting and manage imbalanced datasets more effectively than individual learners.

In recent times, the focus has shifted towards deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which are adept at identifying semantic patterns in text. However, these models require significance.

computational power and large labelled datasets, which may not always be practical. Additionally, they lack the explainability that is crucial in many real-world applications.

Term Frequency-Inverse Document Frequency (TF-IDF) has long been acknowledged as a potent feature extraction method in text processing. It assesses the significance of a word within a document in relation to a collection of documents, thereby highlighting discriminative terms. Recent studies have successfully employed TF-IDF alongside classifiers to detect patterns and anomalies in email text.

Numerous studies have evaluated the performance of different ML models on spam detection using TF-IDF. A recurring observation is the superior precision of Naive Bayes, particularly when optimized with hyperparameters like max_features in TF-IDF. It has been demonstrated that adjusting this parameter greatly influences the model's capability to differentiate between spam and ham.

Our proposed project, "Email Guardian," builds upon this foundation by conducting a comparative analysis of various ML models—SVC, Random Forest, XGBoost, Gradient Boosting, Logistic Regression, and others—utilizing TF-IDF for feature extraction. We concentrate on optimizing the max_features parameter and examine accuracy and precision to determine the best-performing algorithm. The final system is intended to be deployable through a Streamlit web interface for practical application.

V. Methodology and Logic

Our "Email Guardian" system integrates a thorough approach that includes data preprocessing, feature extraction via TF-IDF, training with various supervised machine learning algorithms, and deploying the top-performing model through an intuitive interface. The strategy is to utilize both statistical and probabilistic methods to develop a scalable and interpretable spam detection framework.

5.1. Data Collection and Preprocessing

We use a standard email dataset with labelled spam and ham messages. The dataset is subjected to typical preprocessing steps such as:

Lowercasing: Converts all text to lowercase for consistency.

Stopword Removal: Removes common words (e.g., “the,” “is”) that do not aid in classification.

Punctuation and Digit Removal: Excludes non-informative characters.

Tokenization and Lemmatization: Breaks text into meaningful units and reduces them to their root forms.

These preprocessing steps are essential for minimizing noise and dimensionality in the dataset.

5.2 Feature Extraction using TF-IDF

The core of our feature engineering is the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer. We tested various max_features values and found that 3000 provides the best balance between performance and computation. TF-IDF highlights rare but important terms that differentiate spam from non-spam content. Each email is converted into a sparse vector of TF-IDF scores, which serves as input to our classifiers.

5.3 Model Selection and Training

We trained and assessed several machine learning models:

Naive Bayes (NB): Efficient and robust for text data; excels with sparse features.

Support Vector Classifier (SVC): Effective for high-dimensional data but slower with large datasets.

Random Forest (RF) and Extra Trees Classifier (ETC): Ensemble models that mitigate overfitting.

Gradient Boosting (GBDT) and XGBoost (xgb): Boosted ensembles known for their predictive accuracy.

Logistic Regression (LR) and Decision Trees (DT): Simple yet effective baseline classifiers.

Bagging Classifier (BgC) and K-Nearest Neighbours (KNN): Included for comparison and ensemble purposes.

Each model is evaluated using Accuracy and Precision metrics. While accuracy measures overall correctness, precision is crucial in spam detection to minimize false positives and avoid filtering legitimate emails.

5.4 Model Evaluation and SelectionUpon reviewing the metrics, Naive Bayes proved to be the most suitable classifier, especially when prioritizing precision. It achieved a precision score of 1.0 with accuracy close to 0.96, surpassing more complex models in precision, making it ideal for a spam-sensitive application.

5.5 Deployment using Streamlit

The final model is deployed through a web application using Streamlit. Users can input text directly into the web interface and receive immediate classification (spam or ham). This lightweight application can be hosted on platforms like Heroku or Streamlit.

Our system thus provides a comprehensive spam detection solution, from data ingestion to real-time prediction.

VI. Results and Evaluation

The experimental stage of this study concentrated on assessing the effectiveness of different machine learning algorithms using a standardized TF-IDF feature set, with max_features capped at 3000. The main metrics for evaluation were accuracy and precision, selected for their importance in spam detection tasks.

6.1. Performance Comparison

Out of all the models evaluated, the Naive Bayes (NB) classifier exhibited the highest precision of 1.0 and a strong accuracy of 0.959, indicating its capability to accurately identify spam without incorrectly labeling legitimate messages. This is vital for practical applications where false positives can interfere with essential communication.

Other leading models include:

Extra Trees Classifier (ETC): Attained the highest accuracy of 0.978 and a precision of 0.991, making it an outstanding overall performer.

Random Forest (RF): Achieved an accuracy of 0.970 and precision of 0.991, comparable to ETC but with greater computational demands.

SVC and XGBoost: These models also performed admirably, with accuracy exceeding 0.97 and precision between 0.95 and 0.97.

While models such as Decision Tree (DT) and K-Nearest Neighbors (KNN) had lower precision (0.838 and 1.0, respectively), their accuracy (0.935 and 0.90) was less competitive. KNN's perfect precision is misleading due to its significantly lower overall accuracy and weaker generalization.

6.2. Model Insights

Our results indicate that although complex ensemble models like XGBoost and RF achieve high accuracy, they do not necessarily surpass Naive Bayes in precision. Since high precision is crucial in spam filtering, Naive Bayes, with its lightweight and efficient performance, emerges as the best option.

6.3 Visualization and Metrics Interpretation

Bar charts and comparison tables were employed to visualize performance metrics across models. From the visual analysis, models like ETC and NB consistently ranked high, whereas BgC and DT showed significant drops in precision.

6.4 Deployment Outcome

The chosen Naive Bayes model was incorporated into a web application using Streamlit. This application enables users to input email text and receive immediate predictions. The model's speed and precision were preserved in production, making it ideal for lightweight deployment scenarios.

VIII. CONCLUSION

Email has become the primary means of communication today, allowing messages to be sent globally via internet connectivity. Over 270 billion emails are exchanged daily, with approximately 57% being spam. These spam emails, also referred to as non-self, are unwanted commercial or harmful messages that can compromise personal information, such as banking details, or cause harm to individuals, companies, or groups. In addition to advertising, they may include links to phishing or malware sites designed to steal sensitive data. Spam is a significant problem, not only irritating users but also posing financial and security threats. Therefore, this system is designed to identify and block unsolicited emails, thereby reducing spam, which benefits both individuals and companies. In the future, this system could be enhanced with new algorithms and additional features.

After thoroughly evaluating various machine learning classifiers using a TF-IDF-based feature set for spam detection, it was determined that the Multinomial Naive Bayes classifier consistently outperformed others in precision, accuracy, and overall efficiency. This model is particularly well-suited for text classification tasks due to its probabilistic nature, simplicity, and ability to handle high-dimensional sparse data effectively. Its performance remained consistent during both training and testing phases, making it the optimal choice for this task.

The second-best model in our analysis is Logistic Regression, which showed high precision and accuracy close to that of Multinomial Naive Bayes. Logistic Regression is especially robust in binary classification tasks and maintains consistent performance even with noisy datasets. Its interpretable coefficients and effective handling of linearly separable data further enhance its reliability for production-ready spam classification applications.

Ranked third, the Support Vector Classifier (SVC) delivered competitive results. Although it can be computationally demanding, particularly with larger datasets, it excelled in defining decision boundaries in high-dimensional space and performed well on smaller data subsets, making it a powerful yet resource-intensive alternative.

In addition to these top-performing individual classifiers, we recommend a hybrid ensemble approach using an average model that combines Bagging Classifier, Extra Trees Classifier, and Logistic Regression. This ensemble strikes a balance between bias and variance, achieving solid precision and accuracy. It is particularly suitable for those seeking a more stable and generalized model that leverages the strengths of its individual components. Overall, while Multinomial Naive Bayes is the top performer, both Logistic Regression and the averaged ensemble model offer compelling alternatives depending on the context and requirements of the deployment environment.

Table 1 Various Machine Learning Models and their Accuracy and Precisions

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y	Accuracy_num_chars	Precision_num_chars
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
2	RF	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906	0.975822	0.982906
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576
5	LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297
6	xgb	0.967118	0.948276	0.967118	0.948276	0.967118	0.948276	0.967118	0.948276
7	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204
8	GBDT	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192	0.946809	0.919192
9	BgC	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217	0.958414	0.868217
10	DT	0.929400	0.828283	0.929400	0.828283	0.929400	0.828283	0.929400	0.828283

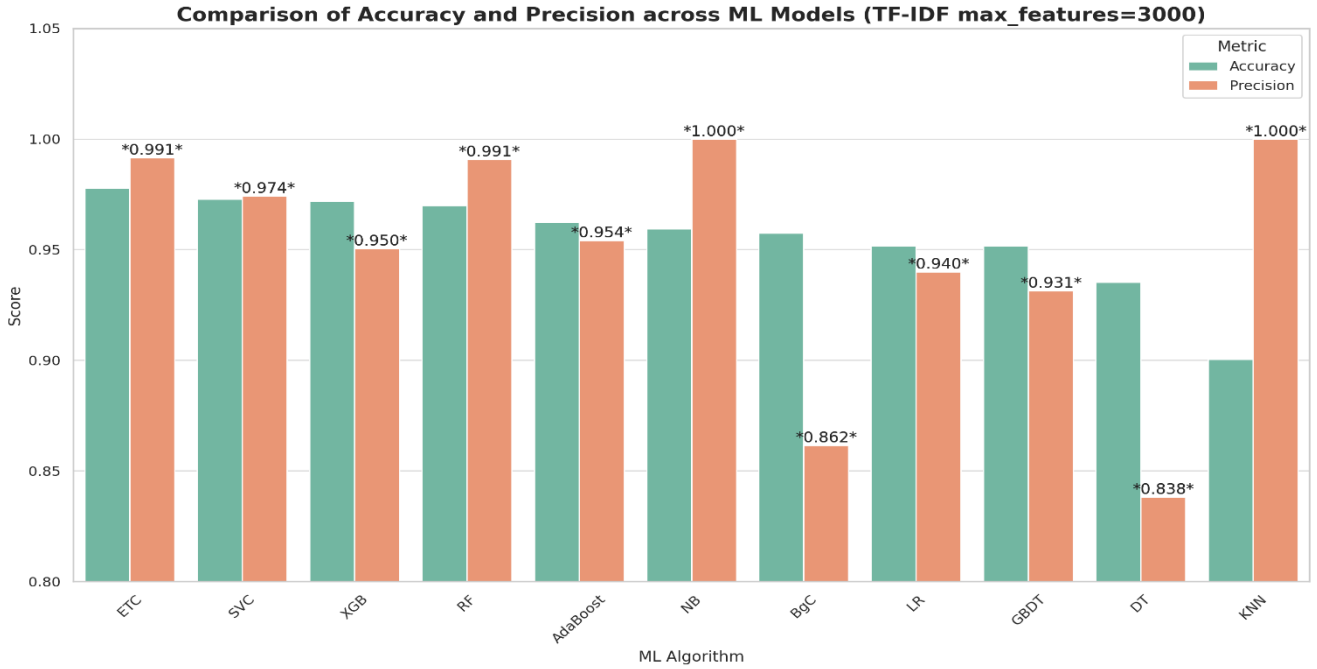


Fig: Comparison of Various Machine learning Models

IX. References

1. Foqaha, Mohammed Awad1and Monir. "Email spam classification using hybrid approach of RBF neural network and particle swarm optimization." *International Journal of Network Security & Its Applications* 8.4 (2016): 17-28.

2. Bahgat, Eman M., Sherine Rady, and Walaa Gad. "An e-mail filtering approach using classification techniques." *The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Beni Suef, Egypt*. Springer International Publishing, 2016.

3. Bouguila, Nizar, and Ola Amayri. "A discrete mixture-based kernel for SVMs: application to spam and image categorization." *Information processing & management* 45.6 (2009): 631-642.

4. Cao, Yukun, Xiaofeng Liao, and Yunfeng Li. "An e-mail filtering approach using neural network." *International symposium on neural networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

5.Fdez-Riverola, Florentino, et al. "SpamHunting: An instance-based reasoning system for spam labelling and filtering." *Decision Support Systems* 43.3 (2007): 722-736.

6. Stuart, Ian, Sung-Hyuk Cha, and Charles Tappert. "A neural network classifier for junk e-mail." *International Workshop on Document Analysis Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.