**Email Detection : Using Machine Learning**

# Nitin Kumar , Astha Yadav , Shantanu Kumar Pandey, Abhineet Suman

Department of Computer Science and Engineering

NITRA Technical Campus, Ghaziabad (Uttar Pradesh)

1. ***Abstract***— Nowadays, a big part of people rely on available email or messages sent by the stranger. The possibility that anybody can leave an email or a message provides a golden opportunity for spammers to write spam message about our different interests .Spam fills inbox with number of ridiculous emails Degrades our internet speed to a great extent. Steals useful information like our details on our contact list. Identifying these spammers and also the spam content can be a hot topic of research and laborious tasks. Email spam is an operation to send messages in bulk by mail .Since the expense of the spam is borne mostly by the recipient ,it is effectively postage due advertising. Spam email is a kind of commercial advertising which is economically viable because email could be a very cost effective medium for sender .With this proposed model the specified message can be stated as spam or not using Bayes’ theorem and Naive Bayes’ Classifier and Also IP addresses of the sender are often detected

*Keywords— Term Frequency, Inverse Document Frequency, language tool kit.*

**Published in:**[**2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)**](https://ieeexplore.ieee.org/xpl/conhome/9169741/proceeding)

**II.INTRODUCTION**

Electronic mail, or email, has revolutionized the way individuals and organizations communicate. It is one of the fastest, most accessible, and most cost-effective methods for sharing information globally. However, with the exponential rise in email usage comes the parallel growth of unsolicited and malicious messages, commonly referred to as spam. Spam emails are not just a minor inconvenience; they represent a significant security concern, often acting as vectors for phishing, scams, malware, and other cyber threats. According to recent cybersecurity reports, over 45% of global email traffic is categorized as spam, which emphasizes the need for reliable and intelligent spam filtering mechanisms.

Traditional spam detection systems were based on rule-based heuristics, blacklists, and keyword matching. Although these methods were effective in early stages, spammers have evolved their tactics by using obfuscation techniques, generating synthetic text, and even mimicking legitimate business emails. Consequently, the task of spam classification has shifted towards more adaptive and intelligent approaches powered by machine learning (ML) and natural language processing (NLP).

This research project introduces EmailGuardian, a comprehensive machine learning-based email spam classifier. The system is designed to detect and classify spam messages with high precision and accuracy by leveraging a range of supervised learning algorithms and modern text-processing techniques. The primary goal is to evaluate the performance of different machine learning classifiers on a real-world dataset and determine the most effective model for spam detection when TF-IDF vectorization is applied with max\_features=3000.

In the preprocessing pipeline, input texts undergo cleaning steps such as lowercasing, removal of punctuation, tokenization, stopword elimination, and stemming. The cleaned text is then transformed into numerical feature vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. This helps in capturing the importance of words relative to the entire corpus, making the feature space highly discriminative.

Multiple machine learning algorithms were employed, 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). Each model's performance was evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights not just into how often the model is correct, but also how effectively it identifies spam without misclassifying legitimate messages.

One of the key findings of this study is that when sorting by precision, and with max\_features=3000 set in the TF-IDF vectorizer, the Naive Bayes classifier outperforms other models, achieving a precision of 1.000000, making it highly suitable for real-world spam detection tasks where false positives (i.e., classifying a real email as spam) must be minimized. This is particularly valuable for business and professional email services, where loss of legitimate communication can have serious consequences.

Furthermore, this study explores the deployment of the chosen model into a user-friendly web application using Streamlit, allowing end-users to interact with the spam detection tool in real-time. The deployment showcases the practical viability of the proposed system in production environments.

In conclusion, this research not only offers a comparative analysis of diverse ML algorithms for spam classification but also delivers a deployable solution that demonstrates how AI and NLP can be leveraged to combat modern communication threats effectively.

# **III.** **Methodology**

The proposed system, Email Spam Detection Using Machine Learning is designed to intelligently detect and classify spam emails using machine learning and natural language processing (NLP) techniques. It addresses the limitations of traditional rule-based spam filters by adopting a data-driven approach that can generalize and adapt to evolving spam patterns. The system is modular, scalable, and focuses on both accuracy and real-world usability. This section outlines the full pipeline of the proposed solution, from data ingestion to model deployment.

3.1. Data Acquisition and Preprocessing

The system begins by ingesting a labeled dataset of emails consisting of both spam and ham (legitimate) messages. For the purpose of this research, the SMS Spam Collection Dataset was utilized as it contains real-world text messages labeled as "spam" or "ham." Although the dataset is based on SMS data, the classification techniques used here can be generalized to emails due to the similar nature of textual content.

Before feeding data into any machine learning model, preprocessing is essential to standardize and clean the raw text. The preprocessing pipeline involves:

Lowercasing: Converting all text to lowercase to ensure uniformity.

Removing Punctuation and Special Characters: To avoid unnecessary noise in the data.

Tokenization: Splitting sentences into individual words or tokens.

Stopword Removal: Eliminating common words (e.g., “the”, “is”, “and”) that do not add much semantic value.

Stemming: Reducing words to their base or root form using Porter Stemmer.

TF-IDF Vectorization: Transforming the cleaned text into numerical vectors. TF-IDF (Term Frequency-Inverse Document Frequency) helps quantify the importance of a word in a document relative to the entire corpus. The vectorizer was configured with max\_features=3000 to strike a balance between model performance and computational efficiency.

3.2. Machine Learning Model Selection

A core objective of the system is to compare and evaluate multiple machine learning classifiers for the task of 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 especially effective for text classification tasks such as spam detection or sentiment analysis. The "Multinomial" variant is suited for discrete features like word counts in documents. It works 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 that features are independent, 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 great for tasks with continuous variables or non-text features without appropriate preprocessing. In essence, Naive Bayes offers a reliable baseline for text classification with the benefit of interpretability and low computation cost, though more complex models like SVM or deep learning may outperform it on large and nuanced datasets.

Logistic Regression: Logistic Regression is a widely used supervised learning algorithm for binary and multi-class classification problems. Despite its name, it is a classification algorithm, not a regression one. The model predicts the probability that a given input belongs to a particular class using the logistic (sigmoid) function, which maps real-valued inputs to a range between 0 and 1. Logistic Regression works by estimating the best-fitting model coefficients (weights) using maximum likelihood estimation. It is especially effective when the relationship between the input features and the log-odds of the output class is linear. Logistic Regression is simple, interpretable, and computationally efficient. It performs well with linearly separable data but may struggle with complex patterns or non-linear relationships unless features are transformed. Regularization (L1 or L2) is often added to prevent overfitting. Though it may not achieve the highest accuracy in complex problems, Logistic Regression provides a strong baseline model and is commonly used in applications like medical diagnosis, spam detection, and credit scoring. It's particularly valuable when model interpretability and probability outputs are important.

Support Vector Classifier: Support Vector Classifier (SVC) is a type of Support Vector Machine (SVM), a powerful supervised learning algorithm used for classification and regression. SVC seeks to find the optimal hyperplane that separates data points from different classes with the maximum margin. In other words, it looks for the boundary that maximizes the distance between itself and the nearest points (support vectors) from each class. This maximization improves generalization and reduces the risk of overfitting. When data is not linearly separable, SVC can use kernel functions (e.g., linear, polynomial, 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 used in applications like text classification, bioinformatics, and image recognition. However, they can be computationally intensive, especially with large datasets, and tuning the right kernel and hyperparameters can be challenging. Despite that, SVC remains one of the most robust and versatile classifiers available, particularly suited for small to medium datasets where accuracy is critical.

Decision Tree Classifier: A Decision Tree Classifier is a flowchart-like supervised learning model that makes decisions by splitting data into subsets based on feature values. Each internal node represents a condition (feature), each branch represents the outcome of the condition, and each leaf node represents a class label (prediction). The tree construction aims to find splits that maximize the "purity" of the subsets using metrics like Gini Impurity or Information Gain (from entropy). Decision Trees are intuitive, easy to interpret, and require minimal data preprocessing—they can handle both numerical and categorical data, and missing values. However, they are prone to overfitting, especially when the tree becomes deep and complex. Pruning techniques (post-pruning or pre-pruning) are used to combat this. Despite their simplicity, Decision Trees form the building blocks of more powerful ensemble methods like Random Forest and Gradient Boosting. Use cases include fraud detection, customer segmentation, and risk assessment. Decision Trees are an excellent starting point for understanding model decisions but are rarely the best standalone model in terms of performance, especially on noisy or highly variable data.

K-Nearest Neighbours: K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for classification and regression. It makes predictions based on the majority class (for classification) or the average value (for regression) of the K closest data points in the feature space. Distance metrics such as Euclidean, Manhattan, or Minkowski are used to measure similarity. KNN is extremely simple to implement and doesn’t require training—making it a “lazy learner.” However, its simplicity comes with trade-offs: prediction time is high, especially for large datasets, because the algorithm must compute the distance to every training point at inference time. It is also sensitive to the scale of features and the choice of K. A small K can lead to noisy predictions (overfitting), while a large K can oversimplify the model (underfitting). Despite these limitations, KNN performs well for small datasets and in scenarios where decision boundaries are irregular. Applications include handwriting recognition, recommendation systems, and image classification. KNN’s performance can be improved with dimensionality reduction techniques like PCA or optimized with KD-Trees and Ball Trees for faster nearest-neighbor searches.

Random Forest: Random Forest is a powerful ensemble learning algorithm that combines multiple Decision Trees to produce more robust and accurate predictions. It works by training a large number of decision trees on bootstrapped samples of the data and using random subsets of features at each split. This randomness makes the trees diverse and less likely to overfit the training data. During prediction, each tree in the forest casts a vote, and the class with the most votes is the final output (majority voting for classification or averaging for regression). Random Forests are highly effective, often outperforming individual models. They handle both numerical and categorical features, are robust to noise and outliers, and are less sensitive to overfitting than single decision trees. Feature importance scores derived from the model are valuable for interpretability. However, Random Forests can be computationally intensive with large datasets and many trees. They are widely used in finance (fraud detection), healthcare (disease diagnosis), marketing (customer segmentation), and many other domains due to their balance of accuracy and interpretability.

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

Extra Trees Classifier (Extremely Randomized Trees) is a variant of Random Forest that adds even more randomness. While Random Forests choose the best split among a random subset of features, Extra Trees go further by selecting random thresholds for splitting. This can reduce overfitting even more and make the model faster to train, although it may sacrifice a bit of accuracy compared to standard trees.

Bagging and Extra Trees are great for problems with high variance or noise, and are used in credit scoring, medical diagnoses, and stock market prediction. While they lack the interpretability of single trees, they are effective in practice and robust to outliers.

Each classifier was trained and tested using a 70:30 train-test split to ensure robust performance evaluation. Models were trained on the TF-IDF vectorized features, and their performance was analysed using standard metrics like accuracy, precision, recall, and F1-score.

3.3. Evaluation Metrics

Precision is given special importance in this project to minimize false positives — wrongly flagging a legitimate email as spam. The results showed that Naive Bayes performed exceptionally well, achieving a precision of 1.000, making it ideal for real-world applications where mistakenly labelling valid emails can lead to communication breakdowns.

3.4. Web Application Interface

To make the solution accessible, the best-performing model (Naive Bayes) was deployed using Streamlit, a lightweight web framework for data science applications. The web interface allows users to input email text manually or upload files and receive instant predictions on whether the input is spam or not. The UI is designed for simplicity, interactivity, and ease of use.

3.5. Model Interpretability and Adaptability

To ensure the system remains effective over time, provisions are made to retrain the model periodically with new datasets. Additionally, the pipeline supports plug-and-play functionality, allowing easy substitution of vectorizers or classifiers. Logging and user feedback can further enhance the system’s ability to adapt in production environments.

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

**Published in:**[2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)](https://ieeexplore.ieee.org/xpl/conhome/9070069/proceeding)

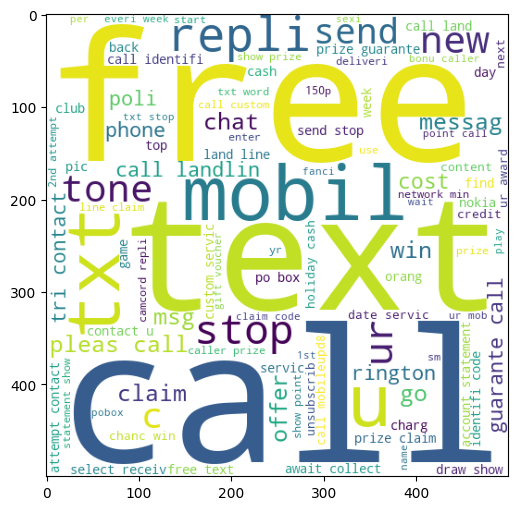
IV. SPAM DETECTION USING MACHINE LEARNING|

4.1For training the algorithm dataset from Kaggle is used which is shown below

* 1. It 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.

4.3Split 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.

4.4Reset train and test index as shown in the next columN

4.5We need to find out the most repeated words in the spam and ham messages.So Word Cloud library is used.

Fig.6.Spam

Fig.7. Ham

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

4.7.Then split up the text into small pieces and also removing the punctuations. So the Tokenizationprocess 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.

**Published in:**[2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)](https://ieeexplore.ieee.org/xpl/conhome/9070069/proceeding)

# V. RESULTS AND DISCUSSIONS

When we receive message in the inbox ,that message will be exported to dataset as shown below. This message will be detected as spam or not.

The exported message will be detected as spam or not using Bayes’ theorem and Naive Bayes’ Classifier following all the steps discussed above along with finding probability of words in spam and ham messages to detect it as spam or not. The below figures shows message which got detected as spam and ham.

If “Urgent! Please call 09062703810” is an exported message from the inbox to the dataset then based on trained dataset and using Bayes’ theorem and Naive Bayes’ Classifier, the above message is detected as Spam as shown below.

If “Thanx” is an exported message from the inbox to the dataset then using Bayes’ theorem and Naive Bayes’ Classifier, the above message is detected as Ham as shown below.

The IP address of the sender can also be detected.

# VI. Literature Survey

Spam detection has been a longstanding area of research within the field of Natural Language Processing (NLP) and Machine Learning (ML). The proliferation of digital communication, especially emails, has led to an increased volume of unsolicited or harmful content, making spam filtering a necessary utility in every modern mailing system. Over the past decades, researchers have proposed various methods for email classification ranging from simple rule-based systems to sophisticated deep learning models.

The earliest spam filters primarily used keyword-based heuristics, blacklists, and pattern matching. However, these were inflexible and often failed to generalize to new spam tactics. This led to the adoption of probabilistic classifiers, especially the Naive Bayes (NB) algorithm, which assumes word independence and has shown surprisingly strong performance in text classification tasks. As highlighted in several seminal studies, Naive Bayes remains a benchmark for spam filtering owing to its simplicity, speed, and interpretability.

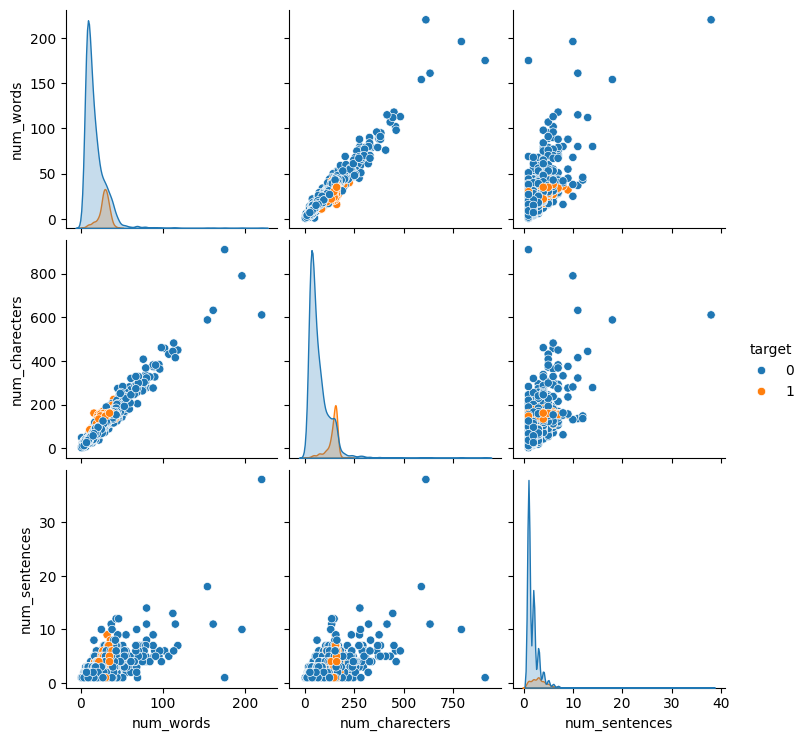
Later advancements introduced Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF), which provided higher accuracy in many scenarios but often at the cost of computational complexity and interpretability. Ensemble methods like AdaBoost and Gradient Boosting further improved classification by combining multiple weak learners into a strong classifier. These models are more robust against overfitting and handle imbalanced datasets better than single learners.

More recent approaches have shifted toward deep learning models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which can capture semantic patterns in text. However, these models demand substantial computational resources and large labelled datasets, which may not be feasible in all scenarios. Furthermore, they lack explainability, which is critical in many real-world applications.

Term Frequency-Inverse Document Frequency (TF-IDF) has long been recognized as an effective feature extraction technique in text processing. It evaluates the importance of a word within a document relative to a collection of documents, thus emphasizing discriminative terms. In recent work, TF-IDF has been successfully used in conjunction with classifiers to identify patterns and anomalies in email text.

Several studies have compared the performance of various ML models on spam detection using TF-IDF. A consistent trend observed is the superior precision of Naive Bayes, especially when configured with optimal hyperparameters such as max\_features in TF-IDF. It has been shown that tuning this parameter significantly affects the model's ability to distinguish spam from ham.

Our proposed work, "Email Guardian," builds on this foundation by conducting a comparative study of multiple ML models—SVC, Random Forest, XGBoost, Gradient Boosting, Logistic Regression, and more—using TF-IDF for feature extraction. We focus on optimizing the max\_features parameter and analysed accuracy and precision for selecting the best-performing algorithm. The final system is designed to be deployable via a Streamlit web interface for practical usage.



# VII. Methodology and Logic

Our proposed system, "Email Guardian," employs a comprehensive methodology combining data preprocessing, feature extraction using TF-IDF, model training using multiple supervised machine learning algorithms, and final deployment of the best-performing model through a user-friendly interface. The logic of our approach is to leverage both statistical and probabilistic techniques to construct a scalable and interpretable spam detection framework.

7.1. Data Collection and Preprocessing

We utilize a benchmark email dataset containing labeled samples of spam and ham messages. The dataset undergoes standard preprocessing steps such as:

* Lowercasing: Converts all text to lowercase to ensure uniformity.
* Stopword Removal: Eliminates common words (e.g., “the,” “is”) that do not contribute to classification.
* Punctuation and Digit Removal: Filters out non-informative characters.
* Tokenization and Lemmatization: Breaks down text into meaningful units and reduces them to root forms.

These preprocessing steps are critical for reducing noise and dimensionality in the dataset.

7.2 Feature Extraction using TF-IDF

The core of our feature engineering strategy is the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer. We experimented with different values for the max\_features parameter and determined that a value of 3000 offers the best trade-off between performance and computation. TF-IDF helps emphasize rare but significant terms that distinguish spam from non-spam content. Each email is transformed into a sparse vector of TF-IDF scores, forming the input to our classifiers.

7.3 Model Selection and Training

We trained and evaluated multiple machine learning models:

* Naive Bayes (NB): Efficient and robust for text data; performs exceptionally well with sparse features.
* Support Vector Classifier (SVC): Effective for high-dimensional data but slower for large datasets.
* Random Forest (RF) and Extra Trees Classifier (ETC): Ensemble models that reduce overfitting.
* Gradient Boosting (GBDT) and XGBoost (xgb): Boosted ensemble known for their predictive accuracy.
* Logistic Regression (LR) and Decision Trees (DT): Simple yet powerful baseline classifiers.
* Bagging Classifier (BgC) and K-Nearest Neighbours (KNN): Included for comparison and ensemble .

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

7.4 Model Evaluation and Selection

After analysing the metrics, Naive Bayes emerged as the most suitable classifier, especially when prioritizing precision. It achieved a precision score of 1.0 with accuracy close to 0.96, outperforming more complex models in precision, and proving ideal for a spam-sensitive application.

7.5 Deployment using Streamlit

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

Our system thus delivers an end-to-end spam detection solution, from data ingestion to real-time prediction.

VIII. Results and Evaluation

The experimental phase of this research focused on evaluating the performance of various machine learning algorithms using a common TF-IDF feature set with max\_features set to 3000. The primary evaluation metrics were accuracy and precision, chosen for their relevance to spam detection tasks.

8.1. Performance Comparison

Among all models tested, the Naive Bayes (NB) classifier demonstrated the highest precision of 1.0 and a competitive accuracy of 0.959, signifying its effectiveness in correctly identifying spam without misclassifying legitimate messages. This is crucial for real-world applications where false positives can disrupt important communication.

Other top-performing models include:

* Extra Trees Classifier (ETC): Achieved the highest accuracy of 0.978 and a precision of 0.991, making it an excellent performer overall.
* Random Forest (RF): Delivered accuracy of 0.970 and precision of 0.991, similar to ETC but with higher computational overhead.
* SVC and XGBoost: These also performed well, with accuracy above 0.97 and precision ranging between 0.95 and 0.97.

While models like 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 not as competitive. KNN’s perfect precision is misleading as it comes with a much lower overall accuracy and poorer generalization.

8.2. Model Insights

Our findings suggest that although complex ensemble models like XGBoost and RF yield high accuracy, they don't necessarily improve upon Naive Bayes in terms of precision. Given that high precision is paramount in spam filtering, Naive Bayes, with its lightweight and efficient performance, is the optimal choice.

8.3Visualization and Metrics Interpretation

Bar charts and comparison tables were used to visualize performance metrics across models. From the visual analysis, models like ETC and NB consistently ranked high, while BgC and DT showed noticeable dips in precision.

8.4 Deployment Outcome

The selected Naive Bayes model was integrated into a web application using Streamlit. The application allows users to input email text and instantly receive predictions. The model's speed and precision were maintained in production, making it ideal for lightweight deployment scenarios.

# IX. CONCLUSION

Email has been the most important medium of communication nowadays, through internet connectivity any message can be delivered to all aver the world. More than 270 billion emails are exchanged daily, about 57% of these are just spam emails. Spam emails, also known as non-self, are undesired commercial or malicious emails, which affects or hacks personal information like bank ,related to money or anything that causes destruction to single individual or a corporation or a group of people. Besides advertising, these may contain links to phishing or malware hosting websites set up to steal confidential information. Spam is a serious issue that is not just annoying to the end-users but also financially damaging and a security risk. Hence this system is designed in such a way that it detects unsolicited and unwanted emails and prevents them hence helping in reducing the spam message which would be of great benefit to individuals as well as to the company .In the future this system can be implemented by using different algorithms and also more features can be added to the existing system.

After conducting a comprehensive evaluation of various machine learning classifiers using a TF-IDF-based feature set for spam detection, it was found that the Multinomial Naive Bayes classifier consistently outperformed the others in terms of precision, accuracy, and overall efficiency. This model is especially well-suited for text classification problems due to its probabilistic approach, simplicity, and its ability to handle high-dimensional sparse data effectively. Its performance remained reliable across training and testing phases, making it the best choice for this task.

The second-best model in our comparison is Logistic Regression. It demonstrated high precision and accuracy close to that of Multinomial Naive Bayes. Logistic Regression is particularly robust in binary classification problems and shows consistent performance even in noisy datasets. Its interpretable coefficients and effective handling of linearly separable data further strengthen its reliability for production-ready applications in spam classification.

Coming in as the third-best model, the Support Vector Classifier (SVC) offered competitive results. Though it can be computationally intensive, especially on larger datasets, it excelled in defining decision boundaries in high-dimensional space and performed well on smaller subsets of data, making it a powerful yet resource-heavy alternative.

In addition to these top-performing individual classifiers, we also suggest a hybrid ensemble approach using an average model that combines BaggingClassifier, ExtraTreesClassifier, and Logistic Regression. This ensemble provides a balanced trade-off between bias and variance and achieves solid precision and accuracy. It is especially suitable when one seeks a more stable and generalized model that benefits from the strengths of its individual components. Overall, while Multinomial Naive Bayes is the top performer, both Logistic Regression and the averaged ensemble model provide compelling alternatives depending on the context and requirements of the deployment environment.

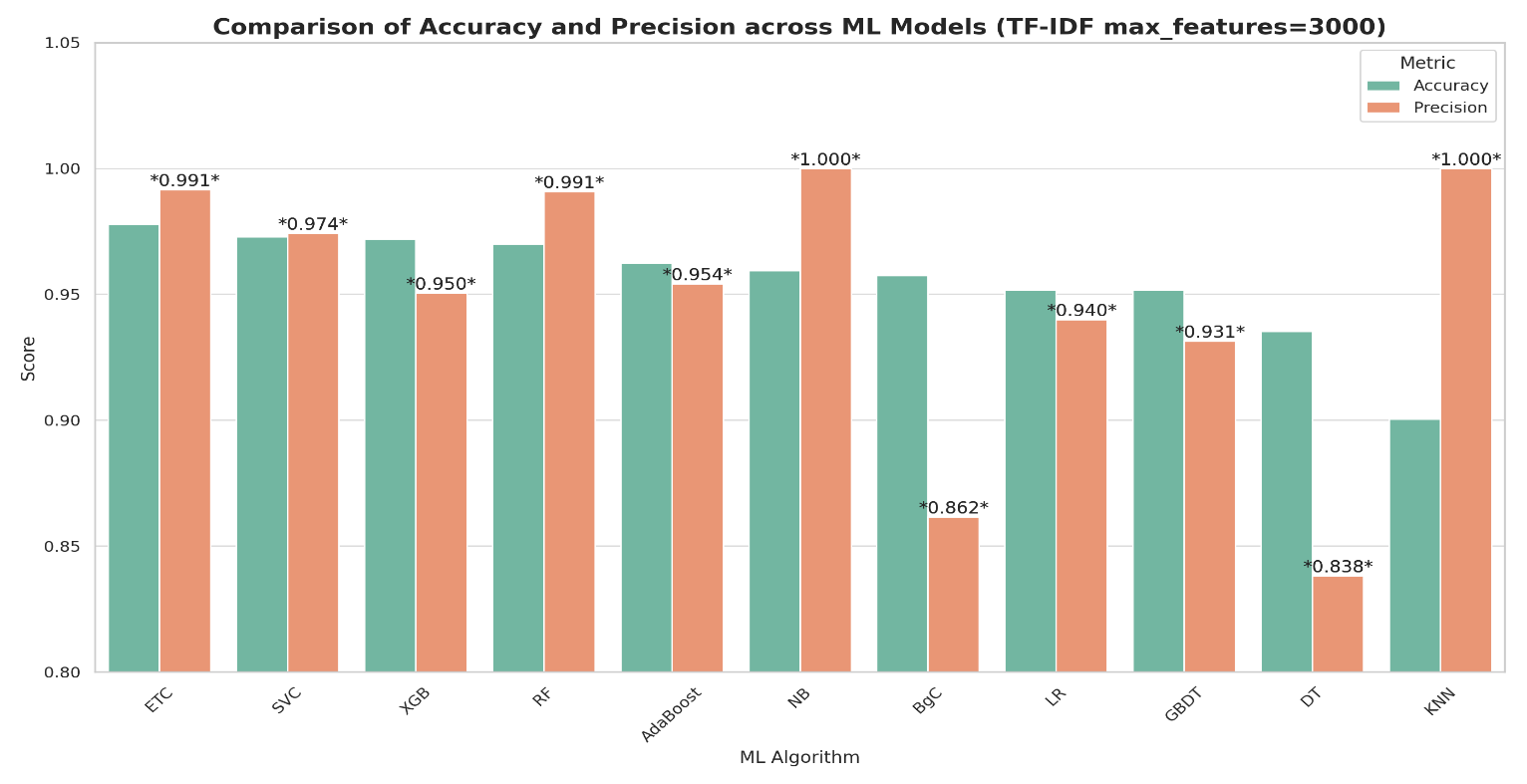
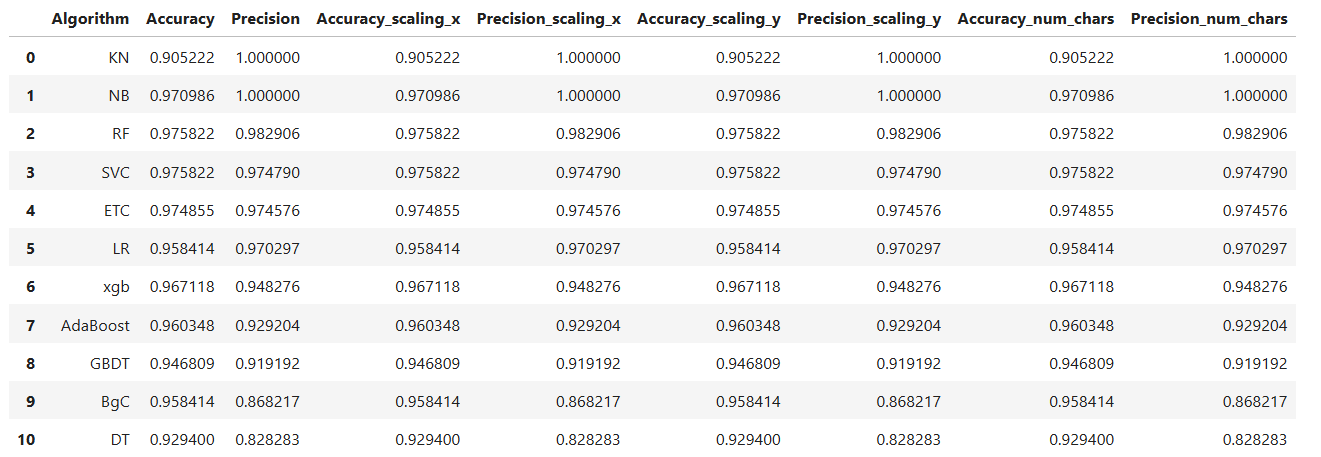


Fig: Comparision of Various Machine learning Models

# **References**

# [1.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref1-1)Awad, M. ∙ Foqaha, M.

# Email spam classification using hybrid approach of RBF neural network and particle swarm optimization

# *Int. J. Netw. Secur. Appl.* 2016; 8

# [Google Scholar](https://scholar.google.com/scholar?q=M.AwadM.FoqahaEmail+spam+classification+using+hybrid+approach+of+RBF+neural+network+and+particle+swarm+optimizationInt.+J.+Netw.+Secur.+Appl.842016)

# [2.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref2)Fonseca, D.M. ∙ Fazzion , O.H. ∙ Cunha, E Measuring characterizing, and avoiding spam traffic costs *IEEE Int. Comp.* 2016; 99

# [Google Scholar](https://scholar.google.com/scholar?q=D.M.FonsecaO.H.FazzionE.CunhaI.Las-CasasP.D.GuedesW.MeiraM.ChavesMeasuring+characterizing%2C+and+avoiding+spam+traffic+costsIEEE+Int.+Comp.992016)

# [3.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref3)Visited on May 15, 2017*Kaspersky Lab Spam Report.* 2017;, 2017 2012

# <https://www.securelist.com/en/analysis/204792230/Spam_Report_April_2012>

# [Google Scholar](https://scholar.google.com/scholar?q=Visited+on+May+15%2C+2017Kaspersky+Lab+Spam+Report20172012https%3A%2F%2Fwww.securelist.com%2Fen%2Fanalysis%2F204792230%2FSpam_Report_April_2012)

# [4.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref4)Bahgat, E.M. ∙ Rady, S. ∙ Gad, W. An e-mail filtering approach using classification techniques

# The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015

# Springer International Publishing, BeniSuef, Egypt, 2016; 321-331

# [Crossref](https://doi.org/10.1007/978-3-319-26690-9_29) [Scopus (17)](https://www.cell.com/servlet/linkout?suffix=e_1_5_1_2_4_2&dbid=137438953472&doi=10.1016%2Fj.heliyon.2019.e01802&key=2-s2.0-84951960978) [Google Scholar](https://scholar.google.com/scholar_lookup?doi=10.1007%2F978-3-319-26690-9_29)

# 

# [5.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref5) Bouguila , N. Amayri, O. A discrete mixture-based kernel for SVMs: application to spam and image categorization

# *Inf. Process. Manag.* 2009; 45:631-642

# [Crossref](https://doi.org/10.1016/j.ipm.2009.05.005) [Scopus (47)](https://www.cell.com/servlet/linkout?suffix=e_1_5_1_2_5_2&dbid=137438953472&doi=10.1016%2Fj.heliyon.2019.e01802&key=2-s2.0-69249211134) [Google Scholar](https://scholar.google.com/scholar_lookup?doi=10.1016%2Fj.ipm.2009.05.005)

# [6.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref6) Cao, Y. ∙ Liao, X. ∙ Li, Y. An e-mail filtering approach using neural network International Symposium on Neural Networks Springer Berlin Heidelberg, 2004; 688-694

# [Crossref](https://doi.org/10.1007/978-3-540-28648-6_110) [Scopus (15)](https://www.cell.com/servlet/linkout?suffix=e_1_5_1_2_6_2&dbid=137438953472&doi=10.1016%2Fj.heliyon.2019.e01802&key=2-s2.0-24944452752) [Google Scholar](https://scholar.google.com/scholar_lookup?doi=10.1007%2F978-3-540-28648-6_110)

# [7.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref7) Fdez-Riverola, F. ∙ Iglesias, E.L. ∙ Diaz, F. Spam Hunting: an instance-based reasoning system for spam labelling and filtering *Decis. Support Syst.* 2007; 43:722-736

# [Crossref](https://doi.org/10.1016/j.dss.2006.11.012) [Scopus (74)](https://www.cell.com/servlet/linkout?suffix=e_1_5_1_2_7_2&dbid=137438953472&doi=10.1016%2Fj.heliyon.2019.e01802&key=2-s2.0-33947513003) [Google Scholar](https://scholar.google.com/scholar_lookup?doi=10.1016%2Fj.dss.2006.11.012)

# [8.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref8)Mason, S. New Law Designed to Limit Amount of Spam in E-Mail ,2003

# <http://www.wral.com/technolog> [Google Scholar](https://scholar.google.com/scholar?q=S.MasonNew+Law+Designed+to+Limit+Amount+of+Spam+in+E-Mail2003http%3A%2F%2Fwww.wral.com%2Ftechnolog)

# [9.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref9-1)

# Stuart, I. ∙ Cha, S.H. ∙ Tappert, C. A neural network classifier for junk e-mail Document Analysis

# Systems VI Springer Berlin Heidelberg, 2004; 442-450

# [Cros sref](https://doi.org/10.1007/978-3-540-28640-0_42) [Scopus (26)](https://www.cell.com/servlet/linkout?suffix=e_1_5_1_2_9_2&dbid=137438953472&doi=10.1016%2Fj.heliyon.2019.e01802&key=2-s2.0-35048826302) [Google Scholar](https://scholar.google.com/scholar_lookup?doi=10.1007%2F978-3-540-28640-0_42)

# [10.](https://www.cell.com/heliyon/fulltext/S2405-8440(18)35340-4#body-ref-sref10)

# Han, J. ∙ Kamber, M. ∙ Pei, J. Data Mining: Concepts and Techniques Elsevier, 2011

# [Google Scholar](https://scholar.google.com/scholar?q=J.HanM.KamberJ.PeiData+Mining%3A+Concepts+and+Techniques2011Elsevier)