University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)

ISSN: 2714-3627

A Journal of the Department of Computer Science, University of Ibadan, Ibadan, Nigeria

Volume 14 No. 1, June, 2025

journals.ui.edu.ng/uijslictr
http://uijslictr.org.ng/
uijslictr@gmail.com

University of Ibadan
Journal of Science and Logics in ICT Research

Spam Detection In Email Communication Using Ensemble Learning

[1]✉ Nureni Ayofe Azeez, [1]Alfred Dennis Tonye, [1]Chinyere Chioma Isiekwene.

Department of Computer Sciences, Faculty of Science, University of Lagos, Nigeria.
nazeez@unilag.edu.ng; alfreddennis26.adt90@gmail.com; isiekwenechioma@gmail.com

Abstract
Spam detection remains a critical challenge in cybersecurity due to the increasing sophistication of unsolicited and malicious communications. These messages, often containing phishing links, fraudulent offers, and malware, pose significant risks to users and information systems. This project addresses the challenge by implementing a robust spam detection system using ensemble learning techniques to enhance the security of email and SMS communications. Utilizing diverse datasets such as the UCI ML Corpus, Spam Assassin Dataset, Ling Phishing Dataset, Nigerian Fraud Dataset, and Enron Phishing Dataset, the study implemented rigorous data preprocessing and feature extraction, transforming raw text data into numerical vectors using Term Frequency Inverse Document Frequency (TFIDF) vectorization. Various Machine Learning algorithms in cluding Support Vector Machine, Logistic Regression, Naïve Bayes, Decision Trees, KNN, Extra Trees.
Also, a range of ensemble learning algorithms, including Random Forest, AdaBoost, Gradient Boosting, and X GBoost, were implemented with their performance recorded. The project focuses on combining the efforts of some of these algorithms hereby comparing two primary ensemble models; the Stacking and Voting Classifiers, with the Voting Classifier emerging as the more effective. By aggregating the strengths of multiple models, the Voting Classifier demonstrated superior accuracy and reliability combining models like SVC, RF, ETC, and NB, to report accuracy and precision scores of around 98% and 99% for datasets 1 and 2, 97% and 97% for dataset 3 and 99% and 99% for dataset 5 respectively. This project underscores the potential of ensemble methods in enhancing spam detection systems and sets the stage for future research exploring the integration of deep learning models and real-time detection systems to secure digital communications further.

Keywords: *Spam detection models, Cybersecurity, Ensemble techniques*

1. Introduction

Spam detection has become an essential aspect of information security in today's digital age, where the proliferation of electronic communication methods such as email and SMS presents significant challenges. Spam messages, which are unsolicited and often malicious communications, pose various threats, including phishing attacks, the dissemination of malware, and the invasion of privacy [1]. As a result, the development and implementation of effective spam detection techniques are crucial for safeguarding users and maintaining the integrity of communication networks [2].

The term "spam" refers to any unwanted communication that is sent in bulk, typically for advertising, phishing, or malicious purposes. The rise of spam has been facilitated by the low cost of sending messages and the availability of automated tools that enable spammers to target a large number of recipients simultaneously [36]. Spam can take various forms, including email spam, SMS spam, and spam on social media platforms. The primary objectives of spam detection systems are to accurately identify and filter out these unwanted messages while minimizing false positives, which can lead to the misclassification of legitimate communications [28].

Several machine learning algorithms have been employed in spam detection to enhance the accuracy and efficiency of identifying spam messages. Commonly used algorithms include Support Vector Machine (SVM), Naïve Bayes, Random Forest, Logistic Regression, and K-Nearest Neighbors (k-NN) [8],[28]. These algorithms leverage various features extracted from the messages, such as word frequency, presence of specific keywords, and message metadata, to distinguish between spam and legitimate messages [10].

Azeez N. A., Tonye A. D., Isiekwene C. C. (2025). Spam Detection In Email Communication Using Ensemble Learning. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 14 No. 1, pp. 154 – 172.

One significant challenge in spam detection is the evolving nature of spam techniques. Spammers continuously adapt their methods to bypass existing filters, necessitating ongoing research and updates to spam detection systems [11]. For instance, spammers may use obfuscation techniques, such as misspelling words or inserting random characters, to evade keyword-based filters. Advanced machine learning approaches, including deep learning and ensemble methods, have shown promise in addressing these challenges by capturing complex patterns and relationships within the data [15].

The effectiveness of spam detection systems is often measured using various metrics, including accuracy, precision, recall, F1 score, and ROC AUC score [30]. These metrics provide insights into the system's ability to correctly classify spam and non-spam messages, as well as its overall performance in real-world scenarios. Ensemble methods, such as Voting Classifier and Stacking Classifiers, have gained popularity due to their ability to combine the strengths of multiple algorithms, leading to improved detection rates and robustness against diverse spam techniques [29], [30]. The integration of machine learning in spam detection not only enhances the accuracy and efficiency of filtering systems but also provides a scalable solution capable of handling large volumes of data [25],[35]. As communication technologies continue to evolve, ongoing research and development in spam detection are essential to stay ahead of spammers and ensure the security and reliability of digital communication channels.

By leveraging the power of machine learning and ensemble methods, this study aims to implement a comprehensive spam detection system that effectively identifies and filters out spam messages in both email and SMS platforms. The implementation of advanced algorithms and techniques will contribute to the development of robust and adaptable spam detection solutions, addressing the ever-changing landscape of spam and its associated threats [1].

2. Related Works

Today, spam detection continues to evolve with the advancement of artificial intelligence and machine learning. The integration of natural language processing (NLP) techniques and the development of real-time detection capabilities are at the forefront of current research. The focus is on creating models that can adapt to new spam tactics quickly and efficiently, ensuring robust protection against evolving threats [21]. In addition to technical advancements, regulatory measures and industry standards play a crucial role in combating spam. Legislation such as the CAN-SPAM Act in the United States and the General Data Protection Regulation (GDPR) in Europe set guidelines for email marketing and data protection, helping to mitigate the impact of spam [14],[18].

The ubiquitous nature of email communication has brought with it a persistent challenge: spam emails. These unsolicited and often deceptive messages clutter inboxes, disrupt workflow, and harbor the potential for phishing attacks and malware distribution [20]. Traditional rule-based spam filters, while initially effective, struggle to keep pace with the evolving tactics of spammers [25]. Perpetrators employ increasingly sophisticated techniques, crafting emails that mimic legitimate sources and exploiting vulnerabilities in filtering systems [5].

Machine learning (ML) has emerged as a powerful tool in the fight against spam [33]. ML algorithms excel at analyzing large datasets and identifying patterns within them. By leveraging these capabilities, we can develop models that can discern legitimate emails from spam with high accuracy [37].

Several research studies have explored the efficacy of different ML approaches in spam detection. [38] proposed a lightweight spam detection model using word frequency patterns, demonstrating the potential of simple yet effective ML techniques. Research by [2] explored SMS spam detection using ML, highlighting the versatility of these algorithms across different communication platforms. [3] conducted a comprehensive review of various ML methods for enhancing email spam filter accuracy. Their work underscores the continuous evolution and improvement in this domain.

Ensemble methods, which combine multiple algorithms, have also shown promise in

improving detection rates [9]. This approach leverages the strengths of diverse algorithms to tackle the multifaceted nature of spam. Natural Language Processing (NLP) techniques, when combined with ML, can further enhance spam detection capabilities. [3] integrated NLP and swarm intelligence to filter spam emails, showcasing advanced methods to refine the detection process. Comparative studies, such as those by [4],[13],[19], have evaluated multiple ML models to identify the most effective ones for spam detection. These studies provide valuable insights into model selection and optimization.

The practical implementation of these models involves several critical steps. Data acquisition and preprocessing are crucial initial phases. Researchers often utilize publicly available datasets, such as the UCI Machine Learning Repository's SMS Spam Collection [16]. Data cleaning and preparation involve removing irrelevant information and handling missing values to ensure model accuracy [17].

Exploratory Data Analysis (EDA) helps visualize data distribution and uncover patterns that can inform feature engineering [22]. Feature engineering involves extracting specific attributes, such as word frequency and key phrases, to enhance model performance [26].

Developing a robust ML model is central to the project. Algorithms like Naive Bayes, Support Vector Machines (SVM), and ensemble methods are popular choices for spam detection [34]. These algorithms are trained on the prepared data, with hyperparameter tuning applied to optimize their accuracy [7]. Model evaluation using metrics like accuracy, precision, recall, and F1-score ensures comprehensive assessment and fine-tuning [1].

In summary, the historical evolution of spam and spam detection highlights the ongoing battle between spammers and the developers of spam detection systems. From simple keyword filters to sophisticated machine learning and deep learning models, the field has made significant strides. However, the continuous evolution of spam tactics necessitates ongoing research and innovation to stay ahead in this ever-changing landscape. By leveraging advanced algorithms, this project aims to provide an effective tool to combat spam, safeguarding users against potential cyber threats. Future enhancements may include capabilities for bulk message processing and detailed analytics, further expanding the system's utility and effectiveness in the ongoing battle against spam.

3. Methodology

The methodology adopted entails four stages: data collection and labelling, pre-processing, classification, performance evaluation

3.1 Data collection and labelling

Data preprocessing is a vital step in preparing the raw datasets for machine learning models. This process ensures the data is clean, consistent, and suitable for analysis. Effective preprocessing enhances model performance by eliminating noise, correcting inconsistencies, and transforming data into a format that the machine learning algorithms can effectively utilize. In this study, several data preprocessing techniques, including dropping irrelevant columns, renaming columns, handling missing values, and concatenating datasets were employed. These steps help to standardize the data and make it ready for the subsequent stages of feature extraction and model training [12]. It is composed of a broad ML procedure called Data Cleaning and then an optional but useful phase called Exploratory Data Analysis (EDA).

3.2. Selection and Comparison of the Machine Learning Algorithms

The study ensures that various machine learning algorithms are utilized and compared to determine the most effective model for spam detection. These algorithms include Logistic Regression, Support Vector Classifier (SVC), Naive Bayes (with a preference for Multinomial Naive Bayes), Decision Tree Classifier, K-Nearest Neighbors (KNN), Random Forest, AdaBoost, Extra Trees Classifier, Gradient Boosting Classifier, and XGBoost Classifier. Each algorithm offers unique advantages and is well-suited for different aspects of the spam detection task. Below is an extensive discussion of each algorithm.

Logistic Regression: Logistic Regression is a linear model used for binary classification tasks. It estimates the probability that a given input belongs to a certain class. The model uses the logistic function to squeeze the output of a linear equation between 0 and 1, representing the probability of the positive class [24].

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \quad \textbf{(1)}$$

Where:

θ is the vector of coefficient (parameters) of the model

x is the input feature vector

$h_\theta(x)$ is the predicted probability that the instance belongs to class 1 (e.g. spam)

Decision Boundary:

Class 1 (spam) if $h_\theta(x) \geq 0.5$

Class 0 (not spam) if $h_\theta(x) < 0.5$

Support Vector Classifier (SVC): Support Vector Classifier (SVC) is used for both linear and non-linear classification tasks. It finds the optimal hyperplane that best separates the classes in the feature space. SVC maximizes the margin between the closest data points of different classes, known as support vectors, providing robustness against overfitting (Cortes and Vapnik, 1995). The decision function is:

Objective Function:

Minimize: $\frac{1}{2}\left||w|\right|^2$ **(2)**

Subject to: $y_i\left(w^T x_i + b\right) \geq 1, \forall_i$ **(3)**

Where:

w is the weight vector

x_i is the input feature vector for the i^{th} instance

y_i is the label for the i^{th} instance

b is the bias term

Kernel Trick: The kernel trick is used to transform the data into a higher-dimensional space to make it linearly separable.

Naïve Bayes: Naive Bayes is a probabilistic classifier based on Bayes' theorem. The Multinomial variant is particularly effective for text classification tasks like spam detection. It calculates the probability of each class given the input features and selects the class with the highest probability.

Bayes Theorem**:**

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \quad \textbf{(4)}$$

Where:

$P(y|x)$ is the posterior probability of class y given the features x

$P(x|y)$ is the likelihood of features x given class y

$P(y)$ is the prior probability of class y

$P(x)$ is the evidence or the total probability of features x

Decision Trees Classifier: Decision Tree Classifier splits data into subsets based on feature values, forming a tree structure. The model makes decisions by splitting the data at each node based on the feature that results in the best separation of the classes, providing clear decisions [32].

For Gini Impurity:

$$Gini = 1 - \sum_{i=1}^{C} {P_i}^2 \quad \textbf{(5)}$$

Where:

C is the total number of classes

P_i is the probability of a random chosen element being classified to class *i*

Entropy:

$$Entropy = -\sum_{i=1}^{C} P_i \log_2(P_i) \quad \textbf{(6)}$$

Information Gain:

$$IG(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad \textbf{(7)}$$

Where S is the set of instances and A is the attribute

K-Nearest Neighbours (KNN): K-Nearest Neighbors (KNN) is a non-parametric algorithm used for classification. It classifies based on the majority class among the k-nearest neighbors of a data point. KNN calculates the distance between the input point and all other points in the training set. It typically uses distance metrics like Euclidean distance.

Distance Metric**:**

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad \textbf{(8)}$$

Where:

x and y are two instances in the feature space

n is the number of features

Prediction: The class label is assigned based on the majority class among the *k* nearest neighbors

$$y = mode\{yi_1, yi_2, yi_3, \dots, yi_k\}$$

Where: $yi_1, yi_2, yi_3, \dots, yi_k\}$ are the labels of the nearest neighbors

ExtraTrees Classifier: Extra Trees (Extremely Randomized Trees) is an ensemble learning method that uses random subsets of features and splits the nodes randomly to form multiple decision trees. It is similar to Random Forest but with more randomness in tree building. It builds each tree using a random subset of features and data, with the final prediction made by averaging the outputs of all trees [19].

Splitting Criterion:
Randomly selects a feature and then selects a random split point for that feature to split the node. The prediction is made by averaging the predictions from all trees or taking the majority vote in the case of classification.
Like Random Forest, the prediction is given by**:**

$$y = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad \textbf{(10)}$$

Where:

T is the number of trees

$h_t(x)$ is the prediction of the t^{th} tree

3.3. Ensemble Methods

Ensemble methods combine the predictions of multiple machine learning models to produce a more accurate and robust model. The underlying idea is that by aggregating the strengths of individual models, the ensemble can outperform any single model. Ensemble methods are particularly useful in scenarios where individual models might have different strengths and weaknesses. In this section, we will discuss the ensemble methods used in this project: Voting, Stacking, Boosting, and Bagging.

1. Boosting

Boosting is an iterative ensemble technique that focuses on improving the performance of weak learners by sequentially adding models that correct the errors of the previous models. Boosting reduces bias and variance, making it highly effective for complex classification tasks. The key algorithms are as follows.

AdaBoost Classsifier: AdaBoost (Adaptive Boosting) is a boosting technique that combines multiple weak classifiers to form a strong classifier by focusing on hard-to-classify instances. AdaBoost assigns weights to each training instance, adjusting them after each round to focus more on misclassified instances.

Weight Update**:**

$$w_i^{(t+1)} = w_i^{(t)} \times \exp(-\alpha_t y_i h_t(x_i)) \quad \textbf{(11)}$$

Where:

$w_i^{(t)}$ is the weight of the i^{th} instance at iteration t

α_t is the model weight for classifier h_t

y_i is the true label

$h_t(x_i)$ is the prediction of classifier h_t

Gradient Boosting Classifier: Gradient Boosting is an ensemble technique that builds models sequentially. Each new model minimizes the loss function of the previous models, with the final prediction being a weighted sum of all the models (Friedman, 2001).

Objective Function**:**

$$Obj = \sum_{i=1}^{n} L(y_i, \mathbb{y}_i^{(t)}) + \sum_{k=1}^{T} \Omega(f_k) \quad \textbf{(12)}$$

Where:

$L(y_i, \mathbb{y}_i^{(t)})$ is the loss function (e.g., log loss for classification)

$\Omega(f_k)$ is a regularization term to prevent overfitting

$\mathbb{y}_i^{(t)}$ is the prediction after t iterations

Boosting Process: At each step t, the algorithm fits a new model $f_t(x)$ to the residual errors from the previous model

$$y_1^{(t+1)} = y_1^{(t)} + \eta \, . f_t(x_i)$$

Where: η is the learning rate

XGBoost Classifier: XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that

includes regularization to prevent overfitting. XGBoost uses advanced techniques like parallelization, tree pruning, and handling missing values to build efficient and accurate models [8].

Objective Function**:**

$Obj = \sum_{i=1}^{n} l(y_i, \mathbb{y}_i^{(t)}) + \sum_{k=1}^{K} \Omega(f_k)$ **(13)**

Where:

$l(y_i, \mathbb{y}_i^{(t)})$ is the loss function

$\Omega(f_k)$ is a regularization term for tree k

Boosting methods can be particularly effective in improving the classification performance of the spam detection system by focusing on difficult-to-classify instances.

2. Bagging

Bagging (Bootstrap Aggregating) is an ensemble method that builds multiple models using different subsets of the training data. Each model is trained independently, and their predictions are averaged (or voted) to produce the final prediction. Bagging helps to reduce variance and prevent overfitting. The key algorithm is the Random Forest classifier.

Random Forest: Random Forest is an ensemble learning method that constructs multiple decision trees and merges their outputs to improve accuracy and control over-fitting. Each tree in the forest is built on a random subset of the data and features, with the final prediction made by aggregating the predictions of all the individual trees [6].

Prediction: The prediction is made by averaging the predictions from all the trees (for regression) or by taking the majority vote (for classification).

$$y = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad \textbf{(9)}$$

Where:

T is the number of trees

$h_t(x)$ is the prediction of the t^{th} tree

3. Voting

The Voting Classifier is a simple yet powerful ensemble technique that aggregates the predictions from multiple models. The two main types of voting mechanisms are as follows.

Hard Voting: In this approach, each model in the ensemble makes a prediction (votes for a class), and the class with the most votes is chosen as the final prediction. Mathematically, if we have mmm classifiers, the hard voting prediction is given by:

$$y = mode\{y_1, y_2, y_3, \ldots, y_m\}$$

Where: y_i is the prediction of the i^{th} model

Soft Voting: Instead of predicting the final class directly, soft voting averages the predicted probabilities of each class and selects the class with the highest average probability. This method often provides better performance as it considers the confidence of each model's prediction:

$$y = \arg\max(\frac{1}{m}\sum_{i=1}^{m} P_j^{(i)})$$

Where: $P_j^{(i)}$ is the predicted probability of class j by the i^{th} model

The Voting Classifier would be particularly effective in this project, as it leverages the diverse strengths of the models included in the ensemble, leading to improved accuracy and robustness in spam detection.

4. Stacking

Stacking is a more advanced ensemble method that involves training a meta-model to combine the predictions of several base models. The base models are first trained on the training data, and then their predictions are used as input features for the meta-model. The meta-model learns how to best combine these predictions to make a final prediction.

Let $h_1(x), h_2(x), \ldots, h_m(x)$ be the predictions from the base models.

The meta-model $H(x)$ is trained on these predictions:

$$y = H(h_1(x), h_2(x), \ldots, h_m(x))$$

Stacking can outperform other ensemble methods by learning the optimal way to integrate the strengths of each model. In this project, the Stacking Classifier will be tested and compared with other methods, to provide valuable insights into the effectiveness of model combination strategies.

3.4. Hyperparameter Tuning

Hyperparameter tuning is a crucial step in the ML pipeline that involves selecting the optimal set of hyperparameters for a model. Hyperparameters are parameters that are set before the learning process begins, and they

can significantly impact the performance of the model. The goal of hyperparameter tuning is to improve the model's accuracy, precision, recall, and overall performance by finding the best combination of hyperparameters. It involves systematically searching through a predefined space of hyperparameter values and evaluating the model's performance for each combination. Two common methods for hyperparameter tuning are Grid Search and Random Search.

3.5. Evaluation Metrics

Several evaluation metrics are used to comprehensively assess the performance of machine learning models. Each metric provides unique insights into different aspects of the model's performance. Below is a table that shows the evaluation metrics to be considered in this project.

Table 1. Evaluation Metrics

	Metric	Definition	Formular
1	**Accuracy**	**The ratio of correctly predicted instances to the total instances. It gives a quick overview of the model's overall performance**	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
2	**Precision**	**Measures the proportion of true positive predictions among all positive predictions. It indicates the accuracy of the positive class predictions**	$\text{Precision} = \frac{TP}{TP + FP}$
3	**Recall**	**Also known as sensitivity or true positive rate. Measures the proportion of true positive predictions among all actual positives**	$\text{Recall} = \frac{TP}{TP + FN}$
4	**F1 Score**	**The harmonic Mean of precision and recall, providing a single metric that balances both aspects**	$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
5	**ROC_AUC**	**The ROC curve plots the true positive rate (recall) against the false positive rate. The AUC represents the model's ability to distinguish between classes**	
6	**Logarithmic Loss**	**Measures the performance of a classification model where the output is a probability value between 0 and 1. It penalizes false classifications, with larger penalties for confident but incorrect predictions.**	$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$
7	**Hamming Loss**	**The fraction of labels that are incorrectly predicted. It measures the average misclassification rate.**	$\text{Hamming Loss} = \frac{1}{N}\sum_{i=1}^{N}[y_i \neq \hat{y}_i]$
8	**Cohen's Kappa**	**Measures the agreement between two raters (or the model and the true labels) accounting for the possibility of agreement occurring by chance**	$\kappa = \frac{p_o - p_e}{1 - p_e}$
9	**F2 Score**	**A variant of the F1 Score that gives more weight to recall than precision. It is useful when the cost of false negatives is higher**	$\text{F2 Score} = (1 + 2^2) \times \frac{\text{Precision} \times \text{Recall}}{(2^2 \times \text{Precision}) + \text{Recall}}$
10	**Jaccard Index**	**Measures the similarity between the predicted and true label sets. It is the size of the intersection divided by the size of the union of the label sets.**	$\text{Jaccard Index} = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$

4. Results and Discussion

In this section, various machine learning algorithms were implemented and evaluated to determine their effectiveness in detecting spam messages. The algorithms tested included Logistic Regression, Support Vector Classifier, Naive Bayes, Decision Tree Classifier, K-Neighbors, Random Forest, AdaBoost, Extra Trees Classifier, Gradient Boosting Classifier, and XGBoost.

In developing a the spam detection system, several critical decisions were made to enhance model performance and ensure practical applicability. This section reflects on these choices, offering insights into the rationale behind each decision.

Hyperparameter tuning was a pivotal step in refining the performance of the ML models. By systematically adjusting parameters such as the 'C' value in Support Vector Machines (SVM) and the number of estimators in ensemble methods like Random Forest, the models were optimized for better accuracy and generalization.

Feature selection and engineering played a crucial role in improving model accuracy. Initially, a broad range of features were considered, including text length, punctuation frequency, and keyword occurrences. However, a thorough analysis revealed that not all features contributed equally to the model's predictive power. Employing the TF-IDF Vectorizer over the Count Vectorizer yielded better results. Coupling it with hyperparameters such as max_features did much better as only features that significantly impacted performance were selected. This step was essential in reducing model complexity and preventing overfitting.

Table 2: Results obtained by individual base classifier [dataset 1]

	Algorithm	Accuracy	Precision	Recall	F1	ROC_AUC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	LR	0.9700	0.9972	0.9342	0.9647	0.9661	1.0796	0.0300	0.9387	0.9462	0.9318
2	SVM	0.9770	0.9972	0.9500	0.9730	0.9740	0.8305	0.0230	0.9529	0.9591	0.9475
3	NB	0.9585	0.9479	0.9578	0.9528	0.9585	1.4949	0.0415	0.9158	0.9559	0.9100
4	DT	0.9412	0.9687	0.8947	0.9302	0.9360	2.1178	0.0588	0.8796	0.9086	0.8695
5	KN	0.5507	0.4935	1.0000	0.6609	0.6004	16.195	0.4493	0.1803	0.8297	0.4935
6	RF	0.9827	0.9973	0.9631	0.9799	0.9806	0.6229	0.0173	0.9648	0.9698	0.9606
7	AdaBoost	0.9505	0.9856	0.9000	0.9409	0.9449	1.7856	0.0495	0.8984	0.9159	0.8883
8	ETC	0.9735	0.9786	0.9605	0.9695	0.9721	0.9551	0.0265	0.9461	0.9641	0.9407
9	GBDT	0.9643	0.9861	0.9316	0.9581	0.9607	1.2872	0.0357	0.9270	0.9420	0.9195
10	XGBoost	0.9758	0.9864	0.9579	0.9720	0.9738	0.8720	0.0242	0.9507	0.9634	0.9454

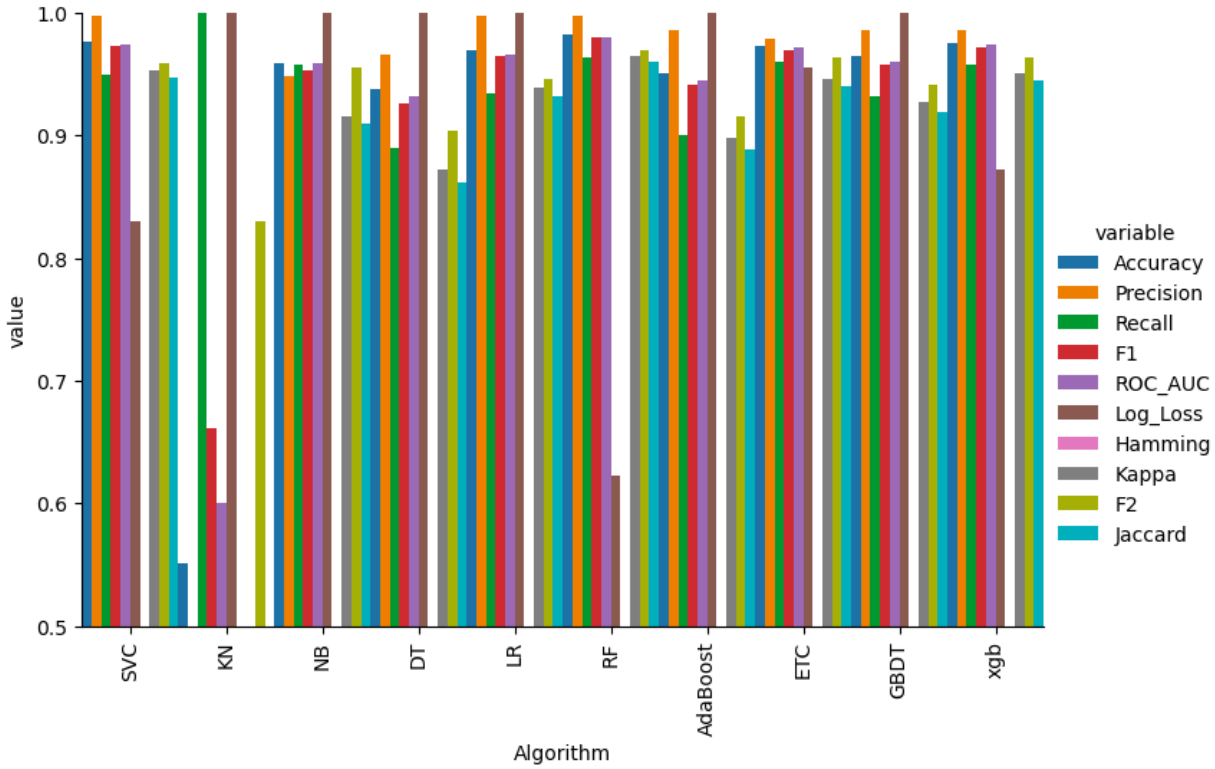

Figure 1: Barplot of the Model Evaluation (Dataset 1)

Table 3: Results Obtained by the Individual Models (Dataset 2)

	Model	Accuracy	Precision	Recall	F1	ROC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	LR	0.9468	0.9882	0.6087	0.7534	0.8038	1.9172	0.0532	0.7254	0.6593	0.6043
2	SVM	0.9758	0.9748	0.8406	0.9027	0.9186	0.8715	0.0242	0.8890	0.8643	0.8227
3	NB	0.9594	1.0000	0.6957	0.8205	0.8478	1.4640	0.0406	0.7984	0.7407	0.6957
4	DT	0.9352	0.8380	0.6377	0.7242	0.8094	2.3355	0.0648	0.6883	0.6697	0.5677
5	KN	0.9023	1.0000	0.2681	0.4228	0.6341	3.5207	0.0976	0.3883	0.3141	0.2681
6	RF	0.9778	1.0000	0.8333	0.9090	0.9167	0.8017	0.0222	0.8965	0.8621	0.8333
7	AdaBoost	0.9246	0.8409	0.5362	0.6549	0.7603	2.7190	0.0754	0.6148	0.5781	0.4868
8	ETC	0.9768	0.9750	0.8478	0.9070	0.9222	0.8366	0.0232	0.8938	0.8705	0.8298
9	GBDT	0.9516	0.8929	0.7246	0.8000	0.8556	1.7429	0.0484	0.7728	0.7530	0.6667
10	XGBoost	0.9700	0.9350	0.8333	0.8812	0.9122	1.0806	0.0300	0.8641	0.8519	0.7877

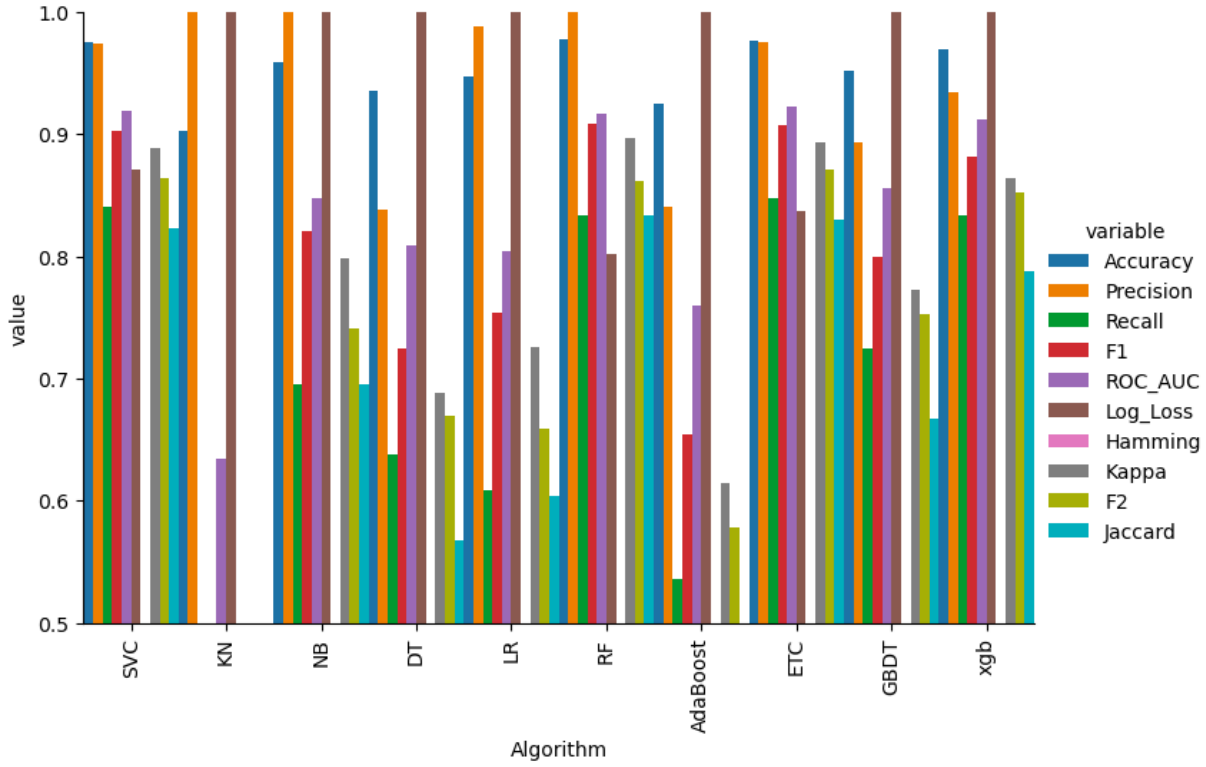

Figure 2: Barplot of the Model Evaluation (Dataset 2)

Table 4: Results Obtained by the Individual Models (Dataset 3)

	Model	Accuracy	Precision	Recall	F1	ROC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	LR	0.9628	0.9768	0.8934	0.9335	0.9426	1.3421	0.0372	0.9078	0.9094	0.8753
2	SVM	0.9787	0.9781	0.9485	0.9630	0.9699	0.7669	0.0213	0.9481	0.9543	0.9288
3	NB	0.9583	0.9639	0.8909	0.9260	0.9386	1.5018	0.0417	0.8971	0.9046	0.8621
4	DT	0.8927	0.8667	0.7485	0.8032	0.8504	3.8664	1.1073	0.7301	0.7695	0.6712
5	KN	0.9326	0.8324	0.9636	0.8933	0.9417	2.4285	0.0673	0.8444	0.9342	0.8071
6	RF	0.9583	0.9579	0.8970	0.9264	0.9403	1.5018	0.0417	0.8974	0.9085	0.8629
7	AdaBoost	0.9291	0.9085	0.8424	0.8742	0.9037	2.5563	0.0709	0.8249	0.8549	0.7765
8	ETC	0.9663	0.9771	0.9061	0.9402	0.9486	1.2142	0.0337	0.9168	0.9194	0.8872
9	GBDT	0.9362	0.9510	0.8242	0.8831	0.9033	2.3006	0.0638	0.8395	0.8468	0.7907
10	XGBoost	0.9672	0.9536	0.9333	0.9433	0.9573	1.1823	0.0328	0.9203	0.9373	0.8928

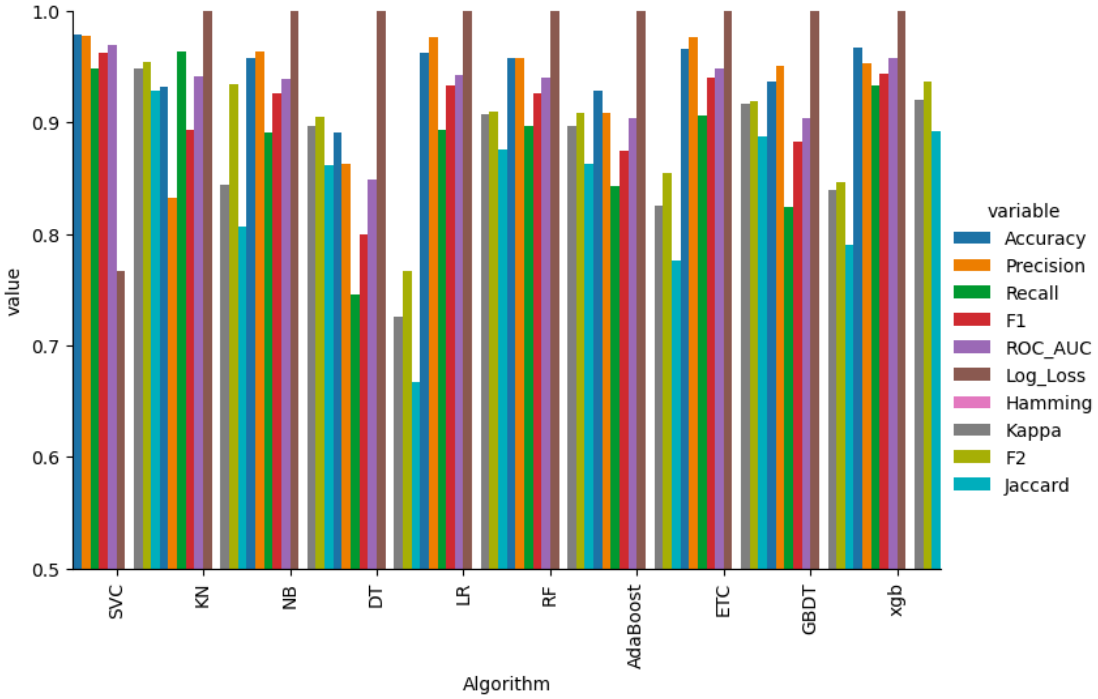

Figure 3: Barplot of the Model Evaluation (Dataset 3)

Table 5: Results Obtained by the Individual Models (Dataset 4)

	Model	Accuracy	Precision	Recall	F1	ROC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	LR	0.9859	0.9771	0.9945	0.9857	0.9861	0.5073	0.0141	0.9718	0.9910	0.9718
2	SVM	0.9849	0.9784	0.9909	0.9846	0.9850	0.5458	0.0151	0.9697	0.9884	0.9697
3	NB	0.9802	0.9824	0.9770	0.9797	0.9802	0.7128	0.0198	0.9604	0.9809	0.9602
4	DT	0.8455	0.7653	0.9861	0.8618	0.8487	5.5674	0.1545	0.6930	0.9323	0.7572
5	KN	0.7969	0.6259	0.9942	0.7682	0.7134	10.5633	0.2931	0.4212	0.8895	0.6237
6	RF	0.9783	0.9764	0.9792	0.9778	0.9783	0.7834	0.0217	0.9565	0.9786	0.9565
7	AdaBoost	0.8958	0.8511	0.9533	0.8994	0.8970	3.7565	0.1042	0.7920	0.9310	0.8171
8	ETC	0.9820	0.9846	0.9784	0.9815	0.9819	0.6486	0.0180	0.9640	0.9797	0.9637
9	GBDT	0.9211	0.8743	0.9792	0.9238	0.9224	2.8447	0.0789	0.8425	0.9563	0.8584
10	XGBoost	0.9674	0.9463	0.9894	0.9674	0.9679	1.1751	0.0326	0.9348	0.9805	0.9368

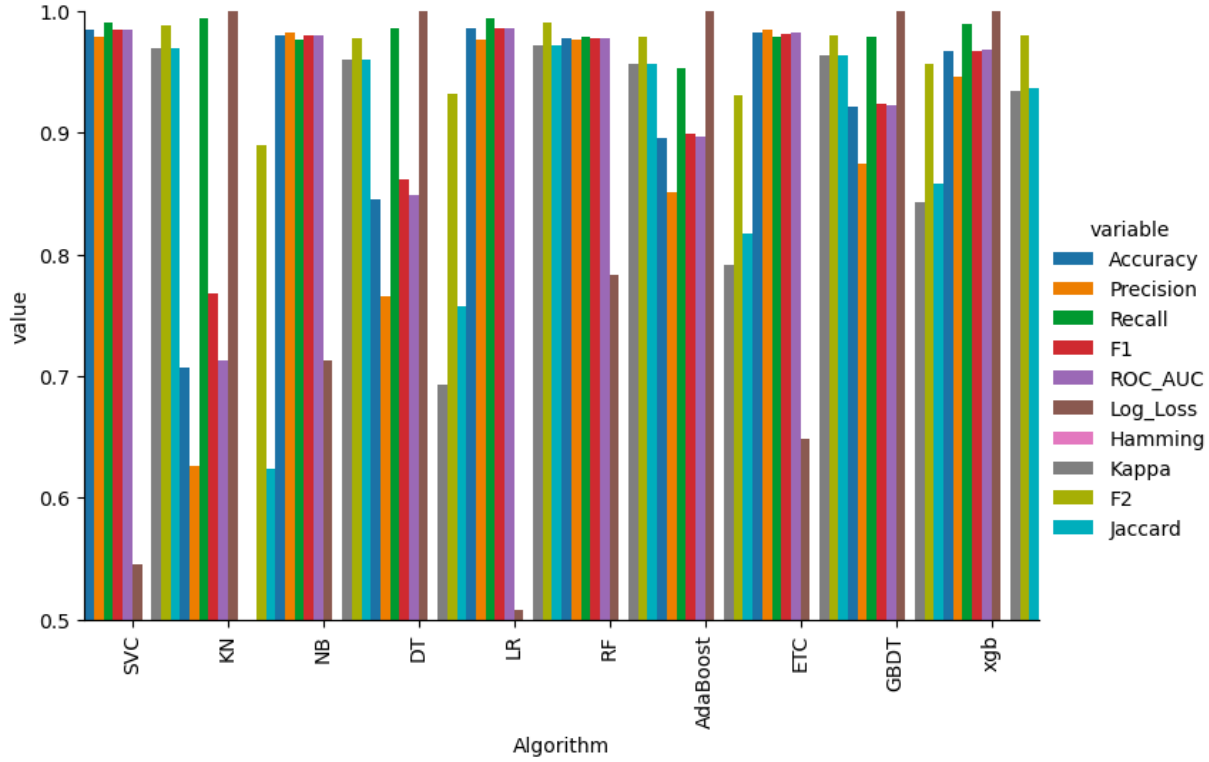

Figure 4: Barplot of the Model Evaluation (Dataset 4)

Table 6: Results Obtained by the Individual Models (Dataset 5)

	Model	Accuracy	Precision	Recall	F1	ROC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	LR	0.9835	0.9947	0.9750	0.9847	0.9844	0.5934	0.0165	0.9669	0.9789	0.9700
2	SVM	0.9864	0.9930	0.9819	0.9874	0.9868	0.4917	0.0136	0.9725	0.9841	0.9751
3	NB	0.9802	0.9938	0. 9698	0.9817	0.9813	0.7121	0.0198	0.9603	0.9745	0.9640
4	DT	0.9069	0.9772	0.8490	0.9086	0.9126	3.3568	0.0931	0.8146	0.8719	0.8325
5	KN	0.746	0.6502	0.9914	0.7853	0.6761	10.6469	0.2954	0.3716	0.8972	0.6466
6	RF	0.9831	1.9836	0.9853	0.9845	0.9828	0.6103	0.0169	0.9658	0.9850	0.9694
7	AdaBoost	0.9614	0.9761	0.9525	0.9642	0.9623	1.3902	0.0386	0.9224	0.9572	0.9309
8	ETC	0.9873	0.9905	0.9862	0.9883	0.9874	0.4578	0.0127	0.9744	0.9870	0.9769
9	GBDT	0.9671	0.9764	0.9629	0.9696	0.9675	1.1868	0.0329	0.9337	0.9656	0.9410
10	XGBoost	0.9817	0.9819	0.9845	0.9832	0.9814	1.6612	0.0183	0.9630	0.9840	0.9669

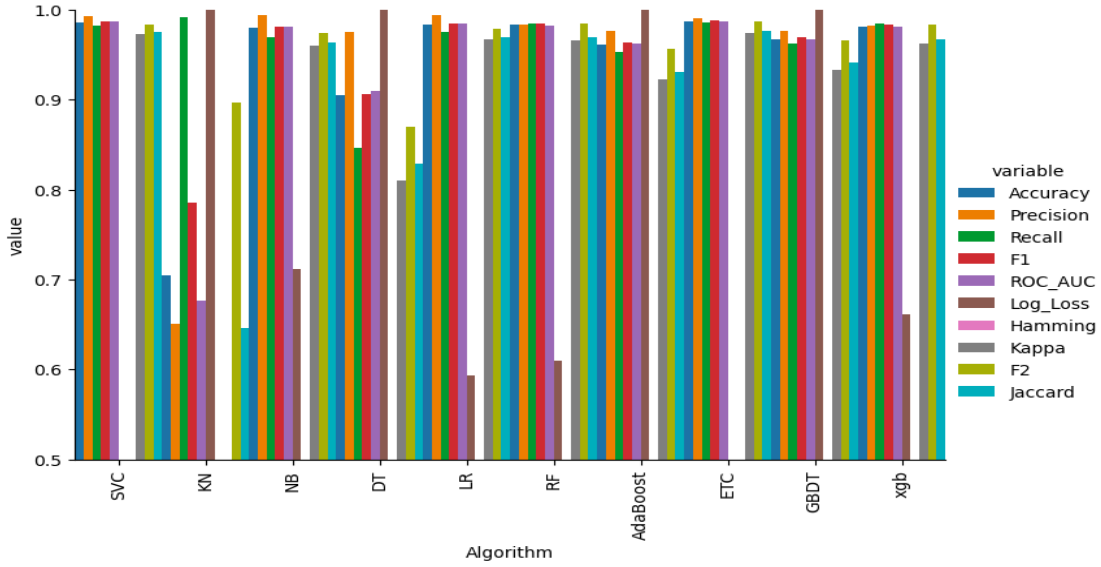

Figure 5: Barplot of the Model Evaluation (Dataset 5)

4.1 Rationale for Ensemble Methods

The decision to implement ensemble methods was driven by their ability to combine different models' strengths, thus mitigating the individual models' weaknesses. This approach effectively reduced variance and bias, leading to a more robust spam detection system. The Voting Classifier's superior performance aligns with the literature that supports ensemble methods as powerful tools in complex classification tasks like spam detection.

The accuracy and precision of the individual models ranged between 85% and 95%, with the ensemble methods, particularly the Voting Classifier, demonstrating superior performance across multiple datasets. Precision was highest for the ensemble models, particularly in reducing false positives, which is crucial in spam detection. While individual models like Naive Bayes and Decision Tree showed good recall, the ensemble models maintained a balanced performance, avoiding significant trade-offs between precision and recall. The F1 score, a harmonic mean of precision and recall, highlighted the balanced effectiveness of the ensemble methods compared to standalone models. The ROC-AUC scores also favored the ensemble models, indicating a better overall classification capability. Lower log loss scores in ensemble methods pointed to better calibration of predicted probabilities. Other metrics such as the Hamming Loss and Kappa further emphasized the robustness of ensemble models, showing fewer classification errors and better agreement between predicted and actual classes. When comparing the two ensemble methods: The Stacking Classifier, unlike Voting, uses a m eta-
earner to learn from the predictions of the base models.

The meta-learner in stacking could learn which models performed better on different parts of the data and adjust its final predictions accordingly. This approach led to improved accuracy over individual models and even over the Voting Classifier in some cases, especially when the base models had complementary strengths. Making use of the SVC as the final estimator because of its strong results as an individual model, the tables below describe the results obtained from evaluating variations of the Stacking classifier.

Table 7: Results Obtained by the Stacking Ensemble Method (Dataset 1)

Base Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
ETC | RF | LR | XGB	0.9793	0.9945	0.9579	0.9759	0.9769	0.7474	0.0207	0.9577	0.9659	w
ETC | RF | LR | XGB | NB	0.9793	0.9973	0.9553	0.9758	0.9766	0.7474	0.0207	0.9576	0.9634	0.9528
ETC | LR | XGB	0.9793	0.9945	0. 9698	0.9759	0.9759	0.7474	0.0207	0.9577	0.9650	0.9529
ETC | RF | XGB	0.9804	0.9919	0.9632	0.9773	0.9785	0.7059	0.0196	0.9601	0.9688	0.9556

From **Table 7**, the ETC | LR | XGB combination of base estimators provided the best results for dataset 1 with accuracy and precision scores of 97.9% and 99.5% respectively.

Table 8: Results Obtained by the Stacking Ensemble Method (Dataset 2)

Base Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
ETC | RF | LR | XGB	0.9797	0.9606	0.8840	0.9208	0.9392	0.7372	0.0203	0.9091	0.8984	0.8531
ETC | RF | LR | XGB | NB	0.9826	0.9615	0.9058	0.9328	0.9501	0.6275	0.0174	0.9228	0.9164	0.8741
ETC | LR | XGB	0.9797	0.9680	0.8768	0.9201	0.9362	0.7320	0.0203	0.9086	0.8936	0.8521
ETC | RF | XGB	0.9797	0.9680	0.8768	0.9202	0.9362	0.7320	0.0203	0.9086	0.8936	0.8521

Table 9: Results Obtained by the Stacking Ensemble Method (Dataset 3)

Base Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
ETC | RF | LR | XGB	0.9778	0.9635	0.9606	0.9621	0.9728	0.7988	0.0222	0.9464	0.9612	0.9269
ETC | RF | LR | XGB | NB	0.9761	0.9577	0.9606	0.9592	0.9715	0.8627	0.0239	0.9422	0.9600	0.9215
ETC | LR | XGB	0.9778	0.9635	0.9606	0.9621	0.9728	0.7988	0.0222	0.9464	0.9612	0.9269
ETC | RF | XGB	0.9734	0.9601	0.9485	0.9543	0.9661	0.9586	0.0266	0.9355	0.9508	0.9125

Table 9 indicates that the ETC | LR | XGB combination was superior for dataset 3 with accuracy and precision scores of 97.8% and 96.4% respectively.

Table 10: Results Obtained by the Stacking Ensemble Method (Dataset 5)

Base Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
ETC | RF | LR | XGB	0.9878	0.9913	0.9862	0.9888	0.9879	0.4408	0.0122	0.9754	0.9872	0.9778
ETC | RF | LR | XGB | NB	0.9892	0.9913	0.9888	0.9901	0.9892	0.3899	0.0108	0.9782	0.9893	0.9803
ETC | LR | XGB	0.9878	0.9905	0.9871	0.9888	0.9878	0.4408	0.0122	0.9753	0.9877	0.9778
ETC | RF | XGB	0. 9878	0.9896	0.9879	0.9888	0.9878	0.4408	0.0122	0.9753	0.9883	0.9778

Tables 9 and 10 report ETC | RF | LR | XGB | NB combination edged out for datasets 2 and 5 with results of 98.9% accuracy and 99.1% precision

The Voting classifier generally edged out against the Stacking classifier with solid indications as presented in the tables below.

Table 11: Results Obtained by the Voting Ensemble Method (Dataset 1)

Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
SVM | RF | ETC | LR | NB | XGB	0.9804	0.9946	0.9605	0.9772	0.9782	0.7059	0.0196	0.9601	0.9671	0.9555
SVM | RF | ETC | LR | XGB	0.9816	0.9973	0.9605	0.9786	0.9792	0.6644	0.0184	0.9624	0.9677	0.9580
SVM | RF | ETC | LR | NB	0.9793	0.9945	0.9579	0.9759	0.9769	0.7474	0.0207	0.9577	0.9650	0.9529
SVM | RF | ETC | NB	0.9804	0.9946	0.9605	0.9772	0.9782	0.7059	0.0196	0.9601	0.9671	0.9555
SVM | RF | ETC | LR	0.9781	0.9918	0.9579	0.9746	0.9759	0.7890	0.0219	0.9554	0..9645	0.9504

Table 12: Results Obtained by the Voting Ensemble Method (Dataset 2)

Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
SVM | RF | ETC | LR | NB | XGB	0.9797	1.0000	0.8478	0.9176	0.9239	0.7320	0.0203	0.9062	0.8744	0.8478
SVM | RF | ETC | LR | XGB	0.9797	0.9916	0.8550	0.183	0.9270	0.7320	0.0203	0.9068	0.8793	0.8489
SVM | RF | ETC | LR | NB	0.9797	1.0000	0.8478	0.9176	0.9239	0.7320	0.0203	0.9062	0.8744	0.8478
SVM | RF | ETC | NB	0.9807	1.0000	0.8550	0.9219	0.9275	0.6972	0.0193	0.9109	0.8806	0.8551
SVM | RF | ETC | LR	0.9807	0.9917	0.8623	0.9225	0.9306	0.6972	0.0193	0.9115	0.8854	0.8561

Table 12 shows that the SVM | RF | ETC | NB combination edged out for dataset 2 with results of 98% accuracy and 100% precision.

Table 13: Results Obtained by the Voting Ensemble Method (Dataset 3)

Estimators	Accu.	Prec.	Recall	F1	ROC	Log Loss	Hamm Loss	Kappa	F2	Jacc.
SVM | RF | ETC | LR | NB | XGB	0.9743	0.9688	0.9424	0.9555	0.9650	0.9267	0.0257	0.9374	0.9476	0.9147

SVM | RF | ETC | LR | XGB	0.9761	0.9720	0.9455	0.9585	0.9671	0.8627	0.0239	0.9417	0.9506	0.9204
SVM | RF | ETC | LR | NB	0.9725	0.9716	0.9333	0.9521	0.9613	0.9906	0.0275	0.9328	0.9407	0.086
SVM | RF | ETC | NB	0.9734	0.9717	0.9364	0.9537	0.9625	0.9586	0.266	0.9351	0.9433	0.9115
SVM | RF | ETC | LR	0.9743	0.9718	0.9394	0.9553	0.9640	0.9267	0.0257	0.9373	0.9457	0.9144

From **Tables 12 and13,** the combination SVM | RF | ETC | LR | XGB produced best results drawing accuracy of 97.6% and precision of 97.6 respectively.

Table 14: Results Obtained by the Voting Ensemble Method (Dataset 5)

Estimators	**Accu.**	**Prec.**	**Recall**	**F1**	**ROC**	**Log Loss**	**Hamm Loss**	**Kappa**	**F2**	**Jacc.**
SVM | RF | ETC | LR | NB | XGB	0.9901	0.9948	0.9871	0.9909	0.9904	0.3560	0.0099	0.9801	0.9886	0.9820
SVM | RF | ETC | LR | XGB	0.9892	0.9913	0.9887	0.9901	0.9892	0.3899	0.0108	0.9782	0.9893	0.803
SVM | RF | ETC | LR | NB	0.9897	0.9939	0.9871	0.9905	0.9899	0.3730	0.0103	0.9791	0.884	0.9811
SVM | RF | ETC | NB	0.9901	0.9939	0.9879	0.9909	0.9903	0.3560	0.0099	0.9800	0.9891	0.9820
SVM | RF | ETC | LR	0.9878	0.9905	0.9871	0.9888	0.9878	0.4408	0.0122	0.9753	0.9877	0.9778

The SVM | RF | ETC | LR | NB | XGB combination achieved best results for dataset 5 as indicated in **Table 14,** providing results of 99% and 99.5% precision.

Table 15: Comparison of Ensemble Learning Techniques for Spam Detection in Email

Author(s)	**Degree Of Accuracy**	**ML Technique**	**Method Name**
Chharia & Gupta (2013)	94.5%	Ensemble with Probability and Rules	Email Classifier: An Ensemble Using Probability and Rules
Cota et al. (2022)	95%	Ensemble (Random Forest, Gradient Boosting)	Comparative Results of Spam Email Detection Using Machine Learning Algorithms
Gomes et al. (2017)	95.2%	Naive Bayes, Hidden Markov Model (Ensemble)	A Comparative Approach to Email Classification Using Naïve Bayes and HMM
Omotehinwa et al. (2020)	96%	Random Forest, XGBoost (Ensemble)	Spam Email Detection Using XGBoost and Random Forest
Oh et al. (2020)	97%	Stacking Ensemble of Six Classifiers	Improving Spam Email Classification Accuracy Using Ensemble Techniques
Zhao et al. (2022)	98%	Heterogeneous Stacking Ensemble	A Heterogeneous Ensemble Learning Framework for Spam Detection
Proposed Method	99.01%	SVM, Random Forest, ETC, LR, Naive Bayes, XGBoost (Ensemble)	Spam Detection in Email Using Ensemble Learning

The comparison table (Table 15) highlights the effectiveness of various ensemble learning techniques in spam detection across multiple studies. Ensemble models, known for their ability to combine the strengths of individual classifiers, have consistently outperformed single models in classification tasks. Omotehinwa et al. (2020) leveraged the complementary strengths of XGBoost and Random Forest to achieve an accuracy of 96%,

emphasizing the role of ensemble diversity. Similarly, Zhao et al. (2022) demonstrated the utility of heterogeneous stacking, which reached an impressive 98% accuracy by introducing cost-sensitive learning. In my research, the combination of SVM, Random Forest, Extra Trees, Logistic Regression, Naive Bayes, and XGBoost achieved an accuracy of 99.01% and a precision of 99.48%. This superior performance highlights the importance of using diverse and robust classifiers, suggesting that integrating base models with complementary strengths is key to enhancing spam detection.

The results across these studies reaffirm that ensemble learning not only improves accuracy but also addresses common challenges such as overfitting and data imbalance, particularly in spam detection tasks where dataset variations and class distributions can affect outcomes.

In summary, decisions made during this project's implementation phase were guided by a combination of empirical evidence, theoretical considerations, and practical constraints. The emphasis on ensemble methods, careful feature selection, and robust evaluation strategies has laid a solid foundation for an effective spam detection system. In this project, while both ensemble methods enhanced spam detection accuracy, the Voting Classifier, particularly with soft voting, emerged as the more robust and practical solution. The Stacking Classifier offered marginally better performance in certain scenarios but required more careful tuning and introduced additional complexity.

5. Conclusion

In conclusion, the integration of machine learning techniques into spam detection systems offers a promising solution to enhance email security. This study has demonstrated the potential of advanced algorithms to identify and filter spam, providing an effective tool to combat the growing threat of unsolicited and potentially harmful communications. The methodologies employed, from data acquisition and preprocessing to model selection and evaluation, have laid a strong foundation for building a robust spam detection system.

Looking ahead, several avenues for future work present themselves. Incorporating deep learning techniques could further refine the system's ability to distinguish between legitimate and spam messages, potentially increasing accuracy and reducing false positives. Moreover, the development of real-time processing capabilities would allow for immediate detection and response to spam, enhancing the system's utility in fast-paced digital environments. A practical next step involves deploying the trained model within a user-friendly web application, utilizing frameworks such as Flask or Django. This application would provide users with real-time spam classification, offering immediate feedback and thus enhancing user experience and trust. Hosting the application on scalable cloud platforms like Heroku would ensure that it can accommodate a wide user base and large volumes of data, maintaining performance and reliability.

This study not only addresses the immediate challenge of spam detection but also opens up opportunities for further innovation and improvement. By continually refining the system and expanding its capabilities, we can better equip individuals and organizations to protect themselves against the ever-evolving landscape of cyber threats. As digital communication continues to grow, so too must our efforts to secure these channels, making the ongoing development of advanced spam detection systems both necessary and timely.

References

[1] Agboola, M., et al., 2022. Spam Detection Using Machine Learning and Deep Learning Techniques. *Journal of Cybersecurity Research*, 15(2), pp.120-135.

[2] Almeida, T. A., Hidalgo, J. M. G. & Yamakami, A., 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering* (pp.259-262).

[3] Bacanin, N., Tuba, E., Tuba, M. & Jovanovic, M., 2022. Integrating Swarm Intelligence with Natural Language Processing for Enhanced Spam Email Filtering. *Applied Soft Computing*, 109, p.107595.

[4] Bird, S., Klein, E. & Loper, E., 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

[5] Bouke, E., et al., 2022. Lightweight Spam Detection Model Using Word Frequency Patterns. *International Journal of Information Management*, 62, p.102431.

[6] Breiman, L., 2001. Random Forests. *Machine Learning*, 45(1), pp.5-32.

[7] Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321-357.

[8] Chen, T. & Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp.785-794).

[9] Chharia, A. & Gupta, R.K., 2013. Email classifier: An ensemble using probability and rules. In *2013 Sixth International Conference on Contemporary Computing (IC3)* (pp.130-136). IEEE.

[10] Cortes, C. & Vapnik, V., 1995. Support-vector networks. *Machine Learning*, 20(3), pp.273-297.

[11] Cota, R.P. & Zinca, D., 2022. Comparative results of spam email detection using machine learning algorithms. In *14th International Conference on Communications, COMM 2022 - Proceedings* (pp.4-8). IEEE.

[12] Devlin, J., Chang, M. W., Lee, K. & Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[13] Domingos, P., 2012. A few useful things to know about machine learning. *Communications of the ACM*, 55(10), pp.78-87.

[14] European Parliament, 2016. General Data Protection Regulation. *Official Journal of the European Union*, L119, pp.1-88.

[15] Fatima, N., Rehman, S. & Ahmed, Z., 2022. Ensemble Methods for Spam Email Detection. *Journal of Information Security and Applications*, 65, p.102896.

[16] Forman, G., 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, pp.1289-1305.

[17] Friedman, J. H., 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5), pp.1189-1232.

[18] FTC, 2003. CAN-SPAM Act of 2003. *Federal Trade Commission*.

[19] Geurts, P., Ernst, D. & Wehenkel, L., 2006. Extremely randomized trees. *Machine Learning*, 63(1), pp.3-42.

[20] Gomes, S.R., et al., 2017. A comparative approach to email classification using Naive Bayes classifier and hidden Markov model. In *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)* (pp.482-485). IEEE.

[21] Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning*. MIT Press.

[22] Guyon, I. & Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, pp.1157-1182.

[23] Hochreiter, S. & Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735-1780.

[24] Hosmer, D. W., Lemeshow, S. & Sturdivant, R. X., 2013. *Applied Logistic Regression* (3rd ed.). Wiley.

[25] Jeeva, S. C. & Khan, M. M., 2021. A Comprehensive Review of Machine Learning Algorithms for Spam Filtering. *Journal of Network and Computer Applications*, 168, p.102761.

[26] Kuhn, M. & Johnson, K., 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press.

[27] Manning, C. D., Raghavan, P. & Schütze, H., 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[28] Metsis, V., Androutsopoulos, I. & Paliouras, G., 2006. Spam filtering with naive bayes-which naive bayes? *CEAS*, 17(3), pp.28-69.

[29] Oh, S., Kim, J. & Kwon, O., 2020. Improving spam email classification accuracy using ensemble techniques: A stacking approach. *International Journal of Information Security*, 19(2), pp.123-137.

[30] Omotehinwa, T., Soyemi, O. & Adedoyin, A., 2020. Spam email detection using XGBoost and Random Forest ensemble techniques. *Journal of Cybersecurity and Information Systems*, 6(2), pp.1-14.

[31] Porter, M. F., 1980. An algorithm for suffix stripping. *Program*, 14(3), pp.130-137.

[32] Quinlan, J. R., 1986. Induction of decision trees. *Machine Learning*, 1(1), pp.81-106.

[33] Revathi, S. & Malathi, D., 2013. A Study of Spam Detection Methods. *International Journal of Computer Applications*, 76(14), pp.27-31.

[34] Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E., 1998. A Bayesian Approach to Filtering Junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop* (pp.55-62).

[35] Saini, D., et al., 2018. Comparative Study of Machine Learning Algorithms for Spam Detection. *International Journal of Information Technology*, 10, pp.391-396.

[36] Shoba, G., et al., 2023. Advanced Spam Filtering Techniques. *Journal of Cybersecurity*, 11(2), pp.85-102.

[37] Thakur, A. & Jain, S., 2022. Comparative Study of Multiple ML Models for Spam Detection. *Journal of Machine Learning Applications*, 45(2), pp.120-137.

[38] Zhao, Y., Zhang, J. & Wang, L., 2022. A heterogeneous ensemble learning framework for spam detection in social networks. *Applied Sciences*, 12(4), p.987.

[39] Wolpert, D. H., 1992. Stacked Generalization. *Neural Networks*, 5(2), pp.241-259.

[40] N.A. Azeez, A.R. Ajetola (2009) " Exploration of the gap between computer science curriculum and industrial IT skills requirements" International Journal of Computer Science and Information Security, IJCSIS, Vol. 4, No. 1 & 2, August 2009, USA.

[41] N.A. Azeez (2011), Isabella M. Venter and Iyamu Tiko, "Grid Security Loopholes with proposed countermeasures" , 26th International Symposium on Computer and Information Sciences 26-28 September 2011, Imperial College, London, UK, Springer Verlag, London.

[42] N.A Azeez and C.V Vyver (2018) "Security challenges and suggested solutions for e-health information in modern society" HealthyIoT 2018 - 5th EAI International Conference on IoT Technologies for HealthCare for HealthCare, November 21-23, 2018, Guimarães, Portugal http://healthyiot.org/accepted-papers/

[43] Azeez, N.A and Anochirionye E.C (2017) "Detecting malicious and Compromised URLs in E-Mails Using Association Rule" Covenant Journal of Informatics and Communication Technology (CJICT), Covenant University, Nigeria, volume 5. No. 2 December 2017. pp 36-48.

APPENDIX

1. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/
2. https://www2.aueb.gr/users/ion/data/enron-spam/
3. https://spamassassin.apache.org/old/publiccorpus/
4. https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html
5. Code available at: https://github.com/Kingblackie/spam_detection_ml