applied sciences

MDPI

Article

Hyperparameter Optimization of Ensemble Models for Spam Email Detection

Temidayo Oluwatosin Omotehinwa [1,*] and David Opeoluwa Oyewola [2]

[1] Department of Mathematics and Computer Science, Federal University of Health Sciences, Otukpo P.M.B. 145, Nigeria
[2] Department of Mathematics and Statistics, Federal University Kashere, Gombe P.M.B. 0182, Nigeria
* Correspondence: temidayo.omotehinwa@fuhso.edu.ng

Abstract: Unsolicited emails, popularly referred to as spam, have remained one of the biggest threats to cybersecurity globally. More than half of the emails sent in 2021 were spam, resulting in huge financial losses. The tenacity and perpetual presence of the adversary, the spammer, has necessitated the need for improved efforts at filtering spam. This study, therefore, developed baseline models of random forest and extreme gradient boost (XGBoost) ensemble algorithms for the detection and classification of spam emails using the Enron1 dataset. The developed ensemble models were then optimized using the grid-search cross-validation technique to search the hyperparameter space for optimal hyperparameter values. The performance of the baseline (un-tuned) and the tuned models of both algorithms were evaluated and compared. The impact of hyperparameter tuning on both models was also examined. The findings of the experimental study revealed that the hyperparameter tuning improved the performance of both models when compared with the baseline models. The tuned RF and XGBoost models achieved an accuracy of 97.78% and 98.09%, a sensitivity of 98.44% and 98.84%, and an F1 score of 97.85% and 98.16%, respectively. The XGBoost model outperformed the random forest model. The developed XGBoost model is effective and efficient for spam email detection.

Keywords: spam detection; spam emails; random forest; XGBoost; ensemble; hyperparameter

Citation: Omotehinwa, T.O.; Oyewola, D.O. Hyperparameter Optimization of Ensemble Models for Spam Email Detection. *Appl. Sci.* **2023**, *13*, 1971. https://doi.org/10.3390/app13031971

Academic Editors: Jae-Hoon Kim and Kichun Lee

Received: 22 November 2022
Revised: 23 January 2023
Accepted: 28 January 2023
Published: 3 February 2023

1. Introduction

Unsolicited emails, popularly referred to as spam, have remained one of the biggest threats to cybersecurity globally. Between October 2020 and September 2021, a total of 336.41 billion emails were sent globally, and about 84% (more than half) of these emails were spam [1]. The huge financial loss resulting from email fraud is quite enormous and increasing. According to the FBI center for crime complaint reports [2], in 2021 about USD2.4 billion was lost as a result of scams associated with business and email account compromises. In the same year, the bureau received 19,954 scam email complaints. The IC3 data also showed that 3729 ransomware incidents were reported with an associated financial loss of over USD49 million. According to the spam and phishing report by Kaspersky on Securelist [3], between February and June 2022, 1.8 million 419 scam emails were detected. These statistics imply that spammers are relentless. Researchers have continued to propose different techniques to combat the spam menace [4–10]. However, the tenacity and perpetual presence of the adversary, the spammer, has necessitated the need for improved efforts at filtering spam. A spam-filtering model with improved accuracy will help in the fight against spam-based fraud. Many current spam-email-detection techniques rely on a single model, which can be prone to errors and overfitting [10–14]. Ensemble models, which combine the predictions of multiple models, have the potential to improve the accuracy and robustness of spam detection. While ensemble models have been widely used in other areas of machine learning, they have not been widely applied to spam email detection. Hyperparameters, such as the number of decision trees in a random forest or the regularization parameter in an extreme gradient boost algorithm, can greatly affect

the performance of a model. However, finding the optimal hyperparameters are often ignored because it is a time-consuming and computationally expensive task. Therefore, this study is aimed at the hyperparameter optimization of the random forest (RF) and extreme gradient boosting (XGBoost) ensemble algorithms. This is in a bid to enhance the predictive accuracy of the two ensemble models and to determine the best-performing model, robust enough for efficient spam email detection. Ensemble algorithms rely on a combination of predictions from two or more base models to obtain an improved prediction performance on a dataset [15]. In this study:

Spam-email-detection models based on the random forest and XGBoost machine-learning algorithms were developed.

The performances of the ensemble models were optimized through hyperparameter tuning.

The performances of the ensemble models were evaluated and compared before and after hyperparameter tuning.

The convergence time of the models were also established.

The other sections of this study are presented thus: In the second section, a brief highlight of related research on spam email classification and detection was presented. The third section described in detail the dataset and preprocessing techniques, methods, and performance evaluation metrics. The results of the experiments are presented in the fourth section. Finally, a conclusion was drawn with a perspective for further studies in the last section.

2. Related Work

In recent times, the machine-learning approach to spam email detection has continued to increase in addition to other spam-filtering techniques such as list-based (Whitelist, Greylist, and Real-time blacklist) and word-based (Heuristic filters, word-based: DNS lookup), challenge-response, and so on.

Several studies have applied machine and deep learning with the intent of improving the performance of spam filters for classifying emails. In this study [16], a technique for detecting spam emails was introduced. This method utilizes a decision-tree-mining approach and focuses on the email header rather than the entire content of the email. An incremental learning algorithm based on the C4.5 decision tree algorithm is also used to improve the technique's ability to adapt to changes in the structure of spam. The model achieved a precision rate of 96%.

This study [17] distinguished between two types of spam emails: complete spam, which is spam that is considered spam by all users, and semi-spam, which is spam that is considered spam by some users but not by others. They developed a method for identifying spam that combines Bayesian filtering for complete spam with a crowdsourcing mechanism for identifying semi-spam. The crowdsourcing aspect of the method involves soliciting reports of spam from contacts or credible users with similar interests. The authors achieved an accuracy rate of 95.1% and believed that their model's performance could be improved by applying the concept of virtual credits to stimulate self-centered nodes to report spam and by enhancing connectivity and throughput.

This study [18] presented an email-filtering approach that utilizes semantic methods, specifically the WordNet ontology, to classify emails as either spam or non-spam. The approach aims to reduce the high dimensionality of email text by applying semantic methods and similarity measures, and then further reduces the number of features through the use of feature-selection techniques such as Principal Component Analysis (PCA) and Correlation Feature Selection (CFS). The proposed approach was tested on the Enron dataset and was found to have a high accuracy rate of above 90% when using the logistic regression classification algorithm, with a reduction in the number of features by over 90%. The proposed method was also found to have a higher accuracy rate and faster performance compared to other related approaches.

The study by [19] combined Particle Swarm Optimization (PSO) with naïve Bayes (NB) to create a new model for classifying emails as spam or non-spam. The model was trained using 1000 emails from the Ling dataset, and features were selected from the bag of words using Correlation-based Feature Selection (CFS). The performance of the new NB and PSO model was compared to that of the ordinary NB model, and it was found that the NB and PSO model had a greater performance for all evaluation metrics (precision, recall, F-measure, and accuracy), with values above 94%, while the ordinary NB model had values below 89% for all metrics.

This study [20] carried out a systematic review of several machine-learning applications and their performances in spam detection. The current trends and open research areas in spam filtering were discussed extensively. The strength and weaknesses of the algorithms, such as the Bayesian classification, random forest, ANNs, SVMs, deep learning, Artificial Immune Systems, and Rough sets, amongst others, were compared. They verified that significant progress has been made and more is required in the struggle to end spamming. Finally, they recommended machine-, deep-, and deep-adversarial-learning algorithms as possible future technology for the effective management of spam emails.

This study [21] investigated various classification-based data-mining techniques such as the J48 decision tree, random forest, naïve Bayes, and SMO for identifying spam emails and analyzing their performance on a spam dataset. WEKA was used to train and explore the performance of the different classifiers and identify the best-performing one for classifying email spam. The classifiers' performance was evaluated based on the standard evaluation metrics used for machine learning models and the training time. random forest outperformed the other models for all metrics evaluated and achieved a weighted F-measure of 95.50%. naïve Bayes also did well in terms of execution time.

This study [22] evaluated the performance of five classifiers: logistic regression, decision tree, naïve Bayes (NB), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). They used the WEKA tool to train and test the algorithms on the Spambase dataset from the UCI machine-learning repository. The decision tree and KNN algorithms had the best performance, with an accuracy rate of 99% for all metrics. However, KNN took longer to converge than the other algorithms.

This study [14] presented a new approach to detecting spam emails that combines the artificial bee colony algorithm with a logistic regression classification model. The proposed method was tested on three publicly available datasets, namely, Enron, CSDMC2010, and TurkishEmail. The model achieved a higher classification accuracy (98.91%) than the other methods considered. The study reported that the proposed method is effective at handling high-dimensional data and performs better than other machine-learning methods, including support vector machine, logistic regression, and naïve Bayes, as well as state-of-the-art techniques from previous studies.

Towards an accurate detection of spam in mobile message communication, this study [23] developed a machine-learning-based approach for detecting spam messages in mobile device communication. Three classifiers–logistic regression (LR), K-Nearest Neighbor (K-NN), and decision tree (DT)–were applied to the SMS spam collection dataset and evaluated on their ability to classify ham and spam messages. The dataset was split into training and testing sets. The results showed that LR had the highest classification performance, with an accuracy of 99%, and outperformed K-NN and DT with 95% and 98% accuracy, respectively.

This study [13] integrated KNN with five bio-inspired algorithms to optimize the spam email detection of the KNN model. The bio-inspired algorithms are Grey wolf optimization, Firefly optimization, Chicken swarm optimization, Grasshopper optimization, and Whale optimization. In the study, alongside the evaluation of the performance of each of the algorithms with KNN, the performance of distance measures such as Manhattan, Euclidean, and Chebychev was measured. The findings revealed that the Whale optimization algorithm integrated with the KNN model is quite promising for most of the evaluation metrics.

In order to improve spam detection, this study [12] examines the use of machine-learning techniques on email text data. Six different algorithms (naïve Bayes, K-Nearest Neighbors, SVM, logistic regression, decision tree, and random forest) are applied to classify emails as spam or non-spam using natural language processing. These algorithms are trained and tested on a dataset, and the results show that logistic regression and naïve Bayes have the highest accuracy rates of up to 99%. The authors propose that the findings of this study could be used to create a more effective spam detection classifier through the combination of different algorithms or filtering methods.

This study [4] developed a Convolutional Neural Network (CNN) model for image-based spam email detection. The CNN model was composed of one input layer and two convolution layers with 32 and 64 kernels in the hidden layer. The CNN model was trained and tested on the Dredze and Image Spam Hunter (ISH) dataset. The model achieved an 88% F1 score on the validation set and a 97% F1 score on the dataset's test samples.

In this study [24], a method for detecting spam emails was proposed that combines the naïve Bayes algorithm with the Markov Random Field. The naïve bayes algorithm was used for the probabilistic classification of emails in the Eron1 dataset, and the Hidden Markov Field was utilized to model the statistical behavior of spam. Feature vectors for incoming messages were created by breaking down the emails into features and weighting them to consider inter-word dependence in the learning algorithms. The hybrid approach was able to address the weaknesses of both individual algorithms and demonstrated high efficiency in terms of accuracy and time consumption for spam detection. The hybrid model was compared to the NB and MRF models and was found to have more accurate predictions and faster convergence, with an execution time of 1500 ms compared to 2750 ms and 7500 ms for the NB and MRF models, respectively.

In this study [25], a novel method for identifying and blocking spam messages sent via Short Message Service (SMS) using a discrete Hidden Markov Model (HMM) that incorporates weighted features was presented. Traditional methods such as Rule-Based Systems (RBS) and Content-Based Filtering (CBF) have shortcomings and newer, more complex hybrid models struggle with performance and speed. A previous study found that the HMM method proposed in this paper is comparable to deep-learning techniques. To enhance the performance, a new approach of weighting and labelling words in SMS messages to format the observation sequence within the HMM is presented. The experiments on open datasets from the University of California, Irvine, shows the new HMM method with weighted features to have improved accuracy, faster training, and filtering speed. Results show that it outperforms the LSTM and is close to CNN in terms of classification accuracy. This method was also evaluated on a Chinese SMS dataset which further reinforced its accuracy and speed.

In this study [26], the effectiveness of 45 different algorithms was tested on a dataset of 1017 emails collected from various Gmail and Hotmail accounts. The goal was to classify the emails as spam or non-spam using the Weka program. The naïve Bayes multinomial and naïve Bayes multinomial updateable algorithms had the highest classification success rate at 94.7886%, followed by the random forest trees algorithm at 93.6087%, Meta. Multi-Class Classifier and Functions SGD at 92.4287%, Functions SMO at 91.7404%, Meta Random Committee at 91.0521%, Bayes naïve Bayes at 90.3638%, and Bayes naïve Bayes updateable at 90.3638%.

The Genetic Decision Tree Processing with Natural Language Processing (GDTPNLP) was proposed by [10] for detecting text-enabled and voice-enabled spam emails. The GDTPNLP, after genetic analysis of incoming emails, assigns a confidence threshold to the emails and compares the value with the trained set. A higher confidence threshold indicates spam, and a lower threshold indicates legitimate emails. The hybrid model's performance was compared to other learning algorithms such as NB, SVM, Nearest Neighbour, and J48. The GDTPNLP achieved an accuracy of 98.6% while NB, SVM, NN, and J48 achieved 80%, 90%, 89%, and 89.7% accuracies, respectively. This study is quite distinct as it considered voice-enabled email. The approach is promising.

This study [27] presented a new method called "category-learning attention" that aims to enhance the performance of machine translation. Unlike the standard approach, which only weighs words within the same text, the proposed method utilizes category-level features, specially designed for short texts, by applying a category differentiation matrix to identify words that are heavily distributed within the same category. This method was further developed into two different variations: "category-learning scaled-dot-product attention" and "category-learning multi-head attention (CL-MHA) mechanisms." These mechanisms were integrated into a bidirectional gate recurrent unit (Bi-GRU) model, which was evaluated using the SMS spam collection dataset from the University of California, Irvine. The results reveal that the proposed CL-MHA mechanism leads to a significant improvement in the performance of the Bi-GRU model for short text filtering, achieving an accuracy of 99.35%, outperforming previous machine-learning models. The method was also tested and verified on three additional datasets, including a Chinese SMS spam dataset, a benchmark movie review dataset, and a benchmark customer review dataset, and achieved an accuracy of 99.46% when applied to the Chinese SMS spam dataset.

3. Methodology

3.1. Dataset

The Enron dataset was used in this research because it is the only substantial collection of an actual email that is public and also because of its high level of usage among researchers. The Enron dataset is made up of 6 main directories, each directory has several subdirectories, each containing emails as a single text file [28]. In this study, the Enron1 dataset was used. These emails were converted to a single CSV file by Marcel Wiechmann [29]. The CSV file contained about 33,000 emails. However, during conversion, some of the email messages were not correctly aligned with their labels. The non-aligning messages were removed alongside the orphaned labels through Microsoft Excel. On completion of the removal, the CSV file contained a total of 32,860 emails. Of the total emails, 16,026 (49%) are legitimate emails (ham) and 16,834 (51%) are spam. The CSV file contained five columns labeled message ID, subject, spam/ham, and date. The subject and date columns were not used in this study. The needed columns were readjusted as serial numbers. Column two contains the class label of each of the emails and column three contains the text of each email.

3.2. Dataset Cleaning

To improve the quality of classification, it is important to get rid of unwanted characters or features that constitute noise from the data. The cleaning activities and functions used are presented in Table 1.

Table 1. Collection of functions used in data cleaning.

Steps	Data Processing Activities	Functions
1	Removal of non-ASCII codes and emoticons	email_text = gsub("[^\x01-\x7F]", "", email_text)
2	Removal of HTML tags	email_text = gsub("<.*/>","",email_text)
3	Removal of numbers	email_text = removeNumbers(email_text)
4	Remove all the URLs	email_text = gsub(" ?(f|ht)tp(s?)://(.*)[.][a-z]+", "", email_text)
5	Remove extra white spaces	email_text = tm_map(email_text, stripWhitespace)
6	Removal of punctuations	email_text = tm_map(email_text,removePunctuation)
7	Removal of stop words	email_text = tm_map(corpus,removeWords, stopwords("english"))
8	Stemming	email_text = tm_map(email_text,stemDocument)
9	Tokenization and Vectorization	doc_mat = DocumentTermMatrix(email_text)
10	Removal of sparse terms	doc_mat_sr = removeSparseTerms(doc_mat, 0.98)

The noise constituent of the dataset such as non-ASCII characters, HTML tags, extra white spaces, URLs, punctuations, numbers, and stop words were removed in steps 1–7 (Table 1). Stop words are a collection of words in any language that occur with a high

frequency but convey considerably less meaningful information about the significance of an expression. The removal of stop words and other noise constituents shrinks the size of the data and reduces the burden of computational expenses in model training, with the potential of improving model performance since there are only meaningful words left to learn from. The essence of stemming is also to reduce the data size by reducing words to their root word. The text-mining map function, defined in the text-mining (tm) package [30], received the parameters specified in steps 5–8 to execute each cleaning activity. The DocumentTermMatrix() and removeSparseTerms() functions are also defined in the tm package in R.

On completion of the data cleaning process, the data was divided into two; the train set and the test set. The train set, which was composed of 70% (23,107) of the original dataset, has 11,282 legitimate emails and 11,825 spam emails (Figure 1). The test set, which is composed of 30% (9753) of the original dataset, has 4744 legitimate emails and 5009 spam emails.

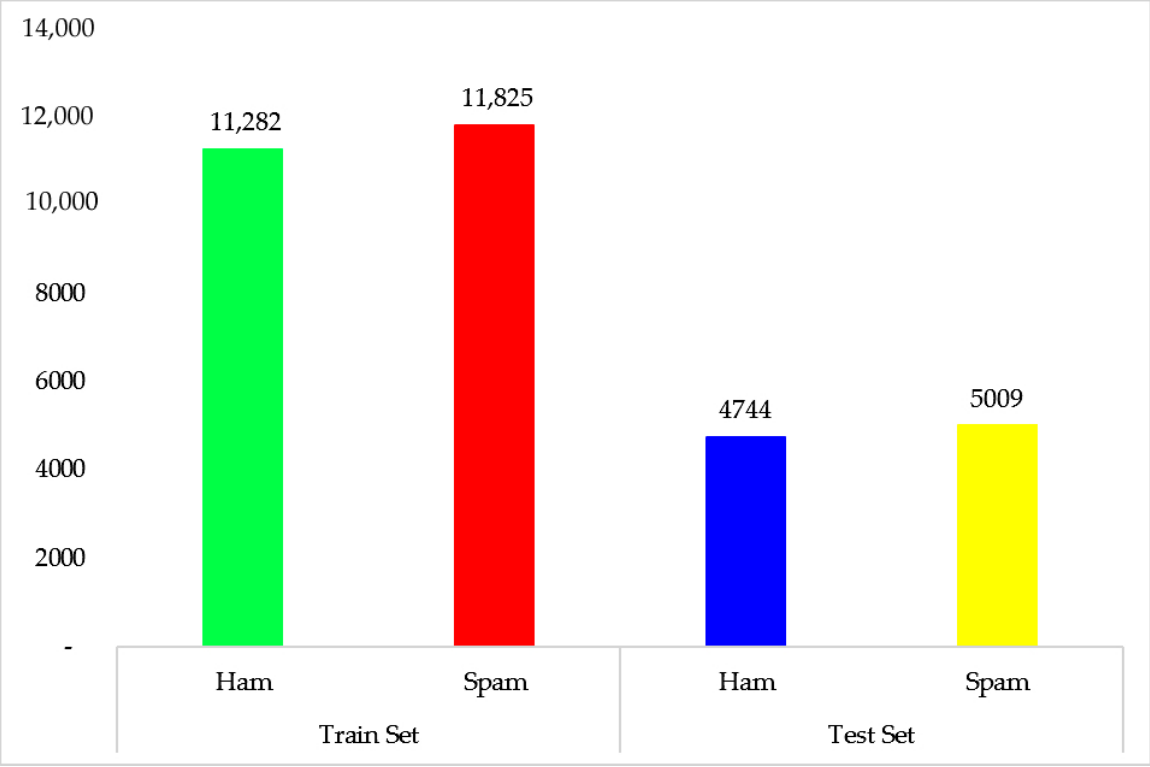

Figure 1. Distribution of ham and spam emails in train and test dataset.

Baseline models of random forest and extreme gradient boost (XGBoost) were developed by training and testing each model independently with 70% and 30% of the preprocessed dataset (Figure 2). All the parameters were set to their default values during the training and testing of the baseline models. The performance of these models on the test data was recorded as the baseline performance to be improved via hyperparameter tuning. To reduce the computation time, only the important predictors were passed to the random forest. The random forest has an inbuilt feature for ranking variables or features based on their importance in arriving at a prediction [31].

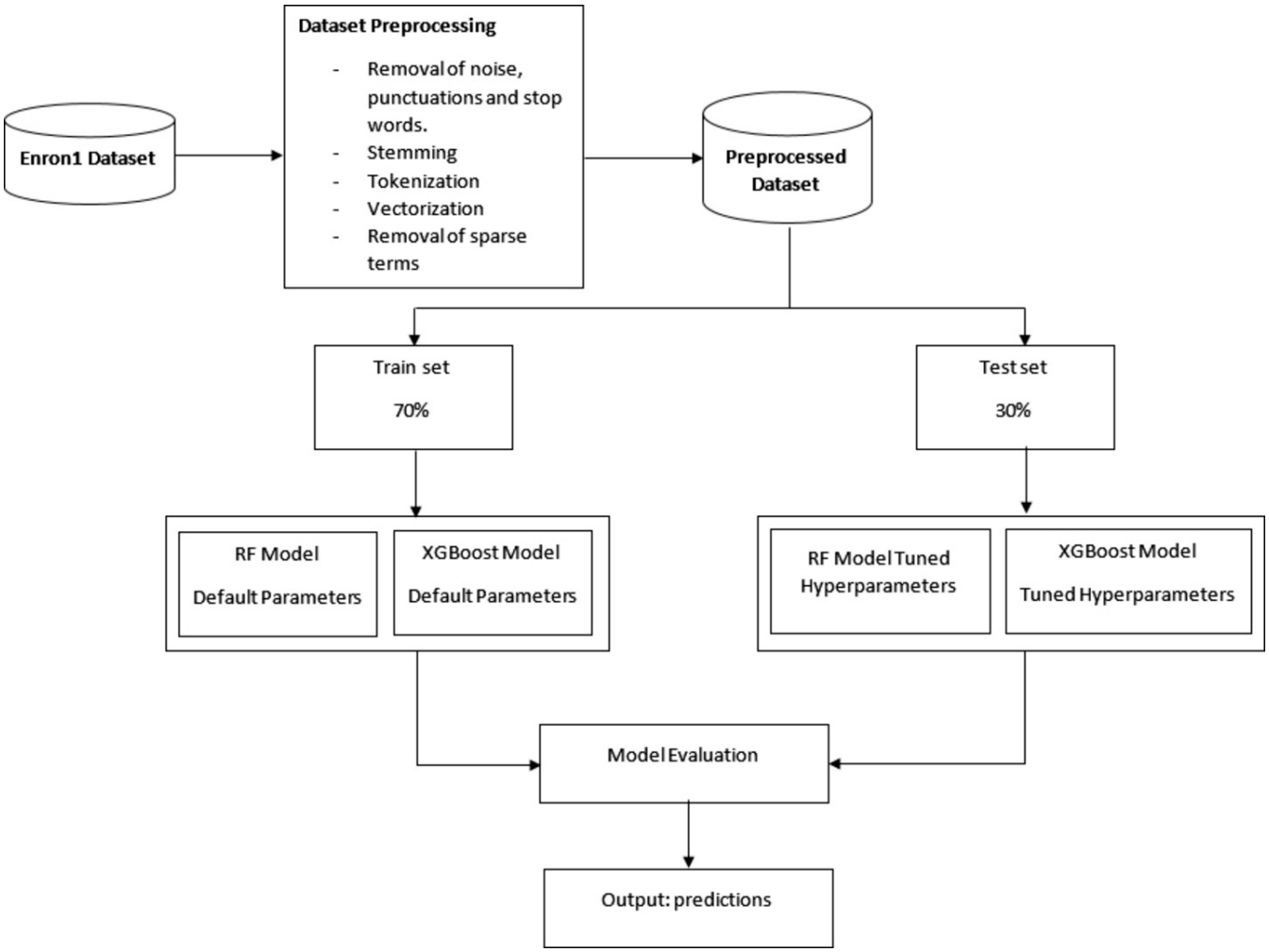

Figure 2. The proposed model workflow.

3.3. Methods and Machine Learning Classifiers

3.3.1. Random Forest

Random forest is a supervised ensemble classifier that is used for classification and regression. It is an ensemble learning method that creates a set of decision trees and combines them to make a final prediction. To arrive at a prediction, the random forest follows these steps:

Step 1: Select a random sample of data from the dataset.

Step 2: Build a decision tree using the sample data.

Step 3: Repeat the process a certain number of times, creating a new decision tree each time.

Step 4: Combine the decision trees by taking the average of their predictions.

Each decision tree in the random forest makes a prediction, and the final prediction is made by taking the average of all the predictions made by the individual decision trees. This helps to reduce overfitting and improve the overall accuracy of the model. The two important hyperparameters that must be defined by the user when generating a random forest are mtry and ntree [32]. In a random forest, mtry is the number of features that are randomly sampled as candidates for splitting at each decision tree node and the ntree is the number of decision trees in the random forest [32]. The mtry parameter determines how much randomness is injected into the model. A smaller mtry value will make the model more deterministic and potentially more accurate but at the cost of a more complex model that is more prone to overfitting. A larger mtry value will make the model more

robust to noise in the data, but at the cost of accuracy. The ntree parameter determines the overall complexity of the model. A larger value of ntree will make the model more accurate but at the cost of increased computational resources and longer training time. The number of trees in a forest (ntree) is not limited by computational resources, but the performance improvement from having a large number of trees is minimal, according to [33]. However, [34] states that computational resources are the limiting factor for the number of trees in a forest.

3.3.2. XGBoost

Extreme gradient boost (XGBoost) is a supervised ensemble machine-learning algorithm. The ensemble technique used by the algorithm is the boosting technique. The algorithm trains Classification and Regression Tree (CART) learners sequentially. Models are trained one after another, and the succeeding models are targeted at optimizing the loss function or residuals of the previous models in a bid to improve predictability. The algorithm works by calculating the similarity scores (Equation (1)) of the residuals of the independent email features at each node of the base CART [35].

$$SS = \frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n * \lambda} \quad (1)$$

$SS = Similarity\ Score$
$y_i = the\ value\ of\ the\ ith\ variable\ to\ be\ predicted$
$f(x_i) = the\ current\ prediction\ for\ independent\ features$
$n = the\ number\ of\ residuals$
$\lambda = regularization\ parameter$

The similarity score of each node within the tree is used to determine the information gain (Equation (2)).

$$G = SS_a - SS_b \quad (2)$$

$SS_a = Similarity\ Score\ after\ split$
$SS_b = Similarity\ Score\ before\ split$

The regularization parameter λ, reduces the effect of outliers and controls the pruning of the trees within the model. A higher value of λ will reduce the value of the similarity score which results in a smaller information gain (Equation (2)). The gain is compared with gamma γ, another important parameter used in the XGBoost algorithm. Gamma determines if there will be a split at a node. If gamma (γ) is less than gain (G), there will be a split at the node else, there will be no split. This serves as a mechanism to control overfitting. A high gamma would result in an extremely pruned tree. In this study, R default values of 0 and 1 were retained for γ and λ, respectively.

The prediction of the base CART model based on the selected leaf node is determined as:

$$Output = \frac{\sum_{i=1}^{n} (y_i - f(x_i))}{n * \lambda} \quad (3)$$

The sum of residuals is divided by the product of the number of residuals and the regularization parameter, λ. The predictions of successive CART models are calculated as in Equation (4).

$$New_p = Previous_p + (eta * Output) \quad (4)$$

$New_p = the\ prediction\ of\ the\ current\ CART\ learner$
$Previous_p = the\ prediction\ of\ the\ Previous\ CART\ learner$
$eta = this\ is\ the\ learning\ rate$

The output (Equation (3)) is the prediction of the base CART learner. The learning rate or eta is another important parameter that reduces the feature weight to prevent overfitting. It also determines how quickly a model will converge. The default value for this parameter in R is 0.3.

The workflow of the XGBoost algorithms discussed in Equations (1)–(4) can be summarized in seven steps as thus:

Step 1: Initialization—A set of decision-tree models is trained to classify a small set of emails as spam or not spam.

Step 2: Boosting—New models are added to the ensemble and trained to correct mistakes made by previous models by focusing on misclassified emails.

Step 3: Gradient Descent—Parameters of new models are optimized by minimizing the ensemble's loss function using gradient descent.

Step 4: Regularization—Regularization is used to prevent overfitting by penalizing models for having too many parameters or complex decision boundaries.

Step 5: Pruning—The decision tree is pruned by removing leaves with low weight to prevent overfitting and improve generalization.

Step 6: Repeat—Steps 2 through 5 are repeated for a fixed number of iterations or until a stopping criterion is met, such as reaching a certain level of accuracy or the number of boosting rounds.

Step 7: Return—The final ensemble of base models is returned as the final model and can be used to classify new emails as spam or not spam by taking a majority vote of the base models.

3.3.3. Hyperparameter Tuning Technique–Grid Search Method

The grid-search technique is an exhaustive search technique for finding optimal hyperparameters over a manually specified range of subsets. The choice of the grid-search technique in this study, among other hyperparameter-tuning techniques such as random search, gradient search, and so on, was informed by the less complex implementation and the fact that its execution can be parallelized. In this study, the grid search was used to determine the optimal values for hyperparameters such as the number of trees (max_depth), mtry, number of iterations (nrounds), and the learning rate (eta). The experimental setup used in searching the hyperparameter space of both random forest and XGBoost is presented in Table 2.

Table 2. Hyperparameter settings for the algorithms.

Algorithm	Hyperparameter Values
	Baseline Models
Random Forest	ntree = 500, mtry = 23
XGBoost	max.depth = 6, nrounds = 100, objective = ”binary:logistic”, eta = 0.3
	Tuned Models
Random Forest	First run: ntree = (1000, 1500) mtry = (7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40) Second run: ntree = (1000, 1500), mtry = (2, 3, 4, 5, 6)
XGBoost	max.depth = (5, 6, 7), nrounds = (70, 140, 210, 280, 350, 420, 490, 560, 630, 700), objective = ”binary:logistic”, eta = (0.20, 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.30), colsample_bytree = 0.6

3.3.4. Cross Validation

The efficiency of both RF and XGBoost was validated through K-fold cross-validation. This technique involves splitting the training set into K numbers of a subset. The K-1 subset of the entire dataset is trained and evaluated on a subset (i.e., 1-fold). This is repeated K number of times with a different subset of K used for evaluation at each iteration [36]. The essence is to rid the models of overfitting and sample bias. In this study K = 10. The 10-fold cross-validation was applied to the tuned models whereas the train/test split was applied to the baseline models.

The choice of random forest and XGBoost for this experimental study was informed by their high performance on similar tasks in other studies. Furthermore, the random forest

model uses the bagging technique which reduces variance and avoids overfitting through inbuilt out-of-bag error estimates. The outcome of several predictions is aggregated to obtain a final result. The XGBoost model uses the boosting technique which sequentially tries to optimize the loss function of a preceding CART learner through successive ones.

4. Results and Discussion

4.1. Experimental Testbed and Settings

The computation was carried out on a Windows 10 Pro 64bit Operating system with 16 GB RAM, 500 GB SSD, and Corei-7-1165G7@2.8GHz processor. The algorithms were implemented in R programming using R version 4.2.1 via Rstudio Integrated Development Environment (IDE). The major libraries used are randomForest, xgboost, tm, Caret, ggplot2, and ROCR. The hyperparameter values were set as presented in Table 2.

4.2. Results and Discussion

To optimize the performance of the baseline random forest model, efforts were targeted at determining the best ntree and mtry by tuning these parameters through grid search. The mtry of the baseline random forest model is 23; hence, mtry values between 6 and 41 were examined during the first run (Table 2). After 10-fold cross-validation, accuracy was used to determine the optimal ntree and mtry which were 1500 and 7, respectively, as shown in Figure 3. At ntree = 1500 and mtry = 7, the highest training accuracy of 0.9750725 was obtained. It took about a week for the model to converge.

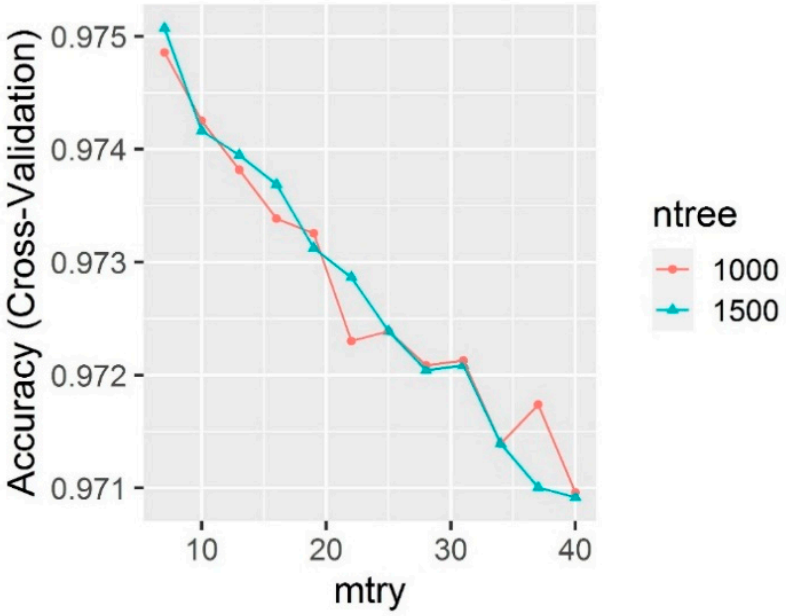

Figure 3. First run: model plot of the 10-fold cross-validation for the determination of best hyperparameter values for random forest.

From Figure 3, it can also be observed that as the value of mtry increases the predictive accuracy of the model was decreasing. This necessitated the need for a second run to examine mtry values between 1 and 7 given ntree values of 1000 and 1500. After about four days, the 10-fold cross-validation training converged, at the accuracy of 0.9752454, the optimal ntree = 1500, and mtry = 6 (Figure 4).

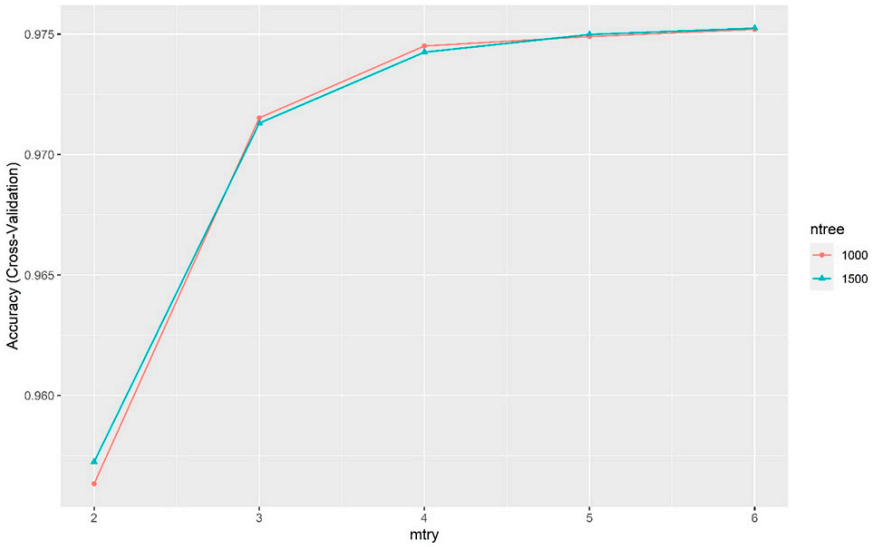

Figure 4. Second run: model plot of the 10-fold cross-validation for the determination of best hyperparameter values for random forest.

The grid search with 10-fold cross-validation based on accuracy returned the optimal values of 7560, and 0.25 for max.depth, nrounds, and eta, respectively, for the XGBoost model. From Figure 5, it can be observed that the accuracy improved progressively as the nrounds increased for each of the max.depths. However, as the number of boosting iterations (nrounds) tends toward 700, the improvement in accuracy became marginal. With a boosting iteration of 560, the accuracy peaked at 0.9781017 on the third face grid where max.depth is 7 and eta is 0.25.

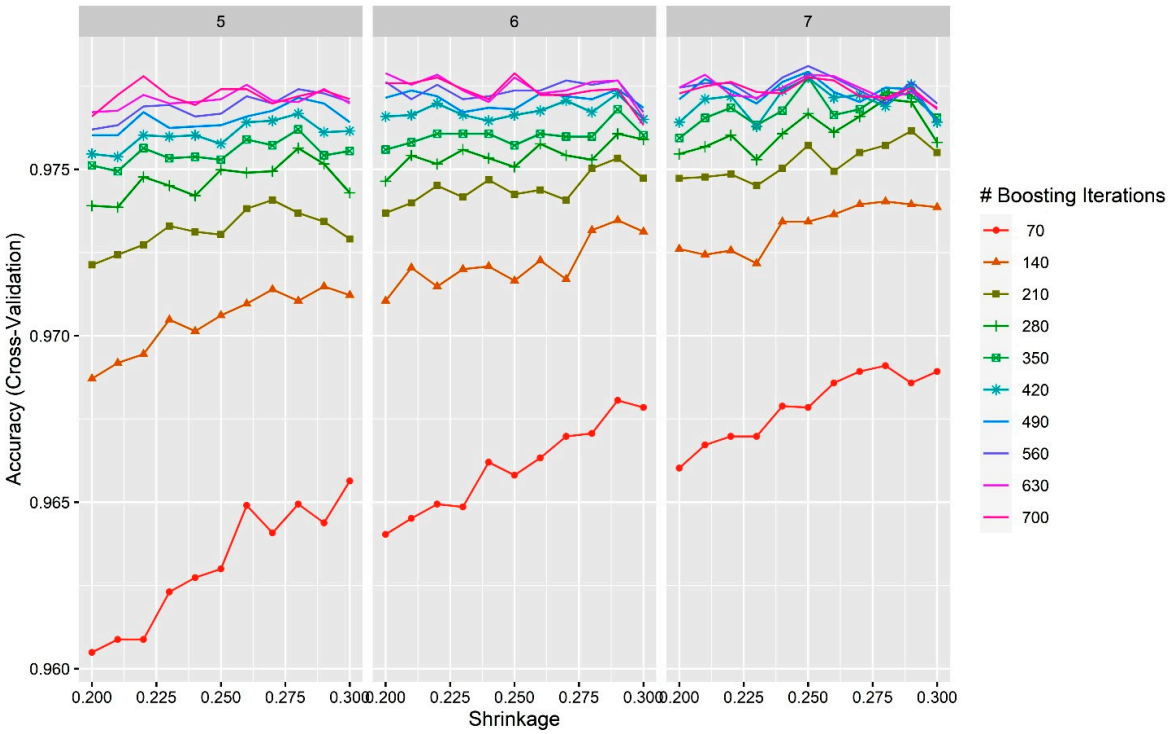

Figure 5. The model plot of the 10-fold cross-validation for the determination of best hyperparameter values for XGBoost. The face grid shows the performance of the model in terms of accuracy at max.depth of 5, 6, and 7 with varying nrounds and learning rates (eta). The colsample_bytree was held constant at 0.6.

4.2.1. Performance Evaluation Metrics

The metrics that were used to evaluate the performance of the ensemble algorithms are accuracy, sensitivity (recall), precision, F1-score, specificity, and the Area under the Receiver Operating Curve (ROC) curve. The true positive (TP), true negative (TN), false positive (FP), and false negative (FN), which are components of the confusion matrix (Table 3), were used to compute the performance metrics. The confusion matrixes for the random forest baseline model, XGBoost baseline model, random forest tuned model, and the XGBoost tuned model are presented in Table 4.

Table 3. A confusion matrix and its constituents.

	Predicted Class	
Actual Class	TP	FN
	FP	TN

Table 4. Confusion matrix of the baseline and tuned spam detection models of random forest (RF) and extreme gradient boost (XGBoost).

	RF Baseline Model			**XGBoost Baseline Model**	
	Spam	Ham		Spam	Ham
Spam	4907	102	Spam	4957	52
Ham	140	4604	Ham	194	4550
	RF Tuned Model			**XGBoost Tuned Model**	
Spam	4931	78	Spam	4951	58
Ham	139	4605	Ham	128	4616

where

Positive (P): Absolute number of spam emails.

Negative (N): Absolute number of ham emails.

True positive (TP): Absolute number of e-mails correctly classified as spam.

True negative (TN): Absolute number of e-mails classified as ham (not spam) and are truly ham.

False positive (FP): Absolute number of e-mails classified as spam, but they are not spam.

False negative (FN): Absolute number of e-mails classified as ham, but they are spam.

Accuracy

Accuracy measures the number of emails correctly predicted against the total number of predictions (Equation (5)). The accuracy metric is a reliable metric for performance evaluation, considering that there is no significant imbalance in the class distribution in the dataset used.

$$Accuracy = \frac{(TP + TN)}{(P + N)} \quad (5)$$

The random forest baseline model, XGBoost baseline model, random forest tuned model, and XGBoost tuned model achieved an accuracy of 0.9752, 0.9748, 0.9778, and 0.9809, respectively. This implies that for all classes 98.09% of the emails were correctly predicted by the tuned XGBoost model which achieved the highest accuracy as shown in Figure 6.

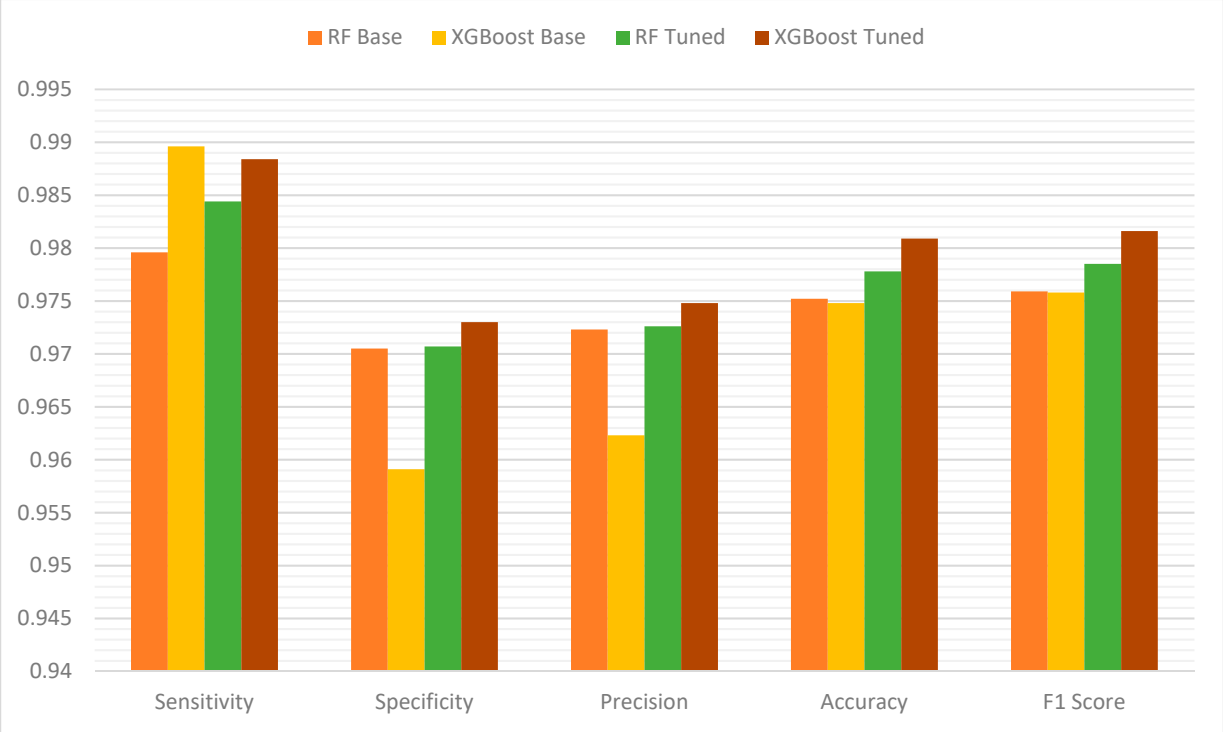

Figure 6. Performance evaluation metric comparison of the baseline and tuned random forest and XGBoost models.

Sensitivity (Recall)

Sensitivity measures the number of positives that are accurately predicted as a positive class for all positive data points. This metric is determined as presented in Equation (6).

$$Recall = \frac{TP}{(TP + FN)} \tag{6}$$

The random forest baseline model, XGBoost baseline model, random forest tuned model, and XGBoost tuned model achieved a sensitivity of 0.9796, 0.9896, 0.9844, and 0.9884, respectively. As observed in Figure 6, the XGBoost baseline model has the highest (sensitivity) spam email detection rate. However, it was outperformed by the other models in terms of accuracy, precision, specificity, and F1 score.

Precision

Precision measures the amount of accurately predicted spam emails against the total number of positive predictions. In other words, it measures the performance of the model with respect to the false positive. It is determined as thus:

$$Precision = \frac{TP}{(TP + FP)} \tag{7}$$

Figure 6 revealed that the tuned XGBoost model has the highest precision of 0.9748. This implies that for every email that was predicted spam, 97% of those emails are spam. The random forest base model, XGBoost-baseline model, and the random forest tuned model achieved a precision of 0.9723, 0.9623, and 0.9726, respectively.

F1-Score

The F1-score is a weighted mean of precision and sensitivity. It is determined as presented in Equation (8). The F1 score helps to measure the impact of false positives

and false negatives in the models. In other words, it balances the impact of low or high precision and recall (sensitivity) on the model.

$$F1 - Score = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity} \quad (8)$$

As seen in Figure 6, the tuned XGBoost model achieved the highest F1 score of 0.9816. The random forest baseline model, XGBoost baseline model, and the random forest tuned model achieved an F1-score of 0.9759, 0.9758, and 0.9785, respectively.

Specificity

Specificity measures the ability of the model to classify hams correctly. It is the percentage of negatives that are correctly detected as a negative class. It is determined as presented in Equation (9). The tuned random forest and XGBoost models were better at detecting legitimate emails, with the XGBoost tuned model achieving the highest specificity of 0.9730.

$$Specificity = \frac{TN}{(TN + FP)} \quad (9)$$

Receiver Operating Curve (ROC)

The ROC is a plot of the number of accurate predictions in the prediction of the positive class against the number of negative samples wrongly predicted as being in a positive class. It ranks the ability of the model to distinguish between the classes of emails. It measures the quality of predictions by the model. The Area Under the ROC Curve (AUC) measures the space under the ROC curve. The AUC ranged from 0 to 1. As the value of the AUC tends to 1, the better its distinguishing ability. The ROC curves for the random forest baseline model, XGBoost baseline model, random forest tuned model, and XGBoost tuned model are shown in Figure 7. The models' Areas Under the Curve (AUC) are 0.9894, 0.9892, 0.9966, and 0.9978, as shown in Figure 7. The AUC for the random forest baseline model, with the highest AUC value of 0.9978, indicates that the tuned XGBoost model distinguished between the spam and ham emails better than the other models.

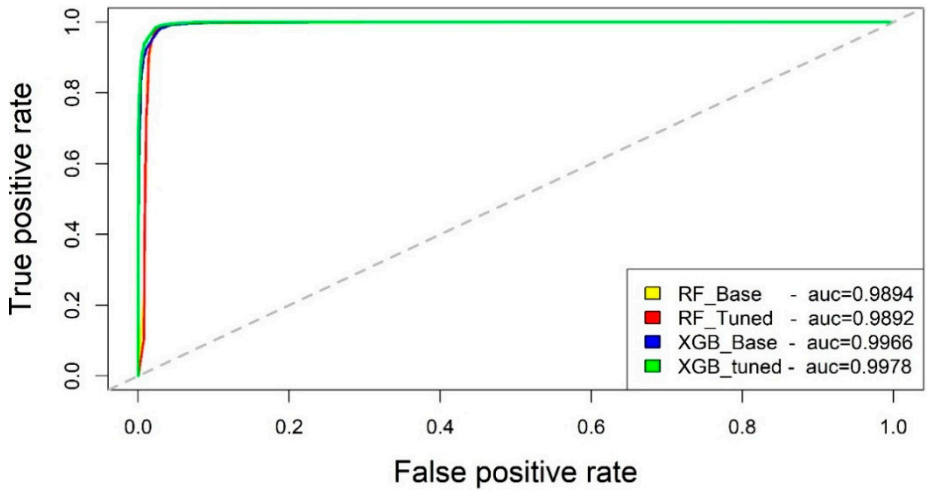

Figure 7. Area Under the Receiver Operating Curves of random forest and XGBoost models. The yellow plot line is the TPR against the FPR plot of the random forest-based model while the red, blue, and green are for the tuned random forest model, XGBoost baseline model, and XGBoost tuned model, respectively.

The hyperparameter tuning with 10-fold cross-validation significantly improved the performance of the ensemble models. Hyperparameter tuning does not improve model performance in all cases [37]. The tuned RF and XGBoost model performed better than

their un-tuned baseline models for all metrics except for the XGBoost baseline model with the highest sensitivity. The tuned XGBoost model outperformed the tuned random forest model for all measures evaluated as shown in Figures 6 and 7. It is also important to note that the XGBoost model is a faster algorithm (the XGBoost algorithm supports parallel computation) when compared with random forest. This is also established in the study by [38]. The random forest baseline model took 3762.38 s to train while it took 55.45 s to train the baseline XGBoost model. The hyperparameter tuning of the XGboost model took 23,925.48 s. The first search of the hyperparameter space for random forest took 585,857.11 s while the second run took 317,912.14 s.

5. Conclusions

This study evaluated and compared the performance of two ensemble models based on the random forest and extreme gradient boost ensemble algorithms. Baseline random forest and XGBoost spam detection models were developed based on the train/test split technique using the default parameters. The grid-search technique with 10-fold cross-validation was applied to search the hyperparameter space to determine the optimal hyperparameter values that optimized the performance of the random forest and XGBoost models. The performance of the baseline models was evaluated and compared with that of the tuned random forest and XGBoost models to examine the impact of hyperparameter tuning. The findings revealed that hyperparameter tuning improved the performance of the random forest and XGBoost models. The results also showed that the tuned XGBoost model outperformed the tuned random forest model for all metrics evaluated. The effectiveness of these ensemble models in spam email detection and classification was demonstrated.

It will be interesting to compare the XGBoost model and deep-learning models for spam detection in a future study in a bid to gain further insight into the development of efficient and effective spam email detection systems.

It is important to note that the distribution of classes in the dataset used in this study is complementarily balanced. The behavior of these models will be different with a significantly imbalanced dataset. Future studies will look at the performance of these models on an imbalanced dataset.

Author Contributions: Conceptualization, T.O.O.; methodology, T.O.O. and D.O.O.; software, T.O.O.; validation, T.O.O. and D.O.O.; formal analysis, T.O.O. and D.O.O.; investigation, T.O.O.; resources, T.O.O. and D.O.O.; data curation T.O.O. and D.O.O.; writing—original draft preparation, T.O.O. and D.O.O.; writing—review and editing, T.O.O. and D.O.O.; visualization, T.O.O. and D.O.O.; supervision, T.O.O.; project administration, T.O.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are openly available in GITHUB at https://github.com/MWiechmann/enron_spam_data (accessed on 17 August 2022). Application for consent to use the dataset is not required.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dixon, S. Global Average Daily Spam Volume 2021. Available online: https://www.statista.com/statistics/1270424/daily-spam-volume-global/ (accessed on 18 July 2022).
2. FBI. Federal Bureau of Investigation: Internet Crime Report 2021. Available online: https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf (accessed on 6 August 2022).
3. Securelist Types of Text-Based Fraud. Available online: https://securelist.com/mail-text-scam/106926/ (accessed on 4 August 2022).
4. Onova, C.U.; Omotehinwa, T.O. Development of a Machine Learning Model for Image-Based Email Spam Detection. *FUOYE J. Eng. Technol.* **2021**, *6*, 336–340. [CrossRef]

5. Bindu, V.; Thomas, C. Knowledge Base Representation of Emails Using Ontology for Spam Filtering. *Adv. Intell. Syst. Comput.* **2021**, *1133*, 723–735. [CrossRef]
6. Kaddoura, S.; Chandrasekaran, G.; Popescu, D.E.; Duraisamy, J.H. A Systematic Literature Review on Spam Content Detection and Classification. *PeerJ Comput. Sci.* **2022**, *8*, e830. [CrossRef] [PubMed]
7. Méndez, J.R.; Cotos-Yañez, T.R.; Ruano-Ordás, D. A New Semantic-Based Feature Selection Method for Spam Filtering. *Appl. Soft Comput.* **2019**, *76*, 89–104. [CrossRef]
8. Ahmed, N.; Amin, R.; Aldabbas, H.; Koundal, D.; Alouffi, B.; Shah, T. Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges. *Secur. Commun. Networks* **2022**, *2022*, 1862888. [CrossRef]
9. Hosseinalipour, A.; Ghanbarzadeh, R. A Novel Approach for Spam Detection Using Horse Herd Optimization Algorithm. *Neural Comput. Appl.* **2022**, *34*, 13091–13105. [CrossRef]
10. Ismail, S.S.I.; Mansour, R.F.; Abd El-Aziz, R.M.; Taloba, A.I. Efficient E-Mail Spam Detection Strategy Using Genetic Decision Tree Processing with NLP Features. *Comput. Intell. Neurosci.* **2022**, *2022*, 7710005. [CrossRef]
11. Ravi Kumar, G.; Murthuja, P.; Anjan Babu, G.; Nagamani, K. An Efficient Email Spam Detection Utilizing Machine Learning Approaches. *Proc. Lect. Notes Data Eng. Commun. Technol.* **2022**, *96*, 141–151.
12. Kontsewaya, Y.; Antonov, E.; Artamonov, A. Evaluating the Effectiveness of Machine Learning Methods for Spam Detection. *Procedia Comput. Sci.* **2021**, *190*, 479–486. [CrossRef]
13. Batra, J.; Jain, R.; Tikkiwal, V.A.; Chakraborty, A. A Comprehensive Study of Spam Detection in E-Mails Using Bio-Inspired Optimization Techniques. *Int. J. Inf. Manag. Data Insights* **2021**, *1*, 100006. [CrossRef]
14. Dedeturk, B.K.; Akay, B. Spam Filtering Using a Logistic Regression Model Trained by an Artificial Bee Colony Algorithm. *Appl. Soft Comput. J.* **2020**, *91*, 106229. [CrossRef]
15. Sagi, O.; Rokach, L. Ensemble Learning: A Survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
16. Sheu, J.J.; Chu, K.T.; Li, N.F.; Lee, C.C. An Efficient Incremental Learning Mechanism for Tracking Concept Drift in Spam Filtering. *PLoS ONE* **2017**, *12*, e0171518. [CrossRef]
17. Liu, X.; Zou, P.; Zhang, W.; Zhou, J.; Dai, C.; Wang, F.; Zhang, X. CPSFS: A Credible Personalized Spam Filtering Scheme by Crowdsourcing. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 1457870. [CrossRef]
18. Bahgat, E.M.; Rady, S.; Gad, W.; Moawad, I.F. Efficient Email Classification Approach Based on Semantic Methods. *Ain Shams Eng. J.* **2018**, *9*, 3259–3269. [CrossRef]
19. Agarwal, K.; Kumar, T. Email Spam Detection Using Integrated Approach of Naïve Bayes and Particle Swarm Optimization. In Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018, Madurai, India, 14–15 June 2018; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019; pp. 685–690.
20. Dada, E.G.; Bassi, J.S.; Chiroma, H.; Abdulhamid, S.M.; Adetunmbi, A.O.; Ajibuwa, O.E. Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems. *Heliyon* **2019**, *5*, e01802. [CrossRef] [PubMed]
21. Saha, S.; DasGupta, S.; Das, S.K. Spam Mail Detection Using Data Mining: A Comparative Analysis. *Smart Innov. Syst. Technol.* **2019**, *104*, 571–580. [CrossRef]
22. Nandhini, S.; Marseline, D.J. Performance Evaluation of Machine Learning Algorithms for Email Spam Detection. In Proceedings of the International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE 2020, Vellore, India, 24–25 February 2020; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2020.
23. Guangjun, L.; Nazir, S.; Khan, H.U.; Haq, A.U. Spam Detection Approach for Secure Mobile Message Communication Using Machine Learning Algorithms. *Secur. Commun. Networks* **2020**, *2020*, 8873639. [CrossRef]
24. Jancy Sickory Daisy, S.; Rijuvana Begum, A. Smart Material to Build Mail Spam Filtering Technique Using Naive Bayes and MRF Methodologies. *Proc. Mater. Today* **2021**, *47*, 446–452. [CrossRef]
25. Xia, T.; Chen, X. A Weighted Feature Enhanced Hidden Markov Model for Spam SMS Filtering. *Neurocomputing* **2021**, *444*, 48–58. [CrossRef]
26. Şimşek, H.; Aydemir, E. Classification of Unwanted E-Mails (Spam) with Turkish Text by Different Algorithms in Weka Program. *J. Soft Comput. Artif. Intell.* **2022**, *3*, 1–10. [CrossRef]
27. Xia, T.; Chen, X. Category-Learning Attention Mechanism for Short Text Filtering. *Neurocomputing* **2022**, *510*, 15–23. [CrossRef]
28. ENRON. The Enron-Spam Datasets. Available online: https://www2.aueb.gr/users/ion/data/enron-spam/ (accessed on 16 August 2022).
29. Wiechmann, M. GitHub—MWiechmann/Enron_spam_data: The Enron-Spam Dataset Preprocessed in a Single, Clean Csv File. Available online: https://github.com/MWiechmann/enron_spam_data (accessed on 17 August 2022).
30. Feinerer, I. Introduction to the Tm Package Text Mining in R. Available online: https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf (accessed on 16 August 2022).
31. Kolog, E.A.; Balogun, O.S.; Adjei, R.O.; Devine, S.N.O.; Atsa'am, D.D.; Dada, O.A.; Omotehinwa, T.O. Predictive Model for Early Detection of Mother's Mode of Delivery with Feature Selection. In *Delivering Distinctive Value in Emerging Economies*; Anning-Dorson, T., Boateng, S.L., Boateng, R., Eds.; Productivity Press: New York, NY, USA, 2022; pp. 241–264. ISBN 9781003152217.
32. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
33. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How Many Trees in a Random Forest? *Proc. Lect. Notes Comput. Sci.* **2012**, *7376*, 154–168.

34. Guan, H.; Li, J.; Chapman, M.; Deng, F.; Ji, Z.; Yang, X. Integration of Orthoimagery and Lidar Data for Object-Based Urban Thematic Mapping Using Random Forests. *Int. J. Remote Sens.* **2013**, *34*, 5166–5186. [CrossRef]
35. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 785–794.
36. Oyewola, D.O.; Dada, E.G.; Omotehinwa, T.O.; Emebo, O.; Oluwagbemi, O.O. Application of Deep Learning Techniques and Bayesian Optimization with Tree Parzen Estimator in the Classification of Supply Chain Pricing Datasets of Health Medications. *Appl. Sci.* **2022**, *12*, 10166. [CrossRef]
37. Hoque, K.E.; Aljamaan, H. Impact of Hyperparameter Tuning on Machine Learning Models in Stock Price Forecasting. *IEEE Access* **2021**, *9*, 163815–163830. [CrossRef]
38. Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A Comparative Analysis of Gradient Boosting Algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]