Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.1, February 2026, pp. 232~244
e-ISSN: 2548-6861

Optimizing Email Spam Detection through Handling Class Imbalance with Class Weights and Hyperparameter Using GridSearchCV

Muhammad Ridho Nursyam [1] *, Muhammad Koprawi [2] *, Dony Ariyus [3] *
* Teknik Komputer, Universitas Amikom Yogyakarta
muhammadridho@students.amikom.ac.id [1], koprawi@amikom.ac.id [2], dony.a@amikom.ac.id [3]

Article Info

Article history:

Received 2025-12-18
Revised 2025-12-28
Accepted 2026-01-08

Keywords:

Spam Detection,
Machine Learning,
Class Imbalance,
GridSearchCV,
Email Spam.

ABSTRACT

Email spam is a major problem in digital communication that can disrupt productivity, burden network resources, and pose a security threat. This research focuses on optimizing spam email detection using a machine learning approach by addressing class imbalance through class weighting and hyperparameter tuning using GridSearchCV. To improve model accuracy and sensitivity, a combination of diverse datasets is applied to provide a wider scope of training data. The models used in this study include Support Vector Machine (SVM), Random Forest, Multinomial Naive Bayes (MNB), and XGBoost. Evaluation is carried out based on metrics such as accuracy, precision, recall, and F1-score, before and after hyperparameter tuning. The experimental results show that SVM produces the highest accuracy after tuning, reaching 97.10%, compared to 96.73% before hyperparameter tuning. In addition, Random Forest, MNB, and XGBoost also show significant improvements, with each model achieving better performance after tuning. Overall, this study shows that dataset merging and class weight adjustment can significantly improve the model's ability to detect spam, as well as provide a basis for implementing the model in a more effective email spam detection system.

I. INTRODUCTION

Email spam is a significant challenge in modern digital communications due to the ever-growing volume of unwanted messages that can disrupt productivity, consume network resources, and pose security threats such as phishing or malware distribution [1] . Early rule-based detection models tend to be less effective in dealing with the increasingly complex variety and evolution of spamming tactics [2] . Machine learning approaches have become the primary solution in spam classification due to their ability to learn directly from historical data and automatically predict spam or non-spam categories, but model performance is still heavily influenced by issues such as class imbalance and the selection of appropriate features in email text data.

This study compares the performance of several machine learning algorithms in email spam detection with an approach that includes combining diverse datasets and handling class imbalance using Class Weight in each model. The analysis includes Support Vector Machine (SVM), Random Forest, Multinomial Naive Bayes (MNB), and XGBoost models, and evaluates the impact of hyperparameter tuning through GridSearchCV on spam detection performance. The expected benefits of this study include improving the sensitivity and accuracy of spam detection systems, and recommending appropriate models for practical implementation.

Numerous prior investigations have analyzed a variety of approaches in machine learning for spam email identification centering on identifying and adopting the optimum method for text representation. A case in point is the study where TF-IDF method is used and is found to perform better than other methods such as Bag of Words (BOW) in spam detection accuracy [3]. In all the studies machine learning techniques like Support Vector Machine (SVM) and Naive Bayes (NB) and Logistic Regression (LR) have been implemented. Although most of these studies demonstrate encouraging progress, a recurring problem is how to refine feature selection and text representation in order to respond to such challenges as class imbalance, and dimensionality reduction which have bearing on the accuracy of the model. In research

[3], with the aim of identifying malware that is concealed in spam emails, a number of machine learning methods were employed using two models of text representation Bag of Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF). The models used are Support Vector Machine (SVM), Naive Bayes (NB), and Logistic Regression (LR). The results of the experiments demonstrated that TF-IDF combined with Logistic Regression had the highest degree of accuracy, at 76.4% with an F1 score of 0.763. Additional pairings using TF-IDF and SVM also yield good results, achieving an accuracy of 75.7%. In addition, the Naive Bayes model with TF-IDF has the fastest execution time, although the accuracy is also the lowest at 74.3%. Overall, these results suggest the use of TF-IDF surpassed the use of BOW, underscoring the significance of appropriate text representation methods on the accuracy of spam email malware detection.

Integrating BERT (Bidirectional Encoder Representations from Transformers) with different machine learning classifiers to identify and classify emails as spam or ham emails [4]. Emails processed in BERT to extract features for text representation and different classification techniques like Logistic Regression, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), and Random Forest are applied. Their experimental study showed that Logistic Regression was the best in accuracy, precision, and F1 score in both datasets. In dataset 2, however this model's accuracy decreased marginally to 95.95% with precision at 96% and an F1 score of 95.92%. Logistic Regression outperformed others in spam classification as seen from SVM and KNN that performed lower than Logistic Regression on the same metrics. This shows that the combination of machine learning with BERT can be immensely helpful in spam detection.

In detecting spam emails, Support Vector Machine (SVM) employs n-gram and word2vec based feature extraction techniques [5]. When compared to other algorithm options, SVM yields the best results accuracy, precision, and F1 score values wise (based on multiple metrics). In the case of uni-gram and bi-gram combinations with SVM, the accuracy level reached was the highest (at 97.6%), and the precision was also the highest (at 98.8%), as was the F1 score (94.9%). This points to the capacity of SVM technique to provide outstanding spam emails signal detection, as SVM technique can deal with highly feature dimensions. Also, because SVM can generalize data well, SVM also demonstrates detection error reductions like false positives and false negatives.

To classify emails as spam using different features from the pre-processed data sets [6]. Classifying spam and ham emails resulted in a 96.90% accuracy score from the SVM algorithm. In addition SVM has spam Recall of 95.00% and spam Precision of 93.12% which indicates a good level of accuracy wherein spam emails are identified. In contrast to more accurate algorithms such as Naïve Bayes, the SVM algorithm is still impressive given its performance on big complex data sets. The strength of SVM is in its ability to class data points with the maximum separation margin making it powerful on large, complex data sets.

According to previous studies, email accounts can be assigned values, and using ML algorithms such as Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF), it is possible to classify email accounts as spam or non-spam [7]. The results showed that Naeive Bayes is highly suitable for classifying spam and non-spam email addresses, especially raising its overall sample classification to 88.17%, while the F1 score was 0.808. The next better was SVM with a sample classification of 80.70% and an F1 score of 0.762. Logistic Regression (LR) attached an F1 score of 0.787 and an overall sample classification of 83.49%. The accuracy for Random Forest was also very good at 85.40% and the F1 score at a good 0.817. NB not only showed a good overall classification. The execution of the other models was just as good, with Random Forest being the backbone of the Logistic Regression and SVM. All the models analyzed the new spam email data set, with NB being the focus of the tuning of the parameters for higher accuracy, F1 score results. The models also showed a good performance with parametric F1 optimization in 10-fold cross-validation spam detection. The four models had shown above good accuracy, demonstrating the ability to predict with spam email addresses flagged as high risk.

In the machine learning domain, Random Forest (RF) is one of the widely utilized algorithms, and the choice of this algorithm is also supported by its valuable and unique attributes [8]. The Spambase dataset comprises a total of 4601 emails accompanied by 58 features and was employed to evaluate the effectiveness of the machine learning algorithm. To enhance the algorithm's ability to identify spam emails more proficiently, the dataset's class imbalance was addressed by employing a random oversampling approach to balance the spam and ham distribution. The results of the experiment exhibited impressive outcomes where the models achieved 97% and spam and ham identification garnered similar performance measures of high precision, recall, and F1 scores. The aforementioned spam detection performance was quantitatively evaluated through the implementation of a confusion matrix and the ROC curve, resulting in an AUC measurement of 0.97. The model comparison also yielded promising results, with the proposed approach achieving a 6% increase span in performance relative to the previous models, attributed to its superiority.

This study aims to enhance the effectiveness of spam detection software by integrating multiple datasets for training. The developed model can learn rarer spam detection patterns with the amalgamated datasets and better capture a broader range of varied patterns. Overfitting would also be less of a risk with the multiple datasets. We represent model training data textually with TF-IDF to translate the spam texts into the numerical features required. The models we are working with are already demonstrated- SVM, MNB, RF, and XGBoost. To alleviate the data class imbalance we employ the Class Imbalance with Class Weight methodology to steer

the model's attention to the minority (spam) class for enhanced misclassification of that class. We hope the model will improve spam detection and classification to minimize misclassification which is a common issue of models working with restricted datasets.

II. METHOD

Figure 1 illustrates the method flow used in the research on Building a Machine Learning Model for Spam Email Classification. The process begins with data collection and dataset merging to create a more diverse dataset. After that, several pre-processing stages are carried out, including tokenization, stop word removal, data cleaning, and stemming. Next, text vectorization is performed using TF-IDF and class imbalance handling with class weights. The data is then divided into training and testing data. The model is trained using various classification algorithms, and hyperparameter search is performed with GridSearchCV to find the best combination. Finally, model evaluation is performed to measure the performance of the built model, and the process ends with the finish stage.

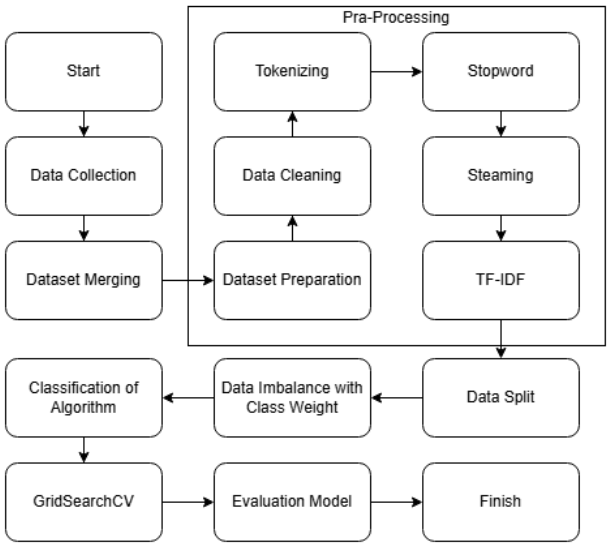

Figure 1. Flow Diagram

A. Dataset Preparation

During the Data Preparation phase, five different datasets were integrated, which allowed for greater dataset variability. The goal of the integration was to enhance the dataset variety, which is an important aspect of developing a model capable of accurately classifying spam emails. Following the integration, the dataset comprised 19,105 data records, and contained two labels spam and non-spam. The distribution of the datasets prior to and following integration is shown in Table I. Prior to the integration, a considerable class imbalance was observed in the smaller dataset, with spam class records being significantly fewer than non-spam class records. Following integration, the class proportions were more evenly distributed, though imbalances were still present in the distribution of the merged datasets. This is a critical aspect for model performance as it is influenced by imbalances in distribution and potential data duplication.

By having a diverse dataset, the built Machine Learning model is expected to be more effective in distinguishing spam emails from non-spam emails, as well as improving the accuracy and generalization ability of the model on new, previously unseen data using GridSearchCV. This process is crucial in the data preparation stage, as the quality and diversity of the dataset significantly affect the final performance of the built classification model.

TABLE I
DISTRIBUTION OF DATASETS

Dataset	Amount
spam.csv [9]	5169
Dataset_sms_spam_v1.csv [10]	628
email_spam_indo.csv link	2620
Spam_ham_dataset.csv link	4993
Emails.csv link	5695

B. Pre-Processing

1) Data Cleaning

Data Cleaning involves several processes to prepare the data for use in model training. Handling missing values or empty columns is done by identifying and removing rows with incomplete data [11] . This is important to ensure that the model only receives valid data and avoids imbalance or confusion during the training process. Furthermore, a text conversion process is also carried out on all data to avoid discrepancies between similar words but with different upper and lower case letters, such as the words "Spam" and "spam". This step aims to allow the model to recognize these words as the same entity without considering the differences in their writing format.

2) Tokenization

In the Tokenization stage, the email text is broken down into smaller units called tokens. Tokenization aims to simplify text processing by breaking the text into simpler words or phrases, so that the model can more easily analyze and understand the information contained therein [12] . This tokenization process is carried out using NLTK (Natural Language Toolkit) which utilizes the word_tokenize function to break the text into words. Each word generated from tokenization will be treated as a separate token, which will then be processed further. In addition, to ensure consistency, all words generated from tokenization are converted to lowercase to avoid differences in recognition between the same word, such as "Spam" and "spam". This tokenization process is an important step in data preparation, because by breaking the text into smaller tokens, the model can more easily identify relevant patterns and features needed for spam email classification.

3) Stopword

In the Stopwords Removal stage, words that are considered not to have significant meaning or contribution to spam classification, known as stop words [13]. Stop words are words that often appear in the text, such as conjunctions, prepositions, and connecting words, which do not provide important information for analysis, for example words like "and", "or", "from", and "to". This stop word removal aims to minimize noise in the data, so that the model can focus more on relevant words, such as more specific and in-depth words that can help in the spam classification process. In this study, the stop word list used includes stop words from Indonesian and English, which are obtained from the NLTK library for English and a stop word list adapted for Indonesian. After the stop word removal process, the resulting data will be cleaner and only focus on important words.

4) Steaming

In the Stemming stage, the process is carried out to change the words that have been previously processed to their basic form [14]. Stemming aims to reduce word variations by removing unimportant affixes (prefixes or suffixes), so that words with similar meanings can be treated as the same entity. For example, words like "berlari" and "lari" will be returned to their basic form "lari". This process is carried out using Sastrawi for Indonesian and PorterStemmer for English, both of which are effective in reducing words to their basic form. This stemming is important to ensure that the model is not trapped by unnecessary word variations and can focus on understanding the main meaning of the text. After the stemming process, these simplified words will become cleaner and more relevant input for the vectorization stage.

5) TF-IDF

In the TF-IDF (Term Frequency-Inverse Document Frequency) stage in this study, it is used to convert the processed text into a numerical representation that can be understood by the machine learning model [15]. After the text cleaning stage which includes tokenization, stop word removal, and stemming, the next step is to convert the text into a vector using TF-IDF. This technique calculates the weight of each word based on its frequency in the document (Term Frequency) and how important the word is in the entire dataset (Inverse Document Frequency). By using TF-IDF, words that appear frequently in one document but rarely appear in other documents will have a higher weight, while words that appear frequently in many documents will get a low weight. This process allows the model to focus more on relevant words in spam classification, such as "offer", "free", or "gift".

C. Data Split

In the Data Splitting stage of this study, the dataset is divided into two main parts, namely training data and testing data [16]. This division process aims to ensure that the model can be trained using one part of the data, while the other part is used to test its performance after training. In this study, the division was carried out with a proportion of 80% for training data and 20% for testing data, which is a standard approach in machine learning to ensure that the model gets enough data to learn, while also being able to be evaluated with data that has not been seen before. Data division is done using the train_test_split function from the sklearn.model_selection library, which randomly splits the dataset while maintaining a balanced distribution of labels across both sets.

D. Class Imbalance with Class Weight

Handling class imbalance that often occurs in spam classification datasets using class imbalance with class weight [17]. Class imbalance occurs when the number of samples in one class, such as spam emails, is much less than in other classes, such as non-spam emails. To overcome this problem, class weight is used which aims to give greater weight to the minority class (spam) so that the model becomes more sensitive to the data [18]. In this case, class weights are calculated using the class_weight method from the sklearn.utils library, using the balanced parameter to balance the classes based on the data distribution. The results of the class weight calculation are then applied to the model during the training process through the class_weight parameter. By giving greater weight to the spam class, the model is expected to pay more attention to classification errors in spam emails, thereby increasing the accuracy in detecting spam even though the amount of data for that class is less.

E. Classification Algorithm

1) Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm used for classification with a very effective approach in handling binary classification problems, such as spam email classification [19]. In this study, SVM is used to distinguish spam and non-spam emails by utilizing features extracted through TF-IDF vectorization. The basic principle of SVM is to find a hyperplane that optimally separates data from two classes, with the largest margin between the two classes [20]. In this case, the spam and non-spam classes are separated in such a way that the SVM can classify new emails based on the position of the data on the right side of the hyperplane. SVM is also very effective in handling high-dimensional datasets, such as text data that has gone through a vectorization process, making it suitable for use in this study. In addition, choosing the right kernel, such as linear or RBF (Radial Basis Function), is key to improving the model's accuracy in detecting relevant patterns in spam emails. Once the model is trained, the SVM can be used to predict whether an unclassified email belongs to the ham or spam email category. In two samples x and x', the radial basis function is expressed in equation (1), where $||x - \acute{x}||^2$is a free parameter that indicates the squared Euclidean distance.

$$K(x, \acute{x}) = e^{-\frac{||x-\acute{x}||^2}{2\sigma^2}} \quad (1)$$

2) Random Forest

Random Forest is an ensemble algorithm consisting of a collection of decision trees used for classification and regression [21]. In this study, Random Forest is used to classify spam emails by utilizing features generated through TF-IDF vectorization. This model works by randomly constructing multiple decision trees, where each tree is constructed using a random subset of the training data and a random subset of the existing features. Each decision tree makes its own prediction, and the prediction results from all trees are then combined through majority voting to produce the final decision. The advantage of Random Forest lies in its ability to handle high-dimensional data, such as text data, as well as its ability to reduce overfitting that often occurs in a single decision tree. In the context of spam email classification, Random Forest is able to handle class imbalance well and has stable performance even in the presence of noise or imperfect data [22]. In classification tasks, the final prediction $\hat{y}$for a data point $x$is determined through majority voting among the trees. If $T_i(x)$is the prediction from the th tree $i$for data point x, then the final prediction can be formulated using equation (2).

$$\hat{y} = mode\{T_1(x), T_2(x), \dots, T_{NTree}(x)\} \quad (2)$$

3) Multinominal Naive Bayes

Multinomial Naive Bayes (MNB) is a probabilistic algorithm used for text classification by assuming that the features (words) in the text are independent, although in reality there may be dependencies between features [23] . In the context of spam email classification, MNB utilizes conditional probability to model the relationship between words in an email and its label (spam or not spam). This model assumes that the distribution of words in the spam and non-spam classes follows a multinomial distribution, meaning each word has a probability of occurring in each class. During training, MNB calculates the probability of each word based on its distribution in each class, then uses Bayes' Theorem to calculate the likelihood that an email belongs to the spam or not spam class. In the Multinomial Naive Bayes model, it is assumed that each feature $x_i$is the result of a multinomial distribution, meaning that the features are counts of words or elements in a category. The probability $P(x_i|y)$for a word $x_i$to be assigned a class $y$can be calculated using the multinomial distribution formula (3).

$$P(x_i|y) = \frac{n_{y,x_i} + a}{\sum_k (n_{y,k} + a)} \quad (3)$$

4) XGBoost

XGBoost (Extreme Gradient Boosting) is a very popular and effective machine learning algorithm for classification tasks, including spam email classification. XGBoost is an implementation of the gradient boosting algorithm, which works by building models iteratively, where each new model tries to correct the errors made by the previous model [24] . The advantage of XGBoost lies in its ability to handle large and complex data, as well as its ability to optimize predictions by reducing bias and variance through regularization techniques. XGBoost works by building a series of decision trees that correct each other's errors, resulting in a more robust and accurate model. In spam email classification, XGBoost is able to handle class imbalance well and provides excellent results in terms of accuracy, precision, and recall. The training process begins with the optimization of the first tree, and as the model iterates through the trees $t$using equation (4).

$$\hat{y_i} = \sum_{k=1}^{t} f_k\,(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \quad (4)$$

Although GridSearchCV allows you to find the most optimal combinations of hyperparameters to improve the performance of models, the computational costs are high. This is especially true for XGBoost, as its hyperparameter tuning takes the most amount of time due to the high amount of time it takes to do iterative boosting as well as the tree training. In order to improve the boosting hyperparameter tuning time, parallel computing was used, which efficiently allocated the workload to the different CPU cores. Even though the amount of computational time was high, the amount of performance gained for the tuned models justified the time.

F. GridSearchCV

GridSearchCV is a hyperparameter search method used to find the best combination of parameters in a machine learning model [25] . With GridSearchCV, various values of the model's hyperparameters are systematically tested to find the configuration that provides the best performance based on a predetermined evaluation metric, such as accuracy. In this process, the user defines a grid containing the hyperparameter values to be tested, such as the kernel in SVM, the number of estimators in Random Forest, or the alpha value in Naive Bayes Multinomial (MNB). GridSearchCV then trains the model with various combinations of hyperparameters in the grid, and measures its performance on the test data using cross-validation to avoid overfitting. This process allows the selection of the model with the most optimal parameters, thereby improving the accuracy and generalization of the model in classifying spam emails.

G. Evaluation Model

The process of assessing model performance in performing classification or prediction tasks, usually uses metrics created from the confusion matrix [26] . Some common metrics applied to evaluate models are accuracy, which measures the proportion of correct predictions; precision, which measures how many positive predictions are actually positive; recall, which measures the model's skill in finding all positive cases; and F1-score, which is the harmonic mean score between

accuracy and recall. The model's skill in classifying positive and negative classes at various thresholds is also often evaluated with AUC-ROC (Area Under the Receiver Operating Characteristic Curve).

To assess evaluation models, we turned to 5-fold cross validation in order to maximize protection against overfitting. Thus, for each fold, 80% of the data was devoted to training and 20% was allocated for testing. The final performance metric was based on averaging the results of the 5 folds. However, in instances where cross validation was omitted, we defaulted to a classic train-test split methodology, where again, 80% of the data was employed for training and 20% for testing.

- Accuracy

Accuracy is an evaluation metric that measures the proportion of correct predictions from the total number of predictions made by the model. Accuracy is calculated using formula (5).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

Where:

TP (True Positive) the number of correct positive predictions.

TN (True Negative) the number of correct negative predictions.

FP (False Positive) is the number of incorrect negative predictions.

FN (False Negative) the number of incorrect positive predictions.

- Precision

Evaluation metric to calculate the level of accuracy of the model in classifying positive data, namely how many positive predictions are actually positive. Precision is calculated using formula (6).

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

- Recall

An evaluation metric that measures how well the model detects all positive cases in the dataset. Recall is calculated using formula (7).

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

- F1-Score

An evaluation metric combining precision and recall to provide a more balanced picture of model performance, especially when there is an imbalance between the two. The F1-Score is calculated as the harmonic mean of precision and recall, with formula (8).

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

- AUC-ROC

The relationship between True Positive Rate (TPR) and False Positive Rate (FPR) is plotted using the ROC curve at various classification thresholds. In formula (9) to calculate TPR.

$$TPR = \frac{TP}{TP + FN} \quad (9)$$

In formula (10) to calculate FPR.

$$FPR = \frac{FP}{FP + FN} \quad (10)$$

After the ROC curve is drawn, to show how well the model distinguishes between positive and negative classes, the AUC is calculated by calculating the area under the ROC curve. AUC ranges from 0 to 1. AUC = 1 indicates a perfect model in classification. AUC = 0.5 indicates a model that is no better than random guessing.

III. Results and Discussion

A. Class Distribution Dataset

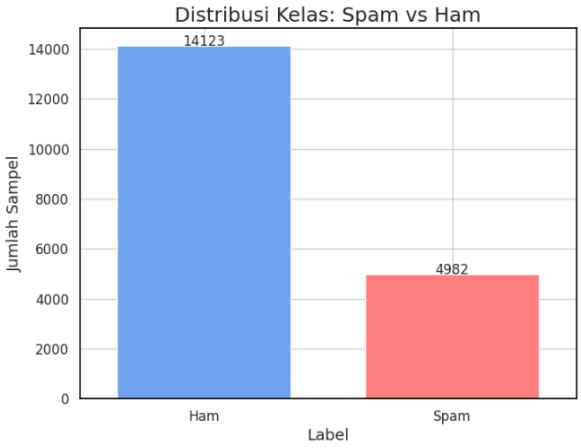

Figure 2. Dataset Class Distribution

Figure 2 shows the class distribution of the final dataset resulting from combining four different datasets. This dataset includes spam and ham emails in both Indonesian and English, with a total of 19,105 samples. This distribution shows that the ham class (non-spam emails) is much more dominant, with 14,123 samples, while the spam class only has 4,982 samples. This imbalance between the number of spam and ham data reflects a common challenge in spam detection, where the spam class is smaller than the ham class. To address this imbalance, Class Imbalance with Class Weight will be used to give a greater weight to the spam class, making the model more sensitive to misclassifications in minority classes.

B. Preprocessing

In the data cleaning stage, data is cleaned to ensure that only valid data is used in model training. This process

involves handling missing values by removing rows with incomplete data, as well as converting text to lowercase to avoid mismatches between similar words, such as "Spam" and "spam." Next, in tokenization, email text is broken down into smaller word tokens, making it easier for the model to understand each text element. After that, a stopword removal stage is performed to remove common words such as "and," "or," and "the" that do not provide much information in spam classification. Finally, a stemming process is used to change words with varying forms, such as "mencari" to "cari," to ensure that the model only learns root words relevant to spam classification. All of these stages produce cleaner, more structured text, ready for use in the vectorization stage.

After the preprocessing stage, the cleaned and processed text is converted into a numerical representation using TF-IDF, which calculates word weights based on their frequency in the document and their importance in the entire dataset. The TF-IDF results help the model recognize more relevant words for spam classification, such as "offer" or "gift." The dataset is then divided into training and testing data with a proportion of 80% for training and 20% for testing, using the data split method. To address the problem of class imbalance, which is an imbalance in the amount of data between the spam and ham classes, class weighting is applied to give more weight to the minority class (spam), making the model more sensitive to errors in the fewer spam classes. This process ensures that the model is trained with more balanced data, which improves the model's generalization ability and accuracy in classifying spam emails.

C. Clean Dataset

TABLE II
DATASET BEFORE PREPROCESSING

No.	E-mail	Label
1	: fw : having iris visit london anita, it seems that i am going to london next week. please see forwarded emails . can you please assist me with my travel arrangements . thanks, iris - - - - - original message - - - - - from : Kaminski , Vince …	0
2	: immediate reply needed dear sir, i am dr james alabi, the chairman of contract award and review committee set up by the federal government of nigeria under the new civilian dispensation to award new contracts and review…	1
3	Ambil tindakan segera atau lewatkan. 003 - 300299717499832716 Perhatian! Pelanggan Nilai # 772 - 00 D 87 "Klaim sistem gratis Anda" atau hubungi 1 - 800 - 823 - 2466 Selamat! Anda telah dipilih untuk menerima sistem …	1

A comparison between Table II and Table III reveals significant changes in the format and structure of the text. In Table II, the email text still contains irrelevant elements such as punctuation, sender information, delivery time, and long, complex sentences, making it difficult to process further. Furthermore, some words in English and Indonesian are still connected without a clear separation.

TABLE III
DATASET AFTER PREPROCESSING

No	E-mail	Label
1	fw iris visit london anita seem go london next week please see forward email please assist travel arrangements thank iris origin message Kaminski Vince …	0
2	immediate reply need dear sir dr jame alabi chairman contract award review committee set feder govern nigeria new civilian dispens award new contract review …	1
3	ambil tindakan segera lewatkan perhatian pelanggan nilai klaim sistem gratis hubungi selamat telah dipilih menerima sistem …	1

After a preprocessing stage that included tokenization, stop word removal, lowercase conversion, and stemming, Table III shows a much cleaner and more structured text, with only relevant words retained. For example, words like "fw," "please," and "thanks" were removed, while important words like "iris," "london," and "travel" were retained. This demonstrates that the preprocessing process successfully streamlined the text by removing unnecessary information and focusing on more relevant words for spam classification, which is crucial for improving the model's accuracy in analyzing spam emails.

D. Performance Before Tuning of Machine Learning Algorithm

Before hyperparameter tuning with GridSearchCV, the performance of the four classification models, SVM, Random Forest, MNB, and XGBoost, was visualized with confusion matrices shown in Figure 3. This confusion matrix shows the models' ability in predicting spam emails (1) and non-spam emails (0). In SVM, the model predicted 2821 instances of the negative (non-spam) class and 875 instances of the positive (spam) class, and both classes had prediction errors that were within the noise. The Random Forest model predicted 2839 instances of the negative (non-spam) class and 856 instances of the positive (spam) class, so the model also had high accuracy across both classes. MNB and XGBoost models predicted with the same distribution albeit a little less with MNB predicting 2766 instances of the negative class and 856 instances of the positive class, and XGBoost predicting 2796 instances of the negative class and 767 instances of the positive class. The models all predicted well, but the confusion matrix illustrates that the models' ability to predict values in the positive (spam) class and thus predict values with accuracy in the positive class, needs to be further optimized in the next tuning phase.

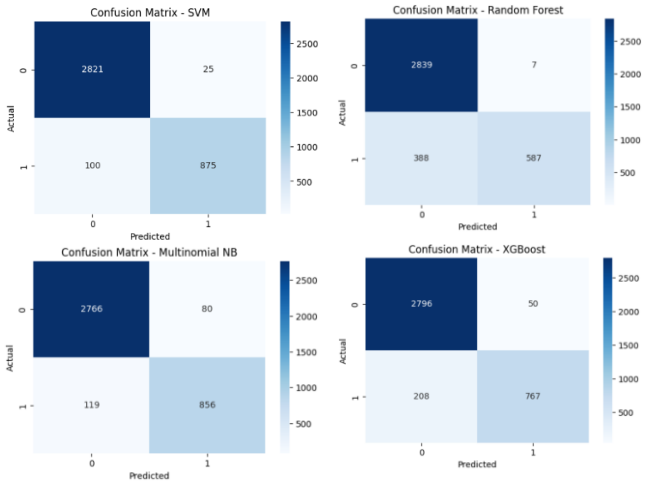

Figure 3. Confusion Matrix Model Before Tuning GridSearchCV

The classification models before hyperparameter tuning were SVM, Random Forest, MNB, and XGBoost, and their performance metrics were evaluated using Receiver Operating Characteristic (ROC) curves (as shown in Figure 4). ROC curves show the relation between TPR and FPR. SVM and MNB models were the most successful as they obtained AUC (Area Under Curve) of 0.99, which means they had an outstanding performance in distinguishing the spam and non-spam classes for the text messages. On the other hand, Random Forest and XGBoost models also successfully distinguished between spam and non-spam messages as well; however their performance was slightly lower (0.98) compared to SVM and MNB. Finally, all four models distinguished spam messages effectively as evidenced by the ROC curves because they were very close to the upper left corner of the figure which represents accurate classification of spam.

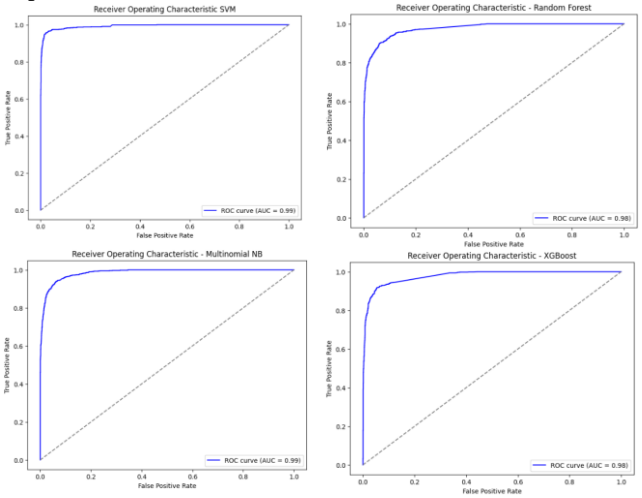

Figure 4. ROC Model Before Tuning GridSearchCV

The K-Fold Cross-Validation curves displayed in Figure 5 depict the outcomes of cross-validation for four classification models (SVM, Random Forest, MNB, XGBoost) in the pre-hyperparameter tuning via GridSearchCV stage. Accuracy for each K-Fold Cross-Validation model over the five folds of the process is presented in each of the five plots. The highest value for the SVM model was in the 3rd fold, with highest accuracy of almost 0.974, while the lowest value of accuracy was in the 2nd fold with close to 0.966. Fold accuracy over the folds of Random Forest model was moderately evenly spaced. The highest value of accuracy was the 3rd fold with (0.897) while the lowest value was in the 4th fold with 0.892. For MNB model, the accuracy fluctuated more with the highest in the 4th fold (0.952) and the lowest in the 2nd fold (close to 0.944). XGBoost was stable with fold 3 being the highest 0.937 and fold 2 being the lowest around 0.933. The models before additional tuning provides information on the extent of their generalization to the data in each fold.

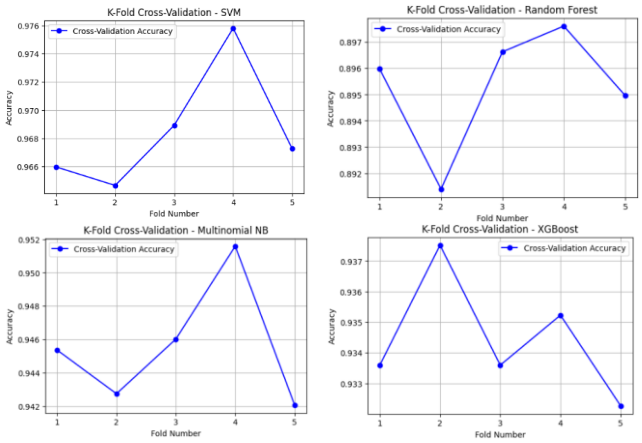

Figure 5. K-Fold Model Before Tuning GridSearchCV

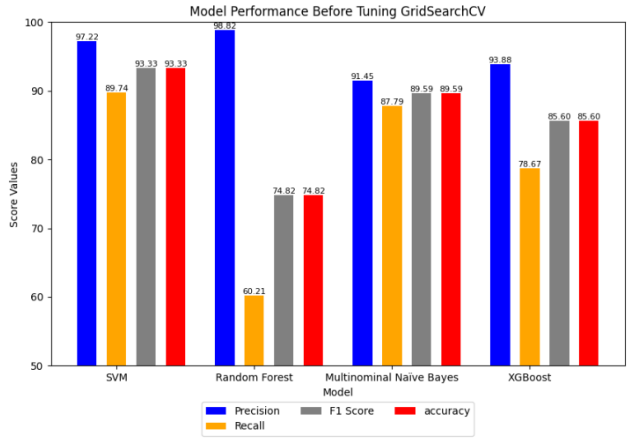

Figure 6. Performance Model Before Tuning GridSearchCV

Among the four classifiers, Support Vector Machine, MNB, Random Forest, and XGBoost, models performance based on Precision, Recall, F1 Score, and Accuracy, before hyperparameter tuning with the help of GridSearchCV, are illustrated in figure 6. Support Vector Machine outperformed the rest in terms of Precision (97.22) and Accuracy (94.79), albeit with a Recall of 69.74. Although SVM was correct in predicting quite a number of correct email, he was not so

sensitive on the spam classes. Random Forest, on the other hand, also had a Precision of 93.33, albeit dominated by a Recall of 60.21. This indicated that while predicting also had quite the accuracy, he was not quite as SVM in spam detecting. MNB was the opposite in that he had quite the number of Recall (sensitivity) with 87.79, albeit Precision was lower with 74.82. Thus, he was in a sense more fashion of detecting spam, albeit had more falsies in hand when predicting (was a higher false positive prediction). XGBoost on the other hand with the same number of 85.60 had a good and balanced score in F1 score where the Precision value also 85.60, Recall of 78.67 thus balancing quite the number of metrics.

E. Performance After Tuning of Machine Learning Algorithm

The SVM, Random Forest, MNB, and XGBoost models and their confusion matrices are displayed in Figure 7 and are the results for the classifiers after the adaptation of hyperparameters and the application of the GridSearchCV technique. The performance of the models improved substantially in these results as compared to the results before tuning. In the SVM model, the amount of erroneous predictions that were made in the spam class prediction fell, with 888 correct predicted spam (positive) class predictions and just 87 incorrect predictions. The positives in the Random Forest model were 894 positive class predictions, and 81 were negatives, and thus there was significant improvement in this model as well. In this case MNB and XGBoost were similar and MNB classified 869 spam emails, while XGBoost 891 spam emails was spam accurate and improved the overall before performance standards. The confusion matrices indicate that tuning results prediction accuracy overall.

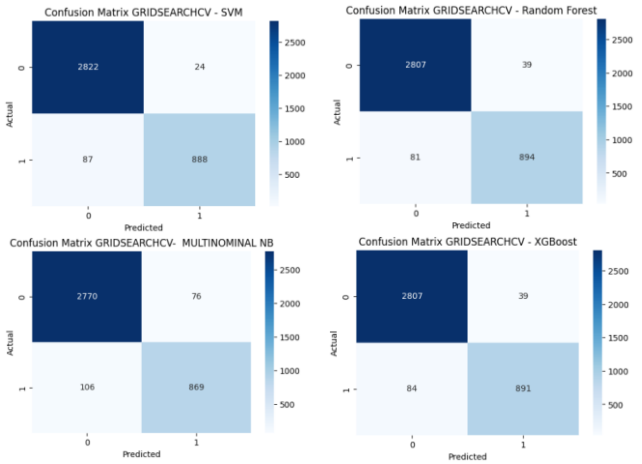

Figure 7. Confusion Matrix Model After Tuning GridSearchCV

Besides basic metrics like accuracy, precision, recall, or F1-score which are useful to understand a model's performance, we have also used other metrics more affected by class imbalance, such as ROC-AUC and the precision-recall curve, to evaluate model performance. These metrics, in particular, are important for the email spam detection problem where the spam class (minority class) is usually less represented. As noted in Figure 8 (precision-recall curves), the models performed much better post hyperparameter tuning, especially in the email spam and non spam classification problem. All tuned models ROC-AUC values increased, including SVM, Random Forest, MNB, and XGBoost, which indicates the models had a better ability to classify the positive (spam) and negative (not spam) classes thereafter. The tuned models precision-recall curves also demonstrated greater sensitivity to the spam class as the post tuning models had better precision and recall for the detection of spam.

Figure 8 presents the Receiver Operating Characteristic (ROC) curves for the four classification models SVM, Random Forest, MNB, and XGBoost post hyperparameter tuning using the GridSearchCV method. These ROC curves depict the relationship between True Positive Rate (TPR) and False Positive Rate (FPR) for the models. All models achieved quite an AUC (Area Under Curve) score of 1, thus SVM and MNB were the most effective AUC 0.99 and capable of detecting spam with the Random Forest and XGBoost models performing AUC 0.98 and 0.99, and thus able to successfully differentiate spam from non spam. These ROC curves, therefore, attest to the fact that hyperparameter tuning led to an improvement in model performance and that all models were able to achieve spam detection with high levels of accuracy.

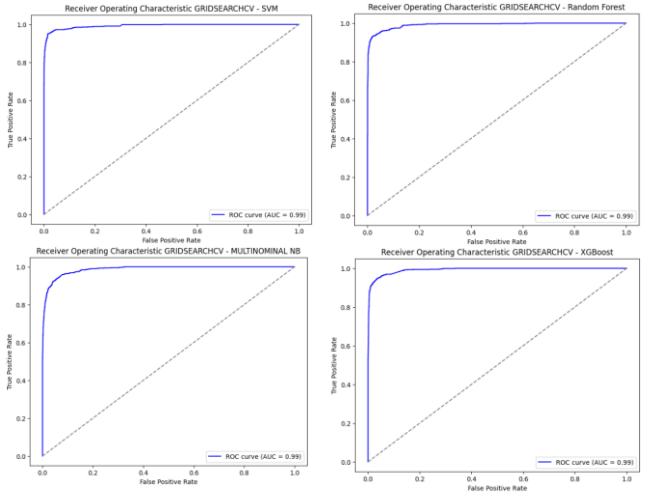

Figure 8. ROC Model After Tuning GridSearchCV

Figure 9 displays the results of K-Fold Cross-Validation for four classification models, SVM, Random Forest, MNB, and XGBoost, following hyperparameter tuning through GridSearchCV. The SVM model demonstrates the greatest improvement in cross-validation accuracy post tuning. Overall, the results suggest improvement in model accuracy following hyperparameter tuning. SVM recorded the highest accuracy in the 3rd fold (around 0.976) and the lowest in the 2nd fold (around 0.967), indicating stable but slightly

changing performance. Random Forest also demonstrates continuous improvement of accuracy as it demonstrates consistency with the highest 3rd fold (around 0.972) and 5th fold (around 0.968) and falling slightly to 4th at around 0.965. The MNB model also displays improvement of accuracy as seen in the 4th fold of around 0.958, as XGBoost demonstrated accuracy in the 3rd fold of around 0.965, but with slightly less performance improvement. Based on the results it can be determined to be slightly more stable following hyperparameter tuning as the accuracy to classify each fold demonstrates more improvement.

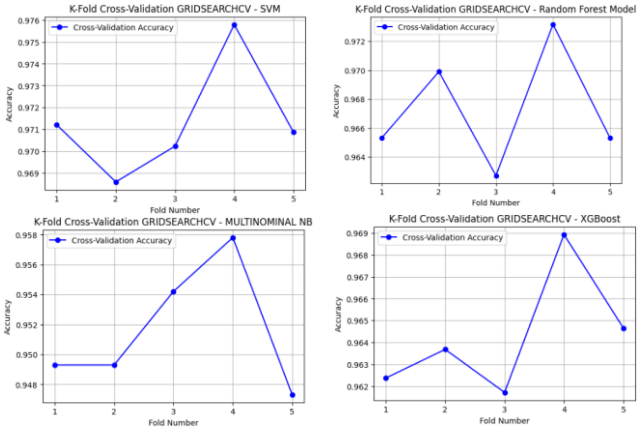

Figure 9. K-Fold Model After Tuning GridSearchCV

In Figure 10, the performances of hyperparameter tuned SVM, Random Forest, MNB, and XGBoost classification models are compared.

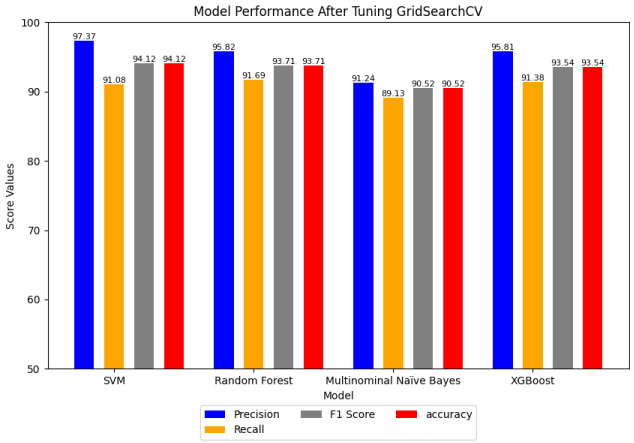

Figure 10. Performance Model After Tuning GridSearchCV

All the models performed well. However, SVM showed the best performance overall with the highest Precision of 97.37 and highest accuracy of 94.12, which denoted that the model was able to detect the positive classes with high accuracy and was able to generalize robustly as well. Random Forest and MNB models performed similarly with Random Forest achieving Precision of 91.69 and accuracy of 94.12 as compared to MNB which had Precision of 91.24 with an accuracy of 93.71. XGBoost also performed well with Precision of 93.54 and accuracy of 93.54, which made the model balanced in recognizing spam and non spam classes. The stacking models displayed very high F1 Scores which showed that hyperparameter tuning was effective in improving the model accuracy and spam detection efficiency due to the improved balance in the models.

F. Comparison of Accuracy Before and After Tuning on the Model

Table IV compares the results of the four different classification models without GridSearchCV hyperparameter tuning. According to this table, SVM achieved the highest accuracy, 96.73%, of the four models. It also achieved high Precision (97.22%), proving that SVM can accurately capture spam emails. However, SVM also achieved lower Recall (89.74%), so this model is also likely to miss some spam emails. For Random Forest, the Precision (98.82%) is higher than SVM, but Recall is lower (60.21%), indicating that this model is also not sensitive to spam. Out of the other two, MNB performed better than XGBoost, having also achieved recall that is pretty decent (89.59%), while XGBoost achieved Precision (93.88%) and Recall (78.67%) that is also pretty decent, but lower than the other two. However, the accuracy of the latter two is a little lower than SVM.

TABLE IV
COMPARISON OF CLASSIFICATION MODEL PERFORMANCE BEFORE GRIDSEARCHCV HYPERPARAMETER

Performance	SVM	Random Forest	MNB	XGBoost
Accuracy	96.73	89.66	94.79	93.25
TP	875	587	896	367
FN	100	388	119	208
TN	2821	2839	2796	2796
FP	25	7	80	50
Precision	97.22	98.82	91.45	93.88
Recall	89.74	60.21	87.79	78.67
F1_Score	93.33	74.82	89.59	85.60

After conducting hyperparameter tuning via GridSearchCV, all models encountered performance enhancements and were documented in Table V. Detecting spam became less problematic for the SVM model, as minor improvements to sensitivity were evidenced with increases in both Recall and accuracy to rates of 91.08% and 97.1%, respectively. Improvements in spam detection directed at the Random Forest model were also documented as notable, where both Recall and accuracy rates were elevated to 91.69% and 96.86%, respectively. Other improvements were documented in the MNB model as evidenced in the Recall increase to 89.13% as well as an F1 Score of 90.52% and XGBoost model as evidenced in the Recall increase to 91.38% as well as an enhanced F1 Score of 93.54%. Ultimately, all models performance were elevated with respect to improvements in accuracy and hyperparameter

tuning via GridSearchCV which allowed for improved sensitivity and classification towards spam detection.

TABLE V
COMPARISON OF CLASSIFICATION MODEL PERFORMANCE AFTER GRIDSEARCHCV HYPERPARAMETER

Performance	SVM	Random Forest	MNB	XGBoost
Accuracy	97.10	96.86	95.24	96.78
TP	888	894	869	891
FN	87	81	106	84
TN	2822	2807	2770	2807
FP	24	39	76	39
Precision	97.37	95.82	91.24	95.81
Recall	91.08	91.69	89.13	91.38
F1_Score	94.12	93.71	90.52	93.54

After hyperparameter tuning with GridSearchCV, out of all tested models, SVM performed the best. Even though SVM had an impressive accuracy of 96.73% prior to tuning, it improved to 97.10% after tuning, with Recall also increasing to 91.08% from 89.74%. This shows SVM became more accurate in predicting spam emails after tuning. Moreover, SVM also had after tuning very high Precision of 97.37%, showing this model not only made accurate classifications but also very effective ones in spam detection. Having an F1 Score of 94.12%, SVM had one of the best Precision to Recall ratios, thereby making it the most ideal model in terms of overall sustenance after tuning for spam detection in emails.

In this research case, the experimental results indicate that the various models (MNB, XG Boost, Random Forest, and SVM) show different degrees of accuracy improvements when hyperparameter tuning is done. A paired t-test determined the statistical significance of the gaps left by the hyperparameter tuning processes for each of the models. The results from the paired t-test indicate that the lack of accuracy changes post hyper-parameter tuning for the SVM model was statistically insignificant ($p = 0.240$). This shows that tuning largely did not trigger noticeable improvements for this model. On the contrary, accuracy improvements for Random Forest ($p = 1.05e-07$), MNB ($p = 0.0037$), and XG Boost ($p = 3.60e-06$) after tuning were statistically significant, and as such, it was confirmed that hyperparameter tuning improved the model performance.

Although this study has primarily focused on the optimization of spam detection using different machine learning algorithms, the developed model has numerous potential applications including commercial use. The model has the potential to be used in email spam filtering systems used by mail servers. If the model is integrated into mail servers, email messages can be analyzed on the fly with spam message detection prior to user messages arriving at the user inboxes. Moreover, the model can be implemented to real-time spam detection applications such as web-based spam filters or email clients to offer real-time message monitoring and spam message detection. The model's potential to resist class imbalance and learn to detect new spam patterns is particularly beneficial for dynamic environments. In practice, this would enhance email security and improve the communication experience of users.

G. Perbandingan Metode dan Hasil Terbaik Studi Ini dengan Penelitian Lain

Table VI examines the accuracy of spam detection models employing different techniques and algorithms from different studies.

TABLE VI
COMPARISON OF ACCURACY IN SPAM DETECTION MODELS WITH DIFFERENT METHODS AND ALGORITHMS

Study Name	Method Used	Algorithm Used	Accuracy (%)
[27]	SMOTE	XGBoost	96.20
[28]	SMOTE, ROS	Naïve Bayes	95.63 ; 95.74
[29]	BERT	Logistic Regression	95.95
[30]	SMOTE	Ensemble Extra Tree	92.10
[8]	Random Oversampling	Random Forest	97
This Study	Class Weight	SVM	97.10

As per the findings, application of SMOTE (Synthetic Minority Oversampling Technique) with XGBoost achieved an accuracy of 96.20%, whereas the combination of SMOTE and Random Oversampling (ROS) with Naive Bayes attained an accuracy of 95.63% to 95.74%. The accuracy achieved by the BERT technique + Logistic Regression was 95.95%. In contrast, the application of SMOTE with Ensemble Extra Tree was only able to achieve an accuracy of 92.10%. Random Oversampling with Random Forest recorded an accuracy of 97%, which is the highest from that study. Using Class Weight technique for the SVM algorithm, this study achieved an accuracy of 97.10%, which is a slight improvement from the prior model. This shows that although different methods and algorithms can yield similar outcomes, the SVM with Class Weight model implemented in this study achieves a superior outcome, marginally outperforming rival systems in the accuracy of spam detection.

IV. CONCLUSION

This paper illustrates the benefits of using blended datasets from multiple sources to enhance predictive accuracy of spam email classification models. The model improves its generalization to unseen data and adapts to varying patterns and noise through blended datasets. With more heterogenous datasets, the model improves its spam email classification accuracy through expansion of the model feature set. Moreover, the technique called Class Imbalance with Class Weight, wherein class weights are adjusted to mitigate the class imbalance problem commonly seen in spam datasets,

was effective. By increasing the spam email class weight, the model becomes more sensitive to false negatives, which benefits the spam detection capabilities of the model.

According to the accuracy reached after the best combination of the hyperparameters was obtained through the use of the GridSearchCV, the best performing SVM model reached an accuracy of 97.10% and a Recall of 91.08% which implies this model was able to detect spam with a greater degree of proficiency. Random Forest also improved in Recall with a value of 91.69% and an accuracy of 96.86%. MNB and XGBoost also showed stable performance with MNB achieving a high value of 90.52% in the F1 Score while XGBoost obtained Recall with a value of 91.38%. These results show that the classification models can achieve greater accuracy and sensitivity in spam email detection when a combination of different datasets and the correct class balancing methods are used. Furthermore, despite the promising results obtained from dataset merging and class weight adjustment, there are several important limitations in this study. First, while the datasets used in this research offer a broad range of spam messages, they may not fully represent the diversity and evolution of real-world spam tactics. As spam evolves rapidly, the model's generalization ability could decrease when exposed to new types of spam that were not included in the training data. Additionally, this study focused solely on traditional machine learning models, while deep learning models like LSTM (Long Short-Term Memory) and RNNs (Recurrent Neural Networks) could potentially yield better performance on more complex data, such as sequences of words in spam emails.

REFERENCES

[1] S. M. M. Rahman, A. H. Sarower, and T. Bhuiyan, "Detection and Classification of Spam Email: A Machine Learning-Based Experimental Analysis," in *Proceedings of Trends in Electronics and Health Informatics*, vol. 1034, M. Mahmud, M. S. Kaiser, A. Bandyopadhyay, K. Ray, and S. Al Mamun, Eds., in Lecture Notes in Networks and Systems, vol. 1034. , Singapore: Springer Nature Singapore, 2025, pp. 241–260. doi: 10.1007/978-981-97-3937-0_17.

[2] G. Nasreen, M. Murad Khan, M. Younus, B. Zafar, and M. Kashif Hanif, "Email spam detection by deep learning models using novel feature selection technique and BERT," Egyptian Informatics Journal, vol. 26, p. 100473, June 2024, doi: 10.1016/j.eij.2024.100473.

[3] L. Á. Redondo-Gutierrez, F. Jáñez-Martino, E. Fidalgo, E. Alegre, V. González-Castro, and R. Alaiz-Rodríguez, "Detecting malware using text documents extracted from spam email through machine learning," in Proceedings of the 22nd ACM Symposium on Document Engineering, San Jose California: ACM, Sept. 2022, pp. 1–4. doi: 10.1145/3558100.3563854.

[4] Y. Guo, Z. Mustafaoglu, and D. Koundal, "Spam Detection Using Bidirectional Transformers and Machine Learning Classifier Algorithms," JCCE, vol. 2, no. 1, pp. 5–9, Apr. 2022, doi: 10.47852/bonviewJCCE2202192.

[5] S. Md. M. Hossain and I. H. Sarker, "Content-based Spam Email Detection Using N-gram Machine Learning Approach," Sept. 14, 2021, MATHEMATICS & COMPUTER SCIENCE. doi: 10.20944/preprints202109.0236.v1.

[6] M. V. Madhavan, S. Pande, P. Umekar, T. Mahore, and D. Kalyankar, "Comparative Analysis of Detection of Email Spam With the Aid of Machine Learning Approaches," IOP Conf. Ser.: Mater. Sci. Eng., vol. 1022, no. 1, p. 012113, Jan. 2021, doi: 10.1088/1757-899X/1022/1/012113.

[7] F. Jáñez-Martino, R. Alaiz-Rodríguez, V. González-Castro, and E. Fidalgo, "Trustworthiness of spam email addresses using machine learning," in Proceedings of the 21st ACM Symposium on Document Engineering, Limerick Ireland: ACM, Aug. 2021, pp. 1–4. doi: 10.1145/3469096.3475060.

[8] M. A. Bouke, A. Abdullah, M. T. Abdullah, S. A. Zaid, H. El Atigh, and S. H. ALshatebi, "A Lightweight Machine Learning-Based Email Spam Detection Model Using Word Frequency Pattern," J. Info. Tech. Comp., vol. 4, no. 1, pp. 15–28, June 2023, doi: 10.48185/jitc.v4i1.653.

[9] T. A. Almeida, J. M. Gómez, and A. Yamakami, "Contributions to the study of SMS Spam Filtering: New Collection and Results".

[10] U. Nuha and C.-H. Lin, Conditional Semi-Supervised Data Augmentation for Spam Message Detection with Low Resource Data. 2024. doi: 10.48550/arXiv.2407.04990.

[11] A. C. Acock, "Working With Missing Values," J of Marriage and Family, vol. 67, no. 4, pp. 1012–1028, Nov. 2005, doi: 10.1111/j.1741-3737.2005.00191.x.

[12] Z. B. Siddique, M. A. Khan, I. U. Din, A. Almogren, I. Mohiuddin, and S. Nazir, "Machine Learning-Based Detection of Spam Emails," Scientific Programming, vol. 2021, pp. 1–11, Dec. 2021, doi: 10.1155/2021/6508784.

[13] S. Sarica and J. Luo, "Stopwords in technical language processing," PLoS ONE, vol. 16, no. 8, p. e0254937, Aug. 2021, doi: 10.1371/journal.pone.0254937.

[14] A. K. Shrivas, A. K. Dewangan, and S. M. Ghosh, "Robust Text Classifier for Classification of Spam E-Mail Documents with Feature Selection Technique," ISI, vol. 26, no. 5, pp. 437–444, Oct. 2021, doi: 10.18280/isi.260502.

[15] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries".

[16] V. R. Joseph, "Optimal ratio for data splitting," Statistical Analysis, vol. 15, no. 4, pp. 531–538, Aug. 2022, doi: 10.1002/sam.11583.

[17] M. Adnan, M. O. Imam, M. F. Javed, and I. Murtza, "Improving spam email classification accuracy using ensemble techniques: a stacking approach," Int. J. Inf. Secur., vol. 23, no. 1, pp. 505–517, Feb. 2024, doi: 10.1007/s10207-023-00756-1.

[18] K. R. M. Fernando and C. P. Tsokos, "Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Confidence Calibration of Deep Neural Networks," IEEE Trans. Neural Netw. Learning Syst., vol. 33, no. 7, pp. 2940–2951, July 2022, doi: 10.1109/TNNLS.2020.3047335.

[19] S. Pudasaini, A. Shakya, S. P. Pandey, P. Paudel, S. Ghimire, and P. Ale, "SMS Spam Detection using Relevance Vector Machine," Procedia Computer Science, vol. 230, pp. 337–346, 2023, doi: 10.1016/j.procs.2023.12.089.

[20] B. Wang and V. Pavlu, "December 8, 2014 based on notes by Andrew Ng.".

[21] M. Nivedha and S. Raja, "Detection of email spam using Natural Language Processing based Random Forest approach," International Journal of Computer Science and Mobile Computing, vol. 11, no. 2, pp. 7–22, 2022.

[22] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[23] M. Abbas, K. A. Memon, A. A. Jamali, S. Memon, and A. Ahmed, "Multinomial Naive Bayes Classification Model for Sentiment Analysis".

[24] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco California USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[25] C. Dewi, F. A. Indriawan, and H. J. Christanto, "Spam classification problems using support vector machine and grid search," Int. J. Appl. Sci. Eng., vol. 20, no. 4, pp. 1–10, 2023, doi: 10.6703/IJASE.202312_20(4).006.

[26] D. Chicco, N. Tötsch, and G. Jurman, "The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy,

bookmaker informedness, and markedness in two-class confusion matrix evaluation," *BioData Mining*, vol. 14, no. 1, p. 13, Feb. 2021, doi: 10.1186/s13040-021-00244-z.

[27] T. A. Assegie, "Evaluation of Supervised Learning Models for Automatic Spam Email Detection," July 27, 2023, In Review. doi: 10.21203/rs.3.rs-3191190/v1.

[28] Rivaldo Jeffmarvin, Hafizh Dzaky, Yusup Ardiyanto, Apriliyanto Dwi Saputra, Deri Irawan, and Jason Bernard Ardianto, "Analisis Perbandingan: SMOTE dan Undersampling pada Klasifikasi Spam Naïve Bayes: Studi Eksperimen perbandingan pada Dataset Email Berbahasa Indonesia," JIITE, vol. 2, no. 2, pp. 377–383, Aug. 2025, doi: 10.63547/jiite.v2i2.92.

[29] Y. Guo, Z. Mustafaoglu, and D. Koundal, "Spam Detection Using Bidirectional Transformers and Machine Learning Classifier Algorithms," JCCE, vol. 2, no. 1, pp. 5–9, Apr. 2022, doi: 10.47852/bonviewJCCE2202192.

[30] Prachi Bhatnagar and Dr. S. D. Degadwala, "Efficient Email Spam Classification with N-gram Features and Ensemble Learning," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol, vol. 10, no. 2, pp. 278–284, Mar. 2024, doi: 10.32628/CSEIT2410220.