Spam Email Detection Using Gradient Boosting, AdaBoost, and SVM with Countvectorize Text Representation

**Abstract** Spam detection in email communication has become a critical area of research due to the increasing volume of unsolicited messages. This paper explores a machine learning-based approach to classify emails as spam or not spam using three classification techniques: Gradient Boosting, AdaBoost, and Support Vector Machines (SVM). We preprocess the email text data and transform it into numerical representations using the Countvectorizer technique. Gradient descent is employed for optimizing model performance. Experimental results demonstrate the effectiveness of these methods and highlight their comparative performance in spam detection tasks.

**1. Introduction**

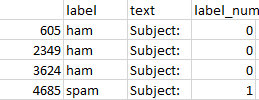
With the evolution of digital communication, email has become a primary medium for personal and professional interactionHowever, spam emails disrupt email communication by reducing security and productivity.Accurate detection of spam emails is therefore a vital task. Traditional rule-based systems for spam detection are increasingly being replaced by machine learning techniques that learn patterns from data and generalize well to unseen samples.

In this study, we focus on three machine learning approaches: Gradient Boosting, AdaBoost, and Support Vector Machines (SVM). Each of these methods has distinct strengths in classification tasks, and we compare their performance in the context of spam detection. Text data from emails is preprocessed and converted into numerical form using the Countvectorizer representation. Gradient descent optimization is employed for training the models.

Furthermore, we review state-of-the-art techniques, including Naive Bayes, which is widely used in text classification tasks due to its simplicity and speed. However, Naive Bayes was not employed in this study because it assumes feature independence, which limits its ability to model complex relationships in the data. Instead, Gradient Boosting, AdaBoost, and SVM were chosen for their ability to capture non-linear patterns and handle high-dimensional feature spaces more effectively.a

**2. Methodology**

**2.1 Dataset and Preprocessing**

We utilized a publicly available on kaggle named „spam\_ham\_dataset” dataset for spam email classification. The dataset contains text messages associated with a number (0/1 for spam/ham) and an unique identifier.. 

The preprocessing steps include:

Text Cleaning: Removal of special characters and numbers, as well as stopwords using the NLTK library.

Tokenization: Splitting text into individual words or tokens.

Stemming and Lemmatization: Reducing words to their root forms to minimize dimensionality, with lemmatization performed using the NLTK library.

**2.2 Text Representation with Countvectorizer**

The cleaned and tokenized text data is transformed into numerical vectors using the CountVectorizer method. CountVectorizer is a method used in natural language processing (NLP) to convert raw text into a numerical representation that can be processed by machine learning algorithms. It works by counting the frequency of each unique word (or token) in a given text corpus and representing this information as a matrix, often referred to as a document-term matrix. This technique counts the frequency of each word in a document, representing the text as a document-term matrix. CountVectorizer is well-suited for this task because it provides a simple and efficient way to represent text data numerically, focusing on raw word occurrences without requiring an in-depth understanding of context.Instead, it provides a simple and fast way to represent text data numerically, which aligns with the requirements of spam detection, where the focus is on distinguishing patterns rather than deep semantic comprehension.

**2.3 Machine Learning Models**

We implemented the following machine learning algorithms:

1. **Gradient Boosting**:

Gradient Boosting is an ensemble method that builds a series of weak learners, typically decision trees, and combines them to form a strong predictive model. Each subsequent model minimizes the error of the previous model through gradient descent.As the base learner in this study was used decision tree with 3 branches.In this paper was used a number of 50 estimators.

For Gradient Boosting, we also recorded the parameters for the 12th estimator:

**Vector of Predictions**:

[0.89792025, 0.39772663,

-0.08611141, 0.31310288, 0.03421035, 0.0246617, 0.06073166, 0.63361465, 0.05045581, 0.13793303]

1. **AdaBoost**:

Adaptive Boosting (AdaBoost) combines weak classifiers iteratively by focusing on misclassified samples, assigning higher weights to these samples in subsequent iterations to improve classification accuracy. As the base learner in this study was used decision tree with 3 branches.In this paper was used a number of 50 estimators.

We recorderded the parameters for the 12th estimator:

**Weighted Error**: 0.3625817445003067

**Alpha**: 0.1128352181850241

**Sample Weights**: [3.24266093e-04, 5.11051471e-04, 6.76118147e-05, 3.51941693e-04, 6.76118147e-05]

**Positions of Selected Samples**: [1281, 2133, 1672, 2295, 1116]

The weighted error reflects the misclassification rate for the weak learner, while alpha determines the contribution of this learner to the final model. Sample weights highlight the emphasis placed on specific samples during the training process, with higher weights indicating more focus on previously misclassified samples. The positions represent the indices of samples that were most relevant in this iteration of training.

1. **Support Vector Machines (SVM)**:

SVM is a robust classifier that finds the optimal hyperplane to separate data points belonging to different classes. I directly use the numbers from the database, where I assign 0 to -1 and leave 1 as it is. I then apply a soft-margin SVM method, which allows for errors during training, using gradient descent. The reason for choosing a soft-margin SVM is that the data is not linearly separable.

Naive Bayes (State-of-the-Art Comparison):

Although not part of the experimental analysis, Naive Bayes is a common baseline for text classification tasks due to its computational efficiency and simplicity. It works on the principle of conditional probability and assumes feature independence. While effective for straightforward tasks, its inability to model complex feature interactions made it unsuitable for this study. The methods chosen (Gradient Boosting, AdaBoost, and SVM) are better suited for tasks where understanding nuanced relationships between features is critical**.**

**2.4 Optimization with Gradient Descent**

Gradient descent was used to optimize the parameters of each model. The learning rate and number of iterations were fine-tuned to achieve optimal performance without overfitting.

**3. Experimental Setup and Results**

**3.1 Experimental Design**

The dataset was divided into training and testing sets using an 75-25 split. Five-fold cross-validation was performed to ensure robustness and prevent overfitting. Model evaluation was based on standard classification metrics, including accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

**3.2 Results and Discussion**

The performance of the three models was evaluated and compared. Table 1 and table 2 summarizes the results:

Table 1

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Precision |
| Gradient Boosting | 0.946636 | 0.950920 |
| AdaBoost | 0.939675 | 0.832168 |
| SVM | 0.972158 | 0.913924 |

Table 2

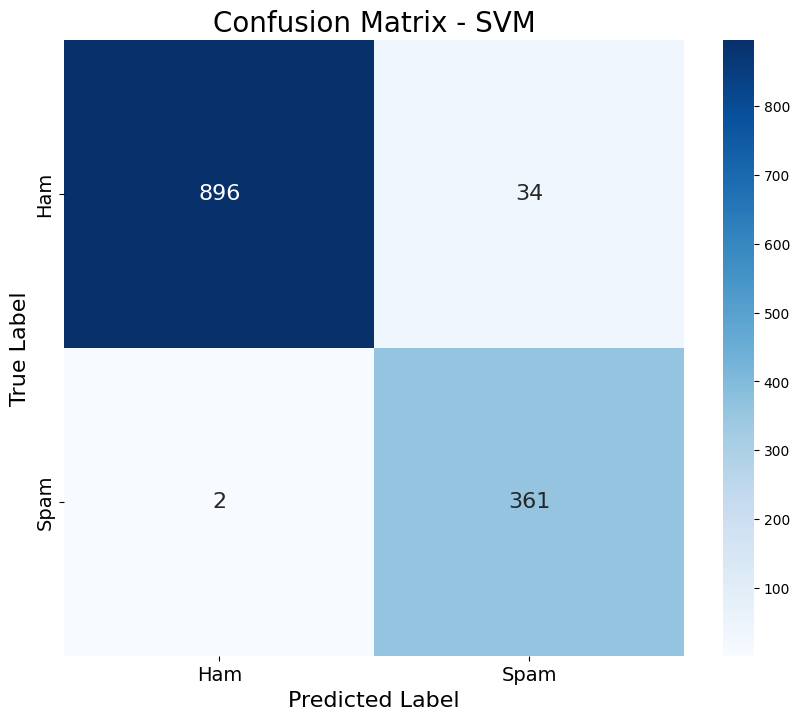
|  |  |  |  |
| --- | --- | --- | --- |
| Model | Recall | F1-Score | AUC-ROC |
| Gradient Boosting | 0.853994 | 0.899855 | 0.918395 |
| AdaBoost | 0.983471 | 0.901515 | 0.953026 |
| SVM | 0.994490 | 0.952507 | 0.978966 |

SVM stands out as the best-performing model, achieving the highest accuracy (97.22%), recall (99.44%), F1-score (95.25%), and AUC-ROC (97.89%). Its high recall indicates it successfully detects nearly all spam emails, while its strong precision reflects a low rate of false positives. This exceptional performance can be attributed to SVM's ability to create an optimal hyperplane for class separation, making it particularly effective for high-dimensional datasets like text data. The use of a soft-margin SVM further enhances its flexibility by allowing small training errors, making it suitable for non-linearly separable data.

Gradient Boosting delivers competitive results, achieving the highest precision (95.09%) among the models, which highlights its effectiveness in minimizing false positives. However, its recall (85.39%) is lower compared to SVM and AdaBoost, which results in fewer spam emails being detected overall.

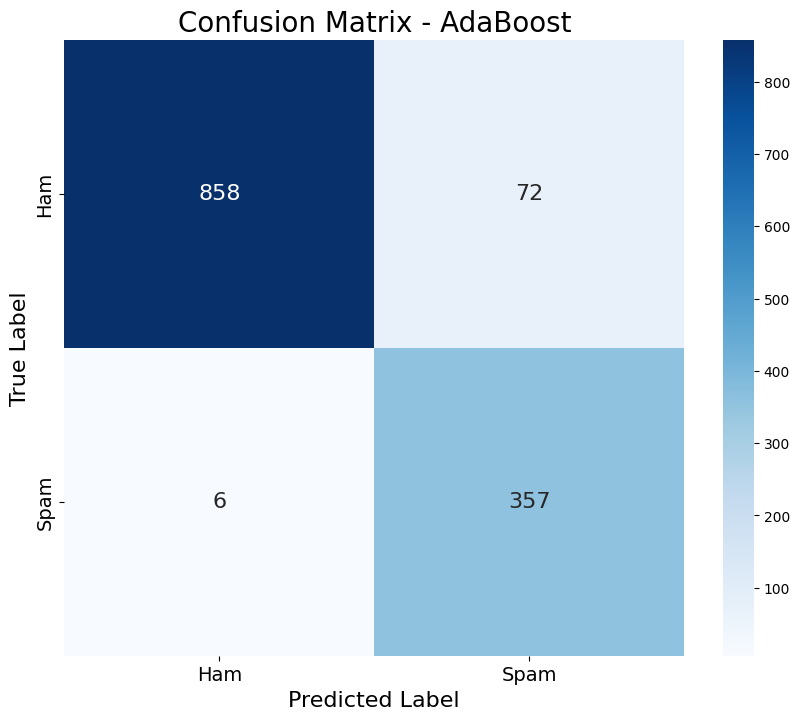
AdaBoost, on the other hand, excels in recall (98.35%), demonstrating its strength in identifying spam emails with very few false negatives. However, it achieves the lowest precision (83.22%) among the three models, indicating a higher likelihood of misclassifying non-spam emails as spam.**4.**

In this study a confusion matrix for each model was generated



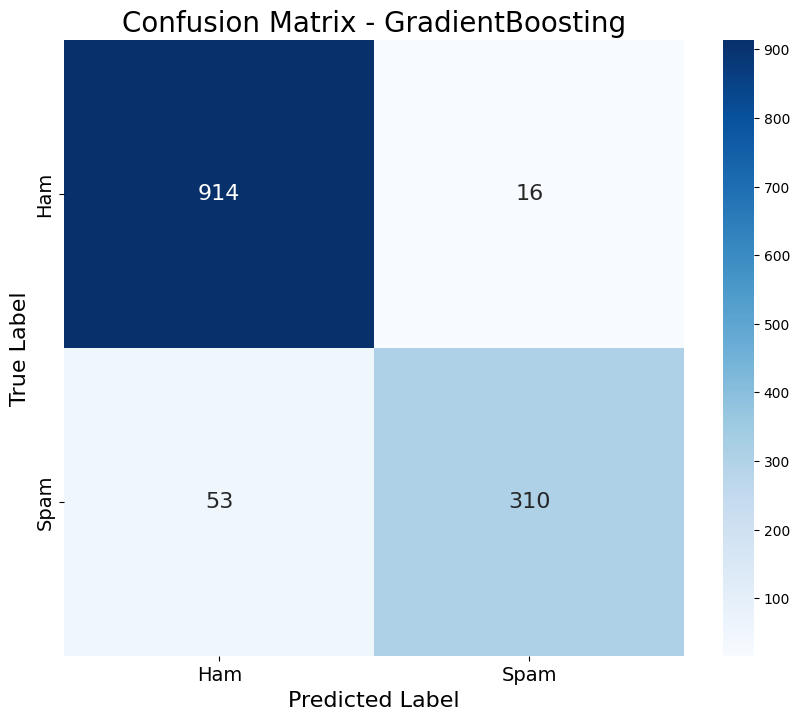
This is a confusion matrix representing the performance of an SVM (Support Vector Machine) classifier in distinguishing between "Ham" (non-spam) and "Spam" messages. We have a number of 896 True Negatives, 34 False Positives only 2 False Negatives and 361 True Positives.

The classifier performs well in identifying "Ham," with very few false positives (34) compared to the total number of true negatives (896).Similarly, it has a high success rate in identifying "Spam," with only 2 false negatives and 361 true positives.The heatmap helps visualize the distribution, with darker colors indicating higher values.



This confusion matrix illustrates the performance of a Adaboost model in classifying messages as either "Ham" (non-spam) or "Spam." The matrix shows that the model correctly classified 858 "Ham" messages as "Ham," but it misclassified 72 "Ham" messages as "Spam." On the other hand, the model successfully identified 357 "Spam" messages as "Spam," with only 6 misclassifications where "Spam" messages were incorrectly predicted as "Ham."

Overall, the model demonstrates good performance, though it appears slightly less accurate than the SVM classifier (based on the comparison). The higher number of false positives (72) in this case indicates the Adaboost model is more prone to misclassifying non-spam messages as spam compared to the SVM. However, it still maintains a strong ability to distinguish between the two classes, with minimal false negatives. The balance between false positives and false negatives can be further evaluated depending on the specific priorities of the classification task.



This confusion matrix illustrates the performance of the Gradient Boosting model in classifying messages as "Ham" (non-spam) or "Spam." The model successfully classified 914 "Ham" messages as "Ham," while only 16 "Ham" messages were incorrectly labeled as "Spam."

For "Spam" messages, the model correctly identified 310 instances, but misclassified 53 as "Ham." This indicates a slight tendency of the model to underestimate "Spam" messages, which could be significant if identifying all spam messages is a priority.

Overall, the Gradient Boosting model demonstrates solid performance, with a strong ability to differentiate between the two classes. However, the relatively higher number of "False Negatives" compared to other models might be a concern, especially in scenarios where spam detection is critical. Fine-tuning the model's parameters or adjusting its decision thresholds could help address this issue.

**Conclusion**

This study demonstrates the effectiveness of machine learning techniques in spam email detection. SVM emerged as the most effective method, followed by AdaBoost and GradientBoosting. The use of Countvectorizer for text representation ensured meaningful feature extraction, while gradient descent optimization enhanced model performance. Future work could explore deep learning approaches and the incorporation of semantic features to further improve classification accuracy.

**References**

1. Freund, Y., & Schapire, R. E. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." Journal of Computer and System Sciences.
2. Bishop, C. M. (2006). "Pattern Recognition and Machine Learning." Springer.
3. Joachims, T. (1998). "Text categorization with Support Vector Machines: Learning with many relevant features." Proceedings of the European Conference on Machine Learning.
4. Chen, T., & Guestrin, C. (2016). "XGBoost: A scalable tree boosting system." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
5. Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*.
6. Schapire, R. E., & Freund, Y. (2012). "Boosting: Foundations and Algorithms." *MIT Press*.
7. Wu, X., & Kumar, V. (2005). "Classification and prediction of financial risk: A machine learning approach." *International Journal of Forecasting*.
8. Sahami, M., Dumais, S. T., Heckerman, D., & Horvitz, E. (1998). "A Bayesian approach to filtering junk e-mail." *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
9. Androutsopoulos, I., Koutsopoulos, A., & Vlahavas, I. (2000). "Boosting algorithms for spam filtering." *Proceedings of the 10th European Conference on Machine Learning*.
10. Bottou, L. (2010). "Large-scale machine learning with stochastic gradient descent." *Proceedings of COMPSTAT'2010*.
11. Ruder, S. (2016). "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747*.
12. Manning, C. D., Raghavan, P., & Schütze, H. (2008). "Introduction to Information Retrieval." *Cambridge University Press*.
13. Jurafsky, D., & Martin, J. H. (2020). "Speech and Language Processing." *Pearson*.
14. Kingma, D. P., & Ba, J. (2014). "Adam: A method for stochastic optimization." *Proceedings of the 3rd International Conference on Learning Representations*.
15. Nocedal, J., & Wright, S. J. (2006). "Numerical Optimization." *Springer*.
16. Berrar, D. (2019). "Cross-validation." *Encyclopedia of Bioinformatics and Computational Biology*.
17. Sokolova, M., & Lapalme, G. (2009). "A systematic analysis of performance measures for classification tasks." *Information Processing & Management*.
18. Dietterich, T. G. (1998). "Approximate statistical tests for comparing supervised classification learning algorithms." *Neural Computation*.