Enhancing SPAM Email Classification

Integration of Sentiment Analysis into a Spam Classifier Based on SVM and Naive Bayes

Diana Grecu, Lauria Rubega, Miriam Espinosa Solana, Dan Loznean,
Horia Ionescu, Vasile Mereuţă, Yinchu Wang

Department of Advanced Computing Sciences
Faculty of Science and Engineering, Maastricht University
Maastricht, The Netherlands

Group 17 - Project 2.2

June 27, 2024

Abstract

Email communication is crucial in the modern digital world, yet the spread of spam emails continues to be a major problem, requiring efficient spam detection systems. Despite their importance, these systems often face challenges such as evolving spam tactics, which make it difficult to keep detection algorithms up to date. Additionally, there can be a high rate of false positives where legitimate emails are incorrectly marked as spam, leading to user frustration and the potential loss of important information. The goal of this research is to determine if emails are spam or not by using Natural Language Processing (NLP) techniques. Specifically, we use Support Vector Machine (SVM) and Naive Bayes classifiers to detect spam emails. These classifiers use statistical characteristics and linguistic patterns to process and analyze email textual content to make a very accurate distinction between spam and authentic emails. In order to improve user engagement and provide a more user-friendly platform for the visualization of categorization results, the project also includes a Graphical User Interface (GUI). We assess our models' performance on a benchmark dataset to show their effectiveness in identifying spam. This report details the methodologies employed, the implementation process, and the results obtained, highlighting the potential of NLP techniques in enhancing email security.

Contents

1 Introduction

Email classification remains a critical challenge in modern information systems, particularly in managing the influx of spam emails that inundate user inboxes on a daily basis. While traditional approaches to spam detection have proven effective to some extent, they often struggle to adapt to the evolving tactics employed by spammers[1]. The current spam detection algorithms are effective, but they could be further improved by integrating sentiment analysis to assess its influence on existing email spam detection techniques. Sentiment analysis, which involves evaluating the emotional tone and subjective information within text, is generally used to gauge public opinion, monitor brand reputation, and analyze customer feedback.

The integration of sentiment analysis introduces a layer of complexity and nuance, offering promising avenues for improving the accuracy and reliability of email classification systems. Our research project seeks to bridge this gap by exploring the efficacy of machine learning techniques, specifically focusing on the comparison between Support Vector Machine (SVM) and Naive Bayes classifiers in classifying spam emails[2]. While SVMs have demonstrated robust performance in various classification tasks, including text classification, Naive Bayes classifiers offer simplicity and efficiency, making them suitable candidates for spam detection in resource-constrained environments.

Furthermore, we aim to investigate the potential of sentiment analysis as a complementary approach to augment the capabilities of our classification models[3]. By analyzing the emotional content of emails, sentiment analysis has been shown to provide valuable insights that can help distinguish between legitimate and spam messages[4]. Our hypotheses are as follows:

1. Integrating sentiment analysis into our classification framework might lead to improved performance.

2. The enhanced framework will better adapt to the dynamic nature of spam campaigns.

By empirically evaluating the performance of SVM and Naive Bayes classifiers and investigating the impact of sentiment analysis on classification accuracy, we aim to contribute to the development of more robust and adaptable email classification systems[5]

capable of mitigating the pervasive threat of spam emails.

1.1 Research Questions

Our research endeavors to address two fundamental elements concerning the utilization of machine learning algorithms for spam email detection:

1. How does the performance of Support Vector Machine (SVM) in classifying spam emails compare to that of Naive Bayes?

The study compares the performance of Support Vector Machine (SVM) and Naive Bayes in classifying spam emails using a labeled email dataset to identify their strengths and weaknesses.

2. To what extent does sentiment analysis improve the performance of our models?

We will investigate the potential of sentiment analysis for improving spam detection models by incorporating it alongside traditional classification methods to enhance the accuracy of email content identification.

Addressing these research questions will provide valuable insights into developing more robust and adaptive spam detection systems. The implications of this research include improved email security and a reduction in the number of false positives and negatives, ultimately leading to a safer and more reliable email communication environment.

2 Methodologies

This section outlines the methodologies employed for the email spam classification task, detailing the data collection and preprocessing steps, as well as the training and testing procedures for both Naive Bayes and Support Vector Machine (SVM) algorithms.

2.1 Data collection and pre-processing

For the **data collection** part, the dataset for our email spam classification task was sourced from the UCI Machine Learning Repository, specifically the **Spambase** dataset[1]. It consists of 4,601 instances, with each instance containing 57 attributes representing different characteristics of the emails. These attributes encompass a diverse range of features, including word frequencies, character frequencies, and other metadata associated with the email content. This dataset comprises a collection of email samples, each labeled as either spam or non-spam, along with various features[6] extracted from the email content.

For further testing, in order to ensure our classifiers perform well on a wide range of emails, we have collected 4 other datasets, namely :

- 'Enron Email Dataset', `https://www.cs.cmu.edu/~./enron/`
- 'SMS Spam Collection Dataset', `https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset`
- 'Spam Emails Dataset', `https://www.kaggle.com/datasets/abdallahwagih/spam-emails`
- 'Spam or Not Spam Dataset', `https://www.kaggle.com/datasets/ozlerhakan/spam-or-not-spam-dataset`

Prior to model training, the datasets underwent thorough **preprocessing** to prepare them for the machine learning algorithms. This preprocessing pipeline encompassed several key steps.

Firstly, **data cleaning**, means that any missing or erroneous values within the dataset were addressed through imputation or removal, ensuring the integrity and consistency of the data.

[1] `https://archive.ics.uci.edu/dataset/94/spambase`

Afterwards, we have proceeded with **feature selection** techniques, which were employed to identify and retain the most relevant attributes for spam classification. This step aimed to reduce dimensionality and mitigate the curse of dimensionality, thus enhancing the efficiency and effectiveness of the learning algorithms.

Importantly, the main dataset, the **Spambase** dataset, has already undergone some preprocessing steps. It has 57 attributes, including both numerical and categorical features. These attributes represent various characteristics of the emails, such as word frequencies, character frequencies, and other metadata. Of course, each instance in the dataset is labeled as either spam (1) or non-spam (0), indicating whether the email is classified as spam or ham.

2.2 Feature extraction and Classification pipeline

This section details our approach to preprocessing, feature extraction, and classification of emails into spam and non-spam categories. The initial step involves standardizing the raw email data sourced from multiple datasets, ensuring uniformity across varied formats. Each email undergoes preprocessing to enhance the quality of the dataset for subsequent analysis. This includes converting text to lowercase, removing HTML tags, URLs, email addresses, punctuation, and numerical digits. Tokenization breaks down text into individual words, followed by removal of stopwords and stemming using the Porter stemming algorithm. Additionally, sentiment analysis using the VADER Sentiment Intensity Analyzer provides further insights into the emotional tone of the messages, capturing sentiment metrics such as negativity, neutrality, positivity, and compound sentiment scores.
Features are extracted directly from the email texts to augment the dataset's richness and enable effective classification. Key features extracted include word frequencies, character frequencies, and capital run lengths. These features are computed using customized word lists and character sets tailored to distinguish between spam and legitimate emails. These techniques can be found in the 'presentiment.py' file.

2.3 Naive Bayes Classifier

Naive Bayes is a popular algorithm in Natural Language Processing (NLP) due to its simplicity and effectiveness, particularly in text classification tasks such as spam detection. It is based on Bayes' theorem, providing a probabilistic framework for predicting the likelihood of a given input belonging to a particular class. In this project, we employ the Multinomial Naive Bayes classifier for the task of classifying emails based on the extracted features. This variant of Naive Bayes is suitable for our dataset's characteristics, assuming a multinomial distribution for discrete features. The dataset is split into training and testing sets (70% for training and 30% for testing) to ensure the model is evaluated on unseen data.

2.3.1 Naive Bayes Training

During the training phase, we systematically optimize the performance of our Multinomial Naive Bayes classifier. Beginning with the preprocessed dataset obtained from the UCI Machine Learning Repository, we apply preprocessing steps to handle missing values and map categorical labels. Subsequently, the dataset is split into training and testing sets, with a 30% test size for evaluation.

A TfidfVectorizer is utilized to transform the textual data into numerical features. The transformed features are then used to train the Multinomial Naive Bayes model. Finally, the trained Multinomial Naive Bayes model, along with the TfidfVectorizer, is saved to a file for future use, ensuring reproducibility and accessibility for subsequent predictions and evaluations.

2.3.2 Naive Bayes Evaluation

After training, we evaluated the model's performance on both the training and testing sets. The key performance metrics used included accuracy, precision, recall, F1 score, and confusion matrix.

The evaluation demonstrated the model's ability to accurately classify emails, with specific metrics offering insights into its strengths and potential areas for improvement.

2.3.3 Tuned Naive Bayes Classifier

To enhance the model's performance, we developed a tuned version of the Naive Bayes classifier, incorporating several advanced techniques:

- **Feature Selection**: The TfidfVectorizer was configured to select the top 5000 features, reducing dimensionality and focusing on the most informative features.
- **Hyperparameter Tuning**: GridSearchCV was used for hyperparameter tuning, specifically adjusting the `alpha` parameter.

These enhancements led to improved accuracy and robustness of the classifier.

2.3.4 Implementation Details

The following steps outline the implementation details:

1. **Data Loading and Preprocessing**: The processed dataset is loaded, and any NaN values in the 'Message' column are filled. Labels are mapped to numerical values.
2. **Data Splitting**: The dataset is split into training and testing sets (70% training, 30% testing).
3. **Feature Extraction**: A TfidfVectorizer is used to transform the text data into numerical features, limited to the top 5000 features.
4. **Model Training**: A Multinomial Naive Bayes classifier is initialized and trained using GridSearchCV to find the best `alpha` parameter.
5. **Model Evaluation**: Performance metrics are calculated for both the training and testing sets, including accuracy, precision, recall, F1 score, and confusion matrix.
6. **Model Saving**: The trained model and TfidfVectorizer are saved to disk for future use.

The final model and its vectorizer are stored, enabling efficient loading and prediction on new data. The trained Multinomial Naive Bayes classifier demonstrated significant improvements in classification accuracy and other performance metrics, validating the effectiveness of the tuning process.

2.4 Support Vector Machines Classifier

Support Vector Machines (SVMs) are a robust supervised learning model[7] well-suited for the email spam classification task. This report outlines the implementation of an SVM for email spam classification using the Spambase dataset, detailing the data preparation, model training, evaluation, and model persistence steps.

The Spambase dataset contains 57 features derived from email content and a binary label indicating whether an email is spam (1) or non-spam (0). The dataset was loaded from a CSV file, with each feature corresponding to characteristics such as word frequencies or the presence of specific keywords. The features (X) and the target variable (y) were then separated. The dataset was split into training and testing sets, with 0.7 of the data allocated for training and 0.3 for testing. This split ensures the model's performance can be assessed on unseen data, promoting generalization.

An SVM with a linear kernel[8] was chosen for this task. The linear kernel is effective for high-dimensional data and is computationally efficient.

The SVM aims to find the optimal hyperplane that separates the spam and non-spam emails with the maximum margin, defined as the distance between the hyperplane and the nearest data points from both classes, known as support vectors.

2.4.1 SVM Training

In the training phase of our Support Vector Machine (SVM) classifier for email spam detection, we aimed to optimize model performance and achieve robust classification results through careful and detailed procedures. Initially, we loaded the preprocessed dataset, which was split into features and target variables, representing the attributes and labels of the email samples, respectively. We trained the classifier utilizing a test size of 30% and a random seed for reproducibility.

2.4.2 SVM Evaluation

After training, the model's performance was evaluated on both the training and testing sets using several metrics: accuracy, precision, recall, F1 score, and confusion matrix. These metrics provide a comprehensive understanding of the model's effectiveness and its ability to generalize to new data.

To facilitate future use without retraining, the trained model was saved using the pickle module. This step ensures that the model can be efficiently reloaded and utilized for predictions or further evaluation. A function was provided to load the model from the saved file, enabling seamless reuse of the trained model.

2.5 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a crucial technique in NLP used to convert textual data into numerical feature vectors. The goal is to reflect the importance of words in a document relative to a collection of documents, referred to as a corpus. This transformation enabled the text data to be in a format suitable for machine learning algorithms and therefore helped in identifying key terms that are significant in distinguishing between spam and non-spam emails. Moreover, it balanced the frequency of terms with their importance, offering a more informative representation of the text data. This way, it reduces the impact of common but less informative words, focusing on unique terms that are more indicative of the content, thus aiding in effective spam detection.
The formula for TF-IDF is given by:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

where:

$$\text{TF}(t, d) = \frac{f_{t,d}}{n_d}$$

$$\text{IDF}(t) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

- $f_{t,d}$ is the number of times term t appears in document d.
- n_d is the total number of terms in document d.
- N is the total number of documents in the corpus.
- $|\{d \in D : t \in d\}|$ is the number of documents containing term t.

The ability to further integrate TF-IDF with Sentiment Analysis was another factor in its adoption. This will be discussed in more detail in the following part.

2.6 Sentiment Analysis

Sentiment Analysis involves determining the sentiment expressed in a piece of text. For our project, we employed the VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis tool, which is particularly effective for analyzing social media texts and informal communication, though it also

performs well for formal communication. VADER provides sentiment scores for negative, neutral, positive, and compound (overall sentiment).

Sentiment scores provide additional features that might be indicative of spam. For example, spam emails might use more extreme sentiments to entice or alarm the reader. Put another way, adding sentiment features could improve the model's classification performance, but it **should be noted** that a text's sentiment influence doesn't always add up to much and could potentially make the model more confused, thus worsening the model's test metrics.

We computed sentiment scores for each email using VADER and the sentiment scores included:

- `sentiment_neg`: Negative sentiment score
- `sentiment_neu`: Neutral sentiment score
- `sentiment_pos`: Positive sentiment score
- `sentiment_compound`: Compound sentiment score (a normalized score between -1 and 1)

Integration with TF-IDF
After computing the TF-IDF vectors for the text data, we integrated these vectors with the sentiment scores to create a comprehensive feature set. Now, the sentiment scores are treated as additional features alongside the TF-IDF vectors. Each email's TF-IDF vector represents the importance of terms in that email. By integrating sentiment scores into this vector, we expand the feature set to include emotional context. We ensured that all sentiment scores were non-negative by shifting them (adding a constant value) to make them compatible with algorithms like Naive Bayes that do not handle negative values well.

3 Experiments

The experiment conducted focuses on evaluating and comparing the performance Support Vector Machine (SVM) and Naive Bayes classifiers in identifying spam emails. The following steps were undertaken to conduct these experiments:

Firstly we collected various datasets containing a significant amount of emails that were labeled either as spam or as non-spam. In total, we collected five datasets on which we trained and tested our program. Initially we used the 'Spambase dataset' for the data collection step of our implementation and later on, for further testing we have collected four additional datasets: 'Enron Email Dataset', 'SMS Spam Collection Dataset', 'Spam Emails Dataset', 'Spam or Not Spam Dataset'.

Once the datasets were collected, we extracted the relevant features from the emails by preprocessing them.

Both the Naive Bayes and Support Vector Machine classifiers were trained using the already collected dataset. The models were then evaluated based on several metrics, including accuracy, precision, recall, F1 score, and confusion matrix. Hyperparameter tuning for the Naive Bayes classifier was performed using GridSearchCV to optimize performance, while the Support Vector Machine used a linear kernel for the training of the model.

Moreover, sentiment analysis was implemented and deployed to find out whether it would stay the same, improve, or worsen our model's performance.

Lastly, the performance of the two classifiers was analyzed and compared before and after the implementation of sentiment analysis to determine whether there is a significant difference in the performance of SVM and Naive Bayes in their ability to classify spam emails.

4 Results

The comparison between the Naive Bayes and the Support Vector Machine classifiers, as well as the test metrics[9] of each of the classifiers, were conducted using multiple metrics such as accuracy, precision, recall, and F1 score across four datasets. Please re-

fer to the **Appendix** in order to see all testing and training metrics, together with the confusion matrices.

4.1 Test Metrics for the Naive Bayes classifier

In this section, we present the performance results of the Naive Bayes classifier.

Test Set Metrics (Without Sentiment Analysis)

The Naive Bayes classifier demonstrated strong performance in identifying spam and non-spam emails, with an accuracy rate of 98%, precision of 97%, recall of 99%, and F1 score of 98%.

Training Set Metrics (Without Sentiment Analysis)

The classifier consistently performs well on the training set, achieving high accuracy, precision, recall, and F1 score metrics, and effectively classifies spam and non-spam instances.

Test Set Metrics with Sentiment Analysis

Sentiment analysis improved classification performance by maintaining accuracy and F1 score of 98%, with minimal deviation in precision and recall scores.

Training Set Metrics with Sentiment Analysis

The classifier demonstrated high performance metrics in sentiment analysis, demonstrating its robust learning and adaptation capabilities across various feature sets.

Performance on Specific Datasets
Dataset 1 without sentiment analysis showed lower accuracy (36%) and precision (16%), while Dataset 2 performed better with an accuracy of 40% and higher precision (21%) and recall (98%).

4.2 Test Metrics for the Support Vector Machine classifier

In this section, we present the performance results of the Support Vector Machine classifier.

Test Set Metrics (Without Sentiment Analysis)

The Support Vector Machine (SVM) classifier demonstrated strong performance in identifying spam and non-spam emails, with an accuracy rate of 98%, precision of 97%, recall of 99%, and F1 score of 98%.

Training Set Metrics (Without Sentiment Analysis)

The SVM classifier consistently performs well on the training set, achieving high accuracy, precision, recall, and F1 score metrics, and effectively classifies spam and non-spam instances.

Test Set Metrics with Sentiment Analysis

Sentiment analysis improved classification performance by maintaining accuracy and F1 score of 98%, with minimal deviation in precision and recall scores.

Training Set Metrics with Sentiment Analysis

The SVM classifier demonstrated high performance metrics in sentiment analysis, demonstrating its robust learning and adaptation capabilities across various feature sets.

Performance on Specific Datasets
Dataset 1 without sentiment analysis showed lower accuracy (23%) and precision (14%), while Dataset 2 performed better with an accuracy of 72% and higher precision (37%) and recall (89%).

4.3 Comparison between the Naive Bayes and the Support Vector Machine classifiers

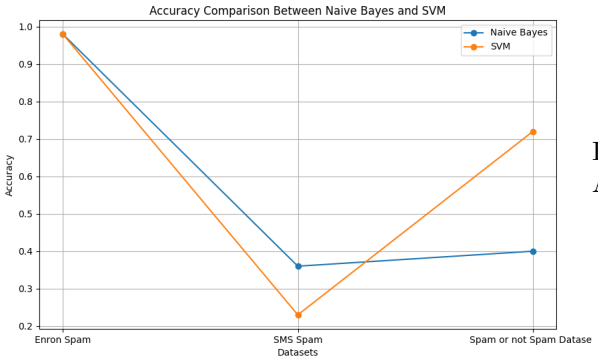

Figure 1: Accuracy Comparison before Sentiment Analysis between Naive Bayes and SVM

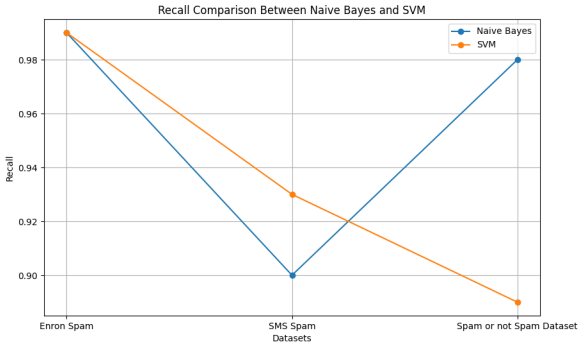

Figure 2: Precision Comparison before Sentiment Analysis between Naive Bayes and SVM

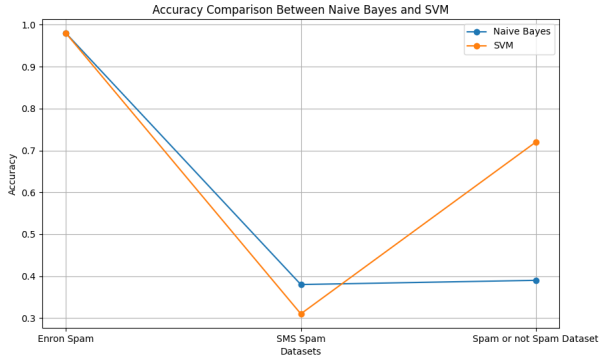

Figure 3: Accuracy Comparison after Sentiment Analysis between Naive Bayes and SVM

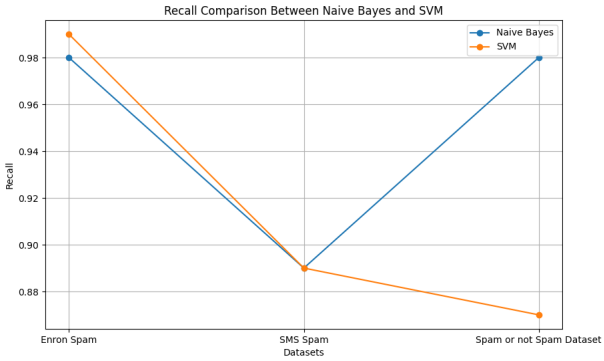

Figure 4: Recall Comparison after Sentiment Analysis between Naive Bayes and SVM

5 Discussion

The comparative analysis between the Naive Bayes and Support Vector Machine (SVM) classifiers highlights significant findings regarding their effectiveness in spam email detection, both with and without sentiment analysis.

5.1 Performance of Naive Bayes Classifier

The Naive Bayes classifier exhibited strong performance on both the training and test sets without sentiment analysis, achieving an impressive accuracy of 98%, with precision, recall, and F1 scores all at 98%. These metrics indicate that the Naive Bayes classifier is highly effective at distinguishing between spam and non-spam emails.

With sentiment analysis, the Naive Bayes classifier maintained similar performance metrics, demonstrating its adaptability to different feature sets. However, performance on specific datasets without sentiment analysis revealed some variability. Dataset 1 showed a lower accuracy of 36% and precision of 16%, while Dataset 2 performed better with an accuracy of 40%, precision of 21%, and recall of 98%.

5.2 Performance of Support Vector Machine Classifier

The SVM classifier also demonstrated excellent performance. Without sentiment analysis, it achieved an accuracy of 98% on the test set, with precision, recall, and F1 scores closely matching those of the Naive Bayes classifier. Incorporating sentiment analysis, the SVM maintained high performance levels, indicating its robustness and adaptability.

The SVM's performance on specific datasets without sentiment analysis showed notable variability. Dataset 1 had an accuracy of 23% and precision of 14%, while Dataset 2 showed much better results with an accuracy of 72%, precision of 37%, and recall of 89%.

5.3 Comparison and Insights

Both classifiers exhibited strong performance metrics, with the SVM slightly outperforming the Naive Bayes classifier in certain areas. The inclusion of sentiment analysis did not substantially affect performance, suggesting robustness to changes in feature extraction methods.

However, the variability in performance across specific datasets indicates that classifier effectiveness can be dataset-dependent. These findings highlight the importance of considering dataset characteristics when selecting and fine-tuning classifiers for spam detection.

In conclusion, both Naive Bayes and SVM classifiers are effective tools for spam detection, with robustness to sentiment analysis (see figure 5). Future work could explore more sophisticated feature extraction techniques or hybrid models to achieve more consistent performance across diverse datasets.

6 Conclusion

In conclusion, we created a system that detects spam emails by implementing two classifiers, the Naive Bayes classifier and the Support Vector Machine classifier, to determine which classifier can better identify spam and non-spam emails across different datasets. Sentiment analysis was also implemented for both models to provide a richer representation of each email. It not only captures the lexical importance of terms but also the emotional context conveyed by those terms. We conducted an experiment to determine whether there is a significant difference between the classifiers before and after sentiment analysis on multiple metrics such as accuracy, precision, recall and F1 score. Additionally, we developed a graphical user interface (GUI) to allow users to input their own emails and see whether our system classifies it as spam or non-spam, depending on the classifier selected.

The experimental results suggest that the Support Vector Machine classifier generally performs better than the Naive Bayes classifier in spam email detection. With sentiment analysis implemented, we have seen that it didn't necessarily improve the performance of our models, denoting that it might not be a useful tool in the spam classification task.

References

[1] M. Diale, T. Celik, and C. Van Der Walt. Unsupervised feature learning for spam email filtering. *Computers and Electrical Engineering*, 74:89–104, 2019.

[2] Tejaswini Gangavarapu, C.D. Jaidhar, and Bharathi Chanduka. Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*, 53:5019–5081, 2020.

[3] Khalid Iqbal and Muhammad Shehrayar Khan. Email classification analysis using machine learning techniques. *Applied Computing and Informatics*, ahead-of-print(ahead-of-print), 2022. Y2 - 2024/05/27.

[4] Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access*, 10:65703–65727, 2022.

[5] Melvin Diale, Turgay Celik, and Christiaan Van Der Walt. Unsupervised feature learning for spam email filtering. *Computers Electrical Engineering*, 74:89–104, 2019.

[6] Marcelo G. Armentano and Analía A. Amandi. Enhancing the experience of users regarding the email classification task using labels. *Knowledge-Based Systems*, 71:227–237, 2014.

[7] Teng Lv, Ping Yan, Hongwu Yuan, and Weimin He. Spam filter based on naive bayesian classifier. *Journal of Physics: Conference Series*, 1575(1):012054, jun 2020.

[8] Wenjuan Li, Weizhi Meng, Zhiyuan Tan, and Yang Xiang. Design of multi-view based email classification for iot systems via semi-supervised learning. *Journal of Network and Computer Applications*, 128:56–63, 2019.

[9] Rakesh Nayak, Salim Amirali Jiwani, and B. Rajitha. Withdrawn: Spam email detection using machine learning algorithm. *Materials Today: Proceedings*, 2021.

Appendix

Model metrics for Naive Bayes Classifier

Test Set Metrics (Without Sentiment Analysis)

- **Accuracy**: 0.98
- **Precision**: 0.97
- **Recall**: 0.99
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 4791 & 154 \\ 69 & 5101 \end{bmatrix}$$

Training Set Metrics (Without Sentiment Analysis)

- **Accuracy**: 0.98
- **Precision**: 0.97
- **Recall**: 0.99
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 11249 & 351 \\ 161 & 11840 \end{bmatrix}$$

Test Set Metrics with Sentiment Analysis

- **Accuracy**: 0.98

- **Precision**: 0.97
- **Recall**: 0.98
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 4794 & 151 \\ 92 & 5078 \end{bmatrix}$$

Training Set Metrics with Sentiment Analysis

- **Accuracy**: 0.98
- **Precision**: 0.97
- **Recall**: 0.98
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 11258 & 342 \\ 226 & 11775 \end{bmatrix}$$

Testing on Specific Datasets

Dataset 1 without sentiment analysis

- **Accuracy**: 0.36
- **Precision**: 0.16
- **Recall**: 0.90
- **F1 Score**: 0.27

Confusion Matrix:

$$\begin{bmatrix} 1348 & 3477 \\ 76 & 671 \end{bmatrix}$$

Dataset 2 without sentiment analysis

- **Accuracy**: 0.40
- **Precision**: 0.21
- **Recall**: 0.98
- **F1 Score**: 0.35

Confusion Matrix:

$$\begin{bmatrix} 706 & 1794 \\ 10 & 490 \end{bmatrix}$$

Dataset 1 with sentiment analysis

- **Accuracy**: 0.38
- **Precision**: 0.17
- **Recall**: 0.89
- **F1 Score**: 0.28

Confusion Matrix:

$$\begin{bmatrix} 1450 & 3375 \\ 80 & 667 \end{bmatrix}$$

Dataset 2 with sentiment analysis

- **Accuracy**: 0.39
- **Precision**: 0.21
- **Recall**: 0.98
- **F1 Score**: 0.35

Confusion Matrix:

$$\begin{bmatrix} 685 & 1815 \\ 11 & 489 \end{bmatrix}$$

Model metrics for SVM Classifier

Test Set Metrics (Without Sentiment Analysis)

- **Accuracy**: 0.98
- **Precision**: 0.97
- **Recall**: 0.99
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 4794 & 151 \\ 36 & 5134 \end{bmatrix}$$

Training Set Metrics (Without Sentiment Analysis)

- **Accuracy**: 0.99
- **Precision**: 0.98
- **Recall**: 1.00
- **F1 Score**: 0.99

Confusion Matrix:

$$\begin{bmatrix} 11411 & 189 \\ 22 & 11979 \end{bmatrix}$$

Test Set Metrics with Sentiment Analysis

- **Accuracy**: 0.98
- **Precision**: 0.98
- **Recall**: 0.99
- **F1 Score**: 0.98

Confusion Matrix:

$$\begin{bmatrix} 4822 & 123 \\ 47 & 5123 \end{bmatrix}$$

Training Set Metrics with Sentiment Analysis

- **Accuracy**: 0.99
- **Precision**: 0.99
- **Recall**: 1.00
- **F1 Score**: 0.99

Confusion Matrix:

$$\begin{bmatrix} 11445 & 155 \\ 22 & 11979 \end{bmatrix}$$

Testing on Specific Datasets

Dataset 1 without sentiment analysis

- **Accuracy**: 0.23
- **Precision**: 0.14
- **Recall**: 0.93
- **F1 Score**: 0.25

Confusion Matrix:

$$\begin{bmatrix} 599 & 4226 \\ 50 & 697 \end{bmatrix}$$

Dataset 2 without sentiment analysis

- **Accuracy**: 0.72
- **Precision**: 0.37
- **Recall**: 0.89
- **F1 Score**: 0.52

Confusion Matrix:

$$\begin{bmatrix} 1729 & 771 \\ 56 & 444 \end{bmatrix}$$

Dataset 1 with sentiment analysis

- **Accuracy**: 0.31
- **Precision**: 0.15
- **Recall**: 0.89
- **F1 Score**: 0.26

Confusion Matrix:

$$\begin{bmatrix} 1060 & 3765 \\ 81 & 666 \end{bmatrix}$$

Dataset 2 with sentiment analysis

- **Accuracy**: 0.72
- **Precision**: 0.36
- **Recall**: 0.87
- **F1 Score**: 0.51

Confusion Matrix:

$$\begin{bmatrix} 1720 & 780 \\ 66 & 434 \end{bmatrix}$$

ROC plot after implementation of Sentiment Analysis

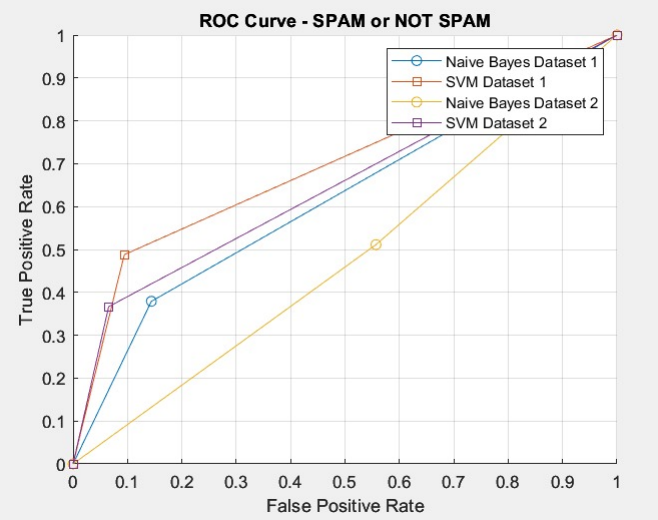

Figure 5: Recall Comparison after Sentiment Analysis between Naive Bayes and SVM