CASE STUDY 3: SPAM EMAIL FILTERING

BY: ANDREEA CAROLINA CRAUS

1 | INTRODUCTION

The purpose of this project is to analyze amd develop a spam filter for the Spam Assassin email dataset provided, which is a collection of emails commonly used for training and testing spam email filters, and employ a decision tree mechanism, mainly Naïve Bayes and Random Forest, to filter out spam emails. The dataset includes the actual content of 9,348 emails, comprising both of spam and legitimate messages, which may include text, HTML, attachments, and other elements that may be typically founds in emails. In addition, the data contains a mix of spam emails that employ various tactics to evade detection, such as misspelled words, obfuscated text, image-based spam, and other techniques. This diversity helps in training models that can handle a wide range of spam types.

A spam filter is a type of classification system designed to automatically identify and segregate incoming emails into two categories: "spam" (unsolicited, often irrelevant or inappropriate messages) and "ham" (legitimate, wanted messages). The primary goal of a spam filter is to accurately and efficiently classify emails to help users manage their inbox and reduce the impact of unwanted or potentially harmful messages. As can be seen in Figure 1, the class distribution for the label “True” (the email is spam) is about almost three times larger proportionally, with 6,951 emails, than for the label “False” (the email is not spam), with 2,397 emails.

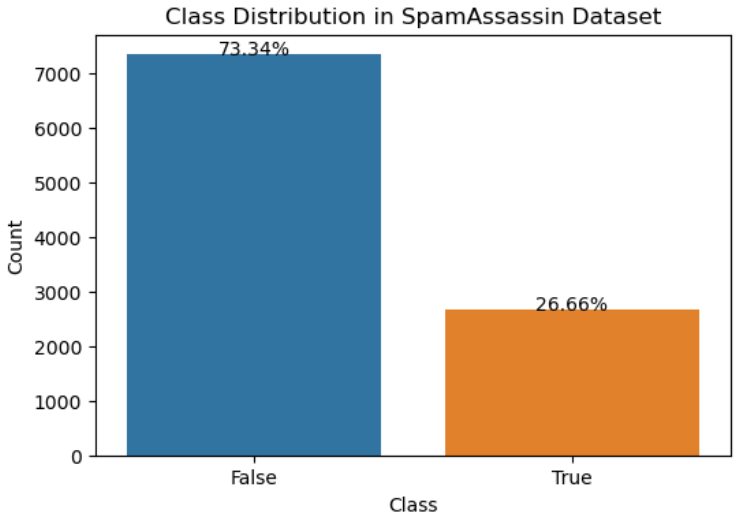

Figure 1

2 | METHODS

2.1 | PREPROCESSING

Preprocessing is crucial when building a spam filter with a decision tree binary classifier. Due to the emails containing diverse and unstructured text, the text cleaning process involves removing the HTML tags, special characters, and non-text elements. In addition, the text is standardized by converting it to lowercase to ensure consistency and reduce the complexity of the data. Random Forest requires numerical input features, therefore, a process tokenization is used to break down the text into individual words, or tokens. In order to transform these tokens into numerical representations, a technique called TF-IDF (Term Frequency-Inverse Document Frequency) is used, which helps identify the words that are important within a specific document but not necessarily common across the entire document collection. Terms with higher TF-IDF scores are considered more discriminative and contribute more to the representation of the document in a feature space.

2.2 | EDA

In Figure 2, two words cloud generated can be seen, used to visualize the prevalence of words in the spam emails compared to the non-spam emails. While these are created without major cleaning of the email content, there is a clear distinction between which words are commonly seen in the spam emails versus the non-spam emails. In addition, in the non-spam emails, there is a wider range of vocabulary with less words being seen as commonly, while the spam emails have some words that have very high occurrences.

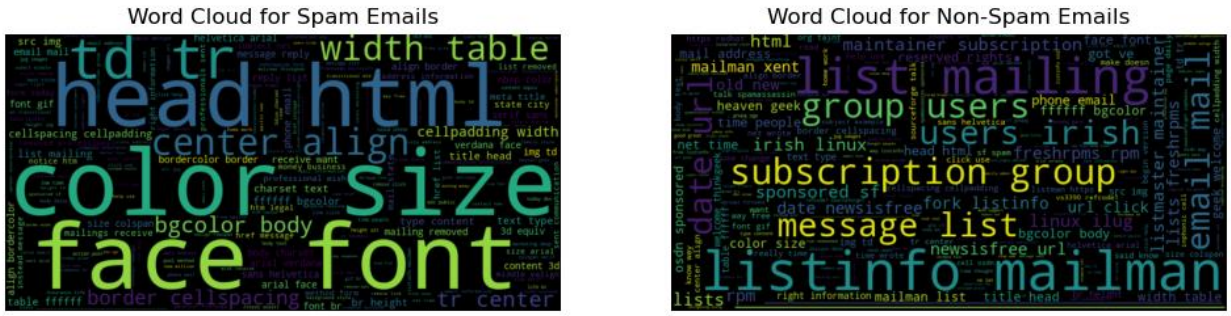

Figure 2

2.3 | Gaussian Naïve Bayes

The Gaussian Naïve Bayes model is a probabilistic classification algorithm based on Bayes theorem and the “Naïve” part comes from the assumption that the features are conditionally independent given the class. This means that knowing the value of one feature doesn’t provide any information about the values of other features, given the class. In the case of Gaussian Naive Bayes, it's assumed that the continuous-valued features within each class follow a Gaussian (normal) distribution. This assumption simplifies the modeling process and allows for the calculation of probabilities based on the mean and variance of each feature

within each class. When making predictions, the model calculates the likelihood of observing the given feature values in each class using the Gaussian probability density function. It then combines these likelihoods with the prior probability of each class to determine the most probable class for the given set of features. The final decision rule involves selecting the class with the highest posterior probability. In binary classification, this often means comparing the ratio of the posterior probabilities for each class.

2.4 | Random Forest

Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve overall predictive performance and robustness. It employs a technique called Bootstrap Aggregating, or Bagging, which creates multiple subsets (bootstrap samples) of the training set by randomly sampling with replacement. Each subet is used to train an individual decision tree. For each decision tree in the forest, a random subset of features is considered at each split point. This introduces diversity among the trees, preventing them from becoming highly correlated and improving the overall performance of the ensemble. The tree is built by recursively partitioning the data based on the chosen features until a certain stopping criterion is met, such as reaching a maximum depth or minimum number of samples per leaf. For classification tasks, each tree in the forest "votes" for a class, and the class with the majority of votes becomes the predicted class. For regression tasks, the predictions from each tree are averaged to obtain the final prediction. The final output of the Random Forest is the aggregation of the individual tree outputs. This ensemble approach often results in more robust and accurate predictions compared to individual decision trees, as it mitigates overfitting and reduces sensitivity to noise in the data.

3 | RESULTS

The final model was created using the Gaussian Naïve Bayes and Random Forrest Classifier library from sklearn. Cross-validation was performed to analyze the performance of the model. This technique assesses the results by systematically partitioning the dataset into multiple subsets, training the model on different combinations of the training and testing sets, then averaging the performance metric. This is crucial to ensure the model's evaluation is less dependent on a specific data split and provides a more reliable estimate of its overall effectiveness. The Naïve Bayes model had a high final average accuracy score of 97%, which measures the overall correctness of predictions. In Figure 3, a confusion matrix can be seen to visualize and summarize the performance of the classification. This tells us that 1,449 of the "True" labels were predicted correctly and 17 were predicted incorrectly and 495 "False" labels were predicted correctly and 38 were predicted incorrectly.

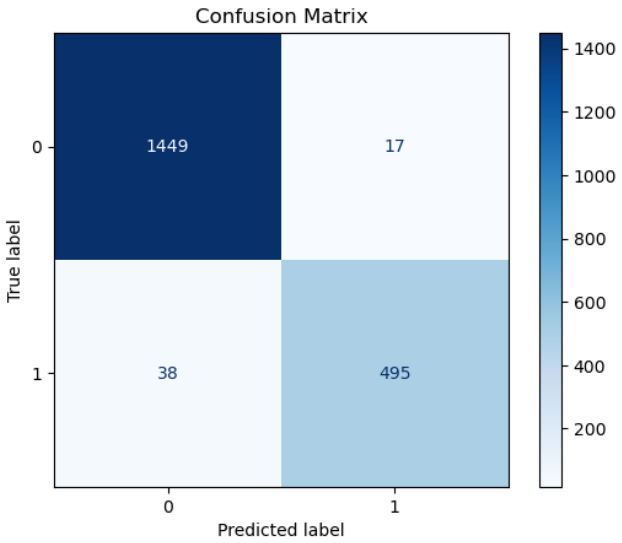

Figure 3

The Random Forest model had a slightly higher average accuracy score at 98%. The confusion matrix for the RF model can be seen in Figure 4, which tells us that 1,453 “True” labels were predicted correctly and 13 incorrectly, and 519 of the “False” labels were predicted correctly and 14 incorrectly. It looks like the Random Forest model had slightly more accurate predictions for the spam email labels.

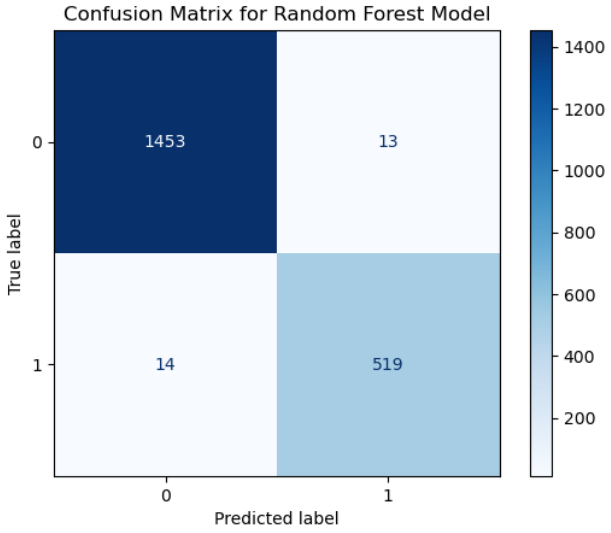

Figure 4

To further assess how well the predicted labels compared to the actual label, an ROC (Receiver Operating Characteristic) curve was generated that can be seen in Figure 5 for the Naïve Bayes model and Figure 6 for Random Forest model. The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate across different probability thresholds.

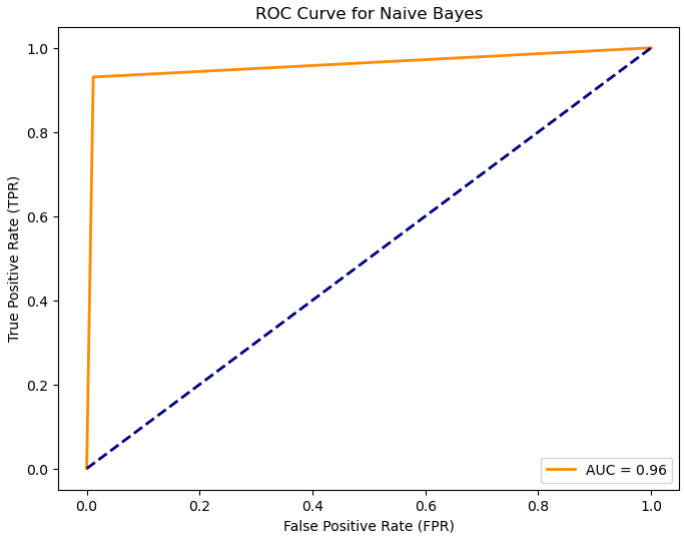

Figure 5

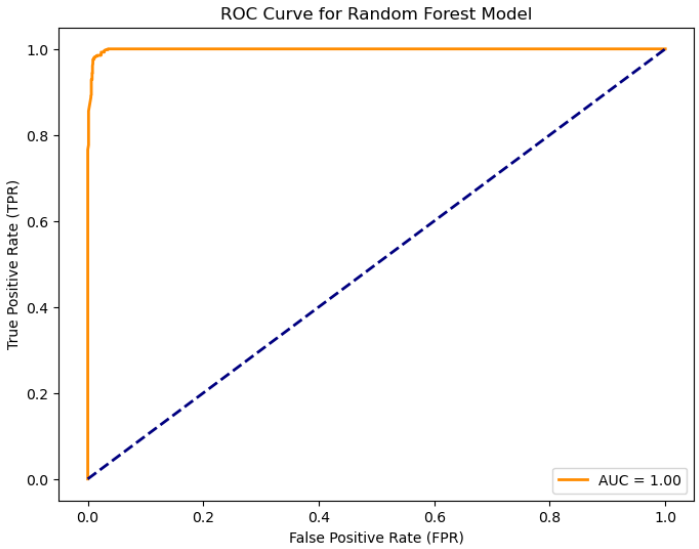

Figure 6

The AUC metric evaluates the model's ability to discriminate between classes across various probability thresholds. The AUC value ranges from 0 to 1, where a higher AUC indicates better performance. For the Naïve Bayes model, an AUC of 0.96 indicates excellent discriminatory power. However, it looks like the Random Forest model actually produce an AUC of 1 which essentially means the model perfectly distinguishes between the two classes. While this sometimes indicates overfitting or bias, judging by the confusion matrix, it seems that these results are indeed accurate with low Type I and Type II errors.

4 | CONCLUSION

In conclusion, both the Naïve Bayes and Random Forest models performed very well on classifying the emails as either spam or non-spam. While the Naïve Bayes model had an accuracy of 97% and an AUC of 0.96 which indicates a very good performance, the Random Forest model performed every better with an accuracy of 98% and an AUC of 1, which indicates excellent discriminatory power. While additional text cleaning could be performed to further improve the accuracy of the results in this context, both Naïve Bayes and Random Forest models seem to perform very well on creating a classification model for distinguishing spam email.

5 | CODE

Relevant code is attached in CarolinaCraus_CaseStudy3.ipynb