Proceedings of the 2nd International Conference on Management Research and Economic Development
DOI: 10.54254/2754-1169/84/20240817

Analysis of Spam Classification Based on Naive Bayes and Random Forest Model

Kejia Li[1,a,*]

[1]*Department of Computer Science, Guangxi University of Science and Technology, Liuzhou, China*
a. k.li.30@student.scu.edu.au
**corresponding author*

Abstract: Spam classification has become more and more significant in email filtering and content auditing systems nowadays. Despite the development of many ways for filtering spam, spammers continue to adopt new methods for spam detection, which has left us overwhelmed with spam. Furthermore, robust, and flexible categorization algorithms are necessary to keep up with the constant evolution of spam tactics. The best method for categorizing and filtering spam now is to use machine learning techniques. In this study, a large spam dataset containing 5572 email instances is used in simulations for the spam classification task. This study comparatively analyzes two prevalent machine learning algorithms, namely, Random Forest and Naive Bayes. A detailed description of both algorithms, including their theoretical foundations and practical implementations in spam detection, is provided. In addition, the data was characterized in the study for training the models as well as making predictions. Finally, the effectiveness and performance of each algorithm is shown in the experimental evaluation using four commonly used performance evaluation metrics. Overall, these results providing insights into their strengths and limitations in practical spam filtering applications.

Keywords: Spam classification, Naive Bayes, Random Forest, Performance evaluation indicators, feature engineering

1. Introduction

Contemporarily, the Internet will inevitably become a necessary component of daily life. It's expected that more people will utilize the internet, and email is already a helpful tool for communicating information and ideas in social and commercial contexts. Along with the rise in email and Internet usage in the past several years, spam has also expanded tremendously [1]. Spam is becoming a major menace to society as well as the Internet due to its growing volume. It has turned into a security concern for the corporate and educational sectors and has evolved from being bothersome to costly and dangerous. The mystery is that it's hard to define spam. What one individual considers spam may not be the same for another [2]. Fortunately or unfortunately, spam will continue to exist, and its global reach will only increase.

As a result, spam filtering can manage the issue in several ways. Numerous approaches have been put forth to address these issues. Static and dynamic approaches are the two categories into which these techniques may be divided. Static approaches use a pre-established address list to identify spam emails. Helfman and Isbell, for instance, worked to include the ability for users to design basic rules for email filtering into various priority folders into the Ishmail filtering system [3]. Users can filter

DOI: 10.54254/2754-1169/84/20240817

and prioritize emails utilizing comparable, but less advanced, filtering subsystems found in numerous email applications. Nevertheless, creating and updating filtering rules is a laborious process [4]. Traditional rule-based spam filtering methods have proven to be insufficiently effective in dealing with the dynamics and evolution of spam technologies. Dynamic approaches for spam email identification consider the email contents and adjust their filtering decisions accordingly, unlike static systems that depend on predefined address lists. Filtering technique relies on a predefined set of terms and phrases that identify spam messages. They mostly utilize general text categorization and data mining approaches through the implementation of machine learning technologies [5]. As a result, machine learning-based classification algorithms have emerged as a promising solution, utilizing labeled datasets for automatic learning to identify and classify spam.

This study examines the categorization performance of Naive Bayes and Random Forest machine learning algorithms for spam filtering. In Section 2, the dataset used in this paper and how it was preprocessed is described. Section 3 discusses the use of feature engineering to analyze data. Section 4 discusses performance standards and assessment measures for the use of classification models. Section 5 describes the two machine learning algorithms used in this research namely Naive Bayes and Random Forests and reports on the findings of the experiment. Section 6 wraps up by outlining the conclusions and prospects.

2. Data

In text categorization tasks, data loading and preprocessing are important steps before feeding the data into a machine learning model. The substantial dataset consisting of 5572 email instances in this study was downloaded from https://www.kaggle.com/datasets/mfaisalqureshi/spam-email. Using the versatile pandas library, known for its efficient handling of structured data, this study obtained the dataset directly from an Excel file and used the read_excel function to seamlessly convert the spreadsheet into a DataFrame object, a resizable, 2D, possibly heterogeneous tabular data structure with designated axes, Table 1 shows the head rows of the data set.

Table 1: The head rows of the dataset

	Message	Category
0	Go until jurong point, crazy.. Available only ...	ham
1	Ok lar... Joking wif u oni...	ham
2	Free entry in 2 a wkly comp to win FA Cup fina...	spam
3	U dun say so early hor... U c already then say...	ham
4	Nah I don't think he goes to usf, he lives aro...	ham

To familiarize ourselves with the dataset and identify any abnormalities or patterns, this study employed several exploratory data analysis (EDA) techniques:

- .head(): onw was able to have a first look at the characteristics and values of the DataFrame by using this approach to acquire the first few items.
- .info(): A succinct description of the DataFrame was given, along with the count of non-null values in each column, thereby allowing us to quickly ascertain the completeness of the dataset.
- .describe(): Offered a statistical summary for numerical columns and an overview of the distribution and central tendencies within these features.
- .shape: Revealed the dimensions of the dataset, giving us the count of entries and features.
- .value_counts(): Applied to categorical columns to determine the frequency of each unique value, which is particularly useful in assessing the balance or imbalance of the target classes.

DOI: 10.54254/2754-1169/84/20240817

To ensure the integrity of the dataset, the study utilized the .drop_duplicates() method to eliminate any instances of duplicate rows. This step is critical to prevent any skewed results due to repeated entries. Missing values can introduce significant bias and inaccuracies in a model's performance. To tackle this, this study substituted all null values with empty strings (df=df.fillna(" ")), ensuring that the subsequent text processing steps operate on a complete dataset without loss of information. Classification algorithms require digital inputs, thus the categorical labels in the 'Category' column were transformed into a numerical format. This was accomplished using the LabelEncoder from the sklearn.preprocessing module, which encodes labels with value between 0 and the number of classes minus 1. Because text is predictive, different features are carefully taken out of the message body. The author has meticulously taken out a number of aspects from the message body. Each message's character, word, and sentence counts were determined to perhaps identify common themes in spam and non-spam messages (spam, for example, tends to employ longer, more repeated wording).

3. Feature Engineering

To improve the caliber of the characteristics that were derived from the text data, it was carefully cleaned. This study used regular expressions, a potent tool for pattern matching and manipulation within text strings, to achieve this. Research created expressions to look for and omit non-alphabetic characters using Python's re module because these are frequently unimportant for spam detection and could contaminate the models. Extraneous spaces were eliminated as well, such as leading, trailing, and numerous consecutive spaces in text strings. The text data is more consistent and has less dimensionality thanks to this standardization of white space. After the text was cleaned, it was changed to lowercase to maintain consistency and avoid having the same term read differently depending on its case.

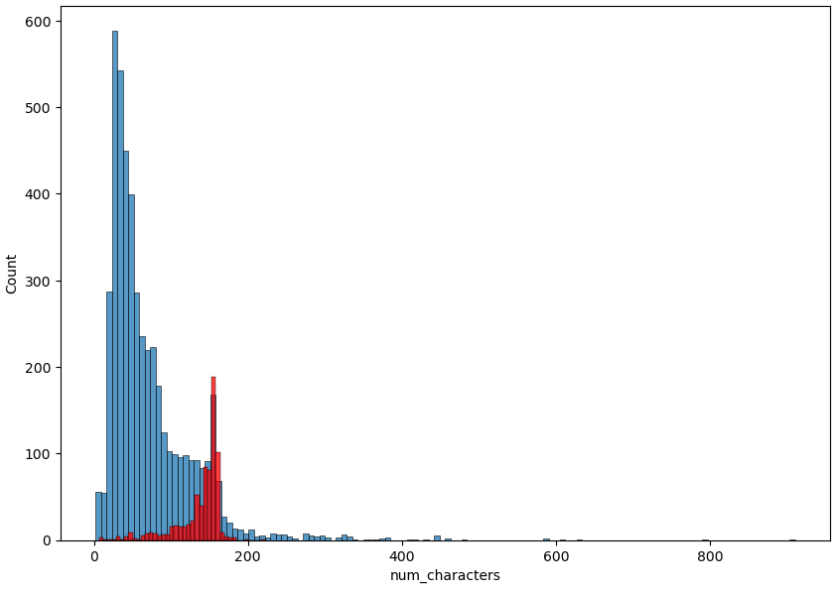

Figure 1: A histogram representing the distribution of the number of characters in non-spam (Category 0) and spam (Category 1) (Photo/Picture credit: Original).

The cleaned text data was transformed into a numerical format using the TfidfVectorizer from the sklearn.feature_extraction.text module. Term Frequency-Inverse Document Frequency (TF-IDF) assesses the significance of each word in a document by comparing its frequency in that document to its frequency in all documents [6]. The author set parameters within TfidfVectorizer to optimize the feature space: The number of features was limited by specifying the max_features parameter, thus

DOI: 10.54254/2754-1169/84/20240817

focusing on the most relevant terms and reducing computational complexity. Stop words, the words that are often utilized but usually have little significance, were removed by setting the stop_words parameter to 'english'. The author adjusted the maximum (max_df) and minimum (min_df) document frequency thresholds to exclude terms that appear too frequently or infrequently across the corpus. Terms that appear in almost all documents or in very few documents tend to be less informative.

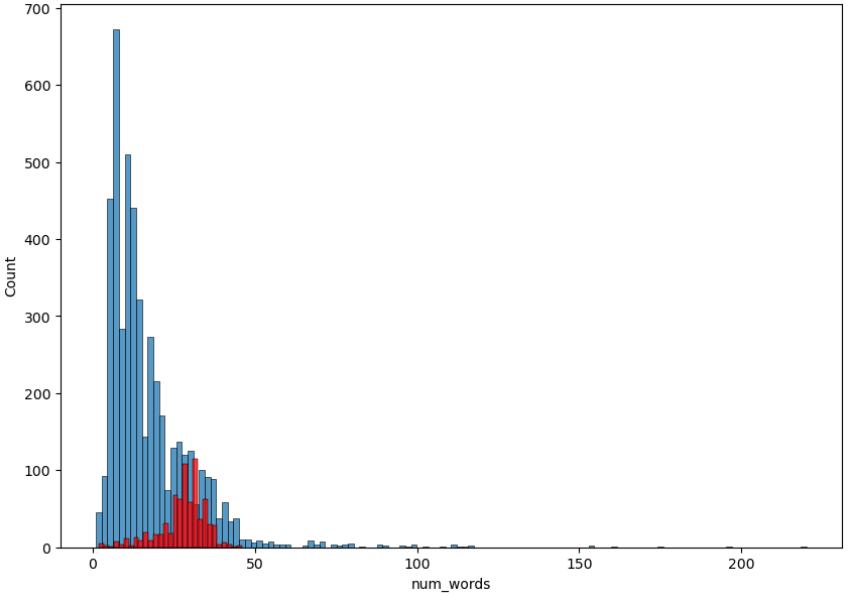

Figure 2: A histogram representing the distribution of number of words in non-spam (Category 0) and spam (Category 1) (Photo/Picture credit: Original).

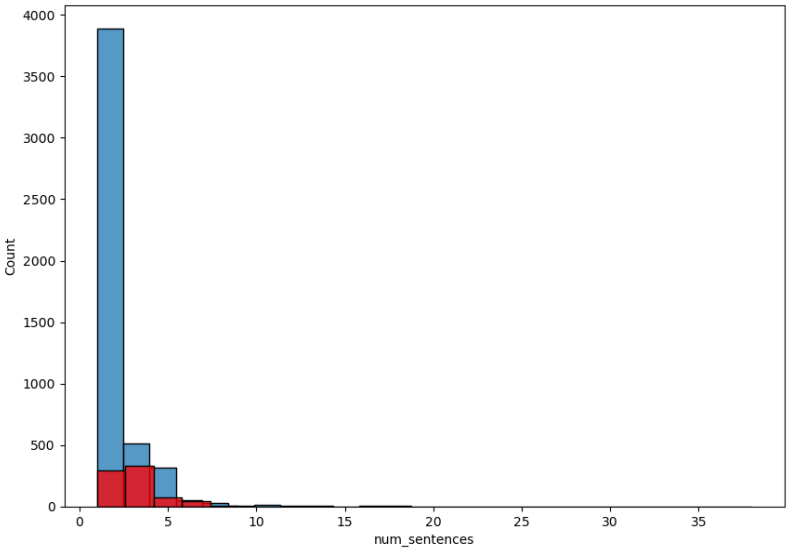

Figure 3: A histogram representing the distribution of number of sentences in non-spam (Category 0) and spam (Category 1) (Photo/Picture credit: Original).

The study utilizes visualization strategies from software packages such as seaborn and matplotlib to gain insight into the feature space and the relationships between different variables. One can look at each feature's distribution by plotting a histogram. Fig. 1, Fig. 2, and Fig. 3 show the distribution of textual statistics from prior phases, including the number of characters, words, and sentences, for

DOI: 10.54254/2754-1169/84/20240817

both spam and non-spam communications. To find potential connections between different features, Author created a correlation heatmap, as shown in Fig. 4. Heatmaps are a helpful tool for quickly identifying multicollinearity or duplication between variables, which can be harmful to certain feature-independent models. This study computed the correlation matrix between the heatmap variables using a tool like corr(), and then the study used seaborn.heatmap to create a graphical representation. This is a crucial step since it tells you which characteristics matter more in distinguishing between spam and non-spam.

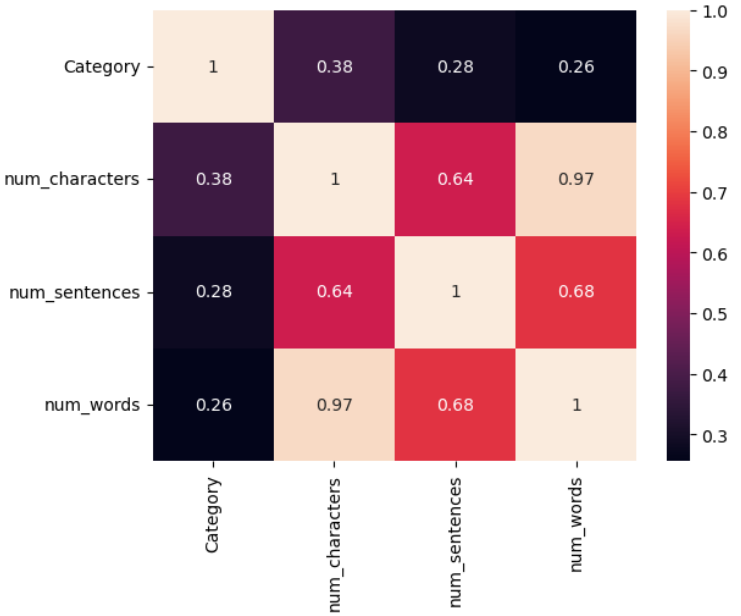

Figure 4: A heatmap between categories, number of characters, number of words and number of sentences (Photo/Picture credit: Original).

4. Performance Standards and Assessment Measures

Examining the performance criteria and evaluation metrics of spam classification models is essential as they help in understanding the performance of the model in different dimensions. The accuracy rate is a crucial indicator of the model's overall categorization precision. The accuracy rating presents what proportion of emails the algorithm correctly categorized, evaluating the model's overall performance. The confusion matrix offers details about the model's individual performance for each category. The confusion matrix enables the analysis of a model's classification performance on spam and non-spam emails, consisting of true and false positives, true and false negatives. This aids in recognizing the prejudices and inaccuracies of the model across many categories and directs future enhancements. A measure that combines the model's recall and accuracy is called the F1 score. Precision rate quantifies the proportion of emails identified as spam by the model that are indeed spam, whereas recall assesses the model's capability to accurately detect all spam emails. The F1 score assesses both precision and recall simultaneously, offering a well-rounded and thorough assessment of the model's performance. Error rate is a crucial indicator that indicates the frequency of errors in the model's overall predictions. By examining these performance standards and evaluation metrics, one may get a thorough comprehension of the model's advantages and limitations, offering direction for future enhancements and refinements. The evaluation metrics used in the testing process can be defined as follows:

The calculation of accuracy involves dividing the total number of test occurrences by the sum of True Negatives and True Positives. This evaluates the classifier's overall accuracy [2].

DOI: 10.54254/2754-1169/84/20240817

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

The computation of recall involves dividing the total number of true positives by the sum of false negatives and true positives. This is the number of items in a class that are incorrectly classified as a different category [2].

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2)$$

The precision may be defined as the ratio of true positive cases to the total of true positive and false positive cases. This indicates the accuracy of correctly detected objects within a specific class [2].

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3)$$

The F1 Score, which measures the balance of a performance of classification model, is the harmonic mean of recall and accuracy across positive and negative categories.

$$\text{F1 Socre} = \frac{2(\text{precision recall})}{\text{precision} + \text{recall}} \quad (4)$$

The total diagonals in matrix represent the number of cases that have been correctly classified; all other cases have been incorrectly classified. This is one approach of utilizing a confusion matrix to evaluate the performance of a classifier [7, 8].

5. Machine Learning Algorithms and Results

5.1. Naive Bayes

In Bayesian statistics, the phrase "Naive Bayes classifier" refers to a basic probabilistic classifier that utilizes the Bays' theorem but makes strong (naive) independence assumptions. "Independent feature model" is a better way to describe the underlying probability model [9]. When it comes to text categorization, Naive Bayes assumes that a word's presence in a document is independent of the presence of other words, which is usually not true in reality, but it simplifies calculations and performs effectively. The Naive Bayes method initially preprocesses the text data by tokenizing it into individual words or tokens. The system gathers all distinct words from the training dataset to establish a vocabulary. Every word in the vocabulary will serve as a feature utilized by the classifier to create predictions. The likelihood of each category (e.g., spam or non-spam) is determined by the occurrence frequency of documents assigned to each category in the training dataset. This establishes an initial probability for each category and then computes the conditional chance of each word in the lexicon appearing in each category. This entails determining the frequency of each word in papers categorized by their respective categories. Naive Bayes Classification The approach computes the a posteriori probability for each category by analyzing the characteristics (words) of the document to categories it. This is calculated by multiplying the conditional probabilities of the phrases in the texts inside each category by the prior probability of that category. The predicted category for the document is determined by selecting the category with the highest posterior probability. This rule is referred to as maximal a posteriori (MAP) estimate.

DOI: 10.54254/2754-1169/84/20240817

5.2. Random Forest

The Random Forest approach in text categorization is based on ensemble learning and decision trees. To enhance the precision of predictions, Random Forest is an ensemble learning approach that combines the predictions of several individual decision trees. Each decision tree is trained individually using a subset of the features and training data, and the final prediction is determined by aggregating the forecasts of all the decision trees. At each split point, a random subset of features (words) is chosen to train each decision tree in a random forest. Random feature selection aids in decorrelating the decision tree and mitigating overfitting, hence enhancing the generalization performance. Random forests use a technique called bootstrap aggregation or bagging, where several bootstrap samples (random samples, replaceable) are taken from the training dataset. Each decision tree is trained on distinct bootstrap samples to guarantee variety among them. The bagging method involves creating a randomized decision tree at each iteration in a random forest, resulting in very accurate predictions. Brayman's random forest is considered a mediocre classifier according to a study [10]. Once all decision trees are trained, predictions are made for each tree. Final predictions in classification tasks are typically made using a majority vote of decision trees. The average of each decision tree's predictions is the final prediction in regression problems. Each tree has a minimum number of samples and a maximum depth, and the number of trees in the forest needed to divide a node are just a few examples of the hyperparameters that may be adjusted in random forests to improve their performance. Hyperparameter tuning is frequently done using techniques such as cross-validation. A 3-Fold cross-validation method was utilized in this investigation.

5.3. Experimental Results

Random Forest and Naive Bayes are compared in terms of their performance in dealing with spam classification. 5572 email cases from a sizable spam dataset were utilized to conduct the simulations. The experimental results are given in Table 2, Table 3 and Table 4. Table 4 shows the accuracy, precision, recall as well as F1 scores using the Naive Bayes model and the Using 3-Fold Random Forest model. This evaluates their performance on the test set. The Naive Bayes and Random Forest model possesses an accuracy of 0.979 and 0.971, a precision of 0.977 and 0.988, a recall of 0.863 and 0.775, and an F1 score of 0.917 and 0.869, separately. Table 2 and Table 3 present the confusion matrix for the Naive Bayes algorithm and 3-Fold Random Forest, respectively.

Random Forest and Naive Bayes are compared in terms of their performance in dealing with spam classification. 5572 email cases from a sizable spam dataset were utilized to conduct the simulations. Table-4 shows the accuracy, precision, recall as well as F1 scores using the Naive Bayes model and the Using 3-Fold Random Forest model. This evaluates their performance on the test set. The Naive Bayes and Random Forest model possesses an accuracy of 0.979 and 0.971, a precision of 0.977 and 0.988, a recall of 0.863 and 0.775, and an F1 score of 0.917 and 0.869, separately. Table 2 and Table 3 present the confusion matrix for the Naive Bayes algorithm and 3-Fold Random Forest, respectively.

Table 2: A Naive Bayes confusion matrix

Actual Class/Predicted Class	Spam	Legitimate
Spam	896	0
Legitimate	26	110

Table 3: A Random Forest confusion matrix

Actual Class/Predicted Class	Spam	Legitimate
Spam	4510	6
Legitimate	144	497

DOI: 10.54254/2754-1169/84/20240817

Table 4: Evaluation data for both models

Model	Accuracy	Precision	Recall	F1 Scores
Navies Bayes	0.979	0.977	0.863	0.917
Random Forest	0.971	0.988	0.775	0.869

6. Conclusion

To sum up, this study explored the effectiveness of the Naive Bayes classifier and the Random Forest classifier in the spam classification task using a dataset containing textual information. This paper pre-processed the data, including cleaning and vectorization of textual information, and then trained the classifiers on the transformed data. Experimental results show that both classifiers perform well in distinguishing between spam and non-spam emails. Cross-validation results show that both classifiers generalize well to unseen data. The accuracy is 0.979 for the Naive Bayes classifier and 0.971 for the random forest classifier. In addition, this research comprehensively analyzes the performance of the classifiers using various evaluation metrics consisting of recall, F1 score, confusion matrix, accuracy together with precision. These metrics provide insight into the ability of the classifiers to correctly categorize spam and non-spam messages, as well as their strengths and weaknesses. There are several avenues for future research and improvement in spam classification. Researchers can enhance classification performance in feature engineering by investigating additional features or text data formats like word embedding or advanced text preparation techniques. When it comes to optimizing algorithms, testing various hyperparameters and ensemble techniques can enhance performance. Moreover, the research utilizes advanced deep learning models for instance transformer-based architectures or recurrent neural networks (RNN) for spam categorization tasks to detect intricate patterns in textual input. Scholars can handle class imbalances in a dataset by utilizing strategies like oversampling, undersampling, or employing complex algorithms specifically built for imbalanced data. One can also implement trained classifiers in live spam filtering systems, including scalability, efficiency, and compatibility with current email platforms. With efforts in these directions, one can further promote the field of spam categorization and thereby make a contribution to the generation of more reliable and efficient solutions to combat unsolicited and potentially harmful messages in various communication channels.

References

[1] *Pu, C. and Webb, S. (2006). Observed Trends in Spam Construction Techniques: A Case Study of Spam Evolution. In CEAS pp. 104-112)*

[2] *Mishra, R. and Thakur, R.S. (2013). Analysis of random forest and Naive Bayes for spam mail using feature selection categorization. International Journal of Computer Applications, 80(3), 42-47.*

[3] *Helfman, J. and Isbell, C. (1995). Ishmail: Immediate identification of important information. Technical report, AT&T Bell Laboratories, MIT Artificial Intelligence Laboratory.*

[4] *Rennie, J. (2000). An application of machine learning to e-mail filtering. In Proc. KDD-2000 Text Mining Workshop pp. 75-80.*

[5] *Yu, B. and Xu, Z.B. (2008). A comparative study for content-based dynamic spam classification using four machine learning algorithms. Knowledge-Based Systems, 21(4), 355-362.*

[6] *Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. Proceedings of the first instructional conference on machine learning, 242(1), 29-48.*

[7] *Ramachandran, A., Dagon, D. and Feamster, N. (2006). Can DNS-based blacklists keep up with bots?. In CEAS.*

[8] *Mishra, R. and Thakur, R.S. (2013). Analysis of random forest and Naive Bayes for spam mail using feature selection categorization. International Journal of Computer Applications, 80(3), 42-47.*

[9] *Hall, M.A. and Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data engineering, 15(6), 1437-1447.*

[10] *Schonlau, M. (2023). The Naive Bayes classifier. In Applied Statistical Learning: With Case Studies in Stata, Cham: Springer International Publishing, pp. 143-160.*