Naïve Bayes Classification for Email Spam Detection

Zain Syed
University of Wollongong in Dubai, Dubai, UAE
Email: zs089@uowmail.edu.au

Omar Taher
University of Wollongong in Dubai, Dubai, UAE
Email: ot756@uowmail.edu.au

A B S T R A C T

Email is one of the cheapest forms of communication that every internet user utilizes, from individuals to businesses. Because of its simplicity and wide availability, it is vulnerable to threats by perpetrators through spam with malicious intents, which are known to have resulted in huge financial losses and have threatened the privacy of millions of individuals. Not all spam emails are malicious, however, they are a nuisance to users regardless. It is because of these reasons that there is a dire need for good spam detection systems that are automatically able to classify emails as spam or ham. This paper aims to do exactly that. The Enron email dataset is first obtained and combined with a second 419 Fraud dataset containing spam emails. Lemmatization with POS tags is performed on the datasets in order to improve the model's performance in terms of time of execution and accuracy. A Naive Bayes approach is proposed along with Grid Search to maximize the model's accuracy with regards to classifying spam emails. Finally, the model is evaluated through various metrics and a comparative analysis is performed to compare the results of the proposed model with other similar research papers that also aim to solve the problem of email spam detection

A R T I C L E I N F O

Keywords:
Email Spam Detection, Machine Learning, NLP, Text Pre-processing, Lemmatization, Classification, Naïve Bayes, Bernoulli, Multinomial, Gaussian, Feature Extraction, TF-IDF, Hyperparameter Tuning, GridSearchCV, Precision-Recall, ROC-AUC

Introduction

In 2021, it was revealed that 3,026,626 emails were sent every second, 67% of them being spam (Andre, 2021). Email spams have been known to cost businesses up to $20.5 billion each year (What's on the Other Side of Your Inbox - 20 SPAM Statistics for 2022, 2022). With humongous amounts of spam being generated day by day, it is crucial for powerful anti-spam filtration systems to be developed in order to prevent such drastic losses.

Text classification is the fundamental concept behind spam detection. Machine learning, natural language processing (NLP) and artificial intelligence come hand in hand to classify all kinds of text with an unsurpassed accuracy. Unlike rule-based classification approaches which classify text into groups manually based on a set of handcrafted linguistic rules, machine learning text classification actually learns to make classifications based on past observations or pre-labeled examples given to the classifier as training data (Text Classification: What It Is and Why It Matters, 2014). Such classifiers are able to learn different associations between texts and establish that a particular output is expected for a particular input. The first step in this process is feature extraction or vectorization, which will be further elaborated in detail below, after which the feature sets along with their predefined labels, or "tags" are used to train the model. This in turn will produce a classifier model that can successfully sort text into different categories, be the classification binary or multiclass. Once the model is thoroughly trained with enough training samples, test data will be provided which will be transformed into feature sets after which accurate predictions are made. Text classification through machine learning and NLP techniques is much faster and efficient as compared to human-crafted rule systems, which consist of more than a few flaws.

In this paper, a Naïve Bayes model was created upon a grid search technique using the tf-idf matrix as the feature extraction method. The model is trained to differentiate spam from ham emails with a high accuracy rate. Section 2.1 to 2.3 gives an overview on the basic pre-requisites needed to understand the working of various aspects and concepts that will be used further in the paper. Section 3 lists out a number of research papers that proposed different ideas and approaches using various classification methods and even a few deep learning approaches to tackle the issue of email spam detection. The accuracies of each are listed out which will be later compared to this study's results in Section 5, which dissertates the results of the proposed model and involves a comparative analysis, linking other research papers' outcomes with the currently discussed model. Section 4 consists of a list of subsections that together aim to describe the methodology behind the suggested approach to the problem. Finally, Section 6 concludes the paper by restating the problem and providing a final overview of how it had been approached and solved.

2.1 Naïve Bayes Classification

The Bayesian Classification is a mathematical classifier based on Bayes Theorem in probability that determines the credible target for a given feature. Naïve Bayes is a significantly less complicated extension of classifiers that make a "naïve" assumption of the features involved, which are independent to each other. Although it may seem like an irrational assumption, it has proven to yield eminent outcomes on

par with far more intricate algorithms. Efficacious applications of NB have been implemented including but not limited to, text classification, weather forecasting, biometrics identification and news article categorization. Let's have a deeper comprehension on how the algorithm works by understanding Bayes' Theorem (Rish, 2001).

Bayes' Theorem is a probabilistic theory that enables calculations of certain probabilities that evince the impact of an event in contrast to another (also known as Conditional Probability).
It can be written as (Garg, 2013):

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \quad \textbf{(Eq. 1)}$$

Where,
P(A) : Probability of the event A occurring
P(B) : Probability of the event B occurring
P(A | B) : Probability of the event A occurring given the event B
P(B | A) : Probability of the event B occurring given the event A

In Eq. 1, P(B) is the probability of A having no information on the event of A. Thus, there can be a hypothesis of A being true or false, thus, Bayes' Theorem can also be written as:

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B \mid A) \times P(A) + P(B \mid \neg A) \times P(\neg A)} \quad \textbf{(Eq. 2)}$$

Where,
P(A) : Probability of the event A occurring while being false
P(B | A) : Probability of the event B occurring given the event A is false

Based on this theorem (Eq. 2), a feature variable can be classified to its target class assuming all other features are independent to each other. Although Naïve Bayes gives staggering results while classifying corpora with limited number of data and document fragments (Muhammad, Bukhori & Pandunata, 2019), one of its weaknesses is that it is prone to yield low accuracies considering the selection of independent variables and probability evaluation that may not run optimally (Nur Hayatin, Gita Indah Marthasari & Nuarini, 2020).

2.2 Text-Vectorization

The significance of data has proliferated to every aspect of our lives. Amongst the different representations of data, the textual form is ceaselessly the most widespread, especially considering its usage in communication across diverse platforms. It is imperative to utilize the ability of machines to generate

imperceptible insights by transmuting the data in a form that is conveniently transparent to the computers. One of the preliminary methods involved in converting text documents in a way that is machine-readable, is by vectorization (Singh & Shashi, 2019).

Text-Vectorization is almost a prerequisite to any Natural Language Processing (NLP) task that involves solving text based problems mathematically. These vectorizers can range from very lucid to really complex models (Singh & Shashi, 2019). An unsophisticated and basic approach to build such vectors would be to generate a numerical representation of all words in the lexicon using binary encoding. This method is called One-hot Encoding and should be avoided as it would lead to dimensional disaster and data sparsity. Some of the more common methods to represent a vector make use of word frequency for each term in a document. Although most of these approaches still fail to solve problems such as dimensionality reduction and feature extraction of text semantics, they are commonly used to solve most classification problems. (Yang et al. 2022). A brief explanation of such methods are as follows:

2.2.1 Bag of Words

In general classification of documents, a Bag of Words (or commonly known as BoW) is a vector representation of term frequencies. It retrieves word occurrences for each sentence or document while considering word duplicates but disregards grammar and its sequence. This form of numerical representation is usually enforced in classification methods where the features used are based on the frequency of each term (Wisam Abdulazeez Qader, et al., 2019). Consider an example of a paragraph containing two sentences that are as follows:

"The moon is bright. It seems to be really bright."

The Bag of Words representation for this corpus is shown in Table 1.

Documents	be	bright	is	it	moon	really	seems	the	to
The moon is bright	0	1	1	0	1	0	0	1	0
It seems to be really bright	1	1	0	1	0	1	1	0	1

Table 1: Bag of Words representation

2.2.2 Word2Vec

The Word2Vec model is a word vector generation method that makes use of word embeddings and consists of skip-gram and Continuous Bag of Words (CBOW). It is designed to contain a disseminated emotion of words that map them to vectors with describable dimensions. Since the context is contemplated while training, a less dense dimensional vector with text semantics is produced. Each term in a document

is intrinsically established as a word sequence (Yang et al. 2022) (shown in Figure 1).

$$d = \left[d^{(1)}, d^{(1)}, \dots, d^{(j)}, \dots, d^{(i)}\ \right]$$

Figure 1: (Yang et al. 2022)

Where,
d(j) indicates word at jth position
d(i) indicates word at last position

To acquire the word vector, a mapping association should be established between the observed word and its context. This is trained using a neural network where the hidden layer weight procured during training is the word vector itself (Yang et al. 2022).

Apart from these models, the Term Frequency - Inverse Document Frequency (TF-IDF), Doc2Vec (an extension of Word2Vec), pre-trained models like Bidirectional Encoder Representations from Transformers (BERT), Embedding from Language Model (ELMo) and many other models are also used to solve various NLP tasks and in most cases help in the classification of textual data (Singh & Shashi 2019).

2.3 Hyper-parameter tuning

Hyperparameters are a deliberate group of choices that precisely influence the training and outcome of machine learning models resulting in a substantial increase in the model's performance. Since the choices are dataset specific and there are no conventional set of parameters that would yield maximum performance, it is essential for an ML model to have the right hyperparameter setting. Most parameters have an extensive scale of values that can be specified. These can be selected based on default values proposed by the developer of the concerned ML algorithm or by finding values suggested by research authors working on a project within a similar domain. For either case, the model may not perform well considering the differences in each set of data. Another approach can be carried out by performing manual search. This requires prior knowledge and experience with the algorithm. Generally, these preconditions are strenuous to accomplish. Therefore, a systematic gradual tuning scheme that requires no expertise or profound knowledge would be a far more favorable alternative (Pannakkong et al. 2022). Some of the widely used hyper-parameter optimization techniques include Grid Search, Random Search, Bayesian Hyper-parameter Optimization with HyperOpt (Shekhar, Bansode & Salim).

3. Literature Review

Malicious and non-malicious spamming through email platforms has become a widespread nuisance for internet users worldwide, the former of which is known to have resulted in severe consequences such as great financial or budgetary misfortunes for businesses, organizations and even individuals. Despite the fact that not all spam messages are malicious, the separation of spam from ham

emails are a huge convenience to individuals and organizations alike, as such emails are usually undesired and not given much attention to. As the development of spam emails continues to grow at an alarming rate, the need for viable phishing and spam recognition systems also increases. Several machine learning algorithms have already been developed for the purposes of anti-spam email filtering. Some examples of which are perceptron-based neural model techniques such as single-layered perceptron (SLP) and multi-layered perceptron (MLP), logical algorithms such as decision trees and random forest classification, and statistical learning techniques such as instance-based techniques and Bayesian networks. Despite the employment of several of the aforementioned classifiers, Naïve Bayes classifiers were found to be particularly popular in commercial and open-source spam filters (Metsis & Paliouras, n.d.). This would be due to the machine learning classifier's simplicity and ease of implementation, accuracy and its linear computational complexity.

Hassan, Mirza and Hussain (2017) used Weka (Waikato Environment for Knowledge Analysis), a software that contains tools for data pre-processing, classification, etc. in combination with the Naïve Bayes classifier model to identify spam emails using a custom dataset that included only the header field attributes from emails such as "Subject", "To", "From", "Date", etc. The authors of the research paper supported the decision to extract the header and disregard the body by claiming that concentrating on the body section of emails when conducting spam filtration is "time consuming and inefficient". Although the main goal of the study was to determine whether a machine learning approach would yield better accuracies as compared to a non-machine learning approach, one conclusion that can be drawn after comparison with other similar studies is that, the elimination of the bodies of emails would not produce as accurate results as previously thought.

One such paper that was discovered to be against the idea of identifying spam emails only through the header sections while ignoring the body completely, was authored by Abiramasundari, Ramaswamy and Sangeetha (2021). In this study, the author claims that not all header features can be useful in identifying spam and may instead result in memory wastage and higher processing time, leading to inefficiency. The reasoning behind this claim is that various businesses such as tourism packages, loan providers, educational institutes, etc. try to approach multiple users for the purpose of aiding business development, and such emails are sent through genuine sources (Abiramasundari, 2021). Therefore, a system that concentrates only on headers would not be able to accurately serve its purpose.

Moving on from the header versus body argument, Abiramasundari's research paper focuses on designing a classification system that proposes Rule Based Subject Analysis (RBSA) and Semantic Based Feature Selection (SBFS) to analyze spam terms and reduce the number of features required for the classification process. The dataset used was the Enron Email Dataset (the same as the one used in this proposed model) from which 5132 emails were used in the study. Four classification algorithms were implemented, namely, Support Vector Machine (SVM), Multinomial NB, Gaussian NB, and Bernoulli NB. The accuracies that were produced were 97%, 95%, 88%, and 87% respectively.

Another paper proposed a framework for spam email detection using machine learning based Naïve Bayes algorithm in combination with computational intelligence-based Particle Swarm Optimization

(PSO), which is a stochastic optimization technique and is used for heuristic global optimization of parameters of the Naïve Bayes algorithm (Parmar et al., 2020). The Ling-Spam dataset was chosen to perform the classification of emails into classes of either ham or spam. The accuracy was found to be quite impressive, standing at 95.50%.

Hassan (2017) proposed a unique concept of investigating the effect of combining text clustering using K-means algorithm with various supervised classification mechanisms (such as K-Nearest Neighbor, Support Vector Machine, Naïve Bayes, Logistic Regression, etc.) and find out if this increases the classification accuracies. The dataset in question was the Enron Spam Dataset. The author's findings show that the conjunction of K-means clustering with supervised classifiers does not necessarily improve the classification of ham and spam for all mails. In addition, when the performance was found to be improved by clustering for the few cases that it did, the accuracy scores were found to be increased very slightly and did not seem worth it as compared to the amount of time it took to build the model that combined the two mechanisms together in the first place. The highest accuracy was found to be produced by the SVM classifier model combined with K-means clustering, with a score of 97.97%.

Soni (2020) made use of an improved intermittent Region-based Convolutional Neural Network model (RCNN) to propose an email spam recognition model named "THEMIS". The structure of emails is initially examined. After which, this model is capable of extracting emails at the header, body, word and character level and then utilizes these powerful functionalities to assess whether an email contains malicious or phishing intents or not. The exploratory outcome reveals that the precision of THEMIS is at an extraordinary rate of 99.84%, which is by far the most successful model as compared to the models of all the other research papers and studies mentioned in this literature review.

More recently, Sahmoud and Mikki (2022) conducted a research where a spam detector model was built using the pre-trained transformer model BERT (Bidirectional Encoder Representations from Transformers), a deep learning model, to classify emails by using the ability of the system to take the context of the email content into its perspective. The model was also trained using several different corpuses such as Enron Emails, SpamAssassin, Ling-Spam, and SMS spam collection corpus. This particular model was discovered to yield staggering accuracies ranging from 97.83% up to 99.28% for each of the different datasets (Sahmoud et al., 2022).

4. Methodology

This section of the paper meticulously demonstrates the implementation of an approach to classify emails into spam or ham utilizing python and its relevant libraries for NLP, Machine Learning and Hyperparameter Tuning. In Figure 2, a flowchart demonstrating the same is illustrated.

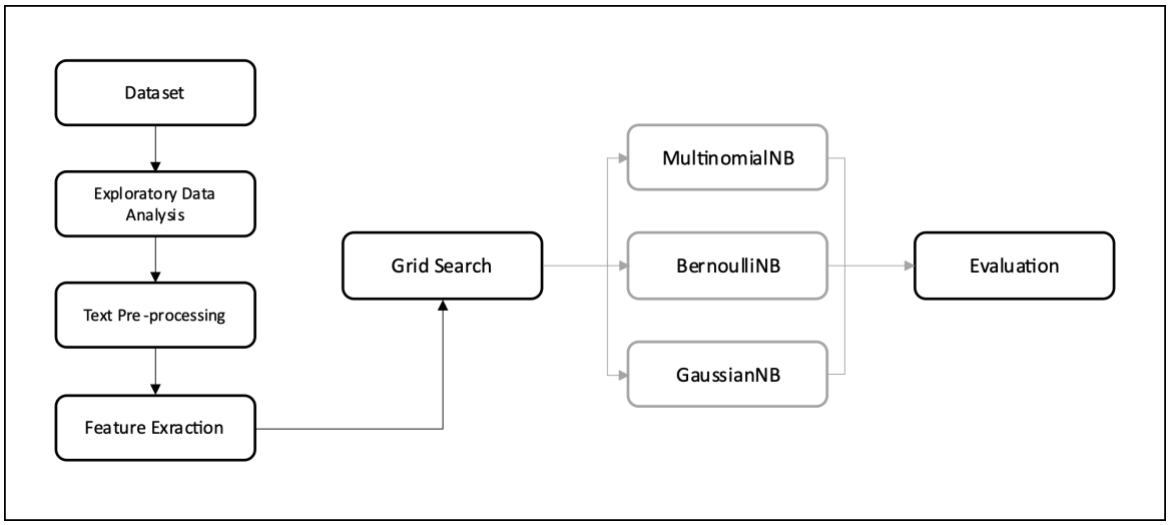

Figure 2: Flowchart for the approach

4.1 Study Dataset

The proposed research makes use of two datasets, one containing legit or ham emails and the other containing spam emails. Details of both corpuses are described below.

The Enron email dataset, a corpus containing approximately 500,000 emails generated by around 150 employees of the Enron Corporation, was used for the purpose of this study. The data had been collected by the Federal Energy Regulatory Commission during its investigation of Enron's collapse in 2001. The dataset initially was found to have a number of integrity issues and privacy concerns, which were later resolved as part of a redaction effort as per requests of the affected employees. This dataset is one of the only substantial collections of "real" email that is publicly available online. Most studies in this field of research are conducted on synthetic data due to an inadequate and real enough benchmark. The Enron email dataset is a touchstone for research in the field of spam detection among others.

The second dataset used in this study is the Fraudulent Email Corpus which consists of fraudulent emails containing criminally deceptive information with respect to the "Nigerian Letter" or "419" fraud, which cost its victims over $1 million in financial losses in 2016 [XX]. The dataset is a collection of more than 2500 fraudulent "Nigerian Letters" with phishing intentions from 1998 to 2007.

Both corpuses have a similar structure so they are both preprocessed the same way and the bodies of these emails are extracted, after which they are combined into one single corpus. However, before the conjunction of the two datasets, sampling is performed on both datasets to randomly extract 2000 samples from each, resulting in a final corpus of 4000 emails.

4.2 Text Pre-processing

Text pre-processing is a mandatory step in any Natural Language Processing task to retain the pivotal words and label them accordingly. To achieve this, various NLP libraries such as Natural Language Toolkit (NLTK), spaCy, email and re (for regular expressions) were used. Figure 3 shows the different stages involved in the pre-processing of text in this approach.

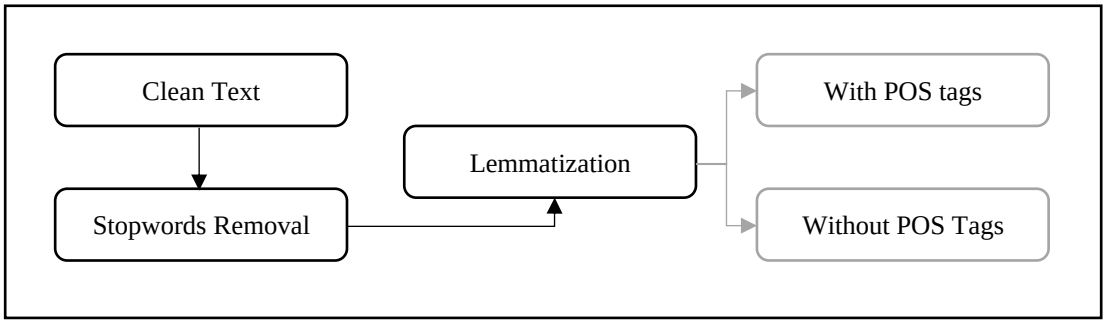

Figure 3: Pre-processing steps

4.2.1 Cleaning the text

Prior to cleaning the text, as mentioned above, the format of each email in the dataset was examined and it was observed that the messages were followed by some metadata such as the sender's and receiver's address, date, subject, type of encoding, along with other data that was irrelevant to this particular study. To extract the content of the messages (also referred to as payload), the "email" library of python was employed. The cleaning process consists of basic text preparation techniques as mentioned below:

- *Lowering of cases:*
  If the same word exists with different cases such as either upper or sentence case, two different instances for the same word would be created by the system. To avoid this, all the words in the corpus were transformed into lowercase to begin with.

- *Regex cleaning:*
  Removal of symbols, punctuations and digits is performed using a basic regular expression statement (from the re library), as this specific domain does not require the presence of such characters.

- *Tokenization (using basic python split):*
  This is done to break up paragraphs and extract meaningful elements (also referred to as tokens) that can be utilized to perform various other tasks.

- *Stop words removal:*
  Stop words are the less significant words that exist in a sentence. The removal of these words depends entirely on the dataset and the task performed. In this approach, these words have no significance in the classification process, therefore, they are removed. NLTK has incorporated an easy-to-use subclass called "stopwords" that contains a list of pre-set stop words to be deleted. This list is referred to during the process of tokenization and the matching words are eliminated.

- *Basic lemmatization:*
  Basic lemmatization (using WordNetLemmatizer from NLTK) is performed on the tokenized version of the corpus. Although this does not change much, it is a common practice to change the plural form of certain words to singular form, such as by removing the letter 's' towards the end.

To combine this multi-step process, a single function was created that expects the corpus as its input and a regular expression that can be applied to clean the text.

4.2.2 Lemmatization with POS tags

Lemmatization is a technique used to combine the analogous words in a way that can be identified as a single element (commonly referred to as the word's "lemma") (Divya Khyani & S, 2021). Table 2 shows an example of a few words and their lemmas. Using spaCy's tokenizer, each word was tokenized along with its POS tag. After which, they are compared to a user-defined list of tags, consisting of nouns, adjectives and verbs, to be lemmatized to. Due to computational constraints, the number of tags in this study were restricted to only three.

Words	Lemma
Corpora	corpus
Playing	play
Played	play
Plays	play
Different	differ

Table 2: Example of words and their vocabulary form (lemma)

4.3 TF-IDF Vectorizer

Term Frequency – Inverse Document Frequency or TF-IDF is a vectorizer utilized extensively for feature extraction and text mining to analyze key correlations between words in a selection of documents (

Das & Chakraborty, n.d.). A step-by-step explanation on how the vectorizer works is delineated below. As the name suggests, TF refers to the word count where higher values imply more significance to a document (Kim & Gil, 2019). Consider an example where two documents, A and B, contain the word "Hadoop". If document A and B consist of 1000 and 10000 words respectively, there is a high probability that "Hadoop" is more occurring in B. However, this does not imply that B is more significant than A. Therefore, there is a need to normalize the term frequency for each document which is resolved by dividing it by the total number of words in that document (shown in Eq. 3) (Kim & Gil, 2019).

$$TF(word \mid doc) = \frac{Frequency\ of\ word\ in\ doc}{No.\ of\ words\ in\ doc} \quad \textbf{(Eq. 3)}$$

The problem of finding the importance of words across documents persists. To achieve this, document frequency (DF) is used. This measures the significance of documents in a corpus by counting the occurrences of a term in it (shown in Eq. 4) (Kim & Gil, 2019).

$$DF(word \mid docs) = \frac{Frequency\ of\ word\ in\ docs}{No.\ of\ docs} \quad \textbf{(Eq. 4)}$$

Since the objective is to know the relevance of a word, the inverse of DF is taken into account. For almost all commonly occurring words, the IDF gives a low score with a relative weightage. However, when analyzing a large dataset, the IDF value spikes which may cause problems. Another complication that may occur is if a fixed vocabulary is chosen, some documents may not contain these terms, resulting in the DF to return zero. To deal with these issues, the log of IDF is taken to dampen the effect of high values and the denominator (containing DF) is increased by 1 to avoid dividing by zeroes (shown in Eq. 5) (Das & Chakraborty, n.d.).

$$IDF(word \mid docs) = \log \frac{No.\ of\ docs}{Frequency\ of\ word\ in\ docs + 1} \quad \textbf{(Eq. 5)}$$

From Eq. 3 and 5 we get,

$$TFIDF = TF \times IDF \quad \textbf{(Eq. 6)}$$

Using Eq. 6 in the spam detection dataset, a vectorizer that identifies the significance of terms in each document is obtained. Using python's sci-kit learn (sklearn) library, the TF-IDF matrix and vector is acquired with ease. This would assist in classifying emails into ham or spam by utilizing the vocabulary derived.

4.5 Exploratory Data Analysis (EDA)

EDA is a vital step in data analysis to uncover underlying patterns and identify abnormalities in a dataset. It is considered good practice to first comprehend the data and then perform the data mining tasks. Initially, the extraction of overlapping words between the two categories and removing them (if there exist any) was taken into consideration. This was done to reduce the number of common terms between ham and spam as it would lead to misclassification if two emails of opposing categories have the same terms. This was achieved by taking the top twenty frequent words from both classes (using basic python and pandas) and visualizing them (using subplots in matplotlib). As can be seen in Figure 4, the common words are 'please', 'company' and 'know'. These are identified and removed using python list functions. One thing to note is that the word 'enron' was found to be very specific to this dataset and had the highest weightage among ham emails. Due to this specificity, this word was removed.

Another feature considered was the correlation between the length of the message and its respective category. This association was tested because it was noticed that a number of similar researches in the field discovered a relationship between the length of the message and its class. Analyzing the correlation matrix plotted using python libraries such as Seaborn and Matplotlib, the resultant scores appeared to be weak being as low as 0.1 or 10% (refer Figure 5).

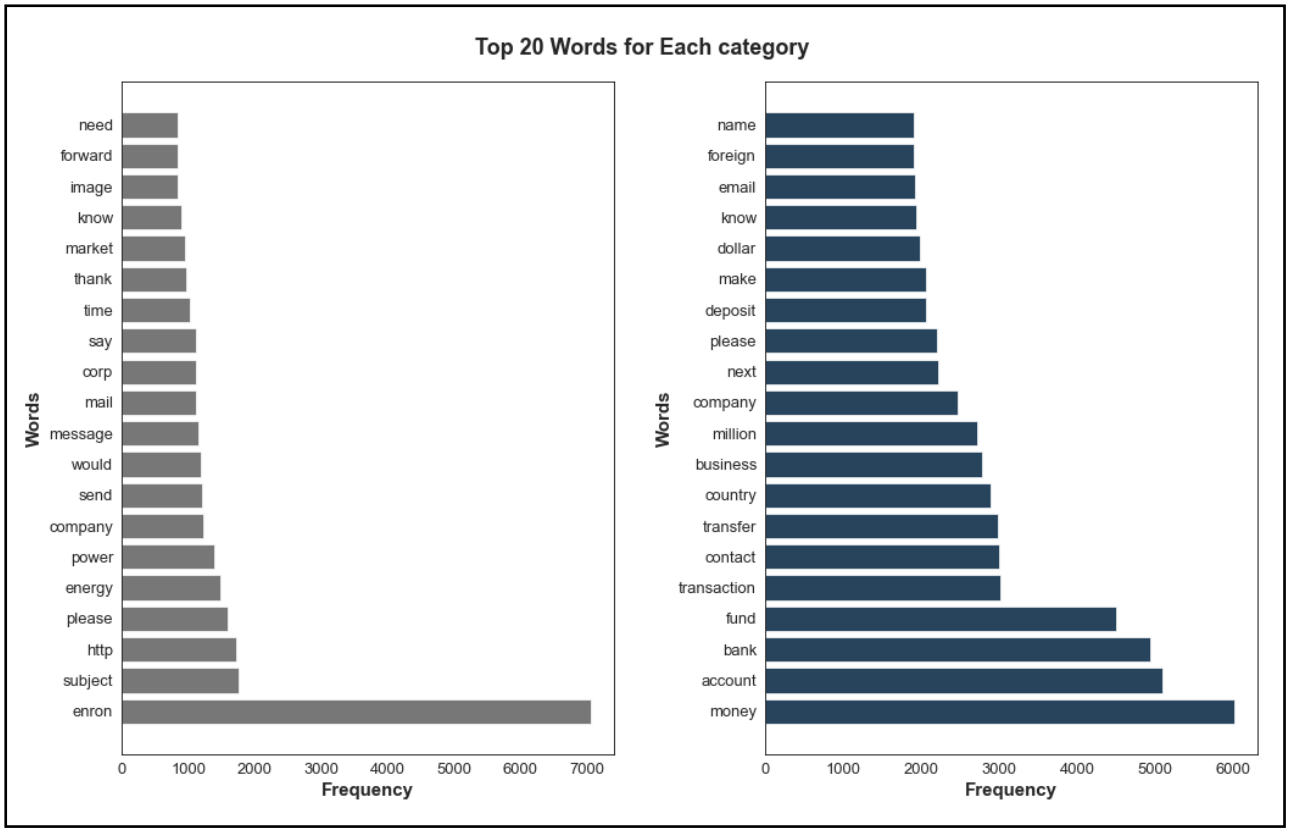

Figure 4: Visualization for overlapping words between ham(left) and spam(right)

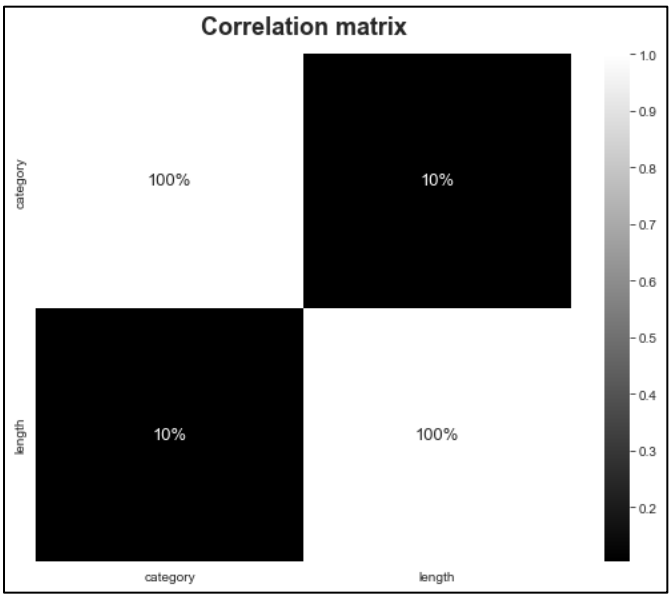

Figure 5: Correlation matrix for length of message and its category

4.6 Grid Search for Variants of Naïve Bayes

4.6.1 Types of Naïve Bayes

To perform the classification, three variants of Naïve Bayes are used as explained below:

- *Bernoulli Naïve Bayes:*
  Based on binary data, the Bernoulli model excels in feature sets where each token has the value 0 or 1 (Raschka, 2014). For example: the features indicate whether a word exists in a document.

- *Multinomial Naïve Bayes:*
  Identical to Bernoulli but performs well in scenarios where the features may have values more than 1 (Raschka, 2014). For example: the features indicate the frequency of each term in a document.

- *Gaussian Naïve Bayes:*
  The above two algorithms deal with categorical features, Gaussian Naïve Bayes stands out when the presumed independent variables are continuous in nature (Raschka, 2014). For example: the commonly used Iris dataset contains continuous features with class variables containing categories of the Iris flower.

Given these points, Multinomial ought to be better performing than Bernoulli and Gaussian. To further analyze these models a grid search is performed using various appropriate parameters.

4.6.2 Grid Search

The Grid Search is a Hyper Parameter Tuning method that iterates through every permutation of parameter combinations where the parameter space is examined and divided into a grid-like structure. Each space in the grid is assessed as hyperparameters to compute its optimum value (Shekhar et al., 2022). Although it is an exhaustive optimization tool, it is preferred over Random Search for this use case since it explores every possible combination rather than randomly selecting hyperparameters. Both the selection of models and the grid search is implemented with ease using the sci-kit learn library.

4.7 Evaluation

In this section, the evaluation metrics used for analyzing the best Naïve Bayes model are demonstrated. The predictions are evaluated in terms of accuracy, precision, recall, f1 score and Receiver Operator Characteristic (ROC) supported by its Area Under Curve (AUC). Before understanding how each of these metrics work, some fundamental concepts regarding outcomes need to be elaborated. Taking into consideration an example from the current study regarding email spam classification, the "true" value refers to the binary classification of 1 (in this case, spam), whereas "false" refers to 0 (ham). Bearing in mind that positives indicate a correct prediction and negatives indicate the opposite, a true positive signifies a correct spam classification, and vice versa in terms of true negatives. The same logic is applied to the classification of ham emails as regards to false positives and false negatives. Using these notions, a brief explanation of each metric is given below:

- *Accuracy*

  This is the most common metric and perhaps the easiest to understand. It calculates the proportion of results that are correct (refer Eq. 7) (Chowdhury & Schoen, 2020).

$$Accuracy = \frac{True\ positives + True\ Negatives}{Total\ Predictions} \quad \textbf{(Eq. 7)}$$

- *Precision*

  Precision indicates the ratio of correctly classified outcomes to the total predicted values (refer Eq. 8) (Chowdhury & Schoen, 2020).

$$Precision = \frac{True\ positives}{True\ positives + False\ Positives} \quad \textbf{(Eq. 8)}$$

- *Recall*

Recall stipulates the ratio of correctly classified outcomes to just the actual values (refer Eq. 9) (Chowdhury & Schoen, 2020).

$$Recall = \frac{True\ positives}{True\ positives + False\ Negatives} \quad \textbf{(Eq. 9)}$$

- *F1 score*

Ideally, a model with both high precision and recall would be the desired result. However, there would be a trade-off between the two metrics, meaning the model can either be tweaked to increase precision, but at the expense of weaker recall or vice versa. F1 score is the combination of the two metrics into one by calculating their harmonic mean. (Refer Eq. 10) (Chowdhury & Schoen, 2020)

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad \textbf{(Eq. 10)}$$

- *ROC – AUC Curve and Score*

The ROC curve is a performance metric for classification models that can be calculated for different threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR). TPR, also known as sensitivity or recall, calculates the True Positives for all truth values (refer Eq. 11, whereas the calculation of FPR is done by taking the difference of Specificity from 1 (refer Eq. 12).

$$TPR = Sensitivity = \frac{TP}{TP + FN} \quad \textbf{(Eq. 11)}$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \quad \textbf{(Eq. 12)}$$

Area Under Curve (AUC) is a measure of separability. The higher the curve, the better the predictions of the model. Figure 6 shows an example of ROC-AUC for models A, B and C. As the graph illustrates, it is quite evident that model A has performed the best since it has the largest AUC.

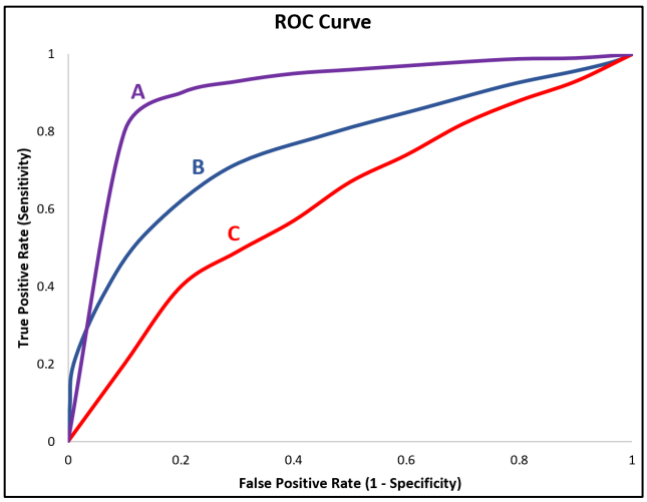

Figure 5: Example of ROC-Curve with AUC (Zach, 2021)

5. Results and Discussion

In this section, the results of the aforementioned models are interpreted followed by a comparative analysis of this paper with similar researches. The results for the GridSearchCV indicating the best hyperparameter choices for each model along with its pre-eminent scores are shown in Table 3.

Model	Best Parameters	Best Score
Bernoulli NB	'alpha' : 0.00001, 'binarize' : 0.0, 'fit_prior' : True	0.988750
Multinomial NB	'alpha' : 0.1, 'class_prior' : [0.7, 0.3], 'fit _prior' : True	0.992500
Gaussian NB	'var_smotthning' : 0.008111308307896872	0.984062

Table 3: Results from GridSearchCV

Analyzing these outcomes, the Multinomial Naïve Bayes model ostensibly yielded the best scores. Using the predictions obtained from the Grid Search, all these models are further evaluated using the various methods mentioned in Section 4.7. For each model, the classification report summarizing the accuracy, precision, recall and f1-score is illustrated below:

	precision	recall	f1 - score	support
Not Spam	0.985112	1.000000	0.992500	397
Spam	1.000000	0.985112	0.992500	403
Accuracy			0.992500	800
macro avg	0.992556	0.992556	0.992500	800
weighted avg	0.992612	0.992500	0.992500	800

Table 4: Classification Report for Bernoulli Naïve Bayes

	precision	recall	f1 - score	support
Not Spam	0.992500	1.000000	0.996236	397
Spam	1.000000	0.992556	0.996264	403
accuracy			0.996250	800
macro avg		0.996278	0.996250	800
weighted avg	0.996278	0.996250	0.996250	800

Table 5: Classification Report for Multinomial Naïve Bayes

	Precision	recall	f1 - score	support
Not Spam	0.989691	0.967254	0.978344	397
Spam	0.968447	0.990074	0.979141	403
accuracy			0.978750	800
macro avg	0.979069	0.978664	0.978743	800
weighted avg	0.978989	0.978750	0.978746	800

Table 6: Classification Report for Gaussian Naïve Bayes

To visualize a more comprehensible representation of the metric scores for each model, a bar graph (shown in Figure 7) has been plotted where the models and the scores are denoted by the x and y axes respectively.

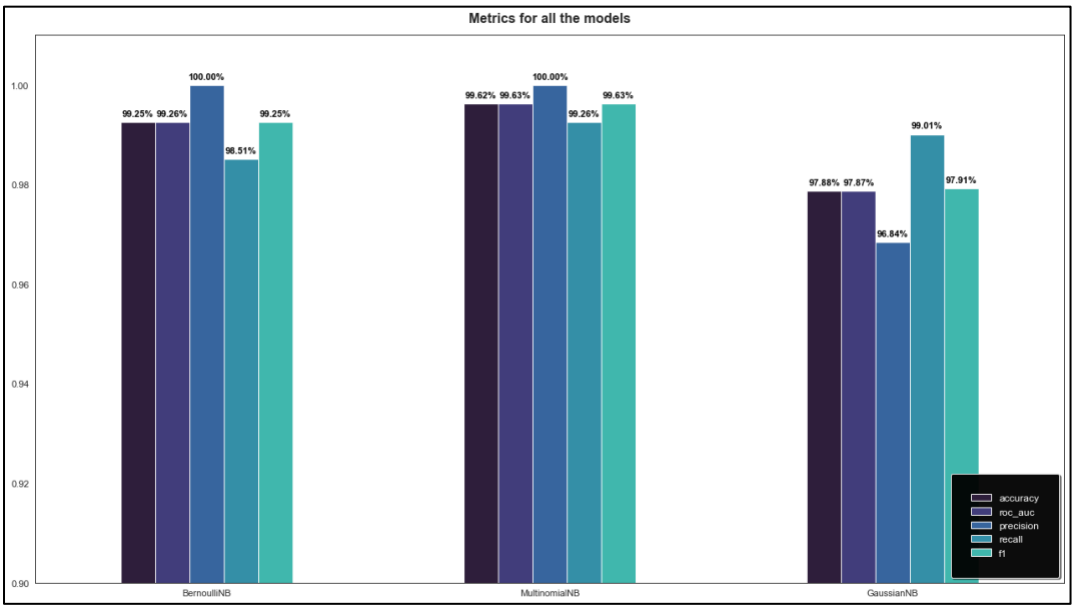

Figure 7: Comparison of all metrics for each model

On examining the above figure and tables, it can be deduced that Multinomial outperforms both Bernoulli and Gaussian Naïve Bayes algorithms. This is due to the fact that the overall scores for its accuracy, ROC-AUC, precision, recall, and f1-score are higher in comparison to that of the other two models. In addition to this, the ROC curve plotted using True Positive Rate and False Positive Rate is given in Figure 8. The curve which is the closest to TPR of 1 has the highest AUC score belonging to the Multinomial NB model, further supporting the previously mentioned claim.

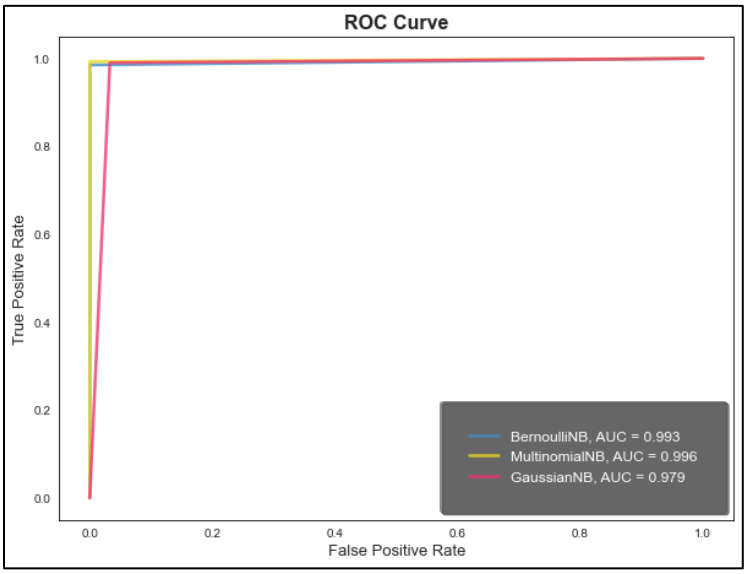

Figure 8: ROC Curve along with the AUC scores for each model

A semantic and rule-based model via supervised learning was suggested by Abiramasundari, Ramaswamy and Sangeetha (2021). In this paper, Rule Based Subject Analysis (RBSA) and Semantic Based Feature Selection (SBFS) methods are integrated with classification models such as Support Vector Machine (SVM), Multinomial, Bernoulli and Gaussian Naïve Bayes. These models were tested on the Enron dataset where the highest accuracy of 97% was observed for the SVM model.

Another approach based on semantic methods using efficient email classification is proposed by Bahgat, Rady and Gad (2017). In this study, several classification models were used to predict the outcome of spam and ham emails for the Enron dataset in an efficient manner. These include Naïve Bayes, Logistic Regression, SVM, J48, Random Forest and Radial Basis Function network. Out of the mentioned algorithms, SVM was identified as the best performing algorithm with an accuracy of 94%.

Comparing the above results to the outcomes produced in this paper, the Multinomial Naïve Bayes model outperforms both the SVM models with an accuracy of 99.33% and a ROC-AUC score of 99.34%.

6. Conclusion and Future Recommendations

The current problem statement, email spam detection, has risen to become one of the most demanding research topics due to an increase in cyber crime with recent advancements in technology. Various classification methods have been developed and trained upon different datasets for this purpose using both machine learning and deep learning techniques combined with a wide variety of feature selection methods, preprocessing approaches, optimization algorithms, etc., as discussed in Section 3, which are all factors that together contribute to the performance of the model. However, it is very difficult to achieve a model that is 100% accurate and perfectly filters out spam emails due to the numerous complexities of the English language such as sarcasm, evolving terminologies in slang among others. Nevertheless, several models have already been developed with considerably high performance scores. To conclude the findings of this paper, out of the three Naive Bayes models that were used to perform the classification on the 4000 emails in the dataset, the Multinomial NB model proved to be the best classifier with an accuracy of 99.34%, with Bernoulli NB and Gaussian NB following not too far behind with scores of xx.x% and xx.x% respectively. Other evaluation parameters used to assess the performance of the models included precision, recall, and f1-score. It is believed that the reason for such high accuracies across all NB models was due to the use of the most optimal parameters obtained from the output of the hyper-parameter tuning algorithm and the initial lemmatization using POS tags of the corpus. Additionally, the identification and removal of overlapping words between the two classes, namely spam and ham, is also believed to be a contributing factor as to the high accuracy of the Bayesian models.

Needless to say, there is plenty of room for improvement for any proposed model. Spammers are getting more and more intelligent in the way they phrase their messages, making them seem as legitimate as possible in order to scam individuals and organizations. The datasets that were used to train the proposed model included spam emails that used a formal tone. Other types of spam messages may include informal sentiments, sarcasm, or other non-professional slang, which is unlike the spam messages the model in this study has been trained on. In such scenarios, the model in question would perform poorly and would not

yield the same accuracies as compared to professional-toned emails. To resolve this issue, the model would be needed to be trained on different spam datasets, containing both formal and informal sentiments. This would ensure the model is able to accurately classify any kinds of spam email it comes across in the future regardless of the level of formality. Other popular spam datasets that are commonly used in anti-spam filtering systems include Spambase, Ling-Spam dataset, and SpamAssassin, to name a few. All these datasets could be combined into one corpus, as other research papers have done previously to further improve the performance of the model. The feature extraction and selection method plays a fundamental role in the performance of the classification. As explained previously in Section 4.3, TF-IDF only takes into consideration the frequency of the words. Using word2vec instead would consider the context of each of the words in the corpus, which would allow the model to perform more generically even on different datasets. To further improve on this, transformer-based models such as BERT or ELMo can be used as they are capable of considering multiple contexts for a particular word, a functionality that is not available in word2vec. Moreover, machine learning models tend to have a directly proportionate relationship between their performance and the size of the dataset. The model in this study was trained upon a dataset of 4000 emails. In the future, another factor that could be considered for the improvement of the system could be increasing the number of emails it is trained upon. It may not be possible to achieve a perfect model, but every advancement towards the improvement of systems like spam detection systems is worth working on as it is a sensitive field which could easily lead to disastrous consequences.

7. References

Abiramasundari, S., Ramaswamy, V., & Sangeetha, J. (2021). (rep.). *Spam filtering using Semantic and Rule Based model via supervised learning.* Retrieved from *https://www.annalsofrscb.ro/index.php/journal/article/view/1405/1174.*

Andre, L. (2021, June 15). *You are on the internet almost daily. You check your email, send replies, maybe browse websites and even click.* Financesonline.com; FinancesOnline.com. *https://financesonline.com/how-much-data-is-created-every-day/*

Bahgat, E. M., Rady, S., Gad, W., & Moawad, I. F. (2018). *Efficient email classification approach based on semantic methods*. Ain Shams Engineering Journal, 9(4), 3259–3269. *https://doi.org/10.1016/j.asej.2018.06.001*

Chowdhury, S., & Schoen, M. P. (2020, October 2). *Research Paper Classification using Supervised Machine Learning Techniques*. ResearchGate; unknown. *https://www.researchgate.net/publication/346853360_Research_Paper_Classification_using_Supervised_Machine_Learning_Techniques?enrichId=rgreq-1c98526e9543b39049d181d973ede8c2-XXX&enrichSource=Y292ZXJQYWdlOzM0Njg1MzM2MDtBUzoxMDUxMTgzNTY1MjY2OTQ1QDE2Mjc2MzMxMDA0ODI%3D&el=1_x_3&_esc=publicationCoverPdf*

Das, B., & Chakraborty, S. (n.d.). *An Improved Text Sentiment Classification Model Using TF IDF and Next Word Negation.* *https://arxiv.org/pdf/1806.06407.pdf*

Divya Khyani, & S, S. B. (2021, January 7). *An Interpretation of Lemmatization and Stemming in Natural Language Processing.* ResearchGate; unknown. *https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing#:~:text=What%20is%20Lemmatization%3F,adds%20meaning%20to%20particular%20words*.

Garg, B. (2022). *Design and Development of Naive Bayes Classifier*. Ndsu.edu. *https://doi.org/http://hdl.handle.net/10365/23048*

Hassan, D. (2017). *Investigating the Effect of Combining Text Clustering with Classification on Improving Spam Email Detection.* In Intelligent Systems Design and Applications (Vol. 557, pp. 120–128). essay, Springer International Publishing AG 2017.

Hassan, M., Mirza, W., & Hussain. (2017). *Header Based Spam Filtering Using Machine Learning Approach*. International Journal of Emerging Technologies in Engineering Research (IJETER), 5. *https://www.ijeter.everscience.org/Manuscripts/Volume-5/Issue-10/Vol-5-issue-10-M-21.pdf*

Kim, S.-W., & Gil, J.-M. (2019). *Research paper classification systems based on TF-IDF and LDA schemes*. Human-Centric Computing and Information Sciences, 9(1). *https://doi.org/10.1186/s13673-019-0192-7*

Kumari, A., & M. Shashi. (2019, August). *Vectorization of Text Documents for Identifying Unifiable News Articles*. ResearchGate; SAI Organization. *https://www.researchgate.net/publication/334884108_Vectorization_of_Text_Documents_for_Identifying_Unifiable_News_Articles*

Metsis, V., & Paliouras, G. (2006). *Spam Filtering with Naive Bayes -Which Naive Bayes? *.* *https://userweb.cs.txstate.edu/~v_m137/docs/papers/ceas2006_paper_corrected.pdf*

Muhammad, A. N., Bukhori, S., & Pandunata, P. (2019). *Sentiment Analysis of Positive and Negative of YouTube Comments Using Naïve Bayes – Support Vector Machine (NBSVM) Classifier*. 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE). *https://doi.org/10.1109/icomitee.2019.8920923*

Nur Hayatin, Gita Indah Marthasari, & Nuarini, L. (2020). *Optimization of Sentiment Analysis for Indonesian Presidential Election using Naïve Bayes and Particle Swarm Optimization.* Jurnal Online Informatika, 5(1), 81–88. *http://join.if.uinsgd.ac.id/index.php/join/article/view/558/148*

Pannakkong, W., Thiwa-Anont, K., Singthong, K., Parthanadee, P., & Buddhakulsomsiri, J. (2022, January). *Hyperparameter Tuning of Machine Learning Algorithms Using Response Surface Methodology: A Case Study of ANN, SVM, and DBN.* Mathematical Problems in Engineering, 2022, 1–17. *https://doi.org/10.1155/2022/8513719*

Parmar, N., Sharma, A., Jain, H., & Amol Krishnat Kadam. (2020) 10th edn, *Email Spam Detection using Naïve Bayes and Particle Swarm Optimization.* 10th edn. rep. Pune, Maharashtra: INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY, pp. 1–7.

Raschka, S. (2014). *Naive Bayes and Text Classification I Introduction and Theory*. *https://arxiv.org/pdf/1410.5329.pdf*

Rish, I. (2001). *An Empirical Study of the Naïve Bayes Classifier*. ResearchGate; unknown. *https://www.researchgate.net/publication/228845263_An_Empirical_Study_of_the_Naive_Bayes_Classifier*

Sahmoud, T., & Mikki, M. (2022). *Spam Detection Using BERT*. Retrieved November 30, 2022, from *https://arxiv.org/ftp/arxiv/papers/2206/2206.02443.pdf*

Shekhar, S., Bansode, A., & Salim, A. (2022, January). *A Comparative study of Hyper-Parameter Optimization Tools*. *https://arxiv.org/pdf/2201.06433.pdf*

Soni, A.N. (2019) 5th edn, *Spam e-mail detection using advanced deep convolution neural network algorithms*. 5th edn. rep. Campbellsville, Kentucky: JOURNAL FOR INNOVATIVE DEVELOPMENT IN PHARMACEUTICAL AND TECHNICAL SCIENCE.

Text Classification: What it is And Why it Matters. (2014). MonkeyLearn. *https://monkeylearn.com/text-classification/*

Wisam Abdulazeez Qader, Musa M.Ameen, & Bilal Ismael Ahmed. (2019, June). *An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges*. ResearchGate; unknown. *https://www.researchgate.net/publication/338511771_An_Overview_of_Bag_of_WordsImportance_Implementation_Applications_and_Challenges*

Yang, X., Yang, K., Cui, T., & He, L. (2022, February 11). *A Study of Text Vectorization Method Combining Topic Model and Transfer Learning.* ResearchGate; unknown. *https://www.researchgate.net/publication/358585623_A_Study_of_Text_Vectorization_Method_Combining_Topic_Model_and_Transfer_Learning*

Zach. (2021, August 9). *How to Interpret a ROC Curve (With Examples)* - Statology. Statology. *https://www.statology.org/interpret-roc-curve/ [Original source: https://studycrumb.com/alphabetizer]*