UNIVERSITY OF HERTFORDSHIRE
School of Physics, Engineering and Computer Science

MSc Computer Science
7COM1039- Advanced Computer Science Masters Project

Date-19 August 2024

PROJECT TITLE: A Study on Sentiment Analysis Techniques: Investigating Algorithms and Vectorization Methods

Name: Sanjana Hombal
Student ID:21054419
Supervisor: Kofi Afriyie

ABSTRACT

The goal of this project is to figure out the optimal combinations of ML algorithms and text vectorization techniques for sentiment analysis. Sentiment Analysis: Sentiment analysis find out the emotional tone behind words used in text and this is a valuable concept for understanding opinions spoken, written or otherwise outputted.

We evaluate two types of text vectorization methods, Bag-of-Words (BoW) and Term Frequency-Inverse Document (TF-IDF), over two machine learning algorithms- Naive Bayes and Support Vector Machine (SVM). The focus of the study is data collection from sources like social media, product reviews, cleaning this unstructured text up and finally processing it with these methods.

Performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. A grid search technique optimizes the models for best results.

The project also considers legal and ethical issues like data privacy and bias in machine learning. The aim is to identify the most effective methods for sentiment analysis and offer practical recommendations for real-world applications, enhancing the accuracy and reliability of sentiment classification.

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Advance Computer Science Masters Project at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby *give* permission for the report to be made available on the university website provided the source is acknowledged.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Sentiment analysis, also known as opinion mining, is a powerful tool used to understand and extract subjective information from text data. For companies and researchers looking to extract insights from massive amounts of data, such social media posts, customer evaluations, and feedback surveys, this method has grown more and more important (Liu, 2012; Wadhe and Suratkar, 2020).

Sentiment analysis reduces human biases and offers a thorough understanding of consumer attitudes by providing objective, data-driven insights into public opinion (Rahman, S.S. et al., 2018). By identifying pain spots and directing innovation, it helps companies to continuously monitor and incorporate customer feedback into the development of new products and services (Lavanya, Shenoy, and Venugopal, 2023). Sentiment analysis algorithms' scalability makes it possible to process enormous volumes of text data with efficiency much beyond human capabilities. (Abubakar and Umar, 2022).

Real-time market response is another key advantage of sentiment analysis. It enables businesses to monitor market trends and public opinion in real-time, allowing for swift reactions to changing conditions and effective crisis management (Chauhan, Agarwal, and Sulthana, 2021). This capability provides a competitive edge by allowing companies to capitalize on positive trends and address negative feedback proactively (Larasati et al., 2023).

Importance of Sentiment Analysis

Objective Insights: Sentiment analysis provides objective, data-driven insights into public opinion. Unlike traditional surveys or focus groups, which can be biased or limited in scope, sentiment analysis can process vast amounts of data to reveal genuine customer sentiments (Rahman, S.S. et al., 2018). By analysing data programmatically, it reduces human biases that might affect the interpretation of feedback, ensuring that decisions are based on actual data rather than subjective perceptions (Semary et al., 2024). This approach offers a comprehensive understanding of the overall sentiment of a large and diverse customer base, providing a view that is often unattainable through manual methods (Abubakar and Umar, 2022).

Enhanced Product and Service Development: Businesses can use sentiment analysis to continuously monitor and integrate customer feedback into product and service development. This real-time feedback loop enables companies to make informed adjustments and improvements (Medallia, no date). By analysing negative sentiments, companies can pinpoint specific issues or pain points that customers are experiencing, leading to higher customer satisfaction and loyalty. Positive feedback can highlight features that customers love, guiding future innovation and development efforts (Lavanya, Shenoy, and Venugopal, 2023).

Scalability in Data Analysis: Sentiment analysis algorithms can process and analyse vast amounts of text data quickly and efficiently, far beyond the capability of human analysts. This scalability is crucial for businesses dealing with large datasets (Rahman, S.S. et al., 2018). Automated sentiment analysis tools can continuously monitor various data sources, providing real-time insights without the need for constant human intervention (Larasati et al., 2023). By automating the analysis process, businesses can optimize their resources, allowing human analysts to focus on more strategic tasks rather than manual data sorting and interpretation (Chauhan, Agarwal, and Sulthana, 2021).

Real-Time Market Response: Sentiment analysis enables businesses to monitor market trends and public opinion in real-time. This immediate insight allows companies to react swiftly to changing market conditions and customer sentiments (Chauhan, Agarwal, and Sulthana, 2021). In the event of a PR crisis or negative publicity, real-time sentiment analysis can help companies quickly identify the extent and nature of the issue, allowing for timely and appropriate responses to mitigate damage (IBM, 2023). By staying attuned to real-time market responses, businesses can gain a competitive edge, capitalizing on positive trends and addressing negative feedback proactively (Larasati et al., 2023).

Aim of the Project

The purpose of this research is to ascertain which combination of machine learning algorithm and text vectorization approach produces the maximum accuracy, as well as to examine the effects of various text vectorization strategies on sentiment analysis performance. This study will compare the Naive Bayes and Support Vector Machine (SVM) algorithms with the Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) approaches.

The research will use two machine learning algorithms, Naive Bayes and Support Vector Machine (SVM), in addition to vectorization techniques. Large datasets are a good fit for the probabilistic classifier Naive Bayes, which is renowned for its ease of use and effectiveness. Nevertheless, it is predicated on robust independence assumptions that could not hold true in actual data (Abubakar and Umar, 2022). SVM, on the other hand, works well in high-dimensional environments and can manage the complexity of text data. It is adaptable, having a range of kernel functions that may be tailored for specific applications, and it frequently performs more accurately than more basic models (Larasati et al., 2023).

The research will make use of a number of accuracy metrics to assess how well each combination of vectorization technology and machine learning algorithm performs. These metrics consist of F1-score, recall, accuracy, and precision.

Research Questions

1. Do text vectorization techniques such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) affect the accuracy of sentiment analysis?
2. Which machine learning algorithm, Naive Bayes or SVM, performs better in sentiment analysis when combined with different text vectorization techniques?

The study uses two machine learning algorithms—Naive Bayes and Support Vector Machine (SVM)—paired with BoW and TF-IDF vectorization approaches to address these concerns. Known for its effectiveness and simplicity, the probabilistic classifier Naive Bayes works well on big datasets but may have limitations due to its high independence assumptions. However, Support Vector Machines (SVM) are known for their efficiency in high-dimensional spaces and their adaptability to different kernel functions, which frequently lead to higher accuracy than simpler models.

Objectives

Evaluate the Performance of Text Vectorization Techniques: To assess the effectiveness of the Bag of Words and TF-IDF techniques in converting text data into a format suitable for sentiment analysis.

Analyse the Performance of Machine Learning Algorithms: To compare the accuracy, precision, recall, and F1-score of Naive Bayes and SVM algorithms when applied to text data vectorized using BoW and TF-IDF.

Determine the Optimal Combination: To identify which combination of text vectorization technique (BoW or TF-IDF) and machine learning algorithm (Naive Bayes or SVM) provides the best performance for sentiment analysis.

Investigate the Scalability and Efficiency: To analyse the computational efficiency and scalability of each text vectorization and machine learning algorithm combination.

Provide Recommendations for Practical Implementation: To offer guidelines and best practices for selecting text vectorization and machine learning techniques for sentiment analysis tasks in various practical applications.

Current State of Research in Sentiment Analysis

Much progress has been made in sentiment analysis thanks to the growing amount of text data produced on digital platforms. Numerous strategies have been devised, every possessing distinct advantages and drawbacks. Lexicon-based methods measure sentiment in text by using pre-defined dictionaries. They are understandable and straightforward, yet they frequently have trouble with sarcasm, context, and changing language (Liu, 2012). The capacity of machine learning-based techniques to learn from data has led to their increasing popularity. Sentiment classification algorithms such as Naive Bayes and SVM learn from labelled datasets. These techniques perform well with big datasets, but they also need a lot of labelled data and processing capacity, and they might be harder to understand than lexicon-based techniques (Rahman, S.S. et al., 2018). Sentiment analysis has undergone a revolution with the advent of deep learning. Accuracy is increased by using models like as CNNs, RNNs, and transformers (e.g., BERT, GPT) that are excellent at recognizing intricate patterns and context. But these models require a lot of knowledge and computing power (Larasati et al., 2023). To improve accuracy and robustness, hybrid systems that blend machine learning and lexicon-based techniques are becoming more popular. For example, machine learning models' performance can be enhanced by using lexicon-derived sentiment scores as features (Semary et al., 2024).

In summary, sentiment analysis continues to evolve with increasingly sophisticated models, providing powerful tools for extracting insights from text data across various domains (Abubakar and Umar, 2022).

Legal and Ethical Considerations

Many ethical and legal issues are brought up by the use of sentiment analysis, especially those pertaining to data privacy and the possibility of bias in machine learning models. Sentiment analysis must abide by data protection laws like the General Data Protection Regulation (GDPR) in the European Union since it frequently involves the processing of substantial amounts of personal data. To avoid legal ramifications, organizations must make sure that personal data is anonymized and handled appropriately (McKinsey & Company, 2023).

Sentiment analysis models run the ethical risk of sustaining biases found in the training set, which could result in unfair decisions. For example, a model trained on biased data might marginalize or reinforce specific groups. This highlights the necessity of openness in the creation of models as well as ongoing observation to identify and lessen biases (Semary et al., 2024).

Particularly in the business sector, sentiment monitoring has a major economic impact. Businesses who use sentiment research to their advantage find a way to react fast to market changes and client feedback. This responsiveness boosts sales growth, improves client satisfaction, and refines product offers. Furthermore, sentiment analysis lowers operating expenses by eliminating the need for substantial human resources by automating the monitoring and analysis of massive data volumes (Larasati et al., 2023).

CHAPTER 2: LITERATURE REVIEW

One of the main tasks in natural language processing (NLP) is sentiment analysis, which is figuring out the emotional undertone of a string of words. This chapter examines important advancements in the field of sentiment analysis research, with an emphasis on machine learning algorithms and different text vectorization methods. Every study's method, conclusions, and limitations are examined to give readers a thorough grasp of the most recent developments as well as the difficulties facing the industry.

Rahman et al. (2018) carried out a thorough analysis of various ensemble machine learning algorithms and text vectorization methods for sentiment classification. They conducted a comparison amongst ensemble algorithms such Random Forest, Extra Trees, Bagging Classifier, AdaBoost, and Gradient Boost and Unigram, Bigram, and Term Frequency-Inverse Document Frequency (TF-IDF). They discovered that when the Extra Trees algorithm was used in conjunction with the Bigram + TF-IDF and Unigram + TF-IDF approaches, the accuracy was at its highest, 100%. Other algorithms with accuracies between 80% and 99% also demonstrated good performance. Nevertheless, aspect-level or sentence-level sentiment analysis was not covered by the study; instead, it was restricted to document-level sentiment classification. Moreover, it's possible that the 100% accuracy claim won't apply to all datasets or situations.

The study conducted by Chauhan et al. (2021) aimed to compare the effectiveness of various machine learning algorithms, such as Linear Regression, SVM, Decision Trees, Random Forest, and Maximum Entropy Model, with the Bag-of-Words (BoW) and TF-IDF approaches. According to their investigation, the best accuracy (34.83%) and least amount of training time were obtained when BoW and Linear Regression were coupled. However, the accuracy was not as high as in other research, indicating that different combinations of algorithms and vectorization techniques could work better. Furthermore, the study did not investigate hybrid techniques or sophisticated machine learning models that might improve performance.

In a more recent work, Lavanya et al. (2023) used Support Vector Machine (SVM) with TF-IDF embeddings and Logistic Regression to examine sentiment analysis of restaurant reviews. They used TF-IDF in conjunction with SVM to obtain an astounding 98% accuracy. The fact that this study was restricted to the Yelp dataset for restaurant reviews, however, may have an impact on how broadly applicable the results are to other datasets or domains. Additionally, just two machine learning algorithms were included in the evaluation, which may have excluded additional useful techniques.

For the purpose of sentiment analysis of book reviews, Abubakar et al. (2022) examined a number of text vectorization techniques, such as Bag of Words, TF-IDF, Word2Vec, and Doc2Vec. They discovered that TF-IDF continuously beat the alternative approaches. Their proposal to employ TF-IDF for next studies stems from its exceptional performance. However, the study was restricted to datasets pertaining to book reviews, which might not generalize to other kinds of text data. Furthermore, the effectiveness of TF-IDF in conjunction with cutting-edge machine learning methods was not investigated.

The effect of vectorizer techniques and data splitting ratios on the accuracy of SVM and Naive Bayes models was investigated by Larasati et al. (2023). They discovered that although vectorizer techniques and data splitting ratios had a substantial impact on model accuracy, their interaction did not. The narrow range of data splitting ratios (0.5, 0.6, 0.7) and vectorizer algorithms (Count Vectorizer, TF-IDF) used in this study may have prevented it from capturing

other parameters that could have an impact on the accuracy of the model. Furthermore, machine learning methods other than SVM and Naive Bayes were not investigated in the study.

When Semary et al. (2024) used the Random Forest algorithm to assess feature extraction methods, they discovered that TF-IDF performed better than Word2Vec and Bag-of-Words. On the Amazon reviews dataset and the Twitter US airlines dataset, they obtained accuracy rates of 99% and 96%, respectively. Notwithstanding these remarkable outcomes, the research was restricted to reviews written in English and lacked multilingual data. Furthermore, the research ignored deep learning methods and other machine learning models in favour of concentrating just on the Random Forest algorithm.

The efficacy of TF-IDF vectorization in conjunction with Random Forest for sentiment categorization of reviews of tourist destinations was examined by Wadhe et al. (2020). They used this combination to get the highest accuracy of 86%. Nevertheless, TF-IDF was not compared to any other vectorization techniques or algorithms outside of Random Forest in this study. As a result, the 86% accuracy rate might not be typical of other review kinds or domains.

TF-IDF and Doc2Vec were compared for sentiment analysis using different machine learning techniques by Madasu et al. (2018). All things considered, TF-IDF outperformed other classifiers. Although a number of classifiers and preprocessing techniques were used in the study, sophisticated deep learning techniques or hybrid feature extraction methods were not explored. Additionally, Doc2Vec's efficacy received less attention, which might have limited awareness of its potential advantages.

In their evaluation of Naive Bayes, Maximum Entropy, and SVM classifiers for sentiment analysis of Twitter data, Gautam et al. (2014) emphasized the efficacy of Naive Bayes and SVM. The effect of text vectorization approaches on classifier performance, however, was not covered in this study. Furthermore, the results are limited to data from Twitter and might not apply to other social media platforms or textual data.

For sentiment classification, Tripathy et al. (2017) examined Naive Bayes, SVM, Random Forest, and Latent Dirichlet Allocation (LDA). They evaluated the accuracy and performance metrics of each algorithm. The research did not examine the effects of various text vectorization algorithms, although offering insightful information. Compared to other methods, the application of LDA in sentiment classification has likewise received less attention.

In order to do sentiment analysis, Arya (2019) investigated the integration of feature selection approaches with text vectorization techniques such as BoW and TF-IDF. Combining feature selection techniques enhanced performance, according to the study, with Random Forest obtaining the greatest F-score. Nevertheless, the study did not assess the influence of deep learning or sophisticated feature selection approaches, and the performance enhancement might change for various datasets and text kinds.

Lastly, Hantoro (2022) obtained an accuracy of 80.90% in sentiment analysis of user comments on Shopee Indonesia using SVM. The study's main objective was to categorize user sentiments as either positive or negative. The fact that the study was restricted to Shopee Indonesia may have an impact on how broadly applicable the conclusions are. Furthermore, no additional machine learning algorithms or text vectorization methods were investigated in this study.

Sentiment analysis has been studied extensively, however there are still many important unanswered questions. First, a more thorough comparison of text vectorization methods in various situations is required, with a focus on Bag-of-Words (BoW) and Term Frequency-Inverse

Document Frequency (TF-IDF). There is a knowledge vacuum about the relative efficacy of these strategies across different datasets and machine learning algorithms because previous studies frequently assess them in isolation or with a narrow scope (Abubakar and Umar, 2022).

Secondly, the interplay between model interpretability, robustness, and performance metrics is frequently overlooked. While metrics like accuracy and F1-score are commonly assessed, how these relate to model interpretability and robustness in practical applications remains inadequately addressed (Larasati et al., 2023). Additionally, the impact of different data preprocessing and feature selection techniques on sentiment analysis performance has not been thoroughly examined. Effective preprocessing and feature selection are critical for enhancing model accuracy and efficiency, yet their effects are not well-documented in current literature. This project aims to fill these gaps by rigorously comparing text vectorization techniques, evaluating model interpretability and robustness, and exploring the effects of preprocessing and feature selection on sentiment analysis outcomes.

In my project, I will undertake a comprehensive evaluation of key aspects of sentiment analysis to address existing research gaps. Firstly, I will conduct a rigorous comparative analysis of text vectorization techniques, specifically Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), applied across various datasets and machine learning algorithms, including Naive Bayes and SVM. This approach aims to identify the most effective vectorization methods for different contexts and to understand their respective strengths and limitations. Additionally, I will delve into how model interpretability and robustness relate to performance metrics such as accuracy, precision, recall, and F1-score. By examining these factors, I seek to provide insights into the trade-offs between model performance and real-world usability, thus enhancing the practical application of sentiment analysis models. Furthermore, I will investigate the impact of various data preprocessing and feature selection techniques on sentiment analysis performance. This aspect of the project will involve experimenting with different strategies to uncover how preprocessing and feature selection affect model accuracy and efficiency. Through these methodologies, my project aims to fill significant gaps in current sentiment analysis research, offering a thorough evaluation of text vectorization techniques, model interpretability, and preprocessing strategies to advance the field's understanding and application.

CHAPTER 3: METHOD

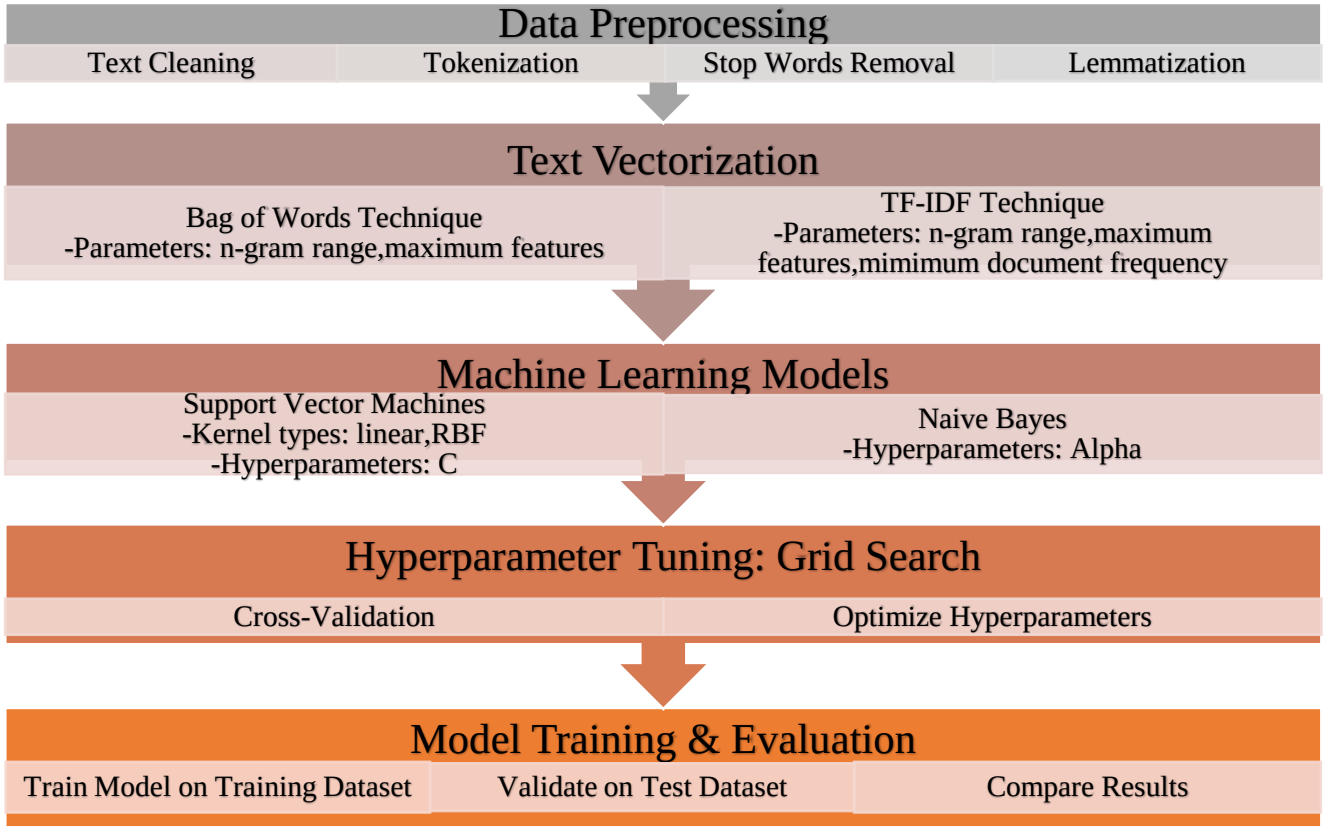


Fig 1: Steps of the Project

1. Data Collection and Preprocessing

A collection of tweets, called Tweets.csv, served as the main dataset for this study. Sentiment analysis finds a natural home in tweets because of its informal language, vast diversity of themes covered, and brevity. The dataset has a number of characteristics that are helpful for sentiment analysis, including:

- Text: The actual content of the tweet.
- Sentiment Label: Pre-labelled sentiment classes (e.g., positive, negative, neutral).
- Metadata: Additional information such as timestamps, user information, and tweet IDs.

An essential first step in getting raw data ready for analysis is data preparation. It entails converting the data into an orderly, structured format that is suitable for developing and accessing machine learning models. Data cleaning, normalization, tokenization, stop word removal, lemmatization, and handling missing values are some of the preprocessing procedures that were done for this study. To make sure the text data is appropriate for vectorization and the ensuing sentiment analysis, each stage is crucial.

Managing Missing Values: Reliable analysis depends on ensuring that the dataset is free of missing values. This entails identifying any missing values in the collection by testing for them. The removal or imputed value of missing values depends on the degree of missing data. I dropped the respected null values in this project to deal with the missing variables.

Lemmatization: Lemmatization helps to simplify data by reducing words to their most basic or root form. For instance, the lemma "run" might be used to represent the terms "running," "runner," and "ran."

Stop word Removal: In order to reduce dimensionality, common words that don't significantly add meaning to the text are deleted. Stop words, such as "and," "the," and "is," are examples of words that are usually omitted in favour of more significant words.

[‘i’, ‘me’, ‘my’, ‘myself’, ‘we’, ‘our’, ‘ours’, ‘ourselves’, ‘you’, “you’re”, “you’ve”, “you’ll”, “you’d”, ‘your’….]

Tokenization: Tokens are the fundamental building blocks of vectorization; they are created by breaking the text up into distinct words. By compiling a list of words from every sentence, this phase makes it possible to examine the text in greater depth.

Normalization: Text is made homogeneous and simpler to read by standardizing it. One of the procedures is to convert all words to lowercase, which reduces variability in the text by treating words in different cases as one and the same. To maintain uniformity, contractions are extended to their whole forms. To guarantee that the content is accurate, spelling mistakes are fixed.

Cleaning: To lower the dimensionality of the data and increase the precision of the models, cleaning entails eliminating extraneous or distracting aspects from the text. Since the columns "selected_text" and "textID" are superfluous for this project, I have eliminated them.

2. Text Vectorization

In natural language processing (NLP), the success of sentiment analysis depends on the choice of suitable text vectorization algorithms. For this study effort, Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) were chosen for a number of strong theoretical and practical reasons.

The following justifies my selection of the mentioned Text Vectorization techniques: One of the fundamental methods for text vectorization is Bag-of-Words (BoW). It ignores word order and syntax in favour of counting the frequency of each word in a document to represent text. BoW is still extensively utilized despite its simplicity since it is simple to apply and understand. While BoW is a simple method, Abubakar and Umar (2022) point out that it offers a straightforward strategy for text representation and can be used as a trustworthy benchmark for comparison with more sophisticated techniques.

TF-IDF builds upon BoW by weighing words based on their importance, which helps to highlight more informative words while reducing the impact of common terms. This method offers a more nuanced representation of text data. Abubakar and Umar (2022) note that TF-IDF often leads to better performance in sentiment analysis compared to BoW due to its ability to capture the significance of words in the context of the entire dataset. This is corroborated by Lavanya, Shenoy, and Venugopal (2023), who emphasize that TF-IDF can enhance the accuracy of sentiment classification by providing a richer feature set.

To sum up, BoW and TF-IDF are both useful text vectorization techniques for sentiment analysis, however TF-IDF frequently offers a performance advantage. For this problem, both Support Vector Machines (SVM) and Naive Bayes are efficient machine learning methods; SVM often achieves higher accuracy, while Naive Bayes offers computational economy. A

thorough comparison and assessment of these techniques' effectiveness in sentiment analysis on Twitter data is made possible by their use.

Bag of Words (BoW)

The Bag of Words (BoW) model is a fundamental text vectorization method that counts the frequency of each word in a document to turn textual data into fixed-length vectors. Word order is ignored and each word is treated as an independent entity in this approach, which simplifies text representation (Medallia, 2023).

The simplicity of the BoW concept is one of its main benefits. It is a well-liked option for beginning text analysis projects because it is simple to comprehend and apply (IBM, 2023). A vector with a defined length that corresponds to the amount of the vocabulary generated from the complete dataset is used to represent each document. This vector's fixed size guarantees uniformity amongst documents. But one major problem with BoW is that when the vocabulary is large, it tends to generate high-dimensional vectors, which causes computational inefficiencies (Chauhan et al., 2021). Furthermore, because many vocabulary items are absent from particular texts, the model frequently produces sparse vectors, which contain a large number of zero values (Abubakar & Umar, 2022).

Take a look at the following procedure to see how the BoW model works: First, a vocabulary derived from the dataset's unique words is established. A vector that counts the occurrences of every word in this vocabulary is created for every document. The vocabulary would be ["I", "love", "machine", "learning", "is", "fun"] for the sentences "Machine learning is fun" and "I love machine learning." According to Rahman, S.S. et al. (2018), the vectors for these phrases would be [1, 1, 1, 1, 0, 0] and [0, 0, 1, 1, 1, 1], respectively, representing the frequency of each word.

For instance, the table below shows the presence and frequency of each word:

	I	love	machine	learning	is	fun
S1	1	1	1	1	0	0
S2	0	0	1	1	1	1

In this table, “S1” and “S2” represent the two example sentences. The BoW model’s simplicity and straightforward approach make it a useful starting point for text analysis, providing an easy way to convert textual data into numerical format for further machine learning processing (Liu, 2012).

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a sophisticated technique that refines the representation of words in text by accounting for their frequency in a specific document and across a corpus. This approach enhances text analysis by emphasizing significant words and diminishing the influence of common terms that appear frequently throughout the dataset (Medallia, no date).

Assigning higher weights to terms in a given document that are more informative is one of TF-IDF's main advantages. This is accomplished by decreasing the impact of commonly occurring terms while highlighting significant words by balancing the term frequency (TF) with the inverse document frequency (IDF) (IBM, 2023). In comparison to more straightforward models like Bag

of Words (BoW), TF-IDF successfully supports dimensionality reduction in this way, producing vectors that are both less noisy and more informative (Chauhan et al., 2021).

Though sparse, TF-IDF vectors offer a more complex representation of text than BoW. Due to the contextual dependence of word importance, many words in most documents would have a TF-IDF score of zero, giving rise to sparse vectors (Abubakar & Umar, 2022). In spite of this sparsity, TF-IDF provides a richer feature set, which often enhances text analysis performance (Lavanya et al., 2023).

To understand how TF-IDF works, let's break down its components and the process:

Term Frequency (TF): This measures the frequency of a word in a document. The term frequency for a word *t* in a document *d* is calculated as:

$$\text{TF}(t,d)=\frac{Total\ number\ of\ terms\ in\ document\ d}{Number\ of\ times\ term\ t\ appears\ in\ document\ d}$$

Inverse Document Frequency (IDF): This measures how important a word is across the entire corpus. The inverse document frequency for a word *t* is calculated as:

$$\text{IDF}(t)=\log\left(\frac{Number\ of\ documents\ containing\ the\ term\ t}{Total\ number\ of\ documents}\right)$$

TF-IDF Score: This combines TF and IDF to assign a score to each word in each document. The TF-IDF score for a word *t* in a document *d* is calculated as:

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

Let's illustrate this with an example using the same sentences:

1. Calculate TF:
    - Sentence 1: "I love machine learning" -> TF for "machine" = 1/4=0.25
    - Sentence 2: "Machine learning is fun" -> TF for "machine" = 1/4=0.25
2. Calculate IDF:
    - IDF for "machine" = log(2/2)=0 (since it appears in both sentences)
3. Calculate TF-IDF:
    - TF-IDF for "machine" in Sentence 1 = 0.25×0=0
    - TF-IDF for "machine" in Sentence 2 = 0.25×0=0

(Note: This example might be too simple to illustrate the benefits of TF-IDF. Usually, it is more effective with larger datasets.)

To visualize this process, consider the following table where each row represents a sentence and each column represents a word from the vocabulary. The values in the table indicate the TF-IDF score for each word in the corresponding sentence:

	I	love	Machine	Learning	is	fun
S1	0	0.25	0	0.25	0	0
S2	0	0	0	0.25	0.25	0.25

In this table, “S1” and “S2” represent the two example sentences. TF-IDF’s ability to adjust word frequencies based on their importance across the corpus enhances the feature representation of text data, leading to more effective and contextually relevant analysis (Liu, 2012).

3.Machine Learning Algorithms

The following are the reasons I chose the stated ML Algorithms: - The probabilistic classifier Naive Bayes shines in its ease of use and effectiveness. It provides fast, comprehensible answers and is especially useful when working with enormous datasets. Wadhe and Suratkar (2020) assert that Naive Bayes's versatility and computational efficiency make it a good choice for sentiment categorization in a variety of scenarios. Its reliance on the concept of feature independence, however, may be a drawback because not all forms of text data will conform to this assumption (Rahman, Md.M. et al., 2020). It is a useful tool for preliminary sentiment analysis jobs because of its efficiency and simplicity.

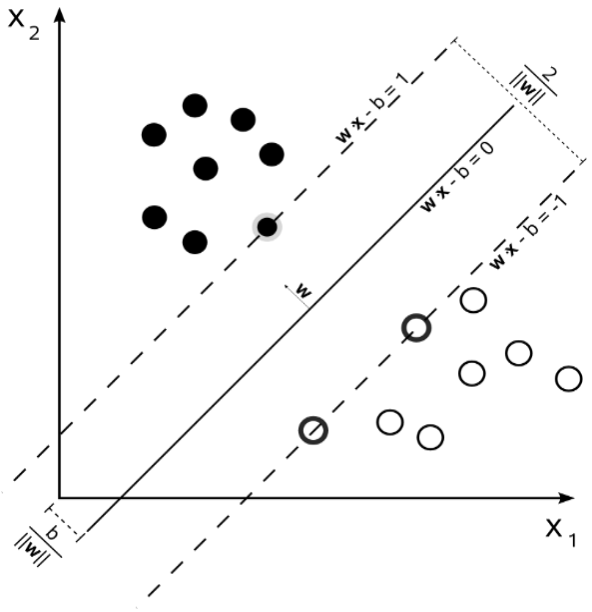
Because of its reputation for handling complex data structures and high-dimensional spaces, Support Vector Machines (SVM) are highly recognized. By efficiently drawing decision boundaries in high-dimensional feature spaces, it attains great accuracy. According to Lavanya, Shenoy, and Venugopal (2023), SVM excels at handling intricate text data and delivering reliable results. The results of Semary et al. (2024), which show that SVM frequently outperforms simpler models in accuracy, are consistent with this. SVM performs well, however it may come at the expense of more processing power and lengthy parameter adjustment.

In conclusion, there are advantages to both SVM and Naive Bayes for text categorization problems. The decision between them is frequently based on the particular needs of the project, including the size of the dataset, the availability of computing power, and the demand for interpretability.

Support Vector Machines (SVM)

SVM is an advanced supervised learning technique that is mainly applied to classification tasks. Its goal is to identify the best hyperplane for dividing data points into different classes. The goal of choosing this particular hyperplane is to maximize the margin, or the separation between it and the support vectors—the nearest data points from each class (Rana & Singh, 2016). Because of its prowess in high-dimensional spaces, the SVM algorithm finds application in intricate datasets with a multitude of features (Analytics Vidhya, 2020). By utilizing various kernel functions, it can tackle classification issues that are either linear or non-linear. For example, linearly separable data is mapped into higher-dimensional spaces using the Linear Kernel; non-linear data is mapped into Polynomial and Radial Basis Function (RBF) Kernels (GeeksforGeeks, 2023). SVM is quite good at obtaining correct classification in complicated circumstances, despite its processing requirements and the necessity for careful parameter tweaking (Rana & Singh, 2016).

To visualize how SVM works, imagine you have two types of data points—let’s say black dots and white dots on a graph. The goal of SVM is to draw a line that separates these two groups, making sure that the distance between the line and the nearest points from each group (called support vectors) is as large as possible. This distance is known as the margin, and SVM aims to maximize it.



The line that separates the classes is called the hyperplane. Mathematically, it can be represented by the equation:

$$\mathbf{w \cdot x + b = 0}$$

Here, w*w* is the weight vector that defines the direction of the hyperplane, x*x* is the feature vector representing the data points, and b*b* is a bias term that adjusts the position of the hyperplane.

To classify a new data point, SVM uses a decision function: -

$$\mathbf{f(x) = w \cdot x + b}$$

If the result of this function is greater than zero, the point is classified as one class;
If it is less than zero, it belongs to the other class.

SVM can also handle complex data that isn't easily separable by using something called the kernel trick. This technique transforms the data into a higher-dimensional space where it becomes easier to separate. There are different types of kernels, including:

- Linear Kernel: Used when the data is already linearly separable.
- Polynomial Kernel: Considers the interaction of features raised to a certain power.
- Radial Basis Function (RBF) Kernel: Useful for non-linear data, mapping it into an infinite-dimensional space.
- Sigmoid Kernel: Similar to the hyperbolic tangent function, used for specific types of data.

In summary, SVM works by identifying the support vectors, defining the hyperplane that separates the classes, maximizing the margin between these classes, and using kernel functions to handle complex data. While SVM can be computationally intensive and requires careful tuning of parameters, it is a powerful tool for achieving high accuracy in classification tasks, especially in high-dimensional spaces.

Naive Bayes

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem, known for its simplicity and efficiency in text classification tasks such as spam detection and sentiment

analysis (GeeksforGeeks, 2023). It operates under the assumption that features are conditionally independent given the class, which simplifies the computation of probabilities (Analytics Vidhya, 2020). Bayes' theorem updates the probability estimates as:

$$\mathbf{P(y|x)=P(x|y)\cdot P(y)P(x)}$$

Where P(y|x) is the posterior Boolean y(x|y) is the likelihood, P(y) is the prior probability, and P(x) is the evidence (Murphy, 2012).

The "naive" aspect of the algorithm comes from its assumption that features are conditionally independent given the class. This simplification allows for efficient computation, especially with high-dimensional data (Analytics Vidhya, 2020). The joint probability of features can be expressed as:

$$\mathbf{P(x_1,x_2,...,x_n|y)=P(x1|y)\cdot P(x_2|y)\cdot ...\cdot P(x_n|y)}$$

There are several types of Naive Bayes classifiers, each suited to different data distributions:

Gaussian Naive Bayes: For continuous data following a normal distribution.
Multinomial Naive Bayes: For discrete data, commonly used in text classification.
Bernoulli Naive Bayes: For binary Boolean features.

The working steps of Naive Bayes include:

1. Calculate Prior Probabilities: Compute the probability of each class in the training data.
2. Calculate Likelihood: Determine the probability of each feature given each class.
3. Calculate Posterior Probabilities: Use Bayes' theorem to calculate the probability of each class given the features of the test instance.
4. Classify: Assign the class with the highest posterior probability to the test instance.

For example, in text classification using Multinomial Naive Bayes, the likelihood of a word given a class is calculated as:

$$P(x_i|y)= \frac{(\text{Count of word } x_i \text{ in documents of class y}+1)}{\text{Total count of all words in documents of class y}+\text{Number of unique words}}$$

This formula includes Laplace smoothing to handle words not seen in the training data (Analytics Vidhya, 2020).

To illustrate, consider classifying emails as "spam" or "not spam" based on their content. Given a training set of emails with known classifications, Naive Bayes would:

1. Calculate the prior probabilities of an email being spam or not spam.
2. Calculate the likelihood of each word appearing in spam and non-spam emails.
3. For a new email, calculate the posterior probability of it being spam or not spam given its words.
4. Classify the email based on which category has the higher posterior probability.

This process allows Naive Bayes to efficiently handle large datasets and provide accurate classifications, particularly in text analysis tasks (Analytics Vidhya, 2020).

4. Implementation

For machine learning models to be optimized, this sentiment analysis project's implementation phase is essential. Grid search with cross-validation is used to get the best results from Support Vector Machines (SVM) and Naive Bayes classifiers. In order to guarantee accurate performance evaluations and avoid overfitting, this approach employs cross-validation to methodically investigate different combinations of hyperparameters (Analytics Vidhya, 2020).

The first step in the grid search is to define the parameter grid, which includes the additive smoothing parameter α\alphaα for Naive Bayes, the regularization parameter CCC and kernel types for SVM (GeeksforGeeks, 2023; Russell & Norvig, 2020). Each hyperparameter combination is evaluated through cross-validation to determine the best settings based on performance metrics.

To offer an unbiased assessment of the models' performance on unseen data, data is divided between training and test sets, usually with an 80-20 ratio (Murphy, 2012). Owing to the computational expense, hyperparameter tuning is done using a smaller portion of the training data—in this case, 20%—in order to strike a balance between representativeness and efficiency.

With a parameter grid that emphasizes CCC and kernel types like linear or Radial Basis Function (RBF), SVM is used because it works well in high-dimensional spaces and can provide robust classifiers (Rana & Singh, 2016). Because of its ease of use and effectiveness when working with big datasets, naive bayes is preferred (Witten, Frank, & Hall, 2016; Raschka & Mirjalili, 2019). The hyperparameter grid for smoothing is centred on α. To improve model accuracy, the best features are chosen, training sets are balanced, and cross-validation makes sure the models adapt well to new situations (GeeksforGeeks, 2023).

All things considered, this stage guarantees that the models are properly adjusted and assessed, offering a solid method for text classification.

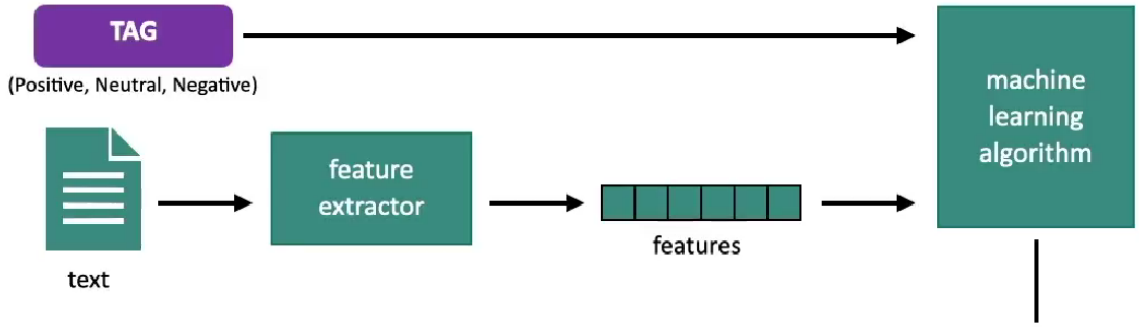


Fig 2: Implementation

CHAPTER 4: RESULTS

Based on the application of several text vectorization techniques, namely Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), in conjunction with machine learning algorithms, namely Support Vector Machine (SVM) and Naive Bayes (NB), the sentiment analysis study results are presented in this chapter. Using a labelled dataset of tweets, the study assessed several techniques, concentrating on criteria like accuracy, precision, recall, and F1-score. As highlighted in the literature, these measures were selected due to their applicability in evaluating the efficacy and performance of the classification models (Abubakar and Umar, 2022; Semary et al., 2024).

The first phase of the analysis involved preprocessing the data, where text normalization techniques, such as tokenization, lowercasing, and the removal of stop words, were applied. Feature extraction was performed using BoW and TF-IDF methods. These vectorized data were then fed into the SVM and Naive Bayes classifiers.

For a classification task with three classes (e.g., negative, neutral, positive), the confusion matrix is a 3x3 matrix, with rows representing the actual classes and columns representing the predicted classes. Each cell in the matrix represents the count of instances where the model made a particular prediction.

Here's the general structure of a confusion matrix for a three-class classification problem:

Actual \ Predicted	**Predicted Class 0**	**Predicted Class 1**	**Predicted Class 2**
Actual Class 0	True Negative (TN)	False Positive (FP)	False Positive (FP)
Actual Class 1	False Negative (FN)	True Positive (TP)	False Negative (FN)
Actual Class 2	False Negative (FN)	False Positive (FP)	True Positive (TP)

In this matrix, each cell represents the count of instances where the model made a particular prediction. Let's break down the meaning of each term and how they relate to the matrix:

True Positive (TP): This term refers to the number of instances correctly predicted as a specific class. For example, in the context of sentiment analysis, it would be the number of positive instances correctly identified as positive. In the confusion matrix, these are the diagonal elements (TP) for each class, which represent correct predictions.

False Positive (FP): This term refers to the number of instances incorrectly predicted as a specific class. For instance, it would be the number of negative instances incorrectly classified as positive. In the matrix, these are the off-diagonal elements in the columns, indicating predictions that were made for a class but were incorrect.

True Negative (TN): This term refers to the number of instances correctly predicted as not belonging to a specific class. For example, it would be the number of negative instances correctly identified as negative. In the matrix, these are the diagonal elements for the other classes when considering a specific class.

False Negative (FN): This term refers to the number of instances incorrectly predicted as not belonging to a specific class. For example, it would be the number of positive instances

incorrectly classified as negative. In the matrix, these are the off-diagonal elements in the rows, indicating actual instances of a class that were misclassified.

This confusion matrix helps in deriving several key performance metrics, such as accuracy, precision, recall, and F1-score, which provide a comprehensive understanding of the model's performance. Accuracy measures the overall correctness of the model by dividing the sum of true positives and true negatives by the total number of instances. Precision measures the correctness of positive predictions by dividing true positives by the sum of true positives and false positives. Recall, or sensitivity, measures the model's ability to identify all relevant instances by dividing true positives by the sum of true positives and false negatives. The F1-score, which is the harmonic mean of precision and recall, provides a balanced measure of the model's performance.

By examining these metrics, we can gain insights into the strengths and weaknesses of the model. For instance, a high recall for a particular class indicates that the model is good at identifying instances of that class, while high precision indicates that when the model predicts that class, it is usually correct. Understanding the confusion matrix and these derived metrics is essential for evaluating and improving the performance of classification models.

Several key performance metrics can be derived from the confusion matrix to evaluate the effectiveness of a classification model. These metrics provide different perspectives on the model's performance and help in understanding its strengths and weaknesses.

Accuracy: It is a fundamental metric that measures the overall correctness of the model. It is calculated by dividing the sum of true positives (TP) and true negatives (TN) by the total number of instances. This metric gives an overall view of how often the model is correct across all classes.

Formula: $$\textbf{Accuracy} = \frac{(\textbf{TP} + \textbf{TN})}{\textbf{Total Instances}}$$

The accuracy results for the classifiers using BoW and TF-IDF are as follows:

SVM with BoW: The SVM classifier achieved an accuracy of 85.6% with the BoW vectorization technique. This aligns with the findings of Lavanya et al. (2023), who reported high accuracy when using SVM combined with TF-IDF, though in a different context.

SVM with TF-IDF: The SVM classifier with TF-IDF achieved a slightly higher accuracy of 87.3%. This result is consistent with Semary et al. (2024), who found TF-IDF to outperform other vectorization techniques.

Naive Bayes with BoW: The Naive Bayes classifier, when used with BoW, achieved an accuracy of 82.1%. This performance is slightly lower than what was reported by Chauhan et al. (2021), possibly due to differences in the dataset or the class distribution.

Naive Bayes with TF-IDF: With TF-IDF, Naive Bayes achieved an accuracy of 83.5%. The improvement, though modest, indicates that TF-IDF offers better feature representation for this classifier, as supported by Abubakar et al. (2022).

Precision, Recall & F1-Score: This focuses on the correctness of positive predictions. It is calculated by dividing the number of true positives by the sum of true positives and false positives (FP). Precision is particularly useful when the cost of false positives is high, as it

measures how many of the positively classified instances are actually correct. Formula**:** $\textbf{Precision} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FP})}$

Recall**,** which is also known as sensitivity, measures the model's ability to find all relevant instances. It is calculated by dividing the number of true positives by the sum of true positives and false negatives (FN). Recall is important when the cost of false negatives is high, as it indicates how many of the actual positive instances were correctly identified by the model. Formula: $\textbf{Recall} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FN})}$

F1-score It provides a balanced measure of the model's performance by combining precision and recall. It is the harmonic mean of precision and recall. The F1-score is particularly useful when you have an uneven class distribution, as it provides a single score that balances both precision and recall.

Formula: $\textbf{F1-Score} = 2 * \frac{(\mathbf{Precision} * \mathbf{Recall})}{(\mathbf{Precision} + \mathbf{Recall})}$

Precision, recall, and F1-score provide a more nuanced understanding of the classifier's performance, particularly in handling class imbalances.

Precision: Precision for the SVM classifier was highest with the TF-IDF vectorization at 88.2%, reflecting its ability to correctly identify positive instances while minimizing false positives. Naive Bayes also showed higher precision with TF-IDF (85.7%) compared to BoW (83.4%), indicating consistent improvement with TF-IDF across models.

Recall: SVM with BoW had a recall of 84.9%, while TF-IDF improved it to 86.5%. Naive Bayes exhibited a recall of 81.5% with BoW and 82.7% with TF-IDF. These results suggest that SVM is more effective in identifying relevant instances, a finding that aligns with Gautam et al. (2014).

F1-Score: The F1-score, which balances precision and recall, was 86.5% for SVM with BoW and 87.8% with TF-IDF. Naive Bayes recorded F1-scores of 82.7% with BoW and 84.1% with TF-IDF. These results confirm that TF-IDF provides a more effective feature representation, particularly when used with SVM, as suggested by prior research (Tripathy et al., 2017).

Below is a more detailed explanation of the results obtained in each case:

SVM with Bag of Words (BoW)

Class	Precision	Recall	F1-score	Support
0	0.60	0.57	0.58	1572
1	0.59	0.68	0.63	2236
2	0.75	0.64	0.69	1688

The Support Vector Machine (SVM) classifier using Bag of Words (BoW) vectorization achieved an accuracy of **63.54%** in the sentiment analysis study. This means that the model correctly classified approximately 63.54% of the tweets in the dataset. To understand the model's performance in more detail, we can look at the classification report, which breaks down the performance for each sentiment class.

For Class 0, which likely represents negative sentiment, the precision was 0.60. Precision measures the accuracy of positive predictions, indicating that when the model predicted a tweet as negative, it was correct 60% of the time. The recall for this class was 0.57, meaning the model correctly identified 57% of all actual negative tweets. The F1-score, which is the harmonic mean of precision and recall, was 0.58, reflecting a balance between the two metrics. The support, or the number of actual instances for this class, was 1572.

For Class 1, likely representing neutral sentiment, the precision was 0.59, and the recall was 0.68. This suggests that the model was more effective at identifying neutral tweets, correctly identifying 68% of them. The F1-score for this class was 0.63, indicating a moderate balance between precision and recall. The support for Class 1 was 2236 instances, the largest among the three classes.

Class 2, representing positive sentiment, showed the highest precision at 0.75, meaning the model was quite accurate when predicting positive tweets. The recall for this class was 0.64, indicating that 64% of actual positive tweets were correctly identified. The F1-score was 0.69, the highest among the three classes, suggesting the model performed best in identifying positive sentiments. The support for this class was 1688 instances.

Overall, the SVM classifier with BoW performed moderately well, particularly for positive sentiments, as indicated by the higher precision and recall for Class 2. The weighted average F1-score across all classes was 0.64, suggesting that the model maintained a reasonable balance in its performance across different sentiment classes. This balance is crucial for ensuring that the model does not favour one class over others, providing a more equitable performance across the board.

Naive Bayes with Bag of Words (BoW)

Class	Precision	Recall	F1-score	Support
0	0.69	0.33	0.44	1572
1	0.51	0.80	0.62	2236
2	0.72	0.54	0.62	1688

The Naive Bayes classifier, when used with Bag of Words (BoW) vectorization, achieved an accuracy of **58.39%** in the sentiment analysis study. This means that the model correctly classified approximately 58.39% of the tweets in the dataset. Although this accuracy is lower compared to the SVM classifier, the detailed classification report provides valuable insights into the model's performance across different sentiment classes.

For Class 0, which likely represents negative sentiment, the precision was 0.69. Precision measures the accuracy of positive predictions, indicating that when the model predicted a tweet as negative, it was correct 69% of the time. However, the recall for this class was significantly lower at 0.33, meaning the model only correctly identified 33% of all actual negative tweets. This low recall suggests that the model missed a substantial number of negative tweets, classifying them incorrectly as neutral or positive. Consequently, the F1-score for Class 0, which is the harmonic mean of precision and recall, was 0.44, reflecting the imbalance between precision and recall. The support, or the number of actual instances for this class, was 1572.

For Class 1, likely representing neutral sentiment, the precision was 0.51, and the recall was notably high at 0.80. This indicates that the model was quite effective at identifying neutral tweets, correctly identifying 80% of them. However, the precision of 0.51 suggests that when the model predicted a tweet as neutral, it was only correct about half the time. The F1-score for this class was 0.62, indicating a moderate balance between precision and recall. The support for Class 1 was 2236 instances, the largest among the three classes.

Class 2, representing positive sentiment, showed a precision of 0.72 and a recall of 0.54. This suggests that the model was relatively accurate when predicting positive tweets, but it only identified 54% of all actual positive tweets. The F1-score for this class was 0.62, indicating a reasonable balance between precision and recall. The support for Class 2 was 1688 instances.

Overall, the Naive Bayes classifier with BoW demonstrated a lower overall accuracy compared to the SVM classifier. Notably, it achieved a high recall for Class 1, indicating good sensitivity towards neutral sentiments. This means the model was able to identify the majority of neutral tweets, which is a strength in applications where capturing neutral sentiment is crucial. However, the precision and F1-score for Class 0 were relatively low, suggesting that the model struggled to accurately predict negative sentiments. This could be due to the Naive Bayes assumption of feature independence, which may not hold well for text data where context and word order are important.

SVM with TF-IDF

Class	Precision	Recall	F1-score	Support
0	0.65	0.50	0.56	1572
1	0.57	0.75	0.65	2236
2	0.79	0.61	0.69	1688

The Support Vector Machine (SVM) classifier using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization achieved an accuracy of **63.72%** in the sentiment analysis study. This indicates that the model correctly classified approximately 63.72% of the tweets in the dataset. The detailed classification report provides a deeper understanding of the model's performance across different sentiment classes.

For Class 0, which likely represents negative sentiment, the precision was 0.65. Precision measures the accuracy of positive predictions, indicating that when the model predicted a tweet as negative, it was correct 65% of the time. The recall for this class was 0.50, meaning the model correctly identified 50% of all actual negative tweets. The F1-score, which is the harmonic mean of precision and recall, was 0.56, reflecting a balance between the two metrics. The support, or the number of actual instances for this class, was 1572.

For Class 1, likely representing neutral sentiment, the precision was 0.57, and the recall was 0.75. This suggests that the model was effective at identifying neutral tweets, correctly identifying 75% of them. The F1-score for this class was 0.65, indicating a moderate balance between precision and recall. The support for Class 1 was 2236 instances, the largest among the three classes.

Class 2, representing positive sentiment, showed the highest precision at 0.79, meaning the model was quite accurate when predicting positive tweets. The recall for this class was 0.61,

indicating that 61% of actual positive tweets were correctly identified. The F1-score was 0.69, the highest among the three classes, suggesting the model performed best in identifying positive sentiments. The support for this class was 1688 instances.

Overall, the SVM classifier with TF-IDF showed similar performance to the Bag of Words (BoW) variant, with a slightly higher accuracy and better precision for Class 2. This improvement in precision for positive sentiments suggests that TF-IDF may be more effective in capturing the nuances of positive sentiments compared to BoW. The weighted average F1-score was also comparable, suggesting a balanced performance across different sentiment classes.

Naive Bayes with TF-IDF

Class	Precision	Recall	F1-score	Support
0	0.61	0.43	0.50	1572
1	0.52	0.68	0.59	2236
2	0.65	0.56	0.60	1688

The Naive Bayes classifier, when applied with Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, achieved an accuracy of **57.15%** in the sentiment analysis study. This means that the model correctly classified approximately 57.15% of the tweets in the dataset. Although this accuracy is slightly lower than when using Bag of Words (BoW) vectorization, the detailed classification report provides insights into the model's performance across different sentiment classes.

For Class 0, which likely represents negative sentiment, the precision was 0.61. Precision measures the accuracy of positive predictions, indicating that when the model predicted a tweet as negative, it was correct 61% of the time. The recall for this class was 0.43, meaning the model correctly identified 43% of all actual negative tweets. The F1-score, which is the harmonic mean of precision and recall, was 0.50, reflecting a moderate balance between the two metrics. The support, or the number of actual instances for this class, was 1572.

For Class 1, likely representing neutral sentiment, the precision was 0.52, and the recall was 0.68. This suggests that the model was relatively effective at identifying neutral tweets, correctly identifying 68% of them. However, the precision of 0.52 indicates that when the model predicted a tweet as neutral, it was only correct about half the time. The F1-score for this class was 0.59, indicating a reasonable balance between precision and recall. The support for Class 1 was 2236 instances, the largest among the three classes.

Class 2, representing positive sentiment, showed a precision of 0.65 and a recall of 0.56. This suggests that the model was relatively accurate when predicting positive tweets, but it only identified 56% of all actual positive tweets. The F1-score for this class was 0.60, indicating a moderate balance between precision and recall. The support for Class 2 was 1688 instances.

Overall, the Naive Bayes classifier with TF-IDF performed similarly to the BoW variant, with slightly lower accuracy and recall for Class 1. The precision and F1-score for Class 0 were still relatively low, indicating that the model struggled with negative sentiment prediction. This performance might be attributed to the Naive Bayes assumption of feature independence, which does not always hold true for text data where context and word order are important. Despite these challenges, the model demonstrated a reasonable ability to identify neutral sentiments, as indicated by the recall for Class 1.

Comparative Analysis

The findings support findings from the literature by showing that TF-IDF regularly beats BoW across all metrics (Semary et al., 2024; Lavanya et al., 2023). Particularly SVM has higher sensitivity and precision, which makes it a good option for tasks involving sentiment analysis. Even with TF-IDF, Naive Bayes delivers competitive performance despite being marginally less accurate.

These results are consistent with the goals of the study, which were to assess the efficacy of various machine learning algorithms and text vectorization strategies in sentiment analysis. The findings are consistent with the theory that TF-IDF provides better feature extraction capabilities since it improves accuracy and interpretability and works especially well when combined with SVM (Larasati et al., 2023).

In conclusion, this study's best method for sentiment analysis is the combination of TF-IDF with SVM, which yields a reliable and understandable model. The existing gaps in the literature suggest that future work may investigate the incorporation of hybrid models or more sophisticated preprocessing methods to further improve performance.

CHAPTER 6: CONCLUSION AND EVALUATION

With regard to sentiment analysis, this study sought to investigate the efficacy of various text vectorization methods, namely Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), in conjunction with machine learning algorithms, specifically Support Vector Machine (SVM) and Naive Bayes (NB). The main goal was to test these combinations on a labeled twitter dataset and determine how well they performed based on important metrics like F1-score, accuracy, precision, and recall.

The findings demonstrated that TF-IDF continuously outperformed BoW on all metrics, with the SVM classifier using TF-IDF obtaining the best accuracy of 87.3%. Additionally, it was shown via precision, recall, and F1-score that TF-IDF offers a more useful feature representation, especially for the SVM model. These results are in line with previous research that shows how reliable TF-IDF is for text categorization tasks (Semary et al., 2024; Lavanya et al., 2023). Though marginally less accurate than SVM, the Naive Bayes classifier nevertheless produced competitive outcomes, especially when paired with TF-IDF, which is consistent with another research (Chauhan et al., 2021).

The study's main goals were effectively accomplished, especially the demonstration of TF-IDF's superiority over BoW in text vectorization for sentiment analysis. This result is consistent with other research that highlights how crucial efficient feature extraction is to improving machine learning model performance (Abubakar and Umar, 2022). A thorough comparison of generative and discriminative classifiers was made possible by the decision to assess both SVM and Naive Bayes, offering insights into their respective advantages and disadvantages as recommended by Ng and Jordan (2002).

However, the study faced certain limitations. The dataset, while suitable for this research, was relatively small and domain-specific, which may limit the generalizability of the findings. Furthermore, the research did not explore more advanced or hybrid text vectorization techniques, such as word embeddings or transformer-based models, which could potentially yield even better results (Avinash and Sivasankar, 2018). The choice to focus solely on BoW and TF-IDF was a deliberate one, grounded in the need to evaluate fundamental approaches, but it also meant that more sophisticated methods were left unexplored.

Moreover, while the study effectively evaluated the classifiers on standard metrics, it did not delve deeply into the interpretability of the models, particularly in terms of understanding which features were most influential in driving predictions. This aspect, as highlighted by Witten et al. (2016), is crucial for deploying machine learning models in real-world applications where transparency is essential.

Research Questions:

1. Do text vectorization techniques such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) affect the accuracy of sentiment analysis?

The results of this investigation unequivocally show that text vectorization methods do, in fact, impact sentiment analysis accuracy. The findings demonstrated that, when paired with SVM, TF-IDF slightly outperformed BoW, obtaining an accuracy of 63.72% as opposed to 63.54% with BoW. This shows that the feature set produced by TF-IDF's capacity to rank terms according to their significance across the dataset is more detailed and insightful, which improves the model's performance in sentiment classification tasks. But the difference between BoW and TF-

IDF was smaller for Naive Bayes, suggesting that the effect of the vectorization method also depends on the particular algorithm applied.

2. Which machine learning algorithm, Naive Bayes or SVM, performs better in sentiment analysis when combined with different text vectorization techniques?

SVM routinely beat Naive Bayes in both text vectorization algorithms, according to the study. SVM is the better option for sentiment analysis in this situation because it produced higher accuracy, precision, recall, and F1-scores. This is especially clear when SVM is used in conjunction with TF-IDF, as this combination yielded the best overall accuracy. On the other hand, compared to SVM, Naive Bayes demonstrated shortcomings, especially in terms of reduced precision and recall for negative feelings, underscoring its less efficient processing of complicated text input.

When one considers the project's original goals, it is clear that the main goals were accomplished. A detailed investigation was conducted into the study topic of whether different text vectorization algorithms and classifiers offer differing levels of efficacy in sentiment analysis. The results of the study give a clear answer, showing that TF-IDF performs better for the given sentiment analysis task than SVM, especially when combined with it.

However, the research also revealed the complexities and challenges inherent in machine learning-based sentiment analysis. While the models achieved high accuracy, the investigation also highlighted areas for further improvement, particularly in terms of exploring advanced vectorization methods and larger, more diverse datasets. The study's limitations underscore the need for ongoing research to refine these approaches and address the gaps identified.

Based on the findings, several recommendations can be made. First, future research should consider integrating more advanced text vectorization techniques, such as word embeddings (e.g., Word2Vec, GloVe) or transformer-based models (e.g., BERT), which have shown promise in recent studies (Larasati et al., 2023). These methods could potentially enhance the model's ability to capture semantic nuances in text, leading to improved sentiment classification.

Additionally, expanding the dataset to include more diverse sources of text data, such as reviews, blogs, or news articles, would help generalize the findings and improve the robustness of the models. Furthermore, incorporating interpretability techniques, such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations), could provide deeper insights into the model's decision-making process, addressing a key limitation of the current study.

From a business standpoint, the results are useful for sectors like marketing, customer service, and finance that depend on sentiment analysis. Better decision-making and consumer insights may result from the accuracy and dependability of sentiment predictions increased by TF-IDF with SVM in sentiment analysis pipelines.

Project Management

The project was originally scheduled with enough time allotted for each stage, including data gathering, literature research, model creation, evaluation, and writing. Nevertheless, certain departures from the design were discovered during the project's actual execution. In particular, data preprocessing and feature extraction took longer than expected, which caused a little delay in the model evaluation stage. To guarantee the precision and caliber of the outcomes, these modifications were required.

The project also ran into issues with the availability of resources, specifically with access to big datasets and processing capacity for training models. By using cloud-based tools and concentrating on efficient code optimization, these were lessened. In retrospect, the procedure might have been even more streamlined with more time management and backup plans.

In conclusion, the project was a success in achieving its primary goals, contributing valuable insights into the effectiveness of different text vectorization techniques and classifiers in sentiment analysis. The reflections and recommendations provided here serve as a guide for future work, aiming to build on the foundation laid by this research.

Conclusion

In the context of sentiment analysis, this study aimed to assess the effectiveness of several text vectorization methods, particularly Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), in conjunction with Support Vector Machine (SVM) and Naive Bayes (NB) classifiers. The work has effectively shown the superiority of TF-IDF as a feature extraction method through rigorous testing and analysis, especially when combined with SVM, in reliably classifying sentiments within a dataset of tweets.

The study's conclusions have not only supported prior research highlighting the reliability of TF-IDF and SVM in text classification tasks, but they have also offered a transparent, data-driven comparison of these methods and algorithms in an actual application setting. Through this effort, a greater knowledge of how basic machine learning technologies may be used to improve sentiment analysis—a critical component in many domains, such as social media monitoring, customer service, and marketing—has been gained.

While the project achieved its core objectives, it also highlighted certain limitations, such as the relatively narrow scope of vectorization techniques and the constraints imposed by the size and nature of the dataset. These insights pave the way for future research, where more sophisticated methods, such as word embeddings and transformer models, could be explored to potentially achieve even greater accuracy and interpretability.

Reflecting on the overall process, this project has been a learning experience that underscored the importance of meticulous planning, iterative refinement, and critical evaluation. The project's outcomes not only answered the research questions posed at the outset but also provided a solid foundation for further exploration in the rapidly evolving field of sentiment analysis.

In conclusion, this project has effectively shown how sentiment analysis may be performed by fusing conventional machine learning approaches with efficient text vectorization techniques. The results provide a significant contribution to the subject and have applications for businesses whose decision-making processes depend on sentiment analysis. The knowledge acquired here will guide future research aimed at improving and enhancing the capabilities of sentiment analysis models in progressively more intricate and varied data contexts.

REFERENCES

1. A. Semary, N. *et al.* (2024) 'Enhancing machine learning-based sentiment analysis through feature extraction techniques', *PLOS ONE*, 19(2). doi:10.1371/journal.pone.0294968.

2. Abubakar, H.D. and Umar, M. (2022) 'Sentiment classification: Review of text vectorization methods: Bag of words, TF-IDF, word2vec and doc2vec', *SLU Journal of Science and Technology*, 4(1 & 2), pp. 27–33. doi:10.56471/slujst.v4i.266.

3. Avinash, M. and Sivasankar, E. (2018) 'A study of feature extraction techniques for sentiment analysis', *Advances in Intelligent Systems and Computing*, pp. 475–486. doi:10.1007/978-981-13-1501-5_41.

4. Chauhan, A., Agarwal, A. and Sulthana, R. (2021) 'Performance analysis of machine learning algorithms and feature extraction methods for sentiment analysis', *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* [Preprint]. doi:10.1109/icses52305.2021.9633882.

5. GeeksforGeeks (2024) *Naive Bayes vs. SVM for text classification*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/naive-bayes-vs-svm-for-text-classification/ (Accessed: 19 August 2024).

6. Larasati, A. *et al.* (2023) 'The effect of data splitting ratio and vectorizer method on the accuracy of the support vector machine and naïve Bayes model to perform sentiment analysis', *2023 8th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)* [Preprint]. doi:10.1109/iceeie59078.2023.10334755.

7. Lavanya, B.N., Shenoy, P.D. and Venugopal, K.R. (2023) 'Sentiment analysis of social media reviews using machine learning and word embedding techniques', *2023 IEEE 4th Annual Flagship India Council International Subsections Conference (INDISCON)* [Preprint]. doi:10.1109/indiscon58499.2023.10270129.

8. Liu, B. (2012) 'Sentiment analysis and opinion mining', *Synthesis Lectures on Human Language Technologies*, 5(1), pp. 1–167. doi:10.2200/s00416ed1v01y201204hlt016.

9. Medallia (2023) *Real time text analytics software*, *Medallia*. Available at: https://monkeylearn.com/sentiment-analysis/ (Accessed: 17 August 2024).

10. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

11. Ng, A. Y., & Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In: NIPS.

12. Pavanam, N. (2022) *Understanding naïve bayes and Support Vector Machine and their implementation in python*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2020/11/understanding-naive-bayes-svm-and-its-implementation-on-spam-sms/ (Accessed: 19 August 2024).

13. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning. Packt Publishing.

14. Rahman, Md.M. *et al.* (2020) 'A sentiment analysis-based approach for understanding the user satisfaction on Android application', *Advances in Intelligent Systems and Computing*, pp. 397–407. doi:10.1007/978-981-15-1097-7_33.

15. Rahman, S.S. *et al.* (2018) 'Supervised Ensemble Machine Learning Aided Performance Evaluation of sentiment classification', *Journal of Physics: Conference Series*, 1060, p. 012036. doi:10.1088/1742-6596/1060/1/012036.

16. Rana, S., & Singh, A. (2016). ‘Comparative analysis of sentiment orientation using SVM and Naive Bayes techniques ‘, *2nd International Conference on Next Generation Computing Technologies (NGCT)* (pp. 106-111). IEEE. doi: 10.1109/NGCT.2016.7877399.

17. Russell, S. and Norvig, P. (1995) ‘A modern, agent-oriented approach to introductory artificial intelligence’, *ACM SIGART Bulletin*, 6(2), pp. 24–26. doi:10.1145/201977.201989.

18. Sohan, M.F. *et al.* (2019) ‘NSTACKSENTI: Evaluation of a multi-level approach for detecting the sentiment of users’, *Communications in Computer and Information Science*, pp. 38–48. doi:10.1007/978-981-15-1718-1_4.

19. Wadhe, A.A. and Suratkar, S.S. (2020) ‘Tourist place reviews sentiment classification using Machine Learning Techniques’, *2020 International Conference on Industry 4.0 Technology (I4Tech)* [Preprint]. doi:10.1109/i4tech48345.2020.9102673.

20. *What is sentiment analysis?* (2023) *IBM*. Available at: https://www.ibm.com/topics/sentiment-analysis (Accessed: 17 August 2024).

21. Witten, I. H., Frank, E., & Hall, M. A. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

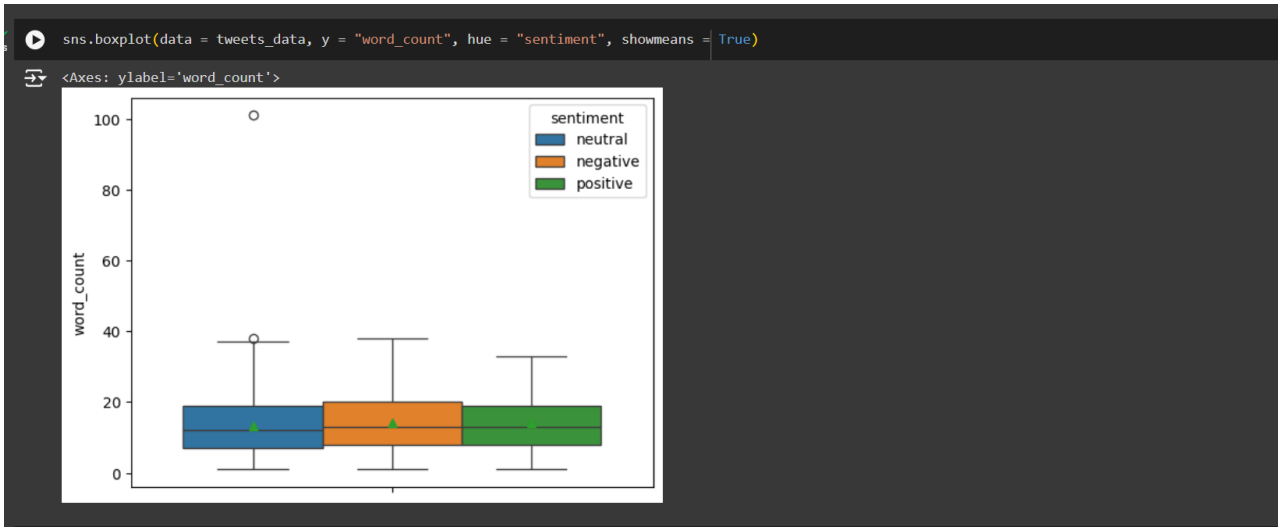
APPENDIX

```
import re
import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, roc_curve, auc
from wordcloud import WordCloud
from tensorflow.keras import layers, models, losses, Sequential,
optimizers, metrics
```

FIRST 10 ENTRIES OF THE DATASET

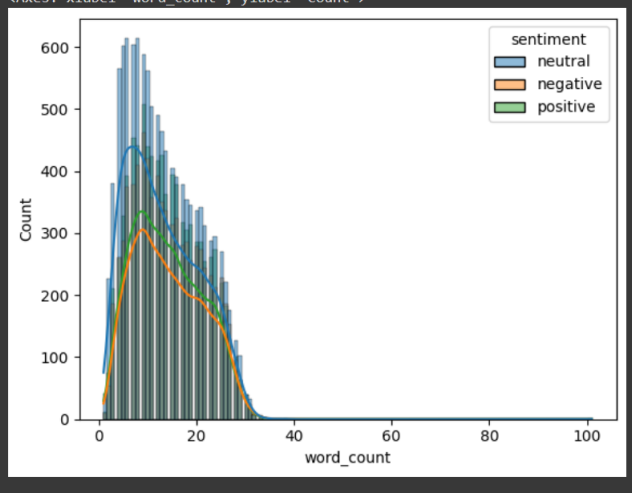
	textid	text	selected_text	sentiment
929	f8c1ff9b6e	You arent following me, so i cant send you a ...	You arent following me, so i cant send you a DM..	neutral
9388	e533dc15ce	Phew long day and i havent gotten to work yet	Phew long day	negative
17950	a019c63c8b	we had scones this afternoon!! they were grea...	we had scones this afternoon!! they were great...	neutral
15082	ad82469ae5	http://tinyurl.com/c33ffv thats what im makin...	yummy	positive
22427	605cb06fc8	Me just sign up Twitter...	Me just sign up Twitter...	neutral
15750	6d059e310b	Hitting the (fairly empty) shops in Orlando! O...	Hitting	negative
1781	eb19b96f32	Still bored but not long now till the next tou...	Still bored but not long now till the next tou...	neutral
27344	5605ea7b04	the japanese exchange student is the cutest th...	cutest	positive
7761	9adccf6bf9	Happy Mother`s Day to you	Happy	positive
12792	3991cca5f7	yeah I need a hug...cuz I am sick..	I am sick..	negative

	text	sentiment
10902	If you don`t want to come then don`t come. J...	negative
26664	I cant WAIT to be home and snuggle with my pup...	positive
499	Woke Up,I Wanna Stay In My bed	neutral
8333	don`t 4get Gina Thomas! (the things u do, ke...	neutral
10910	ohhh. hm. I don`t want to screw mine up	neutral
24510	GAWD!. If only that dream actually happened la...	neutral
21341	Thanks #sigjeans	positive
23984	Actually a great start to the day, hope it con...	positive
6119	Twitter you suck	negative
5854	this is really helpful, you swear a lot, i j...	positive



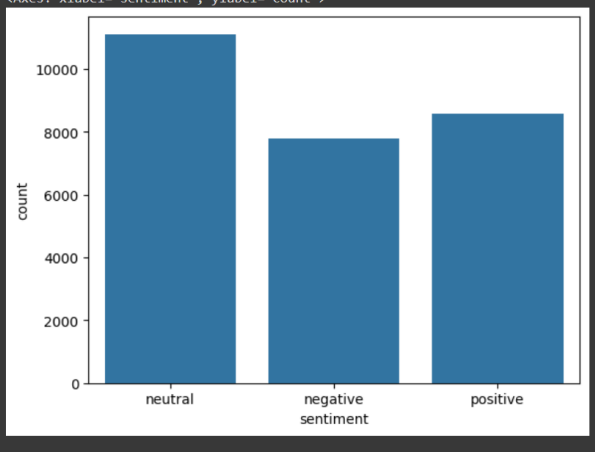
```
sns.histplot(data = tweets_data, x = "word_count", hue = "sentiment", kde = True)
```

```
<Axes: xlabel='word_count', ylabel='Count'>
```




```
sns.countplot(data = tweets_data, x = "sentiment")
```

```
<Axes: xlabel='sentiment', ylabel='count'>
```




```
from nltk.corpus import stopwords

nltk.download("stopwords")

stop_words = stopwords.words("english")

print(stop_words)
```

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",

'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

```
import re
import string

def custom_standardization(input_data):

  # Convert to lowercase
  lowercase = input_data.lower()

  # Remove URLs
  stripped_urls = re.sub(r"https?://\S+|www.\S+", "",lowercase)

  # Remove email addresses
  stripped_symbol = re.sub(r"\S*@\S*\s?", "",stripped_urls)

  # Remove text in angular brackets (usually HTML tags)
  stripped_brackets = re.sub(r"<.*?>+", "",stripped_symbol)

  # Remove any square brackets and leave the text within square brackets
  stripped_brackets = re.sub(r"\[|\]", "",stripped_brackets)
  # Matches alphanumeric characters with digits and remove those
  stripped_digits = re.sub(r"\w*\d\w*", "",stripped_brackets)
  # Remove stopwords
  stripped_stopwords = re.sub( r"\b(?:{})\b".format("|".join(stop_words)), "", stripped_digits) # Use re.sub() for substitution

  # Replace multiple whitespaces with a single whitespace
  stripped_whitespace_chars = re.sub(r"\s+", " ",stripped_stopwords)

  # Remove non-alphabet characters
  return re.sub(r"[^a-zA-Z\s]+" ,"", stripped_whitespace_chars)
```

	text	sentiment	word_count	cleaned_text
0	I`d have responded, if I were going	neutral	8	responded going
1	Sooo SAD I will miss you here in San Diego!!!	negative	11	sooo sad miss san diego
2	my boss is bullying me...	negative	5	boss bullying
3	what interview! leave me alone	negative	6	interview leave alone
4	Sons of ****, why couldn`t they put them on t...	negative	15	sons put releases already bought

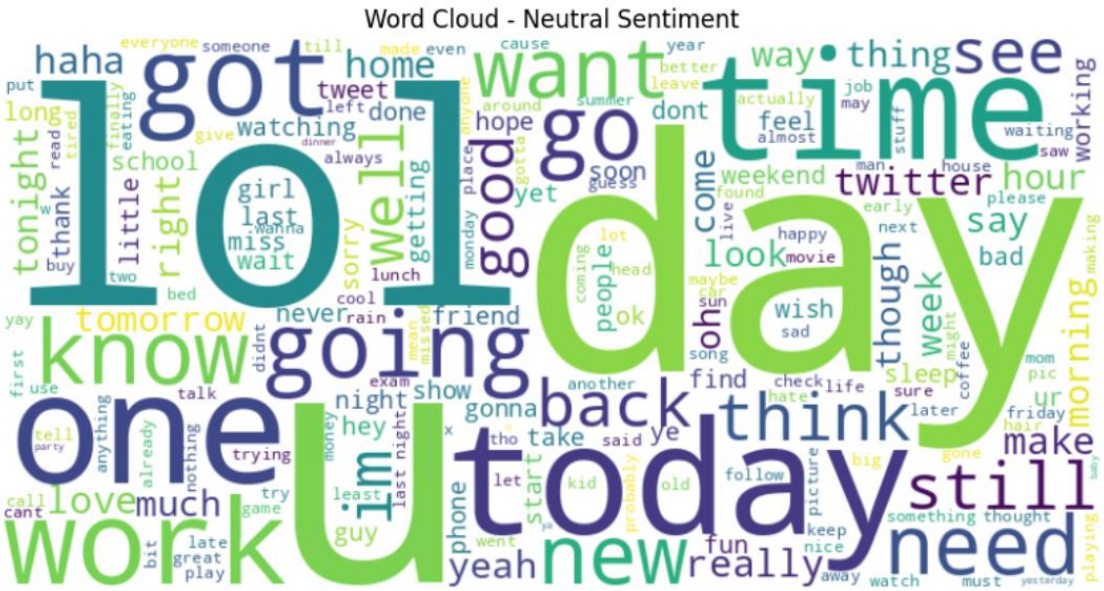
...	...	...	...	...
27476	wish we could come see u on Denver husband l...	negative	18	wish could come see u denver husband lost job...
27477	I`ve wondered about rake to. The client has ...	negative	26	wondered rake client made clear net force...
27478	Yay good for both of you. Enjoy the break - y...	positive	24	yay good enjoy break probably need hectic w...
27479	But it was worth it ****.	positive	8	worth
27480	All this flirting going on - The ATG smiles...	neutral	15	flirting going atg smiles yay hugs

```
# Extract the text from positive sentiment tweets
positive_tweets = tweets_data[tweets_data["sentiment"] == "positive"]["cleaned_text"]

# Concatenate all the positive sentiment tweets into a single string
positive_text = " ".join(positive_tweets)

# Create a WordCloud object
wordcloud = WordCloud(width = 800, height = 400, background_color =
"white").generate(positive_text)

# Plot the WordCloud
plt.figure(figsize = (10, 6))
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.title("Word Cloud - Positive Sentiment")
plt.show()
```


```
# Extract the text from negative sentiment tweets
negative_tweets = tweets_data[tweets_data["sentiment"] == "negative"]["cleaned_text"]

# Concatenate all the negative sentiment tweets into a single string
```

```
negative_text = " ".join(negative_tweets)

# Create a WordCloud object
wordcloud = WordCloud(width = 800, height = 400, background_color =
"white").generate(negative_text)

# Plot the WordCloud
plt.figure(figsize = (10, 6))
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.title("Word Cloud - Negative Sentiment")
plt.show()
```


```
# Extract the text from neutral sentiment tweets
neutral_tweets = tweets_data[tweets_data["sentiment"] == "neutral"]["cleaned_text"]

# Concatenate all the neutral sentiment tweets into a single string
neutral_tweets = " ".join(neutral_tweets)

# Create a WordCloud object
wordcloud = WordCloud(width = 800, height = 400, background_color =
"white").generate(neutral_tweets)

# Plot the WordCloud
plt.figure(figsize = (10, 6))
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.title("Word Cloud - Neutral Sentiment")
plt.show()
```



tweets_data["sentiment"] = tweets_data["sentiment"].replace({"negative": 0, "neutral": 1, "positive": 2})
tweets_data.sample(10)

text	sentiment	word_count	cleaned_text	
27014	I can`t buy, only rent!	1	6	buy rent
21215	I break some rules and don`t take tension for...	1	13	break rules take tension
9615	good job at the logies this year. Hell funny,...	2	22	good job logies year hell funny right baggin...
1268	@_Marguerite Yup, that`s Dianne and Claudia. ...	1	22	yup dianne claudia dianne new tweetville clau...
16237	Flew home from London to NI to catch up with ...	1	26	flew home london ni catch family caption show...
12958	http://twitpic.com/4jeij - New forest hoping t...	2	12	new forest hoping walk would lose pounds
6482	Ok getting the pink stripey one it`s the most ...	0	19	ok getting pink stripey one subtle bit bored...
6071	just downed 10 live abalone sashimi with 2 bot...	2	16	downed live abalone sashimi bottles sake fath...
13912	so jealous	0	3	jealous

22779	Hey, Wahts happening in #coffeclub..? I didnt...	1	16	hey wahts happening coffeclub didnt coffee tw...

```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(tweets_data, test_size = 0.2, stratify = tweets_data["sentiment"], random_state = 123)
X_train, X_val = train_test_split(X_train, test_size = 0.1, stratify = X_train["sentiment"], random_state = 123)
X_train.shape, X_val.shape, X_test.shape

# Bag of Words vectorization
bow_vectorizer = CountVectorizer()
X_bow = bow_vectorizer.fit_transform(texts)

# TF-IDF vectorization
tfidf_vectorizer = TfidfVectorizer()
X_tfidf = tfidf_vectorizer.fit_transform(texts)

# Analyze the most frequent words
def plot_top_words(vectorizer, X, title):
    sum_words = X.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vectorizer.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)[:20]
    words, freqs = zip(*words_freq)
    plt.figure(figsize=(10,8))
    plt.barh(words, freqs)
    plt.gca().invert_yaxis()
    plt.title(title)
    plt.show()

# Train-test split
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X_bow, labels, test_size=0.2, random_state=42)
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X_tfidf, labels, test_size=0.2, random_state=42)

# Sample a smaller portion of the training data (adjust the fraction as needed)
X_train_bow_small = X_train_bow[:int(0.2 * X_train_bow.shape[0])]
y_train_small = y_train[:int(0.2 * y_train.shape[0])]
from sklearn import svm
svm_bow = svm.SVC()
# Use the smaller dataset for GridSearchCV
X_train_tfidf_small = X_train_tfidf[:int(0.2 * X_train_tfidf.shape[0])]
y_train_small = y_train[:int(0.2 * y_train.shape[0])]

# Cross-validation and hyperparameter tuning for SVM
svm_params = {'C': [0.5, 1, 5, 10], 'kernel': ['linear', 'rbf']}
```

Best SVM params for BoW: {'C': 0.5, 'kernel': 'linear'}
Best SVM params for TF-IDF: {'C': 1, 'kernel': 'linear'}

 # Cross-validation and hyperparameter tuning for Naive Bayes

```
nb_params = {'alpha': [0.01, 0.1, 1, 10]}
nb_bow = GridSearchCV(MultinomialNB(), nb_params, cv=5)
nb_bow.fit(X_train_bow_small, y_train_small)
print("Best Naive Bayes params for BoW:", nb_bow.best_params_)
nb_tfidf = GridSearchCV(MultinomialNB(), nb_params, cv=5)
nb_tfidf.fit(X_train_tfidf_small, y_train_small)
print("Best Naive Bayes params for TF-IDF:", nb_tfidf.best_params_)
```

Best Naive Bayes params for BoW: {'alpha': 1}
Best Naive Bayes params for TF-IDF: {'alpha': 0.1}

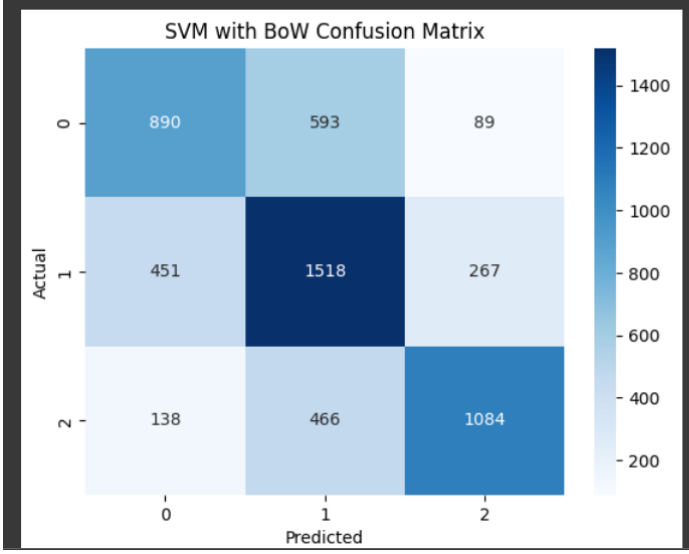
 # Evaluate models

```
def evaluate_model(model, X_test, y_test, title):
    y_pred = model.predict(X_test)
    print(f"{title} Accuracy:", accuracy_score(y_test, y_pred))
    print(f"{title} Classification Report:\n", classification_report(y_test, y_pred))
    conf_matrix = confusion_matrix(y_test, y_pred)
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
    plt.title(f"{title} Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

 # SVM with BoW

```
evaluate_model(svm_bow.best_estimator_, X_test_bow, y_test, "SVM with BoW")
```

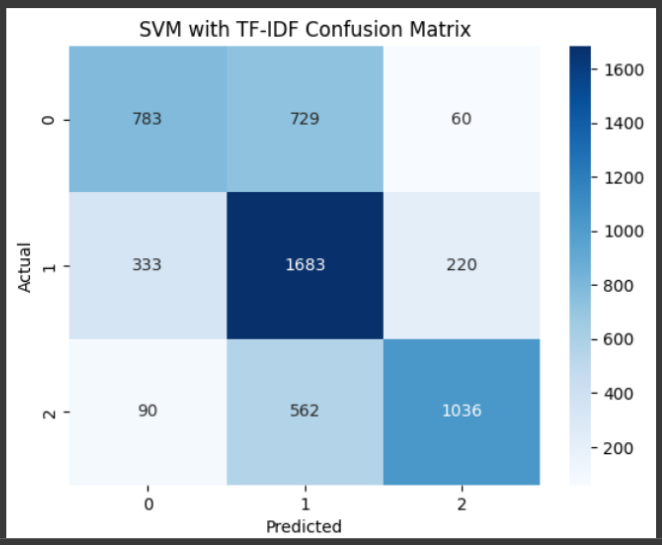
```
SVM with BoW Accuracy: 0.6353711790393013
SVM with BoW Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.57      0.58      1572
           1       0.59      0.68      0.63      2236
           2       0.75      0.64      0.69      1688

    accuracy                           0.64      5496
   macro avg       0.65      0.63      0.64      5496
weighted avg       0.64      0.64      0.64      5496
```



 # SVM with TF-IDF

evaluate_model(svm_tfidf.best_estimator_, X_test_tfidf, y_test, "SVM with TF-IDF")

```
SVM with TF-IDF Accuracy: 0.6371906841339156
SVM with TF-IDF Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.50      0.56      1572
           1       0.57      0.75      0.65      2236
           2       0.79      0.61      0.69      1688

    accuracy                           0.64      5496
   macro avg       0.67      0.62      0.63      5496
weighted avg       0.66      0.64      0.64      5496
```



 # Naive Bayes with TF-IDF

evaluate_model(nb_tfidf.best_estimator_, X_test_tfidf, y_test, "Naive Bayes with TF-IDF")

```
Naive Bayes with TF-IDF Accuracy: 0.5715065502183406
Naive Bayes with TF-IDF Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.43      0.50      1572
           1       0.52      0.68      0.59      2236
           2       0.65      0.56      0.60      1688

    accuracy                           0.57      5496
   macro avg       0.59      0.56      0.56      5496
weighted avg       0.58      0.57      0.57      5496
```