# Sentiment Analysis Report

## 1.0 Introduction

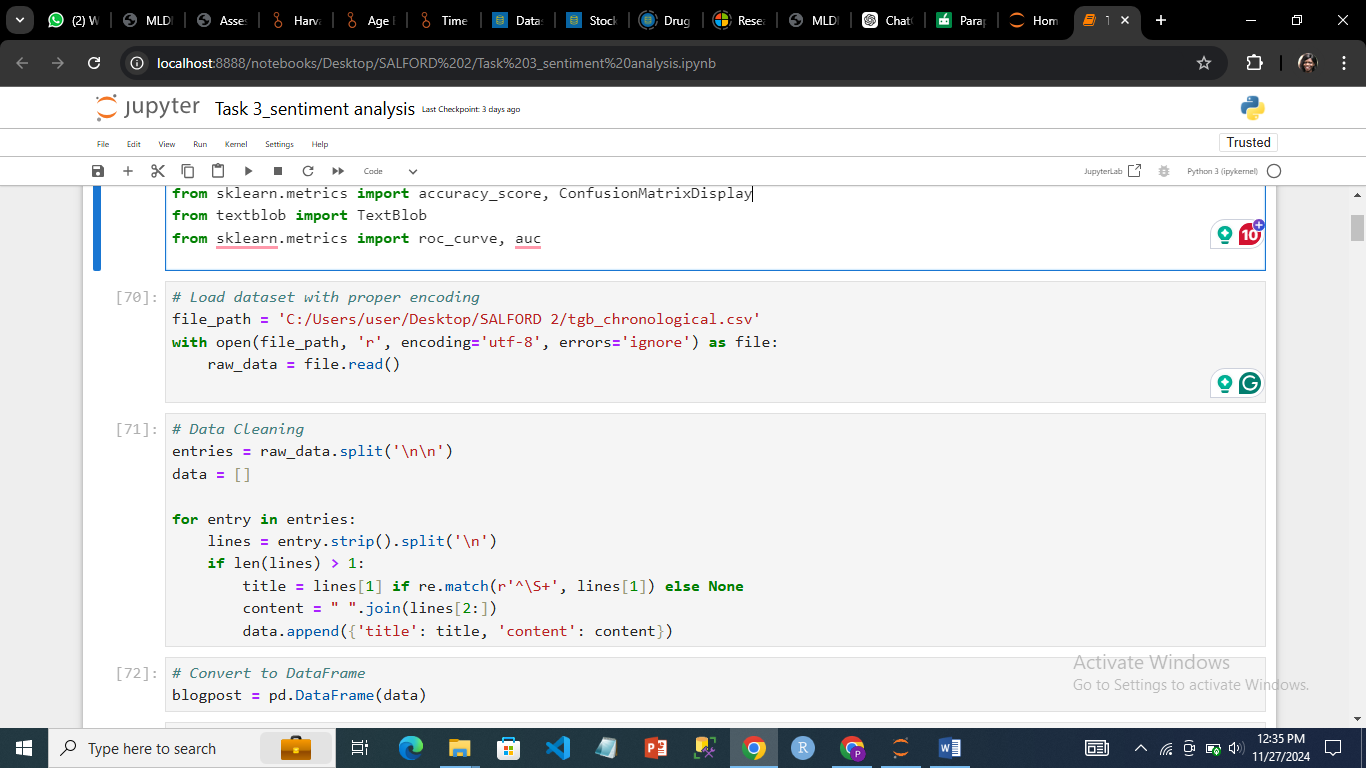
Sentiment analysis, often known as opinion mining, is the process of categorizing textual data as positive, negative, or neutral (Liu, 2012) This approach is commonly utilized in areas such as market research, consumer feedback, and social media monitoring (Pang, 2008). The purpose of this research was to categorize textual data into predetermined sentiment categories such as positive and negative, allowing for a better comprehension of the data's emotional associations.

## 2.0 Dataset Description and Exploratory Data Analysis

The dataset 'Time Goes By blog posts and comments (up to 08/2016)' was obtained from the Harvard Dataverse website. It included 4,151 blog articles from an older adult activist's https://www.timegoesby.net/ blog, including the post date, title, URL, commentators, and comments up to August 2016.

## 2.1 Exploratory Data Analysis

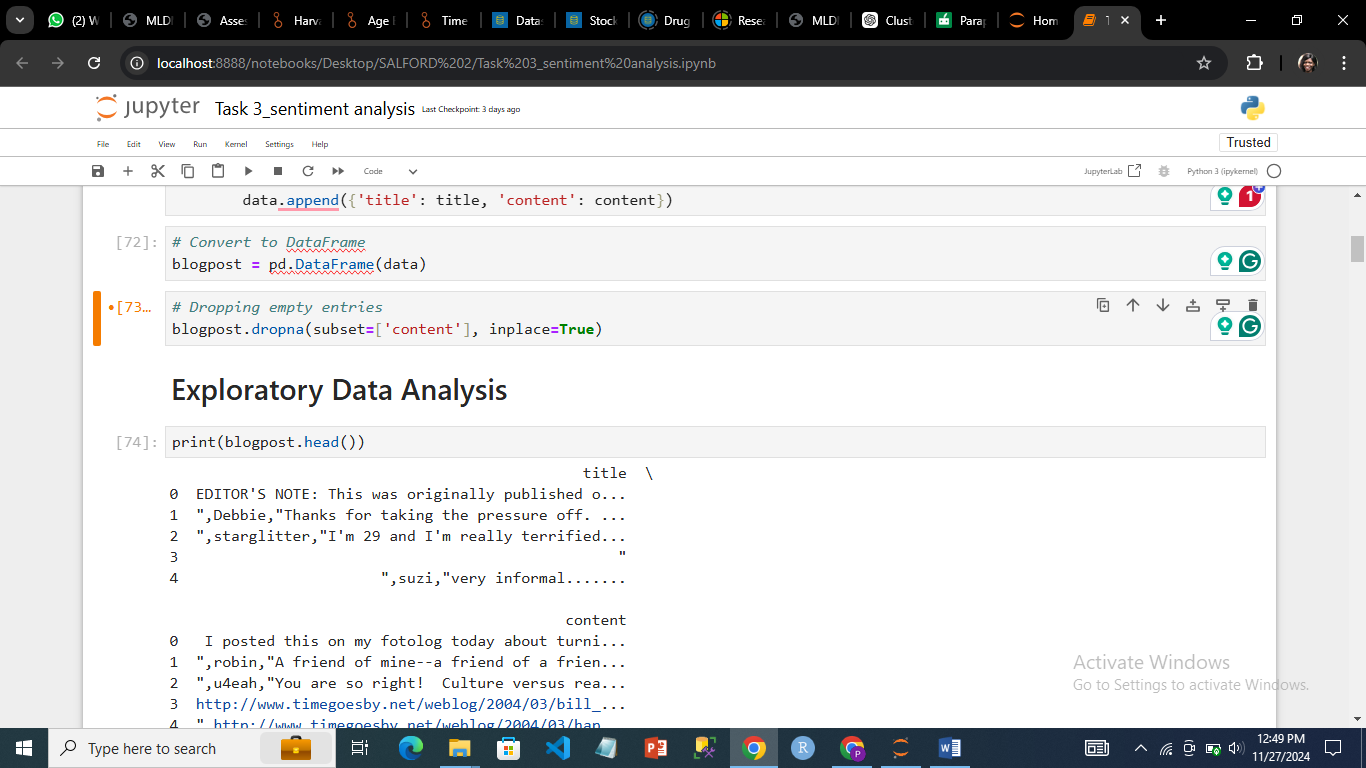
The first step was to import the essential libraries. The dataset was loaded from a CSV file using the open function in read-only mode (r), utf-8 encoding, and errors='ignore' to manage encoding difficulties. The read method saved the file's content as a single string to raw\_data for further processing.



***Importing the dataset***

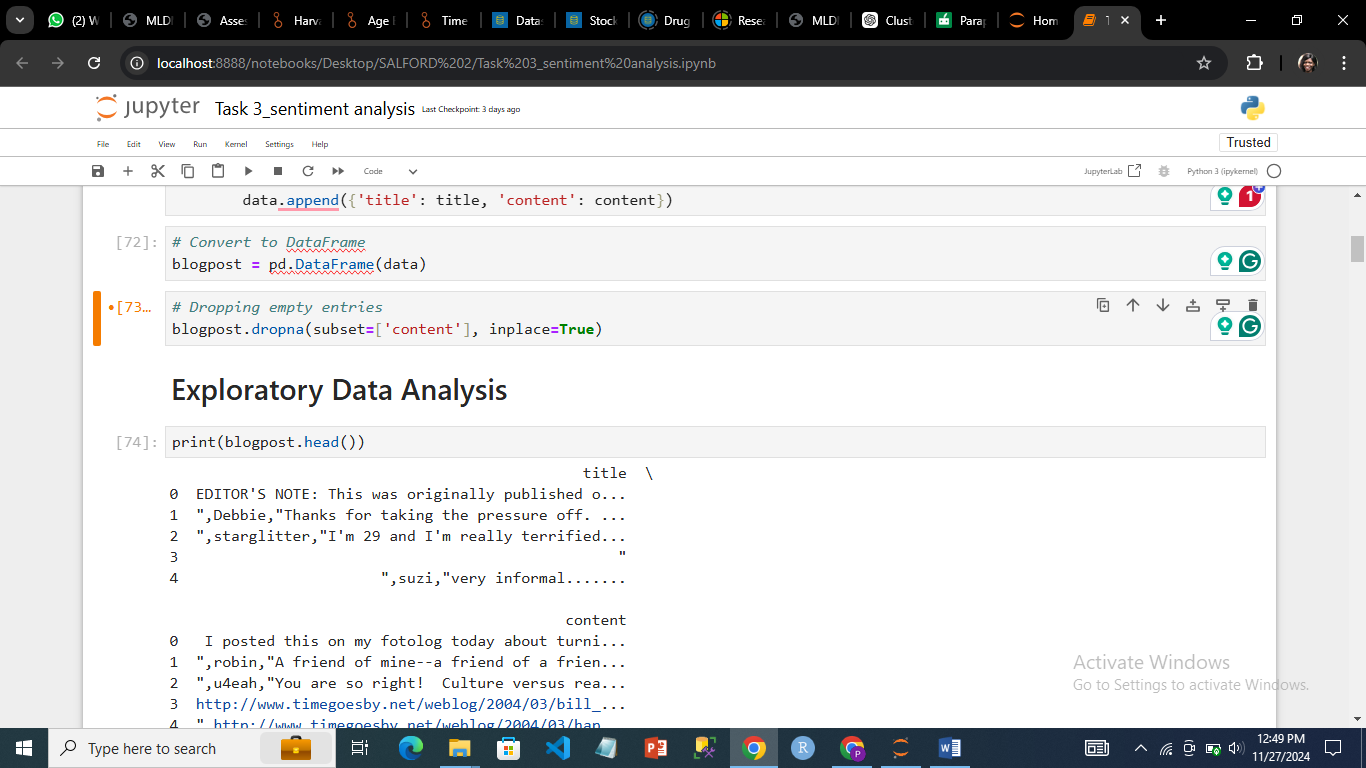
The raw data was cleaned and organised into separate entries using double newline characters (\n\n) as separators. Each submission was further divided into lines, and if it had more than one, the second line was chosen as the title (if it followed a specified pattern). The remaining lines were combined to create the content. The cleaned data was organised as dictionaries in a list, each with a title and accompanying content. This method converted the raw text into a structured format for further analysis.

The structured data was transformed into a single pandas DataFrame for simpler manipulation and analysis. 'title' and 'content' were the two columns of the DataFrame that included the title and content of each entry, respectively. The code eliminated any rows where the 'content' column was null or empty in order to guarantee that only genuine data was kept. By removing extraneous or incomplete rows, this stage improved the quality and accuracy of the analysis that followed and made sure that the focus was on entries with meaningful material.



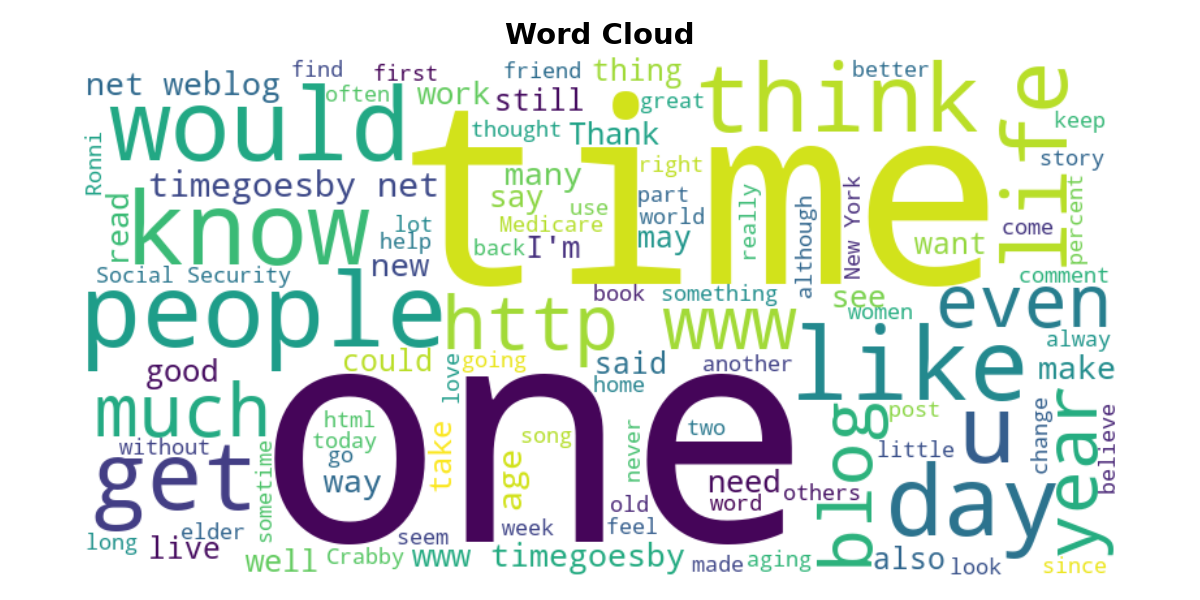
***Dropping empty entries in the dataset***

The initial five rows were shown. As a result, the cleaned data's structure was confirmed, the 'title' and 'content' columns were accurately filled in, and the dataset was prepared for additional examination.



***A snippet of the dataset***

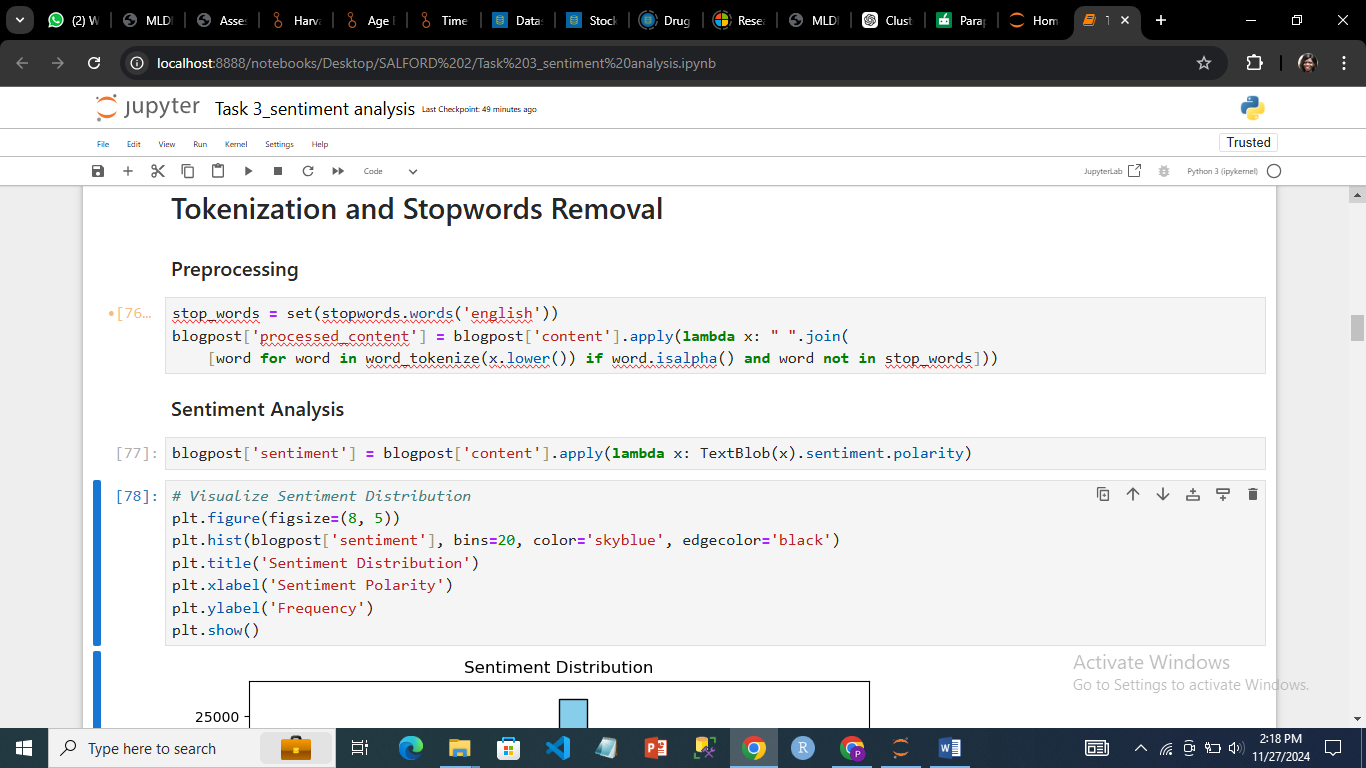
The visualization of a word cloud provided a sense of the dataset.



***A wordcloud snippet of the dataset***

## 3.0 Data Preprocessing

A predetermined list of stopwords—common words like "the," "is," and "in"—that don't significantly add meaning to the analysis were first imported from the nltk package. After that, the text in each entry in the 'content' column was tokenised into individual words and converted to lowercase. To highlight the text's important words, non-alphabetic tokens and stopwords were eliminated. A new column named processed\_content was created to hold the cleansed content. This phase was required to cut down on noise and get the text data ready for more efficient analysis, including topic modelling or sentiment analysis (Turney,2008).

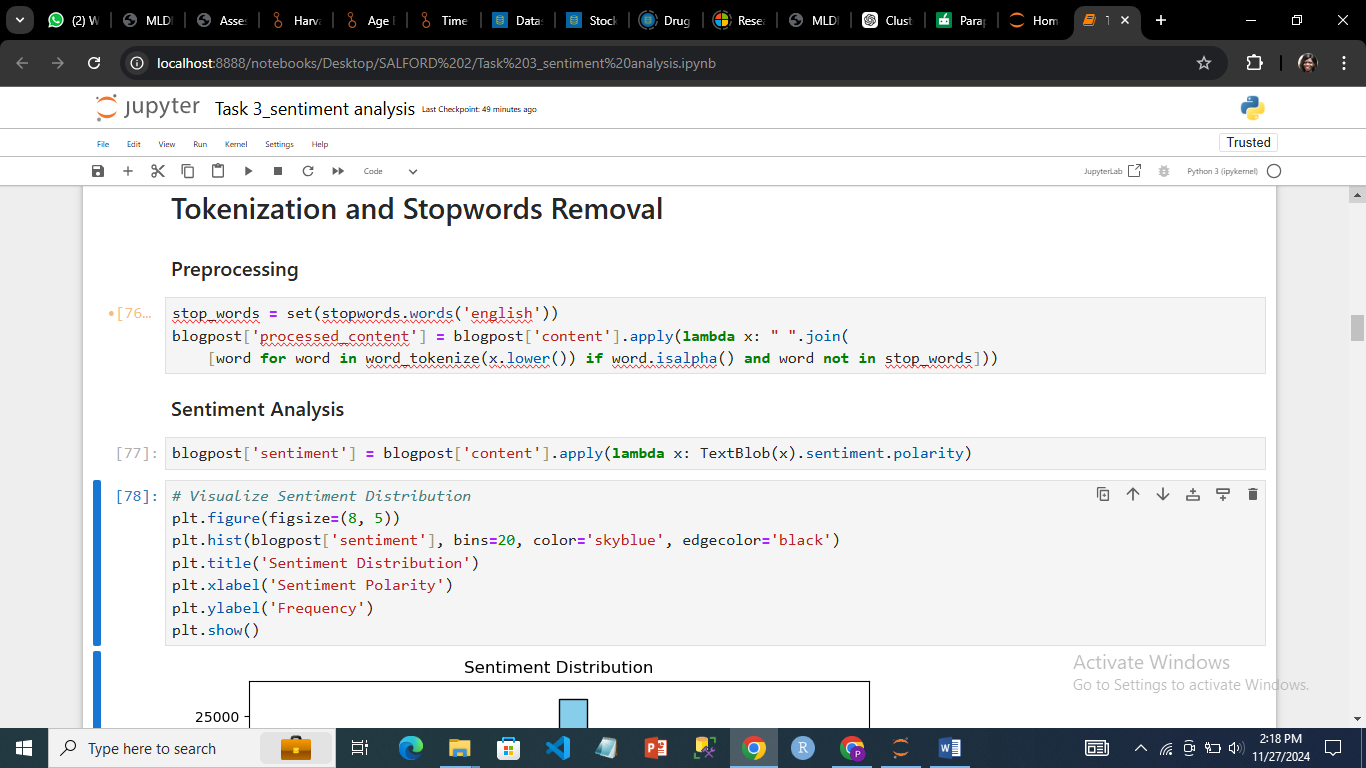


***Data preprocessing***

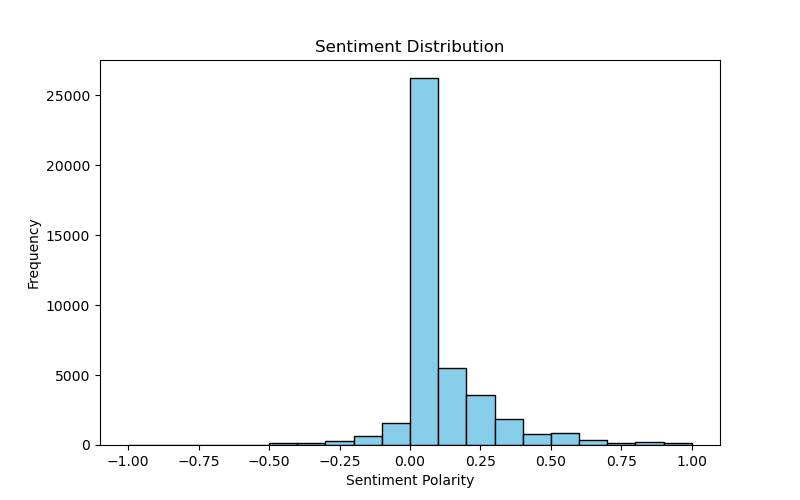
## 4.0 Sentiment Classification

In order to further analyse how sentiment varied throughout the dataset, it was necessary to measure the text's emotional tone. Using the TextBlob library, the code calculated the sentiment polarity of each blog post's content and saved the results in a new column named sentiment. With 0 denoting neutrality, the sentiment polarity scaled from -1 (negative sentiment) to 1 (positive sentiment).

The sentiment polarity score distribution throughout the dataset was displayed by the code. An x-axis representing sentiment polarity (from -1 to 1) and a y-axis representing frequency were used to create a histogram that showed the frequency of various sentiment polarity values. Understanding the overall sentiment trends—such as whether the dataset was primarily composed of neutral, negative, or positive sentiments—was made possible by the visualisation. In order to quickly grasp the dataset's emotional tone and spot any biases or imbalances in sentiment, this phase was crucial.



***Sentiment Analysis and visualization code***

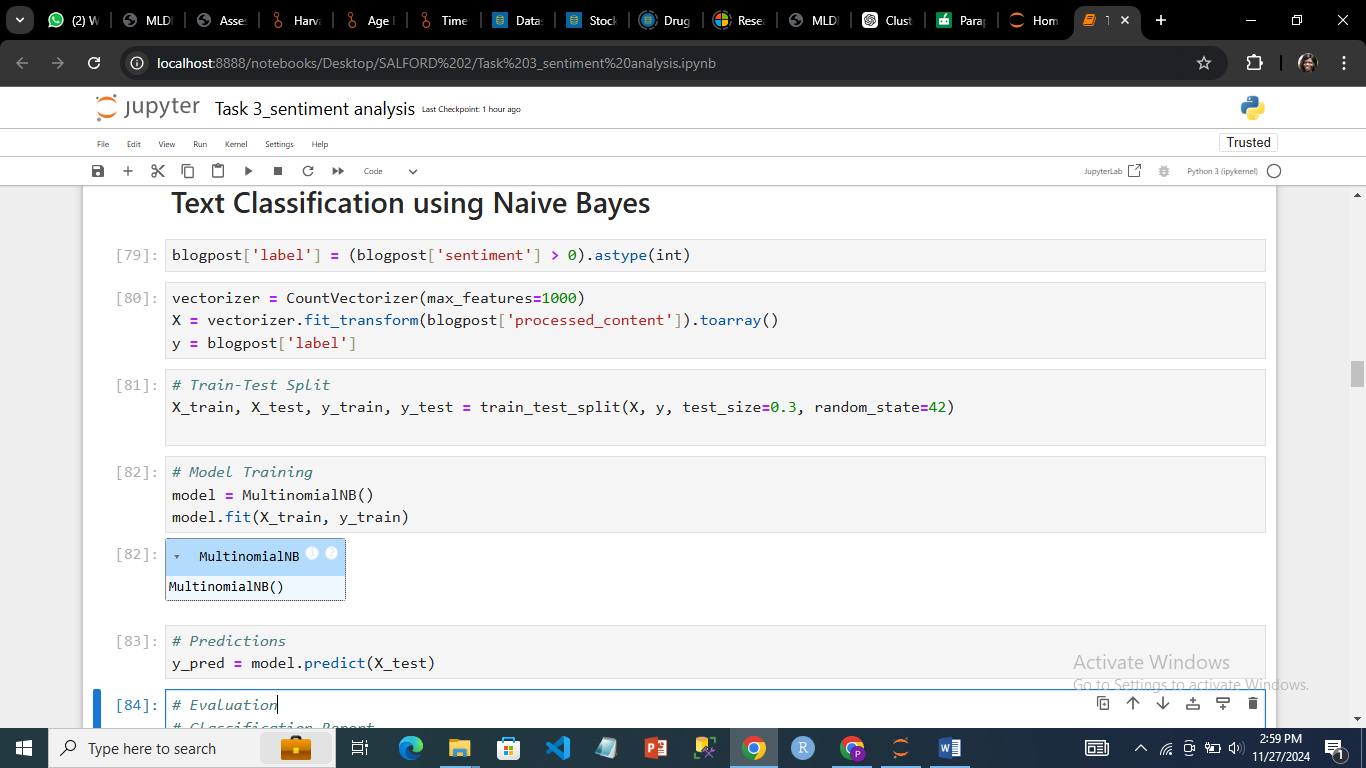


***Sentiment distribution***

## **4.1 Model Training**

A CountVectorizer was used to turn the preprocessed text data into numerical features. This resulted in a matrix of token counts, restricted to the top 1,000 features. Each entry was given a binary label according to the polarity of its attitude, with positive sentiment being denoted by 1 and non-positive sentiment by 0. To ensure a trustworthy assessment of the model's performance, the dataset was then divided into training and test sets, with 70% of the data being used for training and 30% left aside for testing.

A Multinomial Naive Bayes classifier was used due to its effectiveness and adaptability for text classification problems including word frequencies. The model was trained using the training set to determine the correlations between the features and binary sentiment labels. Predictions were produced on the test set, and the model's performance was assessed with a classification report. This report included critical measures such as precision, recall, F1-score, and accuracy to help understand the classifier's efficacy in predicting sentiment labels.

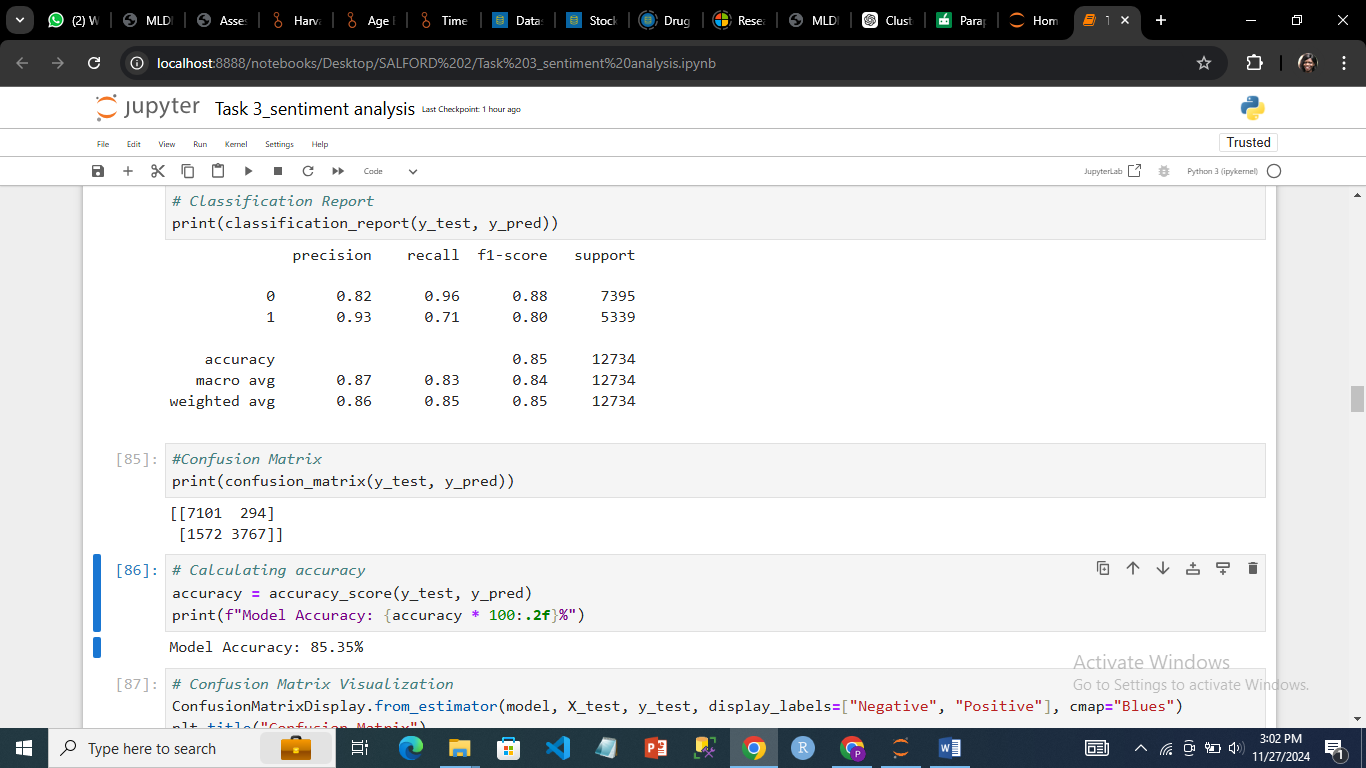


***Text Classification***

## **4.2 Results**

The model's performance was further evaluated using a confusion matrix and an accuracy score. The confusion matrix displayed a split of the model's predictions in comparison to the actual labels, as well as the counts of true positives, false positives, and false negatives. This comprehensive view showed where the model did well and where it misclassified.

The Multinomial Naive Bayes model had an overall accuracy of 85.35%, which meant it accurately predicted sentiment labels for 85.35% of the test data. This score indicated the model's ability to categorize sentiments and offered a clear assessment of its overall performance.

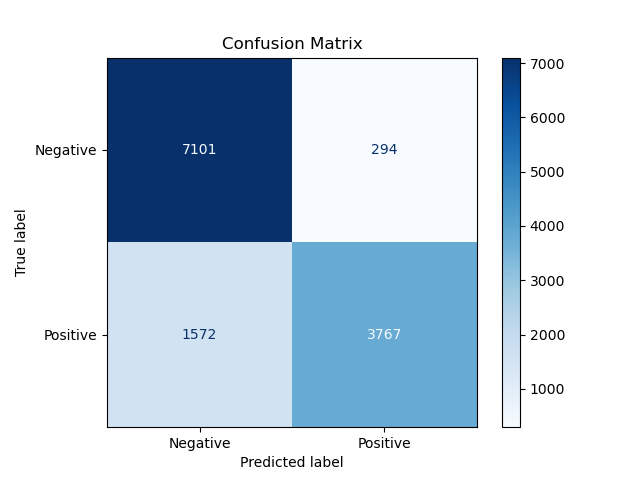


***Code on classification report, confusion matrix and accuracy***

## 5.0 Visualization

The performance of the Multinomial Naive Bayes model in sentiment label classification was shown in the confusion matrix. It was discovered that the model accurately identified 3,767 positive samples (true positives) and 7,101 negative samples (true negatives) out of all estimates. 1,572 positive samples were incorrectly categorized as negative (false negatives), while 294 negative samples were incorrectly classified as positive (false positives).

A heatmap was created to help visualize the confusion matrix and aid in the interpretation of the data. A visual depiction of the matrix's counts was given by the confusion matrix display, which also clearly labelled the positive and negative classes. The model's sentiment categorization strengths and shortcomings could be intuitively understood due to color intensity, which highlighted the distribution of accurate and inaccurate predictions.



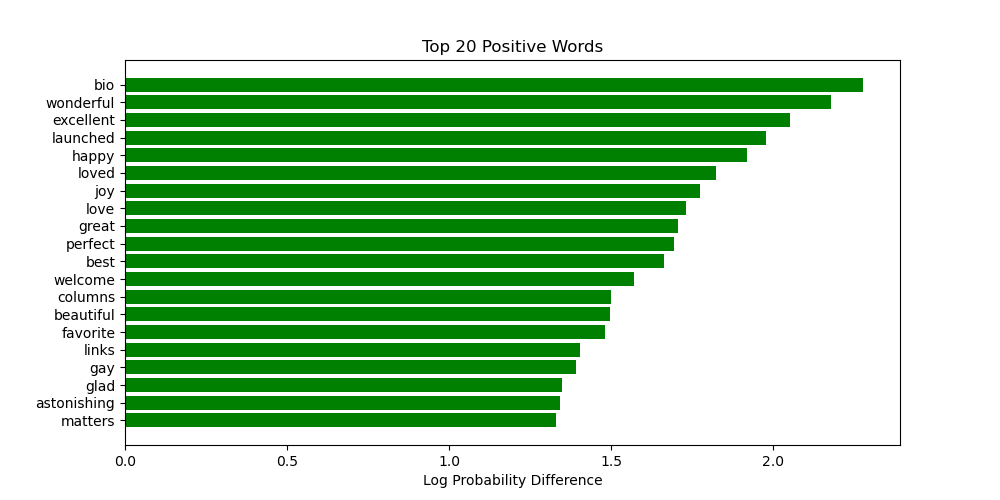
***confusion matrix visualization***

## 5.1 Sentiments

The feature importance of the Multinomial Naive Bayes model was analyzed to identify the words that had the most influence on predicting positive and negative sentiment. This was done by calculating the log probability difference between positive and negative classes for each word in the vocabulary. Words with higher log probability differences were considered more influential for the respective sentiment class.

## 5.1.1 Positive Sentiments.

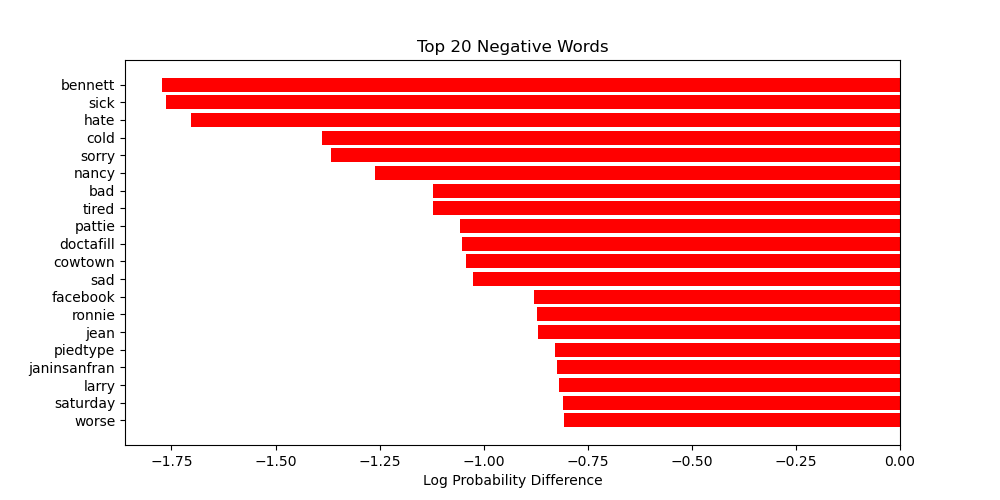
The first plot highlighted the top 20 positive words, showing their log probability differences and emphasizing their strong association with positive sentiment. The second plot displayed the top 20 negative words, with their log probability differences illustrating their strong influence on negative sentiment classification. These visualizations provided intuitive insights into how the model made its predictions and which words contributed most to distinguishing between positive and negative sentiments.



***Positive Sentiments***

## 5.1.2 Negative Sentiments.

The top 20 positive and negative words were extracted based on their log probability differences. Positive words had the largest positive differences, while negative words had the largest negative differences. These words were visualized using horizontal bar plots.

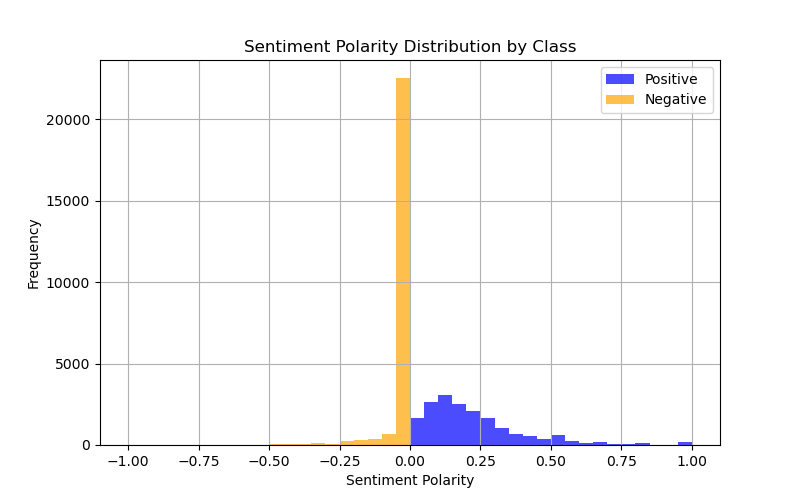


***Negative Sentiments***

## 5.1.3 Sentiments Polarity Distribution

The sentiment distribution by class was visualized to better understand the polarity distribution of positive and negative sentiments. Two histograms were created, one for positive sentiment (label == 1) and one for negative sentiment (label == 0). The histograms displayed the frequency of sentiment polarity values, with positive sentiments shown in blue and negative sentiments in orange. The sentiment polarity was plotted on the x-axis, ranging from negative to positive values, while the y-axis represented the frequency of each polarity.

This visualization provided a clear view of how sentiment polarity was distributed across the two classes, highlighting whether positive or negative sentiments were more extreme or concentrated in certain polarity ranges. The histogram allowed for an intuitive understanding of the model's classification performance, with the distribution of sentiment polarity for each class being easily comparable.



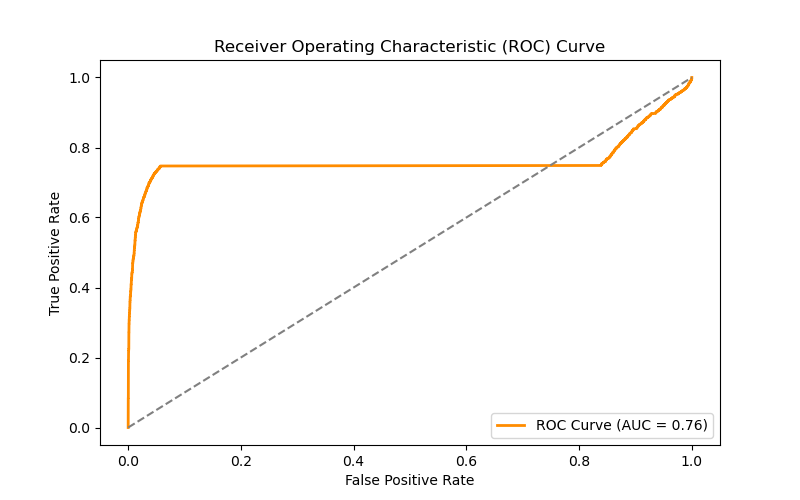
***Sentiment Polarity Distribution***

## 5.2 Receiver Operating Curve (ROC)

The ROC (Receiver Operating Characteristic) curve was computed to evaluate the performance of the Multinomial Naive Bayes model in distinguishing between positive and negative sentiments. The model's predicted probabilities for the positive class were used to compute the false positive rate (FPR) and true positive rate (TPR) at various threshold values. These values were then plotted to generate the ROC curve, which visually represents the trade-off between the true positive rate and false positive rate at different decision thresholds.

The area under the curve (AUC) was calculated to quantify the overall performance of the model. In this case, the AUC was 0.76, indicating that the model had a reasonably good ability to discriminate between positive and negative sentiments. AUC values closer to 1 indicate better performance, with values around 0.5 suggesting no discrimination (similar to random guessing).

The ROC curve was visualized with the FPR on the x-axis and the TPR on the y-axis, and a diagonal reference line was included to represent random guessing. This visualization helped assess the model's classification power and its ability to correctly identify positive and negative sentiment while minimizing false positives.

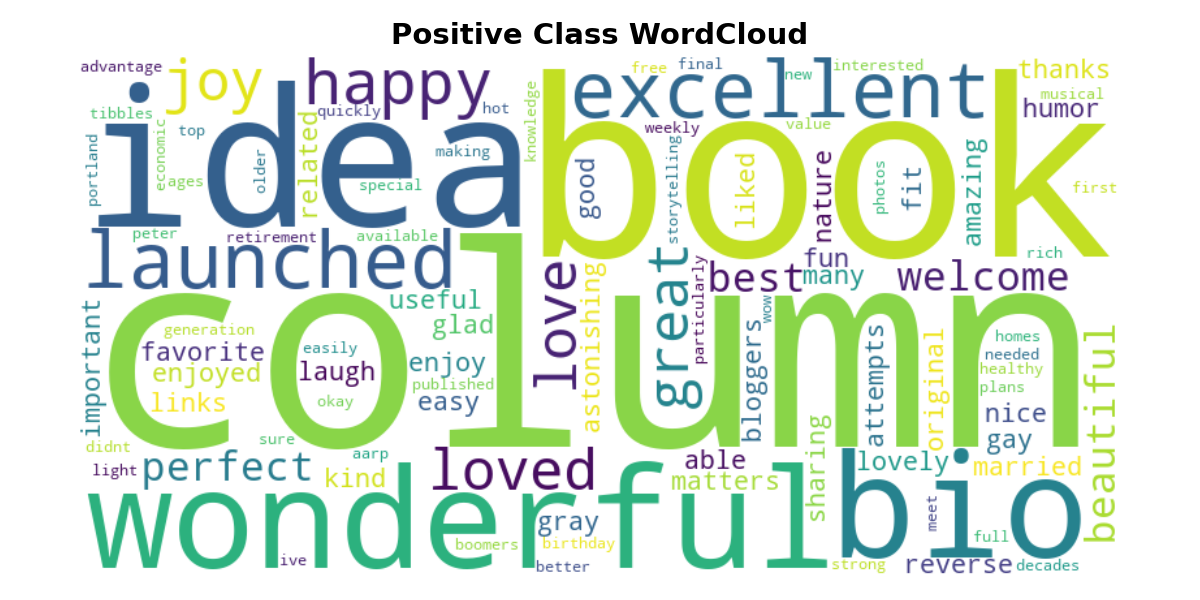


***Receiver Operating Curve***

## 5.3 Top 100 Positive Sentiments

For sentiments, it was generated and visualized, a Word Cloud for the top 100 words associated with the positive sentiment class, as identified by the Multinomial Naive Bayes model. First, the feature probabilities for each word were calculated, with the most influential words for the positive class being identified based on their log probability difference. The top 100 words were selected by sorting the probabilities, and these words were then compiled into a text string suitable for generating a Word Cloud.

A WordCloud was created using the WordCloud library, where the top 100 positive words were visualized with the most frequent words appearing larger. Stopwords, such as common words with little meaning in the context, were excluded to improve the clarity of the visualization. The visualization provided an intuitive way to explore the most significant words contributing to the classification of positive sentiment, making it easier to understand the model's decision-making process.

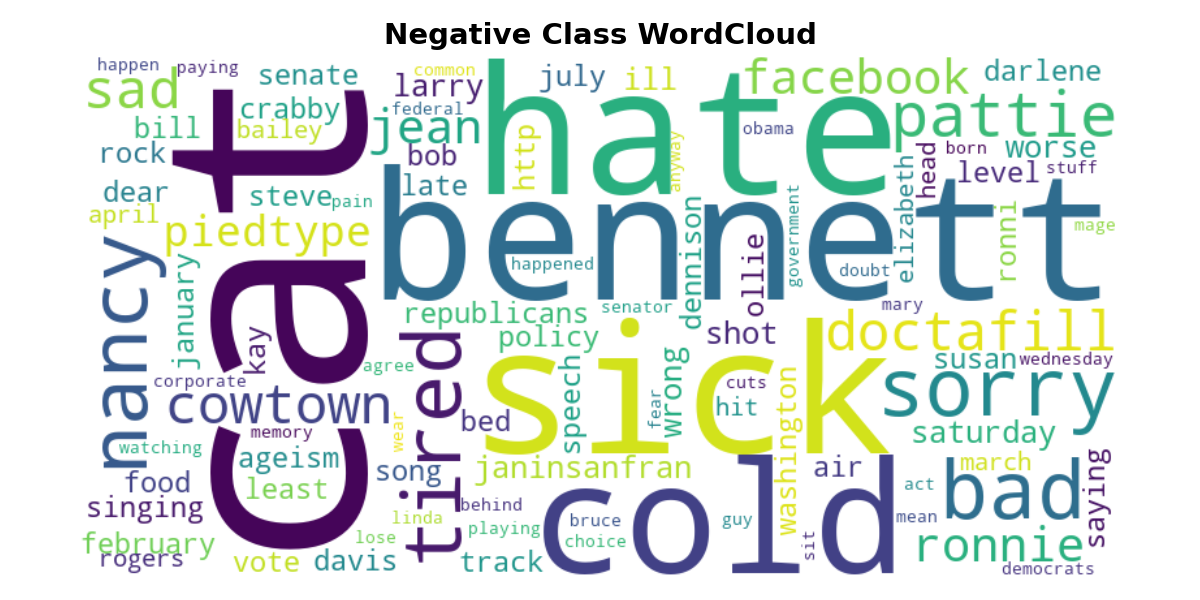


***Top 100 Positive Sentiments***

## 5.4 Top 100 Negative Sentiments

The code was used to generate and visualize a WordCloud for the top 100 words associated with the negative sentiment class, similarly to how the positive class WordCloud was created. The log probability differences for each word were calculated to identify the most influential words for the negative sentiment class. The top 100 negative words were selected by sorting these probabilities, and they were compiled into a text string.

A WordCloud was generated using the WordCloud library, which visualized the most frequent negative words. Words that appeared more frequently were displayed larger, providing a clear view of the vocabulary most associated with negative sentiment. Common stopwords were excluded to focus on the most meaningful words. This visualization helped to highlight key terms that the model used to classify negative sentiments, enhancing the interpretability of the model's predictions.



***Top 100 Negative Sentiments***

**7.0 Conclusion**

In conclusion, sentiment analysis has proven to be a valuable tool for extracting and interpreting emotional tones from textual data. By applying machine learning techniques, such as Multinomial Naive Bayes, we were able to successfully classify sentiment into positive and negative categories based on the content of the text (Pedregosa, 2011). The process involved careful data preprocessing, including tokenization, stopword removal, and vectorization, to ensure that the model could effectively learn patterns from the text.

Through various evaluation metrics, including the ROC curve, AUC score, and confusion matrix, we assessed the model's performance, which demonstrated a strong ability to distinguish between positive and negative sentiments. Visualizations, such as the WordClouds, provided valuable insights into the most influential words driving sentiment classification, offering a more intuitive understanding of the model's decision-making process (Feldman, 2013).

Overall, sentiment analysis not only showcases the power of natural language processing but also offers practical applications across industries, such as customer feedback analysis, social media monitoring, and market research. While the model showed solid accuracy, future improvements, such as fine-tuning the model or exploring different algorithms, could further enhance its performance and robustness.

## References

1. **Liu, B. (2012)**. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.  
   This book provides a comprehensive overview of sentiment analysis, including methods, techniques, and applications in various fields.
2. **Pang, B., & Lee, L. (2008)**. *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.  
   This paper presents a survey of sentiment analysis techniques and methods for opinion mining, along with the challenges in the field.
3. **Turney, P. D., & Littman, M. L. (2003)**. *Measuring Praise and Criticism: Inference of Semantic Orientation from Association*. ACM Transactions on Information Systems, 21(4), 315-346.  
   This paper discusses the early development of sentiment analysis and methods for determining the semantic orientation of words and phrases.
4. **Pedregosa, F., et al. (2011)**. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.  
   The Scikit-learn library is widely used in machine learning, including sentiment analysis, and this paper describes its implementation and functionality.
5. **Feldman, R. (2013)**. *Techniques and Applications for Sentiment Analysis*. Communications of the ACM, 56(4), 82-89.  
   This article explores various techniques and real-world applications of sentiment analysis, providing valuable insights into the methodology.
6. **Sklearn Documentation: Text Classification and Sentiment Analysis**. Retrieved from <https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction>