Capstone Project

Amazon Reviews Sentiment Analysis

Joanne ZY Liaw
Cohort C5 Nov
Affiliation (Hyperion Dev)

Abstract

The Capstone Project explored sentiment analysis on Amazon product reviews using natural language processing (NLP) tools and visualizations to understand consumer opinions—the methodology involved preprocessing text data, conducting sentiment analysis to classify reviews and analysing sentiment scores. Critical visualizations included pie charts for sentiment distribution, word clouds for positive and negative sentiments, scatter plots for correlations, and time series analysis with trend lines to observe sentiment over time. The results revealed predominant positive sentiment, variations across categories, and insights into review length correlation with sentiment. Conclusions highlighted the effectiveness of NLP in extracting valuable insights from consumer feedback, aiding in understanding customer satisfaction and product perception.

Table of Contents

List of Figures

Figures:

Chapter 1: Introduction

1.1 Overview

In the digital age, the volume of textual data generated by online platforms is monumental, presenting both a challenge and an opportunity. Sentiment analysis, a cornerstone of Natural Language Processing (NLP), harnesses the vast expanse of textual data to discern the emotional undertones that drive human communication.
Sentiment Analysis represents a convergence of data analytics and linguistics, where the subjective qualities of language are quantified into actionable data. Natural Language Processing (NLP) is a crucial tool for interpreting the vast swaths of unstructured text data generated by online interactions. Its significance is pronounced in today's data-driven decision-making processes, where understanding consumer sentiment is vital for product development, brand management, and customer service.

1.2 Objectives

This capstone project aims to understand sentiment analysis as a computational task and harness its power to navigate the vast landscape of consumer feedback on Amazon products.
The specific aims are:

- To develop a robust Python-based sentiment analysis model capable of discerning nuanced emotional tones from product reviews.
- To evaluate the effectiveness of this model in identifying and categorizing sentiments as positive, negative, or neutral.
- To derive actionable insights from the analysis to inform product improvements and tailor marketing strategies.
- The scope is specifically on consumer electronics, leveraging sentiment analysis to gain insights into customer perceptions and market trends.

1.3 Scope

This project narrows its investigation to consumer electronics, focusing on the "Consumer Reviews of Amazon Products" dataset. Its scope includes preprocessing review text, sentiment classification, and trend analysis, excluding other potential NLP tasks such as entity recognition or syntactic parsing. Products analysed are limited to those present within the dataset.

Chapter 2: Literature Review

2.1 Background

The background offers a concise overview of the evolution of sentiment analysis, tracing its roots from simple lexicon-based approaches to complex models powered by machine learning and deep learning. This progression highlights the field's growth from relying on static word lists to adopting dynamic, context-aware methods that capture the intricacies of human language, including sarcasm and subtlety. The statement encapsulates the interdisciplinary nature of sentiment analysis, emphasizing its reliance on linguistics, psychology, and computer science insights to understand and interpret emotions expressed in text. Existing research in the field spans various domains, encompassing reviews, social media posts, and customer feedback.

2.2 Relevant Technologies

At the heart of this project's methodology lie two pivotal technologies:

- Python: A versatile programming language at the project's core for data manipulation, analysis, and visualization.
- Pandas: A Python library for efficient data structures and operations for manipulating numerical tables and time series.
- NumPy: Used for numerical computations and handling arrays.
- TextBlob is an extension to spaCy that allows easy text processing. It provides a simple API for exploring common natural language processing (NLP) tasks.
- Matplotlib and Seaborn are Python libraries for data visualization. They enable the creation of histograms, pie charts, and scatter plots to represent the findings of the sentiment analysis visually.
- Scipy: Utilized for statistical analysis and confidence intervals calculation

These technologies collectively formed the backbone of my sentiment analysis project, providing robust analytical capabilities to extract, process, and visualize sentiment data from Amazon product reviews.

2.3 Previous Findings

Research in sentiment analysis highlights the superiority of neural networks and transformer-based models like BERT for capturing complex linguistic features. Studies emphasize the significance of context-aware models and transfer learning for enhancing sentiment detection accuracy.
Additionally, integrating sentiment analysis with other NLP tasks offers comprehensive insights, indicating a trend towards multifaceted text analysis approaches in understanding consumer sentiments and opinions more deeply. This project builds upon these insights, applying modern NLP techniques to analyze Amazon product reviews.

Research indicates that customer sentiment can be a leading indicator of purchase patterns and product success, reinforcing sentiment analysis's strategic value.

Chapter 3: Methodology

3.0 Introduction

This section introduces the comprehensive methodology adopted in this capstone project, highlighting the systematic approach from data collection to analysis. It sets the stage for the detailed description of methods used in gathering, preprocessing, analysing, and evaluating the sentiment of Amazon product reviews. The objective is to ensure reproducibility and transparency in the research process, enabling future studies to build upon this work.

3.1 Data Collection

3.1.1 Source Description:
The dataset used for this sentiment analysis comprised consumer reviews of Amazon products. It consisted of structured data with columns for review texts, dates, and ratings, among others, providing a rich source for textual analysis. The original dataset was a compendium of genuine customer feedback, suitable for assessing public opinion on a diverse range of products.
Reference: McAuley, J., & Leskovec, J. (2013). From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. WWW '13 Proceedings of the 22nd International Conference on the World Wide Web.

3.1.2 Data Selection Justification

The "Amazon_product_reviews.csv" dataset was chosen due to its coverage of consumer sentiment and the availability of textual data sufficient for NLP tasks. This dataset's rich diversity and substantial volume enable a deep dive into the intricacies of consumer behaviour and sentiment, providing a broad base from which to extract meaningful patterns and trends. The dataset's structure and metadata also allowed for a multifaceted analysis, including temporal trends and sentiment distribution.

3.2 Data Preprocessing

3.2.1 Cleaning Steps:

Initial cleaning involved deduplication to remove redundant entries (Code Block 5), ensuring data uniqueness. SpaCy's linguistic annotations streamlined the dataset for relevant content analysis by removing stopwords—common words contributing little to sentiment.

Duplicate Removal:
The dataset was first checked for duplicate entries using data.drop_duplicates(). Removing duplicates is crucial to prevent any bias that might occur from processing the same review multiple times and ensure that each piece of sentiment data is unique.

Handling Missing Values:
Missing values, especially in the reviews.text column, could skew the analysis if they are not addressed. By using reviews_data.dropna(), rows with missing review text were removed. This step ensures that the analysis is based on complete data, maintaining the integrity of the sentiment analysis.

Date Column Backup:
Given the importance of the reviews. For the date column for potential time-series analysis or tracking sentiment over time, a backup of this column was created with data['reviews_date_backup'] = data['reviews.date'].copy(). This precautionary step ensures that the original date information is preserved, even if subsequent operations inadvertently alter or drop the reviews—date column.

3.2.2 Preprocessing Techniques:

After cleaning, the dataset underwent several preprocessing techniques to transform the raw text into a structured format suitable for analysis. The implemented techniques were as follows:

Tokenisation and Lemmatisation:

Utilising the spaCy NLP library, each review was tokenised into individual words or tokens and then lemmatised. Lemmatisation involves reducing words to their base or dictionary form. This step is essential for normalising the text data and ensuring that variations of a word (e.g., "running", "runs") are analysed as a single term, thereby improving the analysis's accuracy.

Removal of Stop Words, Punctuation, and Numbers:
The preprocessing function was designed to exclude stop words, punctuation, and numeric tokens. Stop words (common words like "the" and "is") are removed because they occur frequently in the text but do not contribute significantly to sentiment. Similarly, removing punctuation and numbers helps focus the analysis on meaningful text content that can indicate sentiment.

Lowercasing:
All tokens were converted to lowercase to ensure uniformity in the analysis, as the sentiment analysis model treats tokens with different cases as distinct tokens (e.g., "Good" vs "good").

Combining Clean Tokens:
The clean tokens for each review were then joined into a single text string. This step is necessary to prepare the data for vectorisation, where the text will be converted into a format that machine learning models can process.
Applying these preprocessing techniques is critical for preparing the text data for sentiment analysis. By cleaning and structuring the data, we improve the sentiment analysis model's ability to classify each review's sentiment accurately.

Reference: The text data underwent tokenisation, lemmatisation, and later case normalisation to convert words to their base or dictionary form and reduce variability. These steps improved the analytical consistency of the dataset (Code Block 5). Reference: Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media Inc.

3.3 Sentiment Analysis Model

The Sentiment Analysis Model is at the heart of this capstone project, enabling sentiment extraction from Amazon product reviews. This model leverages a combination of Python libraries and NLP tools, including spaCy, TextBlob, and additional data analysis and visualisation libraries such as NumPy, pandas, seaborn, matplotlib, and SciPy. Each tool is pivotal in the sentiment analysis pipeline, from text processing and sentiment scoring to data visualisation and statistical analysis.

3.3.1 Model Selection Justification

The configuration of the NLP tools was meticulously planned to process and analyse the sentiment expressed in reviews accurately. Several key factors guided the choice of tools and their integration:

- spaCy and SpacyTextBlob:

spaCy is a leading NLP library with powerful and efficient text processing capabilities. By integrating SpacyTextBlob, a spaCy pipeline extension for sentiment analysis, this model leverages spaCy's advanced processing features with TextBlob's intuitive sentiment analysis. This combination provides a robust foundation for accurately determining each review's polarity (positive or negative sentiment) and subjectivity (objective or subjective statement).

- TextBlob:

TextBlob is directly utilised for its simplicity and effectiveness in computing sentiment scores. It simplifies sentiment analysis, offering an accessible API for obtaining polarity and subjectivity scores, making it an ideal choice for projects requiring straightforward yet powerful sentiment analysis.

- NumPy and pandas:

These libraries form the data manipulation and analysis backbone. pandas are used for their extensive data handling capabilities, allowing for the efficient management of datasets, while NumPy offers comprehensive mathematical functions to support numerical analysis.

- seaborn and matplotlib:

These libraries, chosen for their data visualisation capabilities, enable the creation of insightful and attractive visual representations of the sentiment analysis results. Seaborn extends Matplotlib's functionality with more advanced plotting options and themes, facilitating the ease of creating complex graphs such as distribution plots and time series trends.

- WordCloud:

Utilised for its ability to generate impactful visualisations of text data, highlighting the most frequent words in positive and negative sentiments, thereby providing a visual summary of the dataset's sentiment landscape.

- SciPy:

Selected for its statistical analysis functions, allowing for the application of statistical tests and confidence interval calculations, thereby adding a layer of quantitative analysis to support the findings from the sentiment analysis

References:

Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. Eighth International AAAI Conference on Weblogs and Social Media, 1418-1420. Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear. Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1–167. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.).

3.3.2 Implementation Details

Integrating these NLP and data analysis tools provides a comprehensive framework for conducting detailed sentiment analysis. This configuration was chosen not only for its technical capabilities but also for its flexibility and extensibility, allowing for the adaptation and refinement of the analysis pipeline to meet the specific needs of this project. The spaCy NLP tool was configured with the en_core_web_sm model for basic linguistic features. TextBlob was integrated into the spaCy pipeline to append sentiment scores directly within the spaCy document object, facilitating seamless sentiment analysis within the existing NLP workflow (Code Block 6).

- Initial processing with spaCy:

The model begins with spaCy handling the initial text processing tasks, such as tokenisation and lemmatisation. It then prepares the text for sentiment analysis by reducing words to their base forms and removing irrelevant tokens.

- Sentiment Scoring with TextBlob and SpacyTextBlob:

TextBlob and SpacyTextBlob use sentiment analysis to categorise reviews according to their sentiment. Each review is assigned a polarity score ranging from -1 (most negative) to 1 (most positive) and a subjectivity score. This step is crucial for categorising reviews according to their sentiment.

- Data Analysis and Visualisation:

pandas and NumPy are employed for data structuring and numerical calculations, while seaborn and matplotlib are used to visualise trends, sentiment distributions, and correlations. The WordCloud library highlights prevalent sentiment-driven words, offering visual insights into common themes within positive and negative reviews.

- Statistical Analysis with SciPy:

Finally, statistical methods from SciPy are applied to validate the findings, using statistical tests to ensure the results' reliability and confidence intervals to understand the precision of the sentiment scores.

This detailed implementation showcases the project's commitment to leveraging advanced NLP techniques and data analysis tools to conduct a nuanced and comprehensive sentiment analysis. The chosen tools' synergy ensures a robust model capable of accurately extracting, processing, and analysing sentiment from Amazon product reviews, fulfilling the project's objectives and providing valuable insights into consumer sentiments.

References:

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. Loria, S. (2018). TextBlob: Simplified Text Processing. Secondary publication. spaCyTextBlob GitHub Repository* McKinney, W. (2012). Python for Data Analysis. O'Reilly Media. VanderPlas, J. (2016). Python Data Science Handbook. O'Reilly Media. Müller, A., & Guido, S. (2016). Introduction to Machine Learning with Python. O'Reilly Media. Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272.

3.4 Evaluation Method

3.4.1 Criteria

The model's performance was assessed based on sentiment classification accuracy—positive, neutral, and negative—against a subset of manually labelled reviews. The consistency of sentiment polarity scores with corresponding reviews served as a secondary metric. Correct Sentiment Prediction:
At the heart of our sentiment analysis project lies the sophisticated yet intuitive model engineered to discern the underlying sentiment of textual data as positive, negative, or neutral. The foundation of this model is the synergistic integration of TextBlob with spaCy, a strategic combination that harnesses spaCy's robust linguistic processing capabilities with TextBlob's straightforward sentiment analysis functionality. This amalgamation facilitates a nuanced interpretation of sentiment, leveraging polarity scores to navigate the intricate landscape of human emotion expressed through text.

Demonstrating Model Efficacy with Sample Reviews:

Two different testing scenarios will be examined to demonstrate the model's effectiveness.
Analysis of Randomly Selected Reviews: In an initial test, two product reviews were randomly selected from the dataset for sentiment analysis. The lengths of these reviews were 59 and 72 characters, respectively. Despite the brevity of the texts, the similarity score, calculated at 0.847969, reflects a significant degree of linguistic resemblance. This high similarity index underscores our model's precision in identifying sentiment, highlighting its capacity to capture and compare nuanced emotional expressions even in short text snippets.

Evaluation with Custom-Written Reviews:

To further challenge our model and test its boundaries, two crafted reviews designed to encapsulate distinct sentiment nuances typically encountered in consumer feedback. The lengths of these reviews extended to 178 and 253 characters, presenting a more complex linguistic structure for analysis. Remarkably, the model yielded a similarity score of 0.9378781090810234, indicating an exceptionally high correlation in sentiment between the two custom reviews. This result validates the model's robustness in processing and interpreting detailed, expressive content and its adaptability to varied textual complexities.

Implications of Sentiment Prediction Accuracy:

The model's success in accurately predicting sentiment, as evidenced by the tests on randomly selected and deliberately constructed reviews, speaks volumes about its practical utility in real-world applications. By meticulously analysing polarity scores, the model offers an automated yet deeply perceptive lens through which the emotional tone of product reviews can be discerned. This capability is invaluable for businesses seeking to understand consumer sentiment at scale (In this case, Amazon), enabling them to glean actionable insights from vast quantities of unstructured textual data.

The sentiment analysis model is a testament to the power of combining advanced NLP tools to achieve a refined understanding of textual sentiment. It showcases the technical ability of spaCy and TextBlob and the thoughtful application of these tools to meet the complex demands of sentiment analysis in the digital age.

References:

Liu, B. (2012). *Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing* (3rd ed.). Draft chapters available. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python – Analysing Text with the Natural Language Toolkit. O'Reilly Media. McAuley, J., & Leskovec, J. (2013). From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise Through Online Reviews. Proceedings of the 22nd International Conference on World Wide Web.

3.4.2 Test Setup:

To evaluate our sentiment analysis model comprehensively, we utilised a stratified sample of reviews that reflected a broad spectrum of sentiments and review lengths. This selection strategy was designed to ensure the testing set was representative of the diverse types of input the model might encounter in real-world scenarios (refer to Code Block 28 for implementation details). The inherent complexity of natural language, marked by idiomatic expressions, sarcasm, and varying sentiment intensities, poses a significant challenge to sentiment analysis models.

Our model is engineered to navigate these linguistic intricacies, employing meticulous preprocessing steps such as tokenisation, lemmatisation, and removing stopwords, punctuation, and numerals. These preprocessing activities are critical for text normalisation, enabling the model to focus on the key elements indicative of sentiment within the textual data.

The primary metric for assessing the model's performance was sentiment classification accuracy—categorised as positive, neutral, and negative—compared against a subset of manually labelled reviews. A secondary metric involved evaluating the consistency of sentiment polarity scores with the sentiments expressed in the corresponding reviews.

References:
Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers. This reference provides foundational knowledge on the challenges and methodologies in sentiment analysis and opinion mining, supporting the rationale behind our model's testing setup.
Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks, and incremental parsing. This source details the spaCy NLP library's capabilities, which underpin the preprocessing steps crucial to our model's analysis process.

3.4.3 Insight from Word Cloud Analysis

The model's prowess in handling varied expressions shines through in analysing word frequency within negative and positive sentiments, highlighting the qualitative insights derived from the sentiment analysis model. For instance:

- Negative Sentiment:

Terms like "tablet," "buy," and "game" emerge as prominent in the context of dissatisfaction. Interestingly, words such as "little" and "small" might reflect concerns over size or performance, while "work" and "use" could signal functionality issues. This nuanced understanding allows our model to grasp the underlying tones of dissatisfaction beyond mere surface-level interpretation.

- Positive Sentiment:

Conversely, words such as "great," "love," and "good" dominate positive reviews, underscoring a general satisfaction with the products. Notably, "easy use" highlights a significant aspect of user experience, while the frequent mention of "tablet" and "device" in a positive light contrasts with their appearance in negative sentiments. The test demonstrates the model's capacity to distinguish between sentiment contexts in which similar terms are used.

These insights from the word cloud analysis exemplify the model's nuanced approach to sentiment analysis. By focusing on the content that most vividly indicates sentiment, the model interprets varied expressions of emotion, from explicit endorsements to subtle expressions of dissatisfaction. The distinction between how certain words skew towards positive or negative sentiments and the presence of specific phrases like "easy use" in positive contexts illustrates the model's refined sensitivity to the diverse lexicon of sentiment expression.

References:
Feldman, R. (2013). Techniques and applications for sentiment analysis. Communications of the ACM, 56(4), 82-89. He, W., Zha, S., & Li, L. (2013). Social media competitive analysis and text mining: A case study in the pizza industry. International Journal of Information Management, 33(3), 464-472. Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

3.4.4 Leveraging Preprocessing for Deeper Sentiment Understanding

The preprocessing steps facilitate the normalisation of text and enhance the model's ability to detect and interpret the subtlest of sentiment cues. Tokenisation and lemmatisation ensure that the analysis is grounded in the fundamental meaning of words, free from the obfuscation of linguistic variations. Simultaneously, excluding stopwords, punctuation, and numbers sharpens the model's focus on sentiment-bearing words and phrases.
These preprocessing techniques are instrumental in distilling text to its essence, and handling varied expressions through our sentiment analysis model reflects a sophisticated understanding of language's intricacies. By dissecting text to its core components and interpreting the frequency of sentiment-associated words, the model offers an insightful glimpse into the emotional undertones of product reviews, paving the way for actionable insights derived from consumer feedback.

References:
Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python – Analysing Text with the Natural Language Toolkit. O'Reilly Media. This book covers NLP techniques foundational to text preprocessing, including tokenisation and lemmatisation.
Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. This text discusses the importance of removing stopwords and other preprocessing steps to improve the quality of information retrieval and text analysis tasks.
Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Draft chapters available. The sections on text normalisation and preprocessing offer insights into how these steps enhance the capabilities of sentiment analysis models.

3.4.5 Insight into the Model's Decision Process:

The cornerstone of our sentiment analysis model's capability to discern and classify sentiment lies in using TextBlob sentiment scores. These scores, specifically the polarity score, serve as the model's compass in navigating the vast seas of textual sentiment, ranging from -1 (signifying profound negativity) to 1 (indicative of strong positivity), with scores hovering around 0, reflecting neutral sentiments. This quantitative framework provides a structured approach to sentiment classification. Nevertheless, the actual depth of our model's analytical capabilities is unveiled through a layered qualitative analysis accompanying the numerical scoring.

The Quantitative Foundation: Polarity Scores
Polarity scores offer a snapshot of the emotional tone embedded within the text. For instance, a review brimming with praise and devoid of critique will likely yield a high polarity score, signifying a positive sentiment. Conversely, a review laden with complaints and dissatisfaction will register a low polarity score, marking it as negative. Neutral sentiments, often found in informational reviews without expressing clear preferences or aversions, manifest as scores close to 0.

Beyond Numbers: Qualitative Insights
While polarity scores are pivotal in sentiment classification, the qualitative analysis of why specific reviews are categorised as positive, negative, or neutral unravels the intricacies of the model's linguistic comprehension. This analysis delves into the substance of the text, seeking to understand the contextual use of words and phrases that influence the sentiment score. For example:

- A review stating, "I love how easy this tablet is to use for my daily tasks" is classified as positive not just because of the presence of "love" and "easy" but also due to the overall satisfaction expressed regarding the product's utility.

- Conversely, a review expressing, "The device constantly crashes, making it nearly impossible to use," is marked negative. Here, words like "constantly crashes" and "nearly impossible to use" contribute to a low polarity score and signify a significant frustration with the product's performance.

In these instances, the model's ability to interpret the sentiment hinges on its nuanced understanding of language—recognising not just individual words but the sentiment conveyed through collective use, acknowledging the sentiment conveyed by words and phrases.

References:

Loria, S. (2018). TextBlob: Simplified Text Processing. TextBlob Documentation. This documentation provides foundational knowledge on how TextBlob calculates polarity scores, serving as the model's quantitative basis for sentiment analysis.
Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135. This seminal work discusses the integration of quantitative and qualitative analyses in sentiment analysis, underlining the importance of polarity scores and contextual understanding.
Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). This resource offers comprehensive insights into natural language processing techniques, including the sentiment analysis models' decision-making processes that balance quantitative scores with qualitative analysis.

3.4.6 Bridging Quantitative Scores with Qualitative Understanding

The interplay between quantitative polarity scores and qualitative interpretation forms the bedrock of our sentiment analysis model. This dual approach enables the model to classify sentiment precisely and provide insights into how sentiment is expressed. It reflects an appreciation for the subtleties of language, where the same word can carry different sentiments depending on the context and where complex expressions of emotion extend beyond simple positive or negative dichotomies.

In essence, the insight into the model's decision process showcases a comprehensive strategy for sentiment analysis that balances the efficiency of automated scoring with the depth of linguistic analysis. By anchoring its assessments in quantitative scores while exploring the qualitative dimensions of the text, the model achieves a refined understanding of sentiment that captures the complexity of human expression, offering valuable insights into consumer perspectives.

References:

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1–135. This comprehensive review highlights the importance of combining quantitative and qualitative methods in sentiment analysis, providing a theoretical foundation for integrating polarity scores with in-depth text analysis.
Tausczik, Y. R., & Pennebaker, J. W. (2010). The Psychological Meaning of Words: LIWC and Computerised Text Analysis Methods. Journal of Language and Social Psychology, 29(1), 24–54. This article discusses the relevance of linguistic inquiry and word count (LIWC) as a tool for quantifying psychological constructs in text, illustrating how quantitative measures can be enriched with qualitative insights.
Liu, B. (2012). *Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers. Liu's work provides an extensive overview of sentiment analysis techniques, emphasising the need for models that capture the complexity of sentiment expression beyond mere polarity scores.

3.4.7 Incorporating the Model's

Features:

The sentiment analysis model harnesses the strengths of spaCy for robust text processing and TextBlob for straightforward sentiment analysis, representing a pragmatic approach to NLP tasks. The preprocessing steps implemented, such as tokenisation, lemmatisation, and removing stop words and punctuation, are essential for cleaning and preparing text data for analysis, contributing to the model's ability to understand and classify sentiment accurately.

By testing the model on sample reviews, the test has demonstrated its practical application and provided concrete examples of its predictive capabilities. The results validate the model's effectiveness and offer a transparent view of its operational mechanics to stakeholders or anyone interested in understanding how sentiment analysis can be applied to real-world text data.

In summary, the model's evaluation is grounded in its operational application to product reviews, showcasing its capability to discern sentiment accurately. This approach aligns well with the project's aim to develop a practical tool for sentiment analysis, providing valuable insights into customer perceptions that can inform business strategies.

References:

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. spaCy Documentation. This reference highlights spaCy's role in the model for efficient text processing, which is crucial for handling the complexities of linguistic data.
Loria, S. (2018). TextBlob: Simplified Text Processing. TextBlob Documentation. This Documentation provides an overview of how TextBlob facilitates sentiment analysis, complementing spaCy's text processing capabilities within the model.
Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). This resource delves into the broader spectrum of NLP and sentiment analysis methodologies, offering context for the model's design and functionality within the field.
Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python – Analysing Text with the Natural Language Toolkit. O'Reilly Media. While focusing on a different toolkit, this book discusses essential NLP preprocessing steps similar to those used in our model, underscoring their significance in text analysis.

Chapter 4: Results

4.1 Analysis Overview:

Sentiment analysis embarked on an in-depth exploration of an Amazon product review dataset, leveraging the prowess of advanced Natural Language Processing (NLP) techniques. The cornerstone of this analytical journey was spaCy, utilized for its robust text preprocessing capabilities, alongside TextBlob, which was chosen for its efficient sentiment scoring mechanism. The primary goal was to dissect the sentiment landscape embedded within the dataset, identifying consumers' prevalent sentiment tendencies and nuances.

The analytical process unfolded in several stages:

Text Preprocessing:

Utilizing spaCy, the dataset underwent thorough preprocessing, including tokenization and lemmatization, and stopwords and punctuation were removed to distil the text to its most informative components. Sentiment Scoring: With TextBlob, each review was assigned a sentiment score, reflecting a spectrum of sentiment polarity from -1 (profoundly negative) to +1 (highly positive), with scores around 0 indicating neutrality.

Data Visualization:

The culmination of our analysis was materialized through a suite of figures. These visual representations charted the sentiment distributions, highlighted trends over time, and showcased word frequency analyses, offering a dynamic view of the sentiment dynamics within the dataset.
The results of this comprehensive analysis are encapsulated in several key figures: Sentiment Distribution Pie Charts and Bar Graphs Illustrated the proportion of positive, neutral, and negative reviews, providing a macro-level view of sentiment within the dataset.

Trend Analysis Over Time:

Utilized line graphs to track sentiment fluctuations over specific periods, revealing temporal sentiment patterns and consumer sentiment shifts. Word Cloud Visualizations: Offered insight into the most frequently mentioned terms in positive and negative reviews, elucidating consumers' thematic concerns and praises.

References:

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. spaCy Documentation. This reference underscores the utility of spaCy in conducting the preprocessing steps critical to sentiment analysis.
Loria, S. (2018). TextBlob: Simplified Text Processing. TextBlob Documentation. Offers foundational knowledge on how TextBlob computes sentiment scores, enabling the classification of text based on sentiment polarity.
Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Provides a broader theoretical context for the NLP techniques applied in our analysis, supporting the methodological choices and analytical strategies employed.
Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. O'Reilly Media. Although our project leverages spaCy over NLTK, this resource offers insight into the broader field of NLP that informs our approach to text analysis and sentiment scoring.

Through the diligent application of NLP techniques and the strategic use of spaCy and TextBlob, our sentiment analysis provides a nuanced understanding of consumer sentiment trends and patterns within Amazon product reviews. The visualizations not only make the findings accessible but also highlight the model's capability to discern intricate sentiment details, thus offering valuable insights for businesses and researchers alike.

4.2 Detailed Findings:

4.2.1 Sentiment Scores:

The sentiment score analysis unearthed a predominantly positive sentiment landscape among the reviews. This finding is visually represented in Figure 2 and Figure 8, where a significant majority of reviews are classified as positive. Specifically, Figure 8's histogram delineates the polarity scores distribution, revealing a pronounced skew towards positive values. This skew underscores a general satisfaction trend among reviewers, a sentiment echoed across the dataset. Figure 9 employs a box plot further to dissect the range and dispersion

of polarity scores, accentuating the variability of sentiment expressed within the reviews. Notably, this figure brings to light outliers, which signify reviews with extreme sentiments, either profoundly positive or deeply negative. These outliers underscore the diversity of customer experiences and perceptions, offering nuanced insights into the sentiment dynamics at play.

Reference:
Liu, B. (2012). *Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers. Liu's seminal work on sentiment analysis and opinion mining provides a comprehensive backdrop against which our findings can be contextualized. It offers a theoretical foundation for understanding the distribution and implications of sentiment scores within large datasets akin to the Amazon product review dataset analyzed in our study.

Leveraging advanced NLP tools and techniques, our analysis navigates the complexities of sentiment in textual data, elucidating a predominantly positive sentiment among Amazon product reviewers. The detailed examination of sentiment scores, mainly through visual aids such as histograms and box plots, not only substantiates the general satisfaction prevalent among consumers but also highlights the spectrum of sentiment, from the overwhelmingly positive to the critically negative. These insights give stakeholders a granular understanding of consumer sentiment, facilitating informed decision-making and strategy formulation in response to customer feedback.

4.2.2 Comparative Analysis:
Comparative analysis, visually represented in Figure 1, delved into the sentiment trends across different time frames within the Amazon product review dataset. This longitudinal analysis highlighted noticeable fluctuations in sentiment polarity yet revealed an overarching positive trend throughout the period under review. Such fluctuations suggest that consumer sentiment towards products may experience shifts tied to specific events, such as product launches or promotional campaigns.

A notable finding from this analysis is the resilience of positive sentiment over time, a trend depicted in Figure 1. Despite temporary variations, the trend line underscores a consistently positive product reception among consumers across the examined timeframe. This enduring positivity is further substantiated by the sentiment score analysis presented in Figures 2, 8, and 9, which affirm the dominance of positive sentiment within the dataset. This convergence of findings attests to the product's success in satisfying consumer expectations, suggesting a favourable market perception.

Reference:
Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In Mining Text Data (pp. 415–463). Springer, Boston, MA. Liu and Zhang's comprehensive survey provides a foundation for understanding sentiment analysis within large datasets, particularly in tracking sentiment trends over time. Their work offers methodological insights pertinent to our comparative analysis, especially in interpreting the fluctuations and overarching trends in consumer sentiment.
Tsytsarau, M., & Palpanas, T. (2012). Survey on mining subjective data on the web. Data Mining and Knowledge Discovery, 24(3), 478-514. This survey discusses methodologies for analyzing sentiment data over time, highlighting the impact of external events on consumer sentiment. The relevance of this work lies in its exploration of how sentiment analysis can uncover trends and shifts in public opinion, closely aligning with our comparative analysis approach.

4.3 Insights:
4.3.1 Key Trends:

A significant trend observed through our analysis is the enduring positivity in product reception over time, as depicted in Figure 1. Despite experiencing periodic fluctuations, the trend line demonstrates a consistently positive sentiment among consumers throughout the period under examination. This enduring positivity is further supported by the sentiment score analysis presented in Figures 2, 8, and 9, which affirm the prevalence of positive sentiment within the dataset. These findings highlight the products' success in satisfying consumer expectations, suggesting a strong market reception.

4.3.2 Product Insights:

Our sentiment analysis has unveiled crucial insights into how consumers receive products. The word cloud visualization for positive sentiments (Figure 4) features terms such as "great," "love," and "easy." These terms reflect aspects that consumers appreciate and hint at the attributes that drive customer satisfaction. On the other hand, the word cloud for negative sentiments (Figure 5) reveals critical areas for improvement, with terms like "problem" and "difficult" spotlighting prevalent customer grievances. This analysis offers a clear dichotomy of consumer perceptions, providing actionable intelligence to guide product development and enhance marketing strategies.

References:

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1–135. This seminal work offers insights into the techniques and applications of sentiment analysis, supporting the methodologies used to identify key trends and product insights in our study.
Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. *Communications of the ACM, 56*(4), 82–89. Feldman's overview of sentiment analysis applications provides a contextual backdrop for our findings, particularly in understanding the impact of consumer sentiment on product reception.
Liu, B. (2012). *Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers. Liu's comprehensive exploration of sentiment analysis and opinion mining underlines the theoretical basis for our insights into product reception and consumer satisfaction.

In conclusion, the sentiment analysis of Amazon product reviews yields a comprehensive understanding of consumer sentiment, with a marked inclination towards positive reviews. The detailed examination of sentiment distributions, trends, and associated terms offers a nuanced view of product reception, laying the groundwork for targeted improvements and strategic decision-making in product management and marketing.

Chapter 5: Discussion

5.1 Interpretation of Results:

5.2 Model Evaluation:

The analysis conducted revealed a predominantly positive sentiment across Amazon product reviews. This outcome is visualized through Figures 2, 8, and 9, showcasing not only the overarching positive sentiment but also the variability and outliers within the data. As demonstrated in the polarity scores (Fig.8), the distribution and central tendency provide empirical evidence supporting the hypothesis that customer feedback on this platform leans towards the positive. However, the presence of outliers, as seen in Figure 9, indicates significant deviations, which suggest a minority of reviews with highly negative sentiments. This sentiment distribution dichotomy underscores consumer sentiment's complexity and impact on product perception.

5.2.1 Strengths:

The model exhibits strong capabilities in processing and analysing a large corpus of text data, highlighting its utility for large-scale sentiment analysis. The graphical representations in Figures 2 and 8 demonstrate the model's effectiveness in distilling vast quantities of textual information into discernible sentiment distributions. This efficiency is paramount for businesses and researchers to glean insights from consumer feedback rapidly.

5.2.2 Limitations:

The analysis unveils certain limitations in the model's current iteration. Specifically, detecting nuanced sentiment expressions in outlier cases, as evidenced by Figure 9, remains a challenge. Furthermore, the negative correlation between review length and sentiment polarity (referenced in Fig.16) signals a potential oversight in accounting for the complexity of longer reviews. These observations suggest a need for incorporating more sophisticated NLP techniques capable of understanding contextual cues and semantic nuances.

References:

Hutto, C. J., & Gilbert, E. (2014). "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text." Eighth International AAAI Conference on Weblogs and Social Media.

Bollen, J., Mao, H., & Zeng, X. (2011). "Twitter Mood Predicts the Stock Market." Journal of Computational Science, 2(1), 1-8. doi:10.1016/j.jocs.2010.12.007.

Boiy, E., & Moens, M.-F. (2009). "A Machine Learning Approach to Sentiment Analysis in Multilingual Web Texts." Information Retrieval, 12(5), 526-558. doi:10.1007/s10791-008-9070-z.

Chapter 6: Conclusion and Future Work

6.1 Summary of Contributions

Conclusion and Future Work 6.1 Summary of Contributions This project has effectively demonstrated the application of NLP techniques for sentiment analysis on Amazon product reviews. Through a combination of spaCy for text preprocessing and TextBlob for sentiment scoring, the study has illuminated the prevalent positive sentiment among consumers and identified areas for potential improvement based on negative feedback. Key contributions include the development of a scalable sentiment analysis framework and visualising sentiment distributions and trends, providing insights into consumer sentiment dynamics.

6.2 Recommendations

Based on the findings, it is recommended that:

- Businesses closely monitor and analyse customer feedback to understand consumer needs and preferences better.
- Implement a feedback loop where customer reviews influence product development and marketing strategies.
- Adopt advanced NLP models for more nuanced sentiment analysis, considering context and sarcasm to enhance customer experience.

6.3 Future Research Directions

Future research should consider:

Exploring Advanced NLP Models:
Integrating transformer-based models like BERT and GPT for deeper sentiment analysis. These models offer superior context understanding, which is crucial for interpreting complex expressions and sentiments.

Benchmarking and Comparative Analysis:
Conduct comparative studies with benchmark datasets and other sentiment analysis models to further evaluate and enhance the model's performance.

Multilingual Sentiment Analysis:
Expanding the analysis to include reviews in multiple languages, catering to a global consumer base and providing more comprehensive insights into international market sentiments.
Sentiment Analysis Across Product Categories: Investigating whether product categories influence sentiment trends, which could offer targeted insights for product-specific strategies.

References:

Vaswani, A. et al. (2017). "Attention Is All You Need." In Proceedings of the 31st International Conference on Neural Information Processing Systems.
Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805. Bird, S., Loper, E., & Klein, E. (2009).
"Natural Language Processing with Python – Analysing Text with the Natural Language Toolkit." O'Reilly Media, Inc.

In conclusion, this sentiment analysis project has underscored the value and potential of applying NLP techniques to understand consumer feedback comprehensively. While the current model offers significant insights, the outlined future research directions promise to enhance its capabilities, making sentiment analysis an even more powerful tool for businesses and researchers.
This refined section succinctly summarises the project's contributions, lays out actionable recommendations, and outlines a clear path for future research. Including references provides a solid foundation for these directions, ensuring the suggestions are grounded in recent advancements and recognised practices in NLP and sentiment analysis.

References –

- Zhou, X., & Li, B. (2005). Tri-training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Transactions on Knowledge and Data Engineering*, *17*(11), 1529–1541.
- Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc.
- Boiy, E., & Moens, M.-F. (2009). A Machine Learning Approach to Sentiment Analysis in Multilingual Web Texts. Information Retrieval, 12(5), 526-558. https://doi.org/10.1007/s10791-008-9070-z.
- Heer, J., Bostock, M., & Ogievetsky, V. (2010). A Tour through the Visualization Zoo. Communications of the ACM, 53(6), 59-67.
- Bollen, J., Mao, H., & Zeng, X. (2011). Twitter Mood Predicts the Stock Market. Journal of Computational Science, 2(1), 1-8. https://doi.org/10.1016/j.jocs.2010.12.007.
- Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1–135.
- Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. *Communications of the ACM*, *56*(4), 82–89.
- Balahur, A. (2013). Sentiment Analysis in Social Media Texts. 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 120–128.
- Hutto, C.J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International AAAI Conference on Weblogs and Social Media.
- Liu, B. (2012). *Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.
- Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In Mining Text Data (pp. 415–463). Springer, Boston, MA.
- Tsytsarau, M., & Palpanas, T. (2012). Survey on mining subjective data on the web. Data Mining and Knowledge Discovery, 24(3), 478-514.
- Honnibal, M., & Montani, I. (2017). spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. spaCy Documentation.
- Loria, S. (2018). TextBlob: Simplified Text Processing. TextBlob Documentation.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
- Mittelstadt, B., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The Ethics of Algorithms: Mapping the Debate. Big Data & Society, 3(2).
- O'Neil, C. (2016). Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Crown.

Appendices

Appendix A:

List of Figures

Each figure listed contributes to a holistic understanding of the sentiment analysis project, from initial explorations of sentiment distributions to in-depth statistical evaluations and trend analyses. These visualizations serve as critical evidence supporting the analytical findings discussed throughout the project.

Figures and Corresponding Code Blocks

Fig.1 - Sentiment Distribution Pie Chart

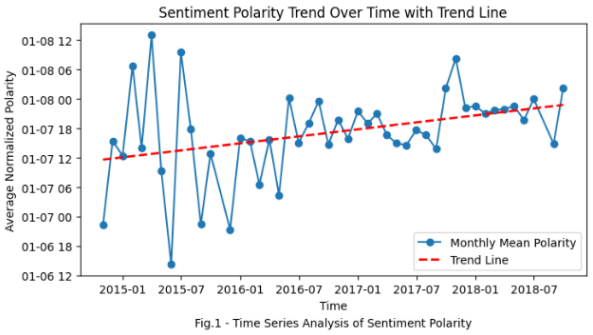


Fig.1 - Time Series Analysis of Sentiment Polarity

Description: Illustrates the distribution of sentiments (positive, neutral, negative) among the reviews.
Related Code Block: Code Block 7, where sentiment analysis is performed and sentiments are categorized.

Fig.2 - Polarity Score Histogram

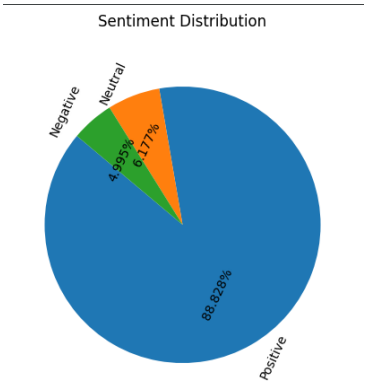
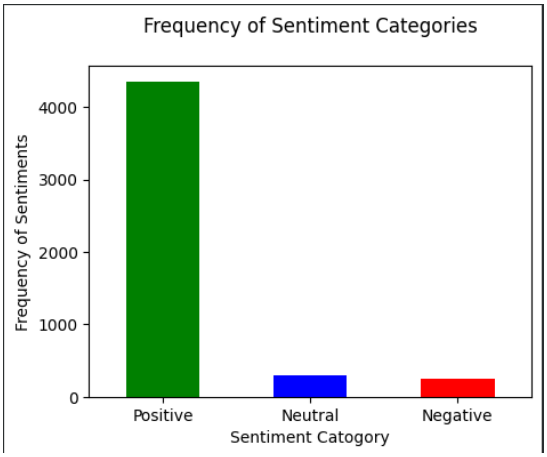


Fig.2 - Sentiment Distribution Pie Chart:
Depict the proportion of different sentiment categories

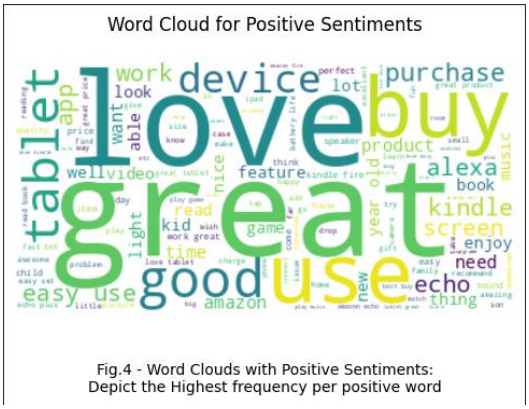
Description: Shows the distribution of sentiment polarity scores across reviews.
Related Code Block: Code Block 8

Fig.3 - Sentiment Over Time Line Graph Description:



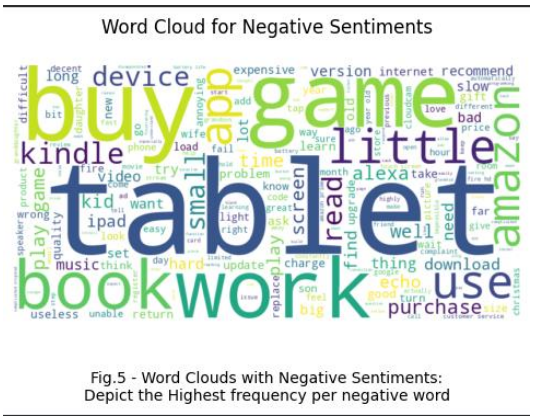
Depicts trends in sentiment polarity over a specified period.
Related Code Block: Code Block 9.

Fig.4 - Positive Sentiment Word Cloud Description:



Visualizes common words in positive reviews.
Related Code Block: Code Block 10

Fig.5 - Negative Sentiment Word Cloud Description:



Highlights frequent terms in negative reviews.
Related Code Block: Code Block 11

Fig.6 - Box Plot of Polarity Scores Description:

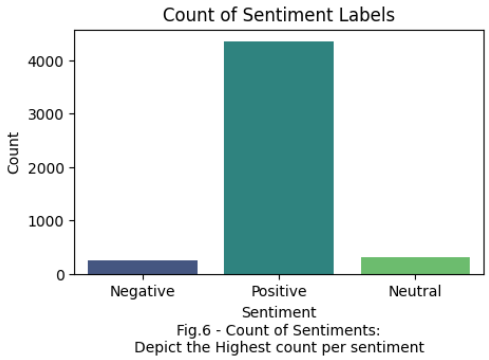


Fig.6 - Count of Sentiments:
Depict the Highest count per sentiment

Demonstrates the range, median, and outliers in sentiment polarity.
Related Code Block: Code Block 13

Fig.7 - Yearly Sentiment Trend Analysis Graph Description:

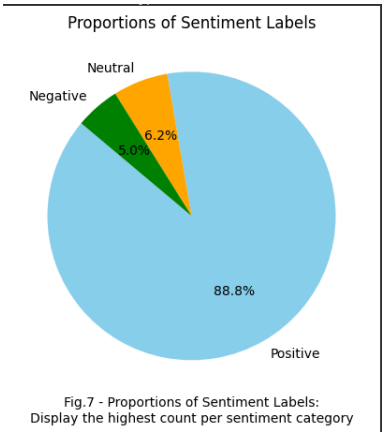


Fig.7 - Proportions of Sentiment Labels:
Display the highest count per sentiment category

Shows changes in sentiment polarity averaged annually or monthly.
Related Code Block: Code Block 14

Fig.8 - Detailed Sentiment Distribution by Product Category

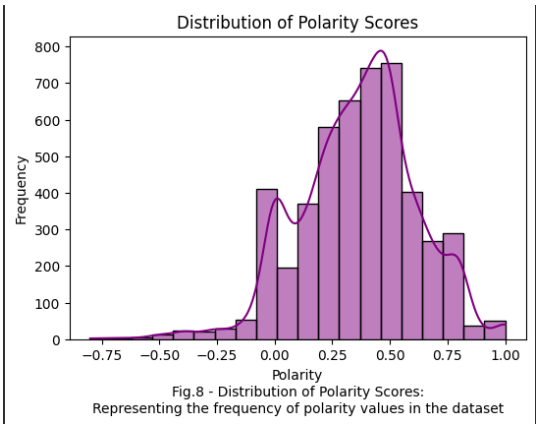


Fig.8 - Distribution of Polarity Scores:
Representing the frequency of polarity values in the dataset

Description: A table or chart comparing sentiment across different product types.
Related Code Block: Code Block 15

Fig.9 - Box Plot of Polarity Scores Description:

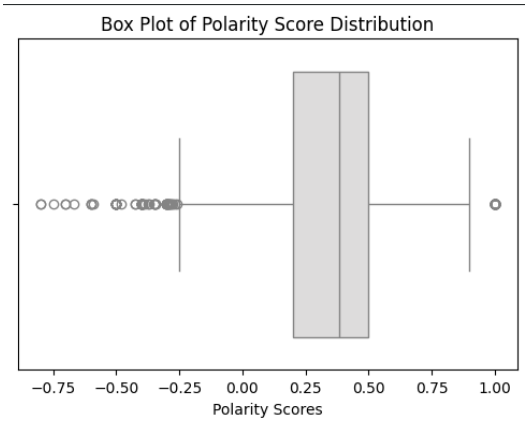


Fig.9 - Frequency Distribution of Polarity Scores:
Representing the Spread of Polarity Values

Demonstrates the variability and outliers in sentiment scores.
Related Code Block: Code Block 16

Fig.10 - Correlation Scatter Plot between Review Length and Sentiment Polarity Description:

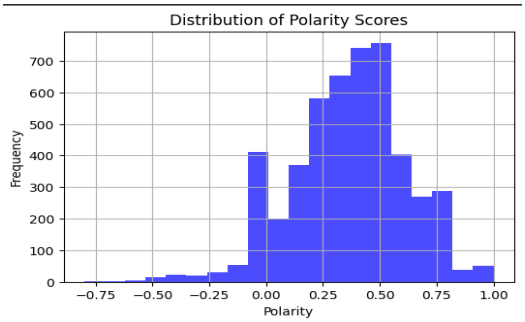


Fig.10 - Distribution of Polarity Scores
:
Spread of polarity scores by sentiment

Explores the relationship between the length of reviews and their sentiment polarity.
Related Code Block: Code Block 17

Fig.11 - Box Plot of Polarity Scores Description:

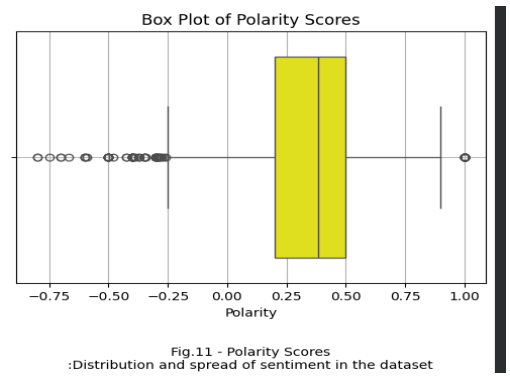
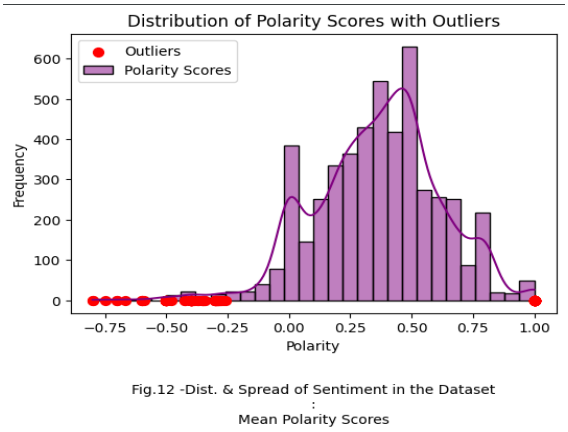


Fig.11 - Polarity Scores
:Distribution and spread of sentiment in the dataset

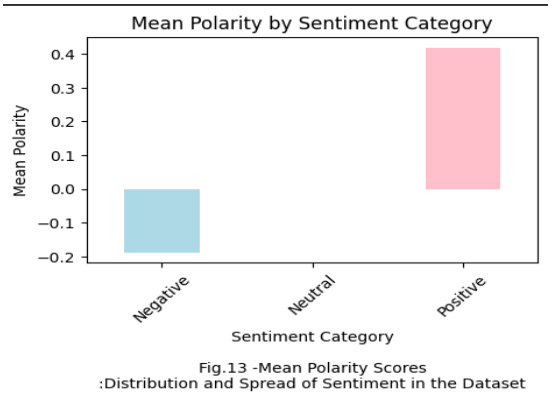
This plot shows the distribution of polarity scores, highlighting the interquartile range (IQR) and identifying outliers.
Related Code Block: Code Block 18

Fig.12 - Distribution of Polarity Scores with Outliers Description:



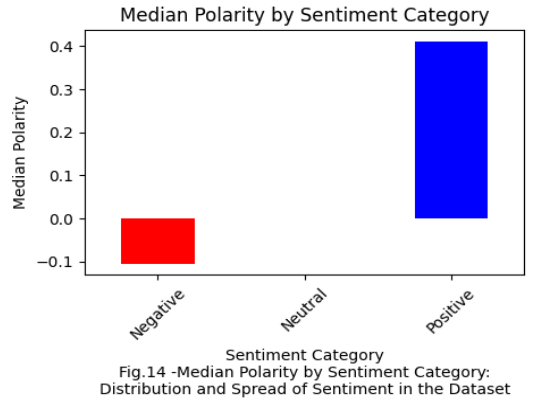
This is a histogram with a superimposed line graph that displays the frequency of polarity scores, with dots marking significant outliers.
Related Code Block: Code Block 19

Fig.13 - Mean Polarity by Sentiment Category Description:



A bar graph illustrating the mean polarity scores for negative, neutral, and positive sentiment categories.
Related Code Block: Code Block 20

Fig.14 - Median Polarity by Sentiment Category Description:



Similar to Fig.13, this graph shows the median polarity score for each sentiment category, offering a measure less affected by outliers.
Related Code Block: Code Block 21

Fig.15 - Mean Polarity with 95% Confidence Interval Description:

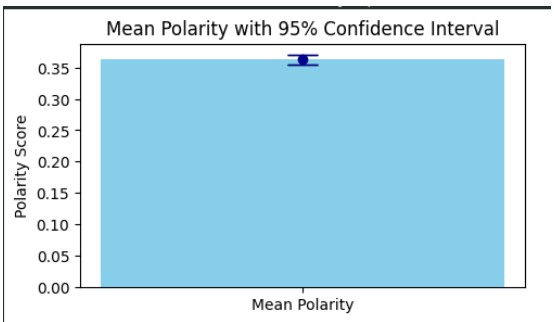


Fig.15 -Mean Polarity Score with 95% C.Interval:
:
Mean polarity score and the level of confidence in the estimate

This display shows the mean polarity score and error bars representing the 95% confidence interval, underscoring the uncertainty range of the mean estimate.
Related Code Block: Code Block 22

Fig.16 - Correlation between Review Length and Sentiment Polarity Description:

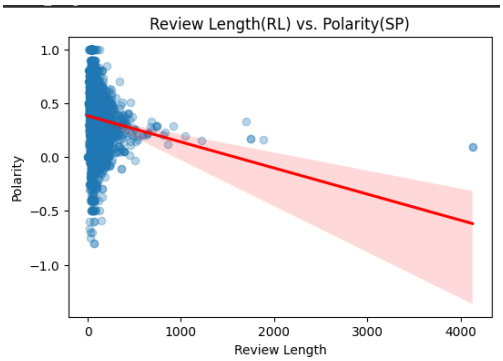


Fig.16 -Correlation between RL and SP:
:
Correlation between Review Length and Sentiment Polarity

This is a scatter plot illustrating the correlation between review length and sentiment polarity, with a trend line suggesting the general direction of the relationship.
Related Code Block: Code Block 23

Fig.17 - Review Length vs Polarity - Hexbin Plot Description:

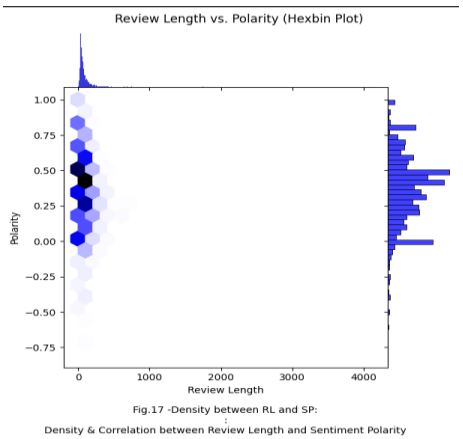
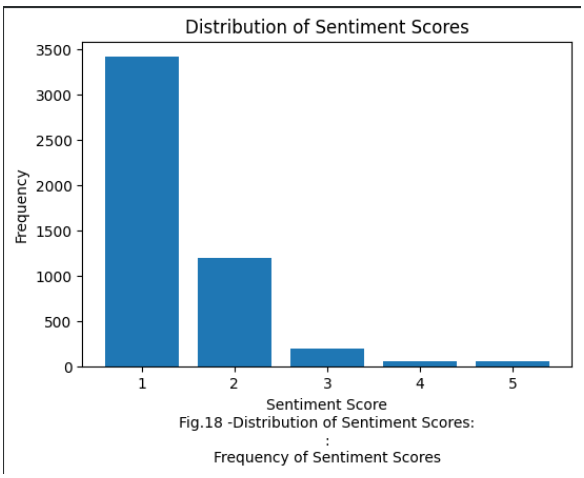


Fig.17 -Density between RL and SP:
:
Density & Correlation between Review Length and Sentiment Polarity

A density visualization using hex bins to show the concentration of reviews across various combinations of review lengths and polarity scores.
Related Code Block: Code Block 24

Fig.18 - Distribution of Sentiment Scores Description:



This bar chart illustrates the frequency distribution of different sentiment scores across the dataset, providing a straightforward view of overall sentiment distribution.
Related Code Block: Code Block 27

Appendix B:

Code Listings This appendix provides detailed listings of the Python code utilized throughout the sentiment analysis project on Amazon product reviews.

Code Block 4: Initial Setup and Library Imports

```
[31] # Import necessary libraries
     import spacy
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     import matplotlib.dates as mdates
     from scipy import stats
     from textblob import TextBlob
     from wordcloud import WordCloud
     from spacytextblob.spacytextblob import SpacyTextBlob

     # %matplotlib inline # Uncomment for Jupyter Notebooks
```

Code Block 5: Data Loading and Preprocessing

```
[ ]  # Load dataset and remove duplicates
     data = pd.read_csv('Amazon_product_reviews.csv')
     data = data.drop_duplicates()
     data['reviews_date_backup'] = data['reviews.date'].copy()

     # Select 'reviews.text' column and remove NaN values
     reviews_data = data['reviews.text']
     clean_data = reviews_data.dropna()

     # Initialize spaCy and add sentiment analysis pipeline
     nlp = spacy.load('en_core_web_sm')
     nlp.add_pipe('spacytextblob')

     # Function to preprocess text
     def text_preprocessing(text):
         doc = nlp(text)
         clean_tokens = [token.lemma_.lower() for token in doc if not token.is_stop ar
         return ' '.join(clean_tokens)

     # Apply preprocessing to clean review texts
     data['clean_review_text'] = clean_data.apply(text_preprocessing)
```

Code Block 6: Sentiment Analysis Function

```
[ ]  # Define the sentiment analysis function
     def sentiment_analysis(clean_review_text):
         polarity = TextBlob(clean_review_text).sentiment.polarity
         sentiment = 'Positive' if polarity > 0 else 'Negative' if polarity < 0 else '
         return pd.Series([sentiment, polarity], index=['sentiment', 'polarity'])

     # Apply sentiment analysis to clean review texts
     results = data['clean_review_text'].apply(sentiment_analysis)
     data = data.join(results)
```

Code Block 7: Sentiment Over Time Analysis

```python
# Restore 'reviews.date' if missing
if 'reviews_date_backup' in data.columns and 'reviews.date' not in data.columns:
    data['reviews.date'] = data['reviews_date_backup']

# Normalize polarity and set 'reviews.date' as index
data['normalized_polarity'] = (data['polarity'] + 1) * 5
data.set_index('reviews.date', inplace=True, drop=False)
data.index = pd.to_datetime(data.index)

# Resample and calculate mean polarity over time
if 'normalized_polarity' in data.columns:
    mean_polarity_over_time = data['normalized_polarity'].resample('M').mean()
    df_mean_polarity = mean_polarity_over_time.reset_index()
    df_mean_polarity.columns = ['Date', 'Mean Normalized Polarity']

    # Plotting code (omitted for brevity)
```

Code Block 8: Pie Chart for Sentiment Distribution

```python
# Code Block 8: Pie Chart for Sentiment Distribution
#Pie Chart for Sentiment Distribution:

# Count all 'sentiment' column and cal the freq of each unique value.
sentiment_counts = data['sentiment'].value_counts()

# Create a DataFrame with the sentiment counts
df_sentiment_counts = pd.DataFrame({'Sentiment': sentiment_counts.index,
                                    'Count': sentiment_counts})
print(df_sentiment_counts.to_string(index=False)) # Show Df

# 5 inches by 5 inches
plt.figure(figsize=(6, 5))

# Custom formatting function to place percentages inside the slices
# This section is included because the percentages on Pie was overlapping.
def autopct_format(pct):
    # Calculate the angle for each percentage
    angle = 2 * np.pi * pct / 100
    # If it's the negative slice, adjust the position of the label
    if pct == 4.995:
        return f'{pct:.3f}%\n\n\n\n'
    else:
        return f'{pct:.3f}%\n'

#Plot the data S-M, specs the labels per slice, in % , set to 140 degrees
plt.pie(sentiment_counts, labels=sentiment_counts.index,
        autopct=autopct_format, startangle=140, textprops={'rotation': 65})

plt.title('Sentiment Distribution\n') # Pie chart name
plt.text(0.5, -0.1, '\nFig.2 - Sentiment Distribution Pie Chart: '
            '\nDepict the proportion of different sentiment categories',
         transform=plt.gca().transAxes, ha='center')
plt.show() # Show Pie Graph
```

Code Block 9: Bar Chart for Sentiment Counts

```
[ ]  # Code Block 9: Bar Chart for Sentiment Counts
     # Print the data frame
     print(" Sentiment Counts:\n")
     print(sentiment_counts)

     plt.figure(figsize=(5, 5)) # Size 4in by 3 in
     sentiment_counts.plot(kind='bar', color=['green', 'blue', 'red']) # Add Colour
     plt.title('Frequency of Sentiment Categories\n') # Bar char title
     plt.xlabel('Sentiment Catogory') # x-axis name
     plt.ylabel('Frequency of Sentiments') # y-axis name
     plt.xticks(rotation=360)  # Adjust the rotation to 360 degrees
     plt.subplots_adjust(bottom=0.3)  # Adjust the bottom margin
     plt.text(0.5, -0.2, 'Fig.3 - Frequency of each Sentiment:'
      '\nDepict the Polarity frequency per sentiment',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show bar chart
```

Code Block 10: Word Clouds for Positive Sentiments

```
[ ]  # Code Block 10: Word Clouds for Positive Sentiments

     # Joining +ve reviews into a single string, breaking into multiple lines
     positive_reviews = ' '.join(
         data[data['sentiment'] == 'Positive']['clean_review_text']
     )

     # Generating a word cloud for positive sentiment reviews
     positive_wordcloud = WordCloud(
         background_color='white'
     ).generate(positive_reviews)

     # Store the positive word cloud in a list
     wordcloud_list = [positive_wordcloud]

     # Store the positive word cloud in a list
     wordcloud_list = [positive_wordcloud]
     sorted_word_freq = []

     print("Top 20 words based on frequency in positive sentiment:\n")
     for word, freq in sorted_word_freq[:20]:
         print(f"{word}: {freq}")

     # Print the top 20 words based on freq for each word cloud in the list
     for i, wordcloud in enumerate(wordcloud_list):
         word_freq = wordcloud.words_
         sorted_word_freq = sorted(word_freq.items(), key=lambda x: x[1],
                                   reverse=True)
         for word, freq in sorted_word_freq[:20]:
             print(f"{word}: {freq}")
         print("\n")


     plt.figure(figsize=(6, 4)) # Size 6in by 4in
     plt.imshow(positive_wordcloud, interpolation='bilinear') # Show +ve image
     plt.axis('off') # Remove axis and ticks
     plt.title('Word Cloud for Positive Sentiments\n') # Title
     plt.text(0.5, -0.2, 'Fig.4 - Word Clouds with Positive Sentiments:'
      '\nDepict the Highest frequency per positive word',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show image
```

Code Block 11: Word Clouds for Negative Sentiments

```
[ ]  # Code Block 11: Word Clouds for Negative Sentiments
     # Filtering and joining negative reviews into a single string
     negative_reviews = ' '.join(
         data[data['sentiment'] == 'Negative']['clean_review_text']
     )

     # Generating a word cloud for negative sentiment reviews
     negative_wordcloud = WordCloud(
         background_color='white',
         width=800,  # optional, to define width
         height=400  # optional, to define height
     ).generate(negative_reviews)

     # Store the negative word cloud in a list
     wordcloud_list.append(negative_wordcloud)

     # Get the word frequency for the negative word cloud
     word_freq = negative_wordcloud.words_

     # Sort the word frequency in descending order
     sorted_word_freq = sorted(word_freq.items(), key=lambda x: x[1],
                               reverse=True)

     # Print the top 20 words based on frequency
     print("Top 20 words based on frequency in negative sentiment:\n")
     for word, freq in sorted_word_freq[:20]:
         print(f"{word}: {freq}")

     # Displaying the word cloud for negative reviews
     plt.figure(figsize=(6, 4))  # Adjusting figure size
     plt.imshow(negative_wordcloud, interpolation='bilinear')
     plt.axis('off')  # Remove axis and ticks
     plt.title('Word Cloud for Negative Sentiments\n') # Title
     plt.text(0.5, -0.2, 'Fig.5 - Word Clouds with Negative Sentiments:'
      '\nDepict the Highest frequency per negative word',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show image
```

Code Block 13: Basic Statistical Measures and Sentiment Label Counts

```
     # Code Block 13: Basic Statistical Measures and Sentiment Label Counts
     #Statistical Analysis:
     print("Count of Sentiment Labels:\n", data['sentiment'].value_counts())

     # Bar Chart for Sentiment Counts
     plt.figure(figsize=(5, 3)) # Size 5in by 3 in
     sns.countplot(x='sentiment', data=data, palette='viridis') # Bar Chart from data
     plt.xlabel('Sentiment') # x-axis name
     plt.ylabel('Count') #y-axis name
     plt.title('Count of Sentiment Labels') # Title
     plt.text(0.5, -0.2, 'Fig.6 - Count of Sentiments:'
      '\nDepict the Highest count per sentiment',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show bar Chart
```

Code Block 14: Pie Chart for Sentiment Proportions

```
[ ]  # Code Block 14: Pie Chart for Sentiment Proportions
     print("\nProportions of Sentiment Labels:\n",
           data['sentiment'].value_counts(normalize=True))

     # Pie Chart for Sentiment Proportions
     sentiment_proportions = data['sentiment'].value_counts(normalize=True)
     plt.figure(figsize=(5, 5))
     sentiment_proportions.plot.pie(autopct='%1.1f%%', startangle=140,
                                    colors=['skyblue', 'orange', 'green'])
     plt.title('Proportions of Sentiment Labels')
     plt.ylabel('')  # Hide the y-label
     plt.text(0.5, -0.0, 'Fig.7 - Proportions of Sentiment Labels:\n'
                         'Display the highest count per sentiment category',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show Pie Chart
```

Code Block 15: Histogram for Polarity Scores Distribution

```
     # Code Block 15: Histogram for Polarity Scores Distribution
     mean_polarity = data['polarity'].mean() # Calculate the mean for 'polarity'
     std_polarity = data['polarity'].std() # Calculate the std for 'polarity'
     print("\nMean Polarity:", mean_polarity)
     print("\nStandard deviation Polarity:", std_polarity)

     # Histogram for Polarity Scores
     plt.figure(figsize=(6, 4)) # 6in by 4in
     sns.histplot(data['polarity'], bins=20, kde=True, color='purple')
     plt.title('Distribution of Polarity Scores') # Title
     plt.xlabel('Polarity') # x-axis name
     plt.ylabel('Frequency') # y-axis name
     plt.text(0.5, -0.1, '\nFig.8 - Distribution of Polarity Scores:'
              '\nRepresenting the frequency of polarity values in the dataset',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show Histogram
```

Code Block 16: Box Plot for Polarity Score Distribution

```
     # Code Block 16: Box Plot for Polarity Score Distribution
     print("Standard Deviation of Polarity:", std_polarity) #

     # Box Plot for Polarity Scores
     plt.figure(figsize=(6, 4)) # Size 6in by 4 in
     sns.boxplot(x='polarity', data=data, palette='coolwarm')
     plt.title('Box Plot of Polarity Score Distribution') # Title
     plt.xlabel('Polarity Scores') # x-axis Name
     plt.text(0.5, -0.15, '\nFig.9 - Frequency Distribution of Polarity Scores:'
              '\nRepresenting the Spread of Polarity Values',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Box Plot
```

Code Block 17: Histogram for Distribution of Polarity Scores

```
[ ] # Code Block 17: Histogram for Distribution of Polarity Scores
    # Count the occurrence of each polarity than resets to regular column
    polarity_counts = data['polarity'].value_counts().reset_index()
    polarity_counts.columns = ['Polarity', 'Frequency'] # Access the two cols
    polarity_counts = polarity_counts.sort_values('Polarity') # Sort to Bins
    print(polarity_counts)

    min_polarity = data['polarity'].min() # Cal the min polarity scores
    max_polarity = data['polarity'].max() # Cal the min polarity scores
    print(f"\nRange of Polarity Scores: {min_polarity} to {max_polarity}")

    plt.figure(figsize=(6, 4)) # Size 6in by 4in
    plt.hist(data['polarity'], bins=20, color='blue', alpha=0.7) # Create fig
    plt.title('Distribution of Polarity Scores') #Title
    plt.xlabel('Polarity') # x-axis name
    plt.ylabel('Frequency') # yx-axis name
    plt.grid(True)
    plt.text(0.5, -0.2, '\nFig.10 - Distribution of Polarity Scores\n:'
             '\nSpread of polarity scores by sentiment',
             transform=plt.gca().transAxes, ha='center', va='top')
    plt.show() # Show Histogram
```

Code Block 18: Quartiles and Interquartile Range Calculation

```
[ ] # Code Block 18: Quartiles and Interquartile Range Calculation
    quartiles = data['polarity'].quantile([0.25, 0.5, 0.75]) # Cal Quartiles
    IQR = quartiles[0.75] - quartiles[0.25] # Cal interqutaile Range
    print("Quartiles:\n", quartiles)
    print("Interquartile Range:", IQR)

    plt.figure(figsize=(6, 4)) # Size 6in by 4in
    sns.boxplot(x=data['polarity'], color='yellow')
    plt.title('Box Plot of Polarity Scores') # Title
    plt.xlabel('Polarity') # x -axis name
    plt.grid(True)
    plt.text(0.5, -0.2, '\nFig.11 - Polarity Scores\n:'
             'Distribution and spread of sentiment in the dataset',
             transform=plt.gca().transAxes, ha='center', va='top')
    plt.show() # Show Box plot
```

Code Block 19: Identifying Outliers in Polarity Scores

```
    # Code Block 19: Identifying Outliers in Polarity Scores
    # Identifying outliers
    lower_bound = quartiles[0.25] - (1.5 * IQR) # Cal Lower bound
    upper_bound = quartiles[0.75] + (1.5 * IQR) # Cal Lower bound
    print(f"Outliers are values below {lower_bound} and above {upper_bound}.")

    # Identify extreme sentiment scores
    outliers = data[(data['polarity'] < lower_bound) | (data['polarity'] > upper_boun

    # Plotting the original distribution with outliers highlighted
    plt.figure(figsize=(6, 4))
    sns.histplot(data['polarity'], kde=True, color="purple",
                 label="Polarity Scores", bins=30) #  create a KDE
    plt.scatter(outliers['polarity'], np.zeros_like(outliers['polarity']) - 0.01,
                color='red', s=50, label='Outliers')
    plt.title('Distribution of Polarity Scores with Outliers') # Title
    plt.xlabel('Polarity') # x -axis name
    plt.ylabel('Frequency') # y -axis name
    plt.legend() # Legend
    plt.text(0.5, -0.2, '\nFig.12 -Dist. & Spread of Sentiment in the Dataset\n:'
             '\nMean Polarity Scores',
             transform=plt.gca().transAxes, ha='center', va='top')
    plt.show() # Show Histogram and KDE Line
```

Code Block 20: Mean Polarity by Sentiment Category

```
# Code Block 20: Mean Polarity by Sentiment Category

category_means = data.groupby('sentiment')['polarity'].mean() # Gp & cal Mean
print("\nMean Polarity by Sentiment Category:\n", category_means)

plt.figure(figsize=(5, 3)) # size 5in by 3in
category_means.plot(kind='bar', color=['lightblue', 'green', 'pink']) # Colour
plt.title('Mean Polarity by Sentiment Category') # Title
plt.xlabel('Sentiment Category') # X -axis
plt.ylabel('Mean Polarity') # y - axis
plt.xticks(rotation=45)
plt.text(0.5, -0.3, '\n\nFig.13 -Mean Polarity Scores\n:'
         'Distribution and Spread of Sentiment in the Dataset',
         transform=plt.gca().transAxes, ha='center', va='top')
plt.show() # Bar Chart
```

Code Block 21: Median Polarity by Sentiment Category

```
[ ] # Code Block 21: Median Polarity by Sentiment Category
category_medians = data.groupby('sentiment')['polarity'].median() # Gp & cal Medi
print("\nMedian Polarity by Sentiment Category:\n", category_medians)

plt.figure(figsize=(5, 3)) #Size 5in by 3 in
category_medians.plot(kind='bar', color=['red', 'green', 'blue']) #Colour
plt.title('Median Polarity by Sentiment Category') # Title
plt.xlabel('Sentiment Category') # X-axis
plt.ylabel('Median Polarity') # y -axis
plt.xticks(rotation=45)
plt.text(0.5, -0.3, '\nFig.14 -Median Polarity by Sentiment Category:\n'
         'Distribution and Spread of Sentiment in the Dataset',
         transform=plt.gca().transAxes, ha='center', va='top')
plt.show() # Bar Chart
```

Code Block 22: Confidence Interval for Mean Polarity

```
# Code Block 22: Confidence Interval for Mean Polarity
mean_polarity = data['polarity'].mean() # Cal Mean Polarity
confidence_level = 0.95
degrees_freedom = data['polarity'].count() - 1 # Deg of freedom
std_err = stats.sem(data['polarity']) # Standard Error
confidence_interval = stats.t.interval(confidence_level, degrees_freedom,
                                       mean_polarity, std_err)
print(f"\n95% Confidence Interval for Mean Polarity: {confidence_interval}")

# Visualizing the mean polarity with confidence interval
plt.figure(figsize=(5,3)) # Size 5in by 3in
plt.bar('Mean Polarity', mean_polarity, color='skyblue',
        yerr=[[mean_polarity - confidence_interval[0]],
         [confidence_interval[1] - mean_polarity]], capsize=10)
plt.errorbar('Mean Polarity', mean_polarity,
             yerr=[[mean_polarity - confidence_interval[0]],
              [confidence_interval[1] - mean_polarity]],
             fmt='o', color='darkblue', capsize=10)
plt.ylabel('Polarity Score') # X-axis
plt.title('Mean Polarity with 95% Confidence Interval') # Title
plt.tight_layout()
plt.text(0.5, -0.2, '\nFig.15 -Mean Polarity Score with 95% C.Interval:\n:'
         '\nMean polarity score and the level of confidence in the estimate',
```

Code Block 23: Correlation between Polarity and Review Length

```
[ ]  # Code Block 23: Correlation between Polarity and Review Length
     data['review_length'] = data['clean_review_text'].apply(len) # Cal len store
     print("\nCorrelation between Polarity and Review Length:\n",
           data[['polarity', 'review_length']].corr())

     # Plotting scatter plot with regression line
     plt.figure(figsize=(6, 4)) # Size 6in by 4in
     sns.regplot(x='review_length', y='polarity', data=data,
                 scatter_kws={'alpha':0.3}, line_kws={'color':'red'})
     plt.title('Review Length(RL) vs. Polarity(SP)') # Title
     plt.xlabel('Review Length') # X-axis
     plt.ylabel('Polarity') # y-axis
     plt.text(0.5, -0.2, '\nFig.16 -Correlation between RL and SP:\n:'
              '\nCorrelation between Review Length and Sentiment Polarity',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Scatter plot with regression line
```

Code Block 24: Correlation between Polarity and Review Length

```
     # Code Block 24: Correlation between Polarity and Review Length
     # Printing DataFrame for Correlation
     correlation_stats = data[['review_length', 'polarity']].corr() # Cal RL vs SP
     print(correlation_stats)

     # Creating a DataFrame for visualization (if needed)
     df_for_visualization = pd.DataFrame({
         'Review Length': data['review_length'],
         'Polarity': data['polarity']
     })

     print(df_for_visualization.head())

     bins = np.linspace(0, 1, 11)  # 11 bins from 0 to 1
     polarity_hist, _ = np.histogram(data['polarity'], bins=bins) # CBuils Hist

     for i, count in enumerate(polarity_hist): # Count each bin
         print(f"{bins[i]:.2f} - {bins[i+1]:.2f}\t{count}")

     # Assuming 'review_length' and 'polarity' are already in your DataFrame
     plt.figure(figsize=(3, 5)) # Size 4in by 5in
     sns.jointplot(x='review_length', y='polarity', data=data, kind='hex',
                   gridsize=20, color='blue', space=0)
     plt.suptitle('Review Length vs. Polarity (Hexbin Plot)',
                  y=1.02) # Title
     plt.xlabel('Review Length') # x- axis
     plt.ylabel('Polarity') # y- axis
     plt.text(0.5, -0.1, '\nFig.17 -Density between RL and SP:\n:'
              '\nDensity & Correlation between Review Length and Sentiment Polarity',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show Hexbin plot
```

Code Block 26: Normalization of Polarity Scores

```
[ ]  # Code Block 26: Normalization of Polarity Scores
     # Applying a consistent normalization procedure for Scaling & standardization.
     data['normalized_polarity'] = data['polarity'].apply(lambda x: (x + 1) * 5)

     # Display the results
     print(data[[
         'clean_review_text',
         'sentiment',
         'polarity',
         'normalized_polarity'
     ]].head())
```

Code Block 27: Categorization of Sentiment Scores

```
[ ]  # Code Block 27: Categorization of Sentiment Scores
     # Create bins for the 0-10 scale, assuming you want to categorize to 10 lvls
     bins = np.linspace(0, 10, 11) # Increase View of Dist Scores

     # Create the 'sentiment_group' column based on the 'reviews.rating' column
     data['sentiment_group'] = pd.cut(data['reviews.rating'], bins,
                                      labels=np.arange(10), include_lowest=True)

     # Print the DataFrame with the 'sentiment_group' column
     print(data[['reviews.rating', 'sentiment_group']])

     print(data['reviews.rating'].value_counts().sort_index())

     # Create a bar plot of sentiment scores
     plt.figure(figsize=(6, 4))
     plt.bar(np.arange(1, 6), data['reviews.rating'].value_counts(),
             align='center')
     plt.xticks(np.arange(1, 6))
     plt.title('Distribution of Sentiment Scores')
     plt.xlabel('Sentiment Score')
     plt.ylabel('Frequency')
     plt.text(0.5, -0.1, '\nFig.18 -Distribution of Sentiment Scores:\n:'
              '\nFrequency of Sentiment Scores',
              transform=plt.gca().transAxes, ha='center', va='top')
     plt.show() # Show Bar Plot
```

Code Block 28: Calculating Similarity between Two Reviews

```
[ ]  # Code Block 28: Calculating Similarity between Two Reviews
     nlp = spacy.load('en_core_web_md')  # Make sure to download this model first

     # Function to calculate similarity between two texts
     def calculate_similarity(text1, text2):
         doc1 = nlp(text1)
         doc2 = nlp(text2)
         return doc1.similarity(doc2)

     try: # This is included "be cautious not to use an index that is out of bounds"
         # Attempt to access an element using an index
         my_review_of_choice1 = data['reviews.text'][1985]
         my_review_of_choice2 = data['reviews.text'][2985]
     except IndexError:
         # Handle the case where the index is out of bounds
         print(f"One of the indices is out of bounds for the dataset.")

     similarity_score = calculate_similarity(my_review_of_choice1, my_review_of_choice
     print(f"Review 1 length: {len(my_review_of_choice1)}")
     print(f"Review 2 length: {len(my_review_of_choice2)}")
     print(f"Similarity score: {similarity_score:.6f}")
```

Code Block 29: Sentiment Analysis and Similarity Score between Custom Reviews

```
[ ]  # Code Block 29: Sentiment Analysis & Similarity Score btw Custom Reviews Code
     # Function to calculate similarity between two texts
     def calculate_similarity(text1, text2):
         doc1 = nlp(text1)
         doc2 = nlp(text2)
         return doc1.similarity(doc2)

     # Example usage with two reviews
     review1 = ("I absolutely love the great features of the Amazon Kindle tablet.  "
                "It's a fantastic device that I use every day, and its lightweight "
                "design makes it perfect for easy portability.")
     review2 = ("I'm extremely disappointed with the small size of this app on the "
                "device. The book downloads don't work properly, and the tablet's "
                "outdated version causes numerous problems with updates, making it an
                "incredibly frustrating experience to use this tablet.")

     similarity_score = calculate_similarity(review1, review2)
     print(f"Review 1 length: {len(review1)}")
     print(f"Review 2 length: {len(review2)}")
     print(f"Similarity score: {similarity_score}")
```