Customer Sentiment Analysis

JSON file parsing

This step involves parsing the contents of an "amazonreviews.json" file, extracting specific fields from each review, and storing the extracted information in a structured format. The script uses the json module to load the JSON data from the file and pandas library to create a DataFrame for further processing.

The script starts by opening the "amazonreviews.json" file and loading its contents as JSON data. It then iterates over each review in the "reviews" list, extracts the desired fields such as review ID, title, body, rating, date, profile name, and verified purchase status, and stores them in a dictionary called "parsed_review". These dictionaries are appended to the "parsed_reviews" list.

After processing all the reviews, the script creates a DataFrame named "df" using the pandas DataFrame function, passing the "parsed_reviews" list as the argument. Finally, the DataFrame is exported to a CSV file named "parsed_reviews.csv" using the to_csv function, ensuring that the index is not included in the exported file.

This step provides a way to convert the JSON data into a structured CSV format, making it easier to analyze and work with the review data using tools like pandas or other data analysis libraries in Python.

In [ ]:
```
# Import the json module
import json

# Import the pandas library and alias it as pd
import pandas as pd
```

In [ ]:
```
# Open the 'amazonreviews.json' file and assign it to the variable 'file'
with open('amazonreviews.json') as file:
    # Load the contents of the file as JSON data and assign it to the variable '
    data = json.load(file)
```

```
"id": "R2OPC1I4QODMT7",
"title": "No smell",
"body": "Just started using so no effects yet but I prefer it to the first one I tried because there was no scent or smell and it dissolves easily",
"asin": "B07Q6YSGF9",
"body_html": "<span>Just started using so no effects yet but I prefer it to the first one I tried because there was no scent or smell and it dissolves easily</span>",
"link": "https://www.amazon.co.uk/gp/customer-reviews/R2OPC1I4QODMT7/ref=cm_cr_getr_d_rvw_ttl?ie=UTF8&ASIN=B07Q6YSGF9",
"rating": 4,
"date": {
  "raw": "Reviewed in the United Kingdom on 8 May 2023",
  "utc": "2023-05-08T00:00:00.000Z"
},
"profile": {
  "name": "Bern",
  "link": "https://www.amazon.co.uk/gp/profile/amzn1.account.AHM3DQZSFM35VO5ROZQAOH4TIC4A/ref=cm_cr_getr_d_gw_btm?ie=UTF8",
  "id": "AHM3DQZSFM35VO5ROZQAOH4TIC4A"
},
"vine_program": false,
"verified_purchase": true,
"review_country": "gb",
"is_global_review": false,
"position": 4,
"page": 2
},
```


In [ ]:
```python
# Assigning the value of 'reviews' from the 'data' dictionary to the variable 'r
reviews = data['reviews']
```

In [ ]:
```python
#  Initialize an empty list called "parsed_reviews"
parsed_reviews = []
```

In [ ]:
```python
# Iterate over each review in the 'reviews' list
for review in reviews:
    # Extract the 'id' from the current review and assign it to 'review_id'
    review_id = review['id']

    # Extract the 'title' from the current review and assign it to 'title'
    title = review['title']

    # Extract the 'body' from the current review and assign it to 'body'
    body = review['body']

    # Extract the 'rating' from the current review and assign it to 'rating'
    rating = review['rating']

    # Extract the 'raw' value from the 'date' field of the current review and as
    date = review['date']['raw']

    # Extract the 'name' from the 'profile' field of the current review and assi
    profile_name = review['profile']['name']

    # Extract the 'verified_purchase' value from the current review and assign i
    verified_purchase = review['verified_purchase']

    # Create a dictionary 'parsed_review' to store the extracted information
    parsed_review = {
        'Review ID': review_id,
        'Title': title,
        'Body': body,
        'Rating': rating,
        'Date': date,
        'Profile Name': profile_name,
        'Verified Purchase': verified_purchase
    }

    # Append the 'parsed_review' dictionary to the 'parsed_reviews' list
    parsed_reviews.append(parsed_review)
```

In [ ]:
```python
# Creating a data frame named 'df' using the pd.DataFrame() function
# The 'parsed_reviews' variable is passed as the argument to create the data fra
df = pd.DataFrame(parsed_reviews)
```

In [ ]:
```python
# This line exports the DataFrame 'df' to a CSV file named 'parsed_reviews.csv'
# The 'to_csv' function is used to convert the DataFrame to a CSV format

df.to_csv('parsed_reviews.csv', index=False)
# The 'parsed_reviews.csv' is the name of the CSV file that will be created or c
# The 'index=False' argument specifies that the DataFrame's index should not be
# This argument ensures that the CSV file only contains the data from the DataFr
```

Review ID	Title	Body	Rating	Date	Profile Name	Verified Purchase
R3OR722EUPAF60	Great value collagen	I have been using hydrolysed collagen peptide products for general joint and skin maintenance, rather than to treat any specific problem. Hence I haven't given a pain relief rating. There are a large number of brands currently on the market of which I've tried more than half a dozen now. Maybe a number of them are actually sourced from the same production lines and only differ in their packaging and branding. There doesn't seem to be a vast degree of difference in the products or their amino acid profiles. The price certainly varies considerably but the consumer experience rather less so. The big brand American imports are expensive and not noticeably better than UK or European sourced collagen products. I guess you pay a premium for their research and the importation shipping costs. Although all brands make a point about being unflavoured or having a 'neutral' taste, its fair to say that they all do have that characteristic slightly stale bovine collagen taste to at least some degree. The very nature of the product presumably makes that unavoidable. The Wellgard is 'OK' in that respect - its not my absolute favourite tasting brand, but certainly tastes no worse than most others I've tried, and seems to dissolve just as easily. In most flavoured drinks, such as coffee for instance, you really don't notice the collagen taste anyway. In terms of clinical efficacy, it seems unlikely that there would be a great deal of difference between the competing brands, so whether or not Wellgard's product actually works any better because of its slightly different amino profile is anyone's guess. However the obvious advantage lies in that it is a well packaged, well presented product coming from the UK, and offers much better value for money than comparable brands imported from overseas.	5	Reviewed in the United Kingdom 🇬🇧 on 4 October 2019	Dr. Aivar Bracka	TRUE
R2ZLQXMUT3LQ34	So far so good	I wasn’t sure what to expect with this product but I’ve read so many different things about how collagen can help when going through the menopause so I gave it a shot. I chose this one because of the great reviews it had. I followed the dose recommendations and added it to my morning coffee. I could not taste a thing! I was so pleased that it was going to be easy to take and not leave a horrible taste. I’ve only been taking it for a week or so, so can’t say I’ve seen any changes yet as it does state it can take around 6 weeks to notice any difference. Fingers crossed it does what it suggests 🤞🏼 So if you want something that’s not overly expensive, don’t taste vile or leave an after taste then give this one a go.	5	Reviewed in the United Kingdom 🇬🇧 on 14 May 2023	Mrs J D Royer	TRUE
R2EM0FLB5NZXBU	Excellent product	I think that this product is so good that I have it on repeat-supply. It does everything that it says on the tin (sorry). Mixing with other juices, and coffee, in preparation for consumption is a breeze. I have to say that I failed to make a satisfactory preparation with red wine... Compared to one other product which had a significant collagen content this is much better, no clumps and therefore no frustration. Those who can remember mixing custard from powder will understand the technique. As far as health benefits are concerned, I was gifted a high-collagen mango-flavoured product by a visitor. By the time that product was used-up I noticed some improvements in my health and therefore tried to buy it again ... not available in the UK it seems. So I did some reading and found this Wellgard product. Recommended!	5	Reviewed in the United Kingdom 🇬🇧 on 10 May 2023	Stephen	TRUE
RSUXEK5HSJMQP	Quality and value	I thought a long time before purchasing but this brand has made it more affordable so I’m having an go and I hope to use long term now at the price. I have been successful adding it to my morning OJ or porridge. I am only 2 weeks in so no noticeable benefit as yet but early days. Seems to be a great product. I did try another brand once but the smell and taste I couldn’t cope with. I don’t have that problem with this brand.	5	Reviewed in the United Kingdom 🇬🇧 on 12 May 2023	Mrs Lorraine Hayward	TRUE
R1OBHF9T92A7CY	Amazing product	Has delivered exactly what it says it would ! Easily dissolved , I have mine in water .. completely tasteless as promised . It’s only been a week and I am noticing a difference already. Really pleased with my purchase , great value compared to other options	5	Reviewed in the United Kingdom 🇬🇧 on 21 May 2023	Mrs M L Hastings	TRUE
R3T658OIY0UBKG	So far good	Only been taking it for a week so hard to comment on Benefits, ask me again in 12 weeks . But as a product I put it in my mid morning coffee each morning it will dissolve just add each spoon separately and give it a few minutes. Once dissolved I can not taste anything. Much better than taking the tablets as they were so big and hard to swallow .	4	Reviewed in the United Kingdom 🇬🇧 on 29 April 2023	Amazon Customer	TRUE
RZ4CWFG3F85AS	Collagen powder	Mixes well in juice, no taste. I have only been taking for a week so can’t say if it is making a difference yet I mix it in my 1000ml bottle and drink throughout the day. Easy to add I my diet	4	Reviewed in the United Kingdom 🇬🇧 on 6 May 2023	Leanne	TRUE
R1TSH5L8YOSOLB	It worked for me 2022	I have been using this product religiously for over 3/4 years now, the first time I tried it, I noticed the weird smell but that doesn’t last, I mix it with warm water, vitamin c powder and drink it first thing in the morning. On busy days i do the same mixture in my water bottle and drink it all day. My nails grow soo fast when I use this product, that am forced to get my nails done more often than necessary. My hair is fuller and very difficult to manage now though it’s hasn’t grown longer, it’s simply heavy. I have seen no change to my skin ever since I started using this product. I however hate the smell it’s leaves in my water bottle at the end of the day. Am still using my pack from 2022, not sure if the formula has been changed in 2023.	5	Reviewed in the United Kingdom 🇬🇧 on 15 April 2023	Amazon Customer	TRUE

Amazon Customer Reviews Sentiment Analysis

This project involves analyzing and visualizing sentiment in a dataset of reviews. The project's steps:

1. Importing the necessary libraries:
    - `numpy` (imported as `np`) for numerical operations
    - `matplotlib.pyplot` (imported as `plt`) for data visualization
    - `WordCloud` from the `wordcloud` library for generating word clouds
    - `stopwords` from the NLTK library for handling stopwords (commonly used words)
    - `TextBlob` from the `textblob` library for sentiment analysis
2. Loading the dataset:
    - The project assumes that there is a file named 'parsed_reviews.csv' containing the parsed reviews data.
    - The CSV file is read using `pd.read_csv` function from the pandas library, and the data is stored in a DataFrame called `df`.
3. Performing sentiment analysis on review titles and bodies:
    - Sentiment polarity is calculated for both the review titles and bodies using the `TextBlob` library.
    - The sentiment polarity values are stored in new columns named 'Title Sentiment' and 'Body Sentiment' in the DataFrame `df`.

- The sentiment polarity is a numerical value indicating the sentiment of the text, where positive values indicate positive sentiment and negative values indicate negative sentiment.

4. Classifying review titles and bodies into positive and negative categories:

   - Based on the sentiment polarity values, the review titles and bodies are categorized as either 'Positive' or 'Negative'.
   - The sentiment categories are stored in new columns named 'Title Sentiment Category' and 'Body Sentiment Category' in the DataFrame `df`.

5. Visualizing the sentiment distribution of review titles:

   - The number of positive and negative review titles is counted using `value_counts()` function on the 'Title Sentiment Category' column.
   - The distribution of sentiment categories is visualized using a bar plot and a pie chart.

6. Visualizing the sentiment distribution of review bodies:

   - Similar to step 5, the number of positive and negative review bodies is counted and visualized using a bar plot and a pie chart.

7. Counting frequent words in positive and negative review titles:

   - Positive and negative review titles are combined into single strings.
   - The frequency of words in the review titles is counted, and common stopwords (e.g., "the," "and," "is") are removed.
   - The word counts are stored in separate `pd.Series` objects named `positive_title_word_counts` and `negative_title_word_counts`.

8. Generating word clouds for positive and negative review titles:

   - Word clouds are created using the `WordCloud` library, representing the most frequent words in the positive and negative review titles.

9. Counting frequent words in positive and negative review bodies:

   - Similar to step 7, the frequency of words in positive and negative review bodies is counted, and stopwords are removed.
   - The word counts are stored in separate `pd.Series` objects named `positive_body_word_counts` and `negative_body_word_counts`.

10. Generating word clouds for positive and negative review bodies:

    - Word clouds are created using the `WordCloud` library, representing the most frequent words in the positive and negative review bodies.

11. Plotting frequent words in positive and negative review titles:

    - Bar plots are created to visualize the most frequent words in positive and negative review titles.
    - The top 10 words with their respective frequencies are displayed, and the bars are colored green for positive words and red for negative words.

12. Plotting frequent words in positive and negative review bodies:

- Similar to step 11, bar plots are created to visualize the most frequent words in positive and negative review bodies, because their content was past the maximum allowed length.

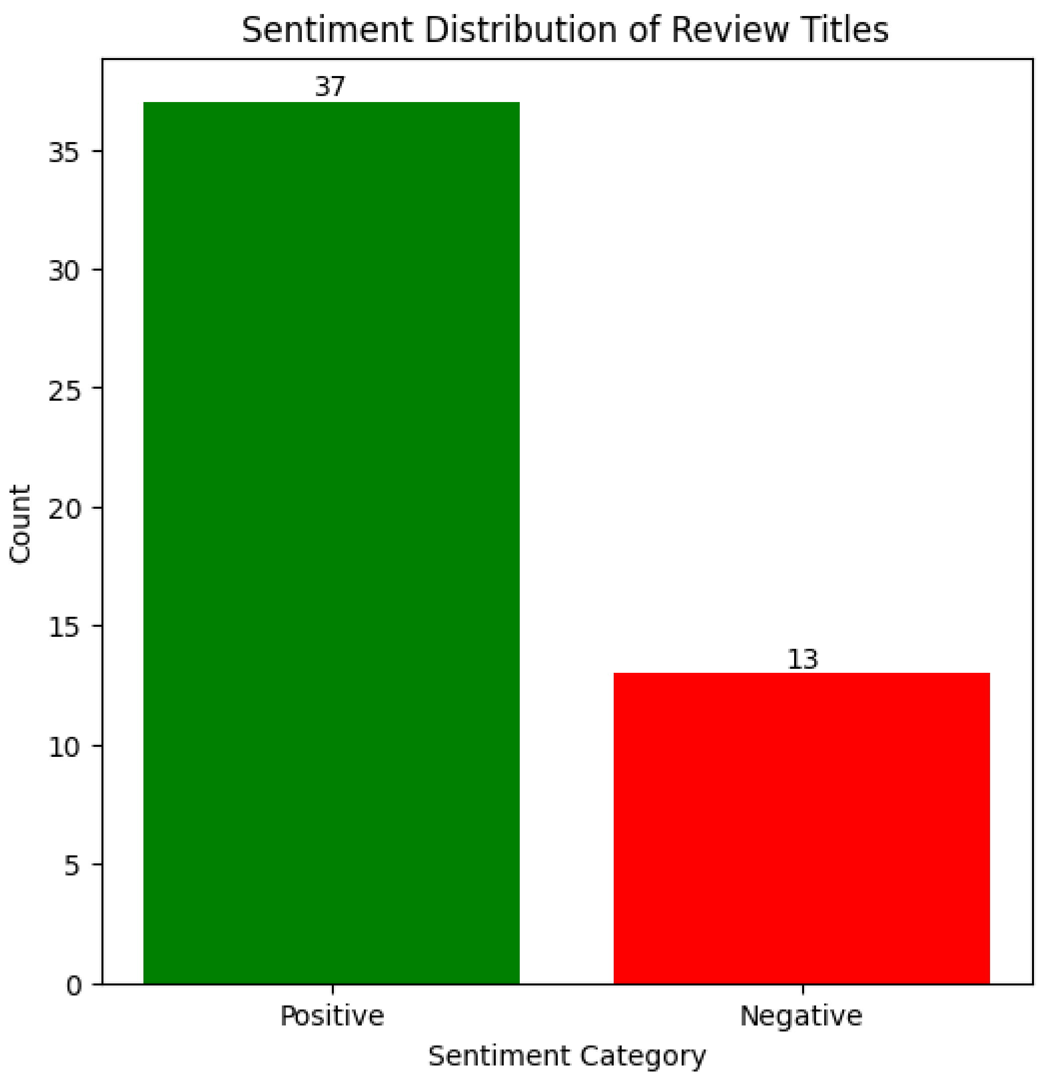
In [ ]:
```
import numpy as np  # Importing the NumPy library
import matplotlib.pyplot as plt  # Importing the Matplotlib library for visualiz
from wordcloud import WordCloud  # Importing WordCloud for generating word cloud
from nltk.corpus import stopwords  # Importing NLTK's stopwords corpus
from textblob import TextBlob  # Importing TextBlob for sentiment analysis
```

In [ ]:
```
# Load the parsed_reviews.csv file
df = pd.read_csv("parsed_reviews.csv")  # Reading the CSV file into a pandas Dat
```

In [ ]:
```
# Perform sentiment analysis on review titles and bodies
df['Title Sentiment'] = df['Title'].apply(lambda x: TextBlob(str(x)).sentiment.p
df['Body Sentiment'] = df['Body'].apply(lambda x: TextBlob(str(x)).sentiment.pol
```

In [ ]:
```
# Classify review titles and bodies into positive and negative categories
df['Title Sentiment Category'] = np.where(df['Title Sentiment'] > 0, 'Positive',
df['Body Sentiment Category'] = np.where(df['Body Sentiment'] > 0, 'Positive', '
```

In [ ]:
```
# Visualize the sentiment distribution of review titles
title_sentiment_counts = df['Title Sentiment Category'].value_counts()  # Counti
plt.figure(figsize=(6, 6))  # Creating a figure with a specific size
plt.bar(title_sentiment_counts.index, title_sentiment_counts.values, color=['gre
plt.title('Sentiment Distribution of Review Titles')  # Adding a title to the pl
plt.xlabel('Sentiment Category')  # Adding label to the x-axis
plt.ylabel('Count')  # Adding label to the y-axis
for i, count in enumerate(title_sentiment_counts.values):  # Adding text labels
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.show()  # Displaying the plot
```



In [ ]:

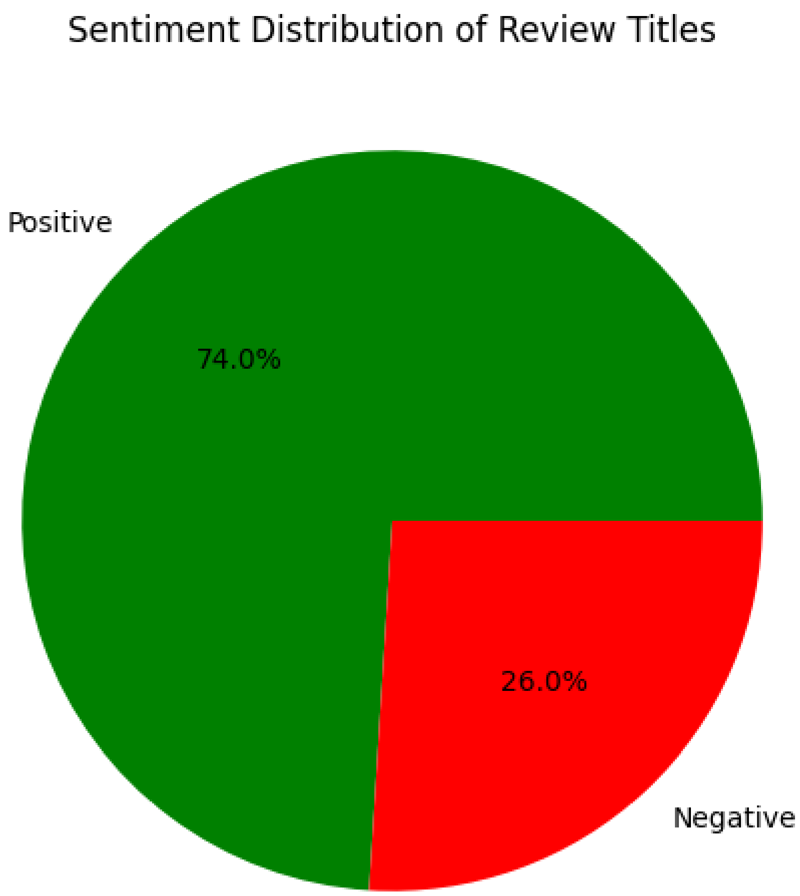
```
title_sentiment_counts
```

Out[ ]:

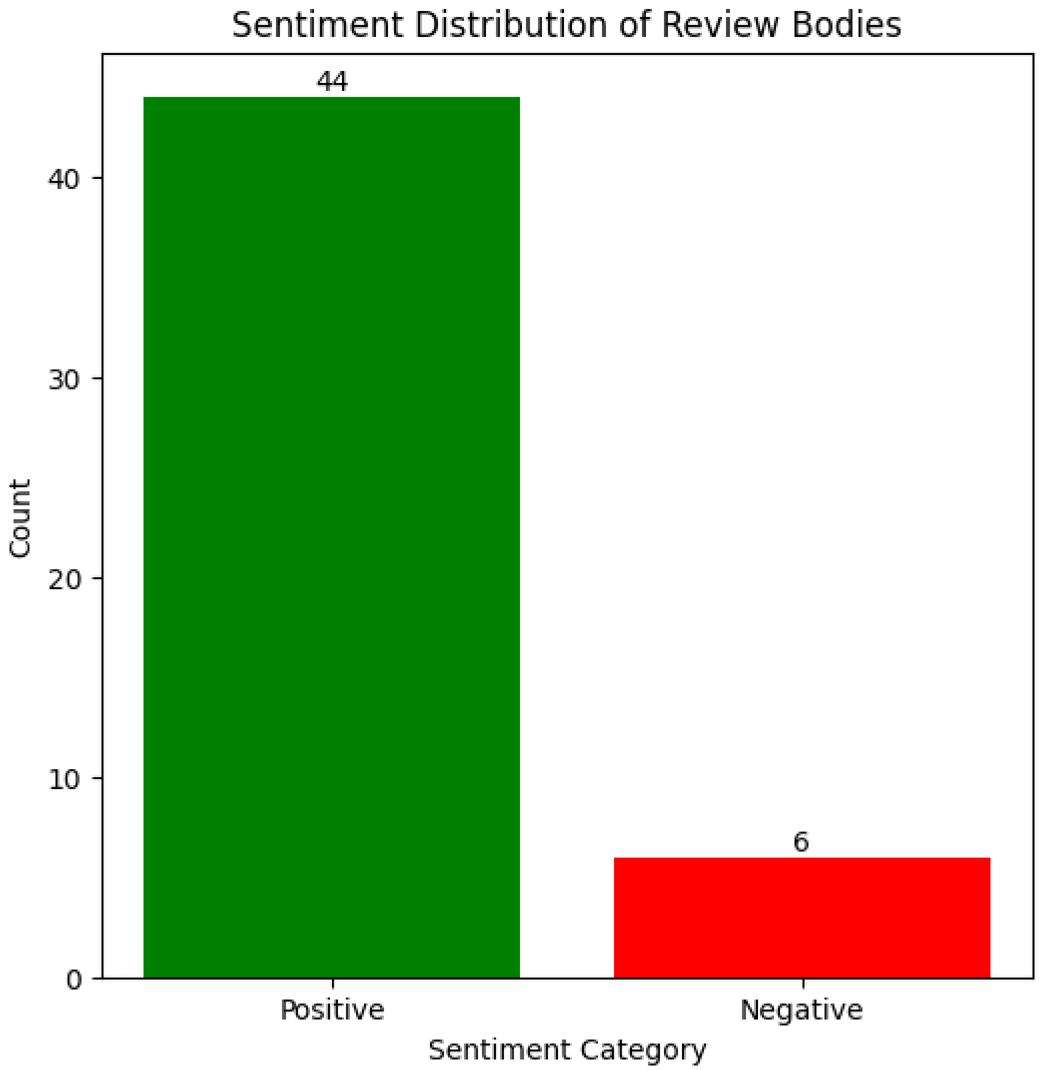
```
Positive    37
Negative    13
Name: Title Sentiment Category, dtype: int64
```

In [ ]:

```
# Visualize the sentiment distribution of review titles
title_sentiment_counts = df['Title Sentiment Category'].value_counts()  # Counti
plt.figure(figsize=(6, 6))  # Creating a figure with a specific size
colors = ['green', 'red']  # Colors for the pie chart
plt.pie(title_sentiment_counts.values, labels=title_sentiment_counts.index, colo
plt.title('Sentiment Distribution of Review Titles')  # Adding a title to the pl
plt.show()  # Displaying the plot
```



In [ ]:
```
# Visualize the sentiment distribution of review bodies
body_sentiment_counts = df['Body Sentiment Category'].value_counts()  # Counting
plt.figure(figsize=(6, 6))  # Creating a figure with a specific size
plt.bar(body_sentiment_counts.index, body_sentiment_counts.values, color=['green
plt.title('Sentiment Distribution of Review Bodies')  # Adding a title to the pl
plt.xlabel('Sentiment Category')  # Adding label to the x-axis
plt.ylabel('Count')  # Adding label to the y-axis
for i, count in enumerate(body_sentiment_counts.values):  # Adding text labels t
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.show()  # Displaying the plot
```



In [ ]:

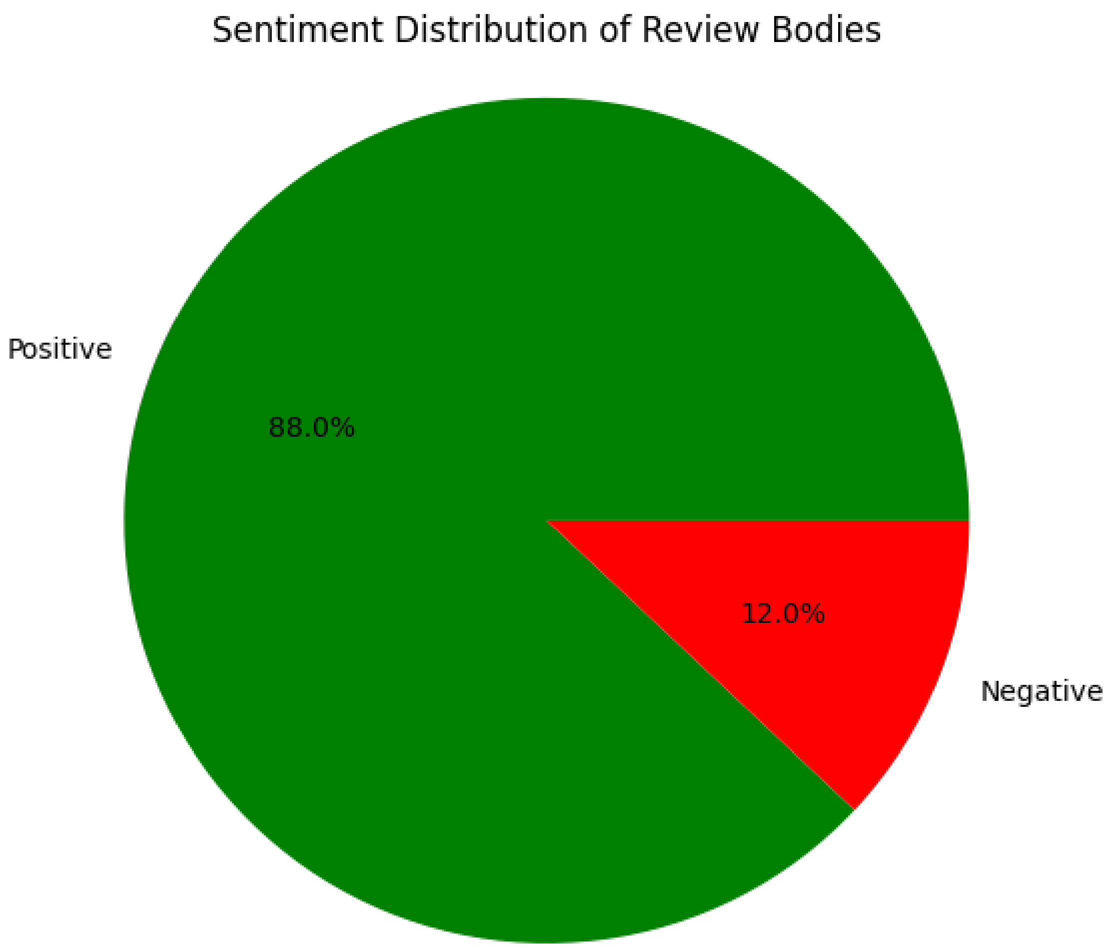
```
body_sentiment_counts
```

Out[ ]:

```
Positive    44
Negative     6
Name: Body Sentiment Category, dtype: int64
```

In [ ]:

```
# Visualize the sentiment distribution of review bodies
body_sentiment_counts = df['Body Sentiment Category'].value_counts()  # Counting

# Create a pie chart for sentiment distribution of review bodies
plt.figure(figsize=(6, 6))  # Creating a figure with a square size
colors = ['green', 'red']  # Colors for the pie chart
plt.pie(body_sentiment_counts.values, labels=body_sentiment_counts.index, colors
plt.title('Sentiment Distribution of Review Bodies')  # Adding a title to the pl
plt.axis('equal')  # Ensuring a circular pie chart

plt.show()  # Displaying the plot
```



The result provided is the count of positive and negative review sentiments for both the review titles and review bodies. Here's the breakdown of the result:

For the review titles:

- Positive: There are 37 reviews with positive sentiment in their titles.
- Negative: There are 13 reviews with negative sentiment in their titles.

For the review bodies:

- Positive: There are 44 reviews with positive sentiment in their bodies.
- Negative: There are 6 reviews with negative sentiment in their bodies.

The counts indicate the number of reviews falling into each sentiment category based on the analysis performed. It suggests that a majority of the review titles and bodies have positive sentiment, with a smaller number of reviews expressing negative sentiment.

There can be several reasons why the counts of positive and negative reviews differ between review titles and review bodies. Here are some possible explanations:

1. **Different emphasis:** Review titles are often concise and provide a brief summary or highlight of the main sentiment or experience expressed in the review body. As a result, reviewers may prioritize expressing their overall sentiment or opinion in the title, leading to a stronger alignment between the title sentiment and the overall sentiment of the review. In contrast, the review body may contain more detailed information, experiences, or discussions, which can contribute to a more nuanced sentiment analysis and potentially result in a different distribution of positive and negative sentiments.

2. **Subjectivity and tone:** Review titles tend to be more subjective and emotional compared to review bodies. Reviewers often use titles to capture their overall sentiment or impression succinctly. This tendency towards subjectivity and emotional expression in titles may lead to a higher concentration of extreme positive or negative sentiments. On the other hand, the review bodies may have a more balanced tone and contain more neutral or nuanced sentiments, resulting in a different distribution.

3. **Length and context:** Review titles are typically shorter than review bodies and may not provide enough context or details to accurately capture the sentiment of the entire review. Due to the limited length, reviewers may focus on expressing extreme sentiments in the title, while the body allows for more elaboration and discussion, potentially leading to a broader range of sentiments and a different distribution.

4. **User behavior and bias:** Reviewers may have different motivations or intentions when writing review titles compared to the review bodies. They may consciously or unconsciously prioritize certain aspects or sentiments in the title, leading to a biased representation of the overall sentiment. Additionally, reviewers may put more effort into crafting the title to attract attention or convey a strong opinion, while the body may be less polished or less focused on sentiment expression.

It's important to consider these factors and the context of the specific dataset when analyzing and interpreting the differences in sentiment distribution between review titles and review bodies.

In [ ]:
```
# Count frequent words in positive and negative review titles
positive_title_words = ' '.join(df[df['Title Sentiment Category'] == 'Positive']
negative_title_words = ' '.join(df[df['Title Sentiment Category'] == 'Negative']
stopwords = set(stopwords.words('english'))  # Creating a set of English stopwor
positive_title_word_counts = pd.Series(positive_title_words.split()).value_count
positive_title_word_counts = positive_title_word_counts.drop(labels=stopwords.ir
negative_title_word_counts = pd.Series(negative_title_words.split()).value_count
negative_title_word_counts = negative_title_word_counts.drop(labels=stopwords.ir
```

In [ ]:
```
# Generate word clouds for positive and negative review titles
positive_title_wordcloud = WordCloud(width=800, height=400, background_color='wh
negative_title_wordcloud = WordCloud(width=800, height=400, background_color='wh
```

In [ ]:
```
# Count frequent words in positive and negative review bodies
positive_body_words = ' '.join(df[df['Body Sentiment Category'] == 'Positive']['
```

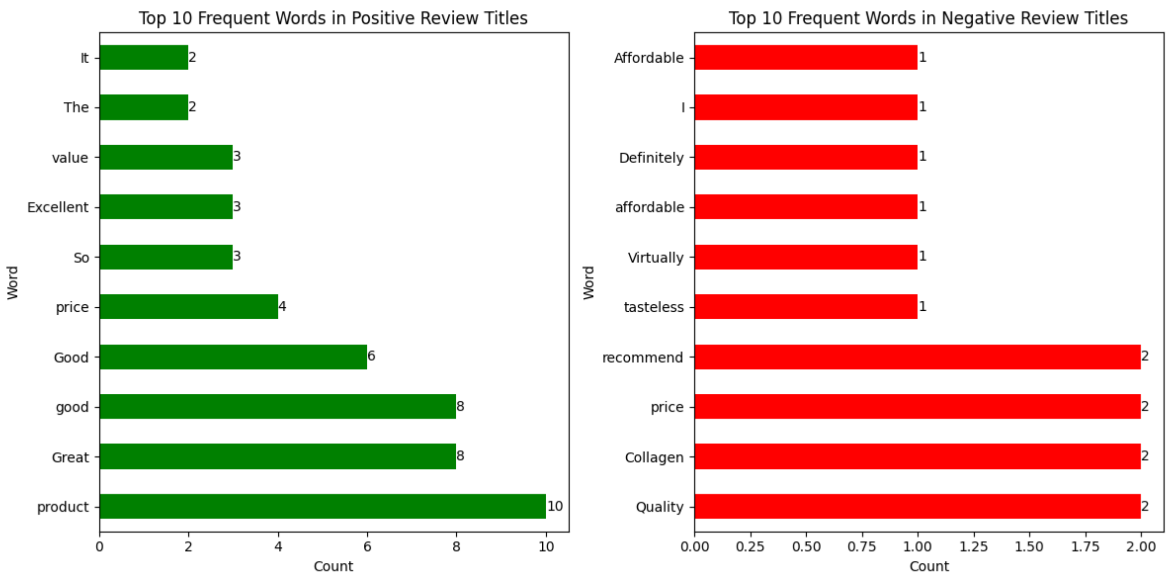
```
negative_body_words = ' '.join(df[df['Body Sentiment Category'] == 'Negative'][''
positive_body_word_counts = pd.Series(positive_body_words.split()).value_counts(
positive_body_word_counts = positive_body_word_counts.drop(labels=stopwords.inte
negative_body_word_counts = pd.Series(negative_body_words.split()).value_counts(
negative_body_word_counts = negative_body_word_counts.drop(labels=stopwords.inte
```

In [ ]:

```
# Generate word clouds for positive and negative review bodies
positive_body_wordcloud = WordCloud(width=800, height=400, background_color='whi
negative_body_wordcloud = WordCloud(width=800, height=400, background_color='whi
```

In [ ]:

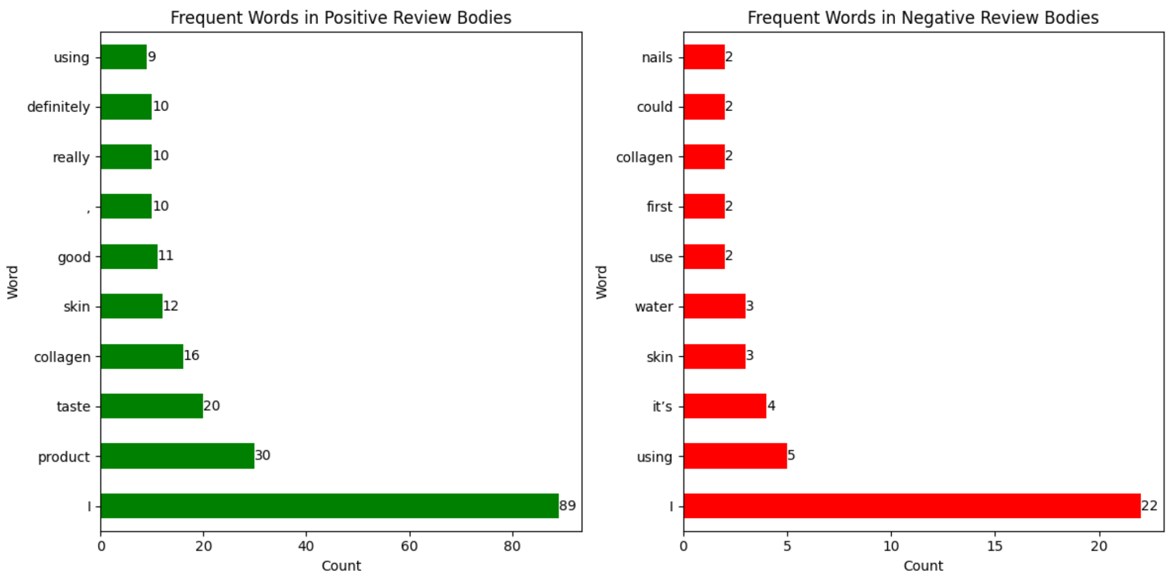
```
# Plot the top 10 frequent words in positive and negative review titles
plt.figure(figsize=(12, 6))  # Creating a figure with a specific size
plt.subplot(1, 2, 1)  # Creating subplots
top_positive_words = positive_title_word_counts.nlargest(10)  # Selecting the to
top_positive_words.plot(kind='barh', color='green')  # Creating a horizontal bar
plt.title('Top 10 Frequent Words in Positive Review Titles')  # Adding a title t
plt.xlabel('Count')  # Adding label to the x-axis
plt.ylabel('Word')  # Adding label to the y-axis
for i, count in enumerate(top_positive_words.values):  # Adding text labels to t
    plt.text(count, i, str(count), ha='left', va='center')
plt.subplot(1, 2, 2)  # Creating subplots
top_negative_words = negative_title_word_counts.nlargest(10)  # Selecting the to
top_negative_words.plot(kind='barh', color='red')  # Creating a horizontal bar p
plt.title('Top 10 Frequent Words in Negative Review Titles')  # Adding a title t
plt.xlabel('Count')  # Adding label to the x-axis
plt.ylabel('Word')  # Adding label to the y-axis
for i, count in enumerate(top_negative_words.values):  # Adding text labels to t
    plt.text(count, i, str(count), ha='left', va='center')
plt.tight_layout()  # Adjusting the spacing between subplots
plt.show()  # Displaying the plot
```



In [ ]:

```
# Plot the frequent words in positive and negative review bodies
plt.figure(figsize=(12, 6))  # Creating a figure with a specific size
plt.subplot(1, 2, 1)  # Creating subplots
positive_body_word_counts[:10].plot(kind='barh', color='green')  # Creating a ho
plt.title('Frequent Words in Positive Review Bodies')  # Adding a title to the p
plt.xlabel('Count')  # Adding label to the x-axis
plt.ylabel('Word')  # Adding label to the y-axis
for i, count in enumerate(positive_body_word_counts[:10].values):  # Adding text
    plt.text(count, i, str(count), ha='left', va='center')
plt.subplot(1, 2, 2)  # Creating subplots
```

```
negative_body_word_counts[:10].plot(kind='barh', color='red')  # Creating a hori
plt.title('Frequent Words in Negative Review Bodies')  # Adding a title to the p
plt.xlabel('Count')  # Adding label to the x-axis
plt.ylabel('Word')  # Adding label to the y-axis
for i, count in enumerate(negative_body_word_counts[:10].values):  # Adding text
    plt.text(count, i, str(count), ha='left', va='center')
plt.tight_layout()  # Adjusting the spacing between subplots
plt.show()  # Displaying the plot
```



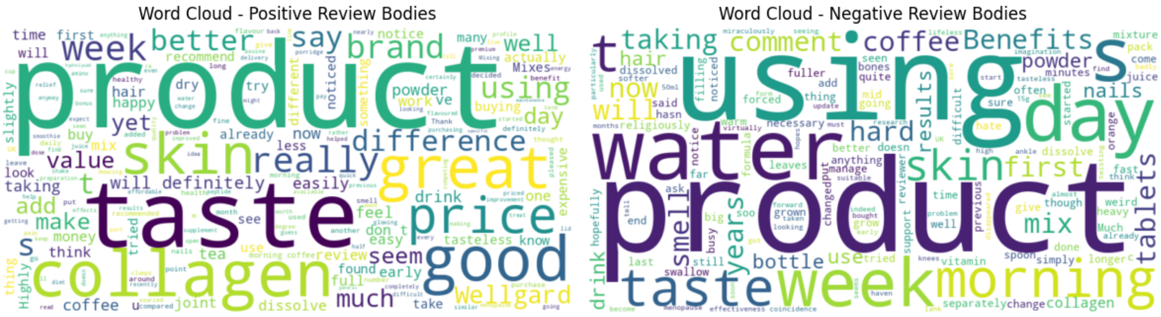
In [ ]:

```
# Display the word clouds for positive and negative review titles
plt.figure(figsize=(12, 6))  # Creating a figure with a specific size
plt.subplot(1, 2, 1)  # Creating subplots
plt.imshow(positive_title_wordcloud, interpolation='bilinear')  # Displaying the
plt.title('Word Cloud - Positive Review Titles')  # Adding a title to the plot
plt.axis('off')  # Removing the axis labels
plt.subplot(1, 2, 2)  # Creating subplots
plt.imshow(negative_title_wordcloud, interpolation='bilinear')  # Displaying the
plt.title('Word Cloud - Negative Review Titles')  # Adding a title to the plot
plt.axis('off')  # Removing the axis labels
plt.tight_layout()  # Adjusting the spacing between subplots
plt.show()  # Displaying the plot
```

In [ ]:

```
# Display the word clouds for positive and negative review bodies
plt.figure(figsize=(12, 6))  # Creating a figure with a specific size
plt.subplot(1, 2, 1)  # Creating subplots
plt.imshow(positive_body_wordcloud, interpolation='bilinear')  # Displaying the
plt.title('Word Cloud - Positive Review Bodies')  # Adding a title to the plot
plt.axis('off')  # Removing the axis labels
plt.subplot(1, 2, 2)  # Creating subplots
plt.imshow(negative_body_wordcloud, interpolation='bilinear')  # Displaying the
plt.title('Word Cloud - Negative Review Bodies')  # Adding a title to the plot
plt.axis('off')  # Removing the axis labels
```

```
plt.tight_layout()  # Adjusting the spacing between subplots
plt.show()  # Displaying the plot
```



Aspect-Based Sentiment Analysis

This step involves analyzing the sentiment of different aspects in a dataset of reviews and determining their importance scores.

1. Initialize two empty dictionaries, `positive_aspect_importance` and `negative_aspect_importance`, to store the importance scores of positive and negative aspects, respectively.

2. Iterate over a list of aspects.

3. Perform sentiment analysis for each aspect:

   - The sentiment polarity of the review bodies is calculated using the `TextBlob` library. The sentiment polarity represents the sentiment of the text, with positive values indicating positive sentiment and negative values indicating negative sentiment.
   - Two new columns are added to the DataFrame `df` for each aspect: one for the sentiment polarity (`aspect + ' Sentiment'`) and one for the sentiment category (`aspect + ' Sentiment Category'`). The sentiment category is determined based on whether the sentiment polarity is greater than 0 (positive) or not (negative).
4. Calculate the average sentiment score for each aspect:

   - The average sentiment score for positive reviews is calculated by taking the mean of the sentiment polarities of the reviews categorized as 'Positive' in the corresponding aspect's sentiment category column.
   - The average sentiment score for negative reviews is calculated in a similar manner for the reviews categorized as 'Negative' in the corresponding aspect's sentiment category column.
   - The importance scores for each aspect are stored in the `positive_aspect_importance` and `negative_aspect_importance` dictionaries, with the aspect name as the key and the average sentiment score as the value.
5. Sort the dictionaries by importance score:

- The `sorted` function is used to sort the `positive_aspect_importance` and `negative_aspect_importance` dictionaries in descending order based on the importance scores.
- The sorted results are assigned to `sorted_positive_aspect_importance` and `sorted_negative_aspect_importance` variables, respectively.

6. Create a figure for visualization:

    - A figure with a size of 12 inches by 6 inches is created using `plt.figure(figsize=(12, 6))`.

7. Create a bar plot for positive reviews:

    - A subplot is created with a position of 1 row, 2 columns, and the first position.
    - A bar plot is created using the data from `sorted_positive_aspect_importance`, where the x-axis represents the aspect names, and the y-axis represents the importance scores.
    - The title, x-label, y-label, and x-axis tick labels are set for the plot.

8. Create a bar plot for negative reviews:

    - A subplot is created with a position of 1 row, 2 columns, and the second position.
    - A bar plot is created using the data from `sorted_negative_aspect_importance`, following the same steps as in the previous plot.

9. Adjust the layout and display the plot:

    - The `tight_layout` function is called to automatically adjust the subplot parameters for better visualization.
    - The `show` function is called to display the plot.

The importance scores indicate the average sentiment polarity for each aspect in positive and negative reviews.

In [ ]:
```
positive_aspect_importance = {}
negative_aspect_importance = {}
```

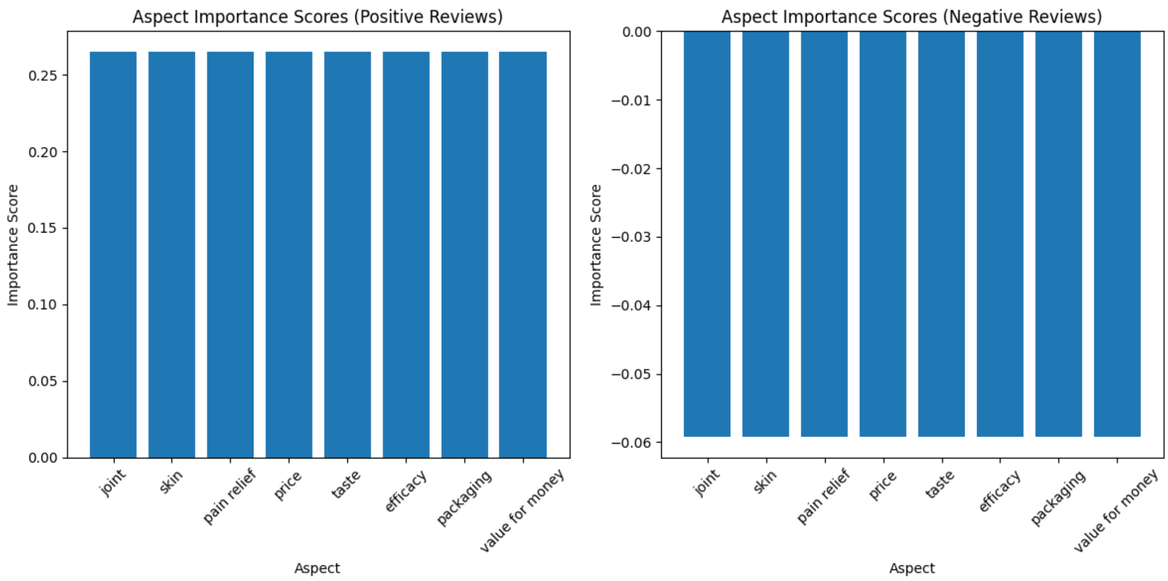
In [ ]:
```
aspects = ['joint', 'skin', 'pain relief', 'price', 'taste', 'efficacy', 'packag
```

In [ ]:
```
for aspect in aspects:
    # Perform sentiment analysis for the current aspect
    df[aspect + ' Sentiment'] = df['Body'].apply(lambda x: TextBlob(str(x)).sent
    df[aspect + ' Sentiment Category'] = df[aspect + ' Sentiment'].apply(lambda 

    # Calculate the average sentiment score for the current aspect
    positive_aspect_importance[aspect] = df[df[aspect + ' Sentiment Category'] =
    negative_aspect_importance[aspect] = df[df[aspect + ' Sentiment Category'] =
```

In [ ]:
```
sorted_positive_aspect_importance = sorted(positive_aspect_importance.items(), k
sorted_negative_aspect_importance = sorted(negative_aspect_importance.items(), k
```

In [ ]:
```
plt.figure(figsize=(12, 6))
```

```python
# Bar plot for positive reviews
plt.subplot(1, 2, 1)
plt.bar(*zip(*sorted_positive_aspect_importance))
plt.title('Aspect Importance Scores (Positive Reviews)')
plt.xlabel('Aspect')
plt.ylabel('Importance Score')
plt.xticks(rotation=45)

# Bar plot for negative reviews
plt.subplot(1, 2, 2)
plt.bar(*zip(*sorted_negative_aspect_importance))
plt.title('Aspect Importance Scores (Negative Reviews)')
plt.xlabel('Aspect')
plt.ylabel('Importance Score')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



Named Entity Recognition (NER)

This step involves analyzing customer reviews from an Amazon dataset and extracting the named entities within the reviews. The named entities are then counted, and the top 20 most frequent entities are visualized using a bar chart.

1. Import the necessary libraries:
    - `pandas` is imported as `pd` to work with the dataset.
    - `spacy` is imported for performing named entity recognition (NER).
    - `matplotlib.pyplot` is imported as `plt` for data visualization.
    - `Counter` is imported from the `collections` module to count the occurrences of entities.
2. Load the pre-trained NER model:
    - The `en_core_web_sm` model from the spaCy library is loaded using `spacy.load()` and assigned to the `nlp` variable.
    - This model is trained on English text and capable of recognizing named entities.

3. Read the CSV file:

    - The customer reviews data is read from a CSV file named "parsed_reviews.csv" into a DataFrame called `df` using `pd.read_csv()`.

4. Perform NER and extract entities:

    - An empty list called `entities` is created to store the extracted named entities.
    - A loop iterates over each review in the 'Body' column of the DataFrame `df`.
    - For each review, the NER model (`nlp`) is applied to obtain a processed document called `doc`.
    - The named entities (`ent.text`) from the processed document are extracted using a list comprehension and appended to the `entities` list.

5. Count the occurrences of each entity:

    - The `Counter` class is used to count the occurrences of each named entity in the `entities` list.
    - The result is stored in the `entity_counts` variable as a dictionary where the entities are the keys and the counts are the values.

6. Prepare data for visualization:

    - The `most_common()` method of `Counter` is used to retrieve the top 20 most frequent entities and their counts.
    - The `zip()` function is applied to separate the entities and counts into separate lists: `labels` and `counts`.

7. Create a bar chart of entity occurrences:

    - A figure with a size of 14 inches by 6 inches is created using `plt.figure(figsize=(14, 6))`.
    - A bar chart is created using the `labels` and `counts` data, where the x-axis represents the entities and the y-axis represents the counts.
    - The title, x-label, y-label, and x-axis tick labels are set for the plot to provide meaningful information.

8. Adjust the layout and display the plot:

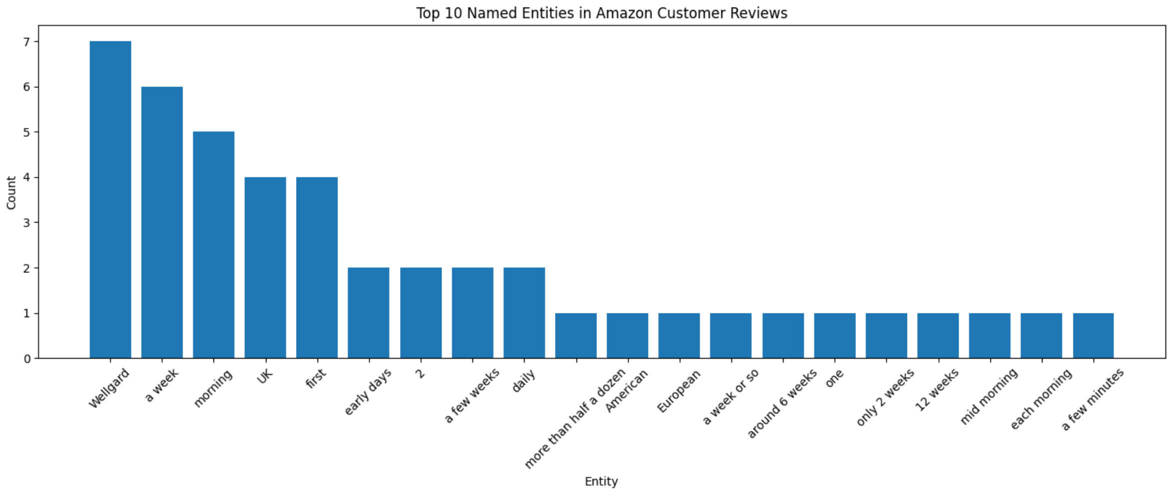
    - The `tight_layout` function is called to automatically adjust the subplot parameters for better visualization.
    - The `show` function is called to display the plot.

In summary, this step focuses on analyzing customer reviews from an Amazon dataset by extracting named entities using a pre-trained NER model. The extracted entities are then counted, and the top 20 most frequent entities are visualized using a bar chart. This analysis helps identify the most commonly mentioned named entities in the customer reviews.

In [ ]:

```
import pandas as pd
import spacy
import matplotlib.pyplot as plt
from collections import Counter
```

```
# Load the pre-trained NER model
nlp = spacy.load("en_core_web_sm")

# Read the CSV file
df = pd.read_csv("parsed_reviews.csv")

# Perform NER on each review and extract entities
entities = []
for review in df['Body']:
    doc = nlp(review)
    entities.extend([ent.text for ent in doc.ents if ent.text.strip()])

# Count the occurrences of each entity
entity_counts = Counter(entities)

# Prepare data for visualization
labels, counts = zip(*entity_counts.most_common(20))  # Top 20 most frequent ent

# Create a bar chart of entity occurrences
plt.figure(figsize=(14, 6))
plt.bar(labels, counts)
plt.title('Top 10 Named Entities in Amazon Customer Reviews')
plt.xlabel('Entity')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

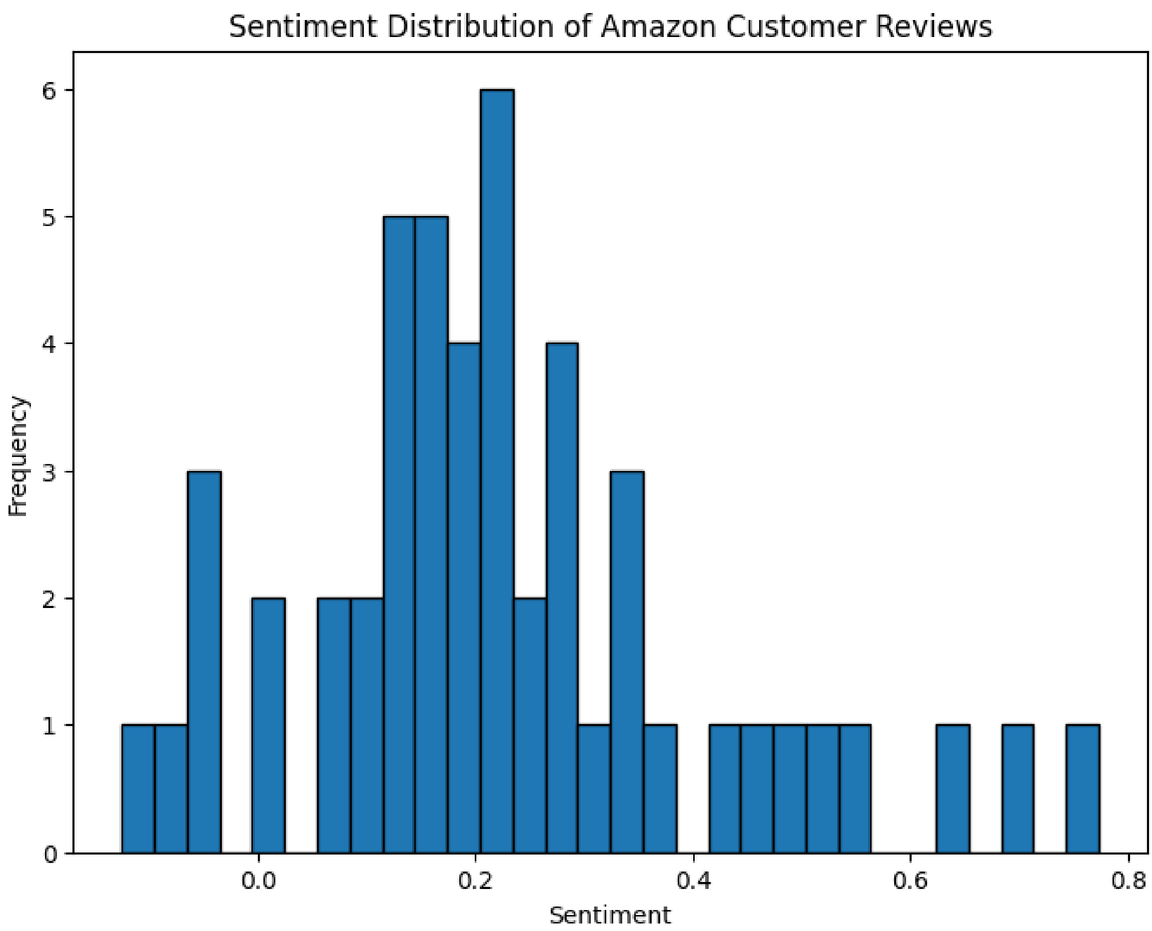


Sentiment Distrubution

In [ ]:

```
import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt

# Load the parsed_reviews.csv file
data = pd.read_csv('parsed_reviews.csv')

# Perform emotion detection on the reviews
data['Sentiment'] = data['Body'].apply(lambda x: TextBlob(x).sentiment.polarity)

# Plot the sentiment distribution
plt.figure(figsize=(8, 6))
```

```
plt.hist(data['Sentiment'], bins=30, edgecolor='k')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')
plt.title('Sentiment Distribution of Amazon Customer Reviews')
plt.show()
```



Word Embeddings and Similarity Analysis

This step involves analyzing Amazon customer reviews using natural language processing (NLP) techniques.

1. Importing the necessary libraries:

    - pandas: A library for data manipulation and analysis.
    - numpy: A library for numerical operations in Python.
    - matplotlib.pyplot: A library for creating visualizations in Python.
    - seaborn: A data visualization library built on top of matplotlib.
    - re: A module for regular expression operations.
    - spacy: A library for NLP tasks, including text processing and entity recognition.
    - sklearn.decomposition.PCA: A class for performing dimensionality reduction using Principal Component Analysis (PCA).
    - sklearn.metrics.pairwise.cosine_similarity: A function for calculating the cosine similarity between vectors.

2. Loading the pre-trained spaCy model:

- The project loads the pre-trained spaCy model 'en_core_web_md', which includes word vectors for English text.

3. Reading the CSV file:

    - The project reads the data from a CSV file named 'parsed_reviews.csv' using the pandas library.

4. Preprocessing the text data:

    - The project defines a function named `preprocess_text` that removes special characters and digits from the text and converts it to lowercase.
    - The function is applied to the 'Body' column of the data DataFrame using the `apply` method, and the preprocessed text is stored in a new column named 'Preprocessed Body'.

5. Calculating word embeddings:

    - The project calculates word embeddings for each review by iterating over the preprocessed text using the spaCy pipeline `nlp.pipe`.
    - The word embeddings are stored in a NumPy array named 'embeddings'.

6. Performing dimensionality reduction using PCA:

    - The project uses PCA to reduce the dimensionality of the word embeddings to two dimensions.
    - The PCA transformation is applied to the 'embeddings' array, and the transformed embeddings are stored in a new array named 'pca_embeddings'.

7. Calculating cosine similarity matrix:

    - The project calculates the cosine similarity matrix between the word embeddings using the `cosine_similarity` function from sklearn.metrics.pairwise.

8. Plotting the reviews in 2D space:

    - The project creates a scatter plot to visualize the reviews in the 2D space.
    - The x-axis and y-axis represent the first and second principal components (PC1 and PC2) of the PCA-transformed embeddings, respectively.
    - The points on the scatter plot are colored based on the 'Rating' column in the data DataFrame.

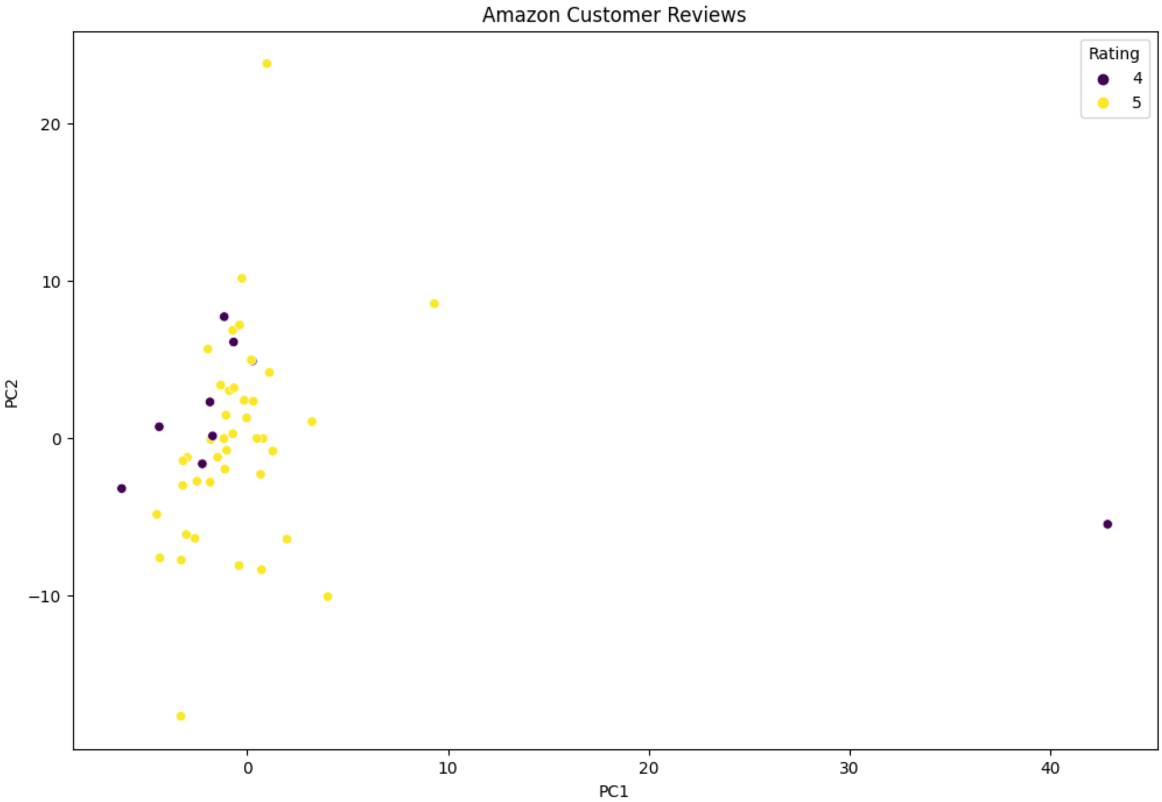
9. Displaying the most similar reviews:

    - The project identifies the index of a specific review based on its 'Review ID' in the data DataFrame.
    - It finds the most similar reviews to the chosen review based on the cosine similarity scores in the similarity matrix.
    - The top 5 similar reviews are printed along with their similarity scores.

Overall, this step focuses on preprocessing textual data, generating word embeddings, visualizing the reviews in a 2D space, and finding similar reviews based on cosine similarity.

In [ ]:
```
import pandas as pd
```

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import spacy
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity

# Load the pre-trained spaCy model
nlp = spacy.load('en_core_web_md')

# Read the CSV file
data = pd.read_csv('parsed_reviews.csv')

# Preprocess the text data
def preprocess_text(text):
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z]', ' ', text)

    # Convert text to lowercase
    text = text.lower()

    return text

# Apply text preprocessing to the 'Body' column
data['Preprocessed Body'] = data['Body'].apply(preprocess_text)

# Calculate word embeddings for each review
embeddings = []
for doc in nlp.pipe(data['Preprocessed Body'].values):
    embeddings.append(doc.vector)
embeddings = np.vstack(embeddings)

# Perform dimensionality reduction using PCA
pca = PCA(n_components=2)
pca_embeddings = pca.fit_transform(embeddings)

# Calculate cosine similarity matrix
similarity_matrix = cosine_similarity(embeddings)

# Plot the reviews in 2D space
plt.figure(figsize=(12, 8))
sns.scatterplot(x=pca_embeddings[:, 0], y=pca_embeddings[:, 1], hue=data['Rating
plt.title('Amazon Customer Reviews')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(title='Rating', loc='best')
plt.show()

# Display the most similar reviews
review_id = 'R3OR722EUPAF60'
review_index = data[data['Review ID'] == review_id].index[0]
similar_reviews = similarity_matrix[review_index].argsort()[::-1][1:6]
print('Most similar reviews to', review_id + ':')
for i, idx in enumerate(similar_reviews):
    print('Similar Review', i+1, ':', data.loc[idx, 'Body'])
    print('Similarity Score:', similarity_matrix[review_index, idx])
    print()
```




```
Most similar reviews to R3OR722EUPAF60:
Similar Review 1 : This is much cheaper than some other products available on Ama
zon. The company claims that the product is superior (based on some scientific da
ta that I can't actually verify). I reckon most of the collagen powders are more 
or less the same and it's actually quite difficult to know if they are helping, u
nless you are using them for the first time. Anyway I'm very happy with this coll
agen powder and will be buying from them again because the price is better. Impor
tant points are that the powder dissolves well - this does. It doesn't have any t
aste - I sometimes notice a slight taste - the final sip of coffee from my mug do
es have a slightly weird taste. But it's not too bad and I can live with it as mo
st of my coffee tastes fine. The only annoying part is that I find the lid diffic
ult to open and close and a screw lid would be more preferable, like my previous 
supplier. But, that's something I'll live with.
Similarity Score: 0.9618843

Similar Review 2 : I think that this product is so good that I have it on repeat-
supply. It does everything that it says on the tin (sorry). Mixing with other jui
ces, and coffee, in preparation for consumption is a breeze. I have to say that I
failed to make a satisfactory preparation with red wine... Compared to one other 
product which had a significant collagen content this is much better, no clumps a
nd therefore no frustration. Those who can remember mixing custard from powder wi
ll understand the technique.  As far as health benefits are concerned, I was gift
ed a high-collagen mango-flavoured product by a visitor. By the time that product
was used-up I noticed some improvements in my health and therefore tried to buy i
t again ... not available in the UK it seems. So I did some reading and found thi
s Wellgard product. Recommended!
Similarity Score: 0.95825714

Similar Review 3 : I have been taking Wellgard’s powered collagen religiously for
the past 2 months, and in that time, I have definitely noticed a difference in th
e condition of my skin. At first I was sceptical, however, I don’t seem to suffer
nearly as many breakouts as I used to, nor does it look as sallow. My skin genera
lly looks and feels so much healthier, and even has a glow to it. There’s not muc
h I can do about the wrinkles these day, but I believe that the collagen has help
ed with these as well, to a certain extent. Another bonus to this product is the 
price. At just under £20 it means the likes of me can afford to treat myself to s
```

```
omething that helps me look and feel better about myself. Which is a good thing.
I am very impressed with this product and will definitely be using it as part of
my daily ritual from now on.
Similarity Score: 0.9480535

Similar Review 4 : Really happy with the product and taking it daily. It's too ea
rly to notice the difference yet , but can leave a review in a few month again. G
reat value for money Will definitely choose a subscription for this product .
Similarity Score: 0.94575113

Similar Review 5 : I have used this product before and noticed my skin was less d
ry and more glowing. I train a lot too and want to support muscle growth and join
t strength which I feel this product does. Amazing price and contains peptides wh
ich is the main selling point of premium collagen so I'm back for more 👌
Similarity Score: 0.9345497
```

Opinion Mining

This step involves analyzing customer reviews from a CSV file using pandas and generating various visualizations.

1. Importing the necessary libraries:
    - pandas: A library for data manipulation and analysis.
    - matplotlib.pyplot: A library for creating visualizations in Python.
    - re: A module for regular expression operations.
2. Reading the CSV file:
    - The project reads the data from a CSV file named 'parsed_reviews.csv' using the pandas library. The data is stored in a DataFrame named 'df'.
3. Sentiment analysis using ratings:
    - The project assigns sentiment labels to each review based on the 'Rating' column. Ratings greater than 3 are labeled as 'Positive', ratings equal to 3 are labeled as 'Neutral', and ratings less than 3 are labeled as 'Negative'.
    - The number of reviews for each sentiment category is counted and stored in the 'sentiment_counts' variable.
4. Word count analysis:
    - The project calculates the word count for each review by splitting the 'Body' column text and counting the number of words.
    - A histogram is plotted to visualize the distribution of review word counts using matplotlib.pyplot. The histogram is divided into 20 bins, and the number of reviews is shown on the y-axis.
5. Date analysis:
    - The project performs analysis based on the 'Date' column. First, it extracts the date information from the 'Date' column using regular expressions, storing the result in the 'Date' column of the DataFrame.

- The 'Date' column is then converted to datetime format using the pd.to_datetime function. Any invalid dates are set to NaT (Not a Time) using the 'errors' parameter.
- The number of reviews per month is counted and stored in the 'monthly_counts' variable.

6. Plotting reviews over time:

- A line chart is created to visualize the number of reviews over time. The x-axis represents the months, and the y-axis represents the number of reviews.
- The 'monthly_counts' data is plotted using matplotlib.pyplot, with markers at each data point. The x-axis labels are rotated for better readability.

7. Verified purchase analysis:

- The project analyzes the distribution of verified and non-verified purchases using the 'Verified Purchase' column.
- The number of occurrences of each category is counted and stored in the 'verified_counts' variable.

8. Plotting a pie chart of verified purchase distribution:

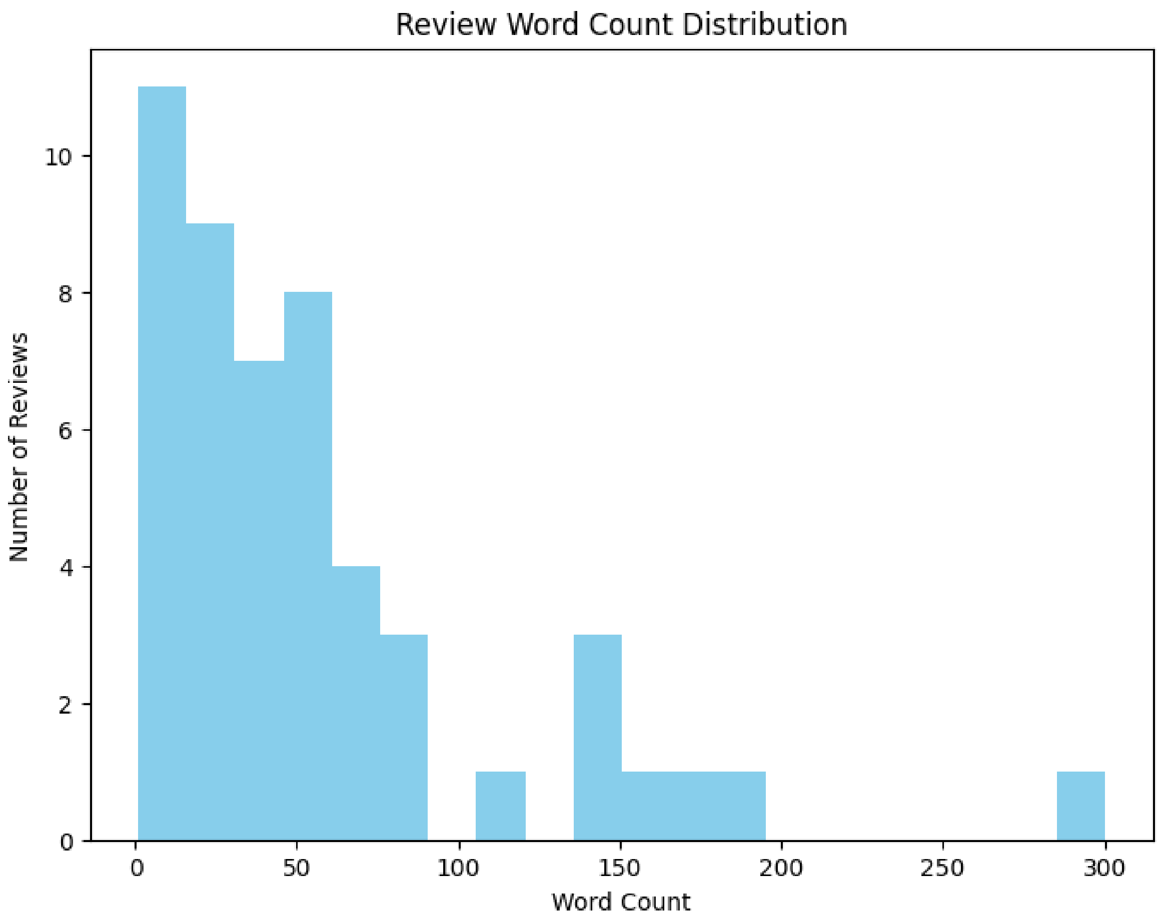
- A pie chart is created to visualize the distribution of verified and non-verified purchases.
- The 'verified_counts' data is plotted using matplotlib.pyplot, and the percentage values are displayed on the chart.

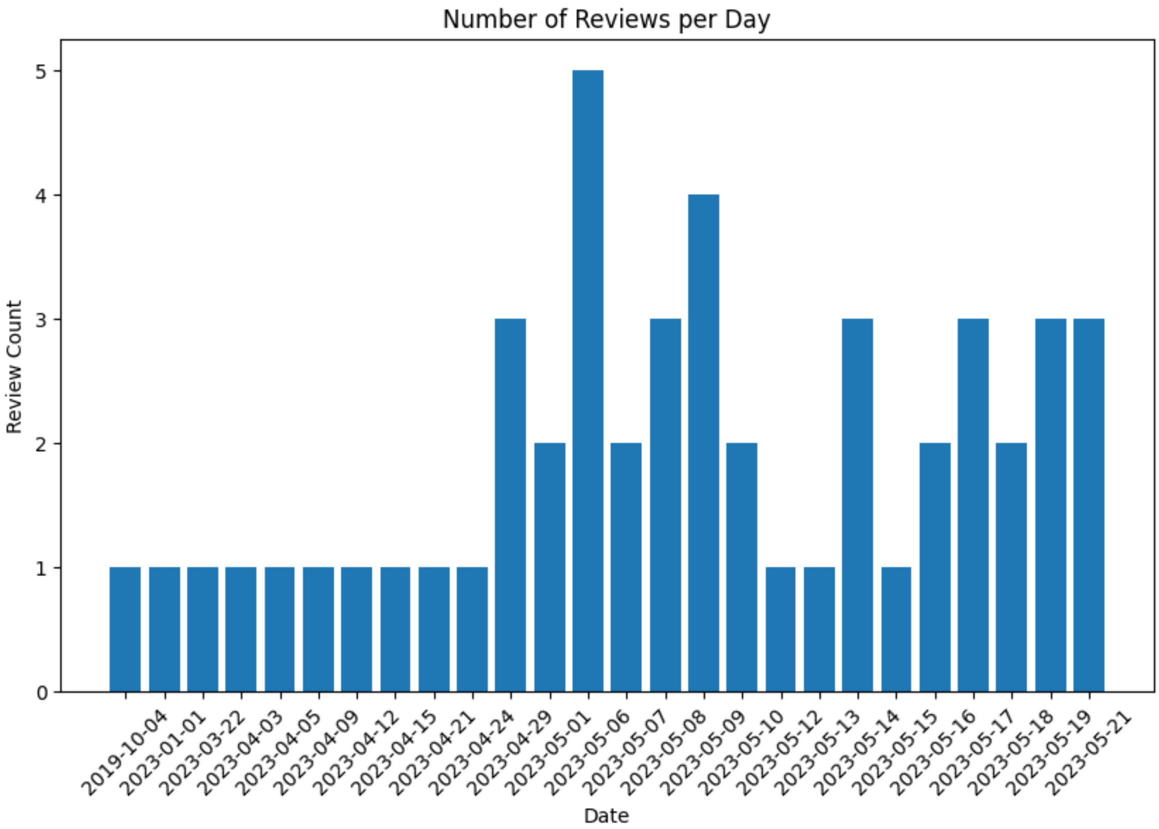
Overall, this step focuses on analyzing customer reviews by performing sentiment analysis, word count analysis, date analysis, and examining the distribution of verified purchases. The visualizations provide insights into sentiment distribution, review length, trends over time, and the proportion of verified purchases.

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re

# Read the CSV file
df = pd.read_csv('parsed_reviews.csv')

# Sentiment analysis using ratings
df['Sentiment'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else 'Neutral

# Count the number of reviews for each sentiment
sentiment_counts = df['Sentiment'].value_counts()


# Word count analysis
df['WordCount'] = df['Body'].apply(lambda x: len(str(x).split()))

# Plot a histogram of review word count
plt.figure(figsize=(8, 6))
plt.hist(df['WordCount'], bins=20, color='skyblue')
plt.title('Review Word Count Distribution')
plt.xlabel('Word Count')
plt.ylabel('Number of Reviews')
plt.show()
```

```python
# Date analysis
df['Date'] = df['Date'].apply(lambda x: re.findall(r'\d+\s\w+\s\d+', x)[0] if is
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Count the number of reviews per day
daily_counts = df['Date'].dt.to_period('D').value_counts().sort_index()

# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.bar(daily_counts.index.astype(str), daily_counts.values)
plt.title('Number of Reviews per Day')
plt.xlabel('Date')
plt.ylabel('Review Count')
plt.xticks(rotation=45)
plt.show()
```





Number of Reviews per Day
Review Count
Date
0
1
2
3
4
5
2019-10-04
2023-01-01
2023-03-22
2023-04-03
2023-04-05
2023-04-09
2023-04-12
2023-04-15
2023-04-21
2023-04-24
2023-04-29
2023-05-01
2023-05-06
2023-05-07
2023-05-08
2023-05-09
2023-05-10
2023-05-12
2023-05-13
2023-05-14
2023-05-15
2023-05-16
2023-05-17
2023-05-18
2023-05-19
2023-05-21