```
'''
from google.colab import drive
drive.mount('/content/drive')
'''
```

```
'\nfrom google.colab import drive\ndrive.mount('/content/drive')\n'
```

```
'''
https://drive.google.com/uc?id=REPLACE_WITH_FILE_ID&export=download

https://drive.google.com/uc?id=1MMnZ1hc8xSI1W_8oAN3Zse_3k3OxRhMx&export=download

'''
```

```
'\nhttps://drive.google.com/uc?id=REPLACE_WITH_FILE_ID&export=download\n\nhttps://driv
e.google.com/uc?id=1MMnZ1hc8xSI1W_8oAN3Zse_3k3OxRhMx&export=download\n\n'
```

```
import pandas as pd

# Load the CSV file
file_path = '/content/drive/MyDrive/DSAI_Capstone/Reviews.csv'
reviews_df = pd.read_csv(file_path, sep=',')

# Display the first few rows of the dataframe
reviews_df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessD
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	

```
reviews_df.shape
```

```
(568454, 10)
```

```
reviews_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568428 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

```
reviews_df.isnull().sum()
```

```
Id                        0
ProductId                 0
UserId                    0
ProfileName              26
HelpfulnessNumerator      0
HelpfulnessDenominator    0
Score                     0
```

```
Time                      0
Summary                  27
Text                      0
dtype: int64
```

```
reviews_df.dropna(inplace=True)
```

```
reviews_df.shape
```

```
(568401, 10)
```

```
reviews_df.drop_duplicates(inplace=True)
```

```
reviews_df.shape
```

```
(568401, 10)
```

```
reviews_df.describe()
```

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	
count	568401.000000	568401.000000	568401.000000	568401.000000	5.6840
mean	284228.120582	1.743903	2.227911	4.183297	1.2962
std	164098.955187	7.636845	8.288820	1.310376	4.8038
min	1.000000	0.000000	0.000000	1.000000	9.3934
25%	142116.000000	0.000000	0.000000	4.000000	1.2712
50%	284225.000000	0.000000	1.000000	5.000000	1.3111
75%	426342.000000	2.000000	2.000000	5.000000	1.3327
max	568454.000000	866.000000	923.000000	5.000000	1.3512

```
reviews_df['Summary'].value_counts()
```

```
Summary
Delicious!                            2462
Delicious                             2316
Yummy!                                1594
Yummy                                 1268
Great product                         1091
                                      ...
Goodie Bag Gift                          1
i buy this in bulk                       1
Makes a super easy jerk marinade         1
Best for Diabetics                       1
Favorite Training and reward treat       1
Name: count, Length: 295734, dtype: int64
```

```
reviews_df['UserId'].nunique()
```

```
256042
```

```
reviews_df['ProductId'].nunique()
```

```
74257
```

```
reviews_df['Text'].nunique()
```

```
393560
```

```
reviews_df['Text'].duplicated().sum()
```

```
174841
```

```
reviews_df[['Text', 'UserId', 'ProductId']].duplicated().sum()
```

```
1309
```

```
reviews_df[['Text', 'UserId', 'Score', 'ProfileName', 'Time']].value_counts().head(10)
```

```
inside cells of body tissues and organs. Only 1% of magnesium is found in blood, but the body works very hard to keep blood levels
of magnesium constant.<br />Magnesium is needed for more than 300 biochemical reactions in the body. It helps maintain normal muscle
and nerve function, keeps heart rhythm steady, supports a healthy immune system, and keeps bones strong. Magnesium also helps
regulate blood sugar levels, promotes normal blood pressure, and is known to be involved in energy metabolism and protein synthesis.
There is an increased interest in the role of magnesium in preventing and managing disorders such as hypertension, cardiovascular
disease, and diabetes. Dietary magnesium is absorbed in the small intestines. Are you taking enough magnesium in the diet? The
recommended daily allowance is 300mgs/day for men, 350mgs for women. Magnesium is extremely safe by mouth - too much simply causes
diarrhea. Try increasing the amount of magnesium you take by mouth until it causes diarrhea, then reduce the dose slightly so it
does not. This is called taking magnesium to bowel tolerance (just like using vitamin C to bowel tolerance.<br /><br />Almonds are a
good source of Fiber.<br />One ounce contains approximately 12% of the RDA for an adult.<br />So, in summary, Almonds are a very
healthy snack, so what's left to say?<br /><br />Two things:<br />1st be cautious about "bowel tolerance" or "happy bowels" if you
prefer.<br />2nd Taste.  I realize that taste is a personal thing.  That's why they make chocolate, strawberry, and vanilla.<br />
<br />Here's my current favorites"<br /><br /> 1st Blue Diamond Dark mint dark chocolate (reminds me of the chocolate mint Girl
Scout Cookies with less sugar).<br /><br />2nd Blue diamond Dark Chocolate Oven Roasted<br />3rd  Blue diamond Blue Butter Toffee<br
/>4thd Blue Diamond Almonds Bold Habanero BBQ (reminds me of  BBQ potato chips again with less sugar).<br />5th Bold Blazin Buffalo
Wing<br />6th Blue Diamond Almonds Cinnamon Brown Sugar<br />7th  Blue Diamond Almonds Lime `n Chili<br />8th  Blue Diamond Almonds
Smokehouse<br />9th  Bold Wasabi & Soy Sauce<br /><br />But that's just me: you might have a completely different take on these
flavors.<br /> 4th  and 5th are very close.<br /><br />Gunner<br />May, 2012
A1TMAVN4CEM8U8  5      Gunner                    1336348800     43
Stash Chamomile Herbal Tea is tea bags with dried, crushed chamomile flowers.<br /><br />Honestly, as far as I'm concerned,
chamomile is chamomile.  I suppose that when you use loose flowers you get sediment in the cup and I have seen this happen with some
brands of chamomile tea bags, these don't do that.<br /><br />I have also had organic chamomile (this isn't) but I can't taste the
difference.<br /><br />Each bag is individually packaged and that is either good or bad depending on how and where you use it and if
you are concerned about excessive packaging.<br /><br />If you like chamomile tea, this is a good choice.
A1UQBFCERIP7VJ  4      Margaret Picky            1321401600     38
The pet food industry can be one of the most infuriating as you start doing research and discovering what trash manufacturers (and
the vets who get paid off by the manufacturers) push onto the unsuspecting public. For reference, don't ask your vet - do some
Internet research at sites like, for example, [...]. What you'll find is that even the supposed "high-end" lines from basically
every major supermarket label are horrible for your fuzzy friends. People who trust the marketing or trust their vets overpay and
feed their cats foods that are full of carcinogens, cheap ingredients that are inappropriate for a cat's digestive system, and
byproducts that provide nothing of value and are basically a way to make cheap profit off of literal garbage. It's a tragedy, as
these same cats end up with shorter lifespans, kidney problems, UTIs, cancer, and a host of other issues, all of which could have
been avoided by a high quality food.<br /><br />This is an A-Grade food. What you'll find here is very high protein content, many
meat meals, potatoes, salmon oil, flax seed, and no grains, which is great as a cat is not an omnivore, it's a carnivore, and it
simply does not need grains. It has 50 percent protein, which is among the highest you'll see in cat food. The first 4 ingredients
are all meats.<br /><br />What you will NOT find here is the kind of junk you'll see in things like Purina One and Science Diet -
byproducts (aka beaks, feathers, bird feet), brewers rice (aka the waste product from making beer), wheat flour (a leading cause of
pet allergies), corn gluten (a cheap waste of space), and things like "liver flavor" (because they don't want to pay for the real
thing). I didn't just pull those ingredients out of the air - I was reading the ingredients list of Purina One. It's pathetic.<br />
<br />Don't feed your cats trash. Pay a little more to feed them good food now, and they'll lead longer, healthier, more satisfied
lives as a result. It's worth it.
A29JUMRL1US6YP  5      HTBK                      1278201600     35
I understand all the complaints about Science Diet. Believe me I avoided this food for quite awhile based on that information. I'm
not thrilled about the filler, would prefer a food with no by-products, and yes in general think it is over priced. But my cat was
throwing up everything else. And I do mean EVERYTHING else. I have tried so many top of the line foods for him and every single one
he threw up. I took him to the vet, had all the xrays done, all the blood work done, put him on antibotics just in case... nothing.
He is a perfectly healthy cat except that he was throwing up food 2 to 3 times a week. I talked to my mom who had 2 Siamese cats
that lived into thier late teens and she said she fed them Science Diet. In fact, one of them had urninary tract problems and the SD
was the only food that stopped his problems. So yes, I understand everything everyone is saying about why SD is so horrible... yet I
can't argue with the results. I fed my cat this and it's been 3 weeks with no vomiting. My mom fed it to her cats and they lived
long, healthy lives. I just can't argue with the results. And even if there were risks it's worth it not to have my poor kitty
completely emptying his stomach 2-3 times a week from food that is making him sick. If your cat keeps throwing up, you've had the
vet check him and there is nothing wrong and no other food is working... try this food. It really did work for my cat.
A25C5MVVCIYT5D  5      Natalie Dawn              1304726400     34
My kids (18 mo and 3 yrs) love the HappyBaby Tots.  We have tried every flavor and they eat all of them.  I love that they are
getting good, organic nutrition, and that the ingredients are wholesome.  They are very convenient to throw in your diaper bag or
purse on-the-go, or to stick in a lunch box for a snack.  We use a spoon and sometimes a bowl at home for self feeding with my
little one, but you can also give the pouch to them to eat directly from the squeeze pouch.  Thank you, HappyBaby, for such great
products!!
```

```
reviews_df.duplicated(subset=['Text', 'UserId', 'Score', 'ProfileName', 'Time']).sum()
```

```
174412
```

Start coding or generate with AI.

```
reviews_df.shape
```

```
(568401, 10)
```

```
reviews_df.drop_duplicates(subset=['Text', 'UserId', 'Score', 'ProfileName', 'Time'], inplace=True)
```

```
reviews_df.shape
```

```
(393989, 10)
```

```
reviews_df.drop_duplicates(subset=['Text'], inplace=True)
```

```
reviews_df.shape
```

```
(393560, 10)
```

```
reviews_df['Text'].nunique()
```

```
393560
```

Start coding or generate with AI.

Start coding or generate with AI.

```
reviews_df_cleaned=reviews_df.copy()
```

```
# Convert the 'Time' column to datetime format
reviews_df_cleaned['Time'] = pd.to_datetime(reviews_df_cleaned['Time'], unit='s')

# Display the first few rows to verify the conversion
reviews_df_cleaned[['Time']].head()
```

	Time
0	2011-04-27
1	2012-09-07
2	2008-08-18
3	2011-06-13
4	2012-10-21

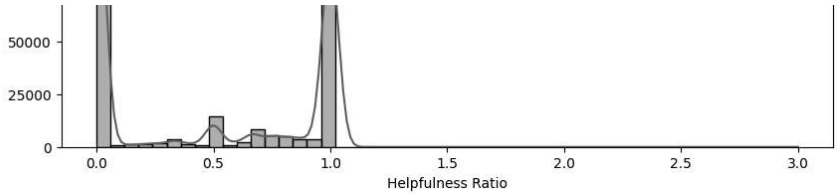
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
```

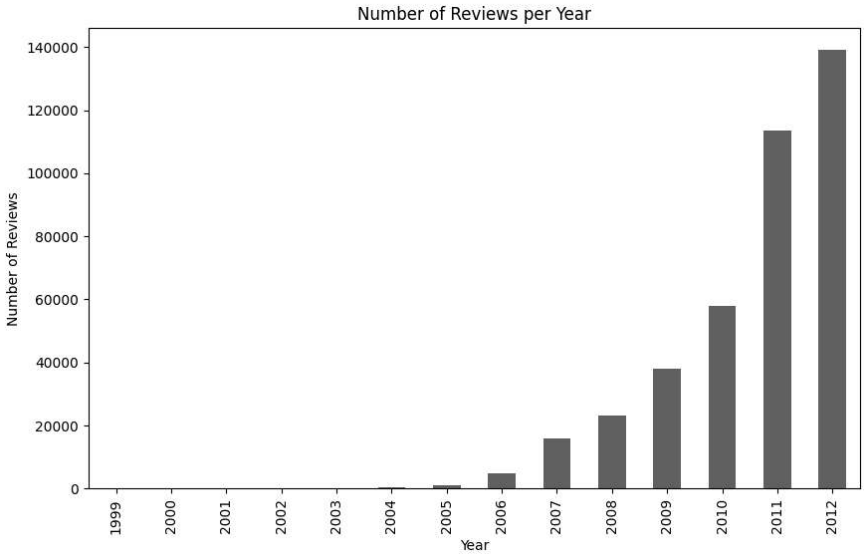
```
# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Initialize lemmatizer and stop words
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Function to preprocess text
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stop words and lemmatize
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word.isalnum() and word not in stop_words]
    # Join tokens back into a single string
    return ' '.join(tokens)

# Apply the preprocessing to the 'Text' and 'Summary' columns
reviews_df_cleaned['Processed_Text'] = reviews_df_cleaned['Text'].apply(preprocess_text)
reviews_df_cleaned['Processed_Summary'] = reviews_df_cleaned['Summary'].apply(preprocess_text)

# Display the first few rows to verify the changes
reviews_df_cleaned[['Time', 'Processed_Text', 'Processed_Summary']].head()
```

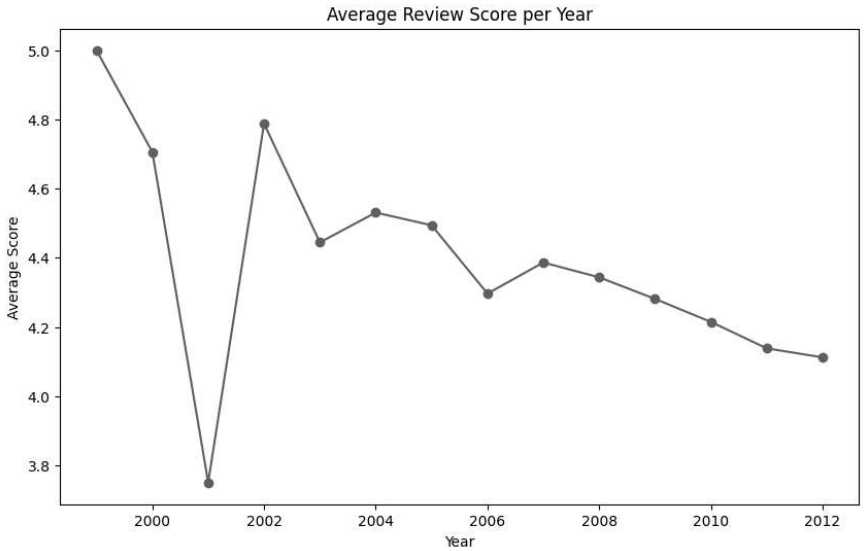
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

	Time	Processed_Text	Processed_Summary
0	2011-04-27	bought several vitality canned dog food produc...	good quality dog food
1	2012-09-07	product arrived labeled jumbo salted peanutsth...	advertised
2	2008-08-18	confection around century light pillowy citrus...	delight say
3	2011-06-13	looking secret ingredient robitussin believe f...	cough medicine
4	2012-10-21	great taffy great price wide assortment yummy ...	great taffy

```
import matplotlib.pyplot as plt
import seaborn as sns

# Basic Statistics
basic_stats = reviews_df_cleaned.describe()

# Distribution of Scores
plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Score'], bins=5, kde=True)
plt.title('Distribution of Product Scores')
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.show()

# Review Length Analysis
reviews_df_cleaned['Review_Length'] = reviews_df_cleaned['Processed_Text'].apply(len)
reviews_df_cleaned['Summary_Length'] = reviews_df_cleaned['Processed_Summary'].apply(len)

plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Review_Length'], bins=50, kde=True)
plt.title('Distribution of Review Lengths')
plt.xlabel('Review Length')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Summary_Length'], bins=50, kde=True)
plt.title('Distribution of Summary Lengths')
plt.xlabel('Summary Length')
plt.ylabel('Frequency')
plt.show()

# Helpfulness Analysis
reviews_df_cleaned['Helpfulness_Ratio'] = reviews_df_cleaned['HelpfulnessNumerator'] / (reviews_df_cleaned['HelpfulnessDenominator'] + 1e-5)

plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Helpfulness_Ratio'], bins=50, kde=True)
plt.title('Distribution of Helpfulness Ratios')
plt.xlabel('Helpfulness Ratio')
plt.ylabel('Frequency')
plt.show()

# Time-Based Analysis
reviews_df_cleaned['Year'] = reviews_df_cleaned['Time'].dt.year
yearly_review_counts = reviews_df_cleaned.groupby('Year').size()
yearly_average_score = reviews_df_cleaned.groupby('Year')['Score'].mean()

plt.figure(figsize=(10, 6))
yearly_review_counts.plot(kind='bar')
plt.title('Number of Reviews per Year')
plt.xlabel('Year')
plt.ylabel('Number of Reviews')
plt.show()

plt.figure(figsize=(10, 6))
yearly_average_score.plot(kind='line', marker='o')
plt.title('Average Review Score per Year')
plt.xlabel('Year')
plt.ylabel('Average Score')
plt.show()
```


50000
25000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Helpfulness Ratio


Number of Reviews per Year
Number of Reviews
140000
120000
100000
80000
60000
40000
20000
0
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Year


Average Review Score per Year
Average Score
5.0
4.8
4.6
4.4
4.2
4.0
3.8
2000
2002
2004
2006
2008
2010
2012
Year

Start coding or generate with AI.

```
!pip install nltk # Make sure nltk is installed
import nltk
nltk.download('vader_lexicon')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.4)
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize the VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Function to get sentiment scores
def get_sentiment(text):
    sentiment_scores = sia.polarity_scores(text)
    return sentiment_scores

# Apply the function to the 'Processed_Text' column
reviews_df_cleaned['Sentiment_Scores'] = reviews_df_cleaned['Processed_Text'].apply(get_sentiment)

# Extract compound scores to determine overall sentiment
reviews_df_cleaned['Compound_Score'] = reviews_df_cleaned['Sentiment_Scores'].apply(lambda x: x['compound'])

# Classify sentiment based on compound score
def classify_sentiment(score):
    if score >= 0.05:
        return 'positive'
    elif score <= -0.05:
        return 'negative'
    else:
        return 'neutral'

reviews_df_cleaned['Sentiment'] = reviews_df_cleaned['Compound_Score'].apply(classify_sentiment)

# Display the first few rows to verify
reviews_df_cleaned[['Processed_Text', 'Sentiment_Scores', 'Compound_Score', 'Sentiment']].head()
```

	Processed_Text	Sentiment_Scores	Compound_Score	Sentiment
0	bought several vitality canned dog food produc...	{'neg': 0.0, 'neu': 0.503, 'pos': 0.497, 'comp...	0.9413	positive
1	product arrived labeled jumbo salted peanutsth...	{'neg': 0.129, 'neu': 0.762, 'pos': 0.11, 'com...	-0.1027	negative
2	confection around century light pillowy citrus...	{'neg': 0.132, 'neu': 0.576, 'pos': 0.292, 'co...	0.8624	positive
3	looking secret ingredient	{'neg': 0.0, 'neu': 0.854, 'pos':	0.4404	positive

```
reviews_df_cleaned['Sentiment'].value_counts()
```

```
Sentiment
positive    353666
negative     30995
neutral       8899
Name: count, dtype: int64
```

Start coding or generate with AI.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Vectorize the text data using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(reviews_df_cleaned['Processed_Text'])
y = reviews_df_cleaned['Sentiment']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Logistic Regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = log_reg.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy: 0.935499034454721
Confusion Matrix:
 [[ 3379    46  2758]
 [  201   251  1391]
 [  665    16 70005]]
Classification Report:
               precision    recall  f1-score   support

    negative       0.80      0.55      0.65      6183
     neutral       0.80      0.14      0.23      1843
    positive       0.94      0.99      0.97     70686

    accuracy                           0.94     78712
   macro avg       0.85      0.56      0.62     78712
weighted avg       0.93      0.94      0.92     78712
```

Start coding or generate with AI.

Start coding or generate with AI.

Double-click (or enter) to edit

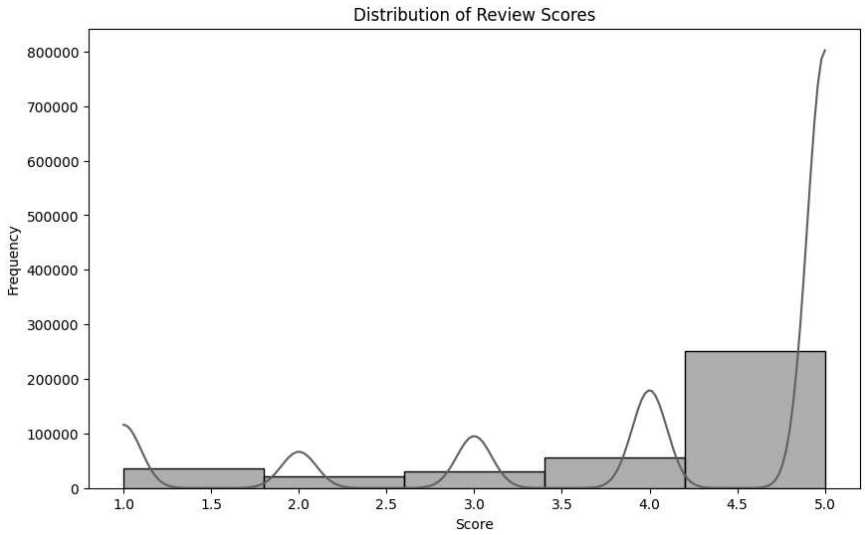
```
# Display dataset overview
print(reviews_df_cleaned.info())
print(reviews_df_cleaned.head())
```

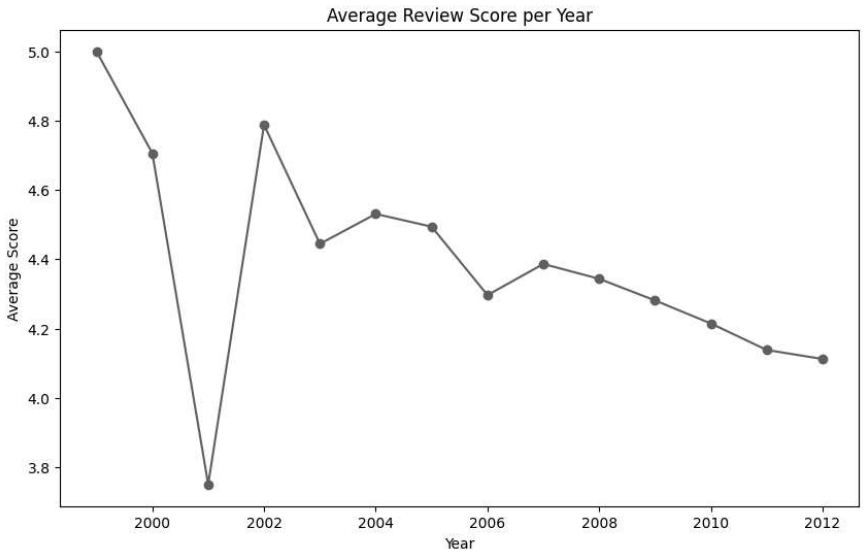
```
<class 'pandas.core.frame.DataFrame'>
Index: 393560 entries, 0 to 568453
Data columns (total 19 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      393560 non-null  int64
 1   ProductId               393560 non-null  object
 2   UserId                  393560 non-null  object
 3   ProfileName             393560 non-null  object
 4   HelpfulnessNumerator    393560 non-null  int64
 5   HelpfulnessDenominator  393560 non-null  int64
 6   Score                   393560 non-null  int64
 7   Time                    393560 non-null  datetime64[ns]
 8   Summary                 393560 non-null  object
 9   Text                    393560 non-null  object
 10  Processed_Text          393560 non-null  object
 11  Processed_Summary       393560 non-null  object
 12  Review_Length           393560 non-null  int64
 13  Summary_Length          393560 non-null  int64
 14  Helpfulness_Ratio       393560 non-null  float64
 15  Year                    393560 non-null  int32
 16  Sentiment_Scores        393560 non-null  object
 17  Compound_Score          393560 non-null  float64
 18  Sentiment               393560 non-null  object
dtypes: datetime64[ns](1), float64(2), int32(1), int64(6), object(9)
memory usage: 58.6+ MB
None
   Id   ProductId          UserId                      ProfileName  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

   HelpfulnessNumerator  HelpfulnessDenominator  Score       Time  \
0                     1                       1      5 2011-04-27
1                     0                       0      1 2012-09-07
2                     1                       1      4 2008-08-18
3                     3                       3      2 2011-06-13
4                     0                       0      5 2012-10-21

                 Summary                                               Text  \
0  Good Quality Dog Food  I have bought several of the Vitality canned d...
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
2  "Delight" says it all  This is a confection that has been around a fe...
3         Cough Medicine  If you are looking for the secret ingredient i...
4            Great taffy  Great taffy at a great price.  There was a wid...

                                     Processed_Text      Processed_Summary  \
0  bought several vitality canned dog food produc...  good quality dog food
1  product arrived labeled jumbo salted peanutsth...             advertised
2  confection around century light pillowy citrus...            delight say
3  looking secret ingredient robitussin believe f...         cough medicine
4  great taffy great price wide assortment yummy ...            great taffy

   Review_Length  Summary_Length  Helpfulness_Ratio  Year  \
0            161              21           0.999990  2011
1            135              10           0.000000  2012
```

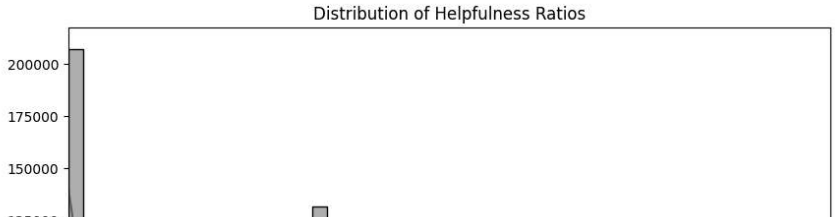
```
# Display number of null values removed
initial_row_count = 568454  # Initial row count provided
final_row_count = reviews_df_cleaned.shape[0]
null_values = 53  # Number of null values removed
print(f"Number of null values removed: {null_values}")

# Display number of duplicate entries removed
duplicates_removed = initial_row_count - null_values - reviews_df_cleaned.shape[0]
print(f"Number of duplicate entries removed: {duplicates_removed}")

# Display sample of preprocessed text
print(reviews_df_cleaned[['Processed_Text', 'Processed_Summary']].head())
```

```
Number of null values removed: 53
Number of duplicate entries removed: 174841
                                        Processed_Text      Processed_Summary
0  bought several vitality canned dog food produc...  good quality dog food
1  product arrived labeled jumbo salted peanutsth...             advertised
2  confection around century light pillowy citrus...            delight say
3  looking secret ingredient robitussin believe f...         cough medicine
4  great taffy great price wide assortment yummy ...            great taffy
```

```
# Distribution of review scores (save or display)
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Score'], kde=True, bins=5)
plt.title('Distribution of Review Scores')
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.savefig('distribution_of_review_scores.png')
plt.show()

# Trends in review scores over time (save or display)
reviews_df_cleaned['Year'] = reviews_df_cleaned['Time'].dt.year
average_score_per_year = reviews_df_cleaned.groupby('Year')['Score'].mean()

plt.figure(figsize=(10, 6))
plt.plot(average_score_per_year.index, average_score_per_year.values, marker='o')
plt.title('Average Review Score per Year')
plt.xlabel('Year')
plt.ylabel('Average Score')
plt.savefig('average_review_score_per_year.png')
plt.show()

# Distribution of helpfulness ratios (save or display)
plt.figure(figsize=(10, 6))
sns.histplot(reviews_df_cleaned['Helpfulness_Ratio'], kde=True, bins=50)
plt.title('Distribution of Helpfulness Ratios')
plt.xlabel('Helpfulness Ratio')
plt.ylabel('Frequency')
plt.xlim(0, 3)
plt.savefig('distribution_of_helpfulness_ratios.png')
plt.show()
```


Distribution of Review Scores
Frequency
Score
800000
700000
600000
500000
400000
300000
200000
100000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Average Review Score per Year
Average Score
Year
5.0
4.8
4.6
4.4
4.2
4.0
3.8
2000
2002
2004
2006
2008
2010
2012


Distribution of Helpfulness Ratios
200000
175000
150000

```
# Sample of sentiment scores and classifications
print(reviews_df_cleaned[['Processed_Text', 'Sentiment_Scores', 'Compound_Score', 'Sentiment']].head())

# Distribution of sentiments
sentiment_distribution = reviews_df_cleaned['Sentiment'].value_counts()
print(sentiment_distribution)
```

```
                                      Processed_Text  \
0  bought several vitality canned dog food produc...
1  product arrived labeled jumbo salted peanutsth...
2  confection around century light pillowy citrus...
3  looking secret ingredient robitussin believe f...
4  great taffy great price wide assortment yummy ...

                                     Sentiment_Scores  Compound_Score Sentiment
0  {'neg': 0.0, 'neu': 0.503, 'pos': 0.497, 'comp...          0.9413  positive
1  {'neg': 0.129, 'neu': 0.762, 'pos': 0.11, 'com...         -0.1027  negative
2  {'neg': 0.132, 'neu': 0.576, 'pos': 0.292, 'co...          0.8624  positive
3  {'neg': 0.0, 'neu': 0.854, 'pos': 0.146, 'comp...          0.4404  positive
4  {'neg': 0.0, 'neu': 0.369, 'pos': 0.631, 'comp...          0.9468  positive
Sentiment
positive    353666
negative     30995
neutral       8899
Name: count, dtype: int64
```

```
# Details on TF-IDF vectorization
print(f"Number of features: {X.shape[1]}")
print("Example feature names:", tfidf_vectorizer.get_feature_names_out()[:10])

# Train/test split details
print(f"Train set size: {X_train.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")

# Model evaluation metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Number of features: 5000
Example feature names: ['0br' '0g' '0gbr' '10' '100' '1000' '1015' '11' '110' '12']
Train set size: 314848
Test set size: 78712
Accuracy: 0.935499034454721
Confusion Matrix:
 [[ 3379    46  2758]
 [  201   251  1391]
 [  665    16 70005]]
Classification Report:
               precision    recall  f1-score   support

    negative       0.80      0.55      0.65      6183
     neutral       0.80      0.14      0.23      1843
    positive       0.94      0.99      0.97     70686

    accuracy                           0.94     78712
   macro avg       0.85      0.56      0.62     78712
weighted avg       0.93      0.94      0.92     78712
```

```
# Original indices of the test set
test_indices = y_test.index

# Fetch the corresponding processed text using the original indices
example_predictions = pd.DataFrame({
    'Processed_Text': reviews_df_cleaned.loc[test_indices, 'Processed_Text'].values,
    'True_Sentiment': y_test.values,
    'Predicted_Sentiment': y_pred
})

# Display example predictions
print(example_predictions.head())
```

```
                                     Processed_Text True_Sentiment  \
0  product provides little taste sweet without co...       positive
1  cat love cat food stand near food dish begin c...       positive
2  knew kit would soldered assembled great fun pr...       positive
3  love water offered amazon vine program already...       positive
4  im used eating medjool date big plump juicy lo...       positive

  Predicted_Sentiment
0            positive
1            positive
2            positive
3            positive
4            positive
```

```
example_predictions[['True_Sentiment', 'Predicted_Sentiment']].value_counts()
```

```
True_Sentiment  Predicted_Sentiment
positive        positive               70005
negative        negative                3379
                positive                2758
neutral         positive                1391
positive        negative                 665
neutral         neutral                  251
                negative                 201
negative        neutral                   46
positive        neutral                   16
Name: count, dtype: int64
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Display performance metrics
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Generate and display confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```

```
Accuracy: 0.935499034454721
Precision: 0.9290909936356478
Recall: 0.935499034454721
F1 Score: 0.9244450455130108
Confusion Matrix:
[[ 3379    46  2758]
 [  201   251  1391]
 [  665    16 70005]]
```

Start coding or generate with AI.

Double-click (or enter) to edit