Amazon Reviews Analysis

November 26, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# load dataset
```

[3]:
```
reviews = pd.read_csv("Amazon_Reviews.csv", encoding='utf8', engine='python')
reviews.head()
```

[3]:
```
      Reviewer Name                     Profile Link Country Review Count  \
0        Eugene ath  /users/66e8185ff1598352d6b3701a      US     1 review
1  Daniel ohalloran  /users/5d75e460200c1f6a6373648c      GB    9 reviews
2          p fisher  /users/546cfcf1000064000197b88f      GB   90 reviews
3         Greg Dunn  /users/62c35cdbacc0ea0012ccaffa      AU    5 reviews
4     Sheila Hannah  /users/5ddbe429478d88251550610e      GB    8 reviews

                Review Date                  Rating  \
0  2024-09-16T13:44:26.000Z  Rated 1 out of 5 stars
1  2024-09-16T18:26:46.000Z  Rated 1 out of 5 stars
2  2024-09-16T21:47:39.000Z  Rated 1 out of 5 stars
3  2024-09-17T07:15:49.000Z  Rated 1 out of 5 stars
4  2024-09-16T18:37:17.000Z  Rated 1 out of 5 stars

                                Review Title  \
0      A Store That Doesn't Want to Sell Anything
1          Had multiple orders one turned up and…
2                     I informed these reprobates
3  Advertise one price then increase it on website
4             If I could give a lower rate I would

                                     Review Text  Date of Experience
0  I registered on the website, tried to order a …  September 16, 2024
1  Had multiple orders one turned up and driver h…  September 16, 2024
2  I informed these reprobates that I WOULD NOT B…  September 16, 2024
3  I have bought from Amazon before and no proble…  September 17, 2024
4  If I could give a lower rate I would! I cancel…  September 16, 2024
```

[4]:
```
# check missing value
```

[5]:
```python
missing_values = reviews.isnull().sum()
missing_values
```

[5]:
```
Reviewer Name           0
Profile Link           51
Country               160
Review Count          159
Review Date           159
Rating                159
Review Title          159
Review Text           159
Date of Experience    267
dtype: int64
```

[6]:
```python
reviews.shape
```

[6]:
```
(21214, 9)
```

[7]:
```python
# drop na values
```

[8]:
```python
reviews_new = reviews.dropna()
reviews_new.shape
```

[8]:
```
(20946, 9)
```

[9]:
```python
# drop Drop any features that are not useful for your model building and␣
↪explain why they are not useful.
```

[10]:
```python
columns_to_drop = ['Reviewer Name', 'Profile Link', 'Country', 'Review Count',
                   'Review Title', 'Date of Experience']
```

[11]:
```python
reviews_df = reviews_new.drop(columns = columns_to_drop)
```

[12]:
```python
reviews_df.head()
```

[12]:
```
                Review Date                  Rating  \
0  2024-09-16T13:44:26.000Z  Rated 1 out of 5 stars
1  2024-09-16T18:26:46.000Z  Rated 1 out of 5 stars
2  2024-09-16T21:47:39.000Z  Rated 1 out of 5 stars
3  2024-09-17T07:15:49.000Z  Rated 1 out of 5 stars
4  2024-09-16T18:37:17.000Z  Rated 1 out of 5 stars

                                         Review Text
0  I registered on the website, tried to order a …
1  Had multiple orders one turned up and driver h…
2  I informed these reprobates that I WOULD NOT B…
3  I have bought from Amazon before and no proble…
4  If I could give a lower rate I would! I cancel…
```

Based on the dataset, the columns I dropped has no contribute directly to the prediction task, and are highly redundant or irrelevant for the model building. Like 'Reviewer Name', 'Profile Link', 'Country', 'Review Count', 'Review Title', 'Date of Experience'.

[13]:
```
# extract rating number from "Rating" column using regular expressions
```

[14]:
```
reviews_df['Rating'] = reviews_df['Rating'].str.extract('(\d)').astype(int)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
/var/folders/6q/k8jdwbv174s78xj3x3kzvn0w0000gn/T/ipykernel_20486/1311482941.py:1
: SyntaxWarning: invalid escape sequence '\d'
  reviews_df['Rating'] = reviews_df['Rating'].str.extract('(\d)').astype(int)
```

[15]:
```
reviews_df['Rating'] = pd.to_numeric(reviews_df['Rating'])
```

[16]:
```
reviews_df.head(5)
```

[16]:
```
                Review Date  Rating  \
0  2024-09-16T13:44:26.000Z       1
1  2024-09-16T18:26:46.000Z       1
2  2024-09-16T21:47:39.000Z       1
3  2024-09-17T07:15:49.000Z       1
4  2024-09-16T18:37:17.000Z       1

                                         Review Text
0  I registered on the website, tried to order a …
1  Had multiple orders one turned up and driver h…
2  I informed these reprobates that I WOULD NOT B…
3  I have bought from Amazon before and no proble…
4  If I could give a lower rate I would! I cancel…
```

[17]:
```
reviews_df['Rating'].dtype
```

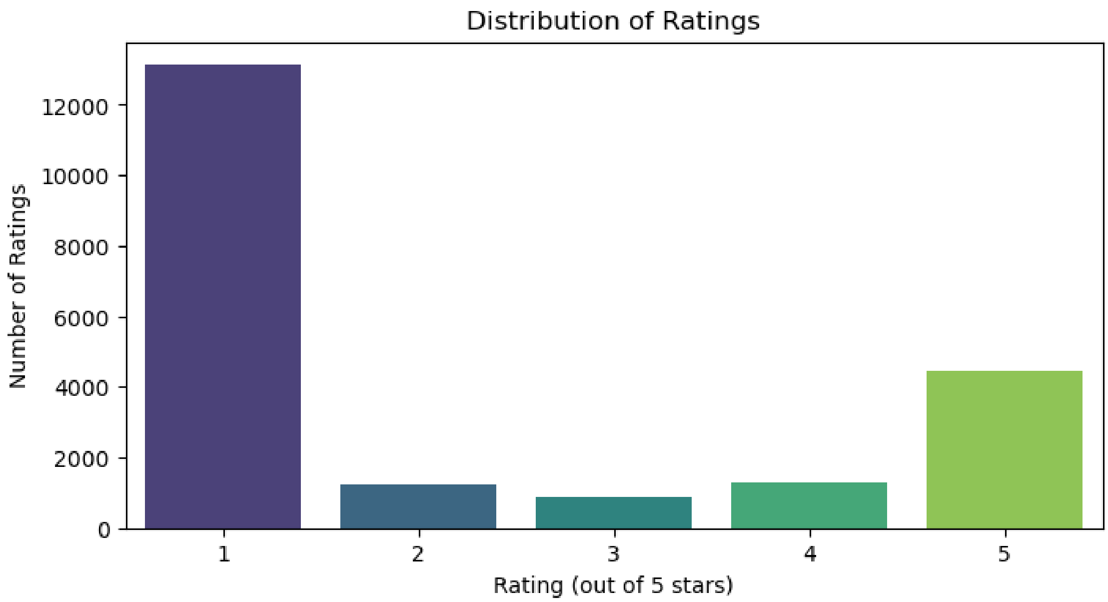
[17]:
```
dtype('int64')
```

[18]:
```
# visualization of rating distribution
```

[19]:
```
plt.figure(figsize = (8,4))
sns.countplot(data = reviews_df, x = 'Rating', order =␣
 ↪sorted(reviews_df['Rating'].unique()), palette='viridis')
plt.title('Distribution of Ratings')
plt.xlabel('Rating (out of 5 stars)')
plt.ylabel('Number of Ratings')
plt.xticks(rotation=0)
plt.show()
```

```
/var/folders/6q/k8jdwbv174s78xj3x3kzvn0w0000gn/T/ipykernel_20486/4122985449.py:2
: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data = reviews_df, x = 'Rating', order =
sorted(reviews_df['Rating'].unique()), palette='viridis')
```



Distribution of Ratings The bar plot shows the distribution of ratings from the customer reviews: Most reviews tend to cluster around lower ratings, particularly 1 and 2 stars. There are fewer reviews with higher ratings (4 and 5 stars), indicating a trend toward negative sentiments in this dataset.

[20]:
```
# analyze trend over time
```

[21]:
```
# convert "Review Date" to datetime and only keep the date
```

[22]:
```
reviews_df['Review Date'] = pd.to_datetime(reviews_df['Review Date']).dt.date
reviews_df.head(5)
```

[22]:

	Review Date	Rating	Review Text
0	2024-09-16	1	I registered on the website, tried to order a …
1	2024-09-16	1	Had multiple orders one turned up and driver h…
2	2024-09-16	1	I informed these reprobates that I WOULD NOT B…
3	2024-09-17	1	I have bought from Amazon before and no proble…
4	2024-09-16	1	If I could give a lower rate I would! I cancel…

[23]:
```
reviews_df['Date-YearMonth'] = pd.to_datetime(reviews_df['Review Date']).dt.
    ↪to_period('M')
reviews_df
```

[23]:
```
      Review Date  Rating                                        Review Text  \
0      2024-09-16       1  I registered on the website, tried to order a …
1      2024-09-16       1  Had multiple orders one turned up and driver h…
2      2024-09-16       1  I informed these reprobates that I WOULD NOT B…
3      2024-09-17       1  I have bought from Amazon before and no proble…
4      2024-09-16       1  If I could give a lower rate I would! I cancel…
…             …       …                                                …
21209  2009-03-22       5  I have had perfect order fulfillment, and fast…
21210  2008-12-31       5  I have had perfect order fulfillment, and fast…
21211  2008-09-16       3  I always find myself going back to amazon beco…
21212  2008-04-28       5  I have placed an abundance of orders with Amaz…
21213  2007-08-27       4  those goods i've ordered by Amazon.com, have b…

      Date-YearMonth
0            2024-09
1            2024-09
2            2024-09
3            2024-09
4            2024-09
…                …
21209        2009-03
21210        2008-12
21211        2008-09
21212        2008-04
21213        2007-08

[20946 rows x 4 columns]
```

[24]:
```
# groupby date and get counts and average rating score for each date
```

[25]:
```
trend = reviews_df.groupby('Review Date').agg(Review_Count = ('Rating',␣
    ↪'size'), Average_Rating = ('Rating', 'mean')).reset_index()
trend
```

[25]:
```
     Review Date  Review_Count  Average_Rating
0     2007-08-27             1        4.000000
1     2008-04-28             1        5.000000
2     2008-09-16             1        3.000000
3     2008-12-31             1        5.000000
4     2009-03-22             1        5.000000
…            …           …             …
3630  2024-09-13            23        1.826087
3631  2024-09-14            12        1.250000
```

```
3632  2024-09-15             4        3.000000
3633  2024-09-16            17        2.000000
3634  2024-09-17             7        1.142857

[3635 rows x 3 columns]
```

[26]:

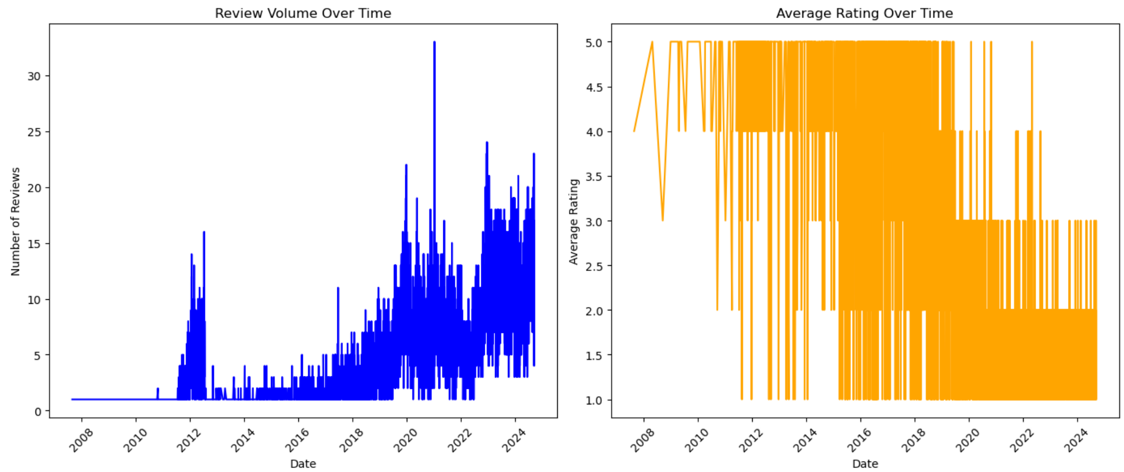
```
trend['Average_Rating'] = trend['Average_Rating'].round().astype(int)
trend.head(5)
```

[26]:

```
  Review Date  Review_Count  Average_Rating
0  2007-08-27             1               4
1  2008-04-28             1               5
2  2008-09-16             1               3
3  2008-12-31             1               5
4  2009-03-22             1               5
```

[27]:

```
plt.figure(figsize = (14, 6))

plt.subplot(1,2,1)
sns.lineplot(data = trend, x = 'Review Date', y = 'Review_Count', color='blue')
plt.title('Review Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Reviews')
plt.xticks(rotation = 45)

plt.subplot(1,2,2)
sns.lineplot(data = trend, x = 'Review Date', y = 'Average_Rating', color =␣
↪'orange')
plt.title('Average Rating Over Time')
plt.xlabel('Date')
plt.ylabel('Average Rating')
plt.xticks(rotation = 45)

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



Trends Over Time Review Volume Over Time: The line plot shows the number of reviews submitted over time. This graph shows an overall upward trend on reviews counts from 2008-2024. A fluctuation in 2012 means there was a significantly higher number of reviews during 2008-2014. The trend starts to go up from 2014 till now, with a peak around end of 2020, when it has the most customer reviews, means the highest customer engagement. Meanwhile, before 2011, there were very low volumns of reviews by customers. Average Rating Over Time: The line plot of average rating over time showing a downward trend in overall rating, which indicating the growing dissatisfaction. Between 2022-2019, there were the period that has higher average ratings, while from 2018 till now, there tends to have more negative reviews than positive reviews. While combing the two graphs, we could see that in some degree, the average ratings correlate with review volume. There is an increase in reviews corresponds but with a drop in ratings, it could indicate that new products or features are not meeting customer expectations in general.

[28]:
```
# process the review text
```

[29]:
```
# convert text into lower case
```

[30]:
```
reviews_df['Review Text'] = reviews_df['Review Text'].str.lower()
```

[31]:
```
# remove punctuation
```

[32]:
```
reviews_df['Cleaned'] = reviews_df['Review Text'].str.replace(r'[^\w\s]', '', regex=True)
```

[33]:
```
# remove stop words
```

[34]:
```
import nltk
from nltk.corpus import stopwords
import re
```

[35]:
```
stop_words = set(stopwords.words('english'))
```

[36]:
```
reviews_df['Review Text no stopwords'] = reviews_df['Cleaned'].apply(
    lambda x: ' '.join([word for word in x.split() if word.lower() not in␣
 ↪stop_words])
)
```

[37]:
```
reviews_df.head(5)
```

[37]:
```
  Review Date  Rating                                       Review Text  \
0  2024-09-16       1  i registered on the website, tried to order a …
1  2024-09-16       1  had multiple orders one turned up and driver h…
2  2024-09-16       1  i informed these reprobates that i would not b…
3  2024-09-17       1  i have bought from amazon before and no proble…
4  2024-09-16       1  if i could give a lower rate i would! i cancel…

  Date-YearMonth                                        Cleaned  \
0        2024-09  i registered on the website tried to order a l…
1        2024-09  had multiple orders one turned up and driver h…
2        2024-09  i informed these reprobates that i would not b…
3        2024-09  i have bought from amazon before and no proble…
4        2024-09  if i could give a lower rate i would i cancell…

                             Review Text no stopwords
0  registered website tried order laptop entered …
1  multiple orders one turned driver phone door n…
2  informed reprobates would going visit sick rel…
3  bought amazon problems happy service price ama…
4  could give lower rate would cancelled amazon p…
```

[38]:
```
# Apply NLTK's PorterStemmer get stemming words
```

[39]:
```
from nltk.stem import PorterStemmer
```

[40]:
```
porter = PorterStemmer()
```

[41]:
```
def stem_words(text): # function to apply stemming
    words = nltk.word_tokenize(text)
    # Apply the PorterStemmer to each word
    stemmed_words = [porter.stem(word) for word in words]
    # Return the stemmed words as a single string
    return ' '.join(stemmed_words)
```

[42]:
```
reviews_df['Review Stemming'] = reviews_df['Review Text no stopwords'].
 ↪apply(stem_words)
```

[43]:
```
reviews_df.head()
```

[43]:
```
  Review Date  Rating                                        Review Text  \
0  2024-09-16       1  i registered on the website, tried to order a ...
1  2024-09-16       1  had multiple orders one turned up and driver h...
2  2024-09-16       1  i informed these reprobates that i would not b...
3  2024-09-17       1  i have bought from amazon before and no proble...
4  2024-09-16       1  if i could give a lower rate i would! i cancel...

  Date-YearMonth                                            Cleaned  \
0        2024-09  i registered on the website tried to order a l...
1        2024-09  had multiple orders one turned up and driver h...
2        2024-09  i informed these reprobates that i would not b...
3        2024-09  i have bought from amazon before and no proble...
4        2024-09  if i could give a lower rate i would i cancell...

                              Review Text no stopwords  \
0  registered website tried order laptop entered ...
1  multiple orders one turned driver phone door n...
2  informed reprobates would going visit sick rel...
3  bought amazon problems happi service price ama...
4  could give lower rate would cancelled amazon p...

                                      Review Stemming
0  regist websit tri order laptop enter detail in...
1  multipl order one turn driver phone door numbe...
2  inform reprob would go visit sick rel told go ...
3  bought amazon problem happi servic price amazo...
4  could give lower rate would cancel amazon prim...
```

[44]:
```python
# drop columns for model building.
```

[45]:
```python
process_columns = ['Review Text', 'Cleaned', 'Review Text no stopwords',␣
↪'Date-YearMonth']
```

[46]:
```python
reviews_df_cleaned = reviews_df.drop(columns = process_columns)
reviews_df_cleaned.head()
```

[46]:
```
  Review Date  Rating                                    Review Stemming
0  2024-09-16       1  regist websit tri order laptop enter detail in...
1  2024-09-16       1  multipl order one turn driver phone door numbe...
2  2024-09-16       1  inform reprob would go visit sick rel told go ...
3  2024-09-17       1  bought amazon problem happi servic price amazo...
4  2024-09-16       1  could give lower rate would cancel amazon prim...
```

[47]:
```python
# word cloud
```

[48]:
```python
from wordcloud import WordCloud
```

```
[49]: # combine all the stemmed words into a string
```

```
[50]: all_reviews = ' '.join(reviews_df['Review Stemming'])
```

```
[51]: # generate the word cloud
```

```
[52]: wordcloud = WordCloud(width=800, height=400, background_color='white',
                            colormap='viridis').generate(all_reviews)
```

```
[53]: plt.figure(figsize=(10, 5))
      plt.imshow(wordcloud, interpolation='bilinear')
      plt.axis('off')
      plt.title('Word Cloud of Reviews')
      plt.show()
```

Word Cloud of Reviews

This word cloud was generated based on the frequency of words in all the reviews. The larger means more frequently occuring terms, and smaller words means less frequent used terms. The most frequently used terms in the reviews are “Amazon”, “custom”, “service”, “order”, “time”, “call”, “package”, “said”, “refund”, “deliver” etc. From the frequent used terms we could see that reviews related to customer service, product, shipping time and also refund is fairly frequent due to some bad reviews.

```
[54]: # Sentiment Analysis Using VADER
```

```
[55]: import pandas as pd
      import nltk
      from nltk.sentiment import SentimentIntensityAnalyzer
```

[56]:
```python
sia = SentimentIntensityAnalyzer()
```

[57]:
```python
def analyze_sentiment(review_text):
    sentiment_scores = sia.polarity_scores(review_text)
    return sentiment_scores['compound']
```

[58]:
```python
reviews_df_cleaned['Sentiment_Score_VADER'] = reviews_df_cleaned['Review␣
↪Stemming'].apply(analyze_sentiment)
```

[59]:
```python
print(reviews_df_cleaned[['Review Stemming', 'Sentiment_Score_VADER']])
```

```
                                         Review Stemming  \
0      regist websit tri order laptop enter detail in…
1      multipl order one turn driver phone door numbe…
2      inform reprob would go visit sick rel told go …
3      bought amazon problem happi servic price amazo…
4      could give lower rate would cancel amazon prim…
…                                                    …
21209  perfect order fulfil fast deliveri amazon prob…
21210  perfect order fulfil fast deliveri amazon help…
21211  alway find go back amazon becous price good ho…
21212  place abund order amazon last coupl year none …
21213  good ive order amazoncom deliv good order even…

       Sentiment_Score_VADER
0                     0.7506
1                     0.1531
2                    -0.6820
3                    -0.7269
4                     0.4939
…                        …
21209                 0.8074
21210                 0.8934
21211                 0.2500
21212                 0.9300
21213                 0.9260

[20946 rows x 2 columns]
```

[60]:
```python
reviews_df_cleaned.head()
```

[60]:

```
  Review Date  Rating                                    Review Stemming  \
0  2024-09-16       1  regist websit tri order laptop enter detail in…
1  2024-09-16       1  multipl order one turn driver phone door numbe…
2  2024-09-16       1  inform reprob would go visit sick rel told go …
3  2024-09-17       1  bought amazon problem happi servic price amazo…
4  2024-09-16       1  could give lower rate would cancel amazon prim…
```

```
   Sentiment_Score_VADER
0                 0.7506
1                 0.1531
2                -0.6820
3                -0.7269
4                 0.4939
```

[61]:
```
# Classify sentiments based on the compound score
```

[62]:
```
def classify_sentiment(score):
    if score >= 0.05:
        return 'Positive'
    elif score <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'
```

[63]:
```
reviews_df_cleaned['Sentiment_VADER'] =␣
↪reviews_df_cleaned['Sentiment_Score_VADER'].apply(classify_sentiment)
```

[64]:
```
# Grouping by sentiment and counting
```

[65]:
```
sentiment_trend = reviews_df_cleaned.groupby(['Review Date',␣
↪'Sentiment_VADER']).size().unstack(fill_value=0)
sentiment_trend
```

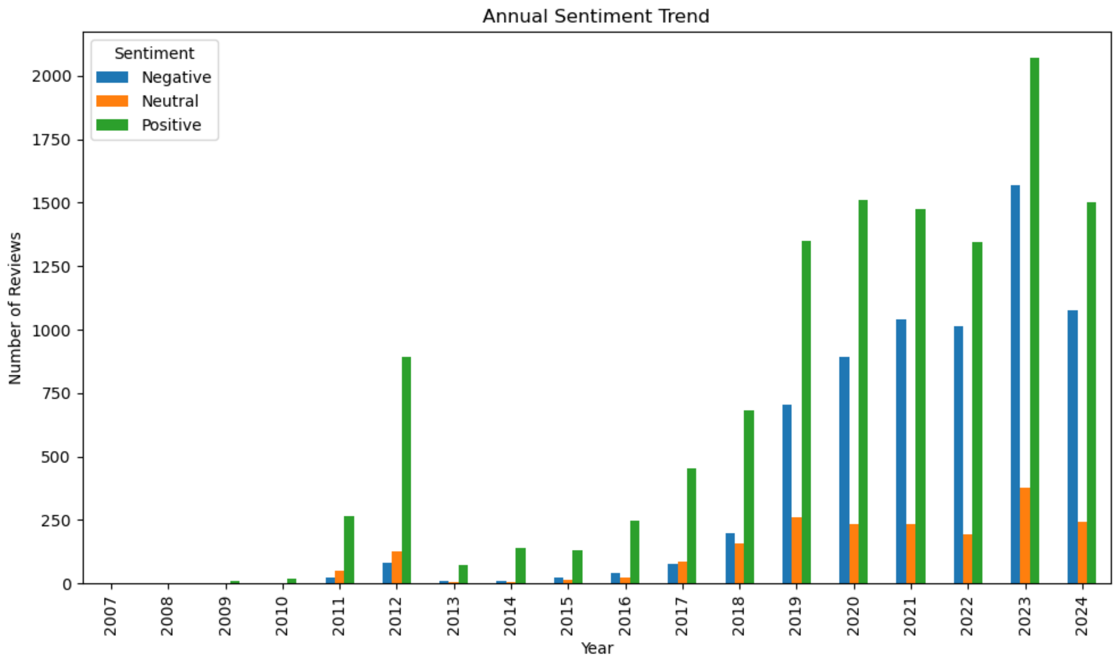
[65]:
```
Sentiment_VADER  Negative  Neutral  Positive
Review Date
2007-08-27              0        0         1
2008-04-28              0        0         1
2008-09-16              0        0         1
2008-12-31              0        0         1
2009-03-22              0        0         1
…                     …       …       …
2024-09-13              9        3        11
2024-09-14              4        1         7
2024-09-15              2        0         2
2024-09-16              4        1        12
2024-09-17              4        0         3

[3635 rows x 3 columns]
```

[66]:
```
sentiment_trend.index = pd.to_datetime(sentiment_trend.index)
```

[67]:
```
sentiment_trend['Year'] = sentiment_trend.index.year
```

[68]:
```
sentiment_trend_over_time = sentiment_trend.groupby('Year').sum()
```

[69]:
```
sentiment_trend_over_time.plot(kind='bar', figsize=(10, 6))
plt.title('Annual Sentiment Trend')
plt.xlabel('Year')
plt.ylabel('Number of Reviews')
plt.legend(title='Sentiment')
plt.tight_layout()
plt.show()
```



The bar graph of review trends over time shows that people are generally postive with Amazon products as the number of positive reviews are always larger than the number of negative reviews. However, recent years sees an increading number of negative feedbacks as compared to the year before 2017. The number of neutral feedbacks are always less than postive and negative feedbacks, and the numbers are fluctuating each year.

[70]:
```
# Perform any data extraction/selection steps and transform features.
```

[71]:
```
# apply TF-IDF vectorizer to the 'Review Stemming' column.
```

[72]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[73]:
```
tfidf = TfidfVectorizer(max_features=1000)
```

[74]:
```
tfidf
```

[74]:
```
TfidfVectorizer(max_features=1000)
```

[75]:
```python
tfidf_df_stemmed = tfidf.fit_transform(reviews_df_cleaned['Review Stemming'])
```

[76]:
```python
tfidf_df_stemmed
```

[76]:
```
<20946x1000 sparse matrix of type '<class 'numpy.float64'>'
        with 576377 stored elements in Compressed Sparse Row format>
```

The TF-IDF matrix represents how important each word is to this dataset. Each row in the matrix corresponds to a review, and each column represents a word (or term) from the reviews, I set the max feature to 500. The value in each cell of the matrix is the TF-IDF score, which indicates the importance of a word in a particular review relative to the entire set of reviews.

[144]:
```python
# split the dataset into training and test set
```

[145]:
```python
from sklearn.model_selection import train_test_split
```

[146]:
```python
X = tfidf_df_stemmed
```

[147]:
```python
y = reviews_df_cleaned['Rating']
```

[148]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 ↪random_state=42)
```

[149]:
```python
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[149]:
```
((16756, 1000), (4190, 1000), (16756,), (4190,))
```

[150]:
```python
# model building
```

[151]:
```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
```

[152]:
```python
nb_model = MultinomialNB()
```

[153]:
```python
nb_model.fit(X_train, y_train)
```

[153]:
```
MultinomialNB()
```

[154]:
```python
y_pred_nb = nb_model.predict(X_test)
```

[155]:
```python
accuracy_score(y_test, y_pred_nb)
```

[155]:
```
0.7773269689737470
```

[156]:
```python
report_dict = classification_report(y_test, y_pred_nb, output_dict=True)
```

```
/opt/anaconda3/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
```

```
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/opt/anaconda3/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[157]:
```
nb_report = pd.DataFrame(report_dict).transpose()
nb_report
```

[157]:

	precision	recall	f1-score	support
1	0.798345	0.983019	0.881109	2650.000000
2	0.000000	0.000000	0.000000	234.000000
3	0.000000	0.000000	0.000000	182.000000
4	1.000000	0.019763	0.038760	253.000000
5	0.701735	0.742824	0.721695	871.000000
accuracy	0.777327	0.777327	0.777327	0.777327
macro avg	0.500016	0.349121	0.328313	4190.000000
weighted avg	0.711176	0.777327	0.709628	4190.000000

For rating 1 and rating 5 the precision, recall and f1-score are both high, indicating that the model performs well in identifying rating 1 and rating 5 instances. For rating 2, 3 and 4, we have zero values for precision, recall, and F1-score, suggesting that the model did not correctly identify any instances of these classes. This is likely due to data imbalance, as these ratings have much lower support (fewer samples). The overall accuracy of 77% indicates that the model is correct in its predictions 77% of the time. However, this metric may be inflated due to the model's high performance on rating 1 and 5 and may not fully reflect its poor performance on other classes.

[91]:
```
# resampling data and do nb model to compare model performance
```

[92]:
```
from imblearn.over_sampling import SMOTE
```

[93]:
```
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

[94]:
```
X_resampled.shape, y_resampled.shape
```

[94]:
```
((65600, 1000), (65600,))
```

[95]:
```
X_train_resampled, X_test_resampled, y_train_resampled, y_test_resampled =␣
↪train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

[96]:
```
nb_model.fit(X_train_resampled, y_train_resampled)
```

[96]:
```
MultinomialNB()
```

[97]:
```
y_pred_nb_resampled = nb_model.predict(X_test_resampled)
```

[98]:
```
accuracy_score(y_test_resampled, y_pred_nb_resampled)
```

[98]:
```
0.583079268292683
```

[99]:
```
report_dict_resampled = classification_report(y_test_resampled,␣
↪y_pred_nb_resampled, output_dict=True)
```

[100]:
```
nb_report_resampled = pd.DataFrame(report_dict_resampled).transpose()
nb_report_resampled
```

[100]:

	precision	recall	f1-score	support
1	0.602644	0.666923	0.633157	2597.000000
2	0.540356	0.597128	0.567325	2646.000000
3	0.599901	0.456678	0.518582	2643.000000
4	0.528532	0.589674	0.557431	2576.000000
5	0.661741	0.606471	0.632901	2658.000000
accuracy	0.583079	0.583079	0.583079	0.583079
macro avg	0.586635	0.583375	0.581879	13120.000000
weighted avg	0.586950	0.583079	0.581879	13120.000000

After resampling dataset, we could see great improvement in the model training process. Rating 2, 3, 4 data had been catpured by the model. The precision, recall and f1-score are all between 60% -80%, which indicates a fairly good performance. The model achieved 70% accuracy, indicating acceptable overall performance. This balanced macro average suggests that the model performs relatively consistently across all classes, which is a good sign for a model trained on imbalanced data. The weighted averages are similar to the macro averages, indicating that the model performs similarly across classes and is not heavily biased toward any particular class.

[101]:
```
# use decisiontree model to train resampled dataset to compare model performance
```

[102]:
```
from sklearn.tree import DecisionTreeClassifier
```

[103]:
```
dt_model = DecisionTreeClassifier(class_weight = {1: 1, 2: 5, 3: 10, 4: 5, 5:␣
↪1}, random_state=42)
```

[104]:
```
dt_model.fit(X_train_resampled, y_train_resampled)
```

[104]:
```
DecisionTreeClassifier(class_weight={1: 1, 2: 5, 3: 10, 4: 5, 5: 1},
                       random_state=42)
```

[105]:
```
y_pred_dt_resampled = dt_model.predict(X_test_resampled)
```

[106]:
```
accuracy_score(y_test_resampled, y_pred_dt_resampled)
```

[106]: 0.7628048780487805

[107]:
```
report_dict_dt = classification_report(y_test_resampled, y_pred_dt_resampled,␣
↪output_dict=True)
```

[108]:
```
dt_report_resampled = pd.DataFrame(report_dict_dt).transpose()
dt_report_resampled
```

[108]:

	precision	recall	f1-score	support
1	0.749093	0.556411	0.638533	2597.000000
2	0.790049	0.846183	0.817153	2646.000000
3	0.716481	0.899735	0.797719	2643.000000
4	0.772711	0.815606	0.793579	2576.000000
5	0.795602	0.694131	0.741410	2658.000000
accuracy	0.762805	0.762805	0.762805	0.762805
macro avg	0.764787	0.762413	0.757679	13120.000000
weighted avg	0.764843	0.762805	0.757908	13120.000000

The decisiontree model was trained using the resampled data, as with more weights on rating 2, 3, 4. The model is correct 76% of the time across all classes, showing moderate overall performance. Macro Precision (0.76), Recall (0.76), and F1-Score (0.75) represents these values indicate that the model performs similarly across all ratings on average, without being biased towards any specific ratings. Weighted Precision (0.76), Recall (0.76) and F1-Score (0.75) suggesting that the model performs reasonably well on classes with more samples.

[109]:
```
# Hyperparameter Tuning to optimize decisiontree performance
```

[110]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[111]:
```
# define param_grid
```

[112]:
```
param_distributions = {
    'max_depth': [5, 10, 15, 20],                   # Limited range of max depths
    'min_samples_split': [2, 5, 10],                # Few options for minimum␣
↪samples to split
    'min_samples_leaf': [1, 2, 4],                  # Few options for minimum␣
↪samples in leaf
    'class_weight': ['balanced', {1: 1, 2: 5, 3: 1, 4: 1, 5: 1}]  # Options for␣
↪class weights
}
```

[113]:
```
# Initialize RandomizedSearch
```

[114]:
```
random_search = RandomizedSearchCV(
    estimator=DecisionTreeClassifier(random_state=42),
    param_distributions=param_distributions,
```

```
    n_iter=10,                # Number of random combinations to try
    cv=3,                     # Number of cross-validation folds
    scoring='f1_weighted',    # Scoring metric
    n_jobs=-1,                # Use all available processors
    random_state=42
)
```

[115]:
```
random_search.fit(X_train_resampled, y_train_resampled)
```

[115]:
```
RandomizedSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=42),
                   n_jobs=-1,
                   param_distributions={'class_weight': ['balanced',
                                                         {1: 1, 2: 5, 3: 1,
                                                          4: 1, 5: 1}],
                                        'max_depth': [5, 10, 15, 20],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10]},
                   random_state=42, scoring='f1_weighted')
```

[116]:
```
y_pred_rs = random_search.predict(X_test_resampled)
```

[117]:
```
report_dict_rs = classification_report(y_test_resampled, y_pred_rs,␣
 ↪output_dict=True)
```

[118]:
```
report_resampled_rs = pd.DataFrame(report_dict_rs).transpose()
report_resampled_rs
```

[118]:

	precision	recall	f1-score	support
1	0.550441	0.672314	0.605304	2597.000000
2	0.681569	0.656841	0.668976	2646.000000
3	0.817949	0.603481	0.694535	2643.000000
4	0.589631	0.626941	0.607714	2576.000000
5	0.536729	0.547028	0.541830	2658.000000
accuracy	0.621037	0.621037	0.621037	0.621037
macro avg	0.635264	0.621321	0.623672	13120.000000
weighted avg	0.635692	0.621037	0.623735	13120.000000

After model tuning, we could see that the model struggles with rating 1, 4, 5 datas, with relatively low precision, recall, and F1-scores. This could be due to class overlap or difficulty in distinguishing these classes. The model performs better on rating 2 and 3, with balanced precision and recall. Class weighting and parameter tuning appear to have helped here. Overall, the hyperparameter tuning by RandomizedSearch seems not perform as well as the DecisionTree model with resampled dataset along with class weight.

[119]:
```
# correlation between rating and reviews
```

[120]:
```
correlation = reviews_df_cleaned[['Rating', 'Sentiment_Score_VADER']].corr().
 ↪iloc[0, 1]
```

[121]:
```
correlation
```

[121]: 0.34663554100608907

A correlation of 0.34 between rating and sentiment score indicates a moderate positive relationship. This means that, generally, as ratings increase, sentiment scores also tend to be more positive. However, the relationship isn't strong, suggesting some variance where the sentiment score does not always match the rating exactly. Customers might give high ratings with neutral language or low ratings with mixed sentiment. For example, someone could rate a product 2 stars but not explicitly express negative sentiment in the text.

Now I would like to examine specific keywords or phrases associated with different ratings to pinpoint what drives satisfaction or dissatisfaction by group reviews by rating.

[122]:
```
# extract top keywords in 1 star, 3 star and 5 star ratings
```

Next I would like to approach to extract common themes or phrases in positive and negative reviews to understand customer feedbacks.

I'll start by classifying each review based on its sentiment score and extracting the top keywords in each ratings.

[160]:
```
def get_top_keywords_by_rating(rating):
    # Filter reviews with the specified rating
    reviews = reviews_df_cleaned[reviews_df_cleaned['Rating'] ==␣
 ↪rating]['Review Stemming']

    tfidf = TfidfVectorizer(max_features=1000, stop_words='english')
    tfidf_matrix = tfidf.fit_transform(reviews)

    # Sum word counts across all reviews for the rating
    word_counts_sum = tfidf_matrix.toarray().sum(axis=0)
    # Create a DataFrame with words and their counts
    keywords_df = pd.DataFrame({'Keyword': tfidf.get_feature_names_out(),␣
 ↪'Count': word_counts_sum})
    # Sort by count in descending order
    top_keywords = keywords_df.sort_values(by='Count', ascending=False)
    return top_keywords
```

[161]:
```
top_keywords_1_star = get_top_keywords_by_rating(1).head(20)
top_keywords_3_star = get_top_keywords_by_rating(3).head(10)
top_keywords_5_star = get_top_keywords_by_rating(5).head(10)
```

[162]:
```
top_keywords_1_star
```

[162]:
```
      Keyword        Count
56     amazon  1024.645530
226    custom   632.862880
615     order   632.310355
```

```
788    servic  606.590626
465      item  556.350467
247  deliveri  523.492347
233       day  472.985617
900      time  450.112768
712    refund  428.871978
34    account  418.774270
246     deliv  418.051876
667     prime  378.054873
944       use  371.319031
275      dont  357.150454
746    review  340.844035
745    return  339.788674
186    compani 336.156890
570     money  333.952105
621    packag  323.300434
672   product  321.505714
```

[126]:
```
top_keywords_3_star
```

[126]:
```
      Keyword      Count
54     amazon  62.126758
733    review  50.804347
875      text  45.545087
232  deliveri  39.051466
596     order  38.582937
371      good  37.451148
777    servic  32.497962
663   product  32.419054
441      item  31.129872
217    custom  29.800525
```

[127]:
```
top_keywords_5_star
```

[127]:
```
      Keyword       Count
52     amazon  336.258152
751    review  230.658597
785    servic  229.864277
890      text  224.519127
50      alway  200.942956
397     great  199.559554
391      good  174.698653
222    custom  164.814465
614     order  158.288855
240  deliveri  154.111553
```

After reviewing the top keyword, I realized that the top keywords overlaps in each rating categories, like “amazon”, “review”, “servic”, so I decided to do feature selection to filter out the most repeated words.

[128]:
```
# do feature selection using Variance Threshold (for TF-IDF features)
```

[129]:
```
from sklearn.feature_selection import VarianceThreshold
```

[130]:
```
reviews_df_cleaned = reviews_df_cleaned.reset_index(drop=True)
```

[131]:
```
selector = VarianceThreshold(threshold=0.001)
```

[163]:
```
custom_stop_words = ['product', 'amazon', 'servic', 'use', 'good', 'buy',␣
↪'review', 'text', 'order']
```

[164]:
```
tfidf = TfidfVectorizer(max_features=2000, stop_words='english').
↪set_params(stop_words=custom_stop_words)
```

[165]:
```
tfidf_matrix = tfidf.fit_transform(reviews_df_cleaned['Review Stemming'])
```

[166]:
```
tfidf_reduced = selector.fit_transform(tfidf_matrix)
```

[167]:
```
tfidf_reduced
```

[167]:
```
<20946x205 sparse matrix of type '<class 'numpy.float64'>'
        with 310727 stored elements in Compressed Sparse Row format>
```

[168]:
```
# get the selected feature names after variance thresholding
```

[169]:
```
selected_features = selector.get_support(indices=True)
reduced_keywords = [tfidf.get_feature_names_out()[i] for i in selected_features]
```

[170]:
```
def get_top_keywords_by_rating_with_variance(rating, top_n=40):
    # Filter rows for reviews with the specified rating
    reviews_indices = reviews_df_cleaned[reviews_df_cleaned['Rating'] ==␣
↪rating].index
    filtered_reviews_matrix = tfidf_reduced[reviews_indices, :]

    # Calculate average TF-IDF scores for each keyword in this rating group
    avg_tfidf_scores = filtered_reviews_matrix.mean(axis=0).A1
    # Create a DataFrame with keywords and their average scores
    keywords_df = pd.DataFrame({'Keyword': reduced_keywords, 'Avg_TFIDF_Score':␣
↪avg_tfidf_scores})
    # Sort by TF-IDF score to get the most significant keywords
    return keywords_df.sort_values(by='Avg_TFIDF_Score', ascending=False).
↪head(top_n)
```

[171]:
```
top_keywords_1_star = get_top_keywords_by_rating_with_variance(1)
top_keywords_3_star = get_top_keywords_by_rating_with_variance(3)
top_keywords_5_star = get_top_keywords_by_rating_with_variance(5)
```

[172]:
```
top_keywords_1_star
```

[172]:

	Keyword	Avg_TFIDF_Score
40	custom	0.045407
85	item	0.038327
47	deliveri	0.034514
70	get	0.034490
43	day	0.032680
1	account	0.030437
147	refund	0.030414
181	time	0.030405
46	deliv	0.029120
52	dont	0.025492
134	prime	0.025140
23	call	0.024066
112	never	0.023936
108	money	0.023850
152	return	0.022951
36	compani	0.022912
122	packag	0.021964
145	receiv	0.021436
202	would	0.021065
119	one	0.020938
154	say	0.020596
58	even	0.020537
14	back	0.020250
25	cancel	0.020124
27	card	0.019710
185	tri	0.019029
183	told	0.018370
37	contact	0.017709
153	said	0.017609
73	go	0.017429
190	wait	0.017371
57	email	0.017195
161	ship	0.016766
74	got	0.016068
26	cant	0.016026
68	found	0.015986
54	driver	0.015774
82	im	0.015727
175	take	0.015694
72	give	0.015545

[173]:
```
top_keywords_3_star
```

[173]:

	Keyword	Avg_TFIDF_Score
68	found	0.072481
47	deliveri	0.042569
85	item	0.031058
70	get	0.030072
181	time	0.029967
43	day	0.028432
161	ship	0.027645
134	prime	0.027123
40	custom	0.026285
97	like	0.025817
133	price	0.023276
122	packag	0.022741
52	dont	0.021901
46	deliv	0.020001
75	great	0.017991
119	one	0.016822
179	thing	0.016747
111	need	0.016501
162	shop	0.016459
64	experi	0.016453
36	compani	0.016294
158	seller	0.015947
143	realli	0.015943
12	arriv	0.015669
104	make	0.015186
167	sometim	0.015074
202	would	0.014953
92	late	0.014650
6	alway	0.014496
66	find	0.014409
82	im	0.014020
15	bad	0.013486
86	ive	0.013407
138	purchas	0.013323
58	even	0.013150
135	problem	0.012929
73	go	0.012749
69	free	0.012737
124	pay	0.012693
204	year	0.012530

[174]:
```
top_keywords_5_star
```

[174]:

	Keyword	Avg_TFIDF_Score
68	found	0.070550
6	alway	0.058957

```
75         great       0.057340
102         love       0.040967
17          best       0.035314
65          fast       0.032489
133        price       0.032464
162         shop       0.030878
47      deliveri       0.030869
181         time       0.029365
40        custom       0.029206
62         excel       0.026593
55          easi       0.025679
135      problem       0.025288
112        never       0.023056
134        prime       0.022636
36       compani       0.022268
85          item       0.022008
61       everyth       0.021863
161         ship       0.021425
64        experi       0.019350
120        onlin       0.018869
204         year       0.018577
7           amaz       0.018504
178        thank       0.018262
84          issu       0.017951
165         site       0.017829
152       return       0.016713
66          find       0.016470
43           day       0.015742
86           ive       0.015740
179        thing       0.015472
70           get       0.015449
111         need       0.015357
141        quick       0.014910
69          free       0.014450
146    recommend       0.014290
143       realli       0.013936
105         mani       0.013919
78          help       0.013899
```

After feature selection, the keywords differs in each category. We could use the insights to address specific issues, improve refund processes. For example, “refund” and “package” appear frequently in 1-star reviews, improve packaging and shipping would be considered to improve customer satisfaction. The top keywords for 5 star reviews tends to show that people enjoyed the quick delivery and the service amazon has offered, since “quick”, “free”, “easi” etc. has been frequently appeared.

[ ]:

[ ]:

```
[ ]:
```