BA data cleaning

September 17, 2024

[15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[27]:
```
df = pd.read_csv("C:/Users/bendh/Desktop/data science/JN/BA_2.csv", index_col=0)
```

[28]:
```
df.head()
```

[28]:

	reviews	rating	date	country
0	Not Verified | A nightmare journey courtesy o…	1.0	8th September 2024	United Kingdom
1	Trip Verified | Absolutely atrocious. LHR-OR…	1.0	6th September 2024	United Kingdom
2	Trip Verified | As someone who flies relentl…	4.0	2nd September 2024	United Kingdom
3	Trip Verified | Flew with British Airways …	2.0	1st September 2024	United Kingdom
4	Trip Verified | Straightforward check in T…	8.0	30th August 2024	United Kingdom

[123]:
```
df.tail()
```

[123]:

	reviews	rating	date	country	corpus
3852	Flew LHR - VIE return operated by bmi but BA a…	10	2012-08-29	United Kingdom	flew lhr vie return operated bmi ba aircraft a…
3853	LHR to HAM. Purser addresses all club passenge…	9	2012-08-28	United Kingdom	lhr ham purser address club passenger name boa…
3854	My son who had worked for British Airways urge…	5	2011-10-12	United Kingdom	son worked british airway urged fly british ai…
3855	London City-New York JFK via Shannon on A318 b…	4	2011-10-11	United States	london city new york jfk via shannon really ni…
3856	SIN-LHR BA12 B747-436 First Class. Old aircraf…	4	2011-10-09	United Kingdom	sin lhr ba b first class old aircraft seat pri…

```
      compound  positive  neutral  negative sentiment_category year_month
3852    0.9840     0.339    0.617     0.044           Positive    2012-08
3853    0.8720     0.292    0.708     0.000           Positive    2012-08
3854    0.4516     0.124    0.807     0.068            Neutral    2011-10
3855    0.9148     0.304    0.696     0.000            Neutral    2011-10
3856    0.8096     0.148    0.678     0.174            Neutral    2011-10
```

[29]:
```
df.dtypes
```

[29]:
```
reviews     object
rating     float64
date        object
country     object
dtype: object
```

[30]:
```
df.isna().sum()
```

[30]:
```
reviews    0
rating     5
date       0
country    2
dtype: int64
```

[31]:
```
df[df['rating'].isna() == 1]
```

[31]:
```
                                                reviews  rating  \
3276  Cabin crew polite unfortunately BA ran out of …      NaN
3409  Phoenix to London - outbound a wonderful and e…      NaN
3433  On past experience I chose BA for our long hau…      NaN
3662  LHR-CPH-LHR Business Class. This is a joke. Sc…      NaN
3696  I flew with British Airways with my mother fro…      NaN

                     date         country
3276  18th February 2015  United Kingdom
3409  10th December 2014   United States
3433  25th November 2014  United Kingdom
3662      31st July 2014  United Kingdom
3696      15th July 2014           Italy
```

[32]:
```
#refilling na ratings based on total rating in the website
```

[33]:
```
df.loc[3276, 'rating'] = 5
```

[35]:
```
print(df.loc[3276])
```

```
reviews    Cabin crew polite unfortunately BA ran out of …
rating                                                 5.0
date                                    18th February 2015
```

```
country                                                 United Kingdom
Name: 3276, dtype: object
```

[36]:
```
df.loc[3409, 'rating'] = 6
```

[37]:
```
df.loc[3433, 'rating'] = 5
```

[38]:
```
df.loc[3662, 'rating'] = 1
```

[39]:
```
df.loc[3696, 'rating'] = 3
```

[40]:
```
df[df['rating'].isna() == 1]
```

[40]:
```
Empty DataFrame
Columns: [reviews, rating, date, country]
Index: []
```

[41]:
```
#refilling na countries based on the review text in the website
```

[42]:
```
df[df['country'].isna() == 1]
```

[42]:
```
                                                reviews  rating  \
3215  I travelled from London to Jo'burg and back on…      2.0
3519  St Lucia to London round trip. Full flight bot…      6.0

                    date country
3215     8th April 2015     NaN
3519  20th October 2014     NaN
```

[43]:
```
df.loc[3519, 'country'] = 'Saint Lucia'
```

[44]:
```
df.loc[3215, 'country'] = 'United Kingdom'
```

[45]:
```
df[df['country'].isna() == 1]
```

[45]:
```
Empty DataFrame
Columns: [reviews, rating, date, country]
Index: []
```

[46]:
```
df.isna().sum()
```

[46]:
```
reviews    0
rating     0
date       0
country    0
dtype: int64
```

[47]:
```
#converting rating from float to integers
```

[48]:
```
df['rating'] = df['rating'].astype(int)
```

[49]:
```
df.dtypes
```

[49]:
```
reviews    object
rating      int32
date       object
country    object
dtype: object
```

[50]:
```
df.head()
```

[50]:

	reviews	rating	\
0	Not Verified | A nightmare journey courtesy o…	1	
1	Trip Verified | Absolutely atrocious. LHR-OR…	1	
2	Trip Verified | As someone who flies relentl…	4	
3	Trip Verified | Flew with British Airways …	2	
4	Trip Verified | Straightforward check in T…	8	

	date	country
0	8th September 2024	United Kingdom
1	6th September 2024	United Kingdom
2	2nd September 2024	United Kingdom
3	1st September 2024	United Kingdom
4	30th August 2024	United Kingdom

[51]:
```
#convert the dates to a date format
```

[52]:
```
from datetime import datetime
import re
```

[53]:
```
# Function to clean and convert date
def convert_date(date_string):
    date_string_clean = re.sub(r'(\d+)(st|nd|rd|th)', r'\1', date_string)
    return pd.to_datetime(date_string_clean, format='%d %B %Y')
```

[54]:
```
# Apply function to the DataFrame column
df['date'] = df['date'].apply(convert_date)
```

[55]:
```
df.head()
```

[55]:

	reviews	rating	date	\
0	Not Verified | A nightmare journey courtesy o…	1	2024-09-08	
1	Trip Verified | Absolutely atrocious. LHR-OR…	1	2024-09-06	
2	Trip Verified | As someone who flies relentl…	4	2024-09-02	
3	Trip Verified | Flew with British Airways …	2	2024-09-01	
4	Trip Verified | Straightforward check in T…	8	2024-08-30	

```
          country
0  United Kingdom
1  United Kingdom
2  United Kingdom
3  United Kingdom
4  United Kingdom
```

[56]:
```
df['date'] = df['date'].dt.strftime('%d-%m-%Y')
```

[57]:
```
df.head()
```

[57]:
```
                                              reviews  rating        date  \
0  Not Verified |  A nightmare journey courtesy o…       1  08-09-2024
1    Trip Verified | Absolutely atrocious. LHR-OR…       1  06-09-2024
2    Trip Verified | As someone who flies relentl…       4  02-09-2024
3    Trip Verified |   Flew with British Airways …       2  01-09-2024
4    Trip Verified |   Straightforward check in T…       8  30-08-2024

          country
0  United Kingdom
1  United Kingdom
2  United Kingdom
3  United Kingdom
4  United Kingdom
```

[58]:
```
#cleaning reviews
```

[59]:
```
df.dtypes
```

[59]:
```
reviews    object
rating      int32
date       object
country    object
dtype: object
```

[60]:
```
def clean_review(review):
    if pd.isna(review):
        return review
    # Define patterns to remove
    patterns = [r'Not Verified', r'  Trip Verified',r'Verified Review', r'|']

    # Remove patterns using regular expressions
    for pattern in patterns:
        review = re.sub(pattern, '', review)

    # Remove extra spaces that may result from removal
    review = ' '.join(review.split())
```

```
    return review
```

[61]:
```
df.head()
```

[61]:
```
                                              reviews  rating        date  \
0  Not Verified |  A nightmare journey courtesy o…       1  08-09-2024
1    Trip Verified | Absolutely atrocious. LHR-OR…       1  06-09-2024
2    Trip Verified | As someone who flies relentl…       4  02-09-2024
3    Trip Verified |   Flew with British Airways …       2  01-09-2024
4    Trip Verified |   Straightforward check in T…       8  30-08-2024

          country
0  United Kingdom
1  United Kingdom
2  United Kingdom
3  United Kingdom
4  United Kingdom
```

[62]:
```
df['reviews'] = df['reviews'].apply(clean_review)
```

[63]:
```
df.head()
```

[63]:
```
                                              reviews  rating        date  \
0  | A nightmare journey courtesy of British Airw…       1  08-09-2024
1  | Absolutely atrocious. LHR-ORD-LHR Round-trip…       1  06-09-2024
2  | As someone who flies relentlessly with Briti…       4  02-09-2024
3  | Flew with British Airways club Europe on Sat…       2  01-09-2024
4  | Straightforward check in T5. New site for cl…       8  30-08-2024

          country
0  United Kingdom
1  United Kingdom
2  United Kingdom
3  United Kingdom
4  United Kingdom
```

[64]:
```
df['reviews'] = df['reviews'].str.strip('| ').str.lstrip()
```

[65]:
```
reviews_data = df['reviews']
```

[66]:
```
df.head()
```

[66]:
```
                                              reviews  rating        date  \
0  A nightmare journey courtesy of British Airway…       1  08-09-2024
1  Absolutely atrocious. LHR-ORD-LHR Round-trip. …       1  06-09-2024
2  As someone who flies relentlessly with British…       4  02-09-2024
3  Flew with British Airways club Europe on Satur…       2  01-09-2024
4  Straightforward check in T5. New site for club…       8  30-08-2024
```

```
          country
0  United Kingdom
1  United Kingdom
2  United Kingdom
3  United Kingdom
4  United Kingdom
```

[67]:
```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

[43]:
```
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\bendh\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\bendh\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[43]:
```
True
```

[44]:
```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\bendh\AppData\Roaming\nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[44]:
```
True
```

[68]:
```
#loop through each review : remove punctuations, small case it , join it and␣
 ↪add it to corp
# Initialize lemmatizer and stopwords
lemma = WordNetLemmatizer()
stop_words = set(stopwords.words("english"))

# Initialize the corpus list
corpus = []

# Loop through each review
for rev in reviews_data:
    # Remove punctuation and keep only alphabetic characters
    rev = re.sub('[^a-zA-Z]', ' ', rev)

    # Convert all characters to lowercase
```

```
    rev = rev.lower()

    # Split the review into a list of words
    rev = rev.split()

    # Lemmatize each word and remove stopwords
    rev = [lemma.lemmatize(word) for word in rev if word not in stop_words]

    # Join the processed words back into a single string
    rev = " ".join(rev)

    # Add the processed review to the corpus
    corpus.append(rev)

# Ensure corpus matches the length of reviews_data
if len(corpus) != len(reviews_data):
    raise ValueError(f"Length mismatch: corpus length is {len(corpus)}, but␣
 ↪reviews_data length is {len(reviews_data)}")

# Add the corpus to the DataFrame
df['corpus'] = corpus
```

[69]:
```
print(len(corpus))  # Should match the number of rows in df
print(len(df['reviews']))
```

```
3857
3857
```

[70]:
```
df['corpus'] = corpus
```

[71]:
```
print(df.loc[0, 'corpus'])
```

```
nightmare journey courtesy british airway worst flight experience year flying
booked ba scheduled depart heathrow sept boarded hr late despite plane gate wait
crew arrive approximately hr flight pilot advised weather radar working would
returning heathrow circled approx min landing member staff present offer advice
alternative flight customer service desk stated would deal await email email
finally arrived hour husband fortunate enough allocated seat flight time boarded
approx hr late plane gate wait crew result around reached venice long wait taxi
meant reach hotel beyond mestre since learnt ba cancelled hundred flight past
several day claiming bad weather yet according flightradar airline flying
heathrow cancelled flight
```

[72]:
```
print(df.loc[0, 'reviews'])
```

```
A nightmare journey courtesy of British Airways. Our worst flight experience in
over 50 years of flying! We were booked on BA470 scheduled to depart Heathrow at
12.05 on 5 Sept - we boarded 1.5 hrs late because despite the plane being at the
gate we had to wait for a crew to arrive. Approximately 1hr into the flight the
```

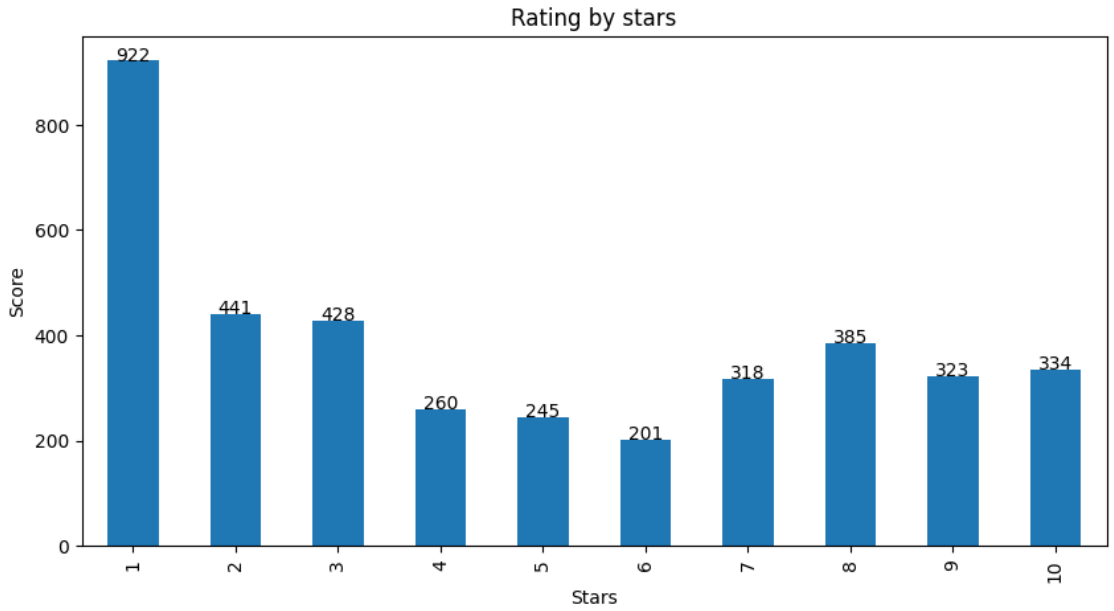
pilot advised that as the weather radar wasn't working, we would be returning to Heathrow where we circled for approx 45 mins before landing. There was no member of staff present to offer advice about an alternative flights and the customer service desk stated they would not deal with it but we had to await an email. An email finally arrived after about an hour and my husband and I were fortunate enough to be allocated seats on the 21.15 flight. This time we boarded approx 2 hrs late again the plane was at the gate but we had to wait for a crew. As a result it was around 3.00 am before we reached Venice and the long wait for taxis meant that we didn't reach out hotel beyond Mestre until 4.00 am. We've since learnt that BA have cancelled hundreds of flights over the past several days claiming 'bad weather' yet according to FlightRadar 24 no other airline flying out of Heathrow cancelled flights

[73]:
```
#see which rating is the most and which one is the least
```

[74]:
```
df['rating'].value_counts()
```

[74]:
```
rating
1     922
2     441
3     428
8     385
10    334
9     323
7     318
4     260
5     245
6     201
Name: count, dtype: int64
```

[75]:
```
counted_values = df['rating'].value_counts().sort_index()
ax = df['rating'].value_counts().sort_index() \
                .plot(kind= 'bar',
                      title= 'Rating by stars',
                      figsize=(10,5))
ax.set_xlabel('Stars')
ax.set_ylabel('Score')
# Annotate each bar with the count value
for index, value in enumerate(counted_values):
    ax.text(index, value + 0.1, str(value), ha='center')
plt.show()
```



[76]:
```
# Calculate the average rating
average_rating = df['rating'].mean()

print(f"The average rating is: {average_rating:.2f}")
```

```
The average rating is: 4.70
```

[77]:
```
# Get unique countries
unique_countries = df['country'].unique()

# Count the number of unique countries
unique_countries_count = len(unique_countries)

print(f"The number of countries is: {unique_countries_count}")
```

```
The number of countries is: 74
```

[78]:
```
# Calculate the sum of the ratings
total_reviews = df['reviews'].count()

print(f"The total number of reviews is: {total_reviews}")
```

```
The total number of reviews is: 3857
```

[79]:
```
df.shape
```

[79]: (3857, 5)

[80]:
```
# Initialize the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()
```

[81]:
```
# Create lists to store sentiment scores
compound_scores = []
positive_scores = []
neutral_scores = []
negative_scores = []
```

[82]:
```
# Loop through the processed reviews
for review in df['corpus']:
    # Get the sentiment scores
    sentiment = analyzer.polarity_scores(review)

    # Append the scores to their respective lists
    compound_scores.append(sentiment['compound'])
    positive_scores.append(sentiment['pos'])
    neutral_scores.append(sentiment['neu'])
    negative_scores.append(sentiment['neg'])
```

[83]:
```
# Add the scores to the DataFrame
df['compound'] = compound_scores
df['positive'] = positive_scores
df['neutral'] = neutral_scores
df['negative'] = negative_scores
```

[84]:
```
def categorize_sentiment(compound_score, rating):
    # First, determine sentiment based on compound score
    if compound_score >= 0.05:
        sentiment = 'Positive'
    elif compound_score <= -0.05:
        sentiment = 'Negative'
    else:
        sentiment = 'Neutral'

   # Adjust sentiment based on the rating
    if rating in [4, 5]:
        sentiment = 'Neutral'
    elif rating <= 3 and sentiment in ['Neutral', 'Positive']:
        sentiment = 'Negative'
    return sentiment
```

[85]:
```
# Apply the function to categorize sentiment
df['sentiment_category'] = df.apply(lambda row:␣
 ↪categorize_sentiment(row['compound'], row['rating']), axis=1)
```

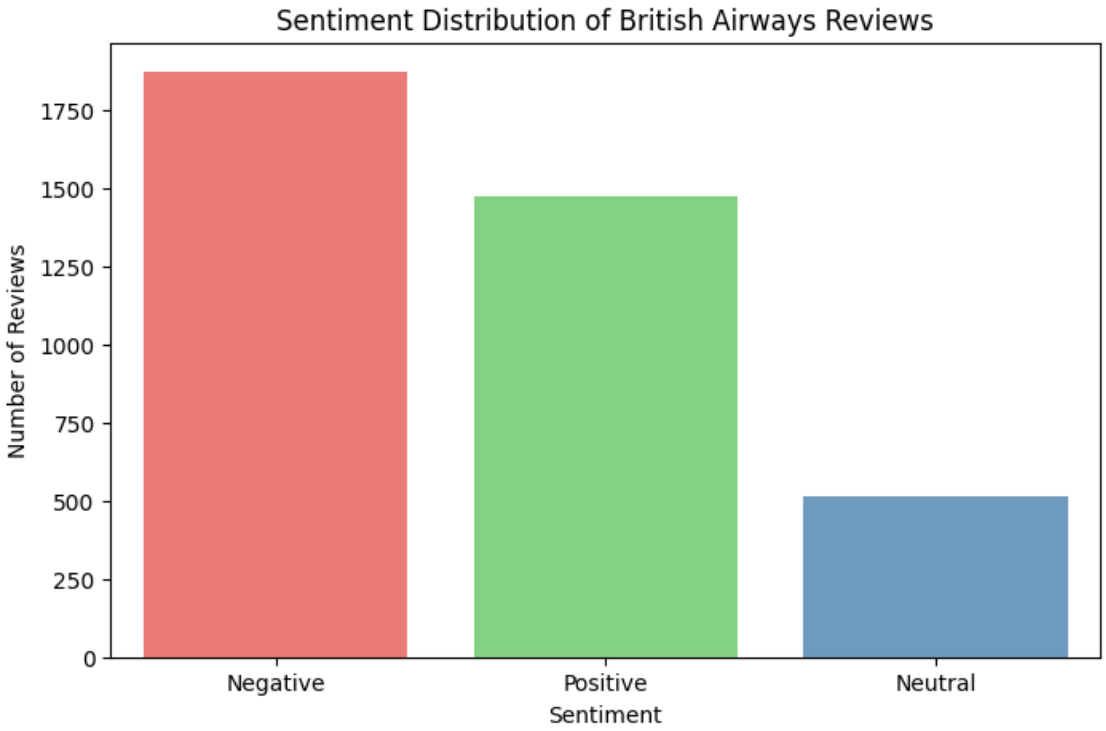
[86]:
```
# Count the number of each sentiment category
sentiment_counts = df['sentiment_category'].value_counts()
```

[87]:
```
sentiment_counts
```

[87]:
```
sentiment_category
Negative    1871
Positive    1472
Neutral      514
Name: count, dtype: int64
```

[88]:
```
# Define a custom color palette
custom_palette = {
    'Positive': '#77DD77',  # Pastel green
    'Neutral': '#5F9BCC',   # Pastel yellow
    'Negative': '#FF6961'   # Pastel red
}

# Plot the sentiment distribution
plt.figure(figsize=(8, 5))
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values,␣
 ↪palette=custom_palette)
plt.title('Sentiment Distribution of British Airways Reviews')
plt.xlabel('Sentiment')
plt.ylabel('Number of Reviews')
plt.show()
```

```
C:\Users\bendh\AppData\Local\Temp\ipykernel_2696\1115212587.py:10:
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.barplot(x=sentiment_counts.index, y=sentiment_counts.values,
palette=custom_palette)
```



[89]:
```
# Show sentiment analysis results for the first review
first_review = df.loc[0, 'reviews']
first_sentiment = df.loc[0, ['compound', 'positive', 'neutral', 'negative',␣
  ↪'sentiment_category']]
print(f"Review: {first_review}\n")
print(f"Sentiment Analysis:\n{first_sentiment}")
```

```
Review: A nightmare journey courtesy of British Airways. Our worst flight
experience in over 50 years of flying! We were booked on BA470 scheduled to
depart Heathrow at 12.05 on 5 Sept - we boarded 1.5 hrs late because despite the
plane being at the gate we had to wait for a crew to arrive. Approximately 1hr
into the flight the pilot advised that as the weather radar wasn't working, we
would be returning to Heathrow where we circled for approx 45 mins before
landing. There was no member of staff present to offer advice about an
alternative flights and the customer service desk stated they would not deal
with it but we had to await an email. An email finally arrived after about an
hour and my husband and I were fortunate enough to be allocated seats on the
21.15 flight. This time we boarded approx 2 hrs late again the plane was at the
gate but we had to wait for a crew. As a result it was around 3.00 am before we
reached Venice and the long wait for taxis meant that we didn't reach out hotel
beyond Mestre until 4.00 am. We've since learnt that BA have cancelled hundreds
of flights over the past several days claiming 'bad weather' yet according to
FlightRadar 24 no other airline flying out of Heathrow cancelled flights
```

```
Sentiment Analysis:
compound               -0.6486
positive                 0.079
neutral                  0.823
negative                 0.098
sentiment_category    Negative
Name: 0, dtype: object
```

[94]:
```
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\bendh\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[94]: True

[95]:
```
from wordcloud import WordCloud
```

[96]:
```
# Initialize a dictionary to store all negative words and their cumulative␣
 ↪scores
all_negative_words = {}
```

[97]:
```
# Loop through each review in the corpus
for review in df['corpus']:
    # Tokenize the review
    review_tokens = word_tokenize(review)

    # Loop through each word in the review
    for word in review_tokens:
        # Get the sentiment score for the word
        sentiment = analyzer.polarity_scores(word)

        # If the word has a negative sentiment, add it to the dictionary
        if sentiment['neg'] > 0:
            if word in all_negative_words:
                all_negative_words[word] += sentiment['neg']
            else:
                all_negative_words[word] = sentiment['neg']
```

[98]:
```
# Print the number of unique negative words found
print(f"Total unique negative words found: {len(all_negative_words)}")
```

```
Total unique negative words found: 917
```

[99]:
```
import matplotlib.colors as mcolors
# Define a custom colormap for red shades
red_colors = ["#660000", "#cc0000", "#ff4d4d", "#ff9999"]  # Dark red to␣
 ↪lighter red
```

```
custom_red_colormap = mcolors.LinearSegmentedColormap.from_list('custom_red',␣
 ↪red_colors, N=256)
# Generate a word cloud for all negative words
wordcloud = WordCloud(width=800, height=400, background_color='white',␣
 ↪colormap=custom_red_colormap).generate_from_frequencies(all_negative_words)
```

[100]:
```
# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Negative Words in All Reviews')
plt.show()
```

[101]:
```
# Initialize a dictionary to store all negative words and their cumulative␣
 ↪scores
all_positive_words = {}
```

[102]:
```
# Loop through each review in the corpus
for review in df['corpus']:
    # Tokenize the review
    review_tokens = word_tokenize(review)

    # Loop through each word in the review
    for word in review_tokens:
        # Get the sentiment score for the word
        sentiment = analyzer.polarity_scores(word)
```

```
        # If the word has a positive sentiment, add it to the dictionary
        if sentiment['pos'] > 0:
            if word in all_positive_words:
                all_positive_words[word] += sentiment['pos']
            else:
                all_positive_words[word] = sentiment['pos']
```

[103]:
```
# Print the number of unique positive words found
print(f"Total unique positive words found: {len(all_positive_words)}")
```

```
Total unique positive words found: 767
```

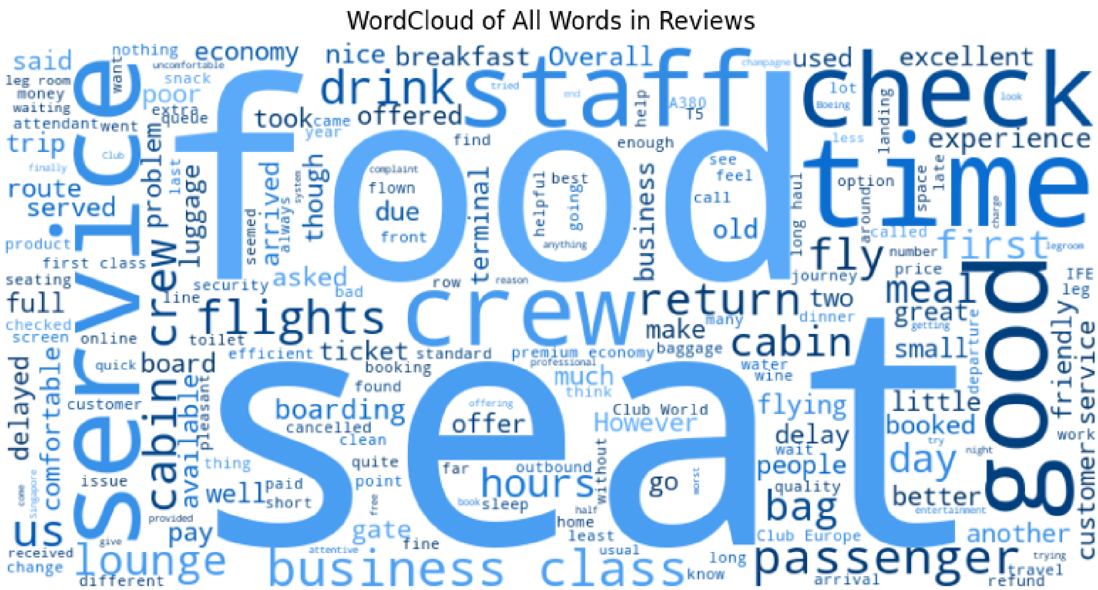
[104]:
```
# Define a custom colormap
colors = ["#004d00", "#006400", "#009900", "#66ff66"]  # Dark green to light␣
 ↪green
custom_colormap = mcolors.LinearSegmentedColormap.from_list('custom_green',␣
 ↪colors, N=256)
# Generate a word cloud for all positive words
wordcloud = WordCloud(width=800, height=400, background_color='white',␣
 ↪colormap=custom_colormap).generate_from_frequencies(all_positive_words)
```

[105]:
```
# Plot the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Positive Words in All Reviews')
plt.show()
```

[106]:
```
# Combine all reviews into a single text
all_reviews = ' '.join(df['reviews'])
stopwords_set = set(word.lower() for word in stopwords.words('english'))
stopwords_set .update(['ba',"british","airway", "airline", "london",␣
 ↪"heathrow", "plane", "choice", "even", "airline","aircraft", "took" \
                 "able" , "ok" , "really", "never", "Airways",".hour", "hour",␣
 ↪"minute", "get", "one", "would", "Gatwick", "also", "back", "way"\
                     "flew" , "made" , "like" , "airport", "could", "despite",␣
 ↪"given", "although", "told", "u", "airlines", "seem", "got", "quite" \
                     "used", "flew", "wife", "able", "say", "next", "take",␣
 ↪"bit", "via" , "minutes", "still", "need", "way", "lhr", "make"\
                     "called", "felt", "rather", "since", "ask", "use",␣
 ↪"left", "flight", "new", "york", "put", "jfk" ])
```

[107]:
```
# Define a custom colormap for blue shades
blue_colors = ["#003366", "#004080", "#0066cc", "#66b3ff"]  # Dark blue to␣
 ↪lighter blue
custom_blue_colormap = mcolors.LinearSegmentedColormap.from_list('custom_blue',␣
 ↪blue_colors, N=256)
# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white',␣
 ↪colormap=custom_blue_colormap, stopwords = stopwords_set ).
 ↪generate(all_reviews)
```

[108]:
```
# Display the WordCloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Remove axis
plt.title('WordCloud of All Words in Reviews')
plt.show()
```

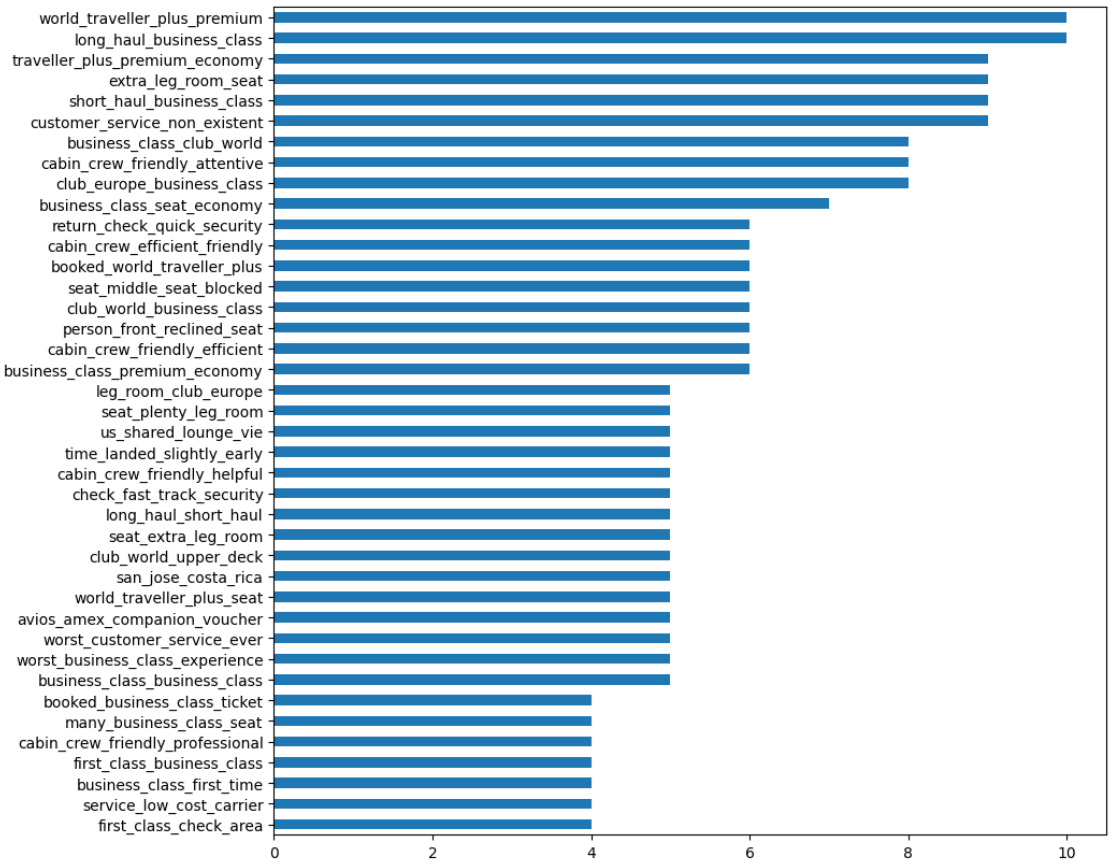


[109]:
```python
import nltk.collocations as collocations
from nltk import FreqDist, bigrams
```

[110]:
```python
reviews = " ".join(df.corpus)
# Clean the text: remove special characters and make everything lowercase
reviews_cleaned = re.sub(r'[^\w\s]', '', reviews.lower())
#split the text of all reviews into a list of words
# Convert all words to lowercase before filtering
words = reviews_cleaned.lower().split(" ")

new_words = [word for word in words if word not in stopwords_set]

def get_freq_dist(new_words,number_of_ngrams):
    from nltk import ngrams
    ## generate bigrams
    ngrams = ngrams(new_words, number_of_ngrams)
    #creating FreqDist
    ngram_fd = FreqDist(ngrams).most_common(40)
    #sort values bu highest frequency
    ngram_sorted = {k:v for k,v in sorted(ngram_fd, key=lambda item:item[1])}
    #join bigram tokens with '_' + maintain sorting
    ngram_joined = {'_'.join(k):v for k,v in sorted (ngram_fd, key=lambda item:
 ↪item[1])}
    #convert to pandas series for easy plotting
    ngram_freqdist = pd.Series(ngram_joined)
    plt.figure(figsize=(10,10))
    ax = ngram_freqdist.plot(kind="barh")
```

```
    return ax

get_freq_dist(new_words,4)
```

[110]: <Axes: >

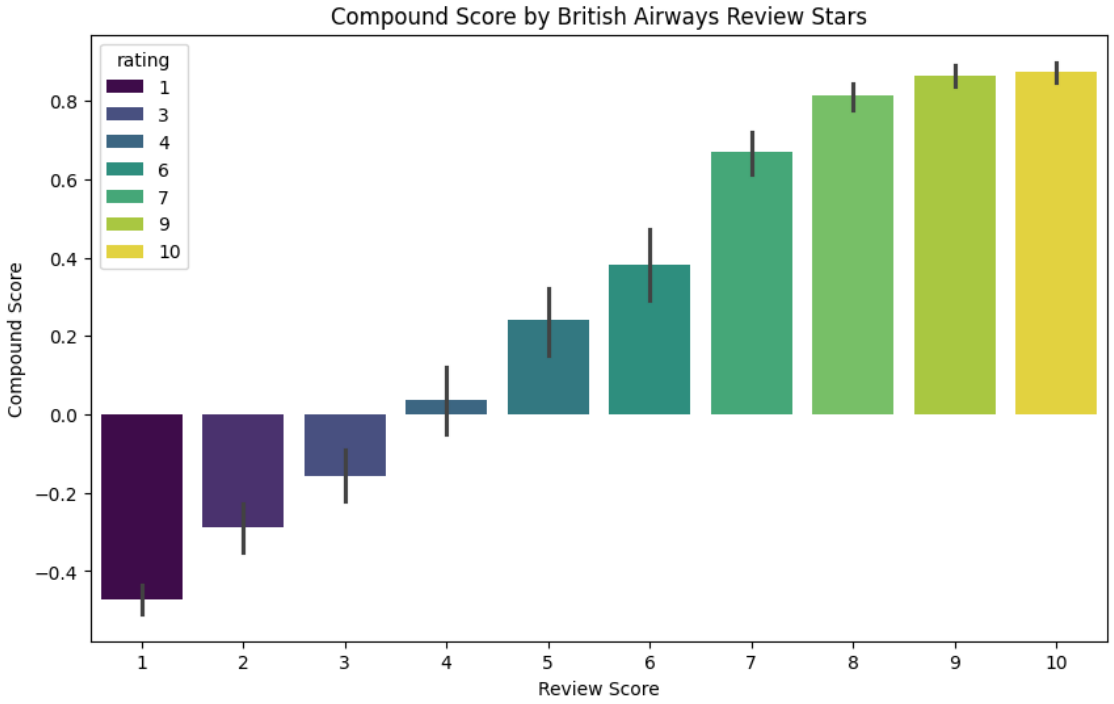


[112]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

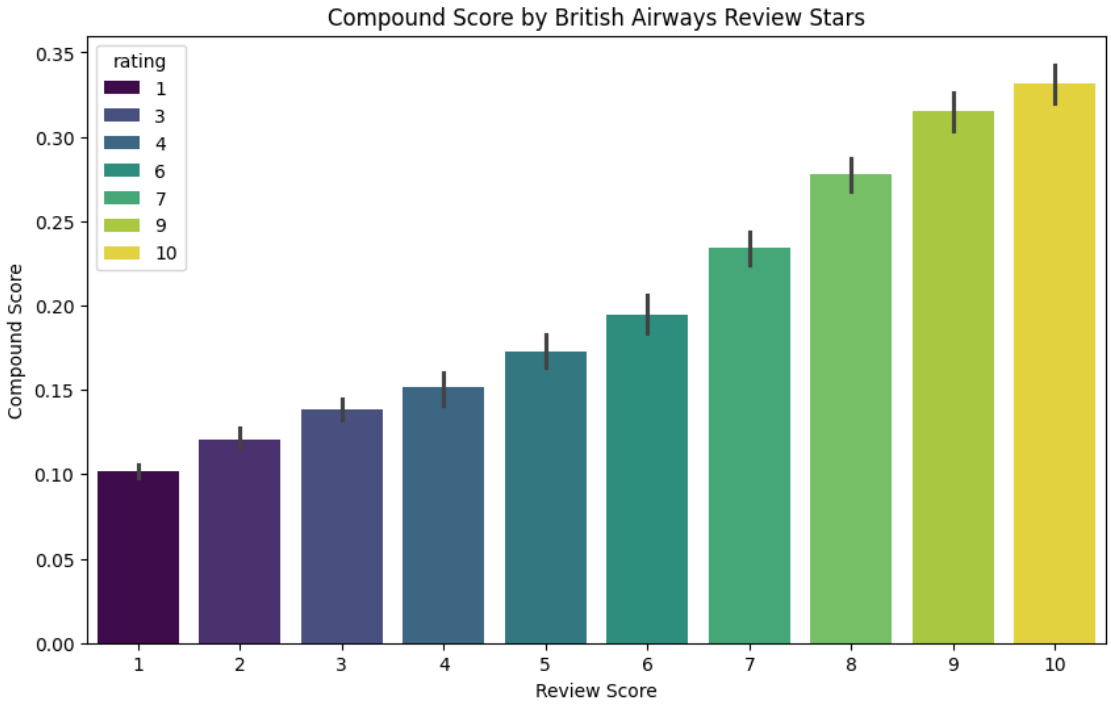
[113]:
```
# Function to get compound sentiment score
def get_compound_score(text):
    return analyzer.polarity_scores(text)['compound']

# Apply the function to get compound scores
df['compound'] = df['reviews'].apply(get_compound_score)
```

[115]:
```
# Plot the barplot
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df, x='rating', y='compound', hue='rating',␣
 ↪palette='viridis')
ax.set_title('Compound Score by British Airways Review Stars')
```

```
ax.set_xlabel('Review Score')
ax.set_ylabel('Compound Score')
plt.legend(title='rating')
plt.show()
```



[116]:
```
# Plot the barplot
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df, x='rating', y='positive', hue='rating',␣
 ↪palette='viridis')
ax.set_title('Compound Score by British Airways Review Stars')
ax.set_xlabel('Review Score')
ax.set_ylabel('Compound Score')
plt.legend(title='rating')
plt.show()
```



[117]:
```
print(df.columns)
```

```
Index(['reviews', 'rating', 'date', 'country', 'corpus', 'compound',
       'positive', 'neutral', 'negative', 'sentiment_category'],
      dtype='object')
```

[118]:
```
# Convert 'date' column to datetime format
df['date'] = pd.to_datetime(df['date'], dayfirst=True, errors='coerce')
```

[119]:
```
df.head()
```

[119]:

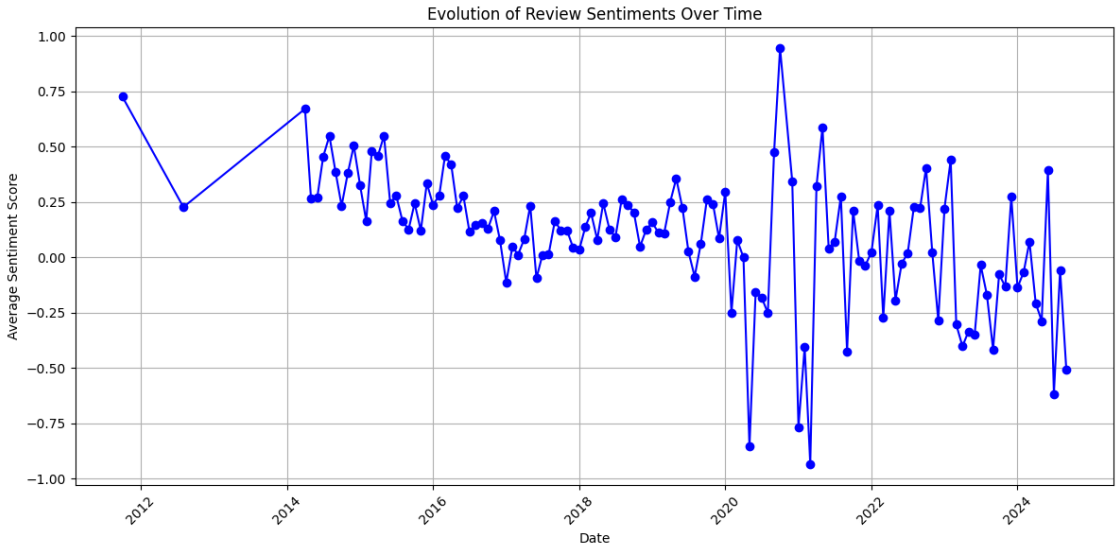
	reviews	rating	date	country	corpus
0	A nightmare journey courtesy of British Airway…	1	2024-09-08	United Kingdom	nightmare journey courtesy british airway wors…
1	Absolutely atrocious. LHR-ORD-LHR Round-trip. …	1	2024-09-06	United Kingdom	absolutely atrocious lhr ord lhr round trip br…
2	As someone who flies relentlessly with British…	4	2024-09-02	United Kingdom	someone fly relentlessly british airway busine…
3	Flew with British Airways club Europe on Satur…	2	2024-09-01	United Kingdom	flew british airway club europe saturday st au…
4	Straightforward check in T5. New site for club…	8	2024-08-30	United Kingdom	straightforward check new site club check work…

	compound	positive	neutral	negative	sentiment_category
0	-0.8520	0.079	0.823	0.098	Negative
1	-0.9600	0.093	0.687	0.220	Negative
2	0.7269	0.225	0.648	0.126	Neutral
3	-0.9371	0.000	0.864	0.136	Negative
4	0.8793	0.222	0.729	0.049	Positive

[120]:
```
# Extract year and month for aggregation
df['year_month'] = df['date'].dt.to_period('M')
```

[121]:
```
# Aggregate average sentiment score by year and month
monthly_sentiment = df.groupby('year_month')['compound'].mean()

# Convert PeriodIndex to Timestamp for plotting
monthly_sentiment.index = monthly_sentiment.index.to_timestamp()
```

[122]:
```
# Plotting
plt.figure(figsize=(12, 6))
plt.plot(monthly_sentiment.index, monthly_sentiment.values, marker='o',␣
 ↪linestyle='-', color='b')
plt.xlabel('Date')
plt.ylabel('Average Sentiment Score')
plt.title('Evolution of Review Sentiments Over Time')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[93]:
```
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk import download
```

[94]:
```
# Preprocessing function
def preprocess(text):
    stop_words = set(stopwords.words('english'))
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    return [word for word in tokens if word not in stop_words]
```

[95]:
```
# Apply preprocessing to the reviews
df['processed_reviews'] = df['reviews'].apply(preprocess)
```

[96]:
```
from gensim import corpora

# Create a dictionary and corpus
dictionary = corpora.Dictionary(df['processed_reviews'])
corpus = [dictionary.doc2bow(doc) for doc in df['processed_reviews']]
```

[97]:
```
from gensim import models

# Apply LDA
num_topics = 5  # You can adjust the number of topics based on your dataset
lda_model = models.LdaModel(corpus, num_topics=num_topics, id2word=dictionary,␣
 ↪passes=15)

# Display topics
topics = lda_model.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

```
(0, '0.018*"ba" + 0.015*"flight" + 0.011*"staff" + 0.011*"boarding"')
(1, '0.016*"seat" + 0.013*"ba" + 0.013*"class" + 0.013*"seats"')
(2, '0.024*"flight" + 0.016*"good" + 0.012*"crew" + 0.011*"service"')
(3, '0.031*"flight" + 0.015*"ba" + 0.010*"british" + 0.010*"airways"')
(4, '0.017*"ba" + 0.015*"british" + 0.015*"airline" + 0.015*"airways"')
```

[98]:
```
import pyLDAvis.gensim_models as gensimvis
import pyLDAvis

# Visualize the topics
lda_viz = gensimvis.prepare(lda_model, corpus, dictionary)
pyLDAvis.display(lda_viz)
```

[98]:
```
<IPython.core.display.HTML object>
```

[ ]: