Amazon Customer Review Analysis using Web Scraping And Sentimental Analysis

Importing Required Modules

In [1]:
```
import requests # To send HTTP/1.1 requests
from bs4 import BeautifulSoup #convert the data return from request to HTML for parsing

import pandas as pd #to conver the reviews to dataframe

import string #remove punctuations
import nltk
from nltk.corpus import stopwords
from nltk import PorterStemmer

import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter # to count the most common 50 words
```

In [2]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\hp\anaconda3\lib\site-packages (1.8.1)
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-packages (from wordcloud) (3.3.4)
Requirement already satisfied: numpy>=1.6.1 in c:\users\hp\anaconda3\lib\site-packages (from wordcloud) (1.20.
1)
Requirement already satisfied: pillow in c:\users\hp\anaconda3\lib\site-packages (from wordcloud) (8.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\hp\anaconda3\lib\site-packa
ges (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->w
ordcloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcl
oud) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib
->wordcloud) (2.8.1)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->w
ordcloud) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\hp\anaconda3\lib\site-packages (5.3.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\hp\anaconda3\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages (from plotly) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import plotly.express as px
from wordcloud import WordCloud
```

In [5]:
```
pip install vaderSentiment
```

```
Requirement already satisfied: vaderSentiment in c:\users\hp\anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: requests in c:\users\hp\anaconda3\lib\site-packages (from vaderSentiment) (2.25.
1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\lib\site-packages (from requests->va
derSentiment) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in c:\users\hp\anaconda3\lib\site-packages (from requests->vaderSen
timent) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\hp\anaconda3\lib\site-packages (from requests->vad
erSentiment) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hp\anaconda3\lib\site-packages (from requests-
>vaderSentiment) (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

Defining Functions to Read Amazon Data and Extract Customer Reviews from it

In [7]:
```
#Create a header that contains your request cookies, without cookies can not scrape amazon data it always shows

def AmazonData(url):
    r = requests.get(url, headers=header)
    return r.text


def ToHTML(url):
    htmldata = AmazonData(url)
    soup = BeautifulSoup(htmldata, 'html.parser')
    return soup
```

In [8]:
```
url ="https://www.amazon.com/Harry-Potter-Sorcerers-Stone-Rowling/dp/059035342X/ref=as_li_ss_tl?ie=UTF8&linkCod
```

In [9]:
```
header = ({'User-Agent':
            'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
            AppleWebKit/537.36 (KHTML, like Gecko) \
            Chrome/90.0.4430.212 Safari/537.36',
            'Accept-Language': 'en-US, en;q=0.5'})
```

In [10]:
```
soup = ToHTML(url) # RAW HTML data
```

Extracting Customer Reviews

Extracting Customer Name

In [11]:
```
def customer_name(soup):
    data_str = ""
    cus_list = []

    for item in soup.find_all("span", class_= "a-profile-name"): # the customer name is under HTML tag span, a-
        data_str = data_str + item.get_text()
        cus_list.append(data_str)
        data_str = ""
    return cus_list
```

In [12]:
```
Customer_List = customer_name(soup)
```

In [13]:
```
print(Customer_List)
```

```
['S. Werkema', 'lilly', 'Allison Geitner', 'Hellogoodbuy', 'Kat', 'Sara Bertrand', 'Linda D. Combs', 'Talulah1
3', 'Tamires Dias', 'MJH', 'Anthony R', 'Mariana Nascimento Ferreira', 'abcde']
```

Extracting Customer Review

In [14]:
```
def customer_review(soup):
    data_str = ""

    for item in soup.find_all("div", class_="a-expander-content reviewText review-text-content a-expander-parti
        data_str = data_str + item.get_text()

    review = data_str.split("\n")
    return review
```

In [15]:
```
Cust_Rev = customer_review(soup)
Customer_Review_list = []
for reviews in Cust_Rev:
    if reviews == "":
        pass
    else:
        Customer_Review_list.append(reviews)
```

In [16]:
```
# get link of see all reviews
review_link = soup.find("a",{'data-hook':'see-all-reviews-link-foot'})
```

In [17]:
```
full_review_link = "https://www.amazon.com/"+review_link['href']
```

In [18]:
```
full_review_link
```

Out[18]:
```
'https://www.amazon.com//Harry-Potter-Sorcerers-Stone-Rowling/product-reviews/059035342X/ref=cm_cr_dp_d_show_al
l_btm?ie=UTF8&reviewerType=all_reviews'
```

In [19]:
```
reviews_list = []
for k in range(50):
    page = requests.get(full_review_link+'&pageNumber='+str(k),headers=header)
    soup2 = BeautifulSoup(page.content)
    for i in soup2.find_all("span",{'data-hook':"review-body"}):
        reviews_list.append(i.text)
```

In [20]:
```
len(reviews_list)
```

Out[20]: 500

Converting the Reviews into a Dictionary

In [21]:
```
Dict_reviews = {'reviews':reviews_list}
reviews_df = pd.DataFrame.from_dict(Dict_reviews)
```

In [22]:
```
reviews_df.head()
```

Out[22]:

	reviews
0	\n\n I'm probably the only person on earth wh...
1	\n\n I got this book, Harry Potter and the So...
2	\n\n This book is a great value! If you can a...
3	\n\n Seen the movies a handful of times but d...
4	\n\n Perfect condition! Had to replace my fir...

Cleaning the Reviews

In [23]:
```
cleaned_reviews = reviews_df['reviews']
```

Removing the '\n' from the reviews

In [24]:
```
reviews_df['reviews'] = reviews_df['reviews'].apply(lambda x : x.strip('\n'))
```

In [25]:
```
reviews_df.head()
```

Out[25]:

	reviews
0	I'm probably the only person on earth who ha...
1	I got this book, Harry Potter and the Sorcer...
2	This book is a great value! If you can affor...
3	Seen the movies a handful of times but decid...
4	Perfect condition! Had to replace my first o...

In [26]:
```
reviews_df.shape
```

Out[26]: (500, 1)

Removing any reviews which is not in English

In [27]:
```
reviews_df = reviews_df[reviews_df.reviews.map(lambda x: x.isascii())]
```

In [28]:
```
reviews_df.shape
```

Out[28]: (465, 1)

Converting All Reviews To LowerCase

In [29]:
```
to_lower = lambda x: x.lower()
cleaned_reviews = cleaned_reviews.apply(to_lower)
```

Removing Puctuations

In [30]:
```
#Translate will transform each character according to the Translation table
remove_punctuations = lambda x: x.translate(str.maketrans('','',string.punctuation))
#Creating a Translation Table by maketrans-the 1st argument will be replaced with 2nd argument(in this case nor
cleaned_reviews = cleaned_reviews.apply(remove_punctuations)
```

Remove Emojis

In [31]:
```
remove_emojis = lambda x: x.encode('ascii', 'ignore').decode('ascii')
cleaned_reviews = cleaned_reviews.apply(remove_emojis)
```

Remove extra whitespaces

In [32]:
```
cleaned_reviews = cleaned_reviews.str.strip()
```

Removing Stop words

In [33]:
```
cleaned_reviews
```

Out[33]:
```
0      im probably the only person on earth who had n...
1      i got this book harry potter and the sorcerers...
2      this book is a great value if you can afford i...
3      seen the movies a handful of times but decided...
4      perfect condition had to replace my first one ...
                             ...
495    recipient is delighted  seeing the film first ...
496    awesome book with a great ending glad i finall...
497    havent read these since i was a kid probably a...
498    condition of book was amazing and the story it...
499                                              awesome
Name: reviews, Length: 500, dtype: object
```

In [34]:
```
#getting stop words from nltk module
nltk.download('stopwords')
Stop_words = stopwords.words("english") #list of stopwords
# removing stopwords
remove_words = lambda x:' '.join([word for word in x.split() if word not in Stop_words])
cleaned_reviews = cleaned_reviews.apply(remove_words)
```

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

In [35]:
```
cleaned_reviews
```

Out[35]:
```
0      im probably person earth never read harry pott...
1      got book harry potter sorcerers stone old copy...
2      book great value afford dont hard cover books ...
3      seen movies handful times decided wanted read ...
4      perfect condition replace first one since stol...
                             ...
495    recipient delighted seeing film first actually...
496    awesome book great ending glad finally started...
497    havent read since kid probably around 9 decide...
498    condition book amazing story addictive captiva...
499                                              awesome
Name: reviews, Length: 500, dtype: object
```

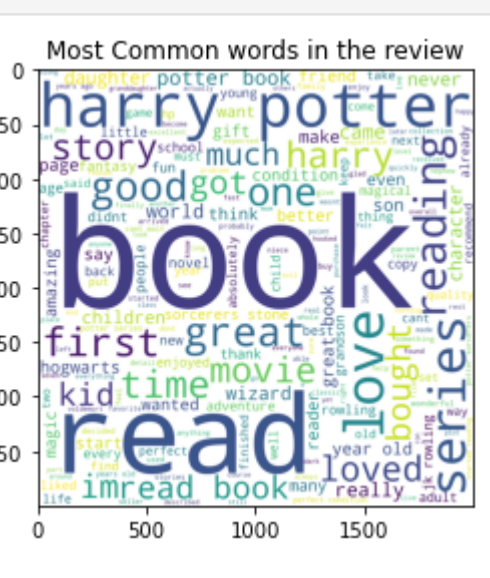
Analyzing the Reviews

Word Cloud to views based on thier frequency

In [36]:
```
word_list = [word for line in cleaned_reviews for word in line.split()]
```

In [37]:
```
wordcloud = WordCloud(height = 2000, width = 2000, background_color = 'white')
wordcloud = wordcloud.generate(' '.join(word_list))
```

In [38]:
```
plt.imshow(wordcloud)
plt.title("Most Common words in the review")
plt.show()
```



50 Most Common Words In Review

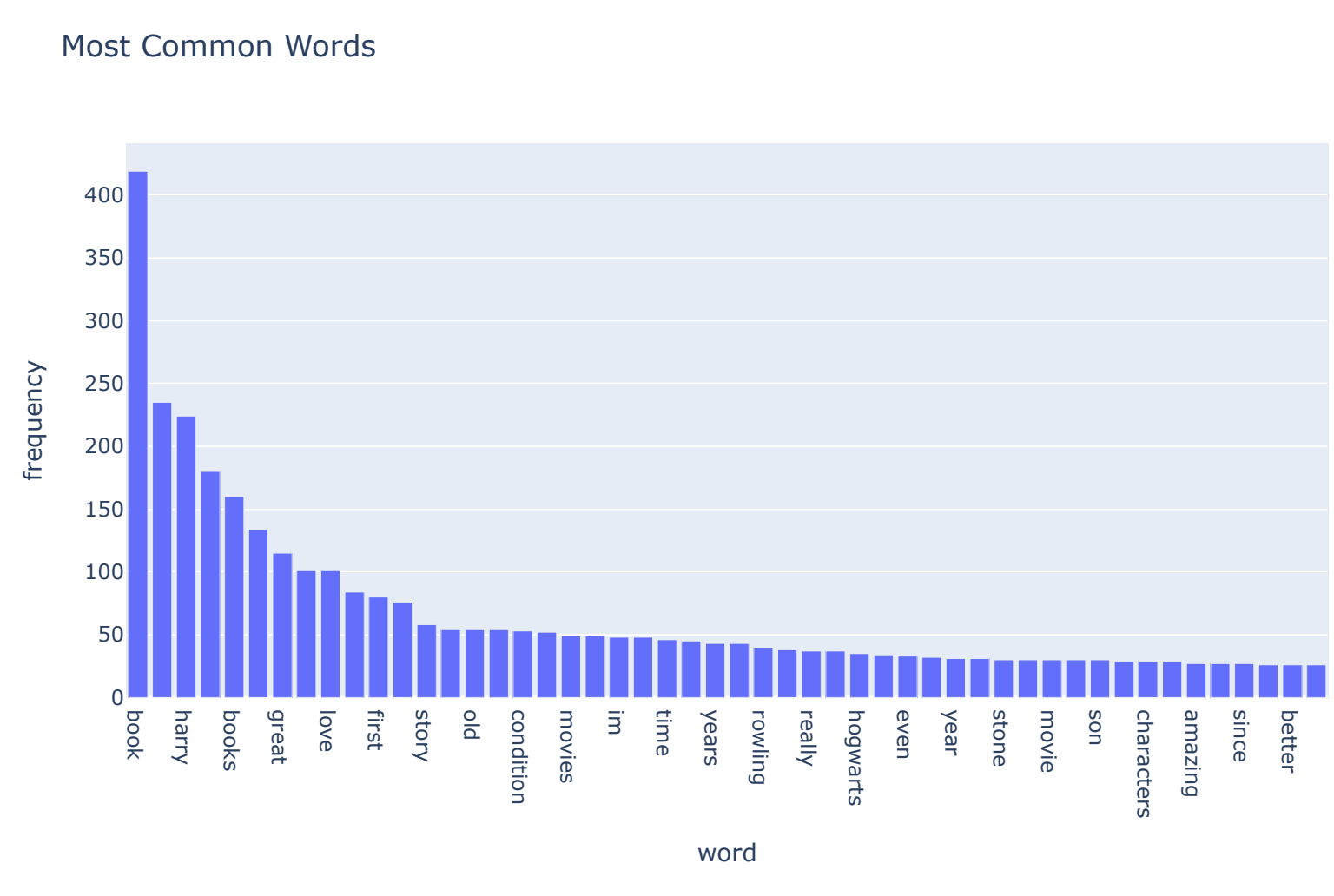
In [39]:
```
#Creating a list of all the words

word_counts = Counter(word_list).most_common(50) #using collections module
words_df = pd.DataFrame(word_counts)
words_df.columns = ['word', 'frequency']
```

In [40]:
```
words_df.head()
```

Out[40]:

	word	frequency
0	book	419
1	read	235
2	harry	224
3	potter	180
4	books	160

In [41]:
```
#plotting the common words using bar chart
px.bar(words_df, x='word',y ='frequency', title = 'Most Common Words')
```



Adding the Cleaned Reviews to the dataframe

In [42]:
```
reviews_df['cleaned_reviews'] = cleaned_reviews
```

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

Out[43]:

	reviews	cleaned_reviews
0	I'm probably the only person on earth who ha...	im probably person earth never read harry pott...
5	The front cover is badly bent ,back cover sh...	front cover badly bent back cover shows normal...
6	I read this book for two reasons. My 13 year...	read book two reasons 13 year old granddaughte...
8	Harry Potter and the Sorcerer's Stone is the...	harry potter sorcerers stone first book series...
9	I am currently rereading the Harry Potter bo...	currently rereading harry potter books first t...

Analyzing the Sentiments of the Reviews

In [44]:
```
sentiment_analyzer = SentimentIntensityAnalyzer()
```

polarity_scores will give score in following categories: Postive,Negative,Neutral,Compound

In [45]:
```
def sentiment_scores(sentence):
    score = sentiment_analyzer.polarity_scores(sentence)
    return score
```

Compund Score is the sum of postive,negative,neutral normalised between -1 and +1. Closer to 1 it is postive

In [46]:
```
def compound_score(sentence):
    compound = sentiment_scores(sentence)
    return compound['compound']
```

In [47]:
```
reviews_df['Sentiment_Score'] = reviews_df['cleaned_reviews'].apply(lambda x: compound_score(x))
```

In [48]:
```
reviews_df.head()
```

Out[48]:

	reviews	cleaned_reviews	Sentiment_Score
0	I'm probably the only person on earth who ha...	im probably person earth never read harry pott...	0.9558
5	The front cover is badly bent ,back cover sh...	front cover badly bent back cover shows normal...	-0.9456
6	I read this book for two reasons. My 13 year...	read book two reasons 13 year old granddaughte...	0.4885
8	Harry Potter and the Sorcerer's Stone is the...	harry potter sorcerers stone first book series...	-0.4019
9	I am currently rereading the Harry Potter bo...	currently rereading harry potter books first t...	0.9986

Labeling Each review as postive,negative,neutral based on the compound score

In [49]:
```
labelize = lambda x: 'neutral' if x == 0 else('postive' if x>0 else 'negative')
reviews_df['Sentiment_label'] = reviews_df.Sentiment_Score.apply(labelize)
```

In [50]:
```
reviews_df.head()
```

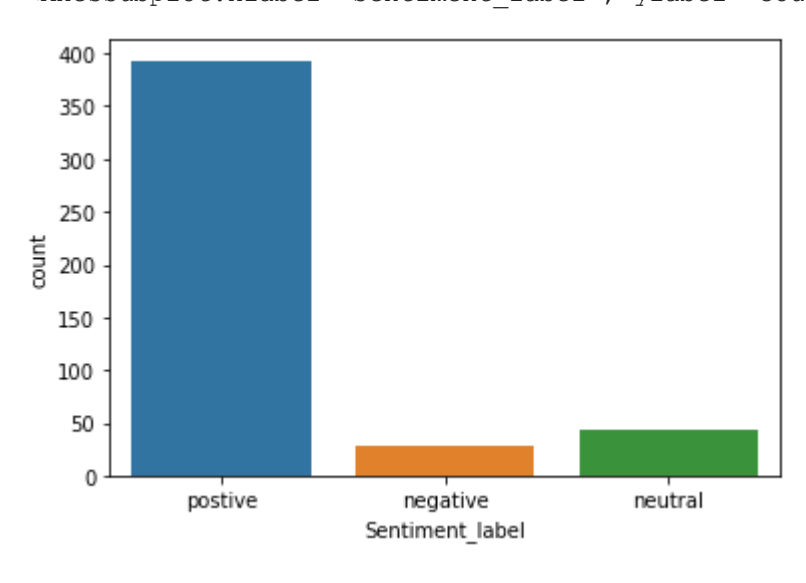
Out[50]:

	reviews	cleaned_reviews	Sentiment_Score	Sentiment_label
0	I'm probably the only person on earth who ha...	im probably person earth never read harry pott...	0.9558	postive
5	The front cover is badly bent ,back cover sh...	front cover badly bent back cover shows normal...	-0.9456	negative
6	I read this book for two reasons. My 13 year...	read book two reasons 13 year old granddaughte...	0.4885	postive
8	Harry Potter and the Sorcerer's Stone is the...	harry potter sorcerers stone first book series...	-0.4019	negative
9	I am currently rereading the Harry Potter bo...	currently rereading harry potter books first t...	0.9986	postive

Plotting the Distribution of Sentiment Label

In [54]:
```
sns.countplot(x = reviews_df['Sentiment_label'])
```

Out[54]: <AxesSubplot:xlabel='Sentiment_label', ylabel='count'>



Word Cloud for postive reviews

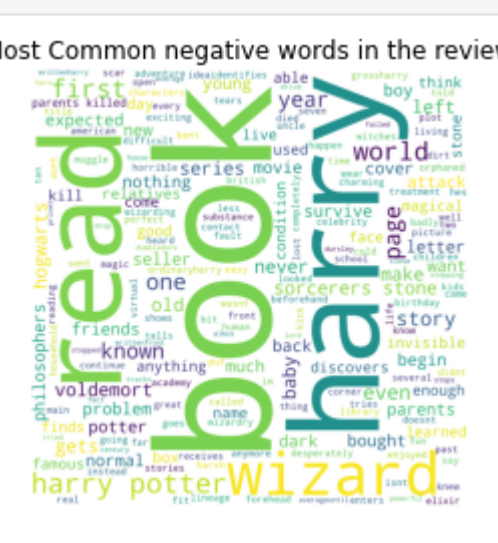
In [63]:
```
wordcloud = wordcloud.generate(''.join(reviews_df.loc[reviews_df['Sentiment_label']=='postive','cleaned_reviews
```

In [64]:
```
plt.imshow(wordcloud)
plt.title("Most Common Postive words in the review")
plt.axis("off")
plt.show()
```

Word Cloud for negative reviews

In [58]:
```
wordcloud = wordcloud.generate(''.join(reviews_df.loc[reviews_df['Sentiment_label']=='negative','cleaned_review
```

In [61]:
```
plt.imshow(wordcloud)
plt.title("Most Common negative words in the review")
plt.axis("off")
plt.show()
```



In [65]:
```
common_words = ['harry','potter','book','harry potter', 'ron','hermoine','hogwarts']
```

In [67]:
```
new_wc = WordCloud(stopwords = common_words, background_color ='white').generate(''.join(reviews_df.loc[reviews
```

In [68]:
```
plt.imshow(new_wc)
plt.title("Most Common negative words in the review")
plt.axis("off")
plt.show()
```

In [ ]: