AMAZON MEN & WOMEN SHOE REVIEWS DATASET

IMPORT LIBRARIES

In [68]:
```
import pandas as pd
!pip install contractions
import contractions
!pip install nltk
import nltk
nltk.download('all')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
!pip install vaderSentment
!pip install textblob
from PIL import image
import re
nltk.download("stopwords")
nltk.download('punkt')
nltk.download('wordnet')
from collections import Counter
```

```
Requirement already satisfied: contractions in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from contractions) (0.0.24)
Requirement already satisfied: anyascii in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from textsearch>=0.0.21->contractions)
(0.3.2)
Requirement already satisfied: pyahocorasick in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from textsearch>=0.0.21->contraction
s) (2.0.0)

[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: nltk in c:\users\kemi\appdata\local\programs\pyt
hon\python311\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\users\kemi\appdata\local\programs\py
thon\python311\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\kemi\appdata\local\programs\p
ython\python311\lib\site-packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\kemi\appdata\local\p
rograms\python\python311\lib\site-packages (from nltk) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\kemi\appdata\local\programs\pyt
hon\python311\lib\site-packages (from nltk) (4.66.1)
Requirement already satisfied: colorama in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from click->nltk) (0.4.6)

[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger_ru is already
[nltk_data]    |       up-to-date!
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package basque_grammars is already up-to-date!
[nltk_data]    | Downloading package bcp47 to
[nltk_data]    |     C:\Users\KEMI\AppData\Roaming\nltk_data...
[nltk_data]    |   Package bcp47 is already up-to-date!
[nltk_data]    | Downloading package biocreative_ppi to
```

DATA COLLECTION AND PREPROCESING

- DATA COLLECTION

In [2]:
```
# Specifying the path to the CSV file
myShoes= r"C:/Users/KEMI/Documents/Sentiment Analysis/Shoes_Data.csv"

# Read the CSV file into a pandas dataFrame
Shoes  = pd.read_csv(myShoes)
```

In [3]:
```
Shoes
```

Out|

	title	price	rating	total_review	product_des	reviews	reviews_rati	Shoe Typ
0	CLYMB Outdoor Sports Running Shoes for Mens Boy	₹279.00	2.9 out of 5 stars	2389 ratings	Elevate your style with this classy pair of Ru...	Not happy with product|| It's not as expected....	1.0 out of 5 stars|| 1.0 out of 5 stars|| 3.0 ...	Men
1	Bourge Men's Loire-z126 Running Shoes	₹479.00	3.9 out of 5 stars	11520 ratings	The product will be an excellent pick for you....	Memory cushioning in these shoes is the best f...	5.0 out of 5 stars|| 1.0 out of 5 stars|| 5.0 ...	Men
2	T-Rock Men's Sneaker	₹430.00	3.3 out of 5 stars	1251 ratings	Flaunt with these stylish and unique red casua...	Worth to its amount|| Go for it|| Perfect|| 5 ...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...	Men
3	Robbie jones Sneakers Casual Canvas Fabric Col...	₹499.00	4.2 out of 5 stars	3 ratings	Robbie Jones Shoes Are Designed To Keeping In ...	Sup quality|| Good but not expected|| Awesome 🔌.!	5.0 out of 5 stars|| 3.0 out of 5 stars|| 5.0 ...	Men
4	Sparx Men's Sd0323g Sneakers	₹499.00	4.2 out of 5 stars	20110 ratings	Sparx is a spectacular range of footwear from ...	Best|| Satisfied!|| Affordable beauty 😳😳😳😳 the...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...	Men
...	...	...	...	...	...	...	...	...
1225	Nike Men's React Vision Running Shoes	₹7256.00	4.4 out of 5 stars	200 ratings	The Nike react vision is a STORY of surreal co...	Must buy|| not have a great fiting but great q...	5.0 out of 5 stars|| 3.0 out of 5 stars|| 5.0 ...	Women
1226	Puma Men's B.O.G Limitless Hi Evoknit Sneakers	₹5822.00	4.3 out of 5 stars	25 ratings	The B.O.G limitless is Puma's key style for th...	Worth buying !|| Classy Bold and Stylish !!|| ...	4.0 out of 5 stars|| 5.0 out of 5 stars|| 3.0 ...	Women
1227	new balance Women's FuelCell Echolucent Runnin...	₹5362.00	4.5 out of 5 stars	817 ratings	Lead the pack in New Balance's Echolucent snea...	size variation in product recd n size chart|| ...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...	Women

Shoes: shows columns and rows present in the Amazon shoes customer reviews

In [4]:
```
Shoes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1230 entries, 0 to 1229
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   title                1230 non-null   object
 1   price                1230 non-null   object
 2   rating               1230 non-null   object
 3   total_reviews        1230 non-null   object
 4   product_description  1230 non-null   object
 5   reviews              1230 non-null   object
 6   reviews_rating       1230 non-null   object
 7   Shoe Type            1230 non-null   object
dtypes: object(8)
memory usage: 77.0+ KB
```

Checking for Null values

In [21]:
```
np.sum(Shoes.isnull().any(axis=1))
```

Out[21]:

```
0
```

The column represent the objects and there are no no null values in the dataset.

In [22]:
```
#Showing the total number of shoes under review

    Shoes.count()
```

Out[22]:

```
title                  1230
price                  1230
rating                 1230
total_reviews          1230
product_description    1230
reviews                1230
reviews_rating         1230
Shoe Type              1230
dtype: int64
```

In [23]:
```
print(Shoes.head(5))
```

```
                                                title    price  \
0    CLYMB Outdoor Sports Running Shoes for Mens Boy  ₹279.00
1              Bourge Men's Loire-z126 Running Shoes  ₹479.00
2                               T-Rock Men's Sneaker  ₹430.00
3  Robbie jones Sneakers Casual Canvas Fabric Col...  ₹499.00
4                       Sparx Men's Sd0323g Sneakers  ₹499.00

               rating  total_reviews  \
0  2.9 out of 5 stars   2389 ratings
1  3.9 out of 5 stars  11520 ratings
2  3.3 out of 5 stars   1251 ratings
3  4.2 out of 5 stars      3 ratings
4  4.2 out of 5 stars  20110 ratings

                                product_description  \
0  Elevate your style with this classy pair of Ru...
1  The product will be an excellent pick for you....
2  Flaunt with these stylish and unique red casua...
3  Robbie Jones Shoes Are Designed To Keeping In ...
4  Sparx is a spectacular range of footwear from ...

                                             reviews  \
0  Not happy with product|| It's not as expected....
1  Memory cushioning in these shoes is the best f...
2  Worth to its amount|| Go for it|| Perfect|| 5 ...
3  Sup quality|| Good but not expected|| Awesome 👌.!
4  Best|| Satisfied!|| Affordable beauty 😳😳😳😳 the...

                                    reviews_rating Shoe Type
0  1.0 out of 5 stars|| 1.0 out of 5 stars|| 3.0 ...       Men
1  5.0 out of 5 stars|| 1.0 out of 5 stars|| 5.0 ...       Men
2  5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...       Men
3  5.0 out of 5 stars|| 3.0 out of 5 stars|| 5.0 ...       Men
4  5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...       Men
```

The above shows sample of 1st five rows of raw customer reviews which makes it necessary to clean the dataset before inputting into the models

In [24]:
```
# Showing the total number of reviews

TotalReviews=Shoes[['total_reviews']]
TotalReviews
```

Out[2

	total_reviev
0	2389 ratings
1	11520 ratings
2	1251 ratings
3	3 ratings
4	20110 ratings
...	...
1225	200 ratings
1226	25 ratings
1227	817 ratings
1228	67 ratings
1229	210 ratings

In [ ]:

In [25]:
```
#Showing the unique features

len(Shoes["total_reviews"].unique()), len(Shoes["title"].unique())
```

Out[25]:

```
(513, 902)
```

In [26]:
```
# Grouping by shoe brands
Shoes.groupby("total_reviews")["title"].unique()
```

Out[26]:

```
total_reviews
1 rating        [VON HUETTE Men's Black/Brown Panny Loafer Sho...
10 ratings      [Red Tape Men's Rso111 Walking Shoes, new bala...
100 ratings     [Longwalk Women Latest Collection Sneakers Shoes]
1006 ratings              [Red Chief Casual Shoes for Men RC2506]
1007 ratings                                   [Puma Men's Shoe]
                                    ...
977 ratings     [Skechers Men's Go Walk 4 Black Nordic Walking...
981 ratings                    [Sparx Men's Sd0631g Casual Shoes]
982 ratings                    [Sparx Men's Sd0631g Casual Shoes]
99 ratings                          [BATA Men's Wonder Rain Shoe]
990 ratings                   [BATA Women's Aroma Fashion Sandals]
Name: title, Length: 513, dtype: object
```

In [ ]:

Showing the total reviews per shoe

In [27]:
```
totalreviews= Shoes[['total_reviews']]
total = pd.DataFrame(totalreviews)

total['total_reviews'] = Shoes['total_reviews'].str.extract('(\d+)', expand
```

In [28]:
```
total['total_reviews']
```

Out[28]:

```
0         2389
1        11520
2         1251
3            3
4        20110
         ...
1225       200
1226        25
1227       817
1228        67
1229       210
Name: total_reviews, Length: 1230, dtype: object
```

In [ ]:

Selecting the columns required for the sentiment analysis

In [4]:
```
Shoes2=Shoes[['reviews','reviews_rating']]
Shoes2
```

Out[

	reviews	reviews_ratin
0	Not happy with product|| It's not as expected....	1.0 out of 5 stars|| 1.0 out of 5 stars|| 3.0 ...
1	Memory cushioning in these shoes is the best f...	5.0 out of 5 stars|| 1.0 out of 5 stars|| 5.0 ...
2	Worth to its amount|| Go for it|| Perfect|| 5 ...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...
3	Sup quality|| Good but not expected|| Awesome 🔌.!	5.0 out of 5 stars|| 3.0 out of 5 stars|| 5.0 ...
4	Best|| Satisfied!|| Affordable beauty 😵😵😵😵 the...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...
...	...	...
1225	Must buy|| not have a great fiting but great q...	5.0 out of 5 stars|| 3.0 out of 5 stars|| 5.0 ...
1226	Worth buying !|| Classy Bold and Stylish !!|| ...	4.0 out of 5 stars|| 5.0 out of 5 stars|| 3.0 ...
1227	size variation in product recd n size chart|| ...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 5.0 ...
1228	Verified Purchase|| Verified Purchase|| Verifi...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 4.0 ...
1229	Great shoe|| excellent quality|| Old manufactu...	5.0 out of 5 stars|| 5.0 out of 5 stars|| 3.0 ...

Splitting the shoe reviews into different rows

In [5]:
```
rew  = []
rat = []
```

```
    for j in Shoes2.index:
        lst = [i for i in Shoes2.iloc[j].reviews.split('||')]
        for k in lst:
            rew.append(k)

    for j in Shoes2.index:
        lst = [i for i in Shoes2.iloc[j].reviews_rating.split('||')]
        for k in lst:
            rat.append(k)

    Shoes2 = pd.DataFrame(list(zip(rew, rat)),
                   columns =['reviews', 'review_rating'])
```

In [6]:
```
Shoes2
```

Out[

	reviews	review_ratir
0	Not happy with product	1.0 out of 5 stars
1	It's not as expected.	1.0 out of 5 stars
2	AVERAGE PRODUCT	3.0 out of 5 stars
3	Pic more beautiful	3.0 out of 5 stars
4	Got damage product. But quality is average fo...	3.0 out of 5 stars
...	...	...
9953	Go for it!	5.0 out of 5 stars
9954	Excellent product	5.0 out of 5 stars
9955	Nice shoe	5.0 out of 5 stars
9956	Nice	5.0 out of 5 stars
9957	Asics shoes are the best	5.0 out of 5 stars

Getting the first digit of the review _rating

In [7]:
```
import re

    def get_first_digit(text):
        match = re.search(r'\d', text)
        if match:
            return match.group(0)
        else:
            return None
```

In [8]:
```
 #Applying the get_first_digit function to 'Review_rating' column
    Shoes2['Review_rating'] = Shoes2['review_rating'].apply(get_first_digit)

    Shoes2['Review'] = Shoes2['reviews'].apply

    Shoes2.head()
```

Out|

	reviews	review_rating	Review_rating	Revie
0	Not happy with product	1.0 out of 5 stars	1	<bound method Series.apply of 0 ...
1	It's not as expected.	1.0 out of 5 stars	1	<bound method Series.apply of 0 ...
2	AVERAGE PRODUCT	3.0 out of 5 stars	3	<bound method Series.apply of 0 ...
3	Pic more beautiful	3.0 out of 5 stars	3	<bound method Series.apply of 0 ...
4	Got damage product. But quality is average	3.0 out of 5 stars	3	<bound method Series.apply of 0 ...

In [9]:
```
Shoes2=Shoes2[['reviews','Review_rating']]
Shoes2
```

Out|

	reviews	Review_ratir
0	Not happy with product	1
1	It's not as expected.	1
2	AVERAGE PRODUCT	3
3	Pic more beautiful	3
4	Got damage product. But quality is average fo...	3
...	...	...
9953	Go for it!	5
9954	Excellent product	5
9955	Nice shoe	5
9956	Nice	5
9957	Asics shoes are the best	5

The above shows information on 'reviews' and 'review rating' to be used for the analysis

In [35]:
```
Shoes2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9958 entries, 0 to 9957
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   reviews        9958 non-null   object
 1   Review_rating  9958 non-null   object
dtypes: object(2)
memory usage: 155.7+ KB
```

In [10]:
```
#checking the ratiings

    rating=[]
    for item in Shoes2['Review_rating']:
        rating+=[int(item[0])]
    Shoes2['Review_rating']=rating
    Shoes2
```

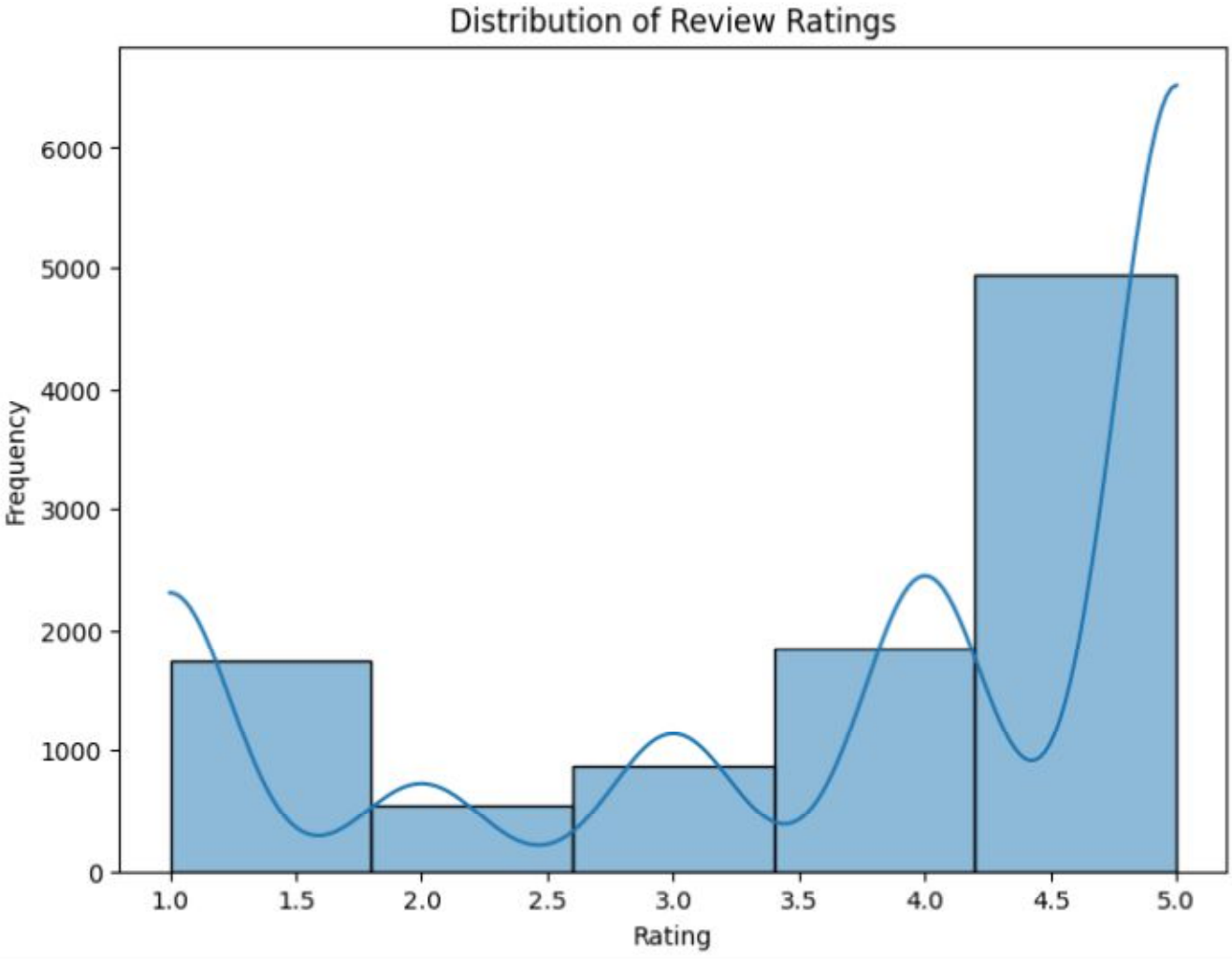
```
C:\Users\KEMI\AppData\Local\Temp\ipykernel_20040\4278088450.py:6: SettingWithC
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  Shoes2['Review_rating']=rating
```

Out[1

	reviews	Review_ratir
0	Not happy with product	1
1	It's not as expected.	1
2	AVERAGE PRODUCT	3
3	Pic more beautiful	3
4	Got damage product. But quality is average fo...	3
...	...	...
9953	Go for it!	5
9954	Excellent product	5
9955	Nice shoe	5
9956	Nice	5
9957	Asics shoes are the best	5

In [37]:
```
# Plotting histogram for Review_rating
    plt.figure(figsize=(8, 6))
    sns.histplot(Shoes2['Review_rating'], bins=5, kde=True)
```

```
plt.title('Distribution of Review Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```

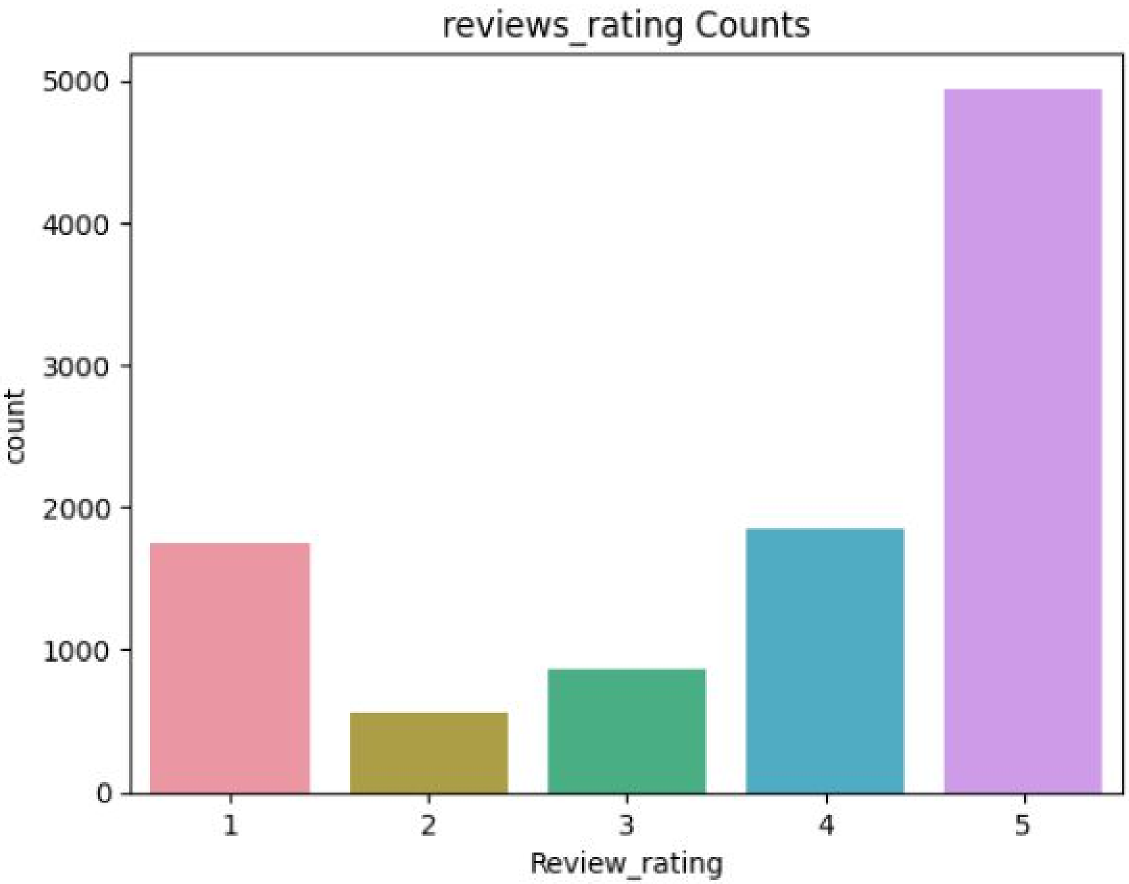


In [38]:
```
# Set labels from 1 and 5
def decrease_label_by_one(label):
    return label - 1

Shoes2['review_rating'] = Shoes2['Review_rating'].apply(decrease_label_by_o
```

```
C:\Users\KEMI\AppData\Local\Temp\ipykernel_19720\3171532756.py:5: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  Shoes2['review_rating'] = Shoes2['Review_rating'].apply(decrease_label_by_on
e)
```

In [39]:
```
color_palette = ['red', 'blue']
sns.set_palette(color_palette)
```

```
sns.countplot(x=Shoes2['Review_rating'])

plt.title('reviews_rating Counts')
plt.show()
```



In [ ]:

Unique symbols

In [40]:
```
# Combine all the text into a single string
textindata = ' '.join(Shoes2['reviews'].astype(str))

# Find unique symbols in the combined text
uniquename = ''.join(set(textindata))

# Print the unique symbols
print(uniquename)
```

```
sGFिLोbॉि🙂c😟प🔥आई😧र😋✊eYErzaIn☑j😂❌😧लउ□p🏃😁B💖🎁❤9यV&H😮g▒इ@E6😊💋NU😇तo□
qc▒7👞ु'Nेढ😐=0👋☺क3»😎😔Iy(:😂झ😖🌟👌द💰#ीL💚_"🤘□dfS□)Wृो😆👎R🌹डवß☺*|🙂😢मज-😭K
Z?1च□x💪ATO8😍+👟न₹❢!💘.2💯✓wX😫बिCPk💸✌ख👎J🙂'vअ😀😑□ळ«⇢😵ैuृ14…□ीद♂😆␣स⭑ht%💥
ह /छ□iM5❤👇👍😒□m😟😌Qगटा।🙂
```

In [ ]:

In [41]:
```
# Showing the reviews and rating from 1 to 5

Shoes2_1=Shoes2[Shoes2['Review_rating']==1]
Shoes2_2=Shoes2[Shoes2['Review_rating']==2]
```

```
    Shoes2_3=Shoes2[Shoes2['Review_rating']==3]
    Shoes2_4=Shoes2[Shoes2['Review_rating']==4]
    Shoes2_5=Shoes2[Shoes2['Review_rating']==5]
    Shoes3=[Shoes2_1,Shoes2_2,Shoes2_3,Shoes2_4,Shoes2_5,]
```

In [42]:
```
Shoes3
```

Out[42]:

```
[                                                reviews  Review_rating  \
 0                               Not happy with product              1
 1                                It's not as expected.              1
 6                                        Worst product              1
 8      Low quality makes pain on heals by sharp edge...              1
 9                                  Do not buy it anyway              1
 ...                                                 ...            ...
 9896                                  Copy lagta hai              1
 9907                                Colour is not good              1
 9932                                     Report abuse              1
 9947                                     Report abuse              1
 9952      Old stock teared in just 1.5 month of running              1

       review_rating
 0                 0
 1                 0
 6                 0
 8                 0
 9                 0
 ...             ...
 9896              0
 9907              0
 9932              0
 9947              0
 9952              0

 [1749 rows x 3 columns],
                                                reviews  Review_rating  \
 5            Bad product different from what was listed              2
 7                                             Don't buy              2
 41                           You get what you pay for              2
 47                                              Shoes              2
 52                                               Nice              2
 ...                                                 ...            ...
 9844                               Verified Purchase              2
 9864                                    Report abuse              2
 9887  Too tight. Old torn off shoe box heavily taped...              2
 9915                               Verified Purchase              2
 9935                                    Report abuse              2

       review_rating
 5                 1
 7                 1
 41                1
 47                1
 52                1
 ...             ...
 9844              1
 9864              1
 9887              1
 9915              1
 9935              1
```

In [43]:
```
Shoes3[0]
```

Out[4

	reviews		Review_rating	review_ratir
0	Not happy with product	1	0	
1	It's not as expected.	1	0	
6	Worst product	1	0	
8	Low quality makes pain on heals by sharp edge...	1	0	
9	Do not buy it anyway	1	0	
...	...	...	...	
9896	Copy lagta hai	1	0	
9907	Colour is not good	1	0	
9932	Report abuse	1	0	
9947	Report abuse	1	0	
9952	Old stock teared in just 1.5 month of running	1	0	

Proportion of Duplicate observations

In [44]:
```
num_duplicates_Shoes2 = Shoes2.duplicated().sum()
prop_duplicates_Shoes2 = Shoes2.duplicated().mean()

print("Number of duplicate observations in the dataset:", num_duplicates_Sh
print("Proportion of duplicate observations in the dataset:", prop_duplicat
```

```
Number of duplicate observations in the dataset: 4465
Proportion of duplicate observations in the dataset: 0.44838320947981525
```

Splitting into Train and Test datasets

In [45]:
```
import numpy as np

# Set a random seed for reproducibility
np.random.seed(3)

# Shuffle the ShoeDataReview indices
Shoes_indices = np.arange(len(Shoes))
np.random.shuffle(Shoes_indices)

# Define the proportion for splitting (70% train, 30 test)
split_ratio = 0.7

# Calculate the split point
split_point = int(len(Shoes_indices) * split_ratio)

# Split the ShoeDataReview into training and testing sets
train = Shoes_indices[:split_point]
```

```
test= Shoes_indices[split_point:]

# Create training and testing ShoeDataReviewFrames
train_Shoes = Shoes.iloc[train]
test_Shoes = Shoes.iloc[test]
```

In [46]:
```
print("Number of rows in the training data:", len(train_Shoes))
print("Number of rows in the test data:", len(test_Shoes))
```

```
Number of rows in the training data: 861
Number of rows in the test data: 369
```

In [47]:
```
Shoes .iloc[train].describe()
```

Out[4

	title	price	rating	total_review	product_des	reviews	reviews_rati	Shoe Ty
count	861	861	861	861	861	861	861	861
unique	677	493	29	413	579	710	630	2
top	Adidas Men Running Shoes	₹499.00	4.1 out of 5 stars	25 ratings	The product will be an excellent pick for you....	Verified Purchase	5.0 out of 5 stars	Men

In [48]:
```
Shoes .iloc[test].describe()
```

Out[4

	title	price	rating	total_review	product_des	reviews	reviews_rati	Shoe Ty
count	369	369	369	369	369	369	369	369
unique	314	264	25	226	274	320	286	2
top	Clarks Men's Formal Shoes	₹599.00	4.1 out of 5 stars	5 ratings	The product will be an excellent pick for you....	size variation in product recd n size chart|| ...	5.0 out of 5 stars	Men

EXPLORATORY DATA ANALYSIS (EDA)

In [ ]:

Wordcloud

In [50]:
```
from wordcloud import WordCloud
```

Wordcloud for general Reviews

In [50]:
```
# Extract the 'reviews' column
text_column = Shoes2["reviews"]

# Join the text from all rows
all_text = ' '.join(text_column)

wordcloud = WordCloud(
```

```
    stopwords=None,
    background_color='black',
    width=800,
    height=400,
).generate(all_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Turn off axis labels
plt.show()
```

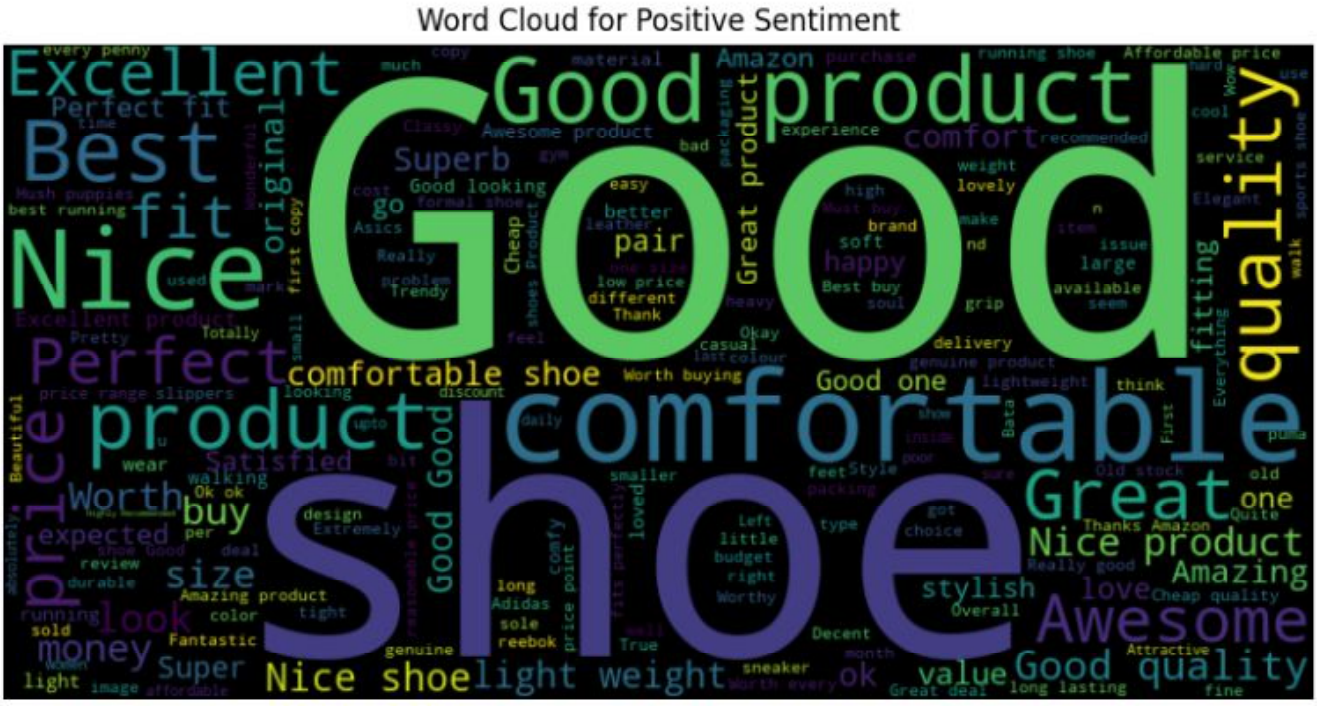
The WordCloud shows sample of the features of the customer reviews in the dataset. It can be seen that the dataset include 'Verified Purchase' and 'Report abuse' which appeared as part of the reviews.

In [51]:
```
from textblob import TextBlob


# Define a function to determine sentiment using TextBlob
def get_sentiment(text):
    analysis = TextBlob(text)
    if analysis.sentiment.polarity > 0:
        return 'positive'
    else:
        return 'negative'

# Apply sentiment analysis to the 'Text' column and create a new 'Sentiment
Shoes2["sentiment"] = Shoes2["reviews"].apply(get_sentiment)

# Filter the DataFrame for positive sentiment
positive_df = Shoes2[Shoes2["sentiment"] == 'positive']
```

```
# Extract the 'Text' column from the filtered DataFrame
positive_text_column = positive_df['reviews']

# Join the text from all rows with positive sentiment
positive_text = ' '.join(positive_text_column)
```

Wordcloud for Positive Reviews

In [52]:
```
import matplotlib.pyplot as plt

# Create a WordCloud object with optional configurations (e.g., stopwords,
wordcloud = WordCloud(
    stopwords=None,  # You can provide your own list of stopwords here if n
    background_color='black',
    width=800,
    height=400,
).generate(positive_text)

# Display the word cloud for positive sentiment using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Turn off axis labels
plt.title('Word Cloud for Positive Sentiment')
plt.show()
```

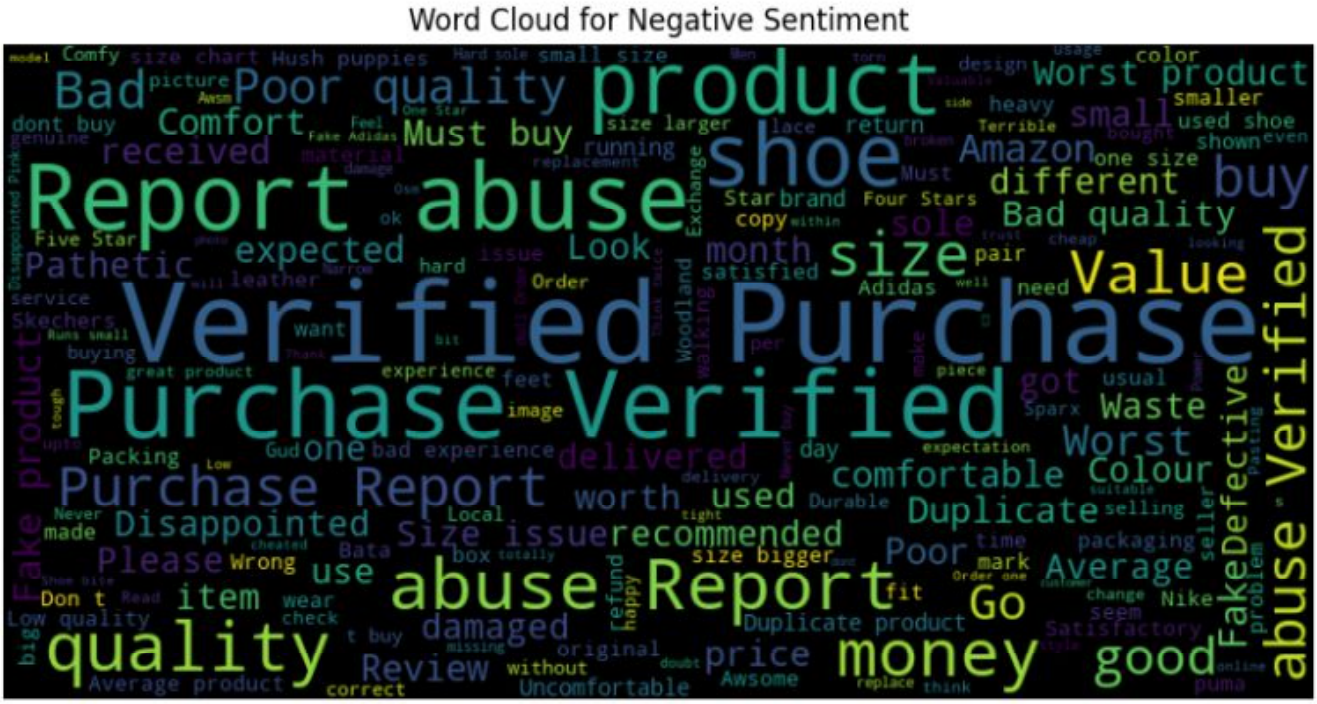


Wordcloud for Negative Reviews

In [53]:
```
# Filter the DataFrame for negative sentiment
negative_df = Shoes2[Shoes2["sentiment"] == 'negative']

# Extract the 'Text' column from the filtered DataFrame
negative_text_column = negative_df['reviews']

# Join the text from all rows with positive sentiment
```

```
    negative_text = ' '.join(negative_text_column)
```

In [54]:
```
# Create a WordCloud object with optional configurations (e.g., stopwords,
wordcloud = WordCloud(
    stopwords=None,  # You can provide your own list of stopwords here if n
    background_color='black',
    width=800,
    height=400,
).generate(negative_text)

# Display the word cloud for positive sentiment using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Turn off axis labels
plt.title('Word Cloud for Negative Sentiment')
plt.show()
```



Frequent words

In [55]:
```
from collections import Counter
```

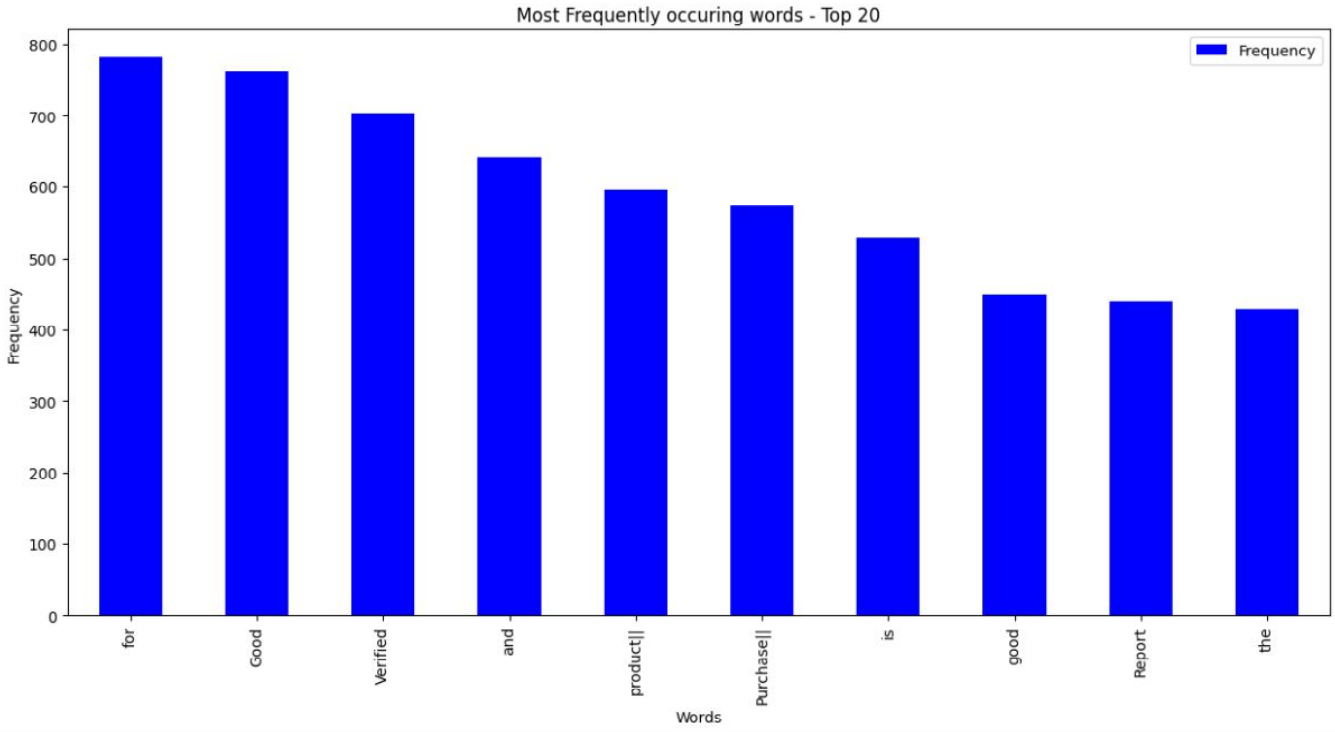
In [56]:
```
## Frequency Words
words = Counter(' '.join(Shoes['reviews'].to_list()).split())
Frequency_words = pd.DataFrame([words]).transpose().reset_index().rename(co
Frequency_words = Frequency_words.sort_values('Frequency',ascending=False).
Frequency_words['Rank'] = Frequency_words['Rank'].apply(lambda x : x+1)
Frequency_words.head(20)
```

Out[5

	Rank	Words	Frequen
0	1	for	781
1	2	Good	762
2	3	Verified	703
3	4	and	642
4	5	product||	595
5	6	Purchase||	573
6	7	is	528
7	8	good	450
8	9	Report	439
9	10	the	429
10	11	product	428
11	12	Very	410
12	13	Not	395
13	14	abuse||	364
14	15	shoes	361
15	16	not	345
16	17	Nice	344
17	18	in	311
18	19	of	310

The above data shows the most frequent words and their ranking as shown on the Amazon shoes reviews dataset

In [57]:
```
Top10 = Frequency_words[['Words','Frequency']].head(10)
Top10.plot(x="Words", y="Frequency", kind='bar',  figsize=(15,7), color = '
plt.title("Most Frequently occuring words - Top 20")
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.show()
```



The graphical representation of the most frequently used words in the Amazon shoes review dataset

Number of Men & Women shoes

In [58]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\kemi\appdata\local\programs\p
ython\python311\lib\site-packages (5.17.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\kemi\appdata\local\p
rograms\python\python311\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from plotly) (23.0)

[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [59]:
```
import plotly.express as px
```

In [60]:
```
men_count = (Shoes['Shoe Type']=='Men').sum()
    #print(men_count)
    women_count = (Shoes['Shoe Type']=='Women').sum()
    #print(women_count)

    toGraph = {'shoe type': ['Men', 'Women'], 'count': [men_count, women_count]

    dftoGraph = pd.DataFrame(data=toGraph)
    fig = px.bar(dftoGraph, x='shoe type', y='count', title='Shoe Count by Gend

    fig.show()
```

In [61]:
```
# Number of men and women shoes under the reviews
toGraph
```

Out[61]:

```
{'shoe type': ['Men', 'Women'], 'count': [856, 374]}
```

Average length of words used by Customers

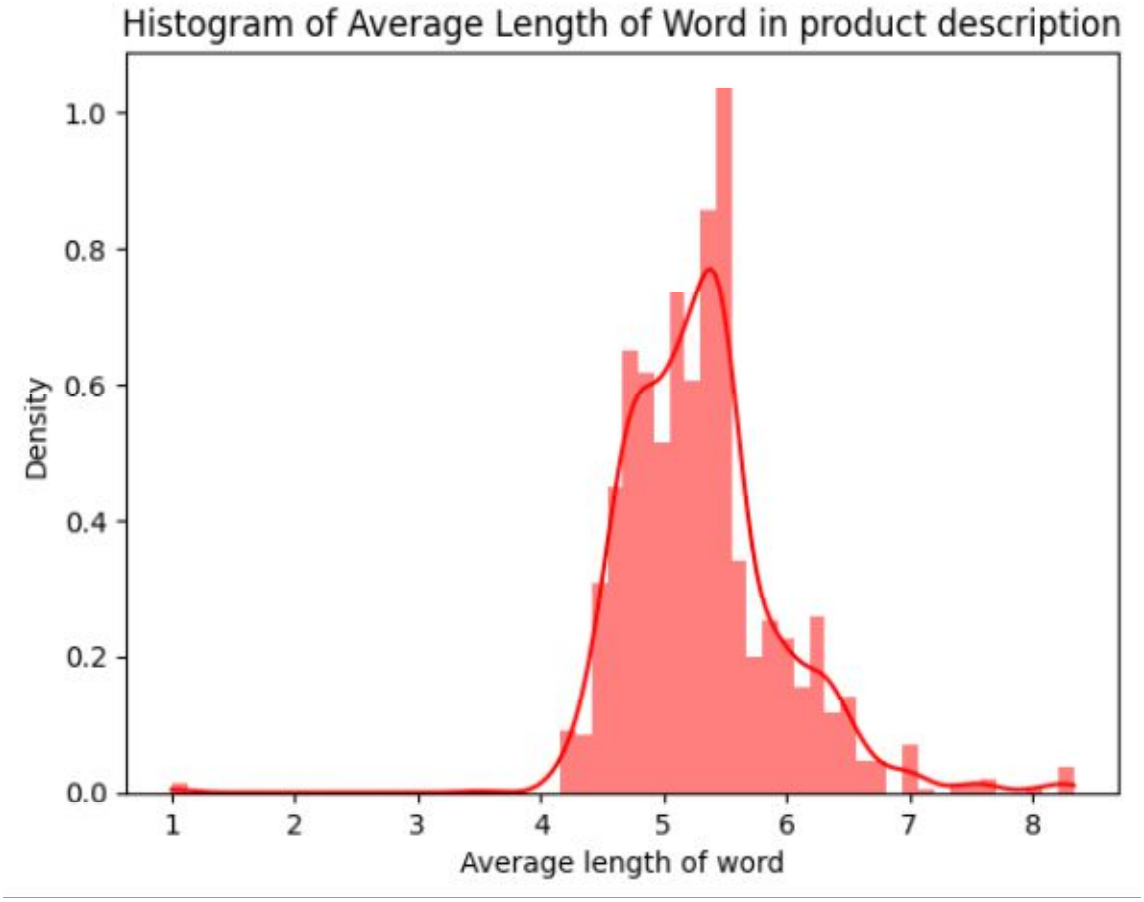
In [62]:
```
# average length of the word in product description.
def avg_leng_word(x):
    return np.sum([len(w) for w in x.split()]) / len(x.split())

Shoes['avg_prod_desc_length'] = Shoes['product_description'].apply(avg_leng
sns.histplot(Shoes['avg_prod_desc_length'], kde=True, stat='density', linew
plt.title('Histogram of Average Length of Word in product description')
plt.xlabel('Average length of word')


fig, ax = plt.subplots(1,2)
sns.histplot(x=Shoes['avg_prod_desc_length'], hue=Shoes['reviews_rating'],
             kde=True, stat='density', linewidth=0, ax=ax[0])
sns.boxplot(y=Shoes['avg_prod_desc_length'], x=Shoes['reviews_rating'], ax=
```

```
ax[0].set_title('Histogram of Average Length of Word in product description
ax[0].set_xlabel('Average length of word')
ax[1].set_title('Boxplots of Average Length of Word in product description'
ax[1].set_xlabel('Product overall category')
ax[1].set_ylabel('Average length of word')
```

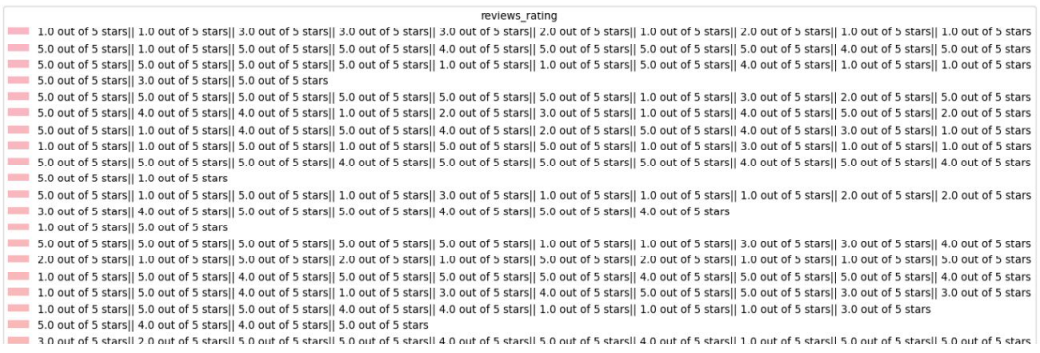
Out[62]:

```
Text(0, 0.5, 'Average length of word')
```

```
C:\Users\KEMI\AppData\Local\Programs\Python\Python311\Lib\site-packages\IPytho
n\core\events.py:89: UserWarning:

Creating legend with loc="best" can be slow with large amounts of data.
```




```
C:\Users\KEMI\AppData\Local\Programs\Python\Python311\Lib\site-packages\IPytho
n\core\pylabtools.py:152: UserWarning:

Creating legend with loc="best" can be slow with large amounts of data.
```



In [ ]:

Heatmap

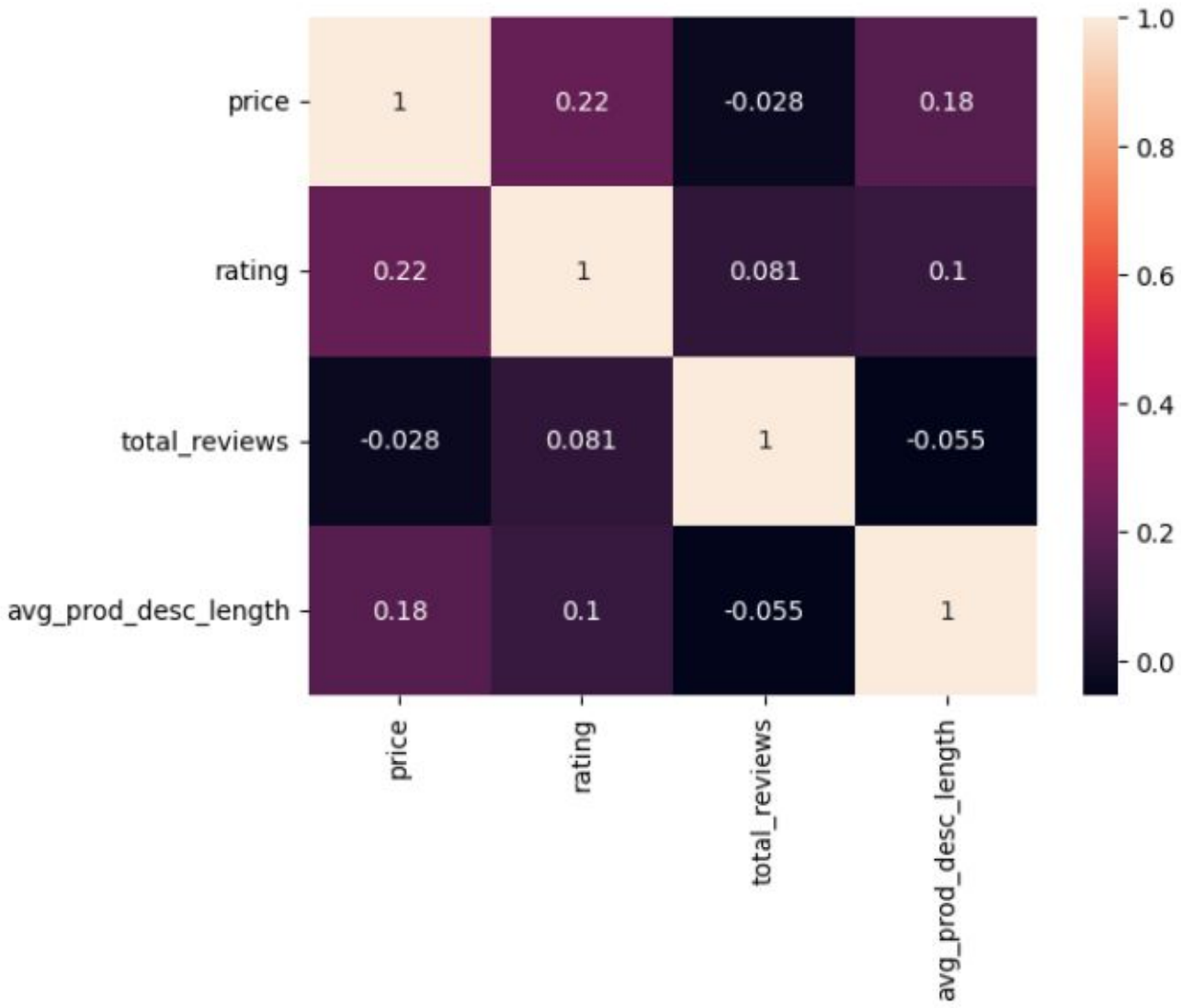
In [63]:
```
Shoes['rating']=Shoes['rating'].apply(lambda x: str(x).replace(' out of 5 s
Shoes['price']=Shoes['price'].apply(lambda x: str(x).replace('₹','') if '₹'
Shoes['total_reviews']=Shoes['total_reviews'].apply(lambda x: str(x).replac
Shoes['total_reviews']=Shoes['total_reviews'].apply(lambda x: str(x).replac
Shoes['rating']=Shoes['rating'].astype(float)
Shoes['price']=Shoes['price'].astype(float)
Shoes['total_reviews']=Shoes['total_reviews'].astype(int)
```

In [64]:
```
sns.heatmap(Shoes.corr(),annot=True)
```

```
C:\Users\KEMI\AppData\Local\Temp\ipykernel_19720\2096493812.py:1: FutureWarnin
g:

The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the val
ue of numeric_only to silence this warning.
```

Out[64]:

```
<Axes: >
```



In [65]:
```
## Count ratings
Shoes2.Review_rating.value_counts(normalize = True)
```

Out[65]:

```
5    0.496485
4    0.186082
1    0.175638
3    0.086664
2    0.055132
Name: Review_rating, dtype: float64
```

Proportion of the ratings in the dataset

In [ ]:

Classifying the reviews into Positive and Negative

In [11]:
```
def classify(x):
    if x < 3:
        return "negative"
    else:
        return "positive"


Shoes2['reviews_rating'] = Shoes2['Review_rating'].apply(classify)
```

```
C:\Users\KEMI\AppData\Local\Temp\ipykernel_20040\4262195023.py:8: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  Shoes2['reviews_rating'] = Shoes2['Review_rating'].apply(classify)
```

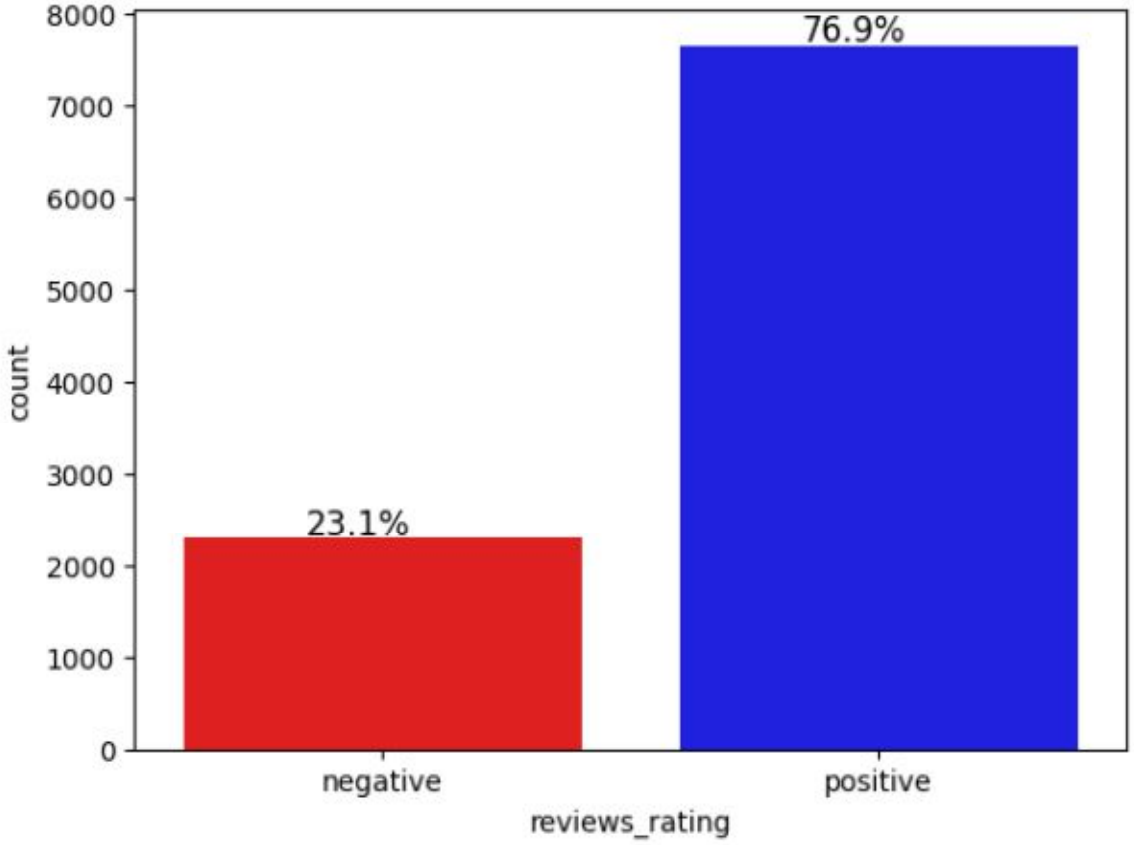
In [12]:
```
Shoes2['reviews_rating'].value_counts()
```

Out[12]:

```
positive    7660
negative    2298
Name: reviews_rating, dtype: int64
```

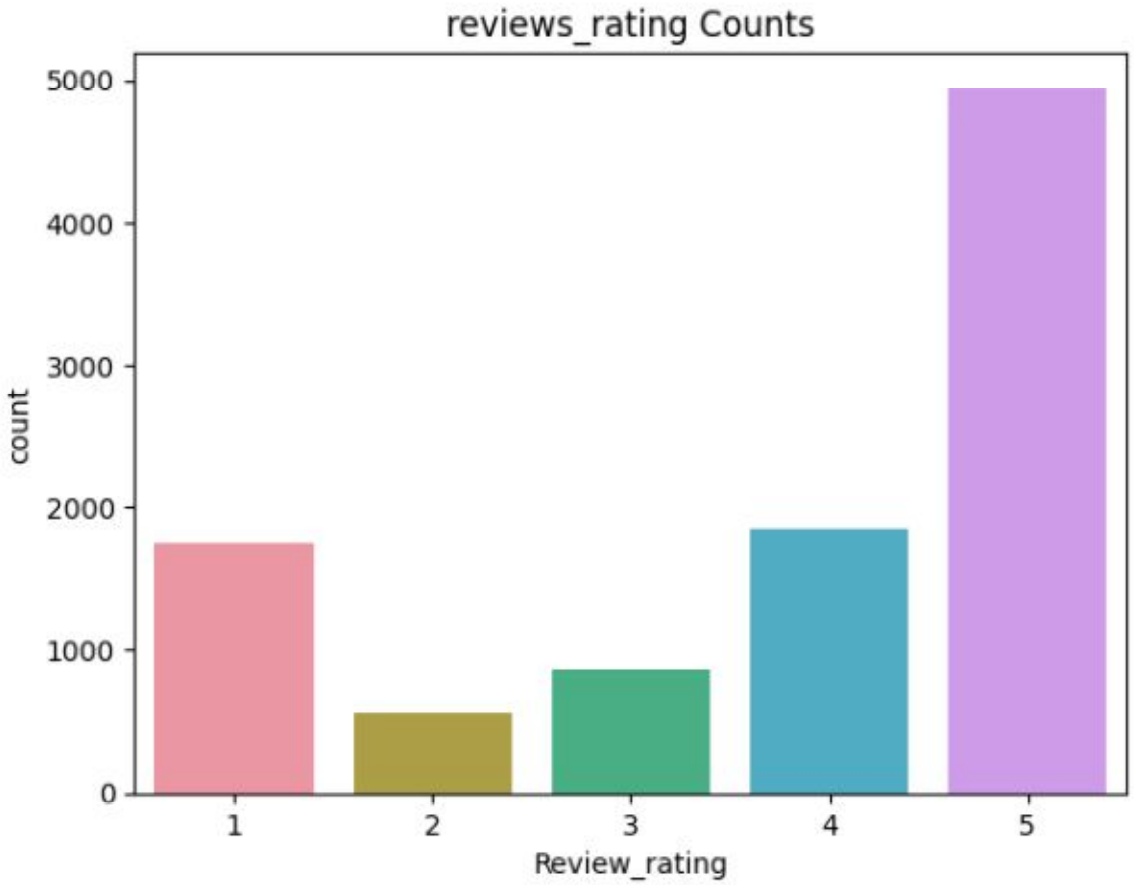
In [58]:
```
# Plot the distribution of the class label
def bar_plot(data, feature):
    # Creating the countplot
    plot = sns.countplot(x = feature, data = data)

    # Finding the length the whole data
    total = len(data)

    # Creating the percentages to each label in the data
    for p in plot.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width() / 2 - 0.05
        y = p.get_y() + p.get_height()
        plot.annotate(percentage, (x, y),ha="center",
            va = "center",
            size = 12,
            xytext = (0, 5),
            textcoords = "offset points")
    plt.show()
```

```
bar_plot(Shoes2 ,'reviews_rating')
```



In [68]:
```
# Set labels from 1 and 5
def decrease_label_by_one(label):
    return label - 1

Shoes2['reviews_rating'] = Shoes2['Review_rating'].apply(decrease_label_by_
```

In [69]:
```
color_palette = ['red', 'blue']
sns.set_palette(color_palette)
sns.countplot(x=Shoes2['Review_rating'])

plt.title('reviews_rating Counts')
plt.show()
```



In [ ]:
In [ ]:
In [ ]:

-- DATA PREPROCESSING

In [13]:
```
# Creating the copy of the data frame

df = Shoes2.copy()
```

In [14]:
```
df
```

Out[1

	reviews	Review_rating	reviews_ratir
0	Not happy with product	1	negative
1	It's not as expected.	1	negative
2	AVERAGE PRODUCT	3	positive
3	Pic more beautiful	3	positive
4	Got damage product. But quality is average fo...	3	positive
...	...	...	...
9953	Go for it!	5	positive
9954	Excellent product	5	positive
9955	Nice shoe	5	positive
9956	Nice	5	positive
9957	Asics shoes are the best	5	positive

Focusing on the columns needed for the sentiment analysis 'reviews' and 'sentiment'

In [16]:
```
df=df[['reviews', 'reviews_rating']]
df
```

Out[1

	reviews	reviews_ratir
0	Not happy with product	negative
1	It's not as expected.	negative
2	AVERAGE PRODUCT	positive
3	Pic more beautiful	positive
4	Got damage product. But quality is average fo...	positive
...	...	...
9953	Go for it!	positive
9954	Excellent product	positive
9955	Nice shoe	positive
9956	Nice	positive
9957	Asics shoes are the best	positive

Total number of Reviews in the dataset

In [17]:
```
df["reviews"].count()
```

Out[17]:

```
9958
```

In [18]:
```
df
```

Out[1

	reviews	reviews_ratir
0	Not happy with product	negative
1	It's not as expected.	negative
2	AVERAGE PRODUCT	positive
3	Pic more beautiful	positive
4	Got damage product. But quality is average fo...	positive
...	...	...
9953	Go for it!	positive
9954	Excellent product	positive
9955	Nice shoe	positive
9956	Nice	positive
9957	Asics shoes are the best	positive

In [ ]:

Uniques symbols

In [19]:
```
all_text_clean = str()
for sentence in df['reviews'].values:
    all_text_clean += sentence

unique_symbols_clean = ''.join(set(all_text_clean))
print(unique_symbols_clean)
```

```
aw💰😍💔' ो😀❌❤ नI्Dल😄GN_|□4+✌गA%💃👌□N□✊'ॆ5👟😋⋯→□😐h😫E»डn👎Cfp"vॆI👋😒«)cझ😁b0W
🔥u😡Y*q□सS@□💥😂☑ू🙂J▒च!⊔□😢ब(रयVe=☹छU❢…🙂2ढ8द&😌हtRE?🙂ी🤣ॊI9P🙂ी💯Bk💖💪xॉ🙂👇
T🌟😟iQइतM😑▒jट💋☆🤙ॎ😄जr-17िअ/ßOH😮3मz✓.X😟Fईउ❤कंळ।😑😇♂oखL1m□y😎:ुZ😟s6☺😪□ं
🏃थ🌹gव🎁प🤮K😆👞∟💞🤘□👍#₹dआc
```

In [20]:
```
all_text_clean = str()
for sentence in df['reviews'].values:
    all_text_clean += sentence

unique_symbols_clean = ''.join(set(all_text_clean))

clean_review_counts = df["reviews"].value_counts()
print(clean_review_counts)
```

```
 Verified Purchase                                                  647
 Report abuse                                                       418
 Good                                                               259
 Good product                                                       116
 Nice                                                                98
                                                                    ...
 Not so great product                                                 1
 A size bigger than usual might work                                  1
 very good product                                                    1
 Not good!                                                            1
 Disaster quality & worst Response from Amazon &Bata teams....        1
Name: reviews, Length: 5189, dtype: int64
```

Removing 'Verified Purchase' and 'Report Abuse'

In [21]:
```
df = df[~df.reviews.str.contains("Report abuse")]
df = df[~df.reviews.str.contains("Verified")]
```

In [22]:
```
df["reviews"].value_counts()
```

Out[22]:

```
 Goo
d
            259
 Good produc
t
    116
 Nic
e
             98
 Value for mone
y
  65
 Nice produc
t
     48

               ...
Feel like an ordinary shoe. Have had few redtape shoes before but this dont mat
ch the feel.      1
 Not so great produc
t                                                                          1
 A size bigger than usual might wor
k                                                        1
 very good produc
t
 1
 It's heavier than expecte
d                                                                    1
Name: reviews, Length: 5184, dtype: int64
```

Removal of Contractions

In [23]:
```
def replace_contractions(text):
        return contractions.fix(text)
df['reviews'] = df['reviews'].apply(lambda x: replace_contractions(x))
```

In [24]:
```
df.head(3)
```

Out[2

	reviews	reviews_ratin
0	Not happy with product	negative
1	It is not as expected.	negative

In [ ]:

-- Tokenization

In [25]:
```
#Breaking down the reviews into single words
df['reviews'] = df.apply(lambda x: nltk.word_tokenize(x['reviews']), axis =
```

In [26]:
```
df.head(3)
```

Out[2

	reviews	reviews_ratin
0	[Not, happy, with, product]	negative
1	[It, is, not, as, expected, .]	negative

In [ ]:

Removal of Digits

In [27]:
```
#Removing numbers and digits
def remove_digit(row):
    tokens = [word for word in row if not word.isdigit()]
    return tokens
```

In [28]:
```
df['reviews'] = df['reviews'].apply(lambda x: remove_digit(x))
```

In [29]:
```
df.head(5)
```

Out[2

	reviews	reviews_ratin
0	[Not, happy, with, product]	negative
1	[It, is, not, as, expected, .]	negative
2	[AVERAGE, PRODUCT]	positive
3	[Pic, more, beautiful]	positive
4	[Got, damage, product, ., But, quality, is,	positive

This will remove any form of digits and numbers from the reviews

In [ ]:

Removal of Non Ascii characters

In [30]:
```
import unicodedata
```

In [31]:
```
# Removing non English alphabets
def remove_non_ascii(row):
    new_row = []
    for word in row:
        new_word = unicodedata.normalize('NFKD', word).encode('ascii', 'ign
        new_row.append(new_word)
```

```
        return new_row
```

In [32]:
```
df['reviews'] = df['reviews'].apply(lambda x: remove_non_ascii(x))
```

In [33]:
```
df.head(5)
```

Out[3

	reviews	reviews_ratin
0	[Not, happy, with, product]	negative
1	[It, is, not, as, expected, .]	negative
2	[AVERAGE, PRODUCT]	positive
3	[Pic, more, beautiful]	positive
4	[Got, damage, product, ., But, quality, is,	positive

This has removed all special characters in the reviews

In [ ]:

Removal of Punctuations

In [34]:
```
def remove_punctuation(row):
    tokens = [word for word in row if word not in string.punctuation]
    return tokens
```

In [35]:
```
df['reviews'] = df['reviews'].apply(lambda x: remove_punctuation(x))
```

In [36]:
```
df.head(5)
```

Out[3

	reviews	reviews_ratin
0	[Not, happy, with, product]	negative
1	[It, is, not, as, expected]	negative
2	[AVERAGE, PRODUCT]	positive
3	[Pic, more, beautiful]	positive
4	[Got, damage, product, But, quality, is,	positive

In [ ]:

Lowercasing

In [37]:
```
def to_lowercase(row):
    words = [word.lower() for word in row]
    return words

df['reviews'] = df['reviews'].apply(lambda x: to_lowercase(x))
```

In [38]:
```
df.head(5)
```

Out[3

	reviews	reviews_ratin
0	[not, happy, with, product]	negative
1	[it, is, not, as, expected]	negative
2	[average, product]	positive
3	[pic, more, beautiful]	positive
4	[got, damage, product, but, quality, is,	positive

In [ ]:

Stopwords removal

In [39]:
```
#removal of stopwords of English words
stop_words = stopwords.words('english')

def remove_stopwords(row):
    words = [word for word in row if word not in stop_words]
    return words

df['reviews'] = df['reviews'].apply(lambda x: remove_stopwords(x))
```

In [40]:
```
df.head(5)
```

Out[4

	reviews	reviews_ratin
0	[happy, product]	negative
1	[expected]	negative
2	[average, product]	positive
3	[pic, beautiful]	positive
4	[got, damage, product, quality, average,	positive

Resetting the index of the dataframe

In [41]:
```
# resetting the index of the dataframe
df.reset_index(drop = True, inplace = True)
```

In [ ]:

Word Normalization

1. Stemming

In [42]:
```
# Create a WordNetLemmatizer object
ps = PorterStemmer()

# Define a function to lemmatize a list of words
def stemmatize_words(word_list):
    return [ps.stem(word) for word in word_list]

# Apply the lemmatize_words function to the 'Text' column of your DataFrame
stemmer = df['reviews'].apply(stemmatize_words)
```

In [43]:
```
stemmer.head(3)
```

Out[43]:

```
0     [happi, product]
1              [expect]
2    [averag, product]
Name: reviews, dtype: object
```

In [ ]:

2. Lemmatization

In [44]:
```
# Create a WordNetLemmatizer object
lm = WordNetLemmatizer()

# Define a function to lemmatize a list of words
def l2. emmatize_words(word_list):
    return [lm.lemmatize(word) for word in word_list]

# Apply the lemmatize_words function to the 'Text' column of your DataFrame
df['reviews'] = df['reviews'].apply(lemmatize_words)
```

In [45]:
```
df.head(5)
```

Out[4

	reviews	reviews_ratin
0	[happy, product]	negative
1	[expected]	negative
2	[average, product]	positive
3	[pic, beautiful]	positive
4	[got, damage, product, quality, average,	positive

Lemmatization for word normalization was selected over stemming because it normalizes the words without losing their original meaning

Putting all together

In [46]:
```
import re
```

In [47]:
```
# saving the cleaned text back to review column

def join_words(row):
    words = ' '.join([word for word in row])
    words = re.sub('[^a-zA-Z]', ' ', words)
    return words
df['reviews'] = df['reviews'].apply(lambda x: join_words(x))
```

In [48]:
```
df
```

Out[4

	reviews	reviews_ratin
0	happy product	negative
1	expected	negative
2	average product	positive
3	pic beautiful	positive
4	got damage product quality average	positive
...	...	...
8811	go	positive
8812	excellent product	positive
8813	nice shoe	positive
8814	nice	positive
8815	asics shoe best	positive

Word Cloud of cleaned words

In [51]:
```
# Extract the 'Text' column
text_column = df['reviews']

# Join the text from all rows
all_text = ' '.join(text_column)

# Creating the wordcloud objects
wordcloud = WordCloud(
    stopwords=None,
    background_color='black',
    width=800,
    height=400,
).generate(all_text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Sentiment Mapping into 1 and 0

In [52]:
```
# Converting the Sentiment to Numeric

sentiment_mapping = {'positive': 1, 'negative': 0}
df['sentiment'] = df['reviews_rating'].map(sentiment_mapping)
```

In [53]:
```
df.head(5)
```

Out[5

	reviews	reviews_rating	sentime
0	happy product	negative	0
1	expected	negative	0
2	average product	positive	1
3	pic beautiful	positive	1
4	got damage product quality	positive	1

In [ ]:

In [54]:
```
neg_reviews = df[df.sentiment == 0]
neg_reviews = neg_reviews.sample(random_state=0)
all_intents = df.reviews.tolist()
```

In [55]:
```
neg_reviews.head(5)
```

Out[5

	reviews	reviews_rating	sentime

In [56]:
```
def get_sample_of_sentiment_reviews(data, sentiment, n, random_seed):
    """
    Get a random sample of reviews for a specified sentiment class from a d
```

```
        Args:
            data (pd.DataFrame): The dataset containing the "Sentiment" column.
            sentiment (str): The sentiment class ('positive', 'neutral', or 'ne
            n (int): The number of reviews to sample.
            random_seed (int): The random seed for reproducibility.

        Returns:
            pd.DataFrame: A DataFrame containing the sampled reviews for the sp
        """
        # Filter the dataset to select reviews of the specified sentiment class
        filtered_reviews = df[df['sentiment'] == sentiment]

        # Sample n reviews with the specified random seed
        sampled_reviews = filtered_reviews.sample(n=n, random_state=random_seed

        return sampled_reviews
```

In [57]:
```
neg_reviews = get_sample_of_sentiment_reviews(df, sentiment=0, n=100, rando

neg_reviews.head()
```

Out[5

	reviews	reviews_rating	sentime
6615	old stock	negative	0
691	poor quality	negative	0
3452	correct fit	negative	0
3984	okay okay type shoe	negative	0

In [58]:
```
pos_reviews = get_sample_of_sentiment_reviews(df, sentiment=1, n=50, random

pos_reviews.head()
```

Out[5

	reviews	reviews_rating	sentime
5582	first impression	positive	1
3721	quality product	positive	1
2340	nice product recommend	positive	1
2540	campus shoe	positive	1

In [59]:
```
all_intents
```

Out[59]:

```
['happy product',
 'expected',
 'average product',
 'pic beautiful',
 'got damage product quality average    ',
 'bad product different listed',
 'worst product',
 'buy',
 'low quality make pain heals sharp edge inside shoe',
 'buy anyway',
 'memory cushioning shoe best feature',
 'poor quality product',
 'best gym n sport',
 'must must must buy',
 'worth price',
 'satisfied',
 'please try shoe',
 'good heel comfy cushion',
 'good lookinggood quality exact fit bit sized',
 'awesome amazon awesome',
 'worth amount',
 'go',
 'perfect',
 'star',
 'itam received',
 'worst product  please buy',
 'nice shoe',
 'pro con',
 'buy',
 'ignore go another option',
 'sup quality',
 'good expected',
 'awesome',
 'best',
 'satisfied',
 'affordable beauty eye catcher',
 'economical soft foot',
 'good looking shoe however buy one size low',
 'must bye puja',
 'much variation size',
 'review day use',
 'get pay',
 'cheap branded spent       x time better        one time',
 'awesome',
 'nice product',
 'nic fitting',
 'poor quality product',
 'shoe',
 'nice',
 'bad',
 'fit comfortably',
 'comfortable fancy'
```

FEATURE EXTRACTION AND MACHINE LEARNING MODELS

Bag of Words(CountVectorizer) (BOW)

In [60]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [61]:
```
# Keeping only 200 features as number of features will increase the process
count_vec = CountVectorizer(max_features = 500)

df_features = count_vec.fit_transform(df['reviews'])

# Convert the data features to array
df_features = df_features.toarray()
```

In [62]:
```
df_features.shape
```

Out[62]:

```
(8816, 500)
```

In [63]:
```
#checking the occurence 1st 2 words
df_features[:2]
```

Out[63]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)
```

Splitting the dataset

In [64]:
```python
# Selecting the targeted features
X = df_features
y = df['sentiment']
```

Balancing the dataset

In [65]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (0.11.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from imbalanced-learn) (1.24.2)
Requirement already satisfied: scipy>=1.5.0 in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (from imbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\kemi\appdata\loc
al\programs\python\python311\lib\site-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kemi\appdata\lo
cal\programs\python\python311\lib\site-packages (from imbalanced-learn) (3.1.0)

[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [71]:
```
from imblearn.over_sampling import SMOTE
from collections import Counter
```

In [72]:
```
print(f'Original dataset shape : {Counter(y)}')

smote = SMOTE(random_state = 42)
X_resampled, y_resampled = smote.fit_resample(X, y)

print(f'Resampled dataset shape {Counter(y_resampled)}')
```

```
Original dataset shape : Counter({1: 6658, 0: 2158})
Resampled dataset shape Counter({0: 6658, 1: 6658})
```

In [ ]:

In [73]:
```
# classification report and confusion matrix

def metrics_score(actual, predicted):

    print(classification_report(actual, predicted))

    cm = confusion_matrix(actual, predicted)

    plt.figure(figsize = (8, 5))

    sns.heatmap(cm, annot = True,  fmt = '.2f', xticklabels = ['positive',

    plt.ylabel('Actual')

    plt.xlabel('Predicted')

    plt.show()
```

In [74]:
```
# Train and Test sets
from sklearn.model_selection import train_test_split
```

In [75]:
```
#split the dataset
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
```

Number of Test and Train Datasets

In [76]:
```
print("Number of rows in the training data:", len(x_train))
print("Number of rows in the test data:", len(x_test))
```

```
Number of rows in the training data: 6171
Number of rows in the test data: 2645
```

1. RANDOM FOREST

RANDOM FOREST MACHINE LEARNNIG ON BOW

In [77]:
```
# Trainng the Model and calculating the accuracy on the TEST DATA

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [78]:
```
rf = RandomForestClassifier(random_state=0, n_jobs=-1)
```

In [79]:
```
#Train the Model
rf.fit(x_train, y_train)

# Get predictions on the test data
y_pred = rf.predict(x_test)

#Get the metrics
metrics_score(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0       0.77      0.63      0.69       647
           1       0.89      0.94      0.91      1998

    accuracy                           0.86      2645
   macro avg       0.83      0.78      0.80      2645
weighted avg       0.86      0.86      0.86      2645
```

In [ ]:

Visualization of the top 50 words used for the Model

In [80]:
```
def get_top50_words(model, all_features):

    # Addition of top 50 feature into top_feature after training the model
    top_features=''

    feat = model.feature_importances_

    features = np.argsort(feat)[::-1]

    for i in features[0:50]:
        top_features+=all_features[i]
        top_features+=','
```

```
        from wordcloud import WordCloud

        wordcloud = WordCloud(background_color = "black", colormap = 'viridis',
                              height = 1000).generate(top_features)

        # Display the generated image:
        plt.imshow(wordcloud, interpolation = 'bilinear')
        plt.figure(1, figsize = (20, 20), frameon='equal')
        plt.title('Top 50 features WordCloud', fontsize = 10)
        plt.axis("off")
        plt.show()
```

In [81]:
```
# Instantiate the feature from the vectorizer
features = count_vec.get_feature_names_out()

# Showing the top 50 words used in the BOW model
get_top50_words(rf, features)
```

In [ ]:

TF-IDF

Term Frequency - Inverse Document Frequency: It makes sure that less importance is given to the most frequent words, and it also considers less frequent words.

In [82]:
```
# TfidfVectorizer to convert text data to numbers.
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [83]:
```
tfidf_vec = TfidfVectorizer(max_features = 500)

# transform the text with TF-IDF
df_features_tfidf = tfidf_vec.fit_transform(df['reviews'])

# Feature shape
df_features_tfidf = df_features_tfidf.toarray()
```

In [84]:
```
# Select the features and target
X_tfidf = df_features_tfidf
y = df.sentiment
```

In [85]:
```
# Split data into training and testing set.
x_train, x_test, y_train, y_test = train_test_split(X_tfidf, y, test_size =
```

In [ ]:

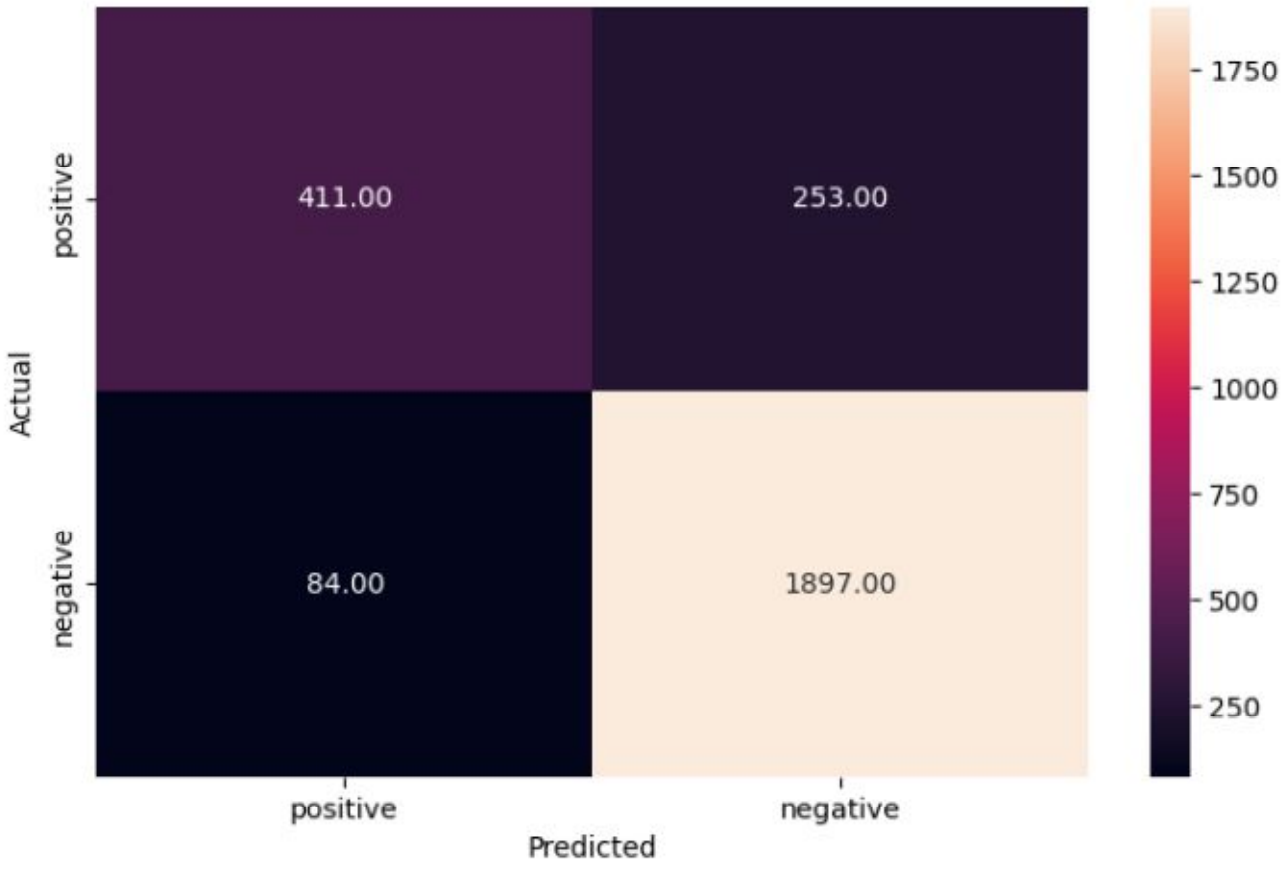
RANDOM FOREST MODEL ON TF-IDF

In [86]:
```
# Training the best model and calculating accuracy on test data
rf_tfidf = RandomForestClassifier(random_state=0, n_jobs =-1)

rf_tfidf.fit(x_train, y_train)
```

Out[8

In [87]:
```
# Make Predictions
y_pred_tfidf = rf_tfidf.predict(x_test)

# Check the Metrics
metrics_score(y_test, y_pred_tfidf)
```

```
              precision    recall  f1-score   support

           0       0.83      0.62      0.71       664
           1       0.88      0.96      0.92      1981

    accuracy                           0.87      2645
   macro avg       0.86      0.79      0.81      2645
weighted avg       0.87      0.87      0.87      2645
```



In [ ]:

Visualization of the top 50 words used for the Model

In [88]:
```
#Instantiate the feature from the vectorizer
features = tfidf_vec.get_feature_names_out()

get_top50_words(rf_tfidf,features)
```

2. SUPPORT VECTOR MACHINE (SVM)

SUPPORT VECTOR MACHINE (SVM) ON TF-IDF

In [89]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

In [90]:
```
model = SVC(kernel='linear', random_state = 10)
model.fit(x_train, y_train)
#predicting output for test data
pred = model.predict(x_test)
```

In [91]:
```
#accuracy score
accuracy_score(y_test,pred)
```

Out[91]:

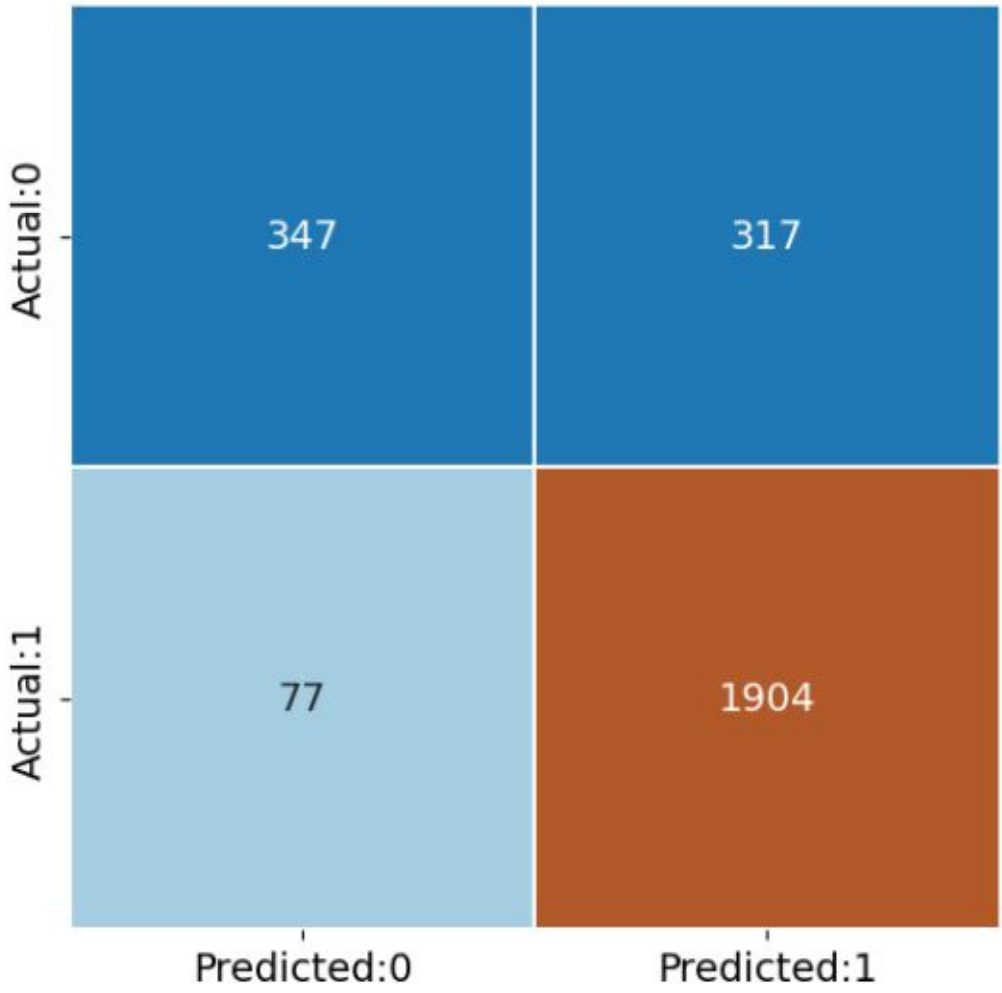
```
0.8510396975425331
```

In [92]:
```
#building confusion matrix
cm = confusion_matrix(y_test, pred)
cm
```

Out[92]:

```
array([[ 347,  317],
       [  77, 1904]], dtype=int64)
```

In [93]:
```
#defining the size of the canvas
plt.rcParams['figure.figsize'] = [6,6]

#confusion matrix to DataFrame
conf_matrix = pd.DataFrame(data = cm,columns = ['Predicted:0','Predicted:1'

#plotting the confusion matrix
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = 'Paired', cbar = F
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.show()
```



In [94]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.52      0.64       664
           1       0.86      0.96      0.91      1981

    accuracy                           0.85      2645
   macro avg       0.84      0.74      0.77      2645
weighted avg       0.85      0.85      0.84      2645
```

SUPPORT VECTOR MACHINE (SVM) ON BOW

In [95]:
```
#split the dataset
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
```

In [96]:
```
model = SVC(kernel='linear', random_state = 10)
model.fit(x_train, y_train)
#predicting output for test data
pred = model.predict(x_test)
```

In [97]:
```
#accuracy score
accuracy_score(y_test,pred)
```

Out[97]:

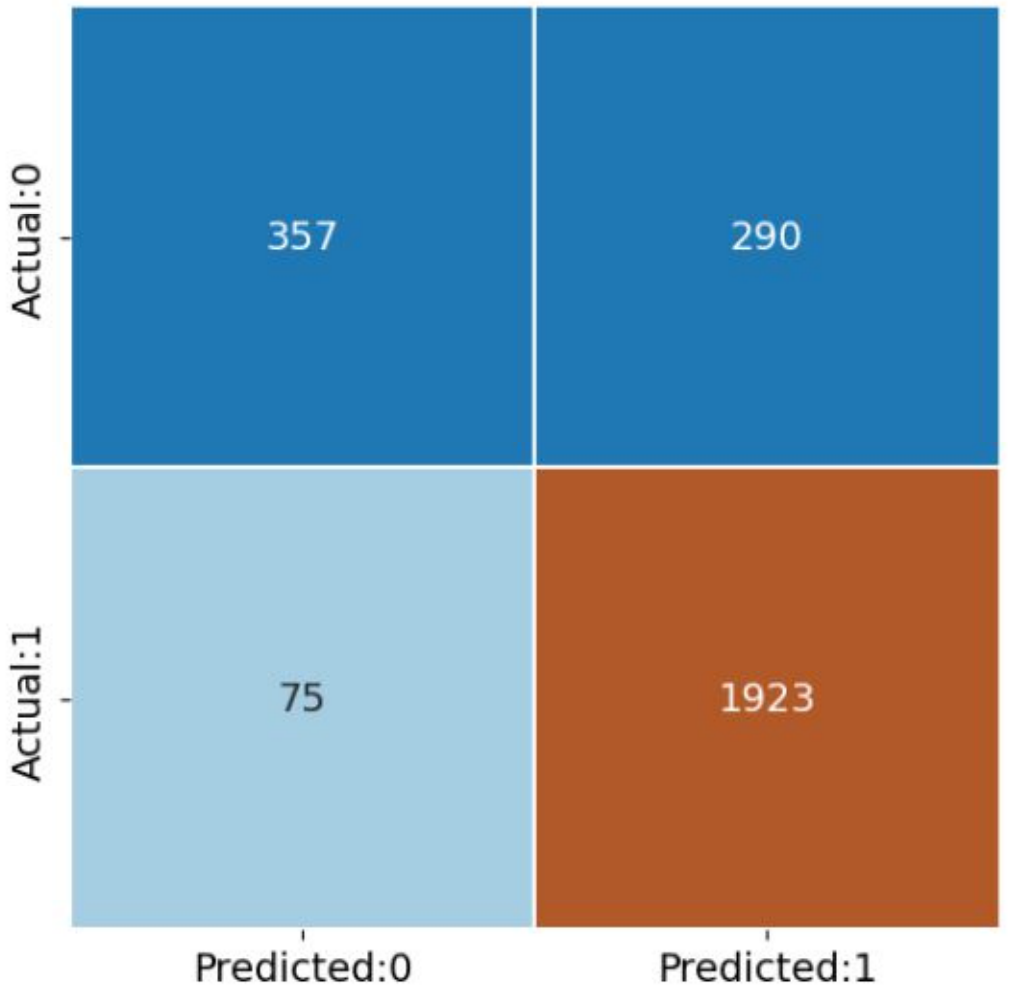
```
0.8620037807183365
```

In [98]:
```
#building confusion matrix
cm2 = confusion_matrix(y_test, pred)
cm2
```

Out[98]:

```
array([[ 357,  290],
       [  75, 1923]], dtype=int64)
```

In [99]:
```
#defining the size of the canvas
plt.rcParams['figure.figsize'] = [6,6]

#confusion matrix to DataFrame
conf_matrix = pd.DataFrame(data = cm2,columns = ['Predicted:0','Predicted:1

#plotting the confusion matrix
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = 'Paired', cbar = F
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.show()
```



In [100]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.55      0.66       647
           1       0.87      0.96      0.91      1998

    accuracy                           0.86      2645
   macro avg       0.85      0.76      0.79      2645
weighted avg       0.86      0.86      0.85      2645
```

DEEP LEARNING MODELS

1. LONG SHORT TERM MEMORY (LSTM)

In [101]:
```
# Import Libraries
!pip install tensorflow
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import SpatialDropout1D
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, SimpleRNN, LSTM, Bid
from sklearn.model_selection import train_test_split
```

```
[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: tensorflow in c:\users\kemi\appdata\local\progra
ms\python\python311\lib\site-packages (2.14.0)
Requirement already satisfied: tensorflow-intel==2.14.0 in c:\users\kemi\appdat
a\local\programs\python\python311\lib\site-packages (from tensorflow) (2.14.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\kemi\appdata\local\pr
ograms\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tenso
rflow) (2.0.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->t
ensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\kemi\appdata\lo
cal\programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0-
>tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\
kemi\appdata\local\programs\python\python311\lib\site-packages (from tensorflo
w-intel==2.14.0->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\kemi\appdata\loc
al\programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->
tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\kemi\appdata\local\progr
ams\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensorfl
ow) (3.10.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->ten
sorflow) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->ten
sorflow) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensor
flow) (1.24.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->t
ensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensorflo
w) (23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=
4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\kemi\appdata\local\programs\pyth
on\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (4.2
5.0)
Requirement already satisfied: setuptools in c:\users\kemi\appdata\local\progra
ms\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensorflo
w) (65.5.0)
Requirement already satisfied: six>=1.12.0 in c:\users\kemi\appdata\local\progr
ams\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->tensorfl
ow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from tensorflow-intel==2.14.0->ten
```

In [ ]:

In [102]:
```
df.head(5)
```

Out[10

	reviews	reviews_rating	sentime
0	happy product	negative	0
1	expected	negative	0
2	average product	positive	1
3	pic beautiful	positive	1
4	got damage product quality	positive	1

Tokenize and convert the dataframe into vectors

In [103]:
```
tokenizer = Tokenizer(num_words = 500)
```

In [104]:
```
tokenizer.fit_on_texts(df["reviews"].values)
```

In [105]:
```
X = tokenizer.texts_to_sequences(df["reviews"].values)
```

Showing the length of words

In [106]:
```
def countSentence(row):
    tokens = word_tokenize(row)
    length = len(tokens)
    return length
```

In [107]:
```
df['Sentence_length'] = df['reviews'].apply(lambda x: countSentence(x))
```

In [108]:
```
# Checking the max words
df['Sentence_length'].describe()
```

Out[108]:

```
count    8816.000000
mean        2.618081
std         1.705321
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        16.000000
Name: Sentence_length, dtype: float64
```

Checking the variance of the number of words in the customer reviews in order to determine the max length of words for the sequence paddings.

In [156]:
```
df[df['Sentence_length'] > 7].count()
```

Out[156]:

```
reviews            172
sentiment          172
Sentence_length    172
dtype: int64
```

Selecting max length of 7 for the padding

In [109]:
```
#padding the sequences
X = pad_sequences(X, maxlen = 7)
```

In [110]:
```
X
```

Out[110]:

```
array([[  0,   0,   0, ...,   0,  47,   2],
       [  0,   0,   0, ...,   0,   0,  33],
       [  0,   0,   0, ...,   0,  54,   2],
       ...,
       [  0,   0,   0, ...,   0,   5,   3],
       [  0,   0,   0, ...,   0,   0,   5],
       [  0,   0,   0, ..., 184,   3,  10]])
```

Labelling the data

In [111]:
```
y = df["sentiment"]
```

Splitting the dataset to Train and Test

In [112]:
```
#split the dataset
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
```

In [113]:
```
x_train.shape
```

Out[113]:

```
(6171, 7)
```

Categorizing the y_train

In [114]:
```
from tensorflow.keras.utils import to_categorical
```

In [115]:
```
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)
```

In [116]:
```
y_train_cat
```

Out[116]:

```
array([[1., 0.],
       [1., 0.],
       [0., 1.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

Building the Model

In [117]:
```
model= Sequential()
model.add(Embedding(500, 100, input_length = X.shape [1]))
model.add(SimpleRNN(100))
model.add(Dense(2, activation = "sigmoid"))

# Compiling the model
model.compile(optimizer = "adam", loss = "binary_crossentropy", metrics =
```

In [118]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 7, 100)            50000

 simple_rnn (SimpleRNN)      (None, 100)               20100

 dense (Dense)               (None, 2)                 202

=================================================================
Total params: 70302 (274.62 KB)
Trainable params: 70302 (274.62 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train the Model

Model 1

In [119]:
```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

In [120]:
```
es = EarlyStopping(patience = 3, verbose = 1)
```

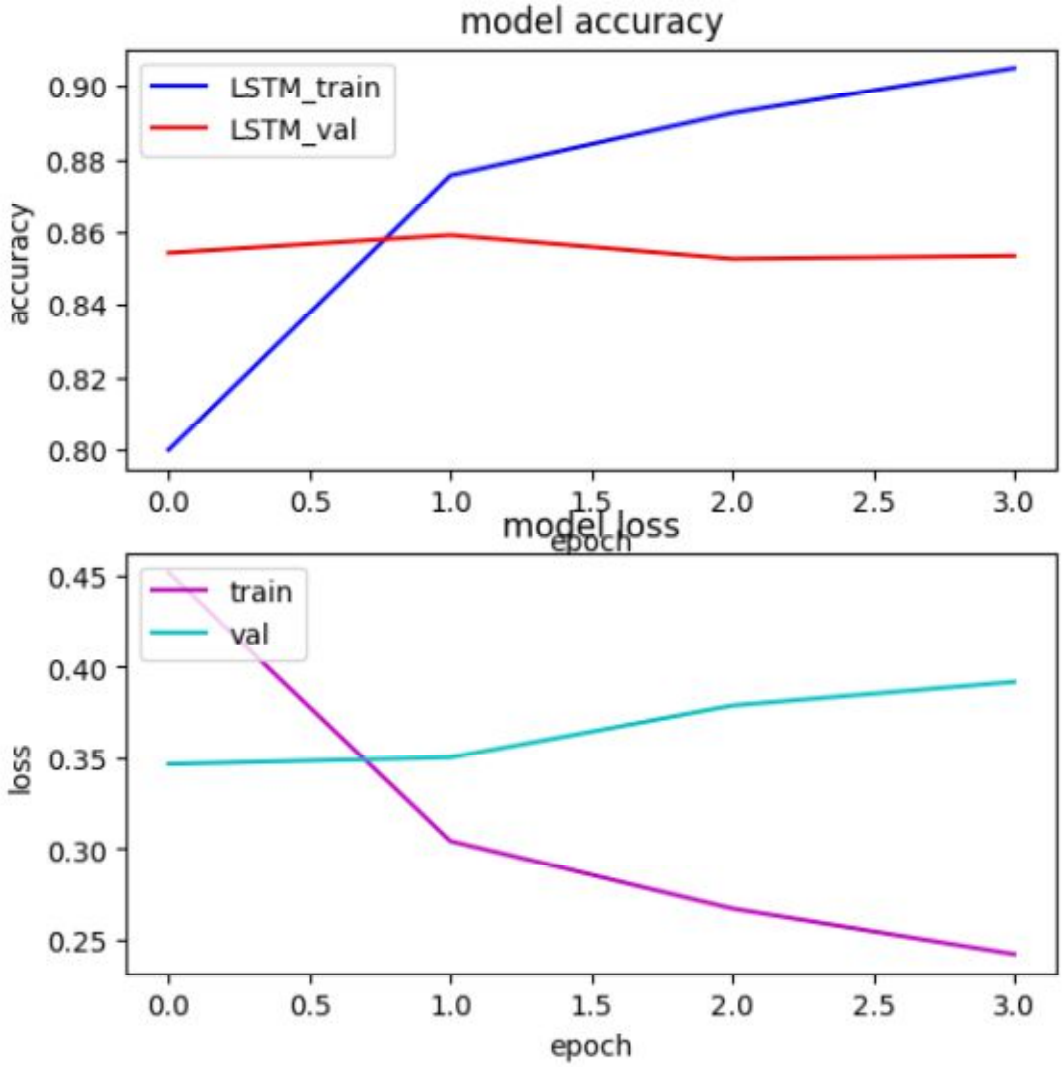
In [121]:
```
history = model.fit(x_train, y_train_cat, batch_size=32, epochs=50, valida
```

```
Epoch 1/50
155/155 [==============================] - 2s 6ms/step - loss: 0.4519 - accurac
y: 0.7998 - val_loss: 0.3466 - val_accuracy: 0.8543
Epoch 2/50
155/155 [==============================] - 1s 5ms/step - loss: 0.3043 - accurac
y: 0.8756 - val_loss: 0.3503 - val_accuracy: 0.8591
Epoch 3/50
155/155 [==============================] - 1s 5ms/step - loss: 0.2671 - accurac
y: 0.8926 - val_loss: 0.3789 - val_accuracy: 0.8526
Epoch 4/50
155/155 [==============================] - 1s 5ms/step - loss: 0.2421 - accurac
y: 0.9050 - val_loss: 0.3919 - val_accuracy: 0.8534
Epoch 4: early stopping
```

In [122]:
```
s, (at, al) = plt.subplots(2,1)
      at.plot(history.history['accuracy'], c= 'b')
      at.plot(history.history['val_accuracy'], c='r')
      at.set_title('model accuracy')
      at.set_ylabel('accuracy')
      at.set_xlabel('epoch')
      at.legend(['LSTM_train', 'LSTM_val'], loc='upper left')

      al.plot(history.history['loss'], c='m')
      al.plot(history.history['val_loss'], c='c')
      al.set_title('model loss')
      al.set_ylabel('loss')
      al.set_xlabel('epoch')
      al.legend(['train', 'val'], loc = 'upper left')
```

Out[122]:

```
<matplotlib.legend.Legend at 0x1f4a4004f10>
```



The LSTM model is slightly overfitting and has a little positive model bias, which indicates that it performs marginally better on the train dataset than the validation dataset.

In [ ]:

Model Improvement

Model 2

In [123]:
```
from tensorflow.keras.layers import Dropout
```

In [124]:
```
model2= Sequential()
      model2.add(Embedding(500, 100, input_length = X.shape [1]))
      model2.add(Dropout(0.2))
      model2.add(SimpleRNN(100))
      model2.add(Dense(2, activation = "sigmoid"))

      # Compiling the model
      model2.compile(optimizer = "adam", loss = "binary_crossentropy", metrics =
```

In [125]:
```
model2.summary()
```

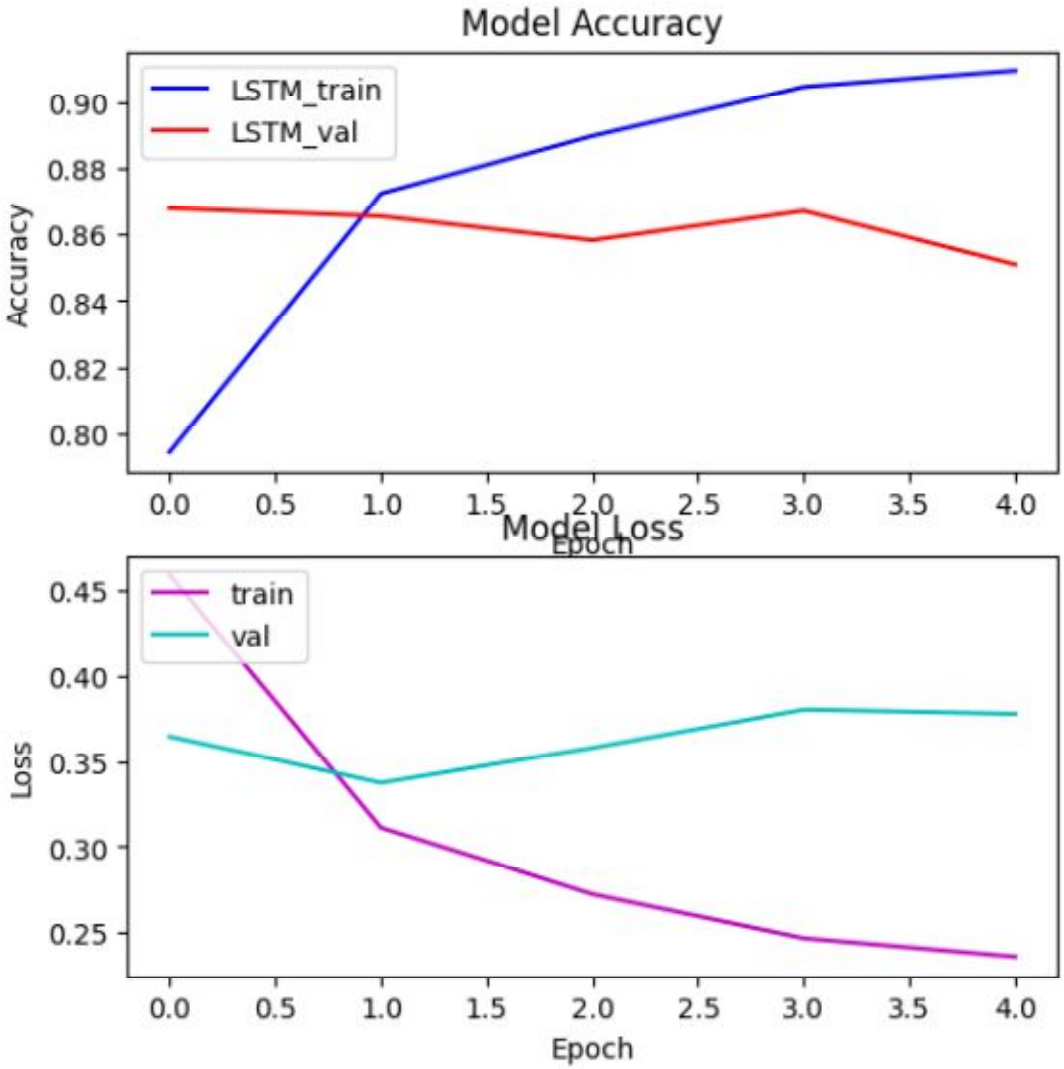
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 7, 100)            50000

 dropout (Dropout)           (None, 7, 100)            0

 simple_rnn_1 (SimpleRNN)    (None, 100)               20100

 dense_1 (Dense)             (None, 2)                 202

=================================================================
Total params: 70302 (274.62 KB)
Trainable params: 70302 (274.62 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [126]:
```
history2 = model2.fit(x_train, y_train_cat, batch_size=32, epochs=50, vali
```

```
Epoch 1/50
155/155 [==============================] - 3s 9ms/step - loss: 0.4592 - accurac
y: 0.7944 - val_loss: 0.3645 - val_accuracy: 0.8680
Epoch 2/50
155/155 [==============================] - 1s 6ms/step - loss: 0.3111 - accurac
y: 0.8724 - val_loss: 0.3374 - val_accuracy: 0.8656
Epoch 3/50
155/155 [==============================] - 1s 5ms/step - loss: 0.2722 - accurac
y: 0.8898 - val_loss: 0.3578 - val_accuracy: 0.8583
Epoch 4/50
155/155 [==============================] - 1s 6ms/step - loss: 0.2467 - accurac
y: 0.9046 - val_loss: 0.3802 - val_accuracy: 0.8672
Epoch 5/50
155/155 [==============================] - 1s 5ms/step - loss: 0.2359 - accurac
y: 0.9094 - val_loss: 0.3778 - val_accuracy: 0.8510
Epoch 5: early stopping
```

In [127]:
```
s, (at, al) = plt.subplots(2, 1,)

      # Plotting accuracy
      at.plot(history2.history['accuracy'], c='b', label='LSTM_train')
      at.plot(history2.history['val_accuracy'], c='r', label='LSTM_val')
      at.set_title('Model Accuracy')
      at.set_ylabel('Accuracy')
      at.set_xlabel('Epoch')
      at.legend(loc='upper left')

      # Plotting loss
      al.plot(history2.history['loss'], c='m', label='train')
      al.plot(history2.history['val_loss'], c='c', label='val')
      al.set_title('Model Loss')
      al.set_ylabel('Loss')
      al.set_xlabel('Epoch')
```

```
al.legend(loc='upper left')

plt.show()
```



The model is overfitting and there is no improvement by adding the Dropout hyperparameter

In [ ]:

2. Bidirectional Encoder Representations from Transformers (BERT)

DistilBERT

In [133]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (4.35.0)
Requirement already satisfied: filelock in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from transformers) (3.13.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in c:\users\kemi\ap
pdata\local\programs\python\python311\lib\site-packages (from transformers)
(0.17.3)
Requirement already satisfied: numpy>=1.17 in c:\users\kemi\appdata\local\progr
ams\python\python311\lib\site-packages (from transformers) (1.24.2)
Requirement already satisfied: packaging>=20.0 in c:\users\kemi\appdata\local\p
rograms\python\python311\lib\site-packages (from transformers) (23.0)
Requirement already satisfied: pyyaml>=5.1 in c:\users\kemi\appdata\local\progr
ams\python\python311\lib\site-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from transformers) (2023.10.3)
Requirement already satisfied: requests in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.15,>=0.14 in c:\users\kemi\appdata\
local\programs\python\python311\lib\site-packages (from transformers) (0.14.1)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from transformers) (0.4.0)
Requirement already satisfied: tqdm>=4.27 in c:\users\kemi\appdata\local\progra
ms\python\python311\lib\site-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec in c:\users\kemi\appdata\local\programs\p
ython\python311\lib\site-packages (from huggingface-hub<1.0,>=0.16.4->transform
ers) (2023.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\kemi\appd
ata\local\programs\python\python311\lib\site-packages (from huggingface-hub<
1.0,>=0.16.4->transformers) (4.8.0)
Requirement already satisfied: colorama in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\kemi\appdat
a\local\programs\python\python311\lib\site-packages (from requests->transformer
s) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from requests->transformers)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from requests->transformers) (20
23.7.22)

[notice] A new release of pip available: 22.3.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [134]:`df.head()`

Out[13

	reviews	reviews_rating	sentiment	Sentence_leng
0	happy product	negative	0	2
1	expected	negative	0	1
2	average product	positive	1	2
3	pic beautiful	positive	1	2
4	got damage product	positive	1	5

Labelling to select features and target

In [135]:
```
X_bert = df["reviews"]
y = df["sentiment"]
```

Split the dataset to Train and test

In [136]:
```
# Split data into training and testing set.
x_train, x_test, y_train, y_test = train_test_split(X_bert, y, test_size =
```

In [137]:
```
from transformers import DistilBertTokenizer, DistilBertForSequenceClassif
import torch
from torch.utils.data import TensorDataset, DataLoader, RandomSampler
```

In [138]:
```
# loading the BERT Tokenizer and Model

#token = df["reviews"]
tokenizer = DistilBertTokenizer.from_pretrained ("distilbert-base-uncased"
bert_model = DistilBertForSequenceClassification.from_pretrained ("distilb
```

```
Some weights of DistilBertForSequenceClassification were not initialized from t
he model checkpoint at distilbert-base-uncased and are newly initialized: ['pre
_classifier.bias', 'classifier.bias', 'classifier.weight', 'pre_classifier.weig
ht']
You should probably TRAIN this model on a down-stream task to be able to use i
t for predictions and inference.
```

In [139]:
```
x_train[0]
```

Out[139]:

```
'happy product'
```

In [140]:
```
sample_reviews = x_train[0]
inputs = tokenizer(sample_reviews, padding = True, truncation = True, retu
```

In [141]:
```
inputs
```

Out[141]:

```
{'input_ids': tensor([[ 101, 3407, 4031,  102]]), 'attention_mask': tensor
([[1, 1, 1, 1]])}
```

In [142]:
```
def preprocess_data(reviews, label, tokenizer):
    inputs = tokenizer(reviews.tolist(), padding = True, truncation = True,
    labels = torch.tensor(label.tolist())
    dataset = TensorDataset(inputs["input_ids"], inputs["attention_mask"], l
    return dataset
```

In [143]:
```
train_dataset = preprocess_data(x_train, y_train, tokenizer)
test_dataset = preprocess_data(x_test, y_test, tokenizer)
```

Loading the train dataset randomly using the RandomSampler

In [144]:
```
batch_size = 32
```

```
#Samplying the train data
train_sampler = RandomSampler(train_dataset)
train_dataloader = DataLoader(train_dataset, sampler = train_sampler, batc
```

In [ ]:

MODEL OPTIMIZATION AND EVALUATION

In [145]:
```
# Setting the optimizer (Adam)
optimizer = torch.optim.Adam(bert_model.parameters(), lr = 1e-5)
```

In [146]:
```
# Setting gradient to a specific time
gradient_accumulation = 10
```

In [147]:
```
#Setting the training loop
bert_model.train()
for epoch in range(2):
    print(f"===============Epoch: {epoch + 1}==============")
    total_loss = 0.0
    for step, batch in enumerate(train_dataloader):

#clear any gradient already accumulated
        optimizer.zero_grad()
        outputs = bert_model(batch[0], attention_mask = batch[1], labels =

#Calculating the loss
        loss = outputs.loss

#Scaling the loss with the gradient accumulation to normalize the loss
        loss = loss / gradient_accumulation

#taking the gradiuent of the loss
        loss.backward()
        if (step + 1) % gradient_accumulation == 0:
         optimizer.step()
         total_loss += loss.item()
         print (f"----------Adjusted weights after {step + 1} steps--------
    print(f"Epoch: {epoch + 1} - Average Loss{total_loss / len(train_datal
```

```
==============Epoch: 1==============
----------Adjusted weights after 10 steps----------
----------Adjusted weights after 20 steps----------
----------Adjusted weights after 30 steps----------
----------Adjusted weights after 40 steps----------
----------Adjusted weights after 50 steps----------
----------Adjusted weights after 60 steps----------
----------Adjusted weights after 70 steps----------
----------Adjusted weights after 80 steps----------
----------Adjusted weights after 90 steps----------
----------Adjusted weights after 100 steps----------
----------Adjusted weights after 110 steps----------
----------Adjusted weights after 120 steps----------
----------Adjusted weights after 130 steps----------
----------Adjusted weights after 140 steps----------
----------Adjusted weights after 150 steps----------
----------Adjusted weights after 160 steps----------
----------Adjusted weights after 170 steps----------
----------Adjusted weights after 180 steps----------
----------Adjusted weights after 190 steps----------
----------Adjusted weights after 200 steps----------
----------Adjusted weights after 210 steps----------
----------Adjusted weights after 220 steps----------
Epoch: 1 - Average Loss0.005838524331064785
==============Epoch: 2==============
----------Adjusted weights after 10 steps----------
----------Adjusted weights after 20 steps----------
----------Adjusted weights after 30 steps----------
----------Adjusted weights after 40 steps----------
----------Adjusted weights after 50 steps----------
----------Adjusted weights after 60 steps----------
----------Adjusted weights after 70 steps----------
----------Adjusted weights after 80 steps----------
----------Adjusted weights after 90 steps----------
----------Adjusted weights after 100 steps----------
----------Adjusted weights after 110 steps----------
----------Adjusted weights after 120 steps----------
----------Adjusted weights after 130 steps----------
----------Adjusted weights after 140 steps----------
----------Adjusted weights after 150 steps----------
----------Adjusted weights after 160 steps----------
----------Adjusted weights after 170 steps----------
----------Adjusted weights after 180 steps----------
----------Adjusted weights after 190 steps----------
----------Adjusted weights after 200 steps----------
----------Adjusted weights after 210 steps----------
----------Adjusted weights after 220 steps----------
Epoch: 2 - Average Loss0.004498216260342577
```

In [ ]:

In [148]:
```
bert_model.eval()
```

```
total_correct = 0
total_samples = 0

with torch.no_grad():
    for step, batch in enumerate(train_dataloader):

        # Forward pass
        outputs = bert_model(batch[0], attention_mask=batch[1], labels=bat
        logits = outputs.logits

        # Calculate accuracy
        _, predicted = torch.max(logits, 1)
        total_samples += batch[2].size(0)  # batch[2] contains the labels
        total_correct += (predicted == batch[2]).sum().item()

# Calculate accuracy
accuracy = total_correct / total_samples
print(f"Validation Accuracy: {accuracy}")
```

```
Validation Accuracy: 0.8006239364719229
```

With balanced F1-scores for both positive and negative classes, the Random Forest models demonstrate remarkable precision (0.89, 0.88) and recall (0.94, 0.96) rates for positive reviews when applied to both the Bag of Words (BOW) and TF-IDF representations. The precision-recall trade-off exhibited by the Support Vector Machine (SVM) model is balanced, and it achieves good accuracy rates in predicting sentiments, both positive and negative.

Excellent results can be seen in the Deep Learning models, particularly in the Long Short-Term Memory (LSTM) models with validation accuracy of 85%. The model shows a minor bias towards the training dataset. These results imply that the LSTM models are skilled at identifying sentiment trends and producing precise forecasts. With a validation accuracy of 80%, the DistilBERT model, an advanced transformer-based architecture, has exceptional performance. This demonstrates the model's capacity to classify sentiments accurately and highlights the value of pre-trained language models in sentiment analysis applications.

In [ ]: