MA336_AI & ML Project - Amazon Product Reviews (Sentimental Analysis)

In [1]:
```
# import libraries
import pandas as pd              # for data cleaning, reshaping, merging, and more.
import numpy as np               # for array operations, mathematical functions, linea

import nltk                      # import the Natural Language Toolkit (NLTK) for text
import re, random, os            # 're' module, for search, match, and manipulate stri

import matplotlib.pyplot as plt  # for MATLAB-like plotting framework including line p
import seaborn as sns            # for creating visually appealing statistical graphic
import math

!pip install spacy
import spacy                                # for basic preprocessing (optional)
!python -m spacy download en_core_web_sm    # for downloads the English language model,

from sklearn.model_selection import StratifiedShuffleSplit  # for test & training the

import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)   # for managing warning
```

```
Requirement already satisfied: spacy in c:\users\mantosh.nandy\anaconda3\lib\site-pac
kages (3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\users\mantosh.nandy
\anaconda3\lib\site-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\mantosh.nandy
\anaconda3\lib\site-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\mantosh.nandy\an
aconda3\lib\site-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\mantosh.nandy\anaconda
3\lib\site-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\mantosh.nandy\anacon
da3\lib\site-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in c:\users\mantosh.nandy\anaconda
3\lib\site-packages (from spacy) (8.2.3)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in c:\users\mantosh.nandy\anacond
a3\lib\site-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\mantosh.nandy\anaconda
3\lib\site-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\mantosh.nandy\anac
onda3\lib\site-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in c:\users\mantosh.nandy\anacond
a3\lib\site-packages (from spacy) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in c:\users\mantosh.nandy\anacond
a3\lib\site-packages (from spacy) (0.9.4)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in c:\users\mantosh.nandy\ana
conda3\lib\site-packages (from spacy) (5.2.1)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\mantosh.nandy\anaconda
3\lib\site-packages (from spacy) (4.65.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\mantosh.nandy\anac
onda3\lib\site-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in c:\users\manto
sh.nandy\anaconda3\lib\site-packages (from spacy) (1.10.8)
Requirement already satisfied: jinja2 in c:\users\mantosh.nandy\anaconda3\lib\site-pa
ckages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in c:\users\mantosh.nandy\anaconda3\lib\sit
e-packages (from spacy) (68.0.0)
Requirement already satisfied: packaging>=20.0 in c:\users\mantosh.nandy\anaconda3\li
b\site-packages (from spacy) (23.1)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\mantosh.nandy\anac
onda3\lib\site-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in c:\users\mantosh.nandy\anaconda3\lib
\site-packages (from spacy) (1.24.3)
Requirement already satisfied: typing-extensions>=4.2.0 in c:\users\mantosh.nandy\ana
conda3\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.7.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mantosh.nandy\ana
conda3\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mantosh.nandy\anaconda3\lib\s
ite-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mantosh.nandy\anaconda3
\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mantosh.nandy\anaconda3
\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2023.7.22)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in c:\users\mantosh.nandy\anaconda3
\lib\site-packages (from thinc<8.3.0,>=8.2.2->spacy) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\users\mantosh.nandy\ana
conda3\lib\site-packages (from thinc<8.3.0,>=8.2.2->spacy) (0.1.4)
Requirement already satisfied: colorama in c:\users\mantosh.nandy\anaconda3\lib\site-
packages (from tqdm<5.0.0,>=4.38.0->spacy) (0.4.6)
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:\users\mantosh.nandy\anaconda
3\lib\site-packages (from typer<0.10.0,>=0.3.0->spacy) (8.0.4)
```

```
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in c:\users\mantosh.nandy
\anaconda3\lib\site-packages (from weasel<0.4.0,>=0.1.0->spacy) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mantosh.nandy\anaconda3\li
b\site-packages (from jinja2->spacy) (2.1.1)
ERROR: Invalid requirement: '#'
```

In [2]:
```
mn_data = pd.read_csv('C:/Users/Mantosh.Nandy/OneDrive/Documents/MA 336 AI_ML/archive/
print(mn_data.shape)           # Printing the shape (number of rows & columns) of the
```

```
(1597, 27)
```

In [3]:
```
mn_data.info()       # overview of the DataFrame's structure and characteristics for a
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1597 entries, 0 to 1596
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    1597 non-null   object
 1   asins                 1597 non-null   object
 2   brand                 1597 non-null   object
 3   categories            1597 non-null   object
 4   colors                774 non-null    object
 5   dateAdded             1597 non-null   object
 6   dateUpdated           1597 non-null   object
 7   dimension             565 non-null    object
 8   ean                   898 non-null    float64
 9   keys                  1597 non-null   object
 10  manufacturer          965 non-null    object
 11  manufacturerNumber    902 non-null    object
 12  name                  1597 non-null   object
 13  prices                1597 non-null   object
 14  reviews.date          1217 non-null   object
 15  reviews.doRecommend   539 non-null    object
 16  reviews.numHelpful    900 non-null    float64
 17  reviews.rating        1177 non-null   float64
 18  reviews.sourceURLs    1597 non-null   object
 19  reviews.text          1597 non-null   object
 20  reviews.title         1580 non-null   object
 21  reviews.userCity      0 non-null      float64
 22  reviews.userProvince  0 non-null      float64
 23  reviews.username      1580 non-null   object
 24  sizes                 0 non-null      float64
 25  upc                   898 non-null    float64
 26  weight                686 non-null    object
dtypes: float64(7), object(20)
memory usage: 337.0+ KB
```

In [4]:
```
des_data=mn_data.copy()
des_data.describe()
```

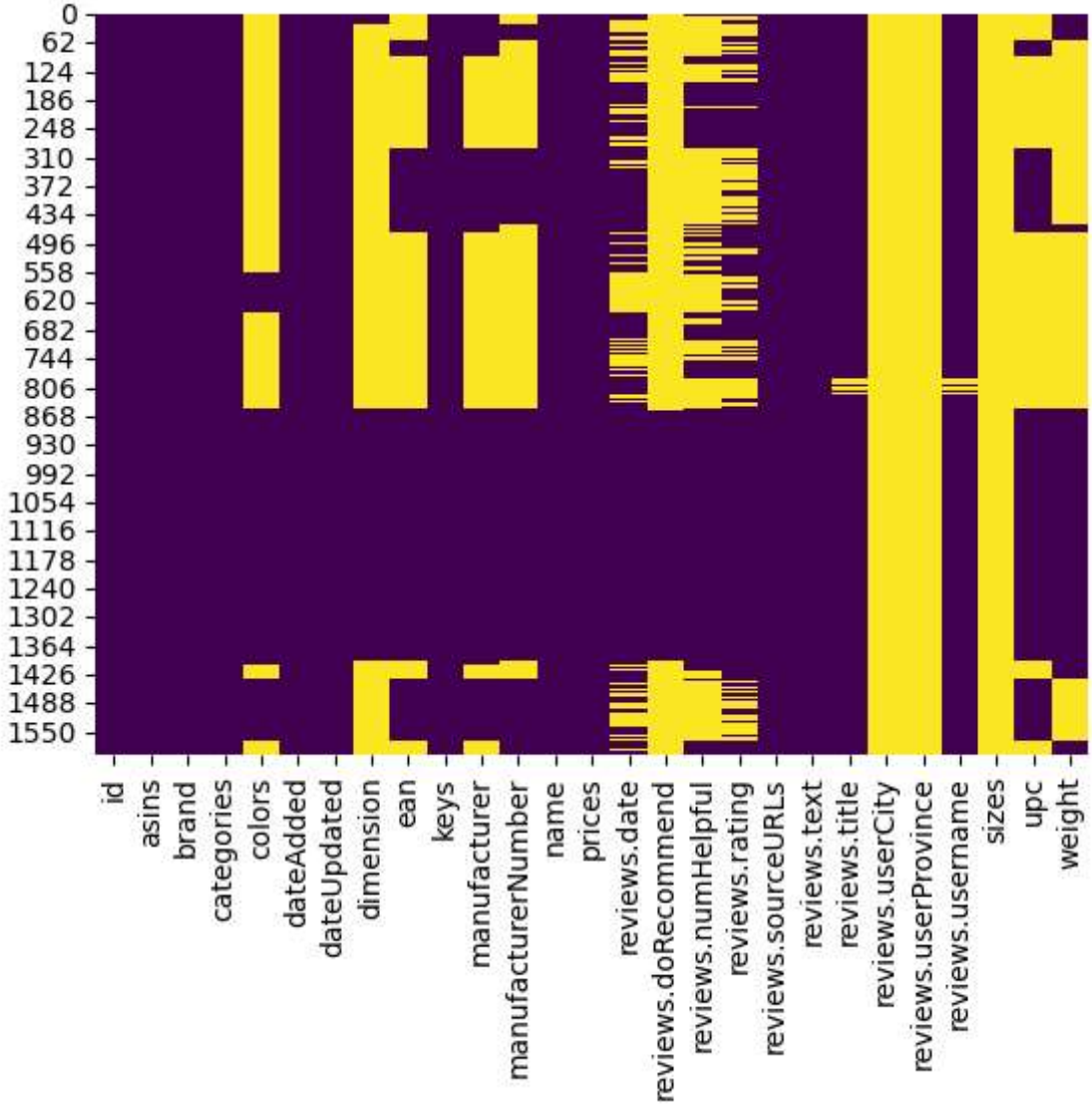
Out[4]:

	ean	reviews.numHelpful	reviews.rating	reviews.userCity	reviews.userProvince	sizes
count	8.980000e+02	900.000000	1177.000000	0.0	0.0	0.0
mean	8.443135e+11	83.584444	4.359388	NaN	NaN	NaN
std	3.416444e+09	197.150238	1.021445	NaN	NaN	NaN
min	8.416670e+11	0.000000	1.000000	NaN	NaN	NaN
25%	8.416670e+11	0.000000	4.000000	NaN	NaN	NaN
50%	8.416670e+11	0.000000	5.000000	NaN	NaN	NaN
75%	8.487190e+11	34.000000	5.000000	NaN	NaN	NaN
max	8.487190e+11	997.000000	5.000000	NaN	NaN	NaN

In [5]:
```
# From Upper analysis we found mean reviews rating comes out to be 4.35.
# Also rating info and review helpful rating is important to analyse data later.
```

In [6]:
```
# Heatmap analysis
sns.heatmap(des_data.isnull(),cbar=False,cmap="viridis")
```

Out[6]:
```
<Axes: >
```



In [7]:
```
mn_data.drop(['reviews.userCity','reviews.userProvince','sizes','ean'],inplace=True,ax
mn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1597 entries, 0 to 1596
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    1597 non-null   object 
 1   asins                 1597 non-null   object 
 2   brand                 1597 non-null   object 
 3   categories            1597 non-null   object 
 4   colors                774 non-null    object 
 5   dateAdded             1597 non-null   object 
 6   dateUpdated           1597 non-null   object 
 7   dimension             565 non-null    object 
 8   keys                  1597 non-null   object 
 9   manufacturer          965 non-null    object 
 10  manufacturerNumber    902 non-null    object 
 11  name                  1597 non-null   object 
 12  prices                1597 non-null   object 
 13  reviews.date          1217 non-null   object 
 14  reviews.doRecommend   539 non-null    object 
 15  reviews.numHelpful    900 non-null    float64
 16  reviews.rating        1177 non-null   float64
 17  reviews.sourceURLs    1597 non-null   object 
 18  reviews.text          1597 non-null   object 
 19  reviews.title         1580 non-null   object 
 20  reviews.username      1580 non-null   object 
 21  upc                   898 non-null    float64
 22  weight                686 non-null    object 
dtypes: float64(3), object(20)
memory usage: 287.1+ KB
```

In [8]:
```
mn_data.describe()
```

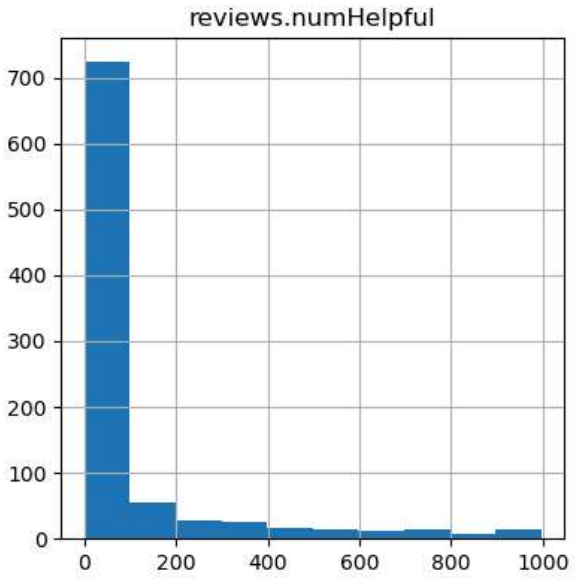
Out[8]:

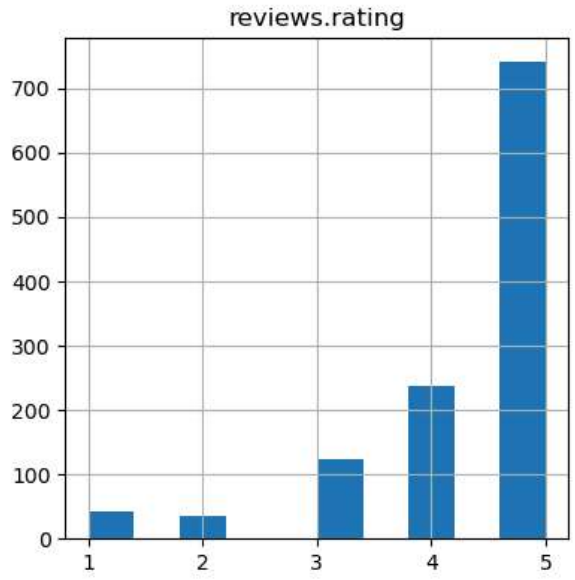
	reviews.numHelpful	reviews.rating	upc
count	900.000000	1177.000000	8.980000e+02
mean	83.584444	4.359388	8.443135e+11
std	197.150238	1.021445	3.416444e+09
min	0.000000	1.000000	8.416670e+11
25%	0.000000	4.000000	8.416670e+11
50%	0.000000	5.000000	8.416670e+11
75%	34.000000	5.000000	8.487190e+11
max	997.000000	5.000000	8.487190e+11

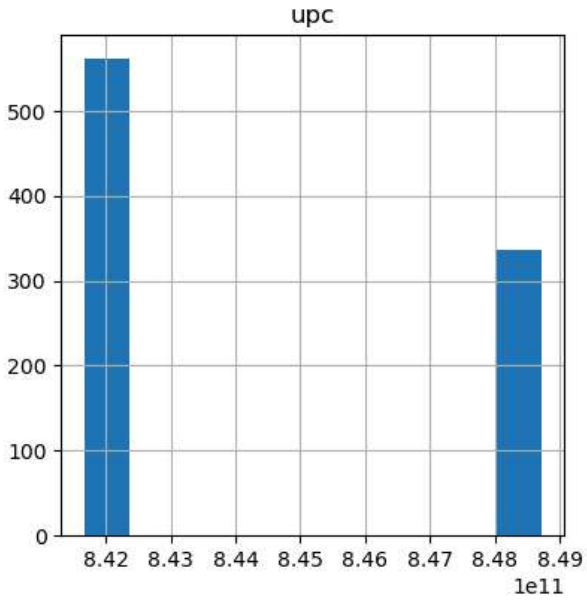
In [9]:
```
# # Calculate the number of unique asins using .unique() function
unique_asins = len(mn_data['asins'].unique())

# # Print the number of unique asins help of format method
print("Number of Unique asins: {}".format(unique_asins))
print(mn_data['asins'].unique())
```

```
Number of Unique asins: 54
['B00QJDU3KY' 'B002Y27P3M' 'B00DU15MU4' 'B01LW1MS9C' 'B01FWSVGQQ'
 'B00DOPNLJ0' 'B00NO8LX7E' 'B00LWHUAF0' 'B00KDRQEYQ' 'B00OQVZDJM'
 'B00QJDVBFU' 'B00VKLBU3Y' 'B010BWYP20' 'B00VKM5NFY' 'B01KIOU4EO'
 'B0117U8JSG' 'B01A08ECQY' 'B01J2G4VBG' 'B00QJDS7I4' 'B00U879XZ8'
 'B010EK1GOE' 'B00LWHUB9A' 'B00ZS0G0PG' 'B01A08E70K' 'B00PD81ETU'
 'B06XDD27LJ' 'B00CX5P8FC' 'B07194GPJV' 'B008GG93YE,B00LWHUBPO,B0051QYGXA'
 'B071NMTDHR' 'B0117U82EM' 'B0711C7WL2' 'B01M3ULMWP' 'B00LWHU9D8'
 'B0716JZKLT' 'B01BRWH8G8' 'B01M4NRFXX' 'B01KVZDFD2' 'B01M4NU4OM'
 'B008GGCAVM,B00DOPNMVM' 'B00KSR13CE' 'B01M71HRMY' 'B01K8B8YA8'
 'B010BWYDYA' 'B01KIOU214' 'B00DOPNK14'
 'B00DOPNO4M,B00BWYQ9YE,B00CYQPMJC,B00CUU1CGY,B00BWYRF7E,B00CYQP3AK'
 'B01BH83OOM' 'B00K5W9WZW' 'B01E9AHU8Q' 'B01HC1S9HU' 'B00HX0SRXW'
 'B00LORGAG6' 'B00NO8JJZW']
```

In [10]:
```
mn_data.hist(figsize=(10,10))
plt.show()
```







Train & Test Split with Stratified Shuffle Method

In [11]:
```
# # Drop the NULL values from the reviews.rating
mn_data_after = mn_data.dropna(subset=['reviews.rating'])

# # Cast the 'reviews.rating' column as integers
mn_data_after['reviews.rating'] = mn_data_after['reviews.rating'].astype(int)

# # Before and after pre-processing - Calculate and print the number of rows
print(f"Before: {len(mn_data)} \nAfter: {len(mn_data_after)}")
```

```
Before: 1597
After: 1177
C:\Users\Mantosh.Nandy\AppData\Local\Temp\ipykernel_10524\3309294776.py:5: SettingWit
hCopyWarning:
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/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  mn_data_after['reviews.rating'] = mn_data_after['reviews.rating'].astype(int)
```

In [12]:
```
# By using StratifiedShuffleSplit, start test & train the datasets
split=StratifiedShuffleSplit(n_splits=5,test_size=0.3, random_state=1)
for train_index,test_index in split.split(mn_data_after,mn_data_after["reviews.rating"
    strshf_train=mn_data_after.iloc[train_index]
    strshf_test=mn_data_after.iloc[test_index]
strshf_train.head()
```

Out[12]:

	id	asins	brand	categories	colors	dateAdded	dateU
1014	AVpfpK8KLJeJML43BCuD	B01BH83OOM	Amazon	Amazon Devices,Home,Smart Home & Connected Liv...	Black	2017-01-04T03:51:17Z	20 13T08
192	AV1T0-J7vKc47QAVgf2T	B01FWSVGQQ	Amazon	Amazon Devices	NaN	2017-07-18T03:53:01Z	20 13T08
464	AVsRjfwAU2_QcyX9PHqe	B01J2G4VBG	Amazon	Amazon Devices & Accessories,Amazon Device Acc...	NaN	2017-03-27T20:56:09Z	20 13T08
982	AVpfpK8KLJeJML43BCuD	B01BH83OOM	Amazon	Amazon Devices,Home,Smart Home & Connected Liv...	Black	2017-01-04T03:51:17Z	20 13T08
208	AV1T19jBvKc47QAVgf3S	B00OQVZDJM	Amazon	Amazon Devices	NaN	2017-07-18T03:57:20Z	20 18T23

5 rows × 23 columns

In [13]:
```
len(strshf_train) , len(strshf_test)
```

Out[13]:
```
(823, 354)
```

In [14]:
```
# Searching for get percentage of each of the ratings of training dataset

strshf_train["reviews.rating"].value_counts(normalize=True)    # normalized to represer
```

Out[14]:
```
reviews.rating
5    0.629405
4    0.200486
3    0.105711
1    0.035237
2    0.029162
Name: proportion, dtype: float64
```

In [15]:
```
# Searching for get percentage of each of the ratings of test dataset

strshf_test["reviews.rating"].value_counts(normalize=True)    # normalized to represer
```

Out[15]:
```
reviews.rating
5    0.629944
4    0.200565
3    0.104520
1    0.036723
2    0.028249
Name: proportion, dtype: float64
```

Exploratory Data Exploration on Training Data Set

In [16]:
```
reviews=strshf_train.copy()
reviews.head(2)
```

Out[16]:

	id	asins	brand	categories	colors	dateAdded	dateU
1014	AVpfpK8KLJeJML43BCuD	B01BH83OOM	Amazon	Amazon Devices,Home,Smart Home & Connected Liv...	Black	2017-01-04T03:51:17Z	20 13T08
192	AV1T0-J7vKc47QAVgf2T	B01FWSVGQQ	Amazon	Amazon Devices	NaN	2017-07-18T03:53:01Z	20 13T08

2 rows × 23 columns

In [17]:
```
len(reviews['name'].unique()),len(reviews['asins'].unique())
```

Out[17]:
```
(62, 54)
```

In [18]:
```
reviews.groupby('asins')['name'].unique()
```

Out[18]:

asins	
B002Y27P3M	
[Kindle Keyboard]	
B008GG93YE,B00LWHUBPO,B0051QYGXA	
[Kindle]	
B008GGCAVM,B00DOPNMVM	
[Kindle Fire HD 7"]	
B00CX5P8FC	
[Amazon Fire TV]	
B00DOPNK14	
[Kindle Paperwhite]	
B00DOPNLJ0	
[Kindle Fire HDX 8.9"]	
B00DOPNO4M,B00BWYQ9YE,B00CYQPMJC,B00CUU1CGY,B00BWYRF7E,B00CYQP3AK	
[Kindle Fire HDX 7"]	
B00DU15MU4	[Certified Refur
bished Amazon Fire TV (Previou...	
B00HX0SRXW	
[Amazon Premium Headphones]	
B00K5W9WZW	[Moshi Anti-Glar
e No Bubble Screen Protector f...	
B00KDRQEYQ	[Certi
fied Refurbished Kindle E-reader]	
B00KSR13CE	
[Kindle Paperwhite]	
B00LORGAG6	[Replacement
Remote for Amazon Fire TV Stick]	
B00LWHU9D8	
[Fire HD 6 Tablet]	
B00LWHUAF0	
[Fire HD 7 Tablet]	
B00LWHUB9A	
[Fire HDX 8.9 Tablet]	
B00NO8JJZW	[Alexa Voice Rem
ote for Amazon Fire TV and Fir...	
B00NO8LX7E	[Amazon Fire TV
Game Controller, All-New Amazo...	
B00OQVZDJM	[Ki
ndle Paperwhite E-reader - Black]	
B00PD81ETU	[Certified Refur
bished Kindle Voyage E-reader ...	
B00QJDS7I4	[Ki
ndle Paperwhite E-reader - Black]	
B00QJDU3KY	
[Kindle Paperwhite]	
B00QJDVBFU	
[Kindle Paperwhite 3G]	
B00U879XZ8	[Certified Refur
bished Kindle Paperwhite E-rea...	
B00VKLBU3Y	
[Fire HD 10 Tablet with Alexa]	
B00VKM5NFY	
[Fire HD 10 Tablet with Alexa]	
B00ZS0G0PG	[Kindle Oasis E-
reader with Leather Charging C...	
B010BWYDYA	
[Fire Tablet with Alexa]	
B010BWYP20	
[Fire Kids Edition Tablet]	
B010EK1GOE	[Kindle Oasis wi

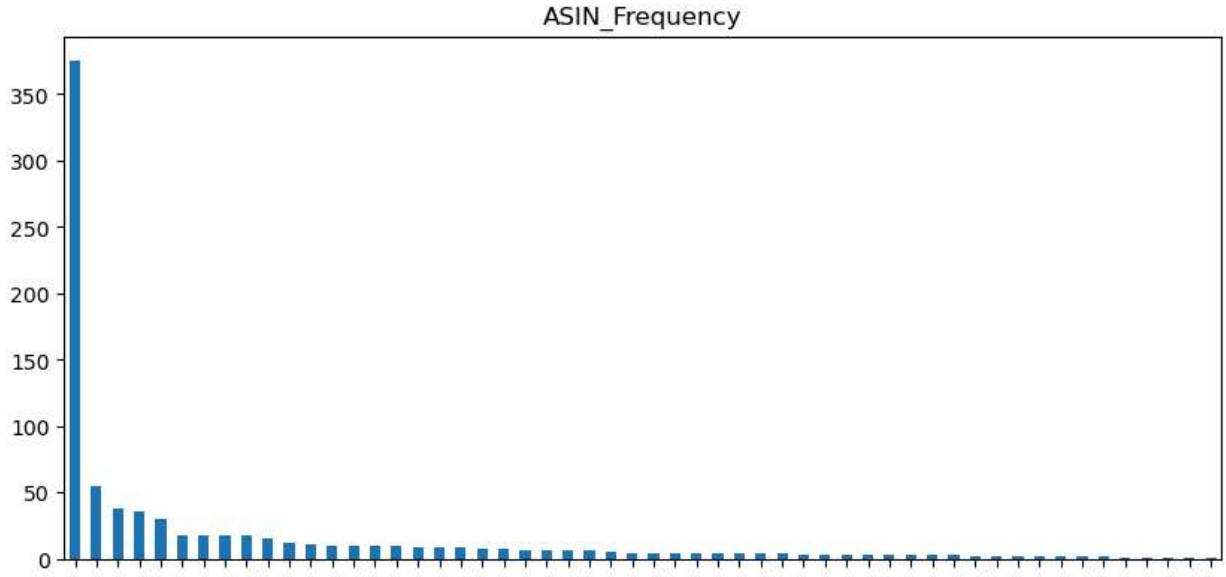
```
th Leather Charging Cover - Bl...
B0117U82EM
[Fire HD 8 Tablet]
B0117U8JSG                                                    [Certifi
ed Refurbished Fire HD 10 Tablet]
B01A08E70K
[Kindle E-reader - Black]
B01A08ECQY                                                  [Certified Ref
urbished Kindle E-reader - Black]
B01BH83OOM                                                [Amazon Tap - Al
exa-Enabled Portable Bluetooth...
B01BRWH8G8
[Fire HD 8 Tablet with Alexa]
B01E9AHU8Q                                                [Alexa Voice Rem
ote for Amazon Echo and Echo Dot]
B01FWSVGQQ                                                [Amazon Tap Slin
g Cover - White, Amazon Tap Sl...
B01HC1S9HU                                                [Amazon Kindle O
asis Premium Leather Battery C...
B01J2G4VBG                                                [Amazon 5W USB O
fficial OEM Charger and Power ...
B01K8B8YA8                                                              [E
cho Dot (2nd Generation) - Black]
B01KIOU214
[Amazon Echo - Black]
B01KIOU4EO
[Echo Show - Black]
B01KVZDFD2                                                [Kindle for Kids
Bundle with the latest Kindle...
B01LW1MS9C                                                [Amazon Echo Dot
Case (fits Echo Dot 2nd Gener...
B01M3ULMWP                                                           [All
-New Fire HD 8 Tablet with Alexa]
B01M4NRFXX                                                        [All-N
ew Fire HD 8 Kids Edition Tablet]
B01M4NU4OM                                                          [Al
l-New Fire 7 Kids Edition Tablet]
B01M71HRMY
[All-New Fire 7 Tablet with Alexa]
B06XDD27LJ                                                [Certified Refur
bished Echo Dot (2nd Generatio...
B0711C7WL2                                                [All-New Amazon
Kid-Proof Case for Amazon Fire...
B0716JZKLT                                                [All-New Amazon
Kid-Proof Case for Amazon Fire...
B07194GPJV                                                [All-New Amazon
Fire 7 Tablet Case (7th Genera...
B071NMTDHR                                                [All-New Amazon
Fire HD 8 Tablet Case (7th Gen...
Name: name, dtype: object
```

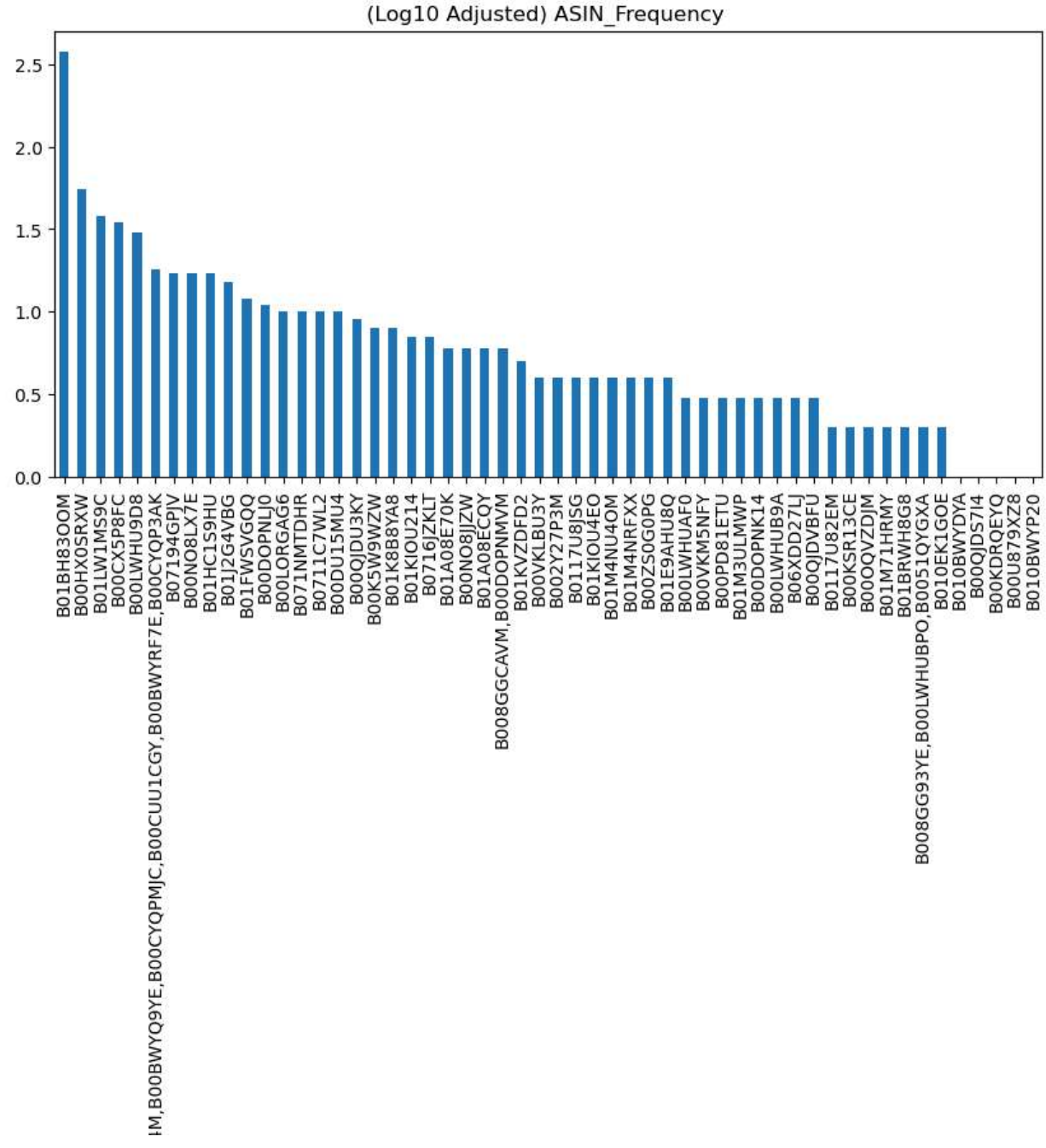
In [19]:
```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 823 entries, 1014 to 931
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   id                   823 non-null    object 
 1   asins                823 non-null    object 
 2   brand                823 non-null    object 
 3   categories           823 non-null    object 
 4   colors               468 non-null    object 
 5   dateAdded            823 non-null    object 
 6   dateUpdated          823 non-null    object 
 7   dimension            384 non-null    object 
 8   keys                 823 non-null    object 
 9   manufacturer         508 non-null    object 
 10  manufacturerNumber   485 non-null    object 
 11  name                 823 non-null    object 
 12  prices               823 non-null    object 
 13  reviews.date         666 non-null    object 
 14  reviews.doRecommend  373 non-null    object 
 15  reviews.numHelpful   615 non-null    float64
 16  reviews.rating       823 non-null    int32  
 17  reviews.sourceURLs   823 non-null    object 
 18  reviews.text         823 non-null    object 
 19  reviews.title        823 non-null    object 
 20  reviews.username     823 non-null    object 
 21  upc                  487 non-null    float64
 22  weight               445 non-null    object 
dtypes: float64(2), int32(1), object(20)
memory usage: 151.1+ KB
```

In [20]:

```
fig = plt.figure(figsize=(10,10))
# we use subplot when we have to see interrelationship btw two graphs
axis1 = plt.subplot(2,1,1)
axis2 = plt.subplot(2,1,2, sharex = axis1)
reviews["asins"].value_counts().plot(kind="bar", ax=axis1,
                                     title="ASIN_Frequency")

np.log10(reviews["asins"].value_counts()).plot(kind="bar", ax=axis2,
                                     title="(Log10 Adjusted) ASIN_Frequency ")
# np.log10 normalises our data to visualise the graph and difference much better
plt.show()
```

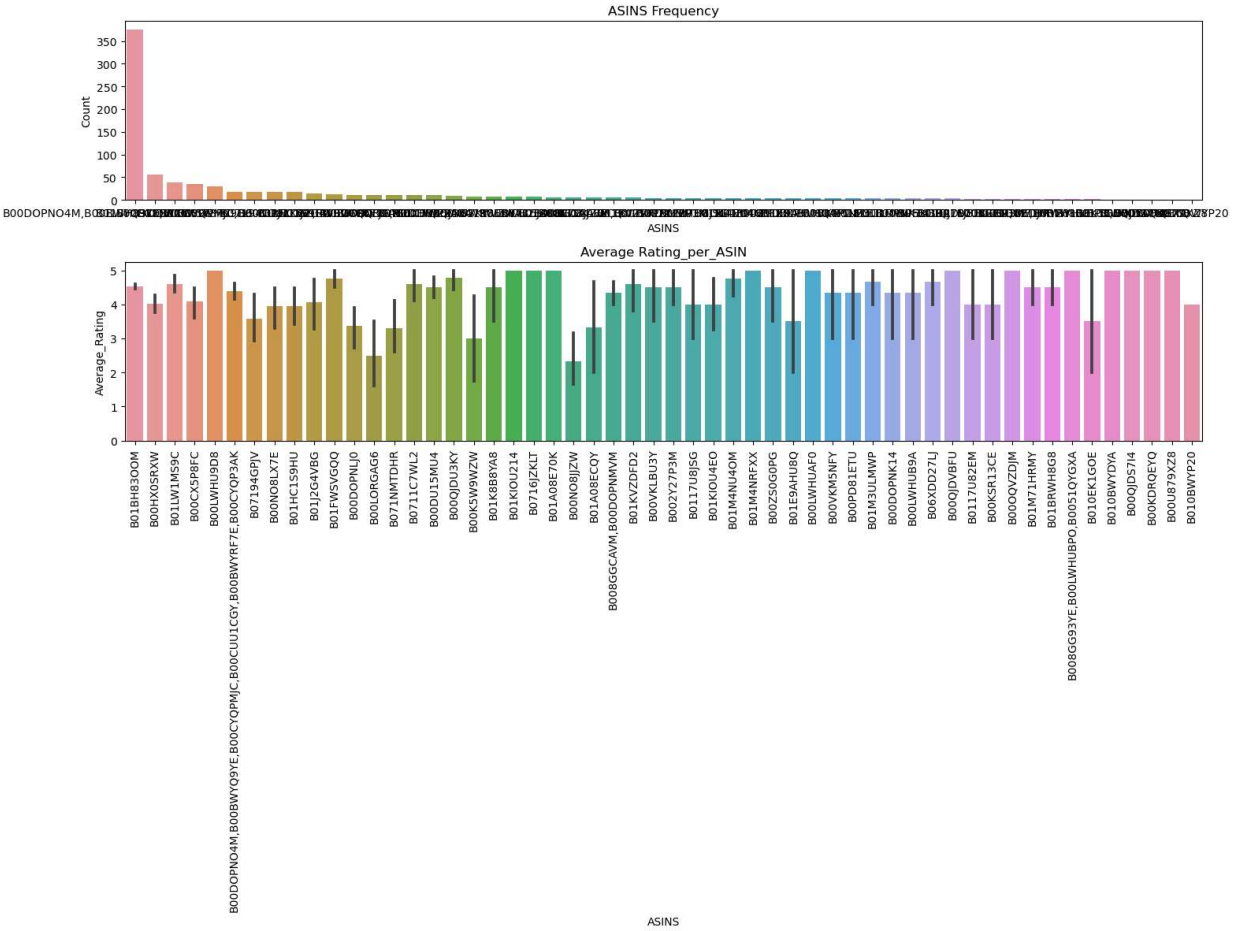

ASIN_Frequency
350
300
250
200
150
100
50
0


(Log10 Adjusted) ASIN_Frequency
2.5
2.0
1.5
1.0
0.5
0.0
B01BH83OOM
B00HX0SRXW
B01LW1MS9C
B00CX5P8FC
B00LWHU9D8
IM,B00BWYQ9YE,B00CYQPMJC,B00CUU1CGY,B00BWYRF7E,B00CYQP3AK
B07194GPJV
B00NO8LX7E
B01HC1S9HU
B01J2G4VBG
B01FWSVGQQ
B00DOPNLJ0
B00LORGAG6
B071NMTDHR
B0711C7WL2
B00DU15MU4
B00QJDU3KY
B00K5W9WZW
B01K8B8YA8
B01KIOU214
B0716JZKLT
B01A08E70K
B00NO8JJZW
B01A08ECQY
B008GGCAVM,B00DOPNMVM
B01KVZDFD2
B00VKLBU3Y
B002Y27P3M
B0117U8JSG
B01KIOU4EO
B01M4NU4OM
B01M4NRFXX
B00ZS0G0PG
B01E9AHU8Q
B00LWHUAF0
B00VKM5NFY
B00PD81ETU
B01M3ULMWP
B00DOPNK14
B00LWHUB9A
B06XDD27LJ
B00QJDVBFU
B0117U82EM
B00KSR13CE
B00OQVZDJM
B01M71HRMY
B01BRWH8G8
B008GG93YE,B00LWHUBPO,B0051QYGXA
B010EK1GOE
B010BWYDYA
B00QJDS7I4
B00KDRQEYQ
B00U879XZ8
B010BWYP20

In [21]:
```
# mean of rating from trianing data set
reviews['reviews.rating'].mean()
```

Out[21]:
```
4.359659781287971
```

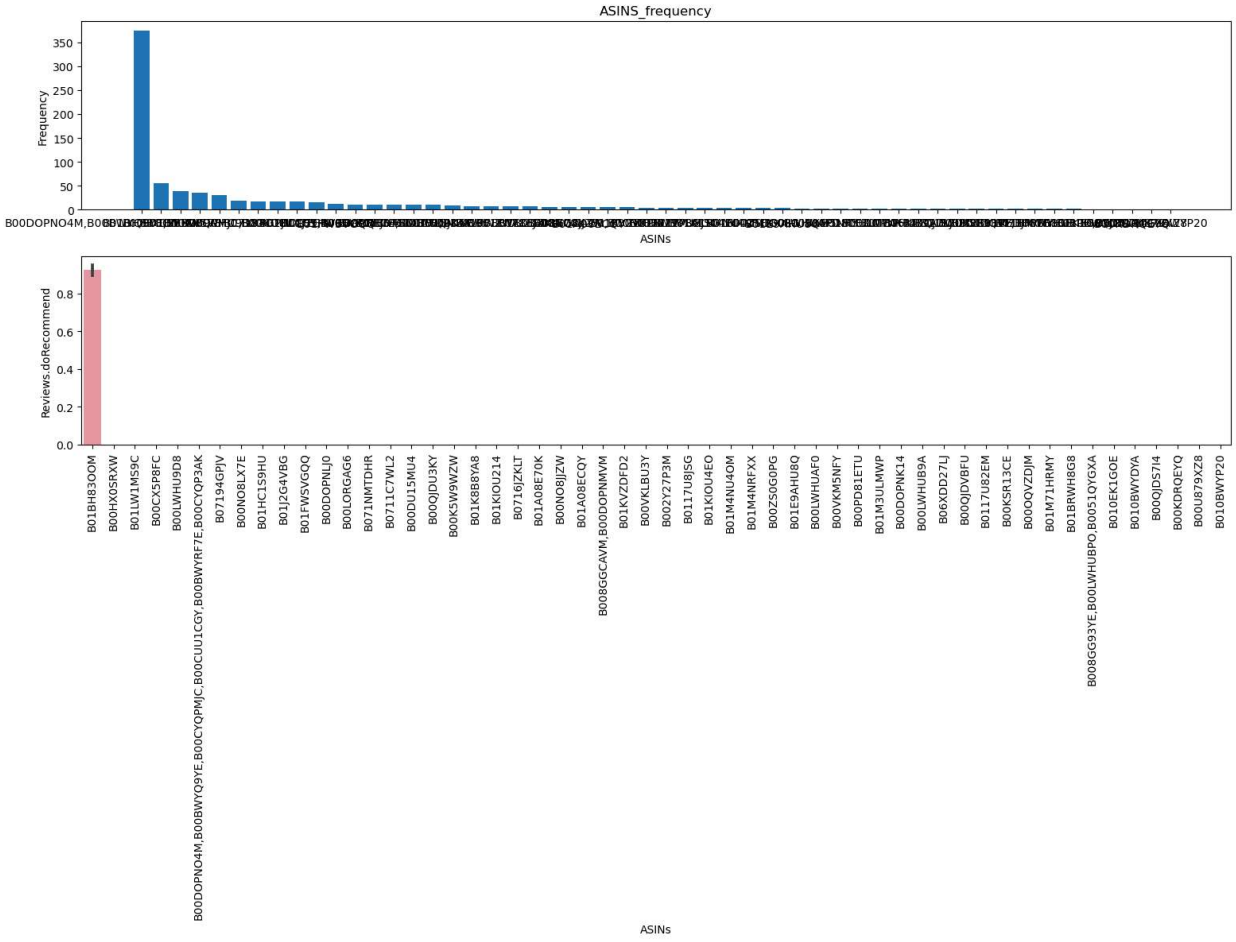
In [22]:
```
# Using pointplot to display number of those people giving ratings
count_asins=reviews['asins'].value_counts().index

# Settng up subplots
fig, axes = plt.subplots(2, 1, figsize=(16,12))

# Plotting ASINS frequency
sns.countplot(x='asins', data=reviews, order=count_asins, ax=axes[0])
axes[0].set_title('ASINS Frequency')
axes[0].set_xlabel('ASINS')
axes[0].set_ylabel('Count')

# Plotting average rating per ASIN
sns.barplot(x='asins', y='reviews.rating', data=reviews, order=count_asins, ax=axes[1]
axes[1].set_title('Average Rating_per_ASIN')
axes[1].set_xlabel('ASINS')
axes[1].set_ylabel('Average_Rating')
axes[1].tick_params(axis='x', rotation=90)

# Show plot
plt.tight_layout()
plt.show()
```



Reviews or Based on Recommendation Products Selling Strategy

In [23]:
```
asins_count = reviews['asins'].value_counts()

fig, ax1 = plt.subplots(2, 1, figsize=(16,12))

ax1[0].bar(asins_count.index, asins_count.values)
ax1[0].set_title('ASINS_frequency')
ax1[0].set_xlabel('ASINs')
ax1[0].set_ylabel('Frequency')

sns.barplot(x="asins", y="reviews.doRecommend", data=reviews, order=asins_count.index,
ax1[1].set_xlabel('ASINs')
ax1[1].set_ylabel('Reviews.doRecommend')
ax1[1].tick_params(axis='x', rotation=90)

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

```
C:\Users\Mantosh.Nandy\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeW
arning: Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\Mantosh.Nandy\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: Ru
ntimeWarning: All-NaN slice encountered
  return function_base._ureduce(a,
```



In [24]:
```
reviews_counts = reviews['asins'].value_counts().reset_index()
reviews_counts.columns = ['asins', 'counts']

avg_revs_rating = reviews.groupby("asins")['reviews.rating'].mean().reset_index()
avg_revs_rating.columns = ['asins', 'avg_revs_rating']

result = pd.merge(reviews_counts, avg_revs_rating, on='asins')

print(result)
```

	asins	counts	avg_revs_rating
0	B01BH83OOM	375	4.536000
1	B00HX0SRXW	55	4.018182
2	B01LW1MS9C	38	4.605263
3	B00CX5P8FC	35	4.085714
4	B00LWHU9D8	30	5.000000
5	B00DOPNO4M,B00BWYQ9YE,B00CYQPMJC,B00CUU1CGY,B0...	18	4.388889
6	B07194GPJV	17	3.588235
7	B00NO8LX7E	17	3.941176
8	B01HC1S9HU	17	3.941176
9	B01J2G4VBG	15	4.066667
10	B01FWSVGQQ	12	4.750000
11	B00DOPNLJ0	11	3.363636
12	B00LORGAG6	10	2.500000
13	B071NMTDHR	10	3.300000
14	B0711C7WL2	10	4.600000
15	B00DU15MU4	10	4.500000
16	B00QJDU3KY	9	4.777778
17	B00K5W9WZW	8	3.000000
18	B01K8B8YA8	8	4.500000
19	B01KIOU214	7	5.000000
20	B0716JZKLT	7	5.000000
21	B01A08E70K	6	5.000000
22	B00NO8JJZW	6	2.333333
23	B01A08ECQY	6	3.333333
24	B008GGCAVM,B00DOPNMVM	6	4.333333
25	B01KVZDFD2	5	4.600000
26	B00VKLBU3Y	4	4.500000
27	B002Y27P3M	4	4.500000
28	B0117U8JSG	4	4.000000
29	B01KIOU4EO	4	4.000000
30	B01M4NU4OM	4	4.750000
31	B01M4NRFXX	4	5.000000
32	B00ZS0G0PG	4	4.500000
33	B01E9AHU8Q	4	3.500000
34	B00LWHUAF0	3	5.000000
35	B00VKM5NFY	3	4.333333
36	B00PD81ETU	3	4.333333
37	B01M3ULMWP	3	4.666667
38	B00DOPNK14	3	4.333333
39	B00LWHUB9A	3	4.333333
40	B06XDD27LJ	3	4.666667
41	B00QJDVBFU	3	5.000000
42	B0117U82EM	2	4.000000
43	B00KSR13CE	2	4.000000
44	B00OQVZDJM	2	5.000000
45	B01M71HRMY	2	4.500000
46	B01BRWH8G8	2	4.500000
47	B008GG93YE,B00LWHUBPO,B0051QYGXA	2	5.000000
48	B010EK1GOE	2	3.500000
49	B010BWYDYA	1	5.000000
50	B00QJDS7I4	1	5.000000
51	B00KDRQEYQ	1	5.000000
52	B00U879XZ8	1	5.000000
53	B010BWYP20	1	4.000000

Sentiment Analysis

In [25]:
```
# 'mapping' dictionary helps to indictae the keys are the rating values and the values
def sentiments(rating):                          # declare sentiments function
    mapping = {5: "Positive", 4: "Positive", 3: "Neutral", 2: "Negative", 1: "Negative
    return mapping.get(rating, "Unknown")        #  get() method to retrieve the sentimen
                                                 #  absence of rating in the dictionary,
```

In [26]:
```
# Call sentiments function to training and testing dataset
strshf_train['Sentiments']=strshf_train['reviews.rating'].apply(sentiments)
strshf_test['Sentiments']=strshf_test['reviews.rating'].apply(sentiments)

strshf_train["Sentiments"][:20]
```

```
C:\Users\Mantosh.Nandy\AppData\Local\Temp\ipykernel_10524\2614189534.py:2: SettingWit
hCopyWarning:
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/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  strshf_train['Sentiments']=strshf_train['reviews.rating'].apply(sentiments)
C:\Users\Mantosh.Nandy\AppData\Local\Temp\ipykernel_10524\2614189534.py:3: SettingWit
hCopyWarning:
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/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  strshf_test['Sentiments']=strshf_test['reviews.rating'].apply(sentiments)
```

Out[26]:
```
1014    Positive
192     Positive
464     Positive
982     Positive
208     Positive
1424     Neutral
1489    Positive
1546     Neutral
1510     Neutral
649     Positive
1573     Neutral
1420    Positive
478      Neutral
1377    Positive
1389    Positive
493     Negative
108     Negative
1086    Positive
961     Positive
1260     Neutral
Name: Sentiments, dtype: object
```

Preparing text data

In [27]:
```
# Define variables
x_trn, x_trn_targetsentiment = strshf_train['reviews.text'], strshf_train['Sentiments'
x_tst, x_tst_targetsentiment = strshf_test['reviews.text'], strshf_test['Sentiments']
```

```
# Print Lengths
print(len(x_trn), len(x_tst))
```

```
823 354
```

Extracting features i.e., tokenizations , stopwords

In [28]:
```
# tokenisation involves breaks sentences into individual words
# stopwords: filtering unwanted words like the ,are , of ,is etc.

x_trn=x_trn.fillna(' ')              # filling na with space
x_tst=x_tst.fillna(' ')
x_trn_targetsentiment=x_trn_targetsentiment.fillna(' ')
x_tst_targetsentiment=x_tst_targetsentiment.fillna(' ')
```

In [29]:
```
# Using count vectorizer counting Text preprocessing and occurence

from sklearn.feature_extraction.text import CountVectorizer

# Fit and transform the x_train data
count_vector = CountVectorizer()
x_trn_counts = count_vector.fit_transform(x_trn)

# Print the shape of the matrix
print(x_trn_counts.shape)          # it dislays how many samples and how many distinct
```

```
(823, 4763)
```

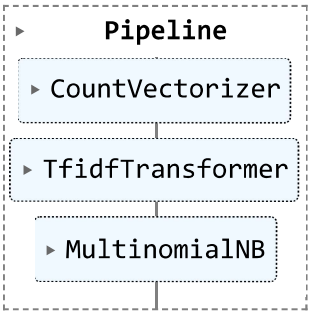
In [30]:
```
# Using tfidf transformer for reduces less meaning words which have higher occurence

from sklearn.feature_extraction.text import TfidfVectorizer      # Downscales stop wor

# Create a TfidfVectorizer instance
tfidf_vectorizer = TfidfVectorizer(use_idf=False)

# Fit-transform the training data
x_trn_tfidf = tfidf_vectorizer.fit_transform(x_trn)

# Get the shape of the transformed training data
x_trn_tfidf.shape
```

Out[30]:
```
(823, 4763)
```

In [31]:
```
from sklearn.naive_bayes import MultinomialNB             # Using Naive Bayes Algorith
from sklearn.pipeline import Pipeline                     # for multiple preprocessing
from sklearn.feature_extraction.text import TfidfTransformer   # helping to represent

# Declare the pipeline
data_multiNB_pipe = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf_nominalNB', MultinomialNB())
]) # using pipeline from sklearn helps a number of tasks to be implemented on every da
   # also be used when to implement multiple models on dataset in sequential manner
```

```
# Fit the pipeline on the training data
data_multiNB_pipe.fit(x_trn, x_trn_targetsentiment)
```

Out[31]:



In [32]:

```
prediction_multiNB=data_multiNB_pipe.predict(x_tst)
prediction_multiNB
```

```
Out[32]: array(['Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Neutral',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Neutral', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Neutral', 'Positive', 'Neutral', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Neutral', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Neutral', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Neutral', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
                'Neutral', 'Positive', 'Positive', 'Positive', 'Positive',
                'Neutral', 'Positive', 'Positive', 'Positive', 'Positive',
```

```
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Neutral', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Neutral',
       'Positive', 'Positive', 'Positive', 'Positive'], dtype='<U8')
```

In [33]:
```
# Perform accuracy using MultinomialNB model

from sklearn.metrics import accuracy_score
print("Accuracy: {}".format(accuracy_score(x_tst_targetsentiment,prediction_multiNB)))
```

```
Accuracy: 0.8615819209039548
```

In [34]:
```
x_tst
```

Out[34]:
```
3       I bought one of the first Paperwhites and have...
80      I am not a casual user of on-demand content an...
500     I got my first Kindle. This is a nice ebook re...
534     In this day and age of rectangles with screens...
101     This is perfect. A lot of people were commenti...
                              ...
1559    While I've purchased items from Amazon for yea...
454     According to the info, the Paperwhite charges ...
381     First I would like to say that I am coming fro...
1287    I have the Echo and I just love it... I bought...
698     Love it! I purchased this as I have had audibl...
Name: reviews.text, Length: 354, dtype: object
```

Applying model on input text

In [35]:
```
 # Perform accuracy based on Linear Support Vector Classifier

from sklearn.svm import LinearSVC
text_clf = Pipeline([("vect", CountVectorizer()),
                                ("tfidf", TfidfTransformer()),
                                ("clf_linearSVC", LinearSVC())])
text_clf.fit(x_trn, x_trn_targetsentiment)

predicted_LinearSVC = text_clf.predict(x_tst)
print('Accuracy: {}'. format(accuracy_score(x_tst_targetsentiment, predicted_LinearSVC
```

```
Accuracy: 0.884180790960452
```

```
C:\Users\Mantosh.Nandy\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:32: Future
Warning: The default value of `dual` will change from `True` to ``'auto'`` in 1.5. Set
the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

In [36]:
```
test_text=["it works well it takes time for it to know you",
            "Built on Android does not mean you can",
              "I will not recomend this porduct",
          ]
text_clf.predict(test_text)
```

Out[36]:
```
array(['Positive', 'Positive', 'Negative'], dtype=object)
```

In [ ]:

In [ ]: