import library

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
import string
import nltk
from nltk.stem import PorterStemmer
import warnings #But we need to hide these warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore', category=DeprecationWarning)
warnings.filterwarnings("ignore",category=UserWarning)
sns.set_style("whitegrid")
%matplotlib inline
np.random.seed(7)
```

In [2]:
```
 df=pd.read_csv('C:/Users/User/Desktop/PROJECTS/Amazon Product Review/1429_1.csv')
print(df)
```

```
                         id  \
0      AVqkIhwDv8e3D1O-lebb
1      AVqkIhwDv8e3D1O-lebb
2      AVqkIhwDv8e3D1O-lebb
3      AVqkIhwDv8e3D1O-lebb
4      AVqkIhwDv8e3D1O-lebb
...                     ...
34655  AVpfiBlyLJeJML43-4Tp
34656  AVpfiBlyLJeJML43-4Tp
34657  AVpfiBlyLJeJML43-4Tp
34658  AVpfiBlyLJeJML43-4Tp
34659  AVpfiBlyLJeJML43-4Tp

                                                    name       asins   brand  \
0      All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...  B01AHB9CN2  Amazon
1      All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...  B01AHB9CN2  Amazon
2      All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...  B01AHB9CN2  Amazon
3      All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...  B01AHB9CN2  Amazon
4      All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...  B01AHB9CN2  Amazon
```

In [3]:
```
print(len(df))
```

```
34660
```

In [4]:
```
df.shape
```

Out[4]:
```
(34660, 21)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'name', 'asins', 'brand', 'categories', 'keys', 'manufacturer',
       'reviews.date', 'reviews.dateAdded', 'reviews.dateSeen',
       'reviews.didPurchase', 'reviews.doRecommend', 'reviews.id',
       'reviews.numHelpful', 'reviews.rating', 'reviews.sourceURLs',
       'reviews.text', 'reviews.title', 'reviews.userCity',
       'reviews.userProvince', 'reviews.username'],
      dtype='object')
```

In [6]:
```
df.head()
```

Out[6]:

	id	name	asins	brand	categories	keys	manufac
0	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	An
1	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	An
	AVqkIhwDv8e3D1O-	All-New Fire HD 8			Electronics,iPad & Tablets,All		

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34660 entries, 0 to 34659
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    34660 non-null  object 
 1   name                  27900 non-null  object 
 2   asins                 34658 non-null  object 
 3   brand                 34660 non-null  object 
 4   categories            34660 non-null  object 
 5   keys                  34660 non-null  object 
 6   manufacturer          34660 non-null  object 
 7   reviews.date          34621 non-null  object 
 8   reviews.dateAdded     24039 non-null  object 
 9   reviews.dateSeen      34660 non-null  object 
 10  reviews.didPurchase   1 non-null      object 
 11  reviews.doRecommend   34066 non-null  object 
 12  reviews.id            1 non-null      float64
 13  reviews.numHelpful    34131 non-null  float64
 14  reviews.rating        34627 non-null  float64
 15  reviews.sourceURLs    34660 non-null  object 
 16  reviews.text          34659 non-null  object 
 17  reviews.title         34654 non-null  object 
 18  reviews.userCity      0 non-null      float64
 19  reviews.userProvince  0 non-null      float64
 20  reviews.username      34653 non-null  object 
dtypes: float64(5), object(16)
memory usage: 5.6+ MB
```

In [8]:
```
df.dtypes
```

Out[8]:
```
id                       object
name                     object
asins                    object
brand                    object
categories               object
keys                     object
manufacturer             object
reviews.date             object
reviews.dateAdded        object
reviews.dateSeen         object
reviews.didPurchase      object
reviews.doRecommend      object
reviews.id              float64
reviews.numHelpful      float64
reviews.rating          float64
reviews.sourceURLs       object
reviews.text             object
reviews.title            object
reviews.userCity        float64
reviews.userProvince    float64
```

In [9]:
```
df.tail()
```

Out[9]:

	id	name	asins	brand	categories	keys	manufactur
34655	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Amaz Digi Services, I
34656	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Amaz Digi Services, I
34657	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Amaz Digi Services, I
34658	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Amaz Digi Services, I
34659	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Amaz Digi Services, I

5 rows × 21 columns

In [10]:
```
df.describe()
```

Out[10]:

	reviews.id	reviews.numHelpful	reviews.rating	reviews.userCity	reviews.userProvince
count	1.0	34131.000000	34627.000000	0.0	0.0
mean	111372787.0	0.630248	4.584573	NaN	NaN
std	NaN	13.215775	0.735653	NaN	NaN
min	111372787.0	0.000000	1.000000	NaN	NaN
25%	111372787.0	0.000000	4.000000	NaN	NaN
50%	111372787.0	0.000000	5.000000	NaN	NaN
75%	111372787.0	0.000000	5.000000	NaN	NaN
max	111372787.0	814.000000	5.000000	NaN	NaN

In [11]:
```
columns_to_remove = ['reviews.userCity', 'reviews.userProvince', 'reviews.id']
df = df.drop(columns=columns_to_remove, axis=1)
```

In [12]:
```
df
```

Out[12]:

	id	name	asins	brand	categories	keys	man
0	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
1	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
2	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
3	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
4	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
...	...	...	...	...	...	...	
34655	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34656	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34657	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34658	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34659	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se

34660 rows × 18 columns

Dealing with missing values

In [13]:
```
np.sum(df.isnull().any(axis=1))
```

Out[13]:
```
34659
```

In [14]:
```
print('Count of columns in the data is:  ', len(df.columns))
print('Count of rows in the data is:  ', len(df))
```

```
Count of columns in the data is:   18
Count of rows in the data is:   34660
```

In [15]:
```
df['reviews.rating'].unique()
```

Out[15]:
```
array([ 5.,  4.,  2.,  1.,  3., nan])
```

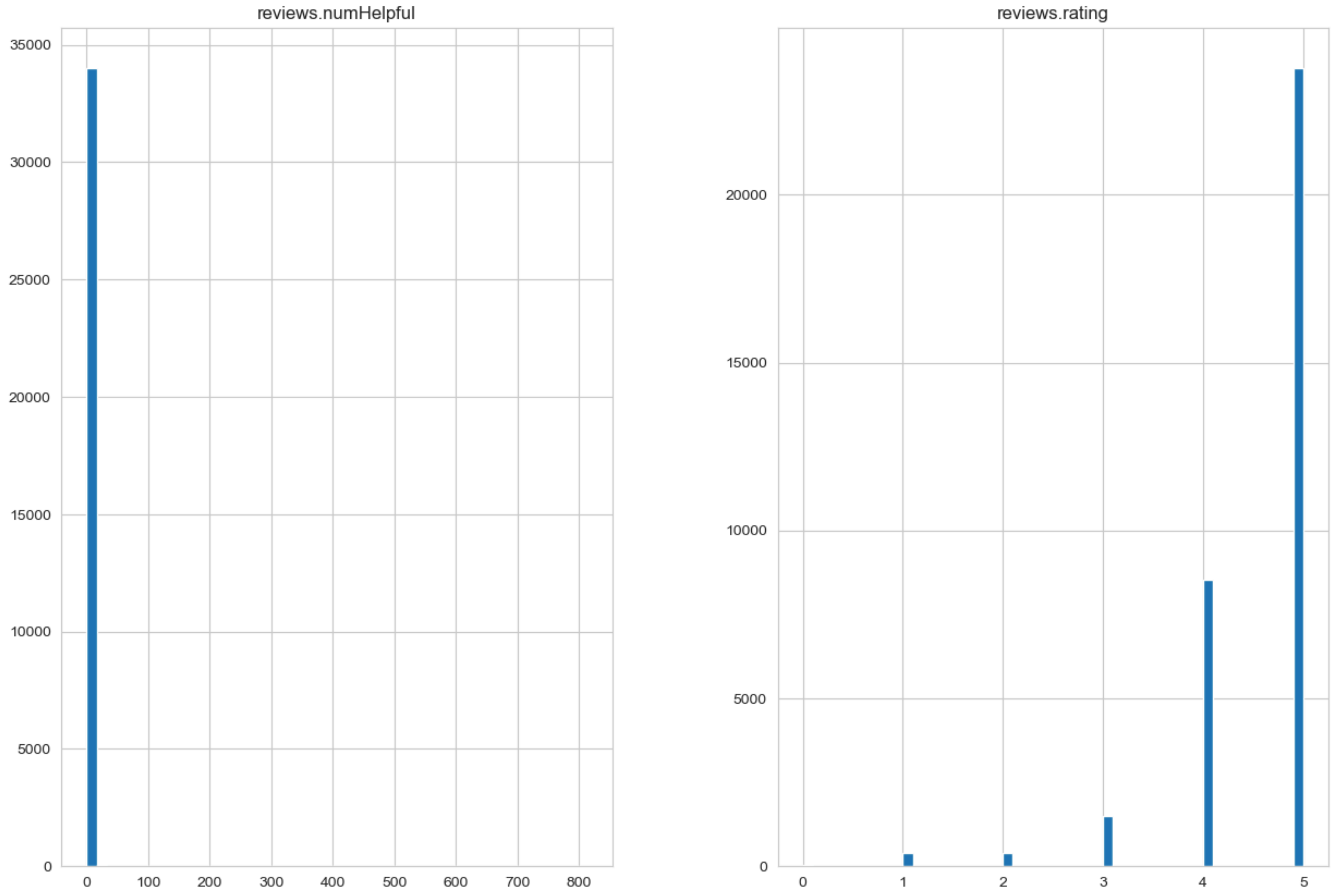
In [16]:
```
df['reviews.rating'].nunique()
```

Out[16]:
```
5
```

In [17]:
```
df['reviews.rating'].fillna(0, inplace=True)
print(df['reviews.rating'].unique())
```

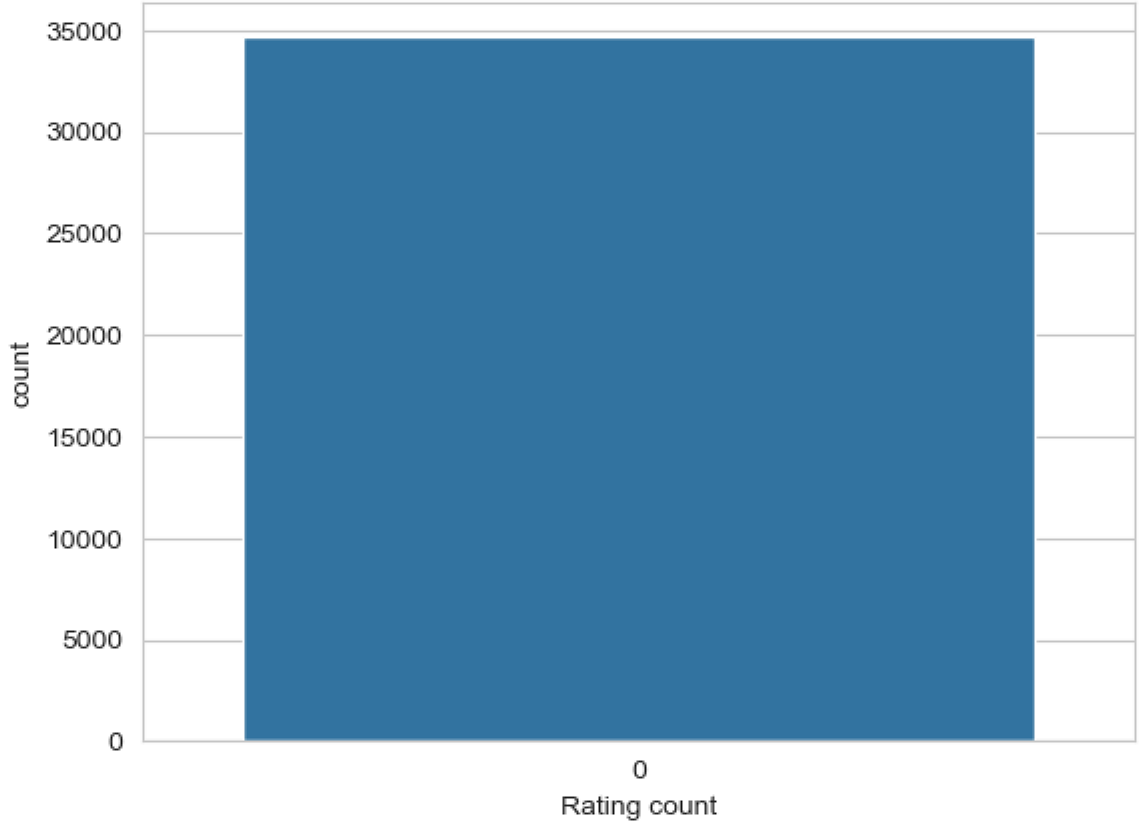
```
[5. 4. 2. 1. 3. 0.]
```

In [18]:
```
df.hist(bins=50, figsize=(15,10))
plt.show()
```



In [19]:
```
sns.countplot(df['reviews.rating'])
plt.xlabel('Rating count')
```

Out[19]:
```
Text(0.5, 0, 'Rating count')
```



Pre-processing Data

using stop words

In [20]:
```
from nltk.corpus import stopwords
df['reviews.text']=df['reviews.text'].str.lower()
# Download the stopwords corpus
nltk.download('stopwords')
```

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

Out[20]:
```
True
```

In [21]:
```
stopwords_list = stopwords.words('english')
```

In [22]:
```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

Out[22]:
```
"i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yo
urselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their,
theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be,
been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, unti
l, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, belo
w, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where,
why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, tha
n, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren,
aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, is
n't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't,
weren, weren't, won, won't, wouldn, wouldn't"
```

In [23]:
```
STOPWORDS = set(stopwords.words('english'))
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
df['reviews.text'] = df['reviews.text'].apply(lambda x: " ".join([word for word in str(x).split() if word.lo
df['reviews.text'].head()
```

Out[23]:
```
0    product far disappointed. children love use li...
1    great beginner experienced person. bought gift...
2    inexpensive tablet use learn on, step nabi. th...
3    i've fire hd 8 two weeks love it. tablet great...
4    bought grand daughter comes visit. set user, e...
Name: reviews.text, dtype: object
```

In [24]:
```
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
```

In [25]:
```
df['reviews.text']= df['reviews.text'].apply(lambda x: cleaning_punctuations(x))
df['reviews.text'].tail()
```

Out[25]:
```
34655    appreciably faster 18 higher amp charger used ...
34656    amazon include charger kindle fact theyre char...
34657    love kindle fire really disappointed kindle po...
34658    surprised find come type charging cords purcha...
34659    spite fact nothing good things say amazon anth...
Name: reviews.text, dtype: object
```

Using Stemming

In [26]:
```

st = nltk.PorterStemmer()

def stemming_on_reviews_text(text):
    return [st.stem(word) for word in text]


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

In [27]:
```
df['reviews.text'].head()
```

Out[27]:
```
0    [p, r, o, d, u, c, t,  , f, a, r,  , d, i, s, ...
1    [g, r, e, a, t,  , b, e, g, i, n, n, e, r,  , ...
2    [i, n, e, x, p, e, n, s, i, v, e,  , t, a, b, ...
3    [i, v, e,  , f, i, r, e,  , h, d,  , 8,  , t, ...
4    [b, o, u, g, h, t,  , g, r, a, n, d,  , d, a, ...
Name: reviews.text, dtype: object
```

Using Lemmatization

In [28]:
```
lm = nltk.WordNetLemmatizer()
def lemmatizer_on_reviews_text(text):
    return[lm.lemmatize(word) for word in df]


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

In [29]:
```
df['reviews.text'].head()
```

Out[29]:
```
0    [id, name, asin, brand, category, key, manufac...
1    [id, name, asin, brand, category, key, manufac...
2    [id, name, asin, brand, category, key, manufac...
3    [id, name, asin, brand, category, key, manufac...
4    [id, name, asin, brand, category, key, manufac...
Name: reviews.text, dtype: object
```

```
In [30]: df
```

Out[30]:

	id	name	asins	brand	categories	keys	man
0	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
1	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
2	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
3	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
4	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	
...	...	...	...	...	...	...	
34655	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34656	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34657	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34658	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se
34659	AVpfiBlyLJeJML43-4Tp	NaN	B006GWO5WK	Amazon	Computers/Tablets & Networking,Tablet & eBook ...	newamazonkindlefirehd9wpowerfastadaptercharger...	Se

34660 rows × 18 columns

```
In [31]: x= df['reviews.text']
         x.head()
```

```
Out[31]: 0    [id, name, asin, brand, category, key, manufac...
         1    [id, name, asin, brand, category, key, manufac...
         2    [id, name, asin, brand, category, key, manufac...
         3    [id, name, asin, brand, category, key, manufac...
         4    [id, name, asin, brand, category, key, manufac...
         Name: reviews.text, dtype: object
```

```
In [32]: y= df['reviews.rating']
         y.tail()
```

```
Out[32]: 34655    3.0
         34656    1.0
         34657    1.0
         34658    1.0
         34659    1.0
         Name: reviews.rating, dtype: float64
```

In [33]:

```
X = df['reviews.text'].astype(str)
```

splitting Data traning= 0.7, testing 0.3

In [34]:

```
# spliting Data for Training and Testing in two parts
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=21)
```

In [35]:

```
y_train
```

Out[35]:

```
30618    5.0
31577    4.0
9981     5.0
32239    5.0
15223    5.0
        ... 
16432    4.0
8964     5.0
5944     4.0
5327     1.0
15305    5.0
Name: reviews.rating, Length: 24262, dtype: float64
```

Uni-gram for results using models

In [36]:

```
#uni-gram
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,1))

# Training data
X_train = vectorizer.fit_transform(X_train)

# Testing data
X_test = vectorizer.transform(X_test)
```

Making prediction on the test set

In [37]:

```
# uni-gram
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
print("Random Forest Result")
rfc = RandomForestClassifier(n_estimators=100, random_state=52)
pred = rfc.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,pred))

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
dt = DecisionTreeClassifier(random_state=50)
print("Decision Tree Result")
DecisionTree=dt.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,DecisionTree))

from sklearn.svm import SVC
print("Support Vector Machine Result")
svm = SVC(kernel='linear', C=2.0, random_state=52)
svm.fit(X_train,y_train)
y_pred=svm.predict(X_test)
print(accuracy_score(y_test,y_pred))

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
print("Logistic Regression Result")
logisticRegresion=lr.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,logisticRegresion))
```

```
Random Forest Result
0.6906135795345258
Decision Tree Result
0.6906135795345258
Support Vector Machine Result
0.6906135795345258
Logistic Regression Result
0.6906135795345258
```

Compute Classification report

In [38]:
```
#uni-gram
print("Random Forest")
print(classification_report(y_test,pred))

print("Decision Tree")
print(classification_report(y_test,DecisionTree))

print("Support Vector Machine")
print(classification_report(y_test,y_pred))

print("Logistic Regression")
print(classification_report(y_test,logisticRegresion))
```

```
Random Forest
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Decision Tree
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Support Vector Machine
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Logistic Regression
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398
```

In [39]:
```
X = df['reviews.text'].astype(str)
```

In [40]:
```
# spliting for Training-Testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=21)
```

bi-gram for results using models

In [41]:

```
#bi-gram
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(2,2))

# Training Data
X_train = vectorizer.fit_transform(X_train)

# Testing Data
X_test = vectorizer.transform(X_test)
```

Making prediction on the test set

In [42]:

```
# bi-gram
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
print("Random Forest Result")
rfc = RandomForestClassifier(n_estimators=100, random_state=52)
pred = rfc.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,pred))

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
dt = DecisionTreeClassifier(random_state=50)
print("Decision Tree Result")
DecisionTree=dt.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,DecisionTree))

from sklearn.svm import SVC
print("Support Vector Machine Result")
svm = SVC(kernel='linear', C=2.0, random_state=52, gamma=0.001)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print(accuracy_score(y_test, y_pred))

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
print("Logistic Regression Result")
logisticRegresion=lr.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,logisticRegresion))
```

```
Random Forest Result
0.6906135795345258
Decision Tree Result
0.6906135795345258
Support Vector Machine Result
0.6906135795345258
Logistic Regression Result
0.6906135795345258
```

Compute Classification report

In [43]:
```
#bi-gram
print("Random Forest")
print(classification_report(y_test,pred))

print("Decision Tree")
print(classification_report(y_test,DecisionTree))

print("Support Vector Machine")
print(classification_report(y_test,y_pred))

print("Logistic Regression")
print(classification_report(y_test,logisticRegresion))
```

```
Random Forest
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Decision Tree
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Support Vector Machine
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Logistic Regression
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398
```

In [44]:
```
X = df['reviews.text'].astype(str)
```

In [45]:
```
# spliting for Training-Testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=21)
```

Tri-gram for results using models

In [46]:

```
#Tri-gram
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(3,3))

# Training Data
X_train = vectorizer.fit_transform(X_train)

# Testing Data
X_test = vectorizer.transform(X_test)
```

Making prediction on the test set

In [47]:

```
# tri-gram
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
print("Random Forest Result")
rfc = RandomForestClassifier(n_estimators=100, random_state=52)
pred = rfc.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,pred))

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
dt = DecisionTreeClassifier(random_state=50)
print("Decision Tree Result")
DecisionTree=dt.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,DecisionTree))

from sklearn.svm import SVC
print("Support Vector Machine Result")
svm = SVC(kernel='linear', C=2.0, random_state=52, gamma=0.001)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print(accuracy_score(y_test, y_pred))


from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
print("Logistic Regression Result")
logisticRegresion=lr.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,logisticRegresion))
```

```
Random Forest Result
0.6906135795345258
Decision Tree Result
0.6906135795345258
Support Vector Machine Result
0.6906135795345258
Logistic Regression Result
0.6906135795345258
```

Compute Classification report

In [48]:

```
#tri-gram
print("Random Forest")
print(classification_report(y_test,pred))

print("Decision Tree")
print(classification_report(y_test,DecisionTree))

print("Support Vector Machine")
print(classification_report(y_test,y_pred))

print("Logistic Regression")
print(classification_report(y_test,logisticRegresion))
```

```
Random Forest
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Decision Tree
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Support Vector Machine
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Logistic Regression
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398
```

In [49]:

```
X = df['reviews.text'].astype(str)
```

In [50]:

```
# spliting for Training-Testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=21)
```

n-gram for results using models

In [51]:

```
#n-gram
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,3))

# Training Data
X_train = vectorizer.fit_transform(X_train)

# Testing Data
X_test = vectorizer.transform(X_test)
```

Making prediction on the test set

In [52]:

```
# n-gram
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
print("Random Forest Result")
rfc = RandomForestClassifier(n_estimators=100, random_state=52)
pred = rfc.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,pred))

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
dt = DecisionTreeClassifier(random_state=50)
print("Decision Tree Result")
DecisionTree=dt.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,DecisionTree))

from sklearn.svm import SVC
print("Support Vector Machine Result")
svm = SVC(kernel='linear', C=2.0, random_state=52)
svm.fit(X_train,y_train)
y_pred=svm.predict(X_test)
print(accuracy_score(y_test,y_pred))

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
print("Logistic Regression Result")
logisticRegresion=lr.fit(X_train, y_train).predict(X_test)
print(accuracy_score(y_test,logisticRegresion))
```

```
Random Forest Result
0.6906135795345258
Decision Tree Result
0.6906135795345258
Support Vector Machine Result
0.6906135795345258
Logistic Regression Result
0.6906135795345258
```

Compute Classification report

In [53]:

```
#n-gram
print("Decision Tree")
print(classification_report(y_test,DecisionTree))

print("Random Forest")
print(classification_report(y_test,pred))

print("Logistic Regression")
print(classification_report(y_test,logisticRegresion))

print("Support Vector Machine")
print(classification_report(y_test,y_pred))
```

```
Decision Tree
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Random Forest
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Logistic Regression
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398

Support Vector Machine
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        13
         1.0       0.00      0.00      0.00       122
         2.0       0.00      0.00      0.00       115
         3.0       0.00      0.00      0.00       442
         4.0       0.00      0.00      0.00      2525
         5.0       0.69      1.00      0.82      7181

    accuracy                           0.69     10398
   macro avg       0.12      0.17      0.14     10398
weighted avg       0.48      0.69      0.56     10398
```

In [54]:

```
# Assuming df is your cleaned DataFrame
df.to_csv('amazon_data.csv', index=False)
```

In [ ]: