In [1]:

```
#importing the necesssary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#reading the dataset

df = pd.read_csv("AI_Capstone_Ecommerce_train_data.csv")
df
```

Out[2]:

	name	brand	categories	primaryCategories	reviews.date	revi
0	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-26T00:00:00.000Z	Pu Frid Gr
1	Amazon - Echo Plus w/ Built-In Hub - Silver	Amazon	Amazon Echo,Smart Home,Networking,Home & Tools...	Electronics,Hardware	2018-01-17T00:00:00.000Z	I pu two in E and
2	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Amazon Echo,Virtual Assistant Speakers,Electro...	Electronics,Hardware	2017-12-20T00:00:00.000Z	optic sh
3	Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 ...	Amazon	eBook Readers,Fire Tablets,Electronics Feature...	Office Supplies,Electronics	2017-08-04T00:00:00.000Z	v Exa l
4	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2017-01-23T00:00:00.000Z	Tl 3rd pu I've
...	...	...	...	...	...	...
3995	Amazon - Echo Plus w/ Built-In Hub - Silver	Amazon	Amazon Echo,Smart Home,Networking,Home & Tools...	Electronics,Hardware	2017-12-08T00:00:00.000Z	It,Äô the p
3996	Amazon Kindle E-Reader 6" Wifi (8th Generation...	Amazon	Computers,Electronics Features,Tablets,Electro...	Electronics	2017-03-31T00:00:00.000Z	I Kind pı
3997	Amazon Tap - Alexa-Enabled Portable Bluetooth ...	Amazon	Amazon Echo,Home Theater & Audio,MP3 MP4 Playe...	Electronics	2017-01-19T00:00:00.000Z	look sp
3998	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2016-05-27T00:00:00.000Z	Tl Ama
3999	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-30T00:00:00.000Z	satis tak

4000 rows × 8 columns

Checking for imbalance in data

In [4]:

```
#Importance:-
  #it's mentioned in the data that our target class is imballanced, so let's check that.
  #even if it was not mention than we must always make sure that our output variable is'nt
  #let's say there are 97 positive and one 3 negative classes , than first of all it'll tra
  #This reduces the models capability to recognize the negative class.
  #Secondly,it will give a very flase conclusion that the model is 99% accurate even if it
  #Suppose we handed our model to the client by trusting the accuracy and the new data cont
  #It'll predict them positive as well.
  #This might lead to wrong bussiness decisions or govt might take false decisions by trust
  #This will lead to huge losses in whatever sector the model is implies.
  #So , we must make sure that our training data is not highly skewed


df['sentiment'].value_counts()
```

Out[4]:

```
Positive    3749
Neutral      158
Negative      93
Name: sentiment, dtype: int64
```

In [5]:

```
df.describe()
```

Out[5]:

	name	brand	categories	primaryCategories	reviews.date	reviews.text	re
count	4000	4000	4000	4000	4000	4000	
unique	23	1	23	4	638	3598	
top	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2017-01-23T00:00:00.000Z	I bought this kindle for my 11yr old granddaug...	(
freq	676	4000	628	2600	99	4	

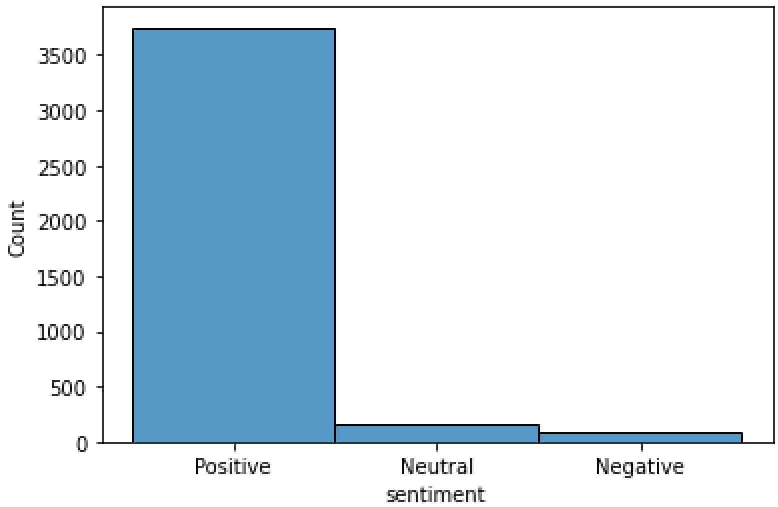
In [32]:

```
#let's visuallise our target class data

sns.histplot(df['sentiment'])
```

Out[32]:

```
<AxesSubplot:xlabel='sentiment', ylabel='Count'>
```



Treating the imbalanced data by Oversampling the minority class

In [5]:

```
#as we can see that the data is highly imbalanced , so we can use either Oversampling or Un
#in this case we will oversample our minority classes by using RandomOverSampler() function

import imblearn
from imblearn.over_sampling import RandomOverSampler
```

In [18]:

```
os = RandomOverSampler(sampling_strategy={'Negative':500 , 'Neutral':750})
a=os.fit_resample(df.iloc[:,0:8],df['sentiment'])
```

In [20]:

```
#let's have a look at our dataset.
#RandomOverSampler returns 2 outputs , 1st is the df on which we performed oversampling, 2n
#we need to see our dataset , so we will use a[0]

df1=a[0]
df1
```

Out[20]:

	name	brand	categories	primaryCategories	reviews.date	revi
0	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-26T00:00:00.000Z	P Fric Gr
1	Amazon - Echo Plus w/ Built-In Hub - Silver	Amazon	Amazon Echo,Smart Home,Networking,Home & Tools...	Electronics,Hardware	2018-01-17T00:00:00.000Z	I p two in E and
2	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Amazon Echo,Virtual Assistant Speakers,Electro...	Electronics,Hardware	2017-12-20T00:00:00.000Z	Alex Doe
3	Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 ...	Amazon	eBook Readers,Fire Tablets,Electronics Feature...	Office Supplies,Electronics	2017-08-04T00:00:00.000Z	V Exa
4	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2017-01-23T00:00:00.000Z	T 3rc pu I've
...	...	...	...	...	...	
4994	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2017-02-04T00:00:00.000Z	Nc s con F
4995	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Amazon Echo,Virtual Assistant Speakers,Electro...	Electronics,Hardware	2018-01-28T00:00:00.000Z	T h nice but

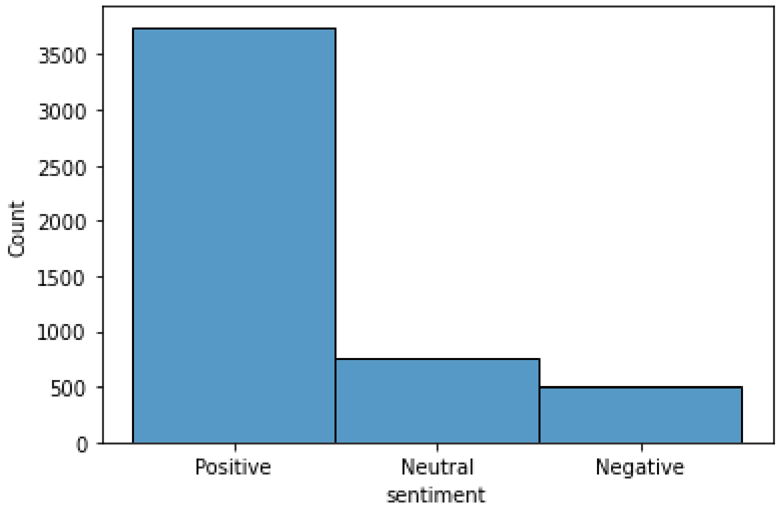
	name	brand	categories	primaryCategories	reviews.date	revi
4996	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	Amazon	Fire Tablets,Computers/Tablets & Networking,Ta...	Electronics	2017-01-12T00:00:00.000Z	for 3 it alth
4997	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Computers,Amazon Echo,Virtual Assistant Speake...	Electronics,Hardware	2017-09-29T00:00:00.000Z	disa Sc give
4998	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Amazon Echo,Virtual Assistant Speakers,Electro...	Electronics,Hardware	2018-02-16T00:00:00.000Z	A us m re

In [34]:

```
sns.histplot(df1['sentiment'])
```

Out[34]:

```
<AxesSubplot:xlabel='sentiment', ylabel='Count'>
```



In [21]:

```
#now let's look at our output classes value_counts()
df1['sentiment'].value_counts()
```

Out[21]:

```
Positive    3749
Neutral      750
Negative     500
Name: sentiment, dtype: int64
```

Converting reviews into tfidf vectors

In [42]:

```
#as mentioned in the guidelines we will convert the reviews into tfidf score by using Tfidf

from sklearn.feature_extraction.text import TfidfVectorizer , CountVectorizer
```

In [43]:

```
tf = TfidfVectorizer()
vectors = tf.fit_transform(df1['reviews.text'])
tokens=tf.get_feature_names()
```

```
C:\Users\Ankita Sharma\anaconda3\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function get_feature_names is deprecated; get_featur
e_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feat
ure_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [44]:

```
vectors=pd.DataFrame(vectors.toarray() , columns=tokens )
vectors
```

Out[44]:

	00	10	100	1000	1000s	1080	10th	10x	11	11yr	...	äù	äú	äúalexa	äúbest	äú
0	0.0	0.0	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	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4994	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	

4999 rows × 4897 columns

In [45]:

```
#creating the final dataset for our model

finaldf = pd.concat((vectors,df1['sentiment']) , axis=1)
finaldf
```

Out[45]:

	00	10	100	1000	1000s	1080	10th	10x	11	11yr	...	äú	äúalexa	äúbest	äúdrop
0	0.0	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	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4994	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

4999 rows × 4898 columns

Preparing the Random Forest Classifier Model for training

In [52]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

In [53]:

```
param_grid={'max_depth':[30,40,50] , 'min_samples_leaf':[1,2]}
gscv=GridSearchCV(RandomForestClassifier(), param_grid=param_grid , cv=20 , verbose=3 )
gscv.fit(vectors,finaldf['sentiment'])
```

```
Fitting 20 folds for each of 6 candidates, totalling 120 fits
[CV 1/20] END .max_depth=30, min_samples_leaf=1;, score=0.908 total time=
8.3s
[CV 2/20] END .max_depth=30, min_samples_leaf=1;, score=0.900 total time=
8.2s
[CV 3/20] END .max_depth=30, min_samples_leaf=1;, score=0.916 total time=
7.7s
[CV 4/20] END .max_depth=30, min_samples_leaf=1;, score=0.932 total time=
7.2s
[CV 5/20] END .max_depth=30, min_samples_leaf=1;, score=0.908 total time=
7.4s
[CV 6/20] END .max_depth=30, min_samples_leaf=1;, score=0.920 total time=
7.3s
[CV 7/20] END .max_depth=30, min_samples_leaf=1;, score=0.968 total time=
7.3s
[CV 8/20] END .max_depth=30, min_samples_leaf=1;, score=0.916 total time=
7.3s
[CV 9/20] END .max_depth=30, min_samples_leaf=1;, score=0.928 total time=
7.3s
```

In [46]:

```
#saving the model weights using pickle library
import pickle
```

In [55]:

```
pickle.dump(gscv , open('AI Capstone Project - Ecommerce' , 'wb') ,protocol=4)
```

In [56]:

```
m=pickle.load(open('AI Capstone Project - Ecommerce','rb'))
```

In [57]:

```
#checking the prediction of our model.

m.predict(np.array(vectors.iloc[2,:]).reshape(1,-1))
```

```
C:\Users\Ankita Sharma\anaconda3\lib\site-packages\sklearn\base.py:450: User
Warning: X does not have valid feature names, but RandomForestClassifier was
fitted with feature names
  warnings.warn(
```

Out[57]:

```
array(['Neutral'], dtype=object)
```

In [48]:

```
#as, the dataset was imbalanced so , the normal ['accuracy'] metrics can be misleading so ,

from sklearn.metrics import f1_score
```

In [65]:

```
#reading the test data for checking the f1_score
test = pd.read_csv("AI Capstone Project - Ecommerce TestData.csv")
```

In [66]:

```
test
```

Out[66]:

	name	brand	categories	primaryCategories	reviews.date	reviews.
0	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	Amazon	Fire Tablets,Computers/Tablets & Networking,Ta...	Electronics	2016-05-23T00:00:00.000Z	Ama kindle has a l free app
1	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Computers,Amazon Echo,Virtual Assistant Speake...	Electronics,Hardware	2018-01-02T00:00:00.000Z	The E Show g additic the Ama
2	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2017-01-02T00:00:00.000Z	Great v from I Buy. Bo Christm
3	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2017-03-25T00:00:00.000Z	I use r for er Facet ,games tc
4	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Computers,Amazon Echo,Virtual Assistant Speake...	Electronics,Hardware	2017-11-15T00:00:00.000Z	This fant item & pers boug
...	...	...	...	...	...	
995	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Computers,Amazon Echo,Virtual Assistant Speake...	Electronics,Hardware	2017-12-07T18:06:07.000Z	We Alexa! L being ab watch n
996	Amazon Tap - Alexa-Enabled Portable Bluetooth ...	Amazon	Amazon Echo,Home Theater & Audio,MP3 MP4 Playe...	Electronics	2017-01-23T00:00:00.000Z	Speak pretty and I that I ca
997	Fire HD 8 Tablet with Alexa, 8" HD Display, 32...	Amazon	Tablets,Fire Tablets,Computers & Tablets,All T...	Electronics	2017-01-18T00:00:00.000Z	Bought these fo 6 and old anc

	name	brand	categories	primaryCategories	reviews.date	reviews.
998	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-12T00:00:00.000Z	Was tol sales pe I could c back
999	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	Amazon	Fire Tablets,Computers/Tablets & Networking,Ta...	Electronics	2017-06-17T00:00:00.000Z	I purcha this as a fo mother.

1000 rows × 8 columns

In [67]:

```
#again we have to repeat the same process and convert the test reviews into tfidf vectors ,

tf1 = TfidfVectorizer()
test_vectors = tf1.fit_transform(test['reviews.text'])
```

In [68]:

```
test_tokens = tf1.get_feature_names()
test_tokens
```

```
C:\Users\Ankita Sharma\anaconda3\lib\site-packages\sklearn\utils\deprecati
on.py:87: FutureWarning: Function get_feature_names is deprecated; get_fea
ture_names is deprecated in 1.0 and will be removed in 1.2. Please use get
_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [69]:

```
test_vectors1 = pd.DataFrame(test_vectors.toarray() , columns=test_tokens)
test_vectors1
```

Out[69]:

	00	10	100	105	11	12	128	128gb	139	15	...	äôre	äôs	äôt	äôve	äù	äùcrest
0	0.0	0.0	0.0	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	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

1000 rows × 2827 columns

Important Note - As our model was trained on a matrix of tfidfvectors of size=(4999,4897) and our test tfidf matrix is of size (1000,2827) .So , model was not predict the test size, as it was expecting the size(4999,4897).So, either we have to pad our test vectors or our model on the new tfidf vector that is built over total data(train+test).

In [70]:

```
#as we know the optimum hyperparameters from our previous training so it will be quite easy

total_data = pd.concat((df1 , test) , axis=0)
total_data
```

Out[70]:

	name	brand	categories	primaryCategories	reviews.date	revie
0	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-26T00:00:00.000Z	Pur o Frida Gre
1	Amazon - Echo Plus w/ Built-In Hub - Silver	Amazon	Amazon Echo,Smart Home,Networking,Home & Tools...	Electronics,Hardware	2018-01-17T00:00:00.000Z	I pur two A in Ec and tv
2	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Amazon Echo,Virtual Assistant Speakers,Electro...	Electronics,Hardware	2017-12-20T00:00:00.000Z	a optio sho
3	Fire HD 10 Tablet, 10.1 HD Display, Wi-Fi, 16 ...	Amazon	eBook Readers,Fire Tablets,Electronics Feature...	Office Supplies,Electronics	2017-08-04T00:00:00.000Z	ve p Exac I v
4	Brand New Amazon Kindle Fire 16gb 7" Ips Displ...	Amazon	Computers/Tablets & Networking,Tablets & eBook...	Electronics	2017-01-23T00:00:00.000Z	Thi 3rd c purc I've b
...	...	...	...	...	...	
995	Amazon Echo Show Alexa-enabled Bluetooth Speak...	Amazon	Computers,Amazon Echo,Virtual Assistant Speake...	Electronics,Hardware	2017-12-07T18:06:07.000Z	V Alex being watc
996	Amazon Tap - Alexa-Enabled Portable Bluetooth ...	Amazon	Amazon Echo,Home Theater & Audio,MP3 MP4 Playe...	Electronics	2017-01-23T00:00:00.000Z	Spe pre an that I

	name	brand	categories	primaryCategories	reviews.date	revie
997	Fire HD 8 Tablet with Alexa, 8" HD Display, 32...	Amazon	Tablets,Fire Tablets,Computers & Tablets,All T...	Electronics	2017-01-18T00:00:00.000Z	Bou these 6 a old
998	All-New Fire HD 8 Tablet, 8" HD Display, Wi-Fi...	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	Electronics	2016-12-12T00:00:00.000Z	Was sales I coul ba
999	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	Amazon	Fire Tablets,Computers/Tablets & Networking,Ta...	Electronics	2017-06-17T00:00:00.000Z	I pur this a moth

5999 rows × 8 columns

In [71]:

```
#Making a new tfidf vector matrix that contains all words(train + test)

total_vectors = tf.fit_transform(total_data['reviews.text'])
total_tokens = tf.get_feature_names()
```

```
C:\Users\Ankita Sharma\anaconda3\lib\site-packages\sklearn\utils\deprecatio
n.py:87: FutureWarning: Function get_feature_names is deprecated; get_featur
e_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feat
ure_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [72]:

```
total_vectors = pd.DataFrame(total_vectors.toarray() , columns=total_tokens)
total_vectors
```

Out[72]:

	00	10	100	1000	1000s	105	1080	10th	10x	11	...	äú	äúalexa	äúbest	äúdr
0	0.0	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	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5994	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
5995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
5996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
5997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
5998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

5999 rows × 5412 columns

In [73]:

```
#Now seperating our vectors properly for checking our prediction score.

train = total_data.iloc[0:4999,]
test = total_data.iloc[4999:-1]

train_vectors = total_vectors.iloc[0:4999,]
test_vectors = total_vectors.iloc[4999:-1]
```

In [74]:

```
#fitting the training set with optimum hyperparameter , max_depth = 45

rfc = RandomForestClassifier(max_depth=45)
rfc.fit(train_vectors,train['sentiment'])
```

Out[74]:

```
RandomForestClassifier(max_depth=45)
```

In [75]:

```
#now let's make prediction on our test vectors
testpred=rfc.predict(test_vectors)
```

In [76]:

```
#yess , now our model is able to predict
testpred
```

Out[76]:

```
array(['Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negative', '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', 'Negative', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
```

In [77]:

```
#import the f1_score and precision functions from sklearn.metrics library to check our scor

from sklearn.metrics import f1_score , precision_score
```

F1 Score of Random Classifier Model

In [78]:

```
f1_score(test['sentiment'] , testpred , average='micro')
```

Out[78]:

```
0.953953953953954
```

Precision Score of Random Classifier Model

In [79]:

```
precision_score(test['sentiment'] , testpred , average='micro')
```

Out[79]:

```
0.953953953953954
```

SVM Model

In [54]:

```
#now training our dataset on Support Vector Machine

from sklearn.svm import SVC
```

In [41]:

```
svc = SVC()
svc.fit(train_vectors , train['sentiment'])
```

Out[41]:

```
SVC()
```

In [42]:

```
testpred=svc.predict(test_vectors)
testpred
```

Out[42]:

```
array(['Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negative', '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', 'Negative', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
```

F1 Score of SVM model

In [43]:

```
f1_score(test['sentiment'] , testpred , average='micro')
```

Out[43]:

```
0.953953953953954
```

LSTM Model - we will use glove vector embeddings for training our LSTM model.

In [59]:

```
#importing the necessary libraries

import tensorflow
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM , Dense , Dropout , Embedding , SpatialDropout1D , Flatten
from keras.preprocessing.text import Tokenizer
from keras.utils.data_utils import pad_sequences
```

In [60]:

```
#reading the glove vector embeddings

file = open("(RNN,Vectors4words)glove.6B.100d.txt" , 'r' , encoding='utf8')
embed=file.readlines()
embed
```

Out[60]:

```
['the -0.038194 -0.24487 0.72812 -0.39961 0.083172 0.043953 -0.39141 0.334
4 -0.57545 0.087459 0.28787 -0.06731 0.30906 -0.26384 -0.13231 -0.20757 0.
33395 -0.33848 -0.31743 -0.48336 0.1464 -0.37304 0.34577 0.052041 0.44946
-0.46971 0.02628 -0.54155 -0.15518 -0.14107 -0.039722 0.28277 0.14393 0.23
464 -0.31021 0.086173 0.20397 0.52624 0.17164 -0.082378 -0.71787 -0.41531
0.20335 -0.12763 0.41367 0.55187 0.57908 -0.33477 -0.36559 -0.54857 -0.062
892 0.26584 0.30205 0.99775 -0.80481 -3.0243 0.01254 -0.36942 2.2167 0.722
01 -0.24978 0.92136 0.034514 0.46745 1.1079 -0.19358 -0.074575 0.23353 -0.
052062 -0.22044 0.057162 -0.15806 -0.30798 -0.41625 0.37972 0.15006 -0.532
12 -0.2055 -1.2526 0.071624 0.70565 0.49744 -0.42063 0.26148 -1.538 -0.302
23 -0.073438 -0.28312 0.37104 -0.25217 0.016215 -0.017099 -0.38984 0.87424
-0.72569 -0.51058 -0.52028 -0.1459 0.8278 0.27062\n',
 ', -0.10767 0.11053 0.59812 -0.54361 0.67396 0.10663 0.038867 0.35481 0.0
6351 -0.094189 0.15786 -0.81665 0.14172 0.21939 0.58505 -0.52158 0.22783 -
0.16642 -0.68228 0.3587 0.42568 0.19021 0.91963 0.57555 0.46185 0.42363 -
0.095399 -0.42749 -0.16567 -0.056842 -0.29595 0.26037 -0.26606 -0.070404 -
0.27662 0.15821 0.69825 0.43081 0.27952 -0.45437 -0.33801 -0.58184 0.22364
-0.5778 -0.26862 -0.20425 0.56394 -0.58524 -0.14365 -0.64218 0.0054697 -0.
```

In [61]:

```
#making a dictionary for our vector embeddings

embed_dict={}
for i in embed:
    a=i.split()
    embed_dict[a[0]] = np.array(a[1:-1] , dtype='float32')
```

In [62]:

```
#let's have a look at our embedding dictionary
embed_dict
```

Out[62]:

```
{'the': array([-0.038194, -0.24487 ,  0.72812 , -0.39961 ,  0.083172,  0.0
43953,
        -0.39141 ,  0.3344  , -0.57545 ,  0.087459,  0.28787 , -0.06731 ,
         0.30906 , -0.26384 , -0.13231 , -0.20757 ,  0.33395 , -0.33848 ,
        -0.31743 , -0.48336 ,  0.1464  , -0.37304 ,  0.34577 ,  0.052041,
         0.44946 , -0.46971 ,  0.02628 , -0.54155 , -0.15518 , -0.14107 ,
        -0.039722,  0.28277 ,  0.14393 ,  0.23464 , -0.31021 ,  0.086173,
         0.20397 ,  0.52624 ,  0.17164 , -0.082378, -0.71787 , -0.41531 ,
         0.20335 , -0.12763 ,  0.41367 ,  0.55187 ,  0.57908 , -0.33477 ,
        -0.36559 , -0.54857 , -0.062892,  0.26584 ,  0.30205 ,  0.99775 ,
        -0.80481 , -3.0243  ,  0.01254 , -0.36942 ,  2.2167  ,  0.72201 ,
        -0.24978 ,  0.92136 ,  0.034514,  0.46745 ,  1.1079  , -0.19358 ,
        -0.074575,  0.23353 , -0.052062, -0.22044 ,  0.057162, -0.15806 ,
        -0.30798 , -0.41625 ,  0.37972 ,  0.15006 , -0.53212 , -0.2055  ,
        -1.2526  ,  0.071624,  0.70565 ,  0.49744 , -0.42063 ,  0.26148 ,
        -1.538   , -0.30223 , -0.073438, -0.28312 ,  0.37104 , -0.25217 ,
         0.016215, -0.017099, -0.38984 ,  0.87424 , -0.72569 , -0.51058 ,
        -0.52028 , -0.1459  ,  0.8278  ], dtype=float32),
```

In [63]:

```
#let's check the no. of features of our word embedding as we will further require it to fil

vec_dim=len(embed_dict['the'])
vec_dim
```

Out[63]:

```
99
```

In [83]:

```
#creating the wordindex dictionary for all the words in our reviews

tokens = Tokenizer()
tokens.fit_on_texts(train['reviews.text'])
wordindex = tokens.word_index

wordindex
```

Out[83]:

```
{'the': 1,
 'to': 2,
 'it': 3,
 'and': 4,
 'i': 5,
 'for': 6,
 'a': 7,
 'is': 8,
 'my': 9,
 'this': 10,
 'of': 11,
 'with': 12,
 'great': 13,
 'tablet': 14,
 'on': 15,
 'was': 16,
 'not': 17,
 'but': 18,
```

In [84]:

```
#creating a sequence of wordindexes for each review in our reviews.text column
seq=tokens.texts_to_sequences(train['reviews.text'])
seq
```

Out[84]:

```
[[81,
  15,
  341,
  3266,
  13,
  56,
  161,
  168,
  383,
  28,
  983,
  4,
  236,
  12,
  2262,
  2263,
  3267,
  262,
```

In [86]:

```
len(wordindex)
```

Out[86]:

```
5060
```

In [87]:

```
#we will create an embedding matrix which will contain the vector embeddings of only those

embed_matrix=np.zeros((len(wordindex)+1,99) )
for k,i in wordindex.items():
    emb_vec = embed_dict.get(k)
    if emb_vec is not None:
        embed_matrix[i,:] = emb_vec
```

In [88]:

```
embed_matrix
```

Out[88]:

```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [-0.038194  , -0.24487001,  0.72812003, ..., -0.52028   ,
        -0.1459    ,  0.82779998],
       [-0.18970001,  0.050024  ,  0.19084001, ..., -0.038175  ,
        -0.39804   ,  0.47646999],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.17184   ,  0.14425001,  0.67543   , ...,  0.36636001,
         0.24167   , -0.19874001],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

In [89]:

```
embed_matrix.shape
```

Out[89]:

```
(5061, 99)
```

In [90]:

```
#let's check the max length of the reviews , so that we can fix fix the size of every revie

lengths=[]
for i in seq:
    lengths.append(len(i))
maxlen=max(lengths)
maxlen
```

Out[90]:

```
1559
```

In [91]:

```
#as lstm layer takes sentences of fix length so we need to make all the reviews of same len
#we will do this by padding all the sentences to make them all equal to the max lengths=155

import pad_sequences
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [93]:

```
padded_seq=pad_sequences(seq , maxlen=maxlen , padding='post')
padded_seq
```

Out[93]:

```
array([[  81,   15,  341, ...,    0,    0,    0],
       [   5,   81,  192, ...,    0,    0,    0],
       [  49,   41, 1102, ...,    0,    0,    0],
       ...,
       [   6, 1520,  964, ...,    0,    0,    0],
       [ 131,  361,  407, ...,    0,    0,    0],
       [  43,    5,   87, ...,    0,    0,    0]])
```

In [94]:

```
padded_seq.shape
```

Out[94]:

```
(4999, 1559)
```

In [95]:

```
#importing one hot encoder to prepare our labels

from sklearn.preprocessing import OneHotEncoder
```

In [96]:

```
#oe = OneHotEncoder(categories=['negative' , 'neutral' , 'positive'])
oe=OneHotEncoder()
```

In [97]:

```
labels = oe.fit_transform(np.array(train['sentiment']).reshape(-1,1))
categories=oe.categories_
labels = pd.DataFrame(labels.toarray() , columns=categories)
labels
```

Out[97]:

	Negative	Neutral	Positive
0	0.0	0.0	1.0
1	0.0	0.0	1.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
4	0.0	0.0	1.0
...	...	...	...
4994	0.0	1.0	0.0
4995	0.0	1.0	0.0
4996	0.0	1.0	0.0
4997	0.0	1.0	0.0
4998	0.0	1.0	0.0

4999 rows × 3 columns

In [ ]:

```

```

In [291]:

```
model = Sequential()

model.add(Embedding(input_dim=len(wordindex)+1 , output_dim=99 , input_length=maxlen , weig

model.add(LSTM(units=50 , return_sequences=True))
model.add(SpatialDropout1D(0.2))

model.add(LSTM(units=50 , return_sequences=True))
model.add(SpatialDropout1D(0.2))

model.add(LSTM(units=50 , return_sequences=True))
model.add(SpatialDropout1D(0.2))

model.add(Flatten())

model.add(Dense(units=50 , activation='relu'))
model.add(Dense(units=3 , activation='softmax'))
```

In [292]:

```
model.compile(optimizer=tensorflow.keras.optimizers.Adam() , loss=tensorflow.keras.losses.C
```

In [293]:

```
model.fit(padded_seq , labels , epochs=30)
```

```
Epoch 1/30
157/157 [==============================] - 286s 2s/step - loss: 0.6587 - c
ategorical_accuracy: 0.7634
Epoch 2/30
157/157 [==============================] - 279s 2s/step - loss: 0.3755 - c
ategorical_accuracy: 0.8522
Epoch 3/30
157/157 [==============================] - 304s 2s/step - loss: 0.1733 - c
ategorical_accuracy: 0.9388
Epoch 4/30
157/157 [==============================] - 287s 2s/step - loss: 0.0957 - c
ategorical_accuracy: 0.9648
Epoch 5/30
157/157 [==============================] - 300s 2s/step - loss: 0.0614 - c
ategorical_accuracy: 0.9802
Epoch 6/30
157/157 [==============================] - 291s 2s/step - loss: 0.0417 - c
ategorical_accuracy: 0.9842
Epoch 7/30
```