

Feedback analysis models Project

This project aims to train and prepare a "feedback analysis model" to be used at the time that we would have the "feedback" feature for any of our products. (spetially the "Market Place Project")

As currently we don't have a "Feedback" feature in our products, so I will be using an "Amazon food reviews and rating" dataset in csv format (which are text reviews and rating the food out of five stars)

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import nltk
nltk.download()

showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

```
In [ ]: import nltk
nltk.download('maxent_ne_chunker')
```

```
In [ ]: import nltk
nltk.download('words')
```

```
In [28]: import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]      C:\Users\ella\AppData\Roaming\nltk_data...
```

```
Out[28]: True
```

```
In [17]: # Read in data
feedback = pd.read_csv('Reviews.csv')
feedback.head()
```

Out[17]:

	Id	ProductId		UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	1	B001E4KFG0	A3SGXH7AUHU8GW		delmartian		1
1	2	B00813GRG4	A1D87F6ZCVE5NK		dll pa		0
2	3	B000LQOCH0	ABXLMWJIXXAIN		Natalia Corres "Natalia Corres"		1
3	4	B000UA0QIQ	A395BORC6FGVXV		Karl		3
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T		Michael D. Bigham "M. Wassir"		0

◀ ▶

In []: `feedback.shape`

In [18]: `feedback = feedback.head(500)`
`feedback.head(5)`

Out[18]:

	Id	ProductId		UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	1	B001E4KFG0	A3SGXH7AUHU8GW		delmartian		1
1	2	B00813GRG4	A1D87F6ZCVE5NK		dll pa		0
2	3	B000LQOCH0	ABXLMWJIXXAIN		Natalia Corres "Natalia Corres"		1
3	4	B000UA0QIQ	A395BORC6FGVXV		Karl		3
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T		Michael D. Bigham "M. Wassir"		0

◀ ▶

Quick EDA to get an idea of what the data set looks like

In [19]:

```
# Value count on Score column to see the number of times each score occurs
```

```
feedback['Score'].value_counts().sort_index()
```

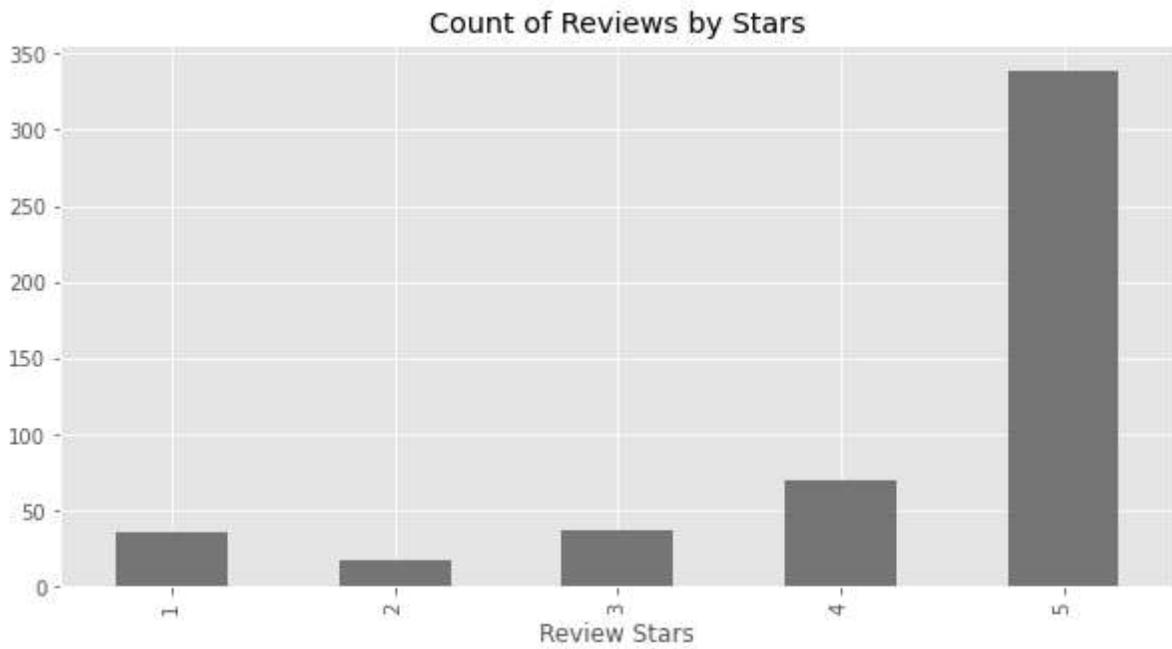
Out[19]:

1	36
2	18
3	37
4	70
5	339

```
Name: Score, dtype: int64
```

In [20]:

```
ax = feedback['Score'].value_counts().sort_index() \
    .plot(kind='bar',
          title='Count of Reviews by Stars',
          figsize=(10, 5))
ax.set_xlabel('Review Stars')
plt.show()
```



the plot we can see from that most of the reviews are 5 stars

Basic NLTK

```
In [21]: ex = feedback['Text'][50]
print(ex)
```

This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the way to go.

the result came back negative

Running NLTK word tokenizer to splits the sentence into the parts of each word in the sentence

```
In [23]: tokens = nltk.word_tokenize(ex)
tokens
```

```
Out[23]: ['This',
 'oatmeal',
 'is',
 'not',
 'good',
 '.',
 'Its',
 'mushy',
 ',',
 'soft',
 ',',
 'I',
 'do',
 "n't",
 'like',
 'it',
 '.',
 'Quaker',
 'Oats',
 'is',
 'the',
 'way',
 'to',
 'go',
 '..']
```

```
In [24]: # showing the first 10 words
tokens[:10]
```

```
Out[24]: ['This', 'oatmeal', 'is', 'not', 'good', '.', 'Its', 'mushy', ',', 'soft']
```

```
In [25]: # Running the nltk pos tag for part of speech tagging
speech_tagged = nltk.pos_tag(tokens)
speech_tagged[:10]
```

```
Out[25]: [('This', 'DT'),
 ('oatmeal', 'NN'),
 ('is', 'VBZ'),
 ('not', 'RB'),
 ('good', 'JJ'),
 ('.', '.'),
 ('Its', 'PRP$'),
 ('mushy', 'NN'),
 (',', ','),
 ('soft', 'JJ')]
```

```
In [26]: # Grouping the Tokens into chunks of text
entities = nltk.chunk.ne_chunk(speech_tagged)
entities pprint()
```

```
(S
  This/DT
  oatmeal/NN
  is/VBZ
  not/RB
  good/JJ
  ./.
  Its/PRP$
  mushy/NN
  ,/,
  soft/JJ
  ,/,
  I/PRP
  do/VBP
  n't/RB
  like/VB
  it/PRP
  ./.
  (ORGANIZATION Quaker/NNP Oats/NNPS)
  is/VBZ
  the/DT
  way/NN
  to/T0
  go/VB
  ./.)
```

Vader Sentiment Scoring

VADER(Valence Aware Dictionary for Sentiment Reasoning) is an NLTK module that provides sentiment scores based on the words used. It is a rule-based sentiment analyzer in which the terms are generally labeled as per their semantic orientation as either positive or negative.

VADER has the advantage of assessing the sentiment of any given text without the need for previous training as we might have to for Machine Learning models. The result generated by VADER is a dictionary of 4 keys neg, neu, pos and compound: neg, neu, and pos meaning negative, neutral, and positive respectively.

This module uses a “bag of words” approach:

- Stop words are removed
- Each word scored and combined to a total score
- This model is not account for relationship between words

```
In [3]: from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

#sia is an created object of "Sentiment Intensity Analyzer"
sia = SentimentIntensityAnalyzer()
```

Testing our object

```
sia.polarity_scores('I am so happy!')
```

The result is: negative as zero, neutral point = 0.318 and positive point= 0.682, So this sentence is mostly positive

```
In [83]: # Test 2  
sia.polarity_scores('This is the worst thing ever.')
```

```
Out[83]: {'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}
```

```
In [4]: sia.polarity_scores('This is the best thing ever.')
```

```
Out[4]: {'neg': 0.0, 'neu': 0.543, 'pos': 0.457, 'compound': 0.6369}
```

The result is: negative point = 0.451, neutral point = 0.549 and positive point= 0 and compound: -0.6249 So this sentence is mostly Negative

```
In [34]: # Running the sia on our tokens  
sia.polarity_scores(ex)
```

```
Out[34]: {'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

The result is: negative point = 0.22, neutral point = 0.78 and positive point= 0 and compound: -0.5448 So this sentence is mostly Negative

```
In [44]: # Run the polarity score on the entire dataset  
# result is a dictionary which stores the result of the below loop which each raw we  
  
result = {}  
for i, row in tqdm(feedback.iterrows(), total=len(feedback)):  
    text = row['Text']  
    myid = row['Id']  
    result[myid] = sia.polarity_scores(text)  
  
0%|          | 0/500 [00:00<?, ?it/s]
```

Here shows the result dictionary with each id that contain the scores of neg, neu, pos and compound which stored in a Panda data frame to make it easier to work with and displayed horizontally

```
In [46]: # here shows the result dictionary with each id that contain the scores of neg, neu  
pd.DataFrame(result).T
```

Out[46]:

	neg	neu	pos	compound
1	0.000	0.695	0.305	0.9441
2	0.138	0.862	0.000	-0.5664
3	0.091	0.754	0.155	0.8265
4	0.000	1.000	0.000	0.0000
5	0.000	0.552	0.448	0.9468
...
496	0.000	0.554	0.446	0.9725
497	0.059	0.799	0.142	0.7833
498	0.025	0.762	0.212	0.9848
499	0.041	0.904	0.055	0.1280
500	0.000	0.678	0.322	0.9811

500 rows × 4 columns

In [61]:

feedback

Out[61]:

	Id	ProductId		UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDeno
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian		1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa		0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"		1	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl		3	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"		0	
...
495	496	B000G6RYNE	APGAA43E3WPN7	Darren		0	
496	497	B000G6RYNE	ABR7HU5H1KNE	Keith		0	
497	498	B000G6RYNE	AJQD2WWJYOYFQ	bubbles		0	
498	499	B000G6RYNE	A16YH487W9ZY00	Bruce G. Lindsay		0	

Id	ProductId		UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
499	500	B000G6RYNE	A83YQC1XOU4CS	J. Baker	0	

So now we have a data frame that has index which is the IDs and the four neg, neu, pos and compound fields, We call this result Vaders

```
In [67]: vaders = pd.DataFrame(result).T
vaders.columns
```

```
Out[67]: Index(['neg', 'neu', 'pos', 'compound'], dtype='object')
```

```
In [68]: feedback.columns
```

```
Out[68]: Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
   'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
  dtype='object')
```

As we want to merge our vaders result, to our original data frame(feedback), so we need to reset the index and rename it to id

```
In [73]: #vaders.reset_index().rename(columns={'index': 'Id'})
vaders["Id"] = feedback["Id"]
#vaders.merge(feedback, how='left')
#vaders.columns
```

```
In [74]: vaders.columns
```

```
Out[74]: Index(['neg', 'neu', 'pos', 'compound', 'Id'], dtype='object')
```

```
In [77]: vaders=vaders.merge(feedback, how='left')
```

Now we have sentiment score added to the orginal fields

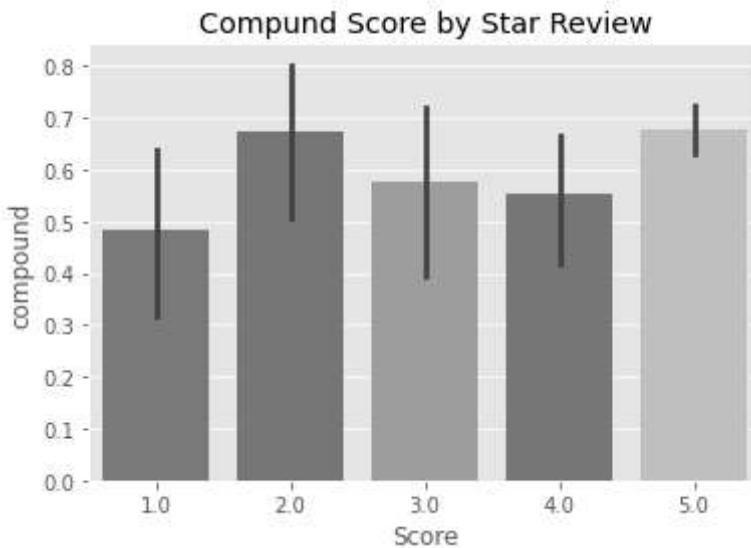
```
In [78]: vaders.head()
```

	neg	neu	pos	compound	Id	ProductId	UserId	ProfileName	Helpful
0	0.000	0.695	0.305	0.9441	2.0	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
1	0.138	0.862	0.000	-0.5664	3.0	B000LQOCHO	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	
2	0.091	0.754	0.155	0.8265	4.0	B000UA0QIQ	A395BORC6FGVXV	Karl	
3	0.000	1.000	0.000	0.0000	5.0	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	
4	0.000	0.552	0.448	0.9468	6.0	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	

Plot VADER results

Runing the Plot on vaders' data by assigning x the score value which is the star review of the person and then compound is going to be our y value and that's the negative to positive

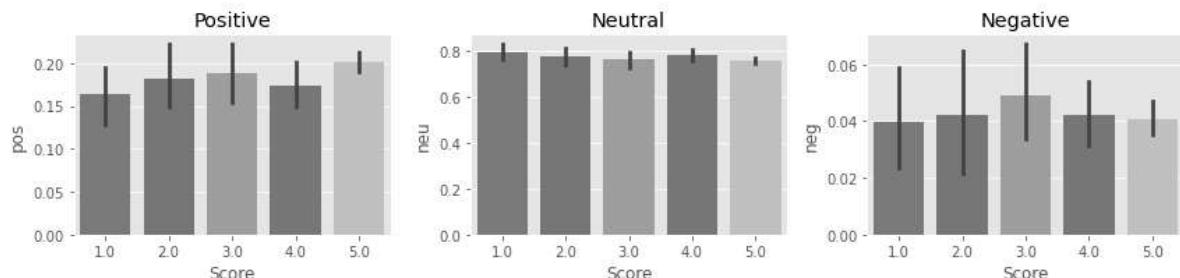
```
In [81]: ax = sns.barplot(data=vaders, x='Score', y='compound')
ax.set_title('Compund Score by Star Review')
plt.show()
```



The plot shows that one star review has lower compound score and the five star view is higher

Running the Plot based on the positive neutral and negative scores

```
In [80]: fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.tight_layout()
plt.show()
```



The plots confirm that our Vader Model is valuable in having the connection between the score of the text and sentiment score and it does relate to the actual rating review of the reviewers

Step2. Roberta Pretrained Model

Roberta model is a pre-trained transformer model

- Use a model trained of a large corpus of data.
- Transformer model accounts for the words but also the context related to other words.

As the Vader model wouldn't pick up on context (relationship between words) so we will be using the Roberta transformer based deep learning models which are very

**popular, because they can pick up on the context as well, so we're going to use from
hugging face which is one of the leaders in these types of models**