SENTIMENT ANALYSIS

1. Using VADER
2. Roberta Pretrained Model from Higging Face
3. Higging Face Pipeline

INITIAL DATA

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

plt.style.use('ggplot')

import nltk
```

```
# Importing and Reading Data
df = pd.read_csv('Reviews.csv')
print(df.shape)
df = df.head(500)
print(df.shape)
```

```
(568454, 10)
(500, 10)
```

```
df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...

Next steps: Generate code with df | View recommended plots | New interactive sheet

Performing *Basic EDA* & *Basic NLTK*

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


```
example = df['Text'][50]
print(example)
```

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

```
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
True
```

```
tokens = nltk.word_tokenize(example)
tokens[:10]
```

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

```
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
True
```

```
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

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

```
nltk.download('words')
```

```
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
True
```

```
entities = nltk.chunk.ne_chunk(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/TO
go/VB
./.)
```

VADER

```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

sia = SentimentIntensityAnalyzer()
```

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

```
{'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}
```

```
sia.polarity_scores('This is the worst thing ever.')
```

```
{'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}
```

```
sia.polarity_scores(example)
```

```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

```
# Run the polarity score on the entire dataset
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(text)
```

```
100%                                                500/500 [00:01<00:00, 375.29it/s]
```

```
vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')
```

```
# Now we have sentiment score and metadata
vaders.head()
```

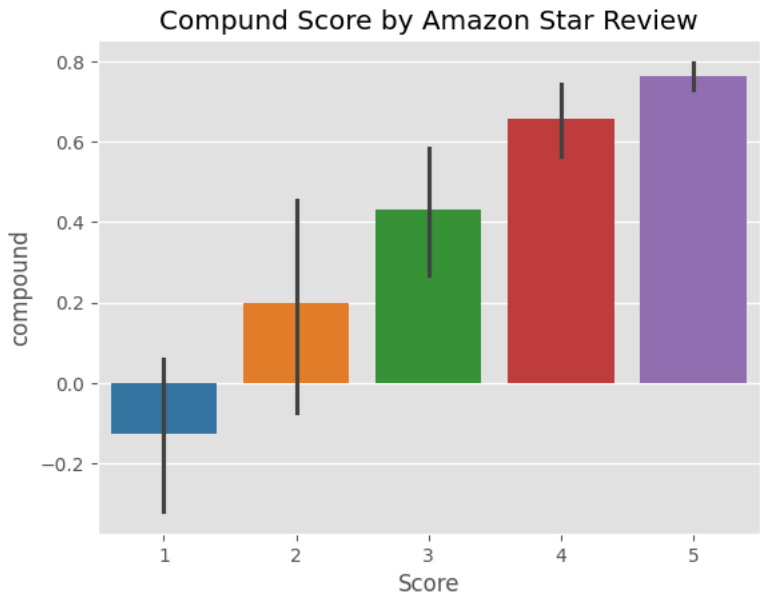
	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	T
0	1	0.000	0.695	0.305	0.9441	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130386

Next steps: Generate code with vaders | View recommended plots | New interactive sheet

Plotting Results from VADER

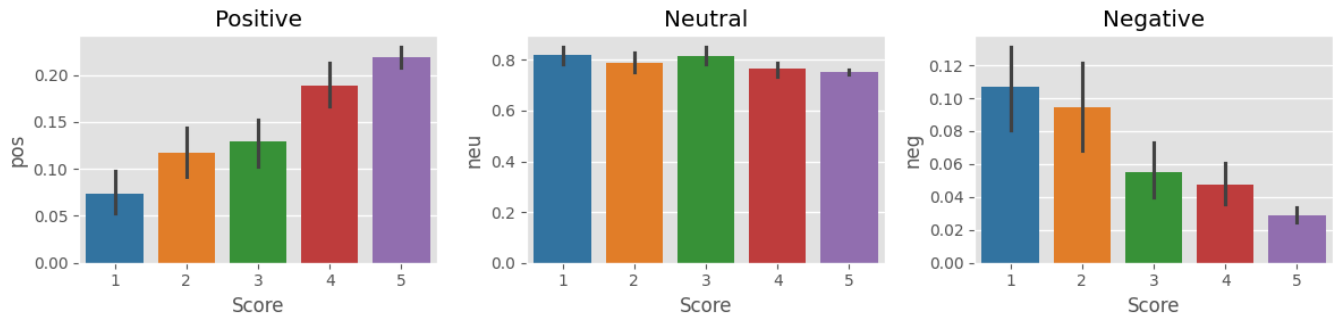
```
ax = sns.barplot(data=vaders, x='Score', y='compound', palette='tab10')
ax.set_title('Compund Score by Amazon Star Review')
plt.show()
```

```
<ipython-input-20-601582de52fd>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.barplot(data=vaders, x='Score', y='compound', palette='tab10')
```




```
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0], palette='tab10')
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1], palette='tab10')
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2], palette='tab10')
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.tight_layout()
plt.show()
```

```
<ipython-input-21-ea6c49704976>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0], palette='tab10')
<ipython-input-21-ea6c49704976>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1], palette='tab10')
<ipython-input-21-ea6c49704976>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2], palette='tab10')
```



Using ROBERTA from Hugging Face

```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
from transformers import pipeline
```

```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:104: UserWarning:
Error while fetching `HF_TOKEN` secret value from your vault: 'Requesting secret HF_TOKEN timed out. Secrets can only be fetched when running from the Colab UI.'.
You are not authenticated with the Hugging Face Hub in this notebook.
If the error persists, please let us know by opening an issue on GitHub (https://github.com/huggingface/huggingface_hub/issues/new).
  warnings.warn(
```

```
# VADER results on example
print(example)
sia.polarity_scores(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the way to go.
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

```
# Run for Roberta Model
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
   'roberta_neg' : scores[0],
   'roberta_neu' : scores[1],
   'roberta_pos' : scores[2]
}
print(scores_dict)
```

```
{'roberta_neg': 0.97635514, 'roberta_neu': 0.020687465, 'roberta_pos': 0.0029573706}
```

```
def polarity_scores_roberta(example):
   encoded_text = tokenizer(example, return_tensors='pt')
   output = model(**encoded_text)
   scores = output[0][0].detach().numpy()
   scores = softmax(scores)
   scores_dict = {
      'roberta_neg' : scores[0],
      'roberta_neu' : scores[1],
      'roberta_pos' : scores[2]
   }
   return scores_dict
```

```
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
   try:
      text = row['Text']
      myid = row['Id']
      vader_result = sia.polarity_scores(text)
      vader_result_rename = {}
      for key, value in vader_result.items():
         vader_result_rename[f"vader_{key}"] = value
      roberta_result = polarity_scores_roberta(text)
      both = {**vader_result_rename, **roberta_result}
      res[myid] = both
   except RuntimeError:
      print(f'Broke for id {myid}')
```

```
100%                                                500/500 [03:25<00:00,  2.71it/s]

Broke for id 83
Broke for id 187
```

```
results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'Id'})
results_df = results_df.merge(df, how='left')
```

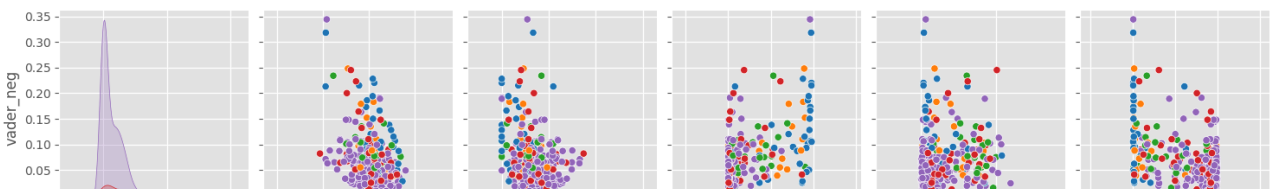
Comparing Scores

```
results_df.columns
```

```
Index(['Id', 'vader_neg', 'vader_neu', 'vader_pos', 'vader_compound',
       'roberta_neg', 'roberta_neu', 'roberta_pos', 'ProductId', 'UserId',
       'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator',
       'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

Combine and Compare

```
sns.pairplot(data=results_df,
             vars=['vader_neg', 'vader_neu', 'vader_pos',
                 'roberta_neg', 'roberta_neu', 'roberta_pos'],
            hue='Score',
            palette='tab10')
plt.show()
```



Review


```
results_df.query('Score == 1') \
  .sort_values('roberta_pos', ascending=False)['Text'].values[0]
```

'I felt energized within five minutes, but it lasted for about 45 minutes. I paid $3.99 for this drink. I could have just drunk a cup of coffee and saved my money.'

```
results_df.query('Score == 1') \
  .sort_values('vader_pos', ascending=False)['Text'].values[0]
```

'So we cancelled the order. It was cancelled without any problem. That is a positive note...'

```
results_df.query('Score == 5') \
  .sort_values('roberta_neg', ascending=False)['Text'].values[0]
```

'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'


```
results_df.query('Score == 5') \
  .sort_values('vader_neg', ascending=False)['Text'].values[0]
```

'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'

Creating Transformer Pipeline

```
from transformers import pipeline

sent_pipeline = pipeline("sentiment-analysis")
```

No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetun
Using a pipeline without specifying a model name and revision in production is not recommended.
Device set to use cpu


```
sent_pipeline('I love sentiment analysis!')
```

[{'label': 'POSITIVE', 'score': 0.9997853636741638}]

```
sent_pipeline('Make sure to like and subscribe!')
```

[{'label': 'POSITIVE', 'score': 0.9991742968559265}]

```
sent_pipeline('hooo')
```