Sentiment Analysis

Import Libraries and Load Datasets

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

plt.style.use('ggplot')
```

```
df = pd.read_csv('/content/sample_data/Reviews.csv')
```

```
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...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...

Next steps: Generate code with df New interactive sheet

```
df.shape
```

```
(36305, 10)
```

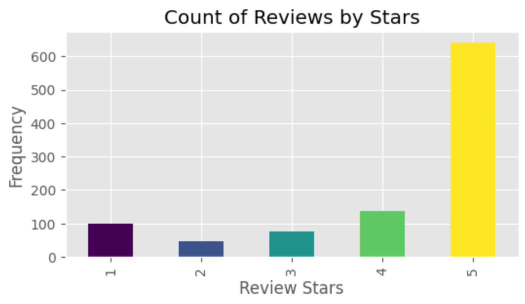
Quick EDA

```
df = df.head(1000)
df.shape
```

```
(1000, 10)
```

```
#Plot count of reviews by Stars
scores = df['Score'].value_counts().sort_index()
colors = plt.cm.viridis(np.linspace(0, 1, len(scores)))
scores.plot(kind='bar',title='Count of Reviews by Stars',
              color=colors,
              figsize=(6,3))

plt.xlabel('Review Stars')
plt.ylabel('Frequency')
plt.show()
```



Basic NLP

```
example = df['Text'][50]
example
```

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

```
from nltk import word_tokenize
nltk.download('punkt_tab')
tokens = word_tokenize(example)
tokens[:10]
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['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]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
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('maxent_ne_chunker_tab')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker_tab.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
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
  ./.)
```

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

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

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

sia = SentimentIntensityAnalyzer()
```

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

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

```
res={}
for i, row in tqdm(df.iterrows(), total=len(df)):
  text = row['Text']
  myid = row['Id']
  res[myid] = sia.polarity_scores(text)
```

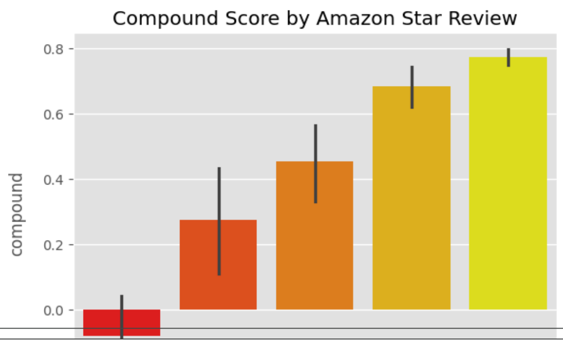
```
100%                                          1000/1000  [00:02<00:00,  293.64it/s]
```

```
vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns = {'index':'Id'})
vaders = vaders.merge(df, how='left')
vaders.head(2)
```

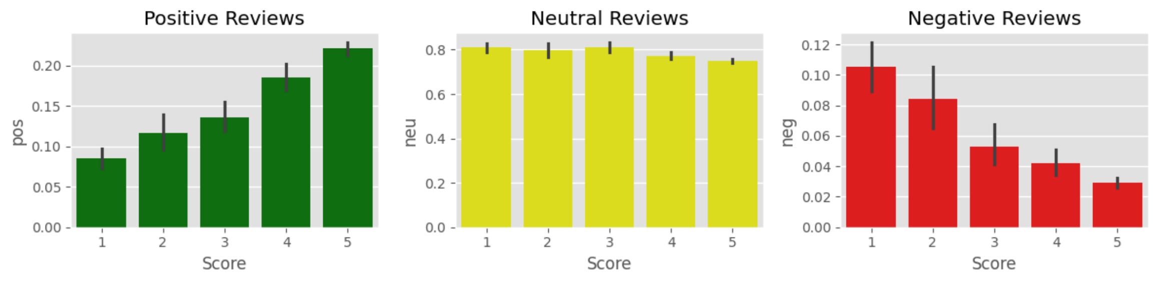
	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	0.000	0.695	0.305	0.9441	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...

Next steps: Generate code with vaders   New interactive sheet

```
# Plot Score Vs Compound Scores plot
colors = plt.cm.autumn(np.linspace(0, 1, len(scores)))
colors = colors.tolist()
ax = sns.barplot(data=vaders, x='Score', y='compound', palette=colors, hue='Score', legend = False)
ax.set_title('Compound Score by Amazon Star Review')
plt.show()
```



```
#Plot graph of Positive, Negative, Neutral
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0], color='Green')
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1], color='yellow')
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2], color='Red')
axs[0].set_title('Positive Reviews')
axs[1].set_title('Neutral Reviews')
axs[2].set_title('Negative Reviews')
plt.tight_layout()
plt.show()
```



Roberta Pretrained Model

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

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

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and res
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json:  100%                                    747/747  [00:00<00:00,  26.5kB/s]
vocab.json:       899k/?  [00:00<00:00,  15.7MB/s]
merges.txt:       456k/?  [00:00<00:00,  9.86MB/s]
special_tokens_map.json:  100%                                    150/150  [00:00<00:00,  6.34kB/s]
pytorch_model.bin:  100%                                    499M/499M  [00:05<00:00,  162MB/s]
```

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

```
{'roberta_neg': 0.9763551354408264, 'roberta_neu': 0.020687464624643326, 'roberta_pos': 0.002957369200885296}
```

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

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

```
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' : float(scores[0]),
        'roberta_neu' : float(scores[1]),
        'roberta_pos' : float(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%                                        1000/1000  [04:40<00:00,    3.81it/s]
Broke for id 83
Broke for id 187
Broke for id 529
Broke for id 540
Broke for id 746
Broke for id 863
```

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

```
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')
```

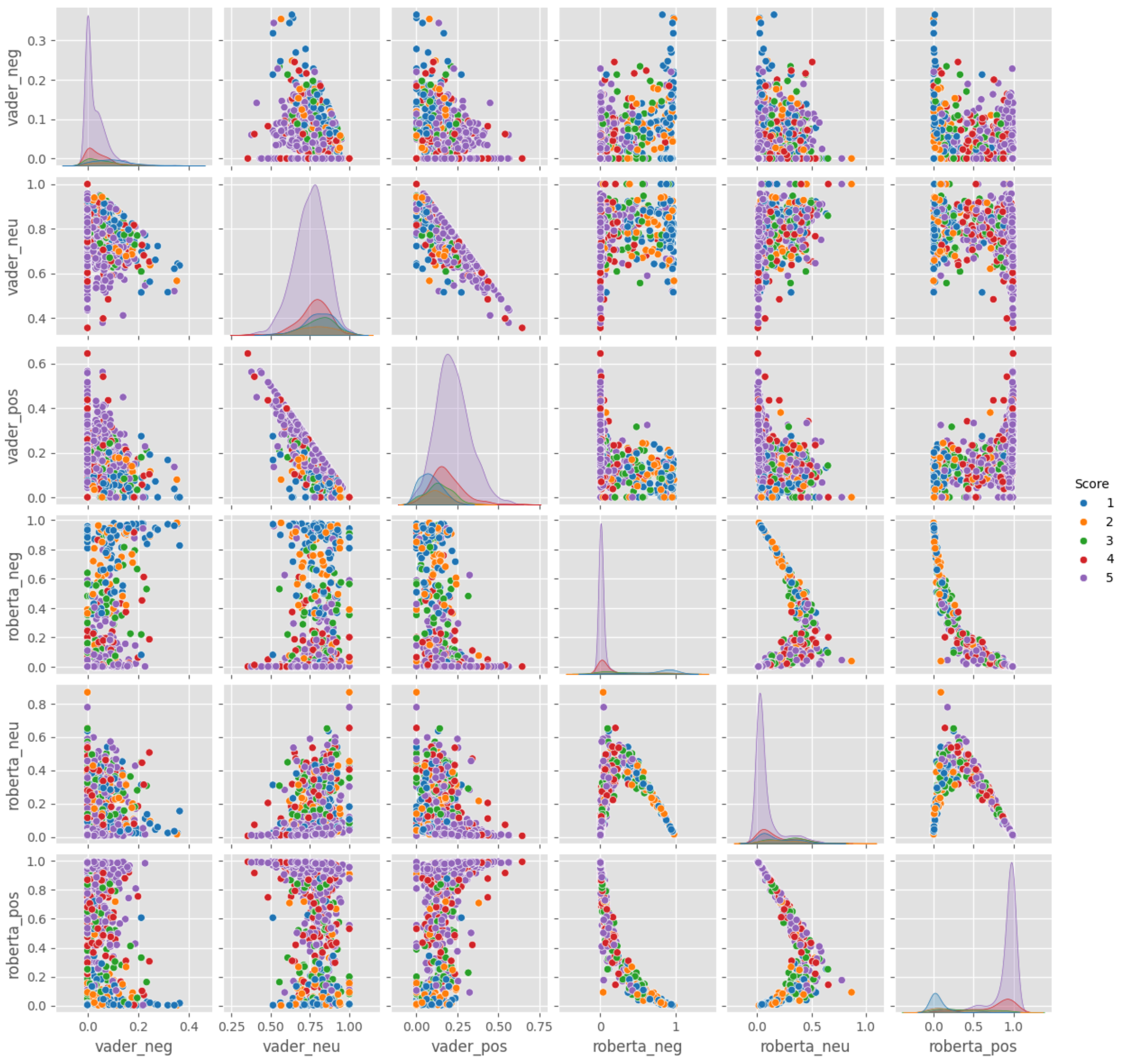
```
results_df.head()
```

	Id	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulne
0	1	0.000	0.695	0.305	0.9441	0.009624	0.049980	0.940395	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	0.138	0.862	0.000	-0.5664	0.508986	0.452413	0.038600	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	0.091	0.754	0.155	0.8265	0.003229	0.098067	0.898704	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	
3	4	0.000	1.000	0.000	0.0000	0.002295	0.090219	0.907486	B000UA0QIQ	A395BORC6FGVXV	Karl	3	
4	5	0.000	0.552	0.448	0.9468	0.001635	0.010302	0.988063	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	

Next steps: Generate code with `results_df`   New interactive sheet

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',
            height=2,
            aspect=1)
plt.show()
```



Review Examples

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

```
'I just wanted to post here that I found small bits of plastic in this food as I was feeding my 9 month old.  Plastic!!! in food!!!! baby food!!!  So please be caref
ul if you buy this or are considering it.<br /><br />My daughter LOVES this food-- it's actually her favorite.  This is the first time we have noticed plastic in it
```

```
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'
```

The Transformers 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
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%                                    629/629 [00:00<00:00, 64.2kB/s]
vocab.txt:    232k/? [00:00<00:00, 6.53MB/s]
```

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

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

```
sent_pipeline(example)
```

```
[{'label': 'NEGATIVE', 'score': 0.9994776844978333}]
```

```
pipeline_res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
  try:
    text = row['Text']
    id = row['Id']
    pipeline_res[id] = sent_pipeline(text)
  except RuntimeError:
    print(f'Escape for id {id}')
```

```
100%                                    1000/1000 [00:06<00:00, 162.05it/s]
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
Token indices sequence length is longer than the specified maximum sequence length for this model (582 > 512). Running this sequence through the model will result in
Escape for id 83
Escape for id 187
Escape for id 529
Escape for id 540
Escape for id 746
Escape for id 863
Escape for id 866
```

```
final_df = pd.DataFrame({"Id": list(pipeline_res.keys()),
                         "Pipeline Result": [pipeline_res[k][0]["label"] for k in pipeline_res]})

final_df = final_df.merge(results_df, on="Id", how="left")
```

```
final_df.head()
```

	Id	Pipeline Result	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos	ProductId	UserId	ProfileName	HelpfulnessNumerator
0	1	POSITIVE	0.000	0.695	0.305	0.9441	0.009624	0.049980	0.940395	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1
1	2	NEGATIVE	0.138	0.862	0.000	-0.5664	0.508986	0.452413	0.038600	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0
2	3	POSITIVE	0.091	0.754	0.155	0.8265	0.003229	0.098067	0.898704	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1
3	4	POSITIVE	0.000	1.000	0.000	0.0000	0.002295	0.090219	0.907486	B000UA0QIQ	A395BORC6FGVXV	Karl	3
4	5	POSITIVE	0.000	0.552	0.448	0.9468	0.001635	0.010302	0.988063	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0

Next steps: Generate code with `final_df` New interactive sheet

```
final_df['Pipeline Result'].value_counts()
```

Pipeline Result	count
POSITIVE	690
NEGATIVE	303

dtype: int64