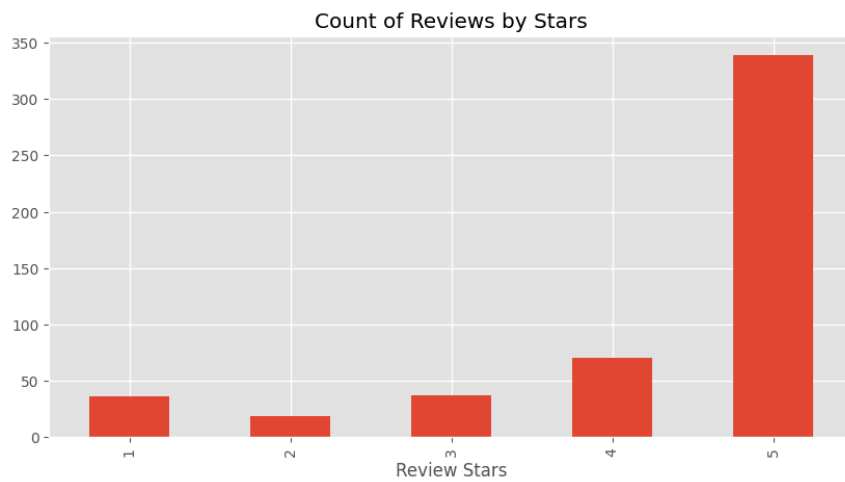


```
ax = df['Score'].value_counts().sort_index() \
    .plot(kind='bar',
          title='Count of Reviews by Stars',
          figsize=(10, 5))
ax.set_xlabel('Review Stars')
plt.show()
```



```
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.

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

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

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

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

```
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
  ./.)
```

```

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 [00:00<00:00, 1694.13it/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()

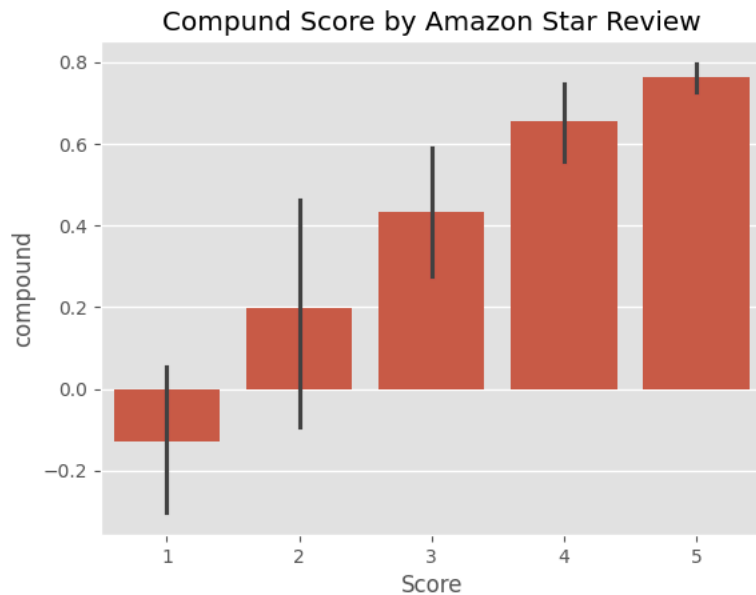
```

	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score
0	1	0.000	0.695	0.305	0.9441	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	!
1	2	0.138	0.862	0.000	-0.5664	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	.
2	3	0.091	0.754	0.155	0.8265	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	,
3	4	0.000	1.000	0.000	0.0000	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	:
4	5	0.000	0.552	0.448	0.9468	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	!

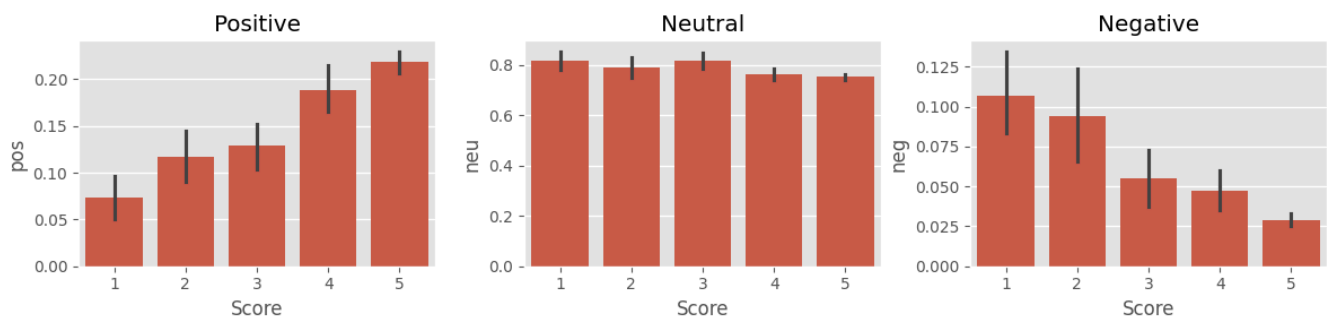
```

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

```



```
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()
```



```
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/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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
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, 50.7kB/s]
```

```
vocab.json: 100% 899k/899k [00:00<00:00, 10.9MB/s]
```

```
merges.txt: 100% 456k/456k [00:00<00:00, 13.5MB/s]
```

```
special_tokens_map.json: 100% 150/150 [00:00<00:00, 6.92kB/s]
```

```
pytorch_model.bin: 100% 499M/499M [00:04<00:00, 95.4MB/s]
```

```
/usr/local/lib/python3.10/dist-packages/torch/_utils.py:831: UserWarning: TypedStorage is deprecated. It will be removed in the fut
return self.fget.__get__(instance, owner)()
```

```
# 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': 0.0029573692}
```

```
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 [01:59<00:00, 3.52it/s]

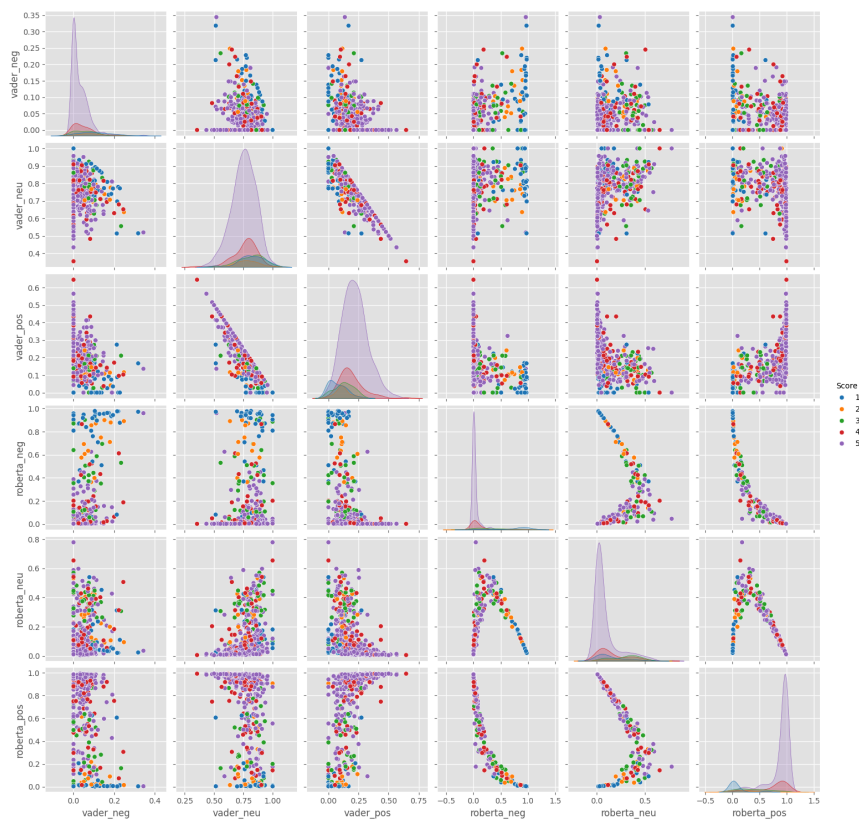
Broke for id 83
Broke for id 187
```

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

```
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')
```

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



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

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

```
from transformers import pipeline

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

No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst
Using a pipeline without specifying a model name and revision in production is not r

config.json: 100%	629/629 [00:00<00:00, 40.3kB/s]
model.safetensors: 100%	268M/268M [00:03<00:00, 80.5MB/s]
tokenizer_config.json: 100%	48.0/48.0 [00:00<00:00, 3.33kB/s]
vocab.txt: 100%	232k/232k [00:00<00:00, 7.96MB/s]

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

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