Sentiment Analysis on Amazon Fine Food Reviews

We will going to work with:

1. Using VADER Sentiment Scoring
2. Using Roberta Pretrained Model
3. Using Transformers Pipeline

Workflow:

1. Importing the necessary libraries.
2. Importing the dataset.
3. Exploring the dataset.
4. Using VADER sentiment scoring.
5. Plotting VADER result.
6. Using Roberta Pretrained model.
7. Reviewing Example.The Transformer Pipeline.
8. The Transformer Pipeline.
9. Combining and comparing.

Importing the necessary libraries

```
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('punkt') #for nltk.word_tokenize
nltk.download('averaged_perceptron_tagger') #for nltk.pos_tag
nltk.download('maxent_ne_chunker') #for nltk.chunk.ne_chunk
nltk.download('words')

# or we can do
nltk.download('all')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger_ru is already
[nltk_data]    |       up-to-date!
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package basque_grammars is already up-to-date!
[nltk_data]    | Downloading package bcp47 to /root/nltk_data...
```

```
[nltk_data]    |   Package bcp47 is already up-to-date!
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package biocreative_ppi is already up-to-date!
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package bllip_wsj_no_aux is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Package brown_tei is already up-to-date!
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Package cess_cat is already up-to-date!
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Package cess_esp is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package comparative_sentences is already up-to-
[nltk_data]    |       date!
```

Importing the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
path_to_data = '/content/drive/MyDrive/Dataset SA/Reviews.csv'
```

```
df = pd.read_csv(path_to_data)
print(df.shape)
df = df.head(500)
print(df.shape)
```

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

Exploring the dataset

```
df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Tex
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I hav bough several c th Vitalit canne d.
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Produc arrive labeled a Jumb Salte

Start coding or generate with AI.

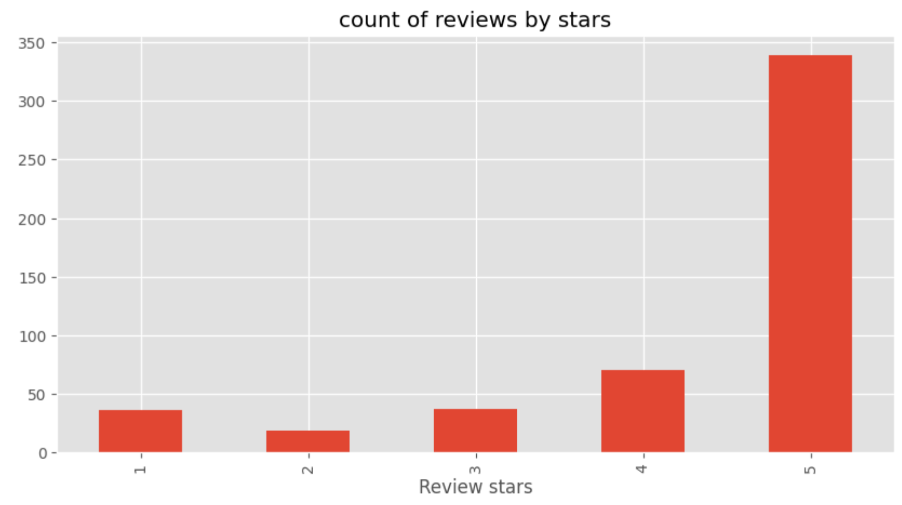
```
df['Text'].values[0]
```

```
'I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more l
ike a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than  most.'
```

```
df['Score']
# number of times each score occours
df['Score'].value_counts()
#sort index
df['Score'].value_counts().sort_index()
```

```
Score
1     36
2     18
3     37
4     70
5    339
Name: count, dtype: int64
```

```
#potting
df['Score'].value_counts().sort_index().plot(kind='bar', title='count of reviews by stars', figsize=(10,5))
axis = df['Score'].value_counts().sort_index().plot(kind='bar', title='count of reviews by stars', figsize=(10,5))
axis.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
  ./.)
```

Using VADER Seniment Scoring

We will use NLTK's SentimentIntensityAnalyzer to get the neg/neu/pos scores of the text.

This uses a "bag of words" approach:

1. Stop words are removed
2. each word is scored and combined to a total score.

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

sia = SentimentIntensityAnalyzer()
```

```
# 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)
```

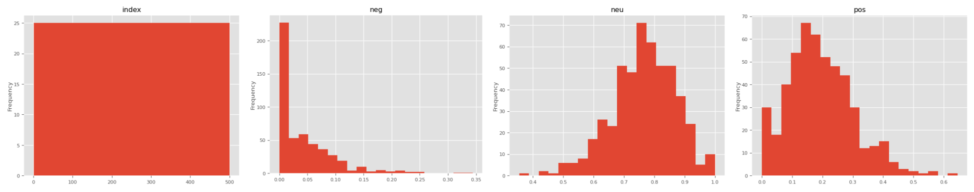
```
100%                                          500/500  [00:00<00:00,  773.51it/s]
```

```
res
#convertin in dataframe
pd.DataFrame(res)
#flip
pd.DataFrame(res).T
```

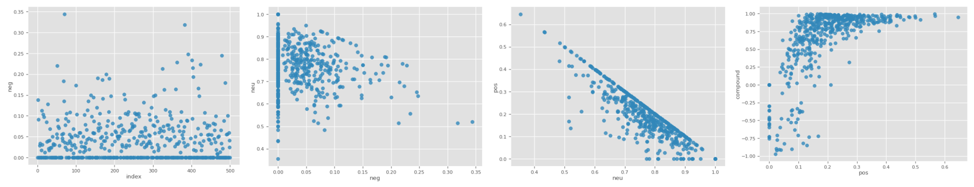
	neg	neu	pos	compound
1	0.000	0.695	0.305	0.9441
2	0.138	0.862	0.000	-0.5664
3	0.091	0.754	0.155	0.8265
4	0.000	1.000	0.000	0.0000
5	0.000	0.552	0.448	0.9468
...	...	...	...	...
496	0.000	0.554	0.446	0.9725
497	0.059	0.799	0.142	0.7833
498	0.025	0.762	0.212	0.9848
499	0.041	0.904	0.055	0.1280
500	0.000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

500 rows × 4 columns

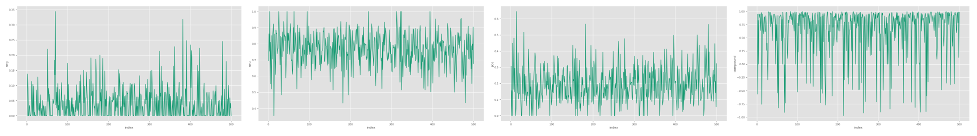
Distributions



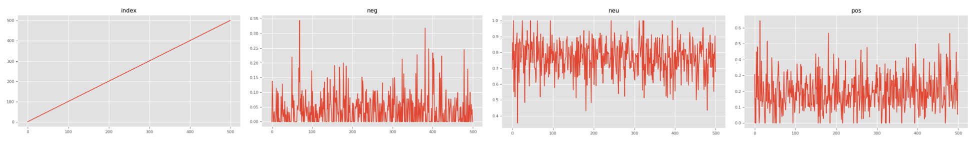
2-d distributions



Time series



Values

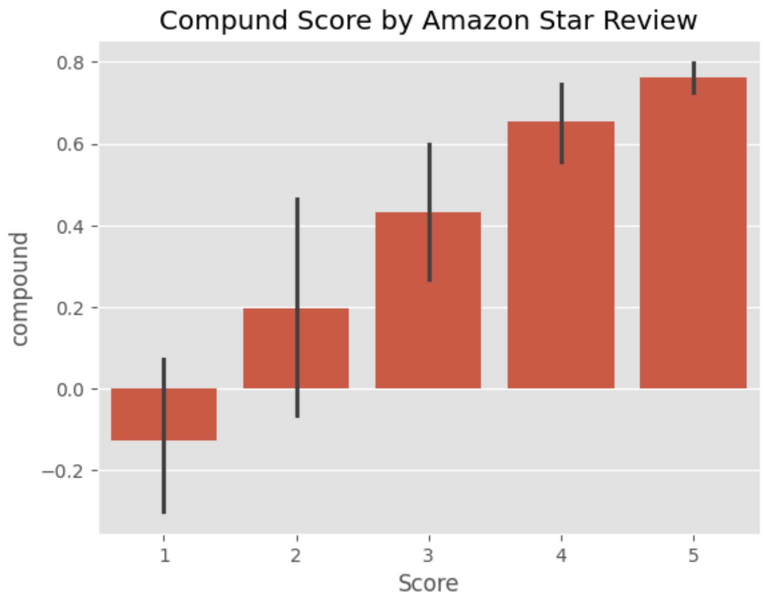


```
vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')


# 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	5	1
1	2	0.138	0.862	0.000	-0.5664	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1
2	3	0.091	0.754	0.155	0.8265	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1
3	4	0.000	1.000	0.000	0.0000	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1
4	5	0.000	0.552	0.448	0.9468	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1

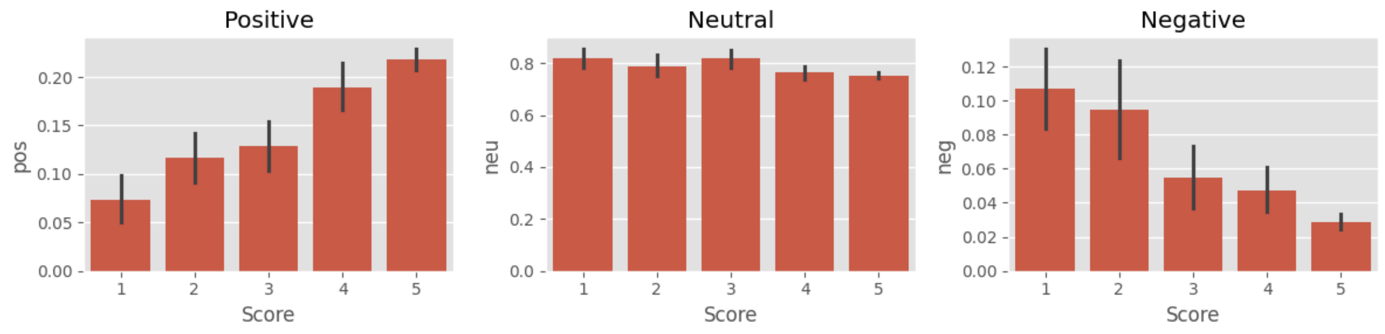
Plotting VADER Result

```
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()
```



Using Roberta Pretrained 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.

```
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)
```

```
# 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 [03:44<00:00,  2.36it/s]
Broke for id 83
Broke for id 187
```

Review Examples:

Positive 1-Star and Negative 5-Star Reviews Lets look at some examples where the model scoring and review score differ the most.

```
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'
```

```
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')
```

The Transformers Pipeline

Quick & easy way to run sentiment predictions

```
from transformers import pipeline

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

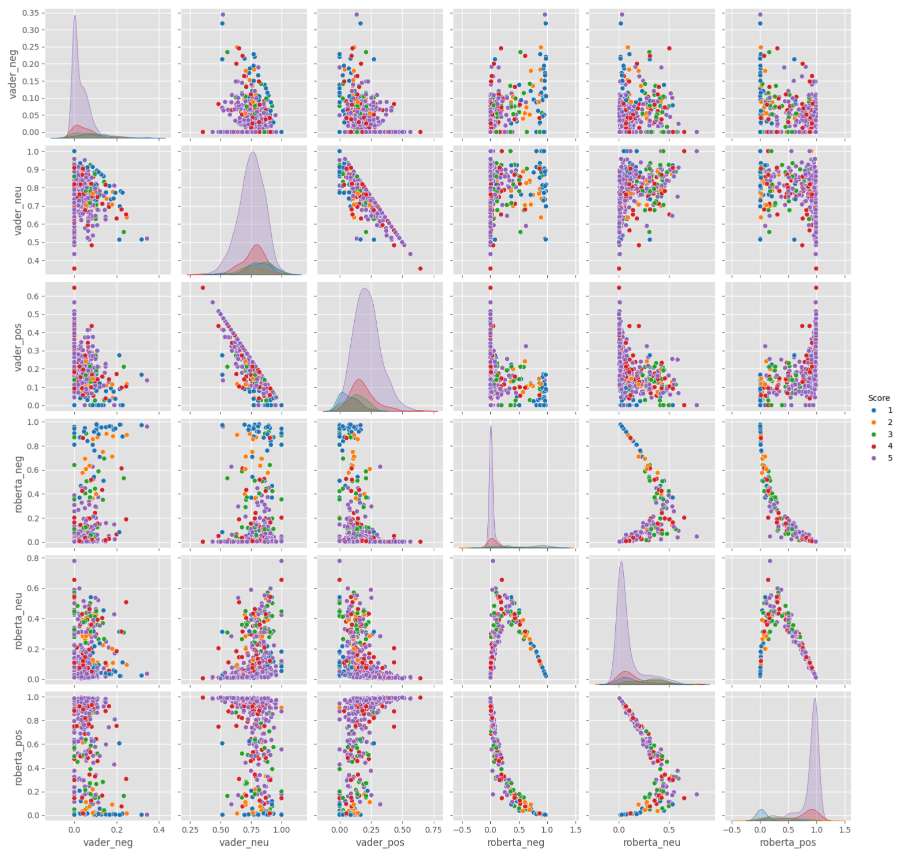
```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b (https://huggingface
Using a pipeline without specifying a model name and revision in production is not recommended.
```

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

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

```
sent_pipeline(example)

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

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()
```


vader_neg
vader_neu
vader_pos
roberta_neg
roberta_neu
roberta_pos
Score