CHAPTER 1

INTRODUCTION

1.1 SENTIMENT ANALYSIS FOR AMAZON FOOD REVIEW

Analyzing sentiment in reviews is indeed a valuable application of machine learning in today's digitized world. With the abundance of reviews available online, it's essential to have tools that can quickly parse through them and provide insights into the overall sentiment of the customers.

The Amazon Fine Food Reviews dataset is a popular choice for sentiment analysis projects due to its extensive collection of food reviews along with their corresponding ratings. Using machine learning algorithms, you can train models to classify these reviews as either positive or negative, providing valuable insights for consumers and businesses alike.

By automating the process of sentiment analysis, businesses can gain valuable insights into customer opinions at scale, allowing them to make informed decisions about their products and services. Moreover, consumers can benefit from these analyses by quickly gauging the overall sentiment of a product before making a purchase decision.

In today's digital age, online reviews play a pivotal role in shaping consumer purchasing decisions. With the exponential growth of e-commerce platforms like Amazon, where millions of products are available at the fingertips of consumers, the importance of analyzing sentiments expressed in reviews has never been more significant. In this context, sentiment analysis, a subfield of Natural Language Processing (NLP), emerges as a crucial tool for extracting insights from vast amounts of textual data.

Facts and Figures

- According to Statista, global e-commerce sales are projected to surpass $6.3 trillion by 2024.
- Pervasive Influence of Online Reviews: Research indicates that a staggering 93% of consumers read online reviews before making a purchasing decision (Source: Podium). Furthermore, BrightLocal's Local Consumer Review Survey found that 87% of consumers trust online reviews as much as personal recommendations.
- Amazon Fine Food Reviews Dataset: One of the most widely used datasets for sentiment analysis is the Amazon Fine Food Reviews dataset, containing over 500,000 reviews of food products from Amazon users. This dataset provides a rich source of textual data for training sentiment analysis models.

Various Techniques for Sentiment Analysis

- **Natural Language Processing (NLP):** NLP techniques are fundamental to sentiment analysis. These techniques involve processing and analyzing human language data, enabling computers to understand, interpret, and generate human-like text. NLP tasks such as tokenization, stemming, and part-of-speech tagging are often employed in sentiment analysis pipelines.
- **Feature Extraction:** Feature extraction methods like Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are commonly used to convert textual data into numerical representations that machine learning algorithms can understand. BoW represents text as a sparse matrix of word occurrences, while TF-IDF assigns weights to words based on their importance in the corpus.
- **Machine Learning Algorithms:** A wide range of machine learning algorithms can be applied to sentiment analysis tasks. These include traditional algorithms such as Support Vector Machines (SVM), Naive Bayes, and Decision Trees, as well as more advanced techniques like Recurrent Neural Networks (RNNs) and Transformers. Supervised learning approaches are typically employed, where models are trained on labeled data to predict the sentiment of unseen reviews.
- **Deep Learning:** Deep learning models, particularly neural networks, have shown remarkable performance in sentiment analysis tasks. Architectures like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks excel at capturing intricate patterns and

dependencies in sequential data, making them well-suited for analyzing textual data with complex sentiments.

These additional techniques broaden the spectrum of sentiment analysis methodologies, showcasing the versatility of approaches available to researchers and practitioners. Whether employing rule-based methods like VADER, leveraging cutting-edge transformer models like ROBERTA, or utilizing user-friendly pipelines provided by platforms like Hugging Face, sentiment analysis practitioners have a diverse toolkit at their disposal to extract valuable insights from textual data across various domains and applications. Sentiment analysis is used to extract valuable insights from vast amounts of textual data, enabling businesses to understand customer sentiments and make data-driven decisions in the competitive e-commerce landscape.

1.2 PURPOSE

Sentiment analysis leverages machine learning and data mining techniques to analyze customer opinions, market trends, and brand reputation across various domains. Its purposes include understanding customer preferences, identifying market trends, managing brand reputation, informing product development, optimizing marketing strategies, and mitigating risks. Ultimately, sentiment analysis enables data-driven decision-making and enhances stakeholder experiences.

1.3 OBJECTIVE

- **Data Collection and Preprocessing**: Gather and preprocess comprehensive datasets containing relevant features from Amazon food reviews, including attributes such as ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, and Text.
- **Exploratory Data Analysis (EDA)**: Explore and analyze the datasets to identify patterns and correlations indicative of sentiment expressed in Amazon food reviews. This involves understanding the distribution of review scores, examining the relationship between review length and sentiment, and exploring temporal trends in reviews.
- **Machine Learning Model Development**: Develop and implement machine learning algorithms, such as logistic regression, support vector machines (SVM), decision trees, and ensemble methods, to build predictive models for sentiment analysis of Amazon food reviews.
- **Model Evaluation**: Evaluate the performance of the developed models using appropriate metrics such as accuracy, precision, recall, and F1 score to assess their effectiveness in classifying reviews as positive, negative, or neutral.
- **Feature Selection**: Utilize feature selection techniques such as recursive feature elimination (RFE), principal component analysis (PCA), and correlation analysis to identify the most informative features from the Amazon food review dataset.
- **Model Optimization**: Optimize the predictive models through hyperparameter tuning and cross-validation to enhance their

generalizability and robustness in sentiment analysis of Amazon food reviews.

- **Model Validation**: Validate the final predictive model using techniques such as k-fold cross-validation or by comparing model performance on independent datasets to ensure its reliability and effectiveness in real-world scenarios.
- **Deployment**: Develop a user-friendly interface or application to deploy the predictive model for real-world use, allowing healthcare professionals to analyze sentiment in Amazon food reviews efficiently and effectively. This interface should provide intuitive access to sentiment analysis results and insights derived from the predictive model.

1.4 METHODOLOGY

1.4.1 DESCRIPTION OF THE DATA SET

Sentiment Analysis for Amazon Food Review

Source - Kaggle Notebook: https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

Description: Analyze and predict sentiment in Amazon food reviews using machine learning techniques.

Solution Suggestion

- **VADER (Valence Aware Dictionary and Sentiment Reasoner**): VADER is a rule-based sentiment analysis tool specifically designed for social media text. It operates by assigning sentiment scores to individual words or phrases based on a pre-defined lexicon that includes sentiment polarity and intensity information. VADER is known for its ability to handle colloquial language, emojis, and slang, making it particularly useful for analyzing sentiment in informal text sources like social media posts and online reviews.
- **RoBERTa (Robustly optimized BERT approach):** Roberta is a transformer-based language model developed by Facebook AI. It is an extension of the BERT (Bidirectional Encoder Representations from Transformers) model, trained on a larger corpus and with additional pre-training objectives. RoBERTa exhibits improved performance on various natural language understanding tasks, including sentiment analysis, due to its enhanced pre-training methodology and larger model size.
- **Hugging Face's Transformers Pipeline:** Hugging Face provides a user-friendly interface for leveraging state-of-the-art transformer models, including BERT, RoBERTa, GPT, and others, through its Transformers library. The Transformers pipeline allows users to perform various NLP tasks, including sentiment analysis, with minimal code and configuration. By utilizing pre-trained transformer models fine-tuned on sentiment analysis datasets, the pipeline can accurately predict sentiment labels for input text, making it a convenient option for practitioners seeking high-performance models without extensive training requirements.

About Dataset

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

Attribute Descriptions

ProductId - Unique identifier for the product

UserId - Unqiue identifier for the user

ProfileName - Profile name of the user

HelpfulnessNumerator - Number of users who found the review helpful

HelpfulnessDenominator - Number of users who indicated whether they found the review helpful or not

Score - Rating between 1 and 5

Time - Timestamp for the review

Summary - Brief summary of the review

Text - Text of the review

1.5 PREPROCESSING THE DATASET

Some common steps in data preprocessing include

Data preprocessing is an important step in the data mining process that involves cleaning and transforming raw data to make it suitable for analysis. Some common steps in data preprocessing include:

- **Data Cleaning** - This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.
- **Data Integration** - This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.
- **Data Transformation** - This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.
- **Data Reduction** - This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.
- **Data Discretization** - This involves dividing continuous data into discrete categories or intervals. Discretization is often used in data mining and machine learning algorithms that require categorical data. Discretization can be achieved through techniques such as equal width binning, equal frequency binning, and clustering.
- **Data Normalization** - This involves scaling the data to a common range, such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales. Common normalization techniques include min-max normalization, z-score normalization, and decimal scaling.

Data preprocessing plays a crucial role in ensuring the quality of data and the accuracy of the analysis results. The specific steps involved in data preprocessing may vary depending on the nature of the data and the analysis goals.

By performing these steps, the data mining process becomes more efficient and the results become more accurate.

Preprocessing in Data Mining: Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

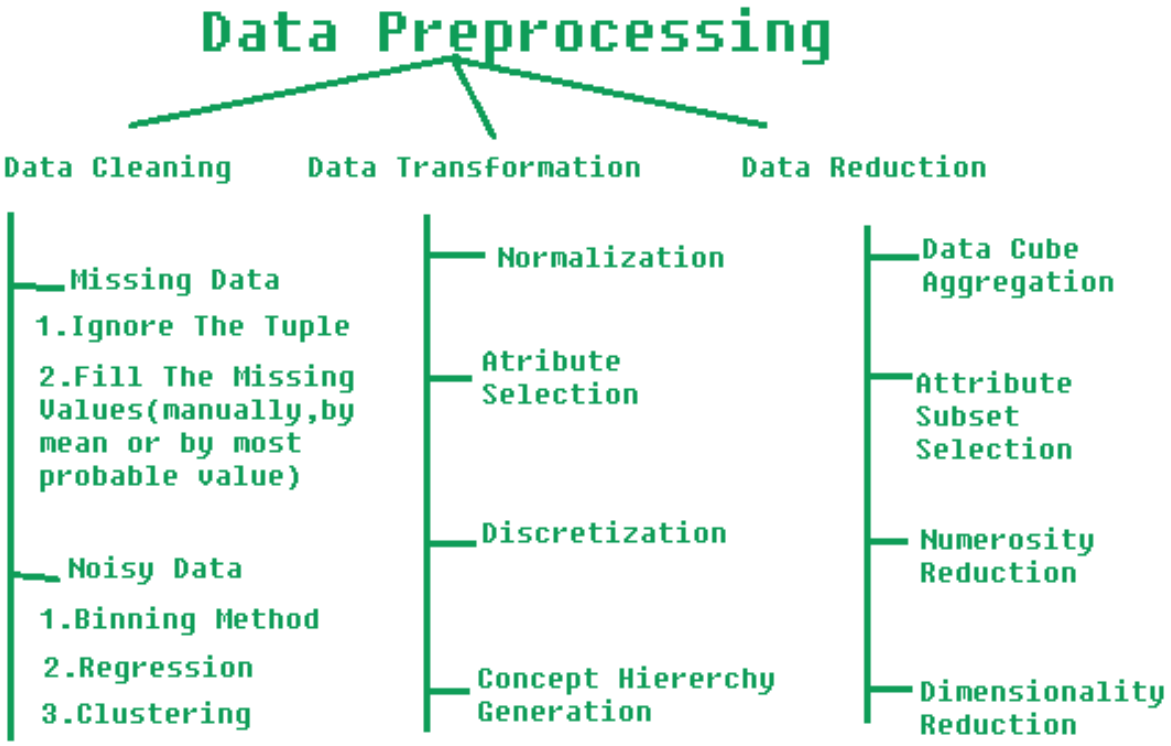

Fig 1.1 Data Preprocessing

Steps Involved in Data Preprocessing:

1. Data Cleaning - The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a) Missing Data - This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

- **Ignore the tuples** - This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
- **Fill the Missing values** - There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

(b) Noisy Data - Noisy data is a meaningless data that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways:

- **Binning Method** - This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
- **Regression** - Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one independent variable.
- **Clustering** - This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation - This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

- **Normalization** - It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)
- **Attribute Selection** - In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
- **Discretization** - This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
- **Concept Hierarchy Generation** - Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

3. Data Reduction - Data reduction is a crucial step in the data mining process that involves reducing the size of the dataset while preserving the important information. This is done to improve the efficiency of data analysis and to avoid overfitting of the model. Some common steps involved in data reduction are:

- **Feature Selection** - This involves selecting a subset of relevant features from the dataset. Feature selection is often performed to remove irrelevant or redundant features from the dataset. It can be done using various techniques such as correlation analysis, mutual information, and principal component analysis (PCA).
- **Feature Extraction** - This involves transforming the data into a lower-dimensional space while preserving the important information. Feature extraction is often used when the original features are high-dimensional and complex. It can be done using techniques such as PCA, linear discriminant analysis (LDA), and non-negative matrix factorization (NMF).
- **Sampling** - This involves selecting a subset of data points from the dataset. Sampling is often used to reduce the size of the dataset while preserving the important information. It can be done using techniques such as random sampling, stratified sampling, and systematic sampling.
- **Clustering** - This involves grouping similar data points together into clusters. Clustering is often used to reduce the size of the dataset by replacing similar data points with a representative centroid. It can be done using techniques such as k-means, hierarchical clustering, and density-based clustering.
- **Compression** - This involves compressing the dataset while preserving the important information. Compression is often used to reduce the size of the dataset for storage and transmission purposes. It can be done using techniques such as wavelet compression, JPEG compression, and gzip compression.

CHAPTER 2

CODING, TESTING AND IMPLEMENTATION

2.1 INTRODUCTION

There are several reasons why Python might be a good choice for creating an AI solution for the visually impaired

- Python is a widely used, general-purpose programming language with a large, active community of developers. This makes it easier to find resources, documentation, and support when working with Python.
- **Dynamically Typed** - Python is a dynamically-typed, interpreted language, which means it can be easier to write and debug compared to compiled languages. This can make it more suitable for rapid prototyping and iterative development.
- **Free** - Python is free to use and distribute and is supported by community. Python interpreter is available for every major platform.
- **Simplicity and Readability** - Python has a simple, easy-to-learn syntax that emphasizes readability and reduces the cost of program maintenance. Large standard library: Python comes with a large standard library that supports many common programming tasks, such as connecting to web servers, reading and writing files, and working with data.
- **Third-party Libraries** - There is a large ecosystem of third-party libraries and frameworks available for Python, including libraries for scientific computing, machine learning, and natural language processing. This makes it easy to add additional functionality to your Python programs.
- **Cross-Platform** - Python is available on a wide range of platforms, including Windows, macOS, Linux, and many others. This makes it a good choice for developing programs that need to run on multiple platforms. Strong community: Python has a large, active community of developers, which makes it easier to find support and resources when you're working with the language.

2.1.1 HARDWARE REQUIREMENTS

The hardware requirements for creating sentiment analysis for amazon food review will depend on the specific capabilities and features you want to include in your system. Here are a few hardware considerations to keep in mind:

- Operating system - You will need a computer with an operating system (e.g. Windows, macOS, Linux) to run the software tools and libraries required to build your system.
- Computing Power - A powerful CPU (Central Processing Unit) is essential for processing large amounts of text data efficiently. Modern multi-core processors or even better, CPUs optimized for parallel processing such as those used in high-performance computing (HPC) systems, can significantly speed up the processing of the data. Alternatively, you can leverage GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units) for even faster processing, especially if you're dealing with deep learning models which can benefit greatly from parallel computation.
- Memory (RAM) - Sufficient RAM is crucial for handling the data and model efficiently, especially when working with large datasets. A minimum of 8GB RAM is usually recommended for basic sentiment analysis tasks, but for more complex analyses or larger datasets, 16GB or more may be required.
- Storage - Adequate storage space is necessary for storing the dataset(s), pre-trained word embeddings or models, and any intermediate or final results generated during the analysis. The amount of storage needed depends on the size of the dataset and any additional resources required for the sentiment analysis.
- Network Connectivity - Depending on your setup, you may need reliable internet access for downloading datasets, accessing APIs (if you're using any external services), or deploying the sentiment analysis model to a cloud platform for scalable processing.
- Backup and Redundancy - Implementing backup systems and redundancy measures can help ensure data integrity and prevent loss of progress or results in case of hardware failures.

2.1.2 SOFTWARE REQUIREMENT

IDE: JUPYTER NOTEBOOK

The IDE used in this project is Jupyter Notebook. All the python files were created in Jupyter Notebook and all the necessary packages were easily installable in this IDE. It is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

2.2 SAMPLE CODE LAYOUTS

LIBRARIES

Pandas:

- **Use:** Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures like Data Frames and Series, which are efficient for handling structured data.
- **Definition:** Pandas simplifies tasks such as reading and writing data, cleaning and transforming data, and performing descriptive statistics. It's widely used in data science, machine learning, and data analysis projects.

NumPy:

- **Use:** NumPy is a fundamental package for numerical computing in Python. It provides support for arrays, matrices, and mathematical functions to operate on these arrays efficiently.
- **Definition:** NumPy's primary object is the ndarray (N-dimensional array), which is a fast and flexible container for large datasets in Python. It is used for mathematical and logical operations on arrays, including linear algebra, Fourier analysis, and random number generation.

Matplotlib:

- **Use:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface and is highly customizable.
- **Definition:** Matplotlib allows users to create a wide variety of plots and charts, including line plots, scatter plots, bar plots, histograms, and more. It's often used for data exploration, presentation, and publication-quality graphics.

Seaborn:

- **Use:** Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Definition:** Seaborn simplifies the process of creating complex visualizations by providing built-in themes and functions for common statistical plots. It's particularly useful for visualizing relationships between variables, such as scatter plots, pair plots, and regression plots.

NLTK (Natural Language Toolkit):

- **Use:** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and more.
- **Definition:** NLTK is widely used in natural language processing (NLP) tasks such as text classification, sentiment analysis, machine translation, and information extraction. It's a valuable tool for researchers, educators, and developers working with textual data.

CODE

Python Sentiment Analysis Project with NLTK and 🤗 Transformers.

VADER (Valence Aware Dictionary and sentiment Reasoner) - Bag of words approach

Roberta Pretrained Model from 🤗

Hugging face Pipeline

Read in Data and NLTK Basics

- **Pandas -** Pandas is a powerful Python library for data manipulation and analysis. It provides data structures like DataFrame and Series, which are highly efficient for working with structured data.
- **NumPy -** NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
- **Matplotlib -** Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface for creating plots and charts, allowing users to create a wide variety of visualizations.
- **Seaborn -** Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations like heatmaps, violin plots, and pair plots.
- **NLTK (Natural Language Toolkit) -** It's a popular Python library for working with human language data. It provides easy-to-use interfaces to

over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. It's widely used in natural language processing (NLP) tasks such as sentiment analysis, text classification, machine translation, and more. In short, nltk is a powerful tool for working with text data in Python.

- **plt.style.use('ggplot') -** Sets the style of Matplotlib plots to mimic the aesthetics of the "ggplot2" package in R, which is known for producing visually appealing and highly customizable plots. The 'ggplot' style in Matplotlib typically features a gray background, bold lines, and distinctive colors for different elements like data points, axes, and labels. This command allows you to quickly apply this specific style to your Matplotlib plots, giving them a consistent and polished look.

The Steps in Machine Learning Projects

1. **AFTERS**
2. **Acquire Data**
3. **Filter Data**
4. **Transform Data**
5. **Explore Data**
6. **Split Data**

1 Acquire Data or Data Acquisition

- **Extract Data**
- **Scrape Data**
- **Query Data**
- **Collect Data**
- **Combine Data**

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
import nltk
```

Reading Data Set

```
# Read in data
df = pd.read_csv('Reviews.csv')
print(df.shape)
df = df.head(500)
print(df.shape)
```

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

Scanning the data set

```
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...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
df.shape
```

```
(500, 10)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Id                      500 non-null    int64
 1   ProductId               500 non-null    object
 2   UserId                  500 non-null    object
 3   ProfileName             500 non-null    object
 4   HelpfulnessNumerator    500 non-null    int64
 5   HelpfulnessDenominator  500 non-null    int64
 6   Score                   500 non-null    int64
 7   Time                    500 non-null    int64
 8   Summary                 500 non-null    object
 9   Text                    500 non-null    object
dtypes: int64(5), object(5)
memory usage: 39.2+ KB
```

```
pd.set_option('display.float_format', lambda x: '%.3f' %x)
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Id	500.000	250.500	144.482	1.000	125.750	250.500	375.250	500.000
HelpfulnessNumerator	500.000	0.952	2.046	0.000	0.000	0.000	1.000	19.000
HelpfulnessDenominator	500.000	1.276	2.489	0.000	0.000	0.000	2.000	19.000
Score	500.000	4.316	1.203	1.000	4.000	5.000	5.000	5.000
Time	500.000	1294819603.200	50724372.281	1107820800.000	1267790400.000	1312977600.000	1334620800.000	1351209600.000

```
df.describe()
```

	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
count	500.000	500.000	500.000	500.000	500.000
mean	250.500	0.952	1.276	4.316	1294819603.200
std	144.482	2.046	2.489	1.203	50724372.281
min	1.000	0.000	0.000	1.000	1107820800.000
25%	125.750	0.000	0.000	4.000	1267790400.000
50%	250.500	0.000	0.000	5.000	1312977600.000
75%	375.250	1.000	2.000	5.000	1334620800.000
max	500.000	19.000	19.000	5.000	1351209600.000

2 Data Filtering

- **Outliers**
- **Error/Noise**
- **Duplicates**
- **Invalid Data**

```
df.shape
```

```
(500, 10)
```

```
df.isnull()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
495	False	False	False	False	False	False	False	False	False	False
496	False	False	False	False	False	False	False	False	False	False
497	False	False	False	False	False	False	False	False	False	False
498	False	False	False	False	False	False	False	False	False	False
499	False	False	False	False	False	False	False	False	False	False

500 rows × 10 columns

Checking values are missing in the data set

```
df.isnull().sum()#*100/len(df)
```

```
Id                      0
ProductId               0
UserId                  0
ProfileName             0
HelpfulnessNumerator    0
HelpfulnessDenominator  0
Score                   0
Time                    0
Summary                 0
Text                    0
dtype: int64
```

```
df.isnull().sum()*100/len(df)
```

```
Id                      0.000
ProductId               0.000
UserId                  0.000
ProfileName             0.000
HelpfulnessNumerator    0.000
HelpfulnessDenominator  0.000
Score                   0.000
Time                    0.000
Summary                 0.000
Text                    0.000
dtype: float64
```

How to detect misssing values

```
# !pip install missingno
```

```
import missingno as mano
```

```
mano.bar(df)
plt.show()
```

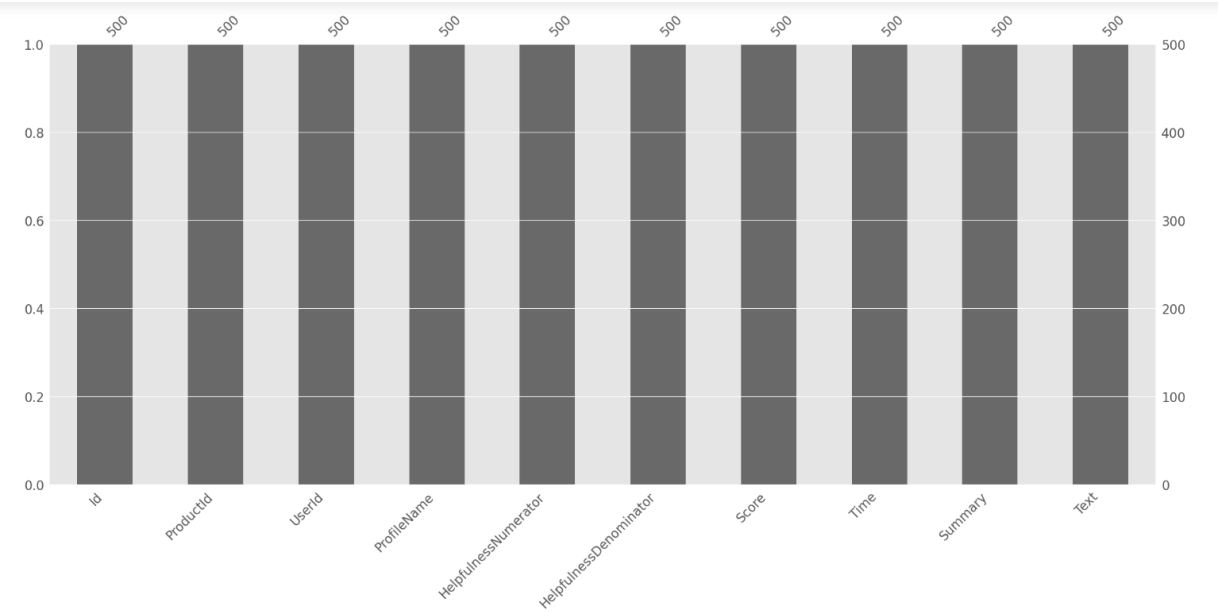

Fig 2.1 Bar graph for checking missing values

```
mano.matrix(df)
plt.show()
```

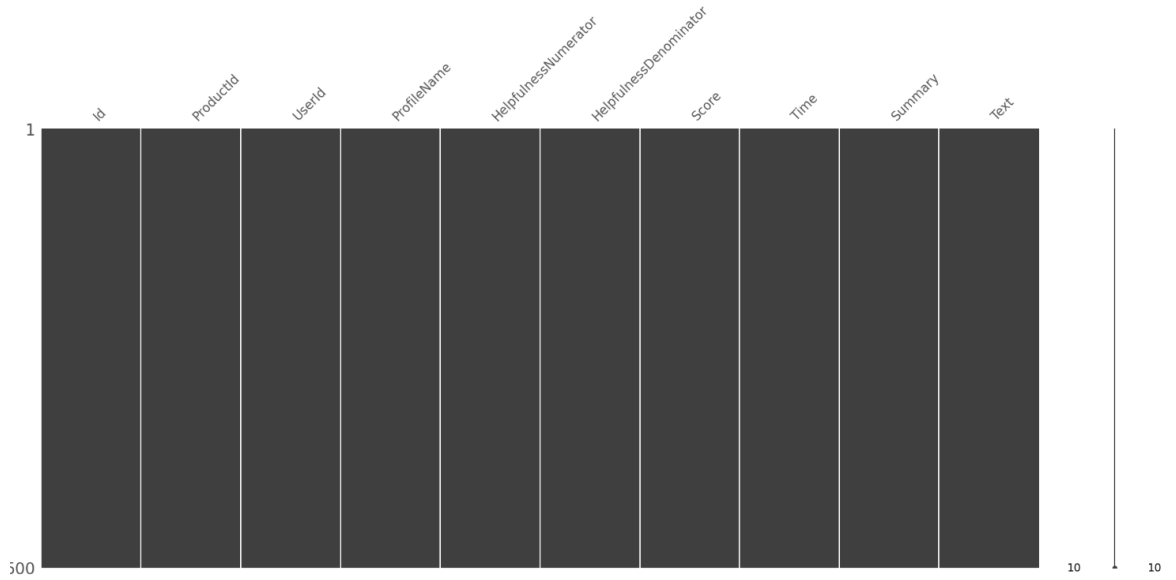

Fig 2.2 Matrix for checking missing values

- **As you can see in the above bar graph and matrix diagram there are no gaps its simply because the data set that is being used has no attributes with null value.**
- **Now let's see an example – What will be the matrix diagram of your data set if it contains null value and how we can fill those null values with the mean of that attribute.**

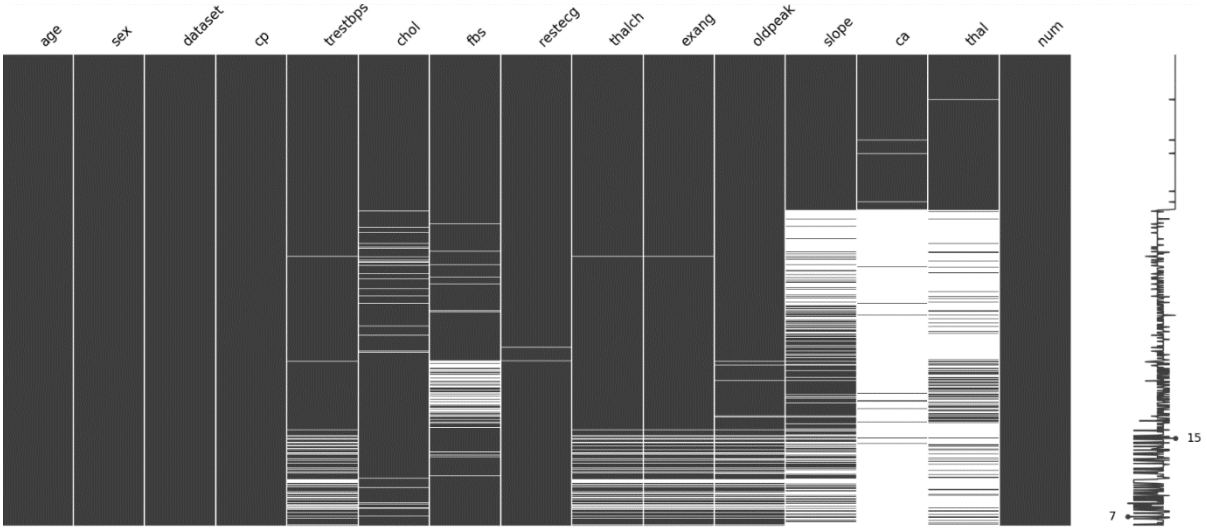

Fig 2.3 Matrix for checking missing values if they exist

```
# Run this code to see the image
import cv2
image = cv2.imread('image.png')
cv2.imshow("Heart Disease Prediction",image)
cv2.waitKey(0)
```

```
-1
```

```
# In this case the data set has no null values but if there is a column with missing values you can fill it by mean value
# df['trestbps'] = df['trestbps'].fillna(df['trestbps'].mean())
# df['chol'] = df['chol'].fillna(df['chol'].mean())
# df['fbs'] = df['fbs'].fillna(df['fbs'].mean())
# df['thalch'] = df['thalch'].fillna(df['thalch'].mean())
# df['exang'] = df['exang'].fillna(df['exang'].mean())
# df['oldpeak'] = df['oldpeak'].fillna(df['oldpeak'].mean())
```

Checking for duplication

```
df.duplicated().sum()
```

```
0
```

3 Data Transformation

- **Formating**
- **Data type conversion**
- **Datetime cast**
- **Translation Mapping**
- **Text to Interger/Float**

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Id                      500 non-null    int64 
 1   ProductId               500 non-null    object
 2   UserId                  500 non-null    object
 3   ProfileName             500 non-null    object
 4   HelpfulnessNumerator    500 non-null    int64 
 5   HelpfulnessDenominator  500 non-null    int64 
 6   Score                   500 non-null    int64 
 7   Time                    500 non-null    int64 
 8   Summary                 500 non-null    object
 9   Text                    500 non-null    object
dtypes: int64(5), object(5)
memory usage: 39.2+ KB
```

Converting Data type object to catergory

```
# df.info()
```

```
# Convert 'column_name' from 'object' to 'category'
# df['ProductId'] = df['ProductId'].astype('category')
# df['UserId'] = df['UserId'].astype('category')
# df['ProfileName'] = df['ProfileName'].astype('category')
# df['Summary'] = df['Summary'].astype('category')
# df['Text'] = df['Text'].astype('category')
```

```
# df.info()
```

- **For learning purpose**
- **We can also assign numbers to these categorical data and vise versa ex age number into [Young, Young Adult, Adult Old]**
- **Converting age from numeric to category**

```
# # Extract the 'age' column
# ages = df['age']

# # Define the bin ranges and category labels
# bins = [0, 25, 40, 55, 70, float('inf')]
# labels = ['Young', 'Young adults', 'Adults', 'Middle aged', 'Old']

# # Bin the ages into categories
# age_categories = pd.cut(ages, bins=bins, labels=labels, right=False)

# # Replace the 'age' column with the categorized ages
# df['age'] = age_categories

# print(df)
```

4 Data Exploration

- **Trends**
- **Relatioships**
- **Anamolies**
- **Patterns**
- **Uni,Bi Multivariate**
- **Hypothesis testing**
- **Data Visualization**

```
df['Score'].unique()
```

```
array([5, 1, 4, 2, 3], dtype=int64)
```

```
df['Score'].value_counts()
```

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

```
df.groupby('Score').mean()
```

```
C:\Users\psriz\AppData\Local\Temp\ipykernel_12596\4169637139.py:1: FutureWarning: The default value of numeric_only in DataFram
eGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select onl
y columns which should be valid for the function.
  df.groupby('Score').mean()
```

Score	Id	HelpfulnessNumerator	HelpfulnessDenominator	Time
1	236.306	0.778	2.611	1307527200.000
2	224.667	0.889	1.833	1286001600.000
3	228.838	1.054	1.892	1274621837.838
4	272.771	1.114	1.229	1286488594.286
5	251.145	0.929	1.047	1297863079.646

```
df['Score'].value_counts().plot(kind = 'bar')
```

```
<Axes: >
```

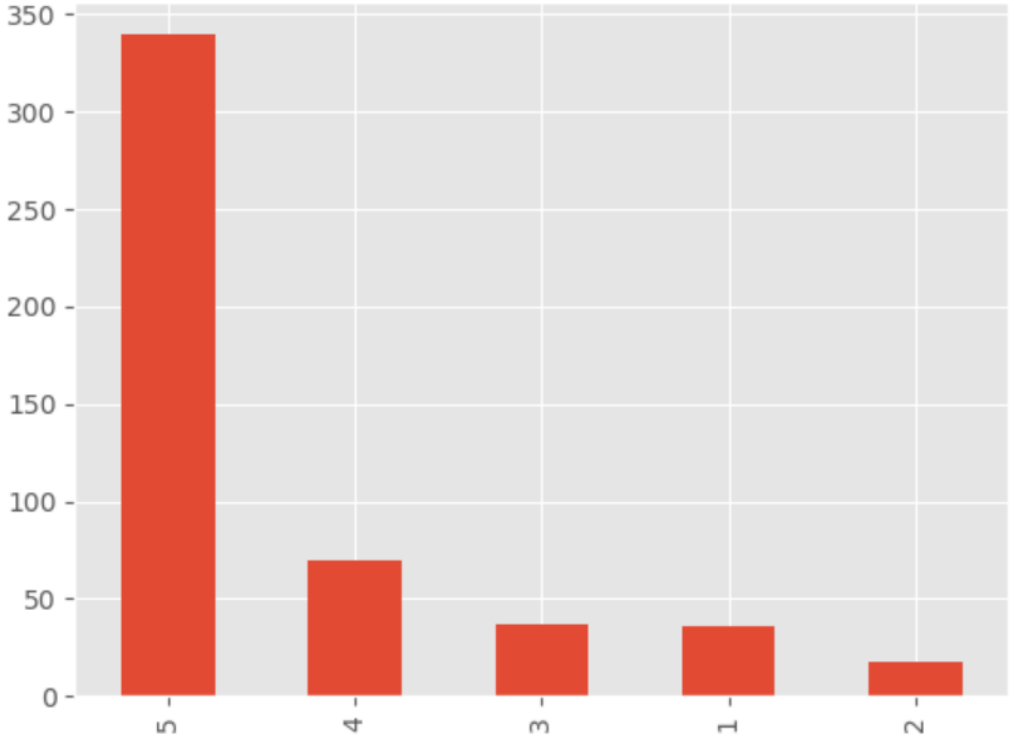

Fig 2.4 Bar graph for score attribute in food review dataset

```
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()
```

Fig 2.5 Bar graph for score attribute sorting values

Scatter plot of numerical data

```
sns.scatterplot(data = df, x= 'Score', y = 'Time')
```

```
<Axes: xlabel='Score', ylabel='Time'>
```

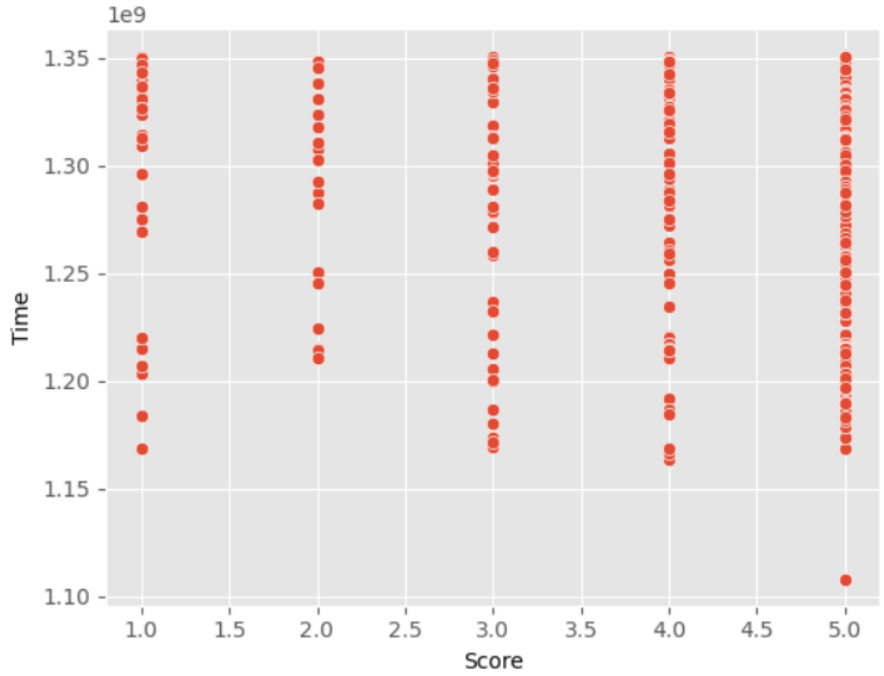

Fig 2.6 Scatter plot of Score with respect to time

```
sns.scatterplot(data = df, x= 'Score', y = 'Time', hue = 'Id')
```

```
<Axes: xlabel='Score', ylabel='Time'>
```

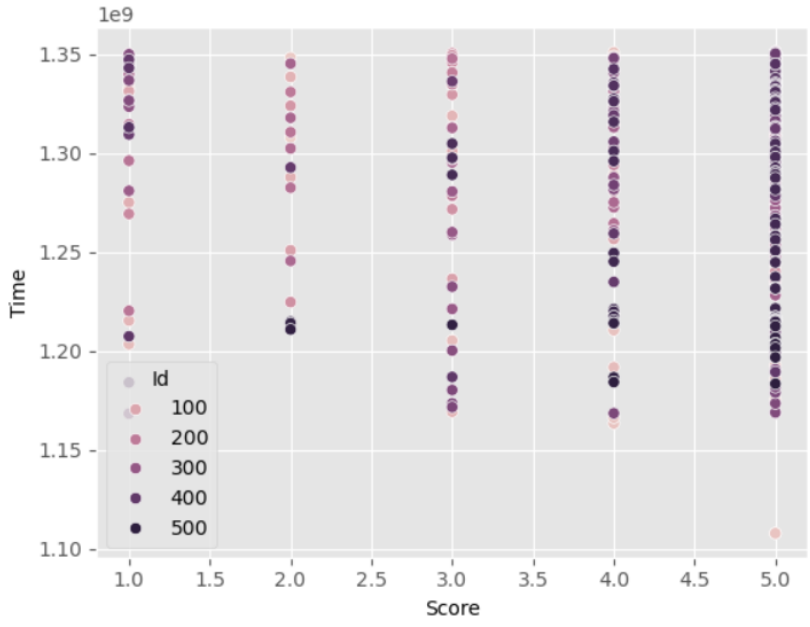

Fig 2.7 Scatter plot of Score and Time wrt Id

```
sns.regplot(data = df, x= 'Score', y = 'Time')
```

```
<Axes: xlabel='Score', ylabel='Time'>
```

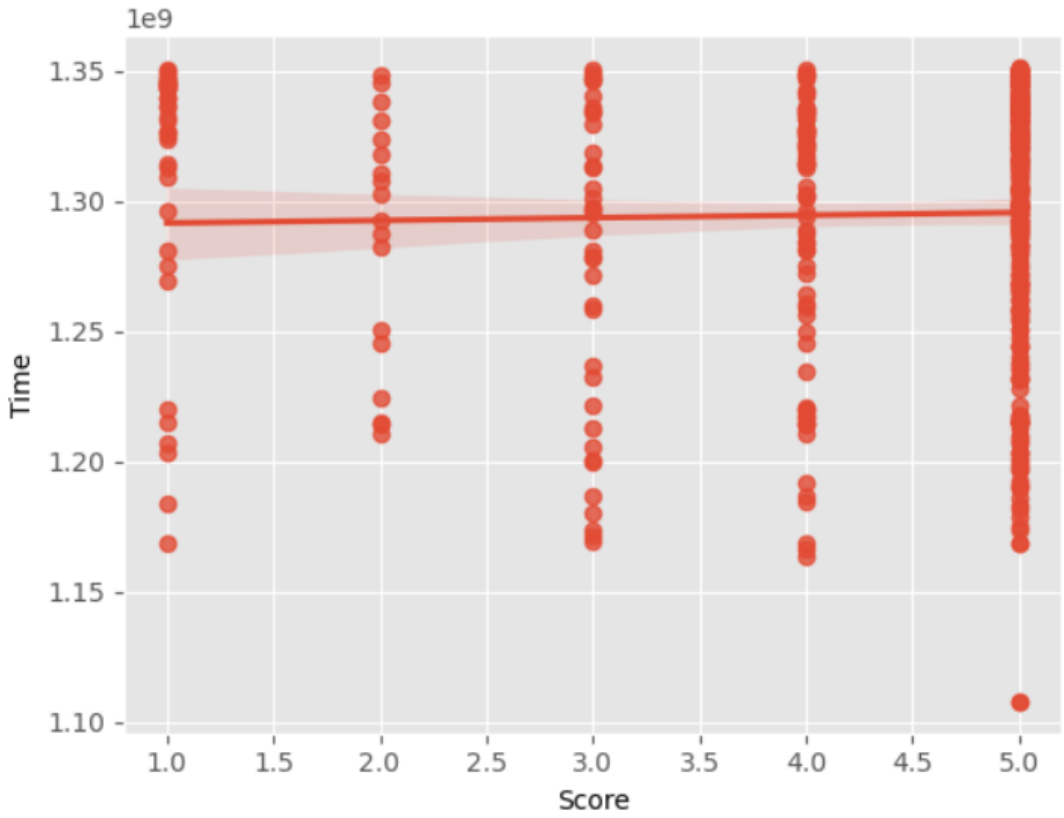

Fig 2.8 Regression plot of Score with respect to time

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Id                      500 non-null    int64 
 1   ProductId               500 non-null    object
 2   UserId                  500 non-null    object
 3   ProfileName             500 non-null    object
 4   HelpfulnessNumerator    500 non-null    int64 
 5   HelpfulnessDenominator  500 non-null    int64 
 6   Score                   500 non-null    int64 
 7   Time                    500 non-null    int64 
 8   Summary                 500 non-null    object
 9   Text                    500 non-null    object
dtypes: int64(5), object(5)
memory usage: 39.2+ KB
```

Separating Categorical variables

```
df_cat = df.select_dtypes(include= ['object'])
df_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ProductId    500 non-null    object
 1   UserId       500 non-null    object
 2   ProfileName  500 non-null    object
 3   Summary      500 non-null    object
 4   Text         500 non-null    object
dtypes: object(5)
memory usage: 19.7+ KB
```

Sorting categories based on the number of diff categories inside that category

```
df_cat.nunique().sort_values()
```

```
ProductId      136
ProfileName    489
UserId         490
Summary        490
Text           499
dtype: int64
```

Basic NLTK

```
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...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
df['Text']
```

```
0      I have bought several of the Vitality canned d...
1      Product arrived labeled as Jumbo Salted Peanut...
2      This is a confection that has been around a fe...
3      If you are looking for the secret ingredient i...
4      Great taffy at a great price.  There was a wid...
                           ...
495    i rarely eat chips but i saw these and tried t...
496    This is easily the best potato chip that I hav...
497    Kettle Chips Spicy Thai potato chips have the ...
498    Okay, I should not eat potato chips, nor shoul...
499    I don't write very many reviews but I have to ...
Name: Text, Length: 500, dtype: object
```

```
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 look
s more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better t
han  most.'
```

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

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\psriz\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
['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
  ./.)
```

Step 1. VADER Sentiment Scoring

- We will use NLTK's Sentiment Intensity Analyzer to get the neg/neu/pos scores of the text.
- This uses a "bag of words" approach:
- Stop words are removed. Stop words like and, the which do not have positive or negative feeling. They are just for the structure of the sentence
- each word is scored and combined to a total score.
- Score can be positive, negative, neutral

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a sentiment analysis tool specifically designed to analyze social media texts. It provides a way to automatically derive sentiment from textual data using a combination of a pre-defined sentiment lexicon and rules that capture syntactic and grammatical nuances.

Here's how VADER works in detail:

1. Sentiment Lexicon

- VADER's sentiment lexicon is a list of words and their associated sentiment scores. These scores range from -4 (most negative) to +4 (most positive). The lexicon includes:
- Common words, including slang and abbreviations.
- Emojis and emoticons.
- Sentiment-laden phrases and idioms.

2. Rule-Based Components

- In addition to the lexicon, VADER uses a set of grammatical and syntactical rules to adjust sentiment scores based on the context in which words appear. These rules account for:
- Punctuation: Exclamation marks and question marks can amplify sentiment intensity. For instance, "good!!!" is more positive than "good".
- Capitalization: Uppercase words can also increase sentiment intensity, e.g., "GOOD" is more intense than "good".
- Degree Modifiers: Words like "very", "extremely", and "somewhat" modify the intensity of the sentiment. For example, "very good" is more positive than "good".

- Conjunctions: The presence of conjunctions like "but" can shift sentiment. For example, "I love this, but I hate that" has mixed sentiment.
- Negation: Words like "not" can flip the sentiment of the word that follows. For instance, "not good" is negative despite "good" being positive.

3. Compound Score

- VADER calculates a compound score by summing up the valence scores of each word, adjusted by the aforementioned rules, and normalizing it to a value between -1 (most negative) and +1 (most positive). This compound score reflects the overall sentiment of the text.

Example

- Consider the sentence: "I absolutely love this product! It's AMAZING :)"

Lexicon Scores:

- "absolutely" (degree modifier)
- "love" (+3)
- "AMAZING" (+4, amplified by capitalization)
- ":)" (+2)

Rule Adjustments:

- "absolutely" modifies the intensity of "love"
- Exclamation mark adds to the intensity
- Capitalization amplifies "AMAZING"
- Emoji contributes positive sentiment

The overall sentiment score will be calculated by summing these values, applying the rules, and then normalizing the score to get the compound sentiment.

- VADER is often used in applications that require real-time sentiment analysis, such as:
- Social media monitoring
- Customer feedback analysis
- Market research
- Implementation

VADER is implemented in Python and is part of the NLTK (Natural Language Toolkit) library, making it easily accessible for developers and data scientists. Here's a basic example of how to use it:

python

Copy code

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

sentence = "I absolutely love this product! It's AMAZING :)"

scores = analyzer.polarity_scores(sentence)

print(scores)
```

This script will output a dictionary with the following sentiment scores:

- neg: Negative sentiment proportion
- neu: Neutral sentiment proportion
- pos: Positive sentiment proportion
- compound: Overall sentiment score normalized between -1 and +1

Conclusion

VADER is a powerful tool for sentiment analysis, particularly effective for short, informal text typical of social media. Its combination of a robust lexicon and rule-based adjustments enables it to capture sentiment nuances accurately.

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

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\psriz\AppData\Roaming\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()
```

Calculating the score for the sentiments in terms of positive, negative, neutral

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

```
{'neg': 0.0, 'neu': 0.213, 'pos': 0.787, 'compound': 0.5719}
```

```
sia.polarity_scores('I am sad')
```

```
{'neg': 0.756, 'neu': 0.244, 'pos': 0.0, 'compound': -0.4767}
```

```
sia.polarity_scores('I am batman and i am here to make justice')
```

```
{'neg': 0.0, 'neu': 0.673, 'pos': 0.327, 'compound': 0.5267}
```

```
example = df['Text'][50]
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 the polarity score on the entire dataset

```
# 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)
```

```
  0%|          | 0/500 [00:00<?, ?it/s]
```

```
res
```

```
{1: {'neg': 0.0, 'neu': 0.695, 'pos': 0.305, 'compound': 0.9441},
 2: {'neg': 0.138, 'neu': 0.862, 'pos': 0.0, 'compound': -0.5664},
 3: {'neg': 0.091, 'neu': 0.754, 'pos': 0.155, 'compound': 0.8265},
 4: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
 5: {'neg': 0.0, 'neu': 0.552, 'pos': 0.448, 'compound': 0.9468},
 6: {'neg': 0.029, 'neu': 0.809, 'pos': 0.163, 'compound': 0.883},
 7: {'neg': 0.034, 'neu': 0.693, 'pos': 0.273, 'compound': 0.9346},
 8: {'neg': 0.0, 'neu': 0.52, 'pos': 0.48, 'compound': 0.9487},
 9: {'neg': 0.0, 'neu': 0.851, 'pos': 0.149, 'compound': 0.6369},
 10: {'neg': 0.0, 'neu': 0.705, 'pos': 0.295, 'compound': 0.8313},
 11: {'neg': 0.017, 'neu': 0.846, 'pos': 0.137, 'compound': 0.9746},
 12: {'neg': 0.113, 'neu': 0.887, 'pos': 0.0, 'compound': -0.7579},
 13: {'neg': 0.031, 'neu': 0.923, 'pos': 0.046, 'compound': 0.296},
 14: {'neg': 0.0, 'neu': 0.355, 'pos': 0.645, 'compound': 0.9466},
 15: {'neg': 0.104, 'neu': 0.632, 'pos': 0.264, 'compound': 0.6486},
 16: {'neg': 0.0, 'neu': 0.861, 'pos': 0.139, 'compound': 0.5719},
 17: {'neg': 0.097, 'neu': 0.694, 'pos': 0.209, 'compound': 0.7481},
 18: {'neg': 0.0, 'neu': 0.61, 'pos': 0.39, 'compound': 0.8883},
 19: {'neg': 0.012, 'neu': 0.885, 'pos': 0.103, 'compound': 0.8957},
```

Converting the vaders score to a pandas data frame and merging it with the actual data frame

Old data set before merging

```
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...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

New data set after merging

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

```
vaders
```

	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
0	1	0.000	0.695	0.305	0.944	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400
1	2	0.138	0.862	0.000	-0.566	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000
2	3	0.091	0.754	0.155	0.827	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600

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

	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Sum
0	1	0.000	0.695	0.305	0.944	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	G Dog
1	2	0.138	0.862	0.000	-0.566	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	I Adve
2	3	0.091	0.754	0.155	0.827	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"D say
3	4	0.000	1.000	0.000	0.000	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	C Me

Plot VADER results

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

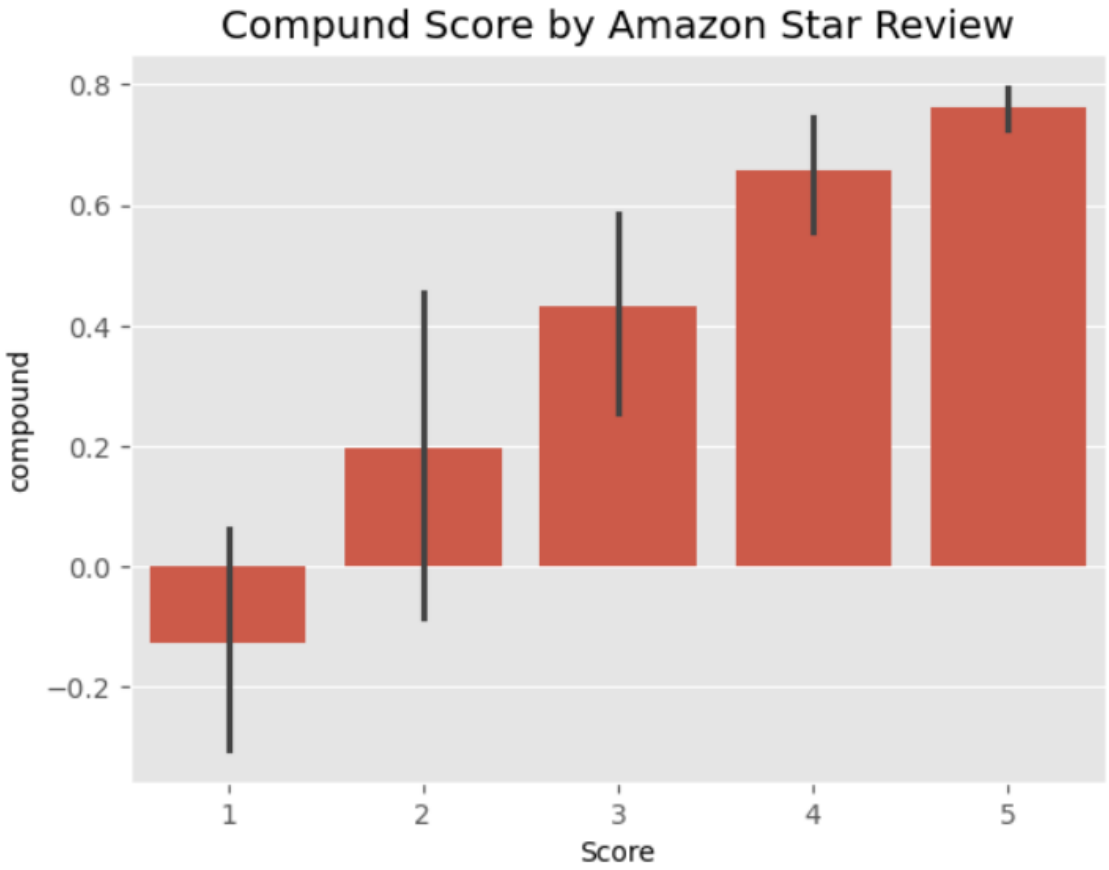

Fig 2.9 Plotting VADER results

```
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()
```

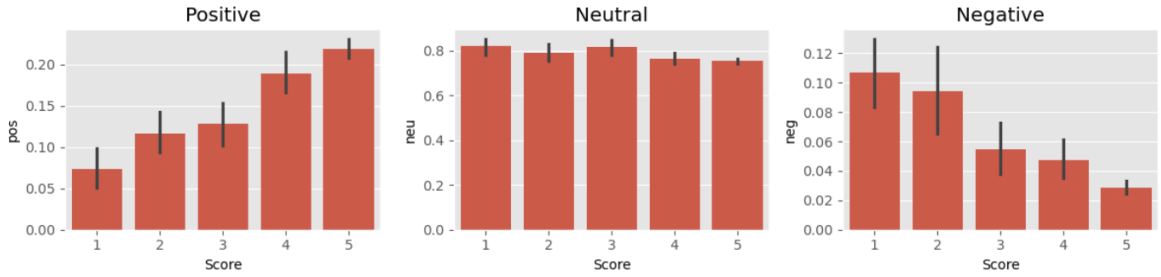

Fig 2.10 Plotting VADERS results based on Pos, Neu, Neg

Step 2. 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': 0.020687481, 'roberta_pos': 0.0029573753}
```

```
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}')
```

```
  0%|          | 0/500 [00:00<?, ?it/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')
```

Compairing columns of original data, vaders and roberto results after merging it with original data frame

```
df.columns
```

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

```
vaders.columns
```

```
Index(['Id', 'neg', 'neu', 'pos', 'compound', 'ProductId', 'UserId',
       'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator',
       'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

```
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')
```

Step 3. 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()
```

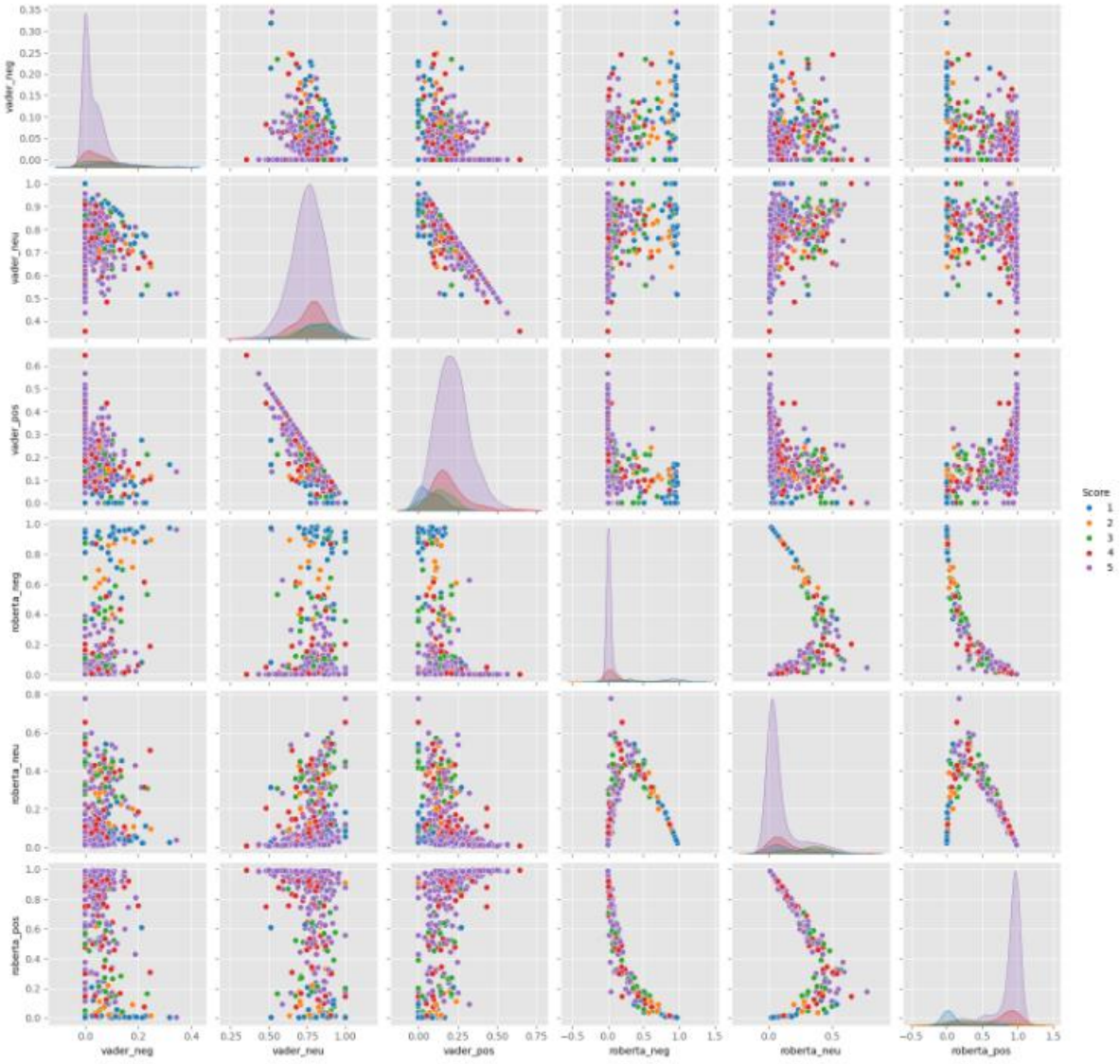

Fig 2.11 Plotting VADERS and ROBERTA results for comparison

Step 4. 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...'
```

Negative sentiment 5-Star view

```
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'
```

Extra: 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-base-uncased-finetuned-sst-2-english and revision af0f99b (https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
```

```
Downloading config.json:   0%|          | 0.00/629 [00:00<?, ?B/s]
```

```
E:\ANACONDA\lib\site-packages\huggingface_hub\file_download.py:133: UserWarning: `huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\psriz\.cache\huggingface\hub. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
```

```
Downloading model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

```
Downloading tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
```

```
Downloading vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

```
sent_pipeline('In the journey of life, obstacles are not roadblocks, but rather stepping stones that refine our essence')
```

```
[{'label': 'POSITIVE', 'score': 0.9989205598831177}]
```

```
sent_pipeline('I am batman and i am here for justice')
```

```
[{'label': 'POSITIVE', 'score': 0.9977778792381287}]
```

```
sent_pipeline('I lost someone i loved')
```

```
[{'label': 'NEGATIVE', 'score': 0.9980044960975647}]
```

```
sent_pipeline('I am depressed')
```

```
[{'label': 'NEGATIVE', 'score': 0.9997759461402893}]
```

CHAPTER 3

CONCLUSION & FUTURE ENHANCEMENTS

3.1 CONCLUSION

The exploratory data analysis (EDA) and sentiment analysis of food reviews have provided several insights into customer preferences, trends, and overall sentiment towards various food products and services. Through systematic data examination and advanced sentiment analysis techniques, we have drawn the following key conclusions:

- **Customer Preferences:** EDA revealed significant patterns in customer preferences. For instance, certain cuisines or dishes were consistently rated higher, indicating a strong preference for these items. Conversely, items with lower ratings provided insight into areas needing improvement.
- **Sentiment Trends:** Sentiment analysis uncovered prevailing sentiments in the reviews. Positive sentiments were associated with specific attributes such as taste, quality of ingredients, and customer service, while negative sentiments often related to issues such as delivery time, food temperature, or packaging.
- **Correlation Between Ratings and Sentiment:** A strong correlation was observed between numerical ratings and sentiment scores, validating the sentiment analysis model's effectiveness. High ratings aligned with positive sentiments and vice versa.
- **Key Influencing Factors:** Through EDA, we identified key factors influencing customer reviews, including price, portion size, and delivery experience. These factors can be targeted for enhancement to improve overall customer satisfaction.
- **Insights for Business Improvement:** The findings provide actionable insights for businesses to enhance their offerings. By focusing on the aspects most valued by customers and addressing common pain points, businesses can improve their products and services, thereby boosting customer satisfaction and loyalty.

By implementing these future enhancements, businesses can derive even greater value from sentiment analysis and EDA, leading to improved customer satisfaction, loyalty, and overall business performance.

3.2 FUTURE ENHANCEMENTS

While the current analysis has yielded valuable insights, several areas can be further explored and enhanced to improve the robustness and applicability of the findings. Future enhancements could include:

- **Advanced Sentiment Analysis Models:** Implementing more sophisticated sentiment analysis models, such as those based on deep learning (e.g., BERT or GPT-based models), could improve the accuracy of sentiment detection, especially for nuanced or context-specific sentiments.
- **Aspect-Based Sentiment Analysis:** Focusing on aspect-based sentiment analysis can provide more granular insights by identifying sentiment towards specific aspects (e.g., taste, service, ambiance) within a review, offering a deeper understanding of customer opinions.
- **Integration of Additional Data Sources:** Including data from social media, blogs, and other customer feedback platforms can provide a more comprehensive view of customer sentiments and trends. This multi-source approach can enhance the reliability of the analysis.
- **Predictive Analytics:** Incorporating predictive analytics to forecast future trends and customer preferences based on historical data can help businesses proactively address potential issues and cater to evolving customer needs.
- **Real-Time Sentiment Analysis:** Developing a real-time sentiment analysis system can provide businesses with immediate feedback on customer opinions, enabling quicker response times and more agile decision-making.
- **Enhanced Visualization Techniques:** Utilizing advanced visualization techniques, such as interactive dashboards, can make the insights more accessible and actionable for business stakeholders, facilitating better data-driven decisions.
- **Customer Segmentation:** Implementing customer segmentation techniques can help identify distinct customer groups with unique preferences and sentiments, allowing for more targeted marketing and personalized service offerings.
- **Automated Feedback Systems:** Developing automated systems to collect and analyze customer feedback in real-time can streamline the review process, making it more efficient and less labor-intensive.

REFERENCE

[1] https://www.stratascratch.com/blog/data-mining-projects-with-source-codes/

[2] https://www.kaggle.com/

[3] https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

[4] https://www.kaggle.com/robikscube

[5] https://www.geeksforgeeks.org/data-mining/

[6] https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

[7] https://www.python.org/

[8] https://pandas.pydata.org/

[9] https://numpy.org/

[10] https://matplotlib.org/

[11] https://seaborn.pydata.org/

[12] https://www.nltk.org/

[13] https://github.com/cjhutto/vaderSentiment

[14] https://huggingface.co/docs/transformers/en/model_doc/roberta

[15] https://huggingface.co/blog/sentiment-analysis-python