Sentiment Analysis (Opinion Mining) with Python — NLP Tutorial

Introduction

A “sentiment” is a generally binary opposition in opinions and expresses the feelings in the form of emotions, attitudes, opinions, and so on. It can express many opinions. For instance, “like,” or “dislike,” “good,” or “bad,” “for,” or “against,” along with others.

By using **machine learning** methods and **natural language processing**, we can extract the personal information of a document and attempt to classify it according to its polarity, such as positive, neutral, or negative, making sentiment analysis instrumental in determining the overall opinion of a defined objective, for instance, a selling item or predicting stock markets for a given company.

Sentiment analysis is challenging and far from being solved since most languages are highly complex (objectivity, subjectivity, negation, vocabulary, grammar, and others). However, that is what makes it exciting to working on [1].

Nowadays, **sentiment analysis** is prevalent in many applications to analyze different circumstances, such as:

- How Twitter users’ attitudes may have changed about the elected President since the US election?
- Is this client’s email satisfactory or dissatisfactory?
- Is this product review positive or negative?
- How are people responding to particular news?
- A consumer uses these to research products and services before a purchase.
- Production companies can use public opinion to define the acceptance of their products and the public demand.
- Moviegoers decide whether to watch a movie or not after going through other people’s reviews.
- The prediction of election outcomes based on public opinion.
- To measure social media performance.
- Along with others.

What is Sentiment Analysis?

Fundamentally, we can **define sentiment analysis** as the computational study of opinions, thoughts, evaluations, evaluations, interests, views, emotions, subjectivity, along with others, that are expressed in a text.

It involves classifying opinions found in text into categories like “positive” or “negative” or “neutral.” Sentiment analysis is also known by different names, such as opinion mining, appraisal extraction, subjectivity analysis, and others.

For example:

"The story of the movie was boring and a waste of time."

The following terms can be extracted from the sentence above to perform sentiment analysis:

- **Opinion Owner:** Audience
- **Object:** Movie
- **Feature:** Story
- **Opinion:** Boring and a waste of time.
- **Polarity:** Negative

Other examples:

- "I like my smartwatch but would not recommend it to any of my friends."
- "I do not like love. It is a waste of time."
- "Titanic is the best movie of all time."
- "I am not too fond of sharp, bright-colored clothes."

Types of Sentiment Analysis

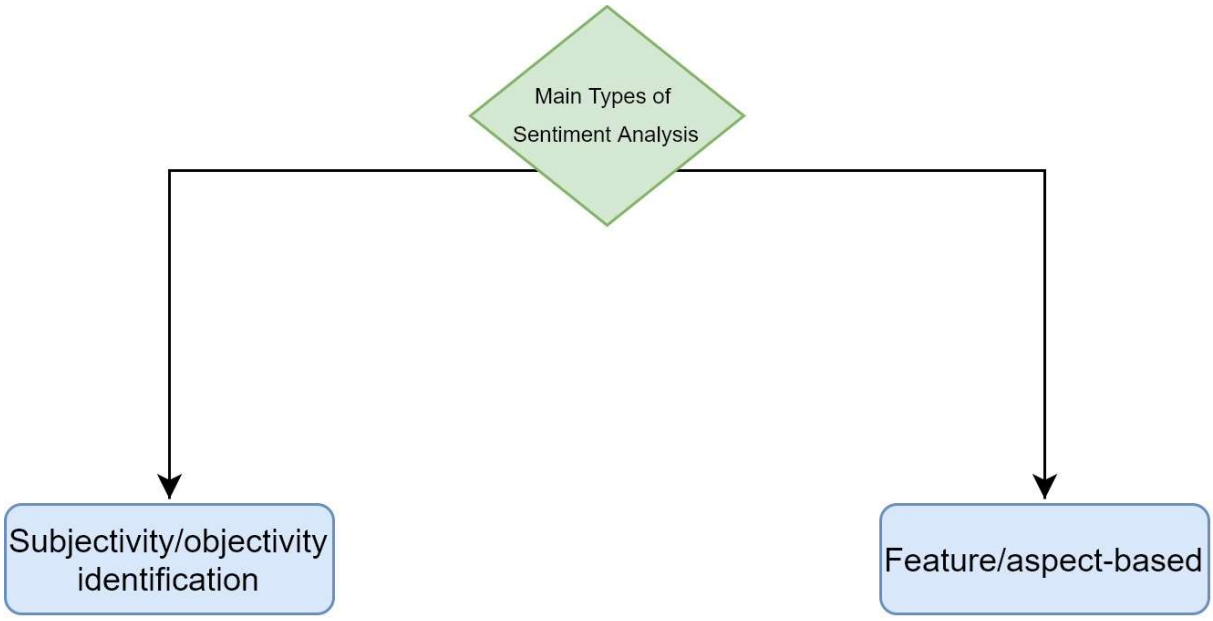

Figure 1: Main types of sentiment analysis.

There are several types of Sentiment Analysis, such as Aspect Based Sentiment Analysis, Grading sentiment analysis (positive, negative, neutral), Multilingual sentiment analysis, detection of emotions, along with others [2].

For this tutorial, we are going to focus on the **most relevant sentiment analysis types** [2]:

- **Subjectivity/objectivity identification.**
- **Feature/aspect-based.**

Subjectivity/objectivity identification

In subjectivity or objectivity identification, a given text or sentence is classified into two different classes:

- **Subjectivity**: It expresses an opinion that describes people's feelings towards a specific topic.
- e.g., The taste of this mango is good.
- **Objective**: It expresses the fact. e.g., This mango is yellow.

The subjective sentence expresses personal feelings, views, or beliefs. Sentiment analysis works great on a text with a personal connection than on text with only an objective connection.

Feature/aspect-based

Feature or aspect-based sentiment analysis analyses different features, attributes, or aspects of a product. Its main goal is to recognize the aspect of a given target and the sentiment shown towards each aspect.

For instance:

*"Today, I purchased a Samsung phone, and my boyfriend purchased an iPhone. We called each other in the evening. The **voice** of my phone was not clear, but the **camera** was good. My girlfriend said the **sound** of her phone was very clear. So, I decided to buy a similar phone because its **voice** quality is very good. So, I bought an iPhone and returned the Samsung phone to the seller."*

Applying aspect extraction to the sentences above:

- Voice.
- Camera.
- Sound.

Sentiment Analysis Architecture

The following diagram makes an effort to showcase the typical sentiment analysis architecture, depicting the phases of applying sentiment analysis to movie data.

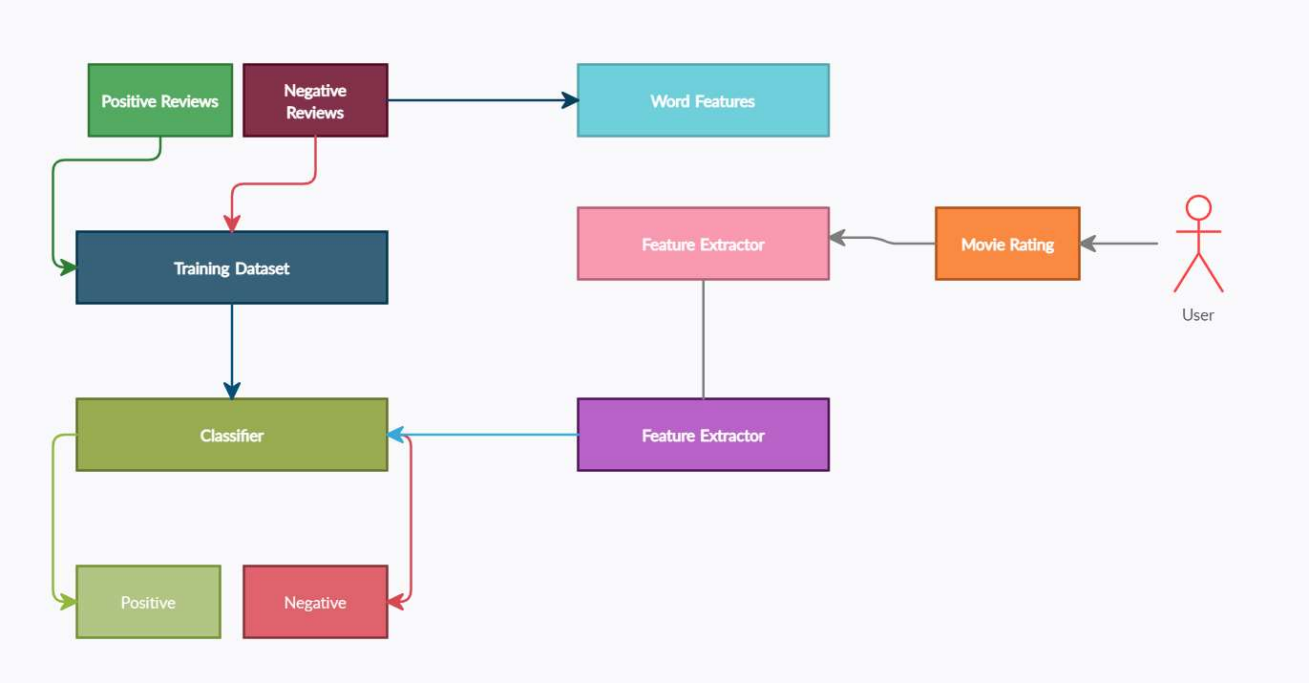

Figure 4: Sentiment analysis architecture.

The control flow of sentiment analysis:

Figure 5: Typical control flow for sentiment analysis.

There are several steps involved in sentiment analysis:

- Data collection.
- Data analysis.
- Indexing.
- Delivery.

Data Collection

- Public sentiments from consumers expressed on public forums are collected like Twitter, Facebook, and so on.

- Opinions or feelings/behaviours are expressed differently, the context of writing, usage of slang, and short forms.

Data Analysis

The data analysis process has the following steps:

1. Text Preparation

- Data is extracted and filtered before doing some analysis.
- Non-textual content and the other content is identified and eliminated if found irrelevant.

2. Sentiment Detection

- Each sentence and word is determined very clearly for subjectivity.
- Sentences with subjective information are retained, and the ones that convey objective information are discarded.

Indexing

- Sentiments can be broadly classified into two groups positive and negative.
- Each subjective sentence is classified into the likes and dislikes of a person.

Delivery

- It is the last stage involved in the process.
- The result is converting unstructured data into meaningful information.
- They are displayed as graphs for better visualization.

Polarity

In sentiment analysis, we use polarity to identify sentiment orientation like positive, negative, or neutral in a written sentence. Fundamentally, it is an emotion expressed in a sentence.

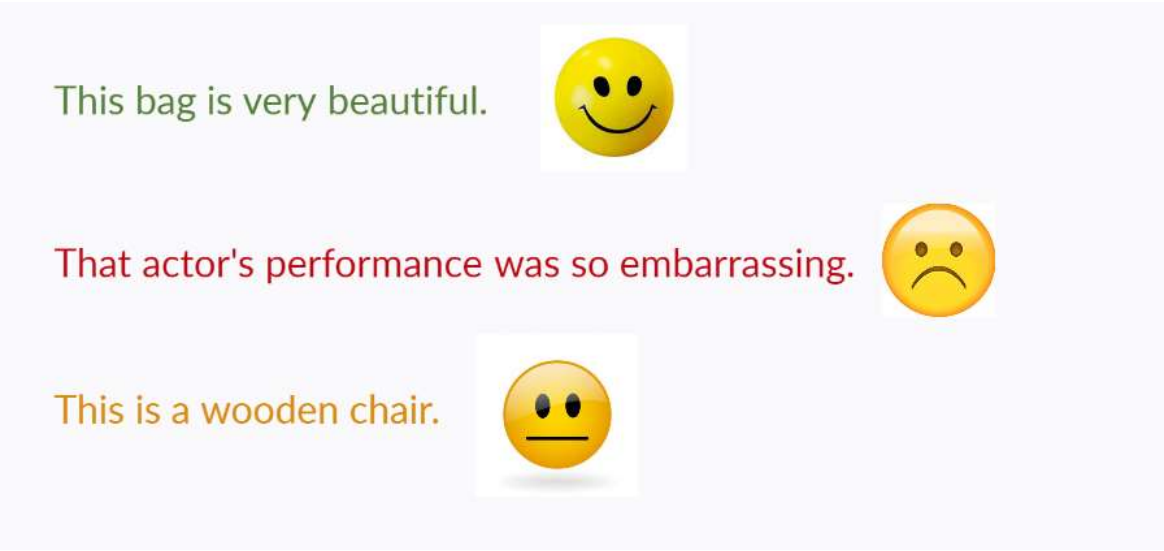

Figure 6: An example of polarity.

Based on the rating, the “Rating Polarity” can be calculated as below:

```
df['Rating_Polarity'] = df['Rating'].apply(lambda x: 'Positive' if x > 3
else('Neutral' if x == 3  else 'Negative'))
```

Methods for Sentiment Analysis

Essentially, sentiment analysis finds the emotional polarity in different texts, such as positive, negative, or neutral. There are two different methods to perform sentiment analysis:

- Lexicon-based method
- Machine Learning method

Lexicon-based method

Lexicon-based sentiment analysis calculates the sentiment from the semantic orientation of words or phrases present in a text.

The **lexicon-based method** has the following ways to handle sentiment analysis:

- Dictionary
- Corpus

Dictionary

It creates a dictionary of positive and negative words and assigns positive and negative sentiment values to each of the words. Its dictionary of positive and negative values for each of the words can be defined as:

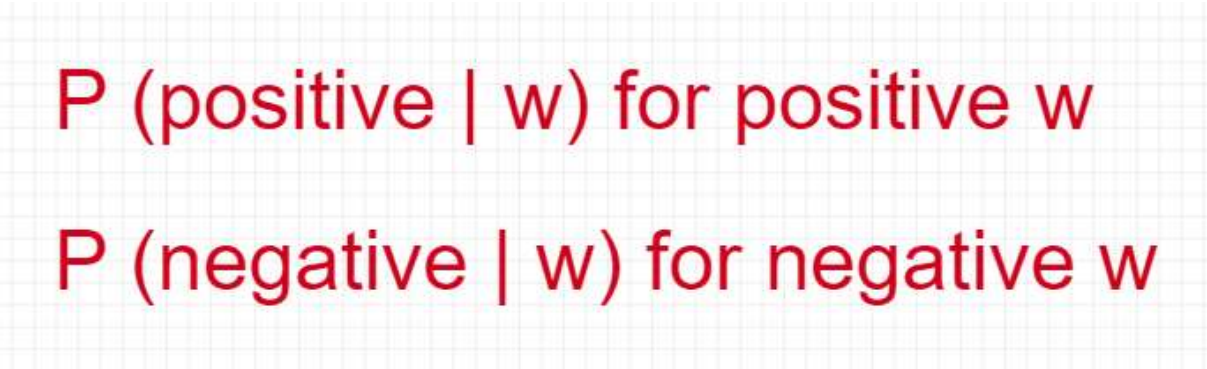

Figure 7: Dictionary for positive and negative words.

Thus, it creates a dictionary-like schema such as:

Key	Value
nice	+2
good	+1
worst	-3
best	+5
terrible	-1.5

Figure 8: Dictionary table for different words.

Based on the defined dictionary, the algorithm's job is to look up text to find all well-known words and accurately consolidate their specific results. Sometimes it applies grammatical rules like negation or sentiment modifier.

For instance, applying sentiment analysis to the following sentence by using a Lexicon-based method:

"I do not love you because you are a terrible guy, but you like me."

Consequently, it finds the following words based on a Lexicon-based dictionary:

- **love**: +5
- **like**: +2
- **terrible**: -1.5

Overall sentiment = +5 + 2 + (-1.5) = +5.5

Accordingly, this sentiment expresses a positive sentiment.
Dictionary would process in the following ways:

- Flat
- With Semantics

Machine Learning method

The machine learning method is superior to the lexicon-based method, yet it requires annotated data sets. It requires a training dataset that manually recognizes the sentiments, and it is definite to data and domain-oriented values, so it should be prudent at the time of prediction because the algorithm can be easily biased.

If the algorithm has been trained with the data of clothing items and is used to predict food and travel-related sentiments, it will predict poorly. Therefore, sentiment analysis is highly domain-oriented and centric because the model developed for one domain like a movie or restaurant will not work for the other domains like travel, news, education, and others.

Baseline Machine Learning Algorithms for Sentiment Analysis

The following **machine learning algorithms** are used for sentiment analysis:

- Feature extraction.
- Tokenization.
- SVM.
- Naive Bayes.
- MaxEnt.

Feature Extraction

The feature extraction method takes text as input and produces the extracted features in any form like lexico-syntactic or stylistic, syntactic, and discourse-based. Primarily, it identifies those product aspects which are being commented on by customers.

Tokenization

Tokenization is a process of splitting up a large body of text into smaller lines or words. It helps in interpreting the meaning of the text by analyzing the sequence of the words.

For example:

"This movie is really good."

After applying tokenization:

[This, movie, is, really, good]

Note: MaxEnt and SVM perform better than the Naive Bayes algorithm sentiment analysis use-cases.

Challenges and Problems in Sentiment Analysis

Sentiment analysis is fascinating for real-world scenarios. However, it faces many problems and challenges during its implementation.

Below are the challenges in the sentiment analysis:

- It is tough if compared with topical classification with a bag of words features performed well.
- In many cases, words or phrases express different meanings in different contexts and domains.

Other challenges of sentiment analysis:

- The main challenge in Sentiment analysis is the complexity of the language.
- Negation has the primary influence on the contextual polarity of opinion words and texts. Negation phrases such as never, none, nothing, neither, and others can reverse the opinion-words' polarities.
- Puzzled sentences and complex linguistics. e.g., "Admission to the hospital was complicated, but the staff was very nice even though they were swamped." Therefore, here → (negative → positive → implicitly negative)

These are some problems in sentiment analysis:

- It is challenging to answer a question — which highlights what features to use because it can be words, phrases, or sentences.
- How to interpret features? It can be a bag of words, annotated lexicons, syntactic patterns, or a paragraph structure.

Data Pre-processing for Sentiment Analysis

Before applying any machine learning or deep learning library for sentiment analysis, it is crucial to do text cleaning and/or pre-processing. It is essential to reduce the noise in human-text to improve accuracy. Data is processed with the help of a **natural language processing** pipeline.

These steps are applied during data pre-processing:

- Normalizing words.
- Removing stop words.
- Tokenizing sentences.
- Vectorizing text.

Use-Case: Sentiment Analysis for Fashion, Python Implementation

Nowadays, online shopping is trendy and famous for different products like electronics, clothes, food items, and others. For instance, e-commerce sells products and provides an option to rate and write comments about consumers' products, which is a handy and important way to identify a product's quality. Based on them, other consumers can decide whether to purchase a product or not. It is also beneficial to sellers and manufacturers to know their products' sentiments to make their products better.

Code implementation in deep learning:

Import all required packages:

```
import pandas as pd
import numpy as np
import seaborn as sns
import re
import string
from string import punctuation
import nltk
from nltk.corpus import stopwordsnltk.download("stopwords")
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

Read data:

```
df = pd.read_csv('clothing_review.csv')df.head()
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses

Drop unnecessary columns:

```
df = df.drop(['Title', 'Positive Feedback Count', 'Unnamed: 0', ], axis=1)df.dropna(inplace=True)
```

Calculate Rating Polarity based on the rating of dresses by old consumers:

Apply the following rules:

- If the existing rating > 3 then polarity_rating = "**Positive**"
- If the existing rating == 3 then polarity_rating = "**Neutral**"
- If the existing rating < 3 then polarity_rating = "**Negative**"

Code implementation based on the above rules to calculate Polarity Rating:

```
df['Polarity_Rating'] = df['Rating'].apply(lambda x: 'Positive' if x > 3 else('Neutral' if x == 3  else 'Negative'))
```

	Clothing ID	Age	Review Text	Rating	Recommended IND	Division Name	Department Name	Class Name	Polarity_Rating
0	767	33	Absolutely wonderful - silky and sexy and comf...	4	1	Initmates	Intimate	Intimates	Positive
1	1080	34	Love this dress! it's sooo pretty. i happene...	5	1	General	Dresses	Dresses	Positive
2	1077	60	I had such high hopes for this dress and reall...	3	0	General	Dresses	Dresses	Neutral
3	1049	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	General Petite	Bottoms	Pants	Positive
4	847	47	This shirt is very flattering to all due to th...	5	1	General	Tops	Blouses	Positive

Figure 10: Polarity_Rating based on the rating.

Visualization

Plotting the rating count visualization:

```
sns.set_style('whitegrid')sns.countplot(x='Rating',data=df, palette='YlGnBu_r')
```

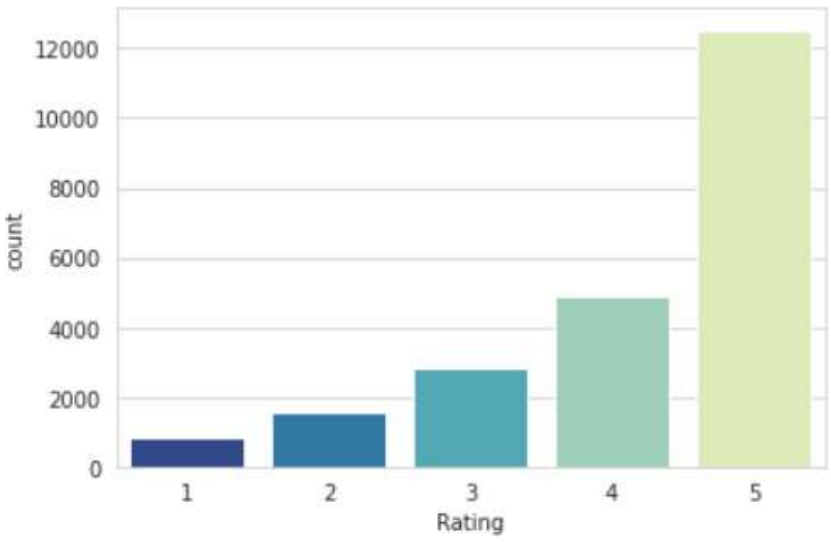

Figure 11: Rating count visualization.

Plot the Polarity rating count graph:

```
sns.set_style('whitegrid')sns.countplot(x='Polarity_Rating',data=df,
palette='summer')
```

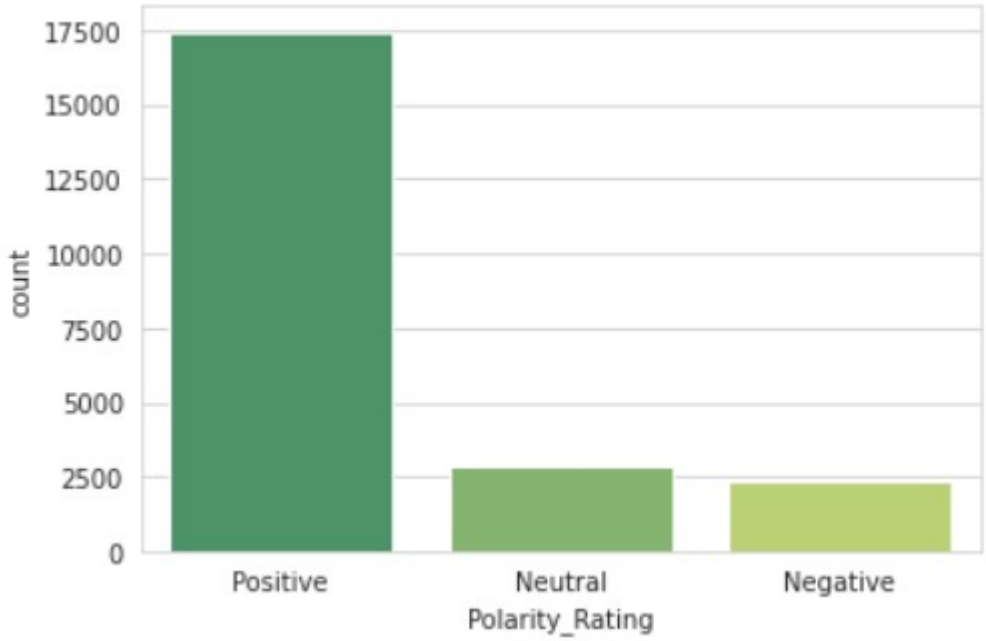

Figure 12: Polarity rating count.

Data Pre-processing

```
df_Positive = df[df['Polarity_Rating'] == 'Positive'][0:8000]df_Neutral =
df[df['Polarity_Rating'] == 'Neutral']df_Negative =
df[df['Polarity_Rating'] == 'Negative']
```

Sample negative and neutral dataset and create a final dataset:

```
df_Neutral_over = df_Neutral.sample(8000, replace=True)df_Negative_over =
df_Negative.sample(8000, replace=True)df = pd.concat([df_Positive,
df_Neutral_over, df_Negative_over], axis=0)
```

Text Preprocessing:

```
def get_text_processing(text):
    stpword = stopwords.words('english')
    no_punctuation = [char for char in text if char not in
string.punctuation]
    no_punctuation = ''.join(no_punctuation)
    return ' '.join([word for word in no_punctuation.split() if
word.lower() not in stpword])
```

Apply the method “get_text_processing” into column “Review Text”:

```
df['review'] = df['Review Text'].apply(get_text_processing)df.head()
```

	Clothing ID	Age	Review Text	Rating	Recommended IND	Division Name	Department Name	Class Name	Polarity_Rating	review
0	767	33	Absolutely wonderful - silky and sexy and comf...	4	1	Initmates	Intimate	Intimates	Positive	Absolutely wonderful silky sexy comfortable
1	1080	34	Love this dress! it's sooo pretty. i happene...	5	1	General	Dresses	Dresses	Positive	Love dress sooo pretty happened find store im ...
3	1049	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	General Petite	Bottoms	Pants	Positive	love love love jumpsuit fun flirty fabulous ev...
4	847	47	This shirt is very flattering to all due to th...	5	1	General	Tops	Blouses	Positive	shirt flattering due adjustable front tie perf...
6	858	39	I aded this in my basket at hte last mintue to...	5	1	General Petite	Tops	Knits	Positive	aded basket hte last mintue see would look lik...

Figure 13: Review column after applying text processing.

It filters out the string punctuations from the sentences.

Visualize Text Review with Polarity_Review column:

```
df = df[['review', 'Polarity_Rating']]df.head()
```

	review	Polarity_Rating
0	Absolutely wonderful silky sexy comfortable	Positive
1	Love dress sooo pretty happened find store im ...	Positive
3	love love love jumpsuit fun flirty fabulous ev...	Positive
4	shirt flattering due adjustable front tie perf...	Positive
6	aded basket hte last mintue see would look lik...	Positive

Figure 14: Review and Polarity_Rating table.

Apply One hot encoding on negative, neural, and positive:

```
one_hot = pd.get_dummies(df["Polarity_Rating"])df.drop(["Polarity_Rating"],
axis=1, inplace=True)df = pd.concat([df, one_hot], axis=1)df.head()
```

	review	Negative	Neutral	Positive
0	Absolutely wonderful silky sexy comfortable	0	0	1
1	Love dress sooo pretty happened find store im ...	0	0	1
3	love love love jumpsuit fun flirty fabulous ev...	0	0	1
4	shirt flattering due adjustable front tie perf...	0	0	1
6	aded basket hte last mintue see would look lik...	0	0	1

Figure 15: One hot encoding.

Apply train test split:

```
X = df["review"].values
y = df.drop("review", axis=1).values
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)
```

Apply vectorization:

```
vect = CountVectorizer()
X_train = vect.fit_transform(X_train)
X_test = vect.transform(X_test)
```

Apply frequency, inverse document frequency:

```
tfidf = TfidfTransformer()
X_train = tfidf.fit_transform(X_train)
X_test = tfidf.transform(X_test)
X_train = X_train.toarray()
X_test = X_test.toarray()
```

Build a Model with Deep Learning

Add different layers to models:

```
model = Sequential()
model.add(Dense(units=12673, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(units=4000, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(units=500, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(units=3, activation="softmax"))
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])
early_stop = EarlyStopping(monitor="val_loss", mode="min", verbose=1,
patience=2)
```

Fit the model:

```
model.fit(
    x=X_train,
    y=y_train,
    batch_size=256,
    epochs=100,
    validation_data=(X_test, y_test),
    verbose=1,
    callbacks=early_stop,
)
```

```
Epoch 1/100
15/66 [=====>.....................] - ETA: 4:18 - loss: 0.9292 - accuracy: 0.5203
```

Figure 16: Training the model.

Evaluation of Model

Evaluation of the model:

```
model_score = model.evaluate(X_test, y_test, batch_size=64, verbose=1)
print("Test accuracy:", model_score[1])
```

```
113/113 [==============================] - 62s 548ms/step - loss: 0.3504 - accuracy: 0.9229
Test accuracy: 0.9229166507720947
```

Figure 17: Testing the accuracy of the model.

Prediction of Result

```
preds = model.predict(X_test)preds
```

Famous Python Libraries for the Sentiment Analysis

These are some of the famous Python libraries for sentiment analysis:

- NLTK (Natural Language Toolkit).
- SpaCy.
- TextBlob.
- Standford CoreNLP.

Applications of Sentiment Analysis

There are many applications where we can apply sentimental analysis methods. Some of these are:

- Market monitoring.
- Keeping track of feedback from the customers.
- Helps in improving the support to the customers.
- Keeping an eye on the competitors.
- Used in Recommendation systems.
- Display of ads on webpages.
- Filtering spam of abusive emails.
- Psychological evaluation.
- Online e-commerce, where customers give feedback.
- Sentiment analysis in social sites such as Twitter or Facebook.
- Understand the broadcasting channel-related TRP sentiments of viewers.

Conclusion

Sentiment analysis aims at getting sentiment-related knowledge from data, especially now, due to the enormous amount of information on the internet. In other words, we can generally use a sentiment analysis approach to understand opinion in a set of documents.

Sentiment analysis is sometimes referred to as opinion mining, where we can use NLP, statistics, or **machine learning methods** to extract, identify, or otherwise characterize a text unit's sentiment content.

Consumers can use sentiment analysis to research products and services before a purchase. Public companies can use public opinions to determine the acceptance of their products in high demand.

For example, moviegoers can look at a movie's reviews and then decide whether to watch a movie or not. Perceiving a sentiment is natural for humans. Also, sentiment analysis can be used to understand the opinion in a set of documents. Hence, **sentiment analysis** is a great mechanism that can allow applications to understand a piece of writing's underlying subjective nature, in which **NLP** also plays a vital role in this approach.