SENTIMENTAL ANALYSIS

A Main Project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
ARTIFICIALINTELLIGENCE
AND MACHINE LEARNING

Submitted by

1)NITHIN KUMAR VEMANA

2)JASMITHA KAMALA PRIYA KOLAMURI

3)RATNA GAYATHRI SANNIDHI

Under the esteemed guidance of
Mr.SRINIVASA RAO
Assistant
Professor

DEPARTMENT OF AIML

SWARNANDHRA COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

(Approved by AICTE, Accredited by NBA & NAAC
and permanently affiliated to JNTU Kakinada)
NARSAPUR – 534275

CERTIFICATE

This is to certify that the project entitled “SENTIMENTAL ANALYSIS”, is being submitted by V.NITHIN KUMAR, K.JASMITHA ,S.GAYATHRI Bearing the REGD.NOS: 20A21A6145, 20A21A6130,20A21A6150 submitted in fulfillment for the award of the degree of “BACHELOR OF TECHNOLOGY ” in “ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING” is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2021-2022 and it has been found worthy of acceptance according to the

INSTRUCTOR	**INSTRUCTOR SIGNATURE**

ACKNOWLEDGEMENT

Dear Satya Kishor Mavuri ,

I hope this email finds you well. I am writing to express my sincere gratitude for the opportunity to undertake an internship project at SNAD DEVELOPERS. I am extremely thankful for the valuable experience and knowledge I gained during my time here.

I want to extend my appreciation to the entire HR team for their efforts in organizing and coordinating the internship program. Special thanks go to my Supervisor for their guidance, support, and mentorship throughout the project duration.

The internship provided me with a unique insight into the inner workings of SNAD DEVELOPERS, and I am truly impressed by the company's commitment to excellence and innovation. The practical exposure has complemented my academic learning and has given me a deeper understanding of the industry.

Moreover, I am grateful for the collaborative and inclusive work environment, where my opinions were valued, and I was given the chance to contribute to real projects. The constructive feedback I received during my internship has been instrumental in my personal and professional growth.

I sincerely believe that this experience will have a significant positive impact on my future endeavors. The knowledge and skills acquired will undoubtedly shape my career path and help me become a more competent professional.

Once again, thank you for granting me this enriching opportunity. I look forward to staying connected with SNAD DEVELOPERS and hope our paths cross again in the future.

With deepest gratitude

NITHIN KUMAR VEMANA

JASMITHA KOLAMURI

RATNA GAYATHRI SANNIDHI

ABSTRACT

- Sentiment Analysis ,a prominent subfield of Natural Language Processing (NLP), plays a crucial role in understanding and interpreting emotions conveyed through textual data.

- This abstract highlights the significance and key aspects of sentiment analysis in the context of modern applications.

- The objective is to explore various techniques and methodologies utilized to classify sentiments as positive, negative, or neutral, enabling us to extract valuable insights from vast amounts of unstructured text data.

- We discuss the challenges faced in sentiment analysis, such as sarcasm, context-dependent expressions, and language nuances, while also addressing the implications of sentiment classification in various domains, including social media, customer feedback, market research, and political discourse.

- Furthermore, the abstract touches upon the ethical considerations related to sentiment analysis, such as privacy and bias.

- By comprehending the sentiment behind the words, sentiment analysis becomes a powerful tool for decision-making and gaining a deeper understanding of human emotions as expressed by textual communication.

CONTENTS

1. Introduction to Sentimental Analysis
    - What is sentiment Analysis
    - Importance in Project Context
    - Read in Data and NLTK Basics

2.Previous Work
- Work
- Analyze IMDB Reviews
- The ideas

3.Methodology of sentiment Analysis
- Approaches to Sentiment Analysis
- Application Scope

4.Vaders

5.Development Facilities

6.Requirement Specification
- Functional Requirements

7.Development Process

- Problem Definition and Scope
- Data Collection
- Data Processing
- Feature Extraction
- Model Selection
- Rule Based Approach
- Machine Learning Algorithms
- Deep Learnings Models
- Model Training

8. Architecture Overview

- Data Collection
- Data Processing
- Model Selection
- Training
- Evaluation
- Inference
- Monitoring And Maintenance
- Feedback Loop

9. Implementation

10. DataFlow Diagram

11. Conclusion

1.What is Sentiment Analysis

1.1 Introduction to Sentimental Analysis

Sentiment analysis, also known as sentiment analysis or opinion mining, is a field of natural language processing (NLP) that focuses on understanding and extracting subjective information from text data. The primary goal of sentiment analysis is to determine the sentiment or emotional tone expressed in a piece of text, whether it is positive, negative, neutral, or a mix of these sentiments.

Sentiment analysis has gained significant importance in recent years due to the vast amount of text data generated on social media platforms, review websites, news articles, customer feedback, and more. It has numerous applications across various industries, including marketing, customer service, finance, and product development. Here's a brief introduction to some key aspects of sentiment analysis:

Sentiment Polarity:

1.Positive

2. Negative

3.Neutral

Methods of Sentiment Analysis:

1.Rule-Based

2.Machine Learning

3.Deep Learning:

Challenges in Sentiment Analysis:

1.Contextual Understanding

2.Multilingual Analysis

3.Data Imbalance

4.Social Media Monitoring

5.Customer Feedback Analysis

6.Stock Market Prediction

7.Product Reviews

8.Political Analysis

1.2 Importance in Project Context

Sentiment analysis holds significant importance in the context of various projects, especially when the project involves analyzing and making decisions based on textual data. Here's why sentiment analysis is crucial in the context of projects focused on sentiment analysis itself.

Data-Driven Decision Making: In sentiment analysis projects, the primary goal is to extract meaningful insights from textual data. Whether it's assessing public sentiment towards a political candidate, tracking customer feedback for a product, or monitoring social media mentions of a brand, sentiment analysis helps project teams make datadriven decisions. This is invaluable for shaping strategies, adjusting campaign tactics, and optimizing project outcomes.

Real-time Feedback: Sentiment analysis provides real-time feedback on how a project or initiative is being perceived by the target audience. This immediate insight allows project managers to make on-the-fly adjustments to ensure the project stays on track and meets its objectives.

Performance Evaluation: Sentiment analysis allows for the evaluation of project performance based on public sentiment. If the sentiment is overwhelmingly positive, it indicates success and may lead to continued efforts in the same direction. Conversely, if sentiment is negative, it's a signal to reevaluate the project's approach and make necessary changes.

Risk Mitigation: Sentiment analysis can help project teams identify potential risks early on. By monitoring sentiment trends, teams can anticipate issues or negative reactions and proactively address them, reducing the impact on the project's success.

Enhanced Stakeholder Communication: Project stakeholders, such as clients, investors, or the public, often want to understand how a project is perceived. Sentiment analysis provides a clear and quantifiable way to communicate project sentiment to stakeholders, helping to build trust and transparency.

Accurate Sentiment Classification: The core goal of a sentiment analysis project is to accurately classify and quantify sentiments within text data. This accuracy is vital for drawing meaningful conclusions and making data-driven decisions.

Project Customization and Targeting: In project settings, sentiment analysis allows for tailored and precise strategies. For instance, in marketing campaigns, understanding sentiment helps customize messaging to resonate with the target audience's emotional disposition, thus improving engagement and conversion rates.

1.3 Read in Data and NLTK Basics

Reading in data and using NLTK (Natural Language Toolkit) basics are essential steps when working with text data and natural language processing tasks. NLTK is a Python library that provides tools and resources for working with human language data. Importing NLTK:

First, you need to import the NLTK library in your Python script. You can do this using the following code.

```
import nltk

pip install nltk
```

Downloading NLTK Resources:

NLTK provides a variety of datasets, corpora, and resources that you may need for different NLP tasks. To download these resources, you can use NLTK's download function:

```
nltk.download('all')
```

This command downloads all available NLTK resources, but you can specify specific resources if needed.

Reading Text Data.

To work with text data, you'll need to read it into your Python script. There are various ways to do this, depending on your data source. Here are a few common scenarios: Reading from a Text File.

Reading from a CSV File:

If your text data is stored in a CSV file, you can use libraries like pandas to read it:

```
import pandas as pd
```

```
df = pd.read_csv('your_csv_file.csv')
```

NLTK provides tools for tokenization, which is the process of splitting text into individual words or tokens. Here's how you can tokenize text using NLTK

```
from nltk.tokenize import word_tokenize tokens = word_tokenize(text_data)
```

This will give you a list of words or tokens from your text data.

2 .Previous work

2.1 work :

- we saw some of the best projects based on the sentimental analysis in this part we inspire our project based on the sentimental analysis, we are elaborating their features and what makes them successful.

2.2 Analyze IMDb Reviews:

Analyze IMDb reviews was the highly awaited analysis of the present days .It became very successful for its reviews. It predicts the reviews of the movies either it is good to see or not. Now-a-days ever one are watching the IMDb reviews to go and watch the movies. It is very successful analyze now-a-days.

2.3 The idea:

Our Main idea is to analysis the data of Amazon product review. For this we take the dataset of the Amazon product review.

3.Methodology of sentiment Analysis

Data Collection: The first step in sentiment analysis is gathering textual data relevant to your analysis. This data can come from various sources, including social media, customer reviews, news articles, surveys, and more. The quality and quantity of your data are crucial factors

Data Preprocessing: Raw text data often requires preprocessing to make it suitable for analysis. Common preprocessing steps include text cleaning (removing special characters, HTML tags, etc.), tokenization (splitting text into words or tokens), and removing stopwords. Depending on the task, you may also perform stemming or lemmatization.

Labeling or Annotation (if supervised): In supervised sentiment analysis, you need labeled data, where each piece of text is associated with a sentiment label (e.g., positive, negative, neutral). Labeling can be done manually or through automated methods.

Feature Extraction: This step involves converting text data into numerical features that machine learning models can understand. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings (e.g., Word2Vec, GloVe), and more advanced methods like BERT embeddings.

Model Selection: Choose an appropriate sentiment analysis model or algorithm based on your data and objectives. Common choices include rule-based models, machine learning models (e.g., Naive Bayes, Support Vector Machines, Random Forests), and deep learning models (e.g., LSTM, BERT).

Model Training (if supervised): If you're using a supervised approach, train your model on the labeled dataset. This involves feeding the features extracted from the training data into the model and adjusting its parameters to minimize the prediction error.

Model Evaluation: Assess the performance of your sentiment analysis model using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score) and techniques like cross-validation. Fine-tune your model if necessary to improve its performance.

Deployment: Once your sentiment analysis model is trained and evaluated, deploy it in a production environment where it can analyze real-time or batch text data. This deployment can be through APIs, web applications, or other integration methods.

Monitoring and Maintenance: Continuously monitor the performance of your sentiment analysis system in a production environment. Reevaluate and retrain the model periodically to adapt to changing language and sentiment trends.

3.1 Approaches to Sentiment Analysis:

Rule-Based Approaches: Rule-based methods use predefined sets of rules and lexicons to determine sentiment. For example, a list of positive and negative words can be used to assign sentiment scores to text.

Supervised Learning: In supervised learning, sentiment labels are provided for a labeled dataset. Machine learning models are trained on this data to predict sentiment labels for new, unlabeled text. Common algorithms include Naive Bayes, Support Vector Machines, and Random Forests.

Unsupervised Learning: Unsupervised methods involve clustering or topic modeling to group similar text based on sentiment. This approach doesn't require labeled data but may require manual interpretation of clusters.

Semi-Supervised Learning: This approach combines elements of both supervised and unsupervised learning. It uses a small amount of labeled data and a larger amount of unlabeled data to build sentiment models.

Deep Learning: Deep learning models, particularly recurrent neural networks (RNNs) and transformer-based models like BERT, have gained popularity due to their ability to capture complex contextual relationships in text. They excel in sentiment analysis tasks, especially when large datasets are available.

3.2 Application Scope:

Marketing and Brand Management: Monitoring social media sentiment, assessing the impact of marketing campaigns, and understanding consumer opinions about products and brands.

Customer Service: Analyzing customer feedback and reviews to improve service quality and address customer issues.

Finance and Stock Market: Predicting stock market trends based on news sentiment, analyzing sentiment in financial reports, and making investment decisions.

Product Development: Identifying areas for product improvement based on user feedback and reviews.

Political Analysis: Assessing public sentiment toward political candidates, parties, and policies during elections.

Healthcare: Improving patient experiences and healthcare services by analyzing patient feedback.

E-commerce and Retail: Enhancing customer experiences by analyzing product reviews and optimizing pricing strategies.

Media and Entertainment: Personalizing content recommendations, predicting box office success, and monitoring audience reactions.

Customer Satisfaction Surveys: Analyzing open-ended survey responses to gauge customer satisfaction.

Market Research: Analyzing customer opinions to understand market trends and customer preferences.

Fake News Detection: Identifying misleading or false information in news articles and social media posts.

4. VADER

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data.

Vader is easy to use and does not require extensive training or complex model setup. It is available as a Python library, making it accessible to a wide range of users, including those without a deep understanding of machine learning or NLP techniques.

Vader sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.

In addition to individual valence scores, Vader calculates a compound score for the entire text. This compound score represents the overall sentiment intensity of the text. It takes into account both the valence scores of the individual words and their relative positions within the text. The compound score can range from -1 (extremely negative) to 1 (extremely positive), with 0 indicating a neutral sentiment. The compound score is a valuable metric for understanding the overall sentiment of a piece of text.

Vader is used in various real-world applications, including social media monitoring, customer feedback analysis, political analysis, market research, and chatbot development, to name a few. Its ease of integration and quick results make it an attractive choice in these domains.

Vader provides users with immediate sentiment scores, making it suitable for obtaining quick insights into the sentiment of a text without the need for extensive data preprocessing or model training.

For Example- Words like *'love', 'enjoy', 'happy', 'like'* all convey a positive sentiment. Also Vader is intelligent enough to understand the basic context of these words, such as *"did not love"* as a negative statement. It also understands the emphasis of capitalization and punctuation, such as *"ENJOY"*

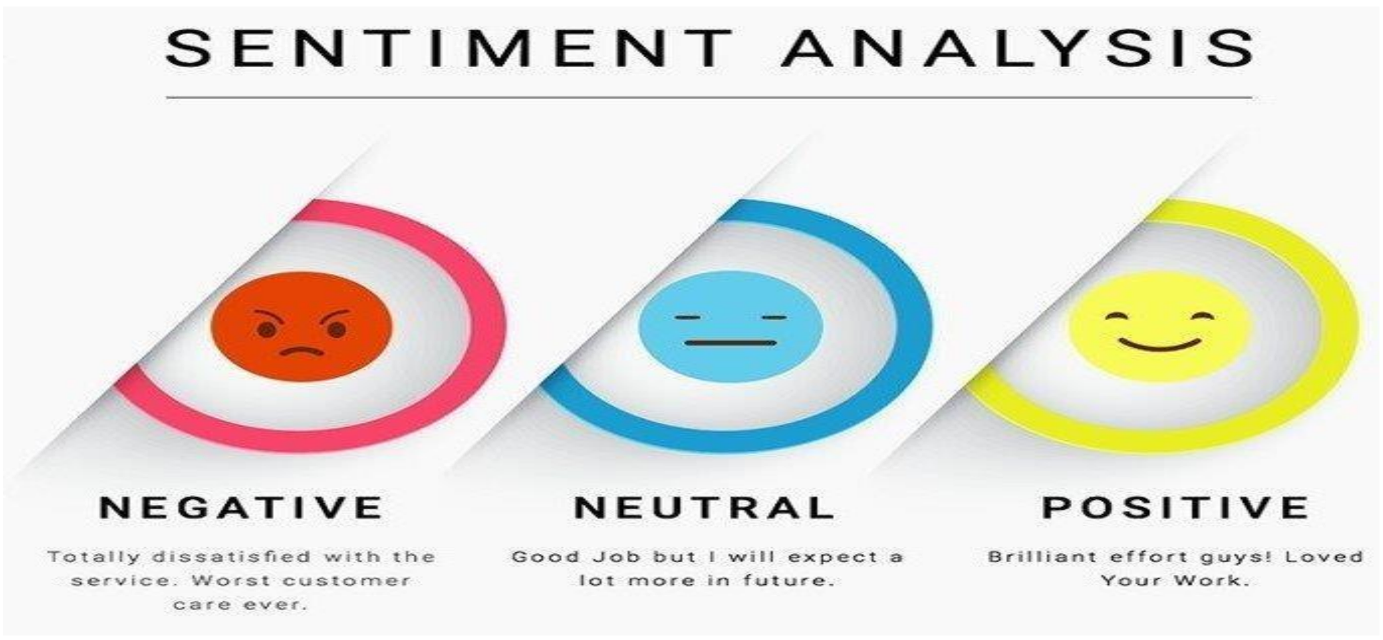
While Vader is primarily designed for English text, it has been adapted and extended for other languages, making it versatile for sentiment analysis tasks in multiple languages

5.Development Facilities :

Software's Constraints:

Software required for the project are:

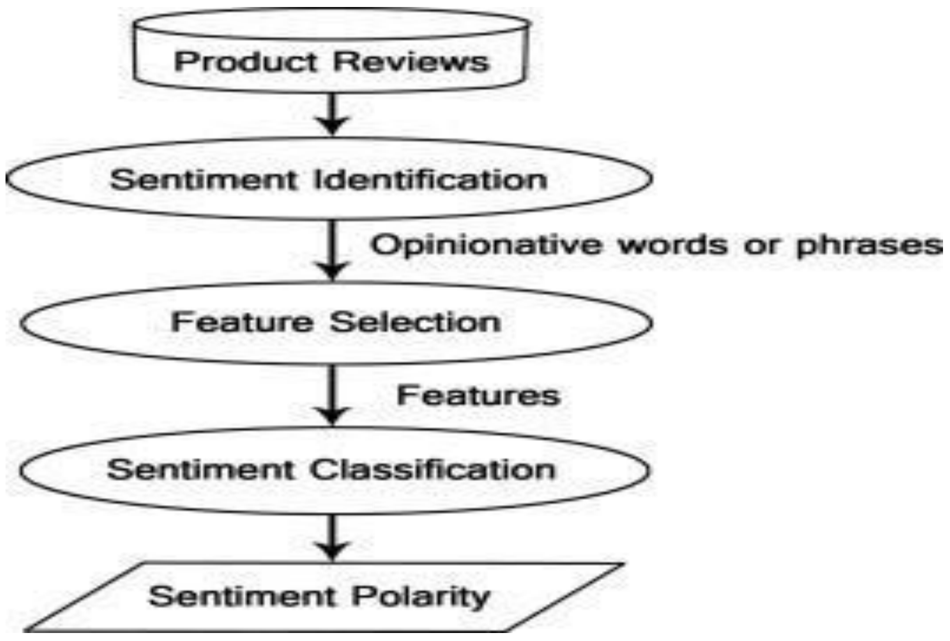
- Programming Language:
- Sentiment Lexicons:
- Machine Learning Models:
- Data Preprocessing Tools:
- Data Visualization Tools:
- NLP Libraries:
- Web Scraping Tools
- Anaconda navigator
- Cloud Services


SENTIMENT ANALYSIS
NEGATIVE
Totally dissatisfied with the service. Worst customer care ever.
NEUTRAL
Good Job but I will expect a lot more in future.
POSITIVE
Brilliant effort guys! Loved Your Work.

6.Requirement specification

6.1.Functional Requirements :

- Text input
- Preprocessing
- Sentiment Classification
- Multi-Lingual Support
- Context Understanding
- Customization
- Scalability
- Accuracy and Confidence Level
- Feedback Loop
- API or Integration
- Reporting and Visualization
- Real-time Analysis
- Handling Negation and Intensity
- Handling Emojis and Emoticons
- User Authentication



6.1.1 Text input :

- The system should be able to accept various forms of text input, such as short sentences, paragraphs, or longer documents.

6.1.2 Preprocessing :

- The system should perform text preprocessing tasks like removing special characters, punctuation, and converting text to lowercase to ensure consistent analysis.

6.1.3 Sentiment Classification :

- The core functionality is to classify the sentiment of the input text as positive, negative, or neutral. The system should accurately determine the sentiment expressed in the text.

6.1.4 Multi-Lingual Support:

- The system might need to support sentiment analysis for multiple languages, not just English.

6.1.5 Context Understanding :

- It should consider the context of the text, as sentiment can heavily depend on the context. For instance, sarcasm might convey a different sentiment than the literal meaning of the words.

6.1.6 Customization :

- The system might allow users to customize or fine-tune sentiment analysis for specific domains or industries, enhancing accuracy for specialized content.

6.1.7 Scalability :

- The system should handle a large volume of text inputs efficiently, especially in applications where real-time or batch processing is required.

6.1.8 Accuracy and Confidence Levels:

- The system should provide a sentiment score or label along with a confidence level to indicate the reliability of the analysis.

6.1.9 Feedback Loop:

- In some cases, users might be able to provide feedback on the accuracy of the sentiment analysis results, which can help improve the system over time.

6.1.10 API or Integration:

- If the sentiment analysis is part of a larger application, there should be an API or integration mechanism to allow seamless usage.

6.1.11 Reporting and Visualization:

- The system might generate reports or visualizations that summarize the sentiment distribution or trends in analyzed text.

6.1.12 Real-time Analysis:

- For applications like social media monitoring, the system might need to provide sentiment analysis in near real-time.

6.1.13 Handling Negation and Intensity:

- The system should be able to understand negation and intensity modifiers that can change the sentiment text.

6.1.14 User Authentication:

- In applications where sentiment analysis is performed based on user-generated content, there might be a need for user authentication to ensure data privacy and security.

7.Development process

The development process for sentiment analysis involves a series of steps to build, train, and deploy a sentiment analysis model or system. Here's a general outline of the process:

7.1 Problem Definition and Scope:

Define the scope of sentiment analysis: Determine the type of text data you will be analyzing (e.g., product reviews, social media posts) and the sentiment categories (e.g., positive, negative, neutral) you want to classify.
Identify the target audience and intended use cases for the sentiment analysis results.

7.2 Data Collection:

- Collect a diverse dataset of text samples that represent the sentiment expressions relevant to your domain and use cases. ○ Label the data with corresponding sentiment categories (e.g., positive, negative) through manual annotation or using existing sentiment labels.

7.3 Data Preprocessing:

- Clean the text data by removing noise, special characters, and irrelevant information.
- Tokenize the text into words or subword units. ○ Normalize the text by converting to lowercase and removing punctuation. ○ Handle stopwords and perform stemming or lemmatization as needed.

7.4 Feature Extraction:

- Convert the processed text data into numerical representations that machine learning algorithms can work with.
- Common techniques include using bag-of-words, TF-IDF (Term FrequencyInverse Document Frequency), and word embeddings (e.g., Word2Vec, GloVe).

7.5 Model Selection:

- Choose a suitable sentiment analysis algorithm or model architecture.

7.6 Rule-based approaches:

- Using predefined rules and dictionaries to determine sentiment.

7.7 Machine learning algorithms:

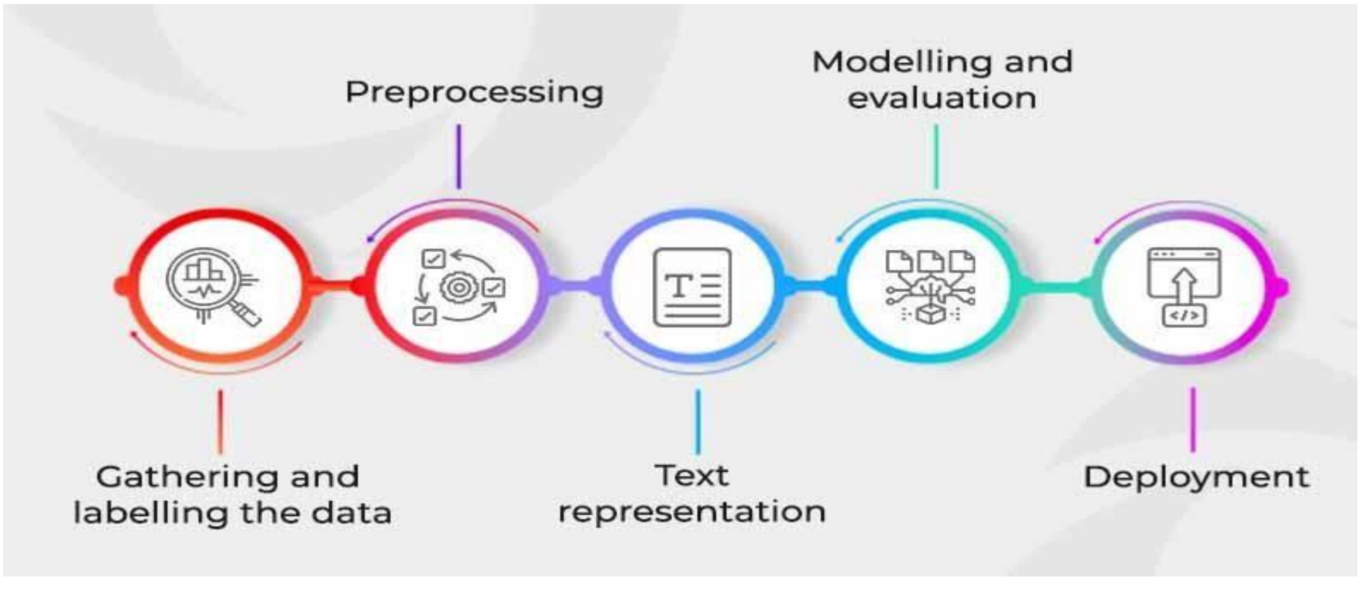
- Using algorithms like Naive Bayes, Support Vector Machines (SVM), or decision trees.

7.8 Deep learning models:

- Using neural networks, such as Recurrent Neural Networks (RNNs) or Transformer-based models (like BERT or GPT).

7.9 Model Training:

- Split the labeled dataset into training, validation, and testing sets. ○ Train the selected model using the training set and tune hyperparameters using the validation set.
- Evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and confusion matrices.



8. Architecture overview

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) technique that aims to determine the sentiment or emotional tone expressed in text data. Here's an architecture overview of sentiment analysis:

8.1 Data Collection:

- The first step is to gather a dataset containing text samples with associated sentiment labels (e.g., positive, negative, neutral). This data can be collected from various sources like social media, reviews, or surveys.

8.2 Data Preprocessing:

- Raw text data needs to be preprocessed to make it suitable for analysis. This involves tasks such as tokenization (breaking text into words or phrases), lowercasing, removing punctuation, and stop words.

8.3 Feature Extraction:

- Text data needs to be converted into numerical features that machine learning models can work with. Common methods include TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings like Word2Vec or GloVe.

8.4 Model Selection:

- **Choose an appropriate sentiment analysis model. Common choices include:**

Rule-based Models:

- These models rely on predefined rules and lexicons to determine sentiment based on words and phrases.

Machine Learning Models:

- Supervised models like Support Vector Machines (SVM), Naive Bayes, or more advanced deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Transformer-based Models:

- State-of-the-art models like BERT, GPT, and their variants have achieved impressive results in sentiment analysis tasks.

8.5 Training:

- If using a machine learning or deep learning model, the selected model is trained on the labeled dataset. During training, the model learns to predict sentiment based on the features extracted from the text.

8.6 Evaluation:

- The model's performance is assessed using metrics like accuracy, precision, recall, and F1-score on a separate validation dataset. Hyperparameter tuning may be necessary to optimize performance.

8.7 Deployment:

- Once the model is trained and evaluated satisfactorily, it can be deployed in a real-world application. This could be a web service, API, or integrated into an existing application.

8.8 Inference:

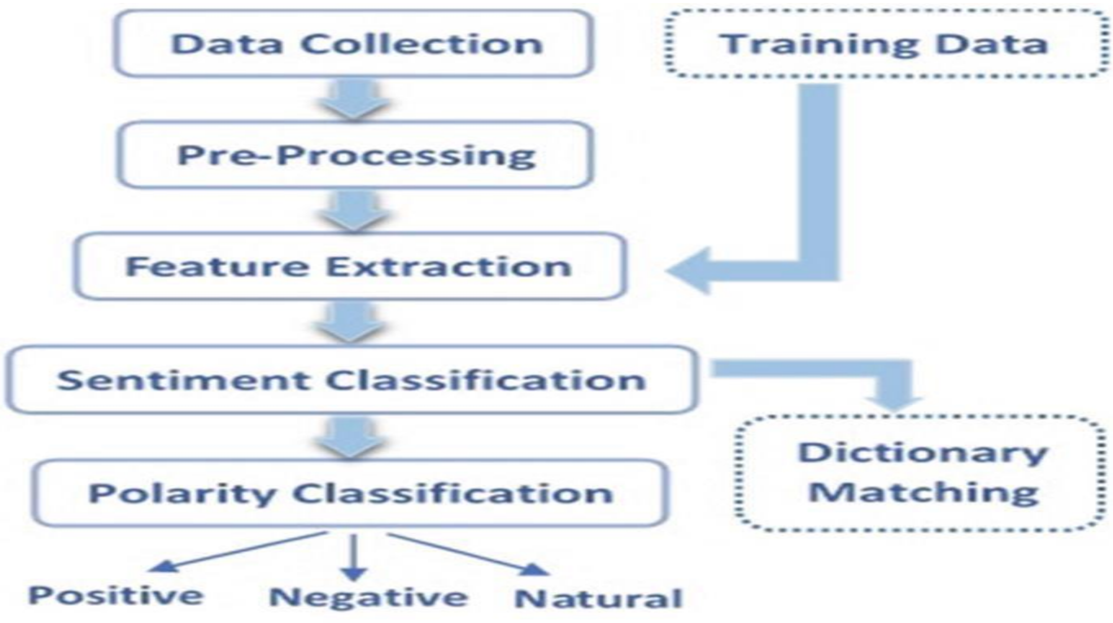
- In the deployment phase, users can input text, and the model predicts the sentiment of that text.

8.9 Monitoring and Maintenance:

- Continuous monitoring and updates are essential to maintain model performance over time. New data and evolving language trends may require periodic retraining.

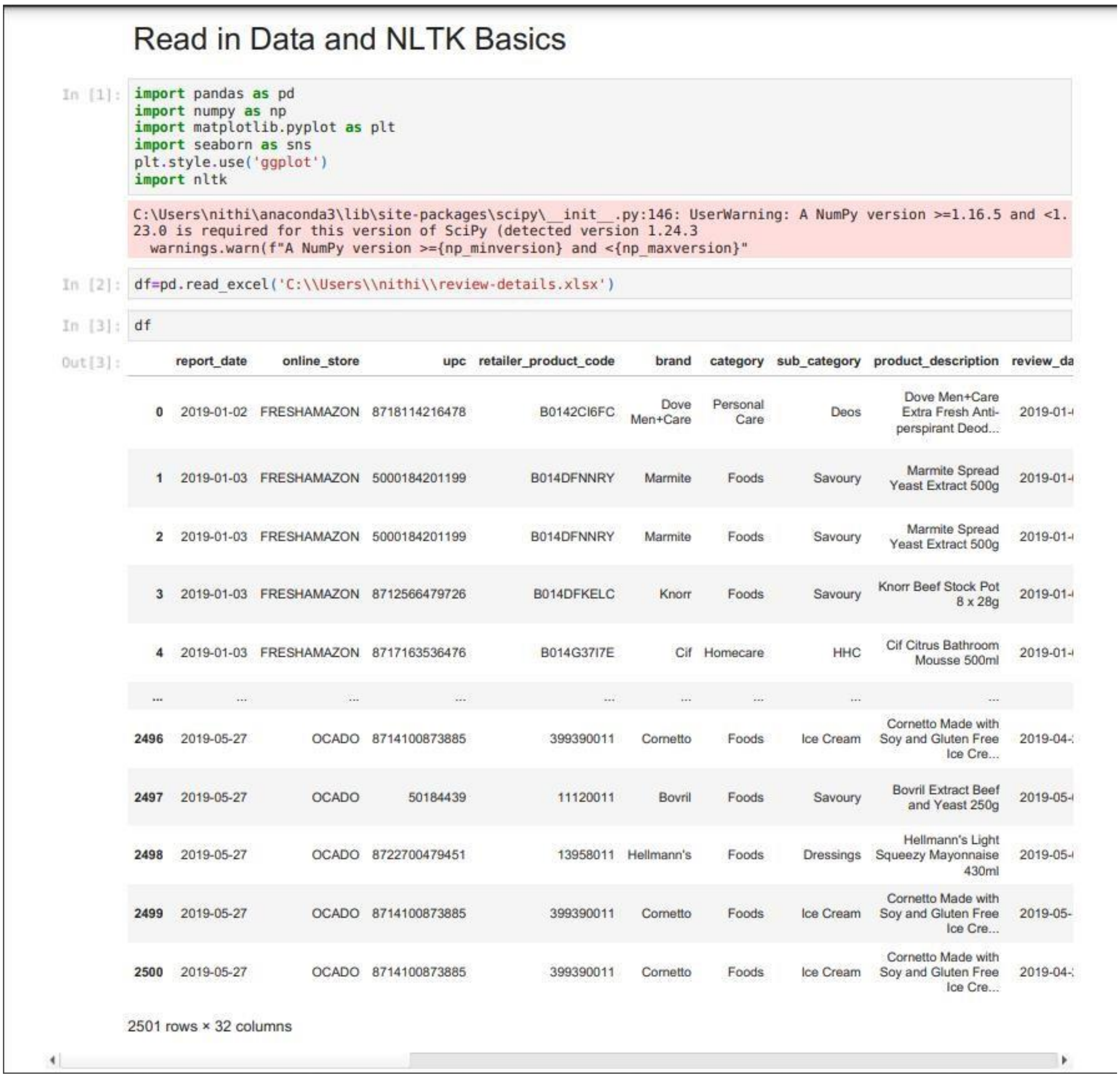
8.10 Feedback Loop:

- User feedback can be collected to improve the model further. This feedback can be used to update the model or retrain it to handle new sentiment expressions.



This architecture provides a high-level overview of the sentiment analysis process, which involves collecting, preprocessing, modeling, and deploying sentiment analysis systems to gain insights from textual data.

9. Implementation



Read in Data and NLTK Basics

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
import nltk
```

```
C:\Users\nithi\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.
23.0 is required for this version of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```python
df=pd.read_excel('C:\\Users\\nithi\\review-details.xlsx')
```

In [3]:
```python
df
```

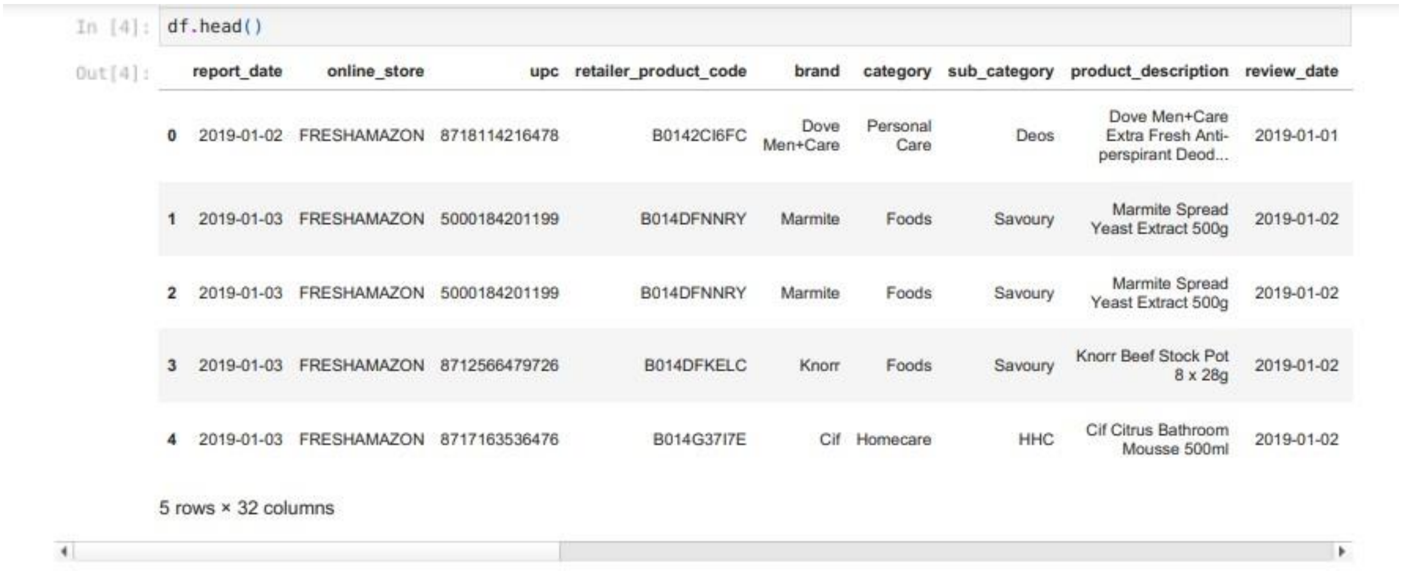
Out[3]:

	report_date	online_store	upc	retailer_product_code	brand	category	sub_category	product_description	review_da
0	2019-01-02	FRESHAMAZON	8718114216478	B0142CI6FC	Dove Men+Care	Personal Care	Deos	Dove Men+Care Extra Fresh Anti-perspirant Deod...	2019-01-
1	2019-01-03	FRESHAMAZON	5000184201199	B014DFNNRY	Marmite	Foods	Savoury	Marmite Spread Yeast Extract 500g	2019-01-
2	2019-01-03	FRESHAMAZON	5000184201199	B014DFNNRY	Marmite	Foods	Savoury	Marmite Spread Yeast Extract 500g	2019-01-
3	2019-01-03	FRESHAMAZON	8712566479726	B014DFKELC	Knorr	Foods	Savoury	Knorr Beef Stock Pot 8 x 28g	2019-01-
4	2019-01-03	FRESHAMAZON	8717163536476	B014G37I7E	Cif	Homecare	HHC	Cif Citrus Bathroom Mousse 500ml	2019-01-
...	...	...	...	...	...	...	...	...	
2496	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-04-
2497	2019-05-27	OCADO	50184439	11120011	Bovril	Foods	Savoury	Bovril Extract Beef and Yeast 250g	2019-05-
2498	2019-05-27	OCADO	8722700479451	13958011	Hellmann's	Foods	Dressings	Hellmann's Light Squeezy Mayonnaise 430ml	2019-05-
2499	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-05-
2500	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-04-

2501 rows × 32 columns

Different operations using NLTK.

- Tokenization: The breaking down of text into smaller units is called tokens.
- Lower case conversion ○ Stop Words removal ○ Stemming
- Lemmatization
- Parse tree or Syntax Tree generation
- POS Tagging ○ Conclusion

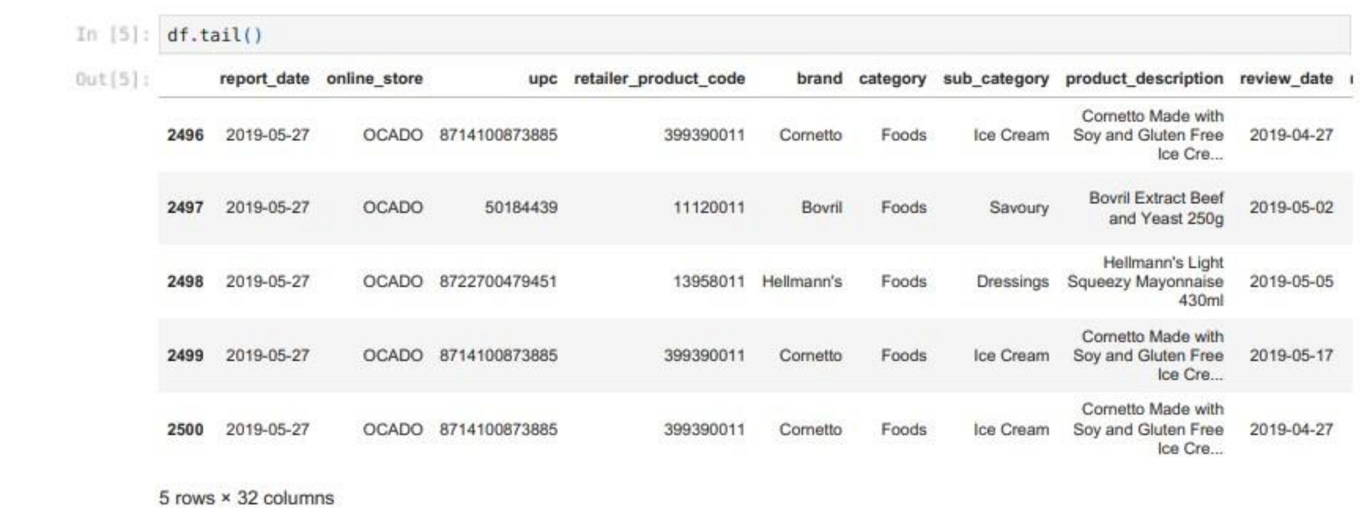


```
In [4]: df.head()
```

Out[4]:

	report_date	online_store	upc	retailer_product_code	brand	category	sub_category	product_description	review_date
0	2019-01-02	FRESHAMAZON	8718114216478	B0142CI6FC	Dove Men+Care	Personal Care	Deos	Dove Men+Care Extra Fresh Anti-perspirant Deod...	2019-01-01
1	2019-01-03	FRESHAMAZON	5000184201199	B014DFNNRY	Marmite	Foods	Savoury	Marmite Spread Yeast Extract 500g	2019-01-02
2	2019-01-03	FRESHAMAZON	5000184201199	B014DFNNRY	Marmite	Foods	Savoury	Marmite Spread Yeast Extract 500g	2019-01-02
3	2019-01-03	FRESHAMAZON	8712566479726	B014DFKELC	Knorr	Foods	Savoury	Knorr Beef Stock Pot 8 x 28g	2019-01-02
4	2019-01-03	FRESHAMAZON	8717163536476	B014G37I7E	Cif	Homecare	HHC	Cif Citrus Bathroom Mousse 500ml	2019-01-02

5 rows × 32 columns



```
In [5]: df.tail()
```

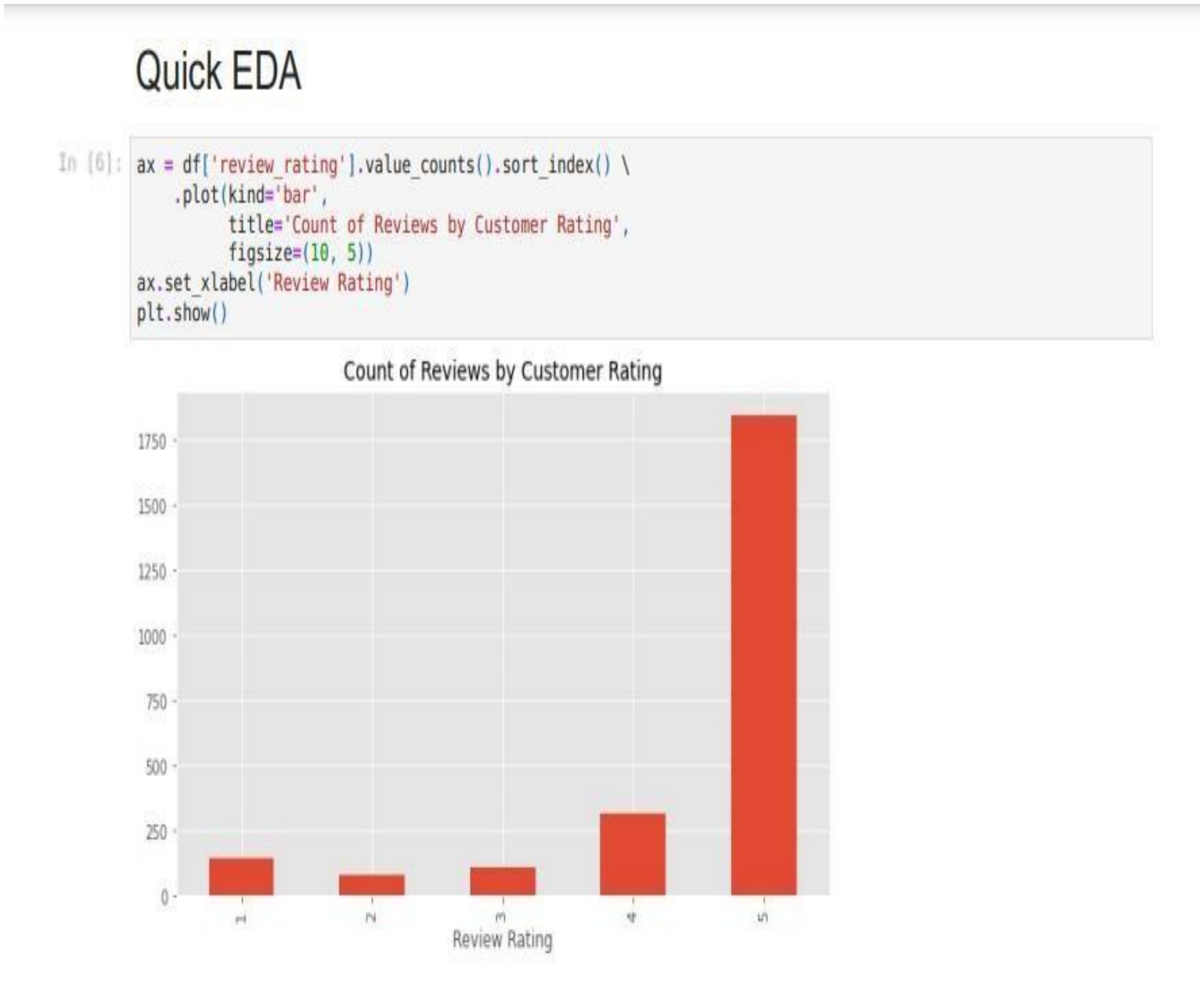
Out[5]:

	report_date	online_store	upc	retailer_product_code	brand	category	sub_category	product_description	review_date
2496	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-04-27
2497	2019-05-27	OCADO	50184439	11120011	Bovril	Foods	Savoury	Bovril Extract Beef and Yeast 250g	2019-05-02
2498	2019-05-27	OCADO	8722700479451	13958011	Hellmann's	Foods	Dressings	Hellmann's Light Squeezy Mayonnaise 430ml	2019-05-05
2499	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-05-17
2500	2019-05-27	OCADO	8714100873885	399390011	Cornetto	Foods	Ice Cream	Cornetto Made with Soy and Gluten Free Ice Cre...	2019-04-27

5 rows × 32 columns

9.1 Quick EDA

- Exploratory Data Analysis (EDA) is the process by which the data analyst becomes acquainted with their data to drive intuition and begin to formulate testable hypotheses. This process typically makes use of descriptive statistics and visualizations.



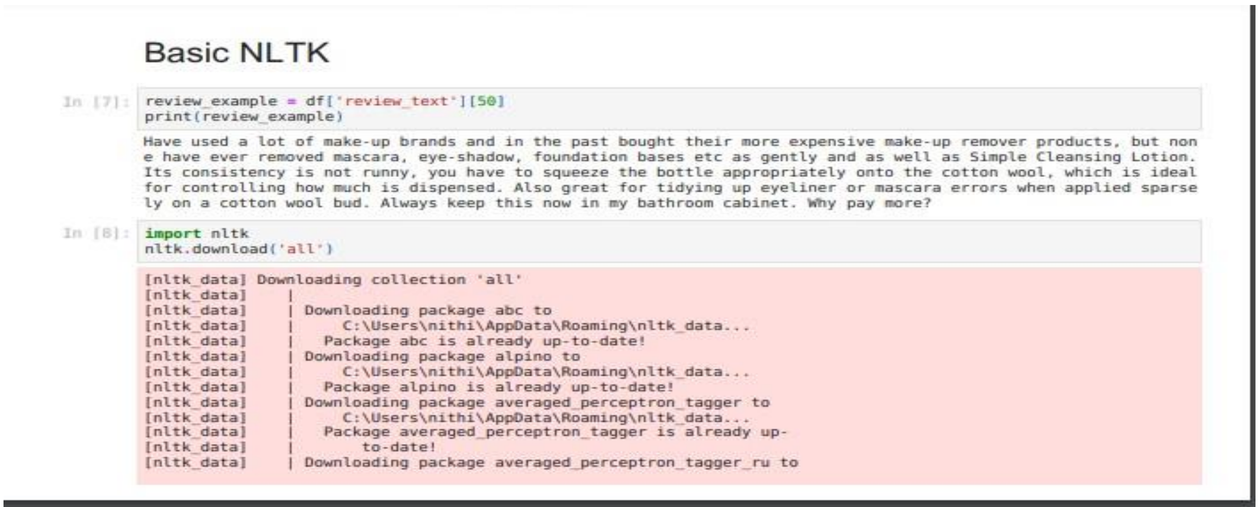
Quick EDA

```
In [6]: ax = df['review_rating'].value_counts().sort_index() \
            .plot(kind='bar',
                  title='Count of Reviews by Customer Rating',
                  figsize=(10, 5))
        ax.set_xlabel('Review Rating')
        plt.show()
```

Exploratory Data Analysis (EDA) is the process of summarization of a dataset by analyzing it. It is used to investigate a dataset and lay down its characteristics. EDA is a fundamental process in many Data Science or Analysis tasks.

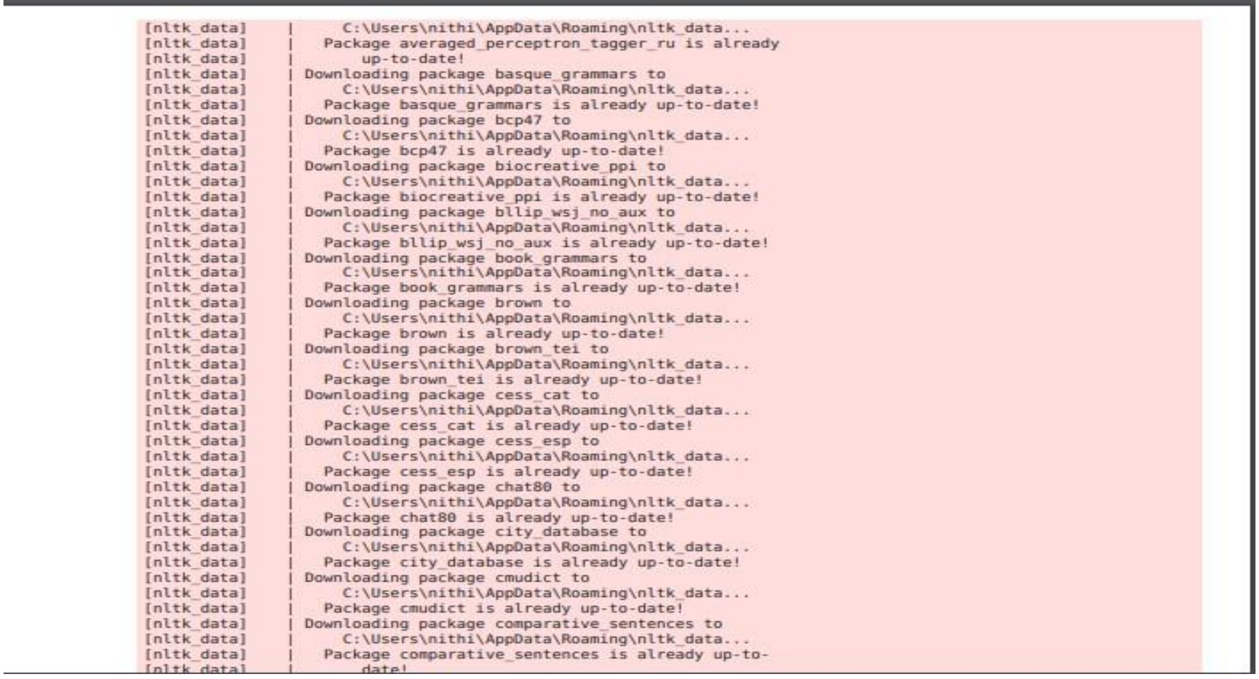
9.2 Basic NLTK:

- The Natural Language Toolkit (NLTK) is a popular open-source library for natural language processing (NLP) in Python. It provides an easy-to-use interface for a wide range of tasks, including tokenization, stemming, lemmatization, parsing, and sentiment analysis.
- NLTK is widely used by researchers, developers, and data scientists worldwide to develop NLP applications and analyze text data.
- One of the major advantages of using NLTK is its extensive collection of corpora, which includes text data from various sources such as books, news articles, and social media platforms. These corpora provide a rich data source for training and testing NLP models.

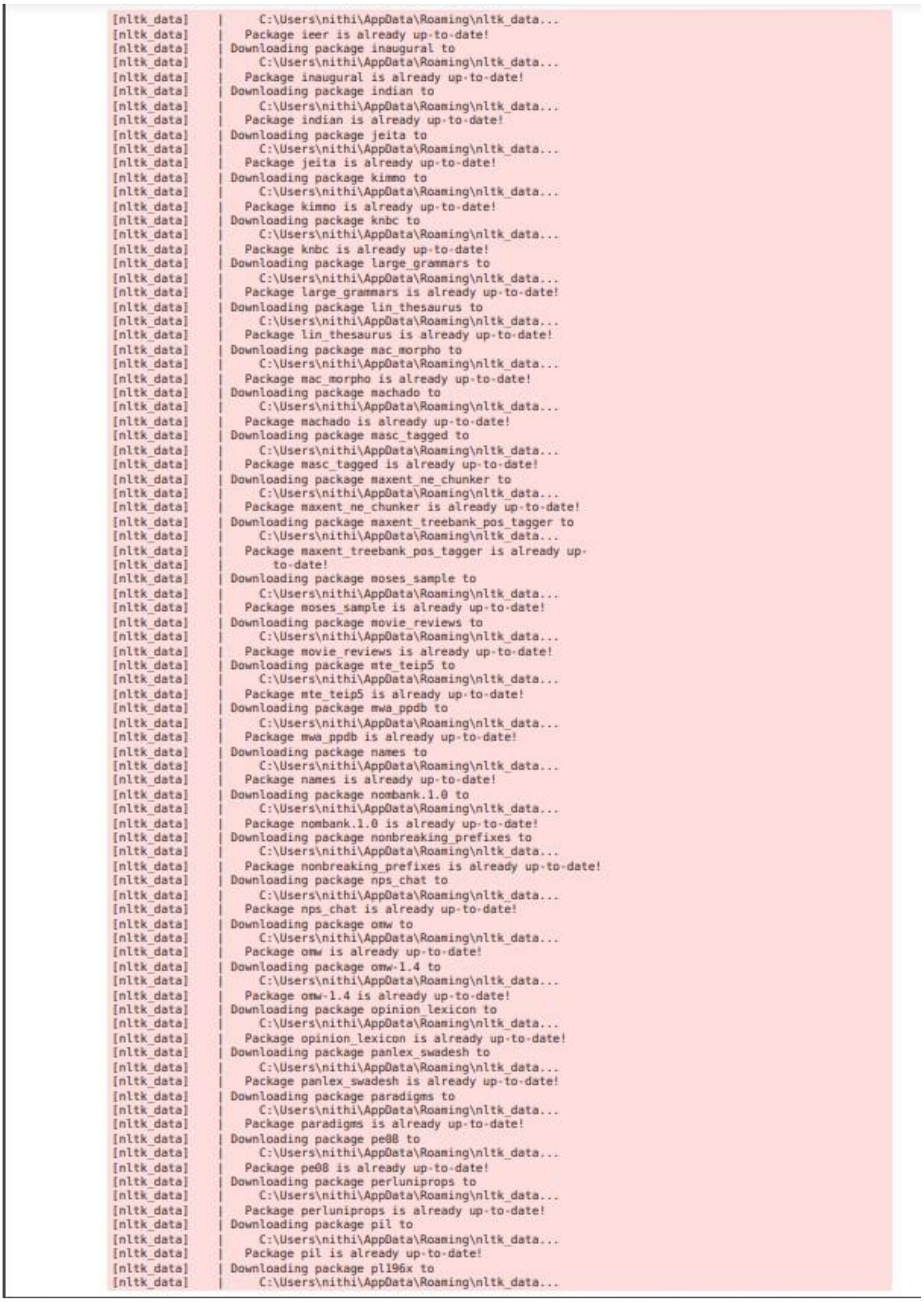


Basic NLTK

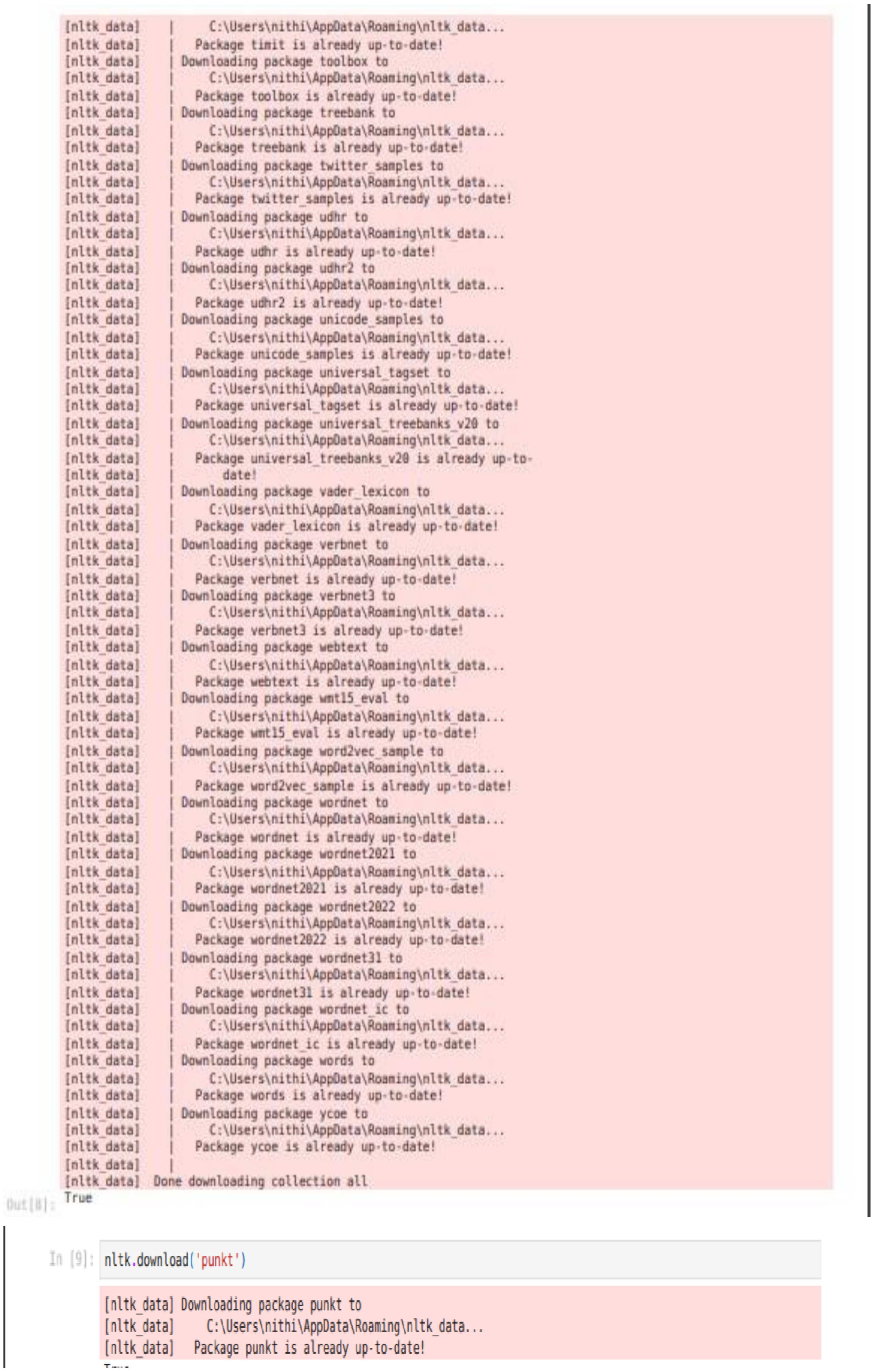
```
In [7]: review_example = df['review_text'][50]
        print(review_example)

Have used a lot of make-up brands and in the past bought their more expensive make-up remover products, but non
e have ever removed mascara, eye-shadow, foundation bases etc as gently and as well as Simple Cleansing Lotion.
Its consistency is not runny, you have to squeeze the bottle appropriately onto the cotton wool, which is ideal
for controlling how much is dispensed. Also great for tidying up eyeliner or mascara errors when applied sparse
ly on a cotton wool bud. Always keep this now in my bathroom cabinet. Why pay more?

In [8]: import nltk
        nltk.download('all')

[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\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]    |     C:\Users\nithi\AppData\Roaming\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]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package basque_grammars is already up-to-date!
[nltk_data]    | Downloading package bcp47 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package bcp47 is already up-to-date!
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package biocreative_ppi is already up-to-date!
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package bllip_wsj_no_aux is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package brown to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package brown_tei to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package brown_tei is already up-to-date!
[nltk_data]    | Downloading package cess_cat to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package cess_cat is already up-to-date!
[nltk_data]    | Downloading package cess_esp to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package cess_esp is already up-to-date!
[nltk_data]    | Downloading package chat80 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package comparative_sentences is already up-to-
[nltk_data]    |       date!
```



```
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package ieer is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package inaugural is already up-to-date!
[nltk_data]    | Downloading package indian to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package indian is already up-to-date!
[nltk_data]    | Downloading package jeita to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package jeita is already up-to-date!
[nltk_data]    | Downloading package kimmo to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package kimmo is already up-to-date!
[nltk_data]    | Downloading package knbc to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package knbc is already up-to-date!
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package large_grammars is already up-to-date!
[nltk_data]    | Downloading package lin_thesaurus to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package lin_thesaurus is already up-to-date!
[nltk_data]    | Downloading package mac_morpho to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package mac_morpho is already up-to-date!
[nltk_data]    | Downloading package machado to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package machado is already up-to-date!
[nltk_data]    | Downloading package masc_tagged to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package masc_tagged is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package maxent_treebank_pos_tagger is already up-
[nltk_data]    |         to-date!
[nltk_data]    | Downloading package moses_sample to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package moses_sample is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package mte_teip5 to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package mte_teip5 is already up-to-date!
[nltk_data]    | Downloading package mwa_ppdb to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package mwa_ppdb is already up-to-date!
[nltk_data]    | Downloading package names to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package names is already up-to-date!
[nltk_data]    | Downloading package nombank.1.0 to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package nombank.1.0 is already up-to-date!
[nltk_data]    | Downloading package nonbreaking_prefixes to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package nonbreaking_prefixes is already up-to-date!
[nltk_data]    | Downloading package nps_chat to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package nps_chat is already up-to-date!
[nltk_data]    | Downloading package omw to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package omw-1.4 is already up-to-date!
[nltk_data]    | Downloading package opinion_lexicon to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package opinion_lexicon is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package panlex_swadesh is already up-to-date!
[nltk_data]    | Downloading package paradigms to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package paradigms is already up-to-date!
[nltk_data]    | Downloading package pe08 to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package pe08 is already up-to-date!
[nltk_data]    | Downloading package perluniprops to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package perluniprops is already up-to-date!
[nltk_data]    | Downloading package pil to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |     Package pil is already up-to-date!
[nltk_data]    | Downloading package pl196x to
[nltk_data]    |       C:\Users\nithi\AppData\Roaming\nltk_data...
```



```
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package toolbox to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package udhr to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package universal_treebanks_v20 is already up-to-
[nltk_data]    |       date!
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package vader_lexicon is already up-to-date!
[nltk_data]    | Downloading package verbnet to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package verbnet is already up-to-date!
[nltk_data]    | Downloading package verbnet3 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package verbnet3 is already up-to-date!
[nltk_data]    | Downloading package webtext to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wmt15_eval to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wmt15_eval is already up-to-date!
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package word2vec_sample is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet2022 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet2022 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package ycoe to
[nltk_data]    |     C:\Users\nithi\AppData\Roaming\nltk_data...
[nltk_data]    |   Package ycoe is already up-to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

```
Out[8]: True
```

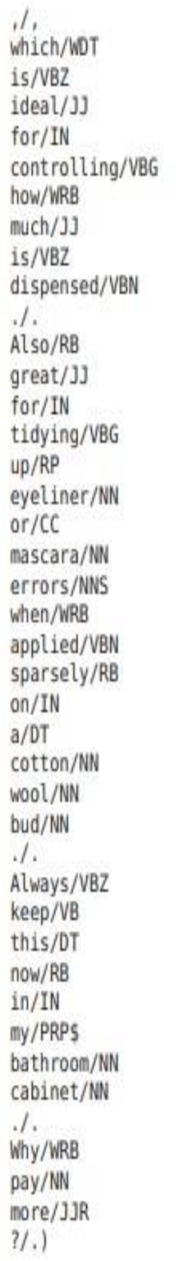
```
In [9]: nltk.download('punkt')
```

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

Out[14]:

In [15]:
```
entities = nltk.chunk.ne_chunk(tagged)
entities.pprint()
```

```
(S
  Have/VBP
  used/VBN
  a/DT
  lot/NN
  of/IN
  make-up/JJ
  brands/NNS
  and/CC
  in/IN
  the/DT
  past/JJ
  bought/VBD
  their/PRP$
  more/RBR
  expensive/JJ
  make-up/NN
  remover/NN
  products/NNS
  ,/,
  but/CC
  none/NN
  have/VBP
  ever/RB
  removed/VBN
  mascara/NN
  ,/,
  eye-shadow/JJ
  ,/,
  foundation/NN
  bases/NNS
  etc/VBP
  as/IN
  gently/RB
  and/CC
  as/RB
  well/RB
  as/IN
  (PERSON Simple/NNP Cleansing/NNP Lotion/NNP)
  ./.
  Its/PRP$
  consistency/NN
  is/VBZ
  not/RB
  runny/JJ
  ,/,
  you/PRP
  have/VBP
  to/TO
  squeeze/VB
  the/DT
  bottle/NN
  appropriately/RB
  onto/IN
  the/DT
  cotton/NN
  wool/NN
```



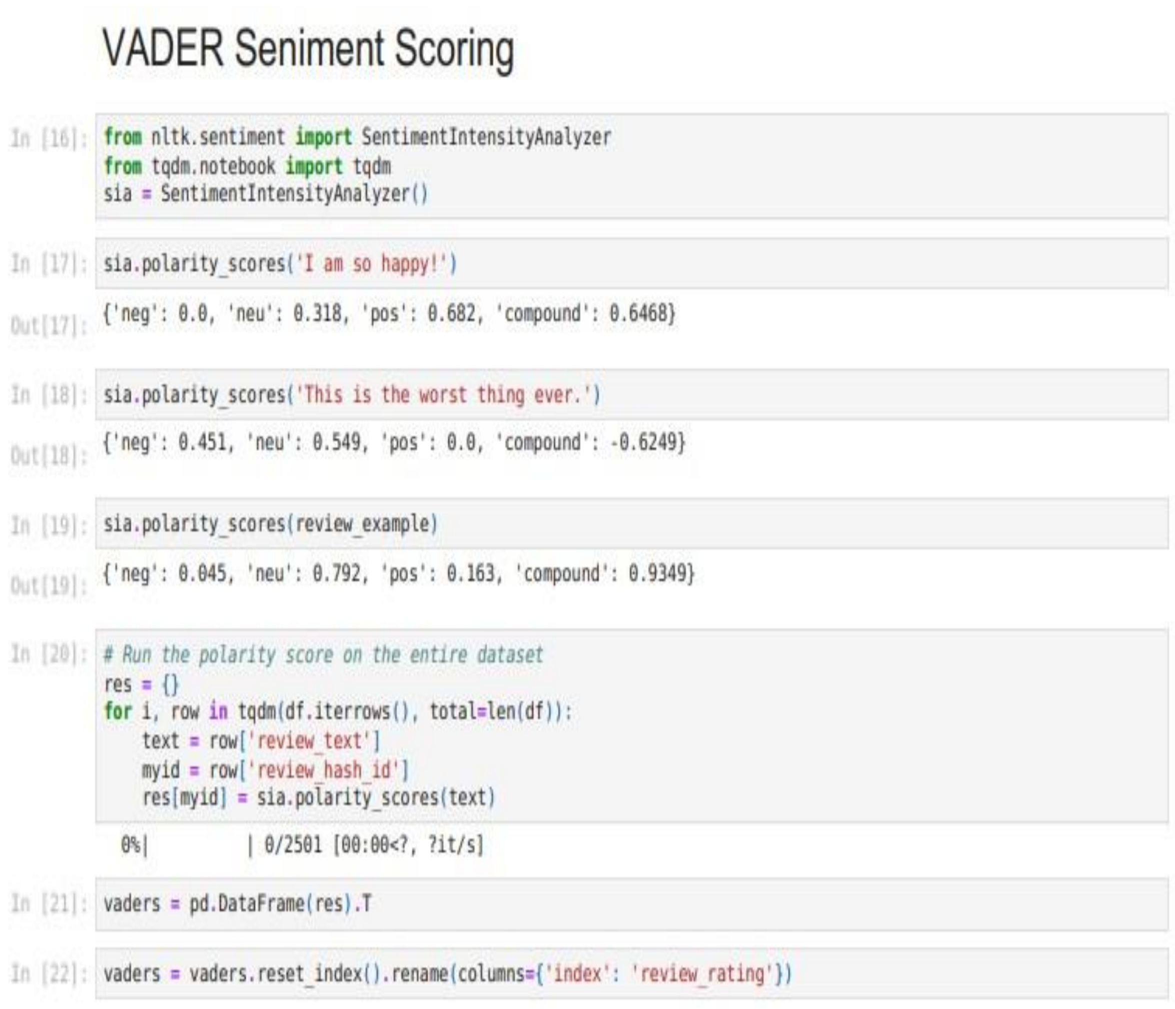
```
,/,
which/WDT
is/VBZ
ideal/JJ
for/IN
controlling/VBG
how/WRB
much/JJ
is/VBZ
dispensed/VBN
./.
Also/RB
great/JJ
for/IN
tidying/VBG
up/RP
eyeliner/NN
or/CC
mascara/NN
errors/NNS
when/WRB
applied/VBN
sparsely/RB
on/IN
a/DT
cotton/NN
wool/NN
bud/NN
./.
Always/VBZ
keep/VB
this/DT
now/RB
in/IN
my/PRP$
bathroom/NN
cabinet/NN
./.
Why/WRB
pay/NN
more/JJR
?/.)
```

A lot of the data that you could be analyzing is unstructured data and contains human-readable text. Before you can analyze that data programmatically, you first need to preprocess it.

In this tutorial, you'll take your first look at the kinds of **text preprocessing** tasks you can do with NLTK so that you'll be ready to apply them in future projects. You'll also see how to do some basic **text analysis** and create **visualizations**.

9.3 VADER Sentiment scoring:

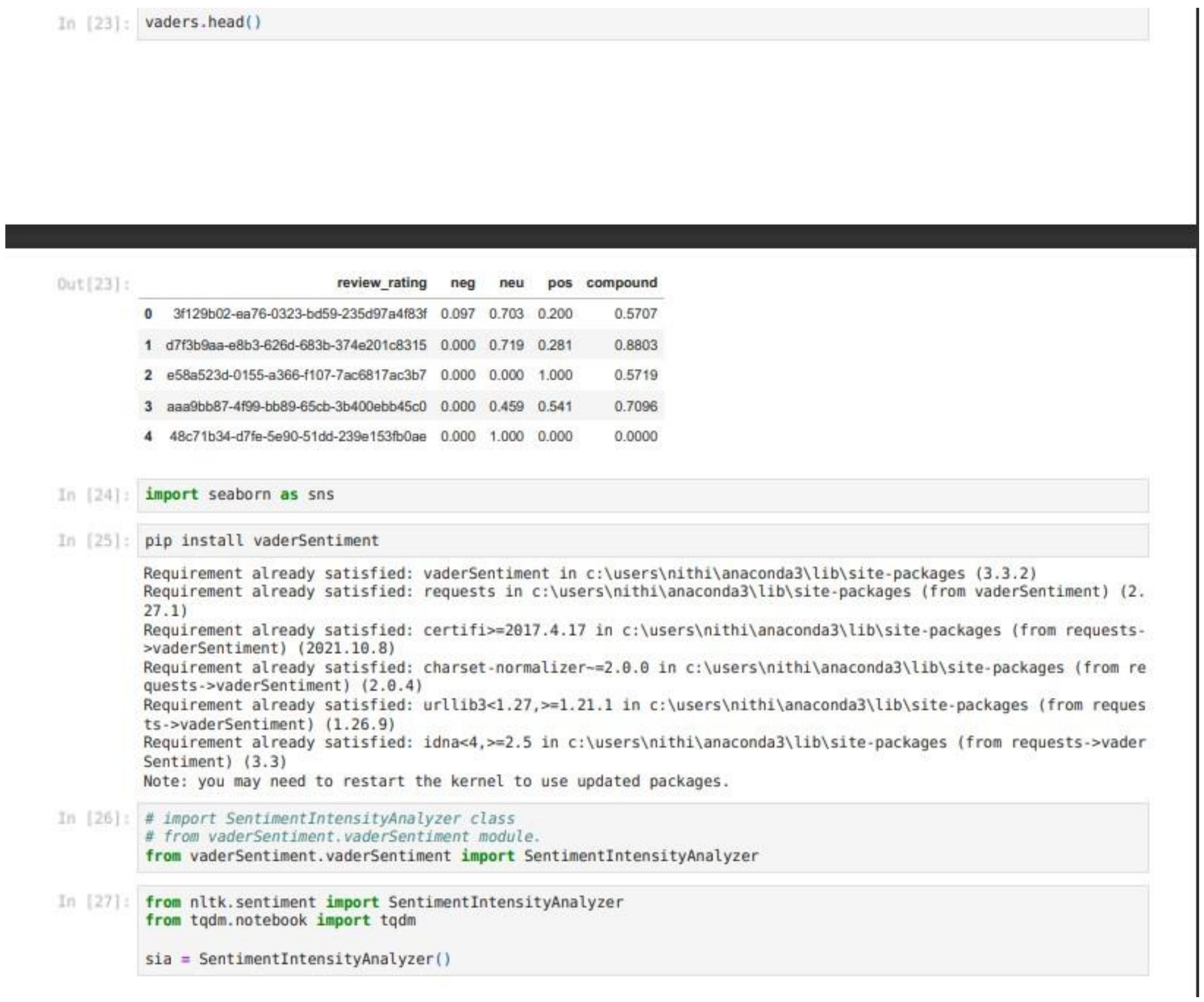
- VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. ○ VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative.
- VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.



VADER Seniment Scoring

```
In [16]: from nltk.sentiment import SentimentIntensityAnalyzer
         from tqdm.notebook import tqdm
         sia = SentimentIntensityAnalyzer()

In [17]: sia.polarity_scores('I am so happy!')

Out[17]: {'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}

In [18]: sia.polarity_scores('This is the worst thing ever.')

Out[18]: {'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}

In [19]: sia.polarity_scores(review_example)

Out[19]: {'neg': 0.045, 'neu': 0.792, 'pos': 0.163, 'compound': 0.9349}

In [20]: # Run the polarity score on the entire dataset
         res = {}
         for i, row in tqdm(df.iterrows(), total=len(df)):
             text = row['review_text']
             myid = row['review_hash_id']
             res[myid] = sia.polarity_scores(text)

           0%|          | 0/2501 [00:00<?, ?it/s]

In [21]: vaders = pd.DataFrame(res).T

In [22]: vaders = vaders.reset_index().rename(columns={'index': 'review_rating'})
```

- VADER sentiment analysis (well, in the Python implementation anyway) returns a sentiment score in the range -1 to 1, from most negative to most positive.
- The sentiment score of a sentence is calculated by summing up the sentiment scores of each VADER-dictionary-listed word in the sentence.



```
In [23]: vaders.head()
```

Out[23]:

	review_rating	neg	neu	pos	compound
0	3f129b02-ea76-0323-bd59-235d97a4f83f	0.097	0.703	0.200	0.5707
1	d7f3b9aa-e8b3-626d-683b-374e201c8315	0.000	0.719	0.281	0.8803
2	e58a523d-0155-a366-f107-7ac6817ac3b7	0.000	0.000	1.000	0.5719
3	aaa9bb87-4f99-bb89-65cb-3b400ebb45c0	0.000	0.459	0.541	0.7096
4	48c71b34-d7fe-5e90-51dd-239e153fb0ae	0.000	1.000	0.000	0.0000

```
In [24]: import seaborn as sns

In [25]: pip install vaderSentiment

Requirement already satisfied: vaderSentiment in c:\users\nithi\anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: requests in c:\users\nithi\anaconda3\lib\site-packages (from vaderSentiment) (2.
27.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\nithi\anaconda3\lib\site-packages (from requests-
>vaderSentiment) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from re
quests->vaderSentiment) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\nithi\anaconda3\lib\site-packages (from reques
ts->vaderSentiment) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\nithi\anaconda3\lib\site-packages (from requests->vader
Sentiment) (3.3)
Note: you may need to restart the kernel to use updated packages.

In [26]: # import SentimentIntensityAnalyzer class
         # from vaderSentiment.vaderSentiment module.
         from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

In [27]: from nltk.sentiment import SentimentIntensityAnalyzer
         from tqdm.notebook import tqdm

         sia = SentimentIntensityAnalyzer()
```

- Text Sentiment Analysis is a really big field with a lot of academic literature behind it. However, its tools really just boil down to two approaches: the lexical approach and the machine learning approach.

- Lexical approaches aim to map words to sentiment by building a lexicon or a ‘dictionary of sentiment.’ We can use this dictionary to assess the sentiment of phrases and sentences, without the need of looking at anything else.

- Sentiment can be categorical — such as {negative, neutral, positive} — or it can be numerical — like a range of intensities or scores. Lexical approaches look at the sentiment category or score of each word in the sentence and decide what the sentiment category or score of the whole sentence is.

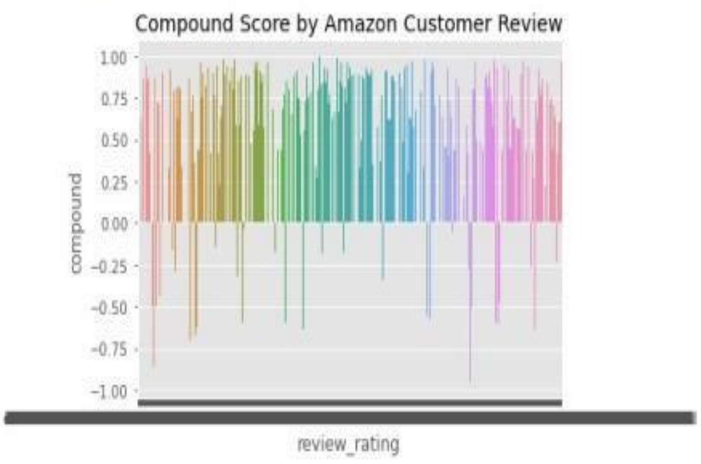
- The power of lexical approaches lies in the fact that we do not need to train a model using labeled data, since we have everything we need to assess the sentiment of sentences in the dictionary of emotions.

9.4 Plot VANDER result:

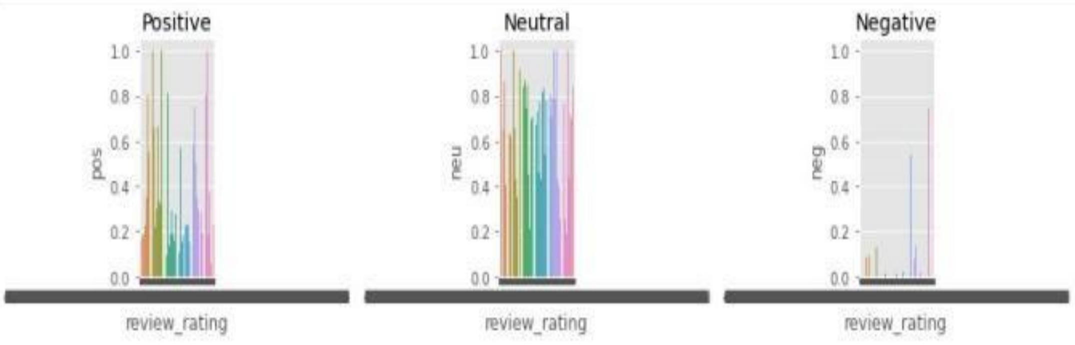
- Vandermonde matrices or polynomials with sentiment analysis.
- Vandermonde matrices and polynomials are mathematical constructs primarily used in numerical analysis and polynomial interpolation.
- While they may not have a direct application in sentiment analysis, you can use Python libraries like NumPy and Matplotlib to visualize sentiment analysis results or represent sentiment data in various ways.

Plot VADER results

```
In [28]: ax = sns.barplot(data=vaders, x='review_rating', y='compound')
         ax.set_title('Compound Score by Amazon Customer Review')
         plt.show()
```



```
In [29]: fig, axs = plt.subplots(1, 3, figsize=(12, 3))
         sns.barplot(data=vaders, x='review_rating', y='pos', ax=axs[0])
         sns.barplot(data=vaders, x='review_rating', y='neu', ax=axs[1])
         sns.barplot(data=vaders, x='review_rating', 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()
```



9.5 Roberta Pretrained Model:

- RoBERTa (A Robustly Optimized BERT Pretraining Approach) is a pretrained language model developed by Facebook AI. It is based on the same architecture as BERT (Bidirectional Encoder Representations from Transformers) but uses a few optimization techniques to improve its performance on various natural language understanding tasks. RoBERTa was trained on a large corpus of text from the internet and has become popular for a wide range of NLP tasks due to its strong performance.

   Here's an overview of RoBERTa's key features and characteristics:

Architecture:

- RoBERTa, like BERT, is a transformer-based model. It consists of multiple layers of self-attention mechanisms and feed-forward neural networks.

Training Data:

- RoBERTa was trained on a massive amount of text data, including text from the internet, books, and articles. It was trained on a larger dataset for longer durations compared to BERT.

Pretraining Tasks:

- RoBERTa uses two main pretraining tasks: masked language modeling (similar to BERT) and next sentence prediction. The model learns to predict masked words in a sentence and to determine whether two sentences are consecutive or not.

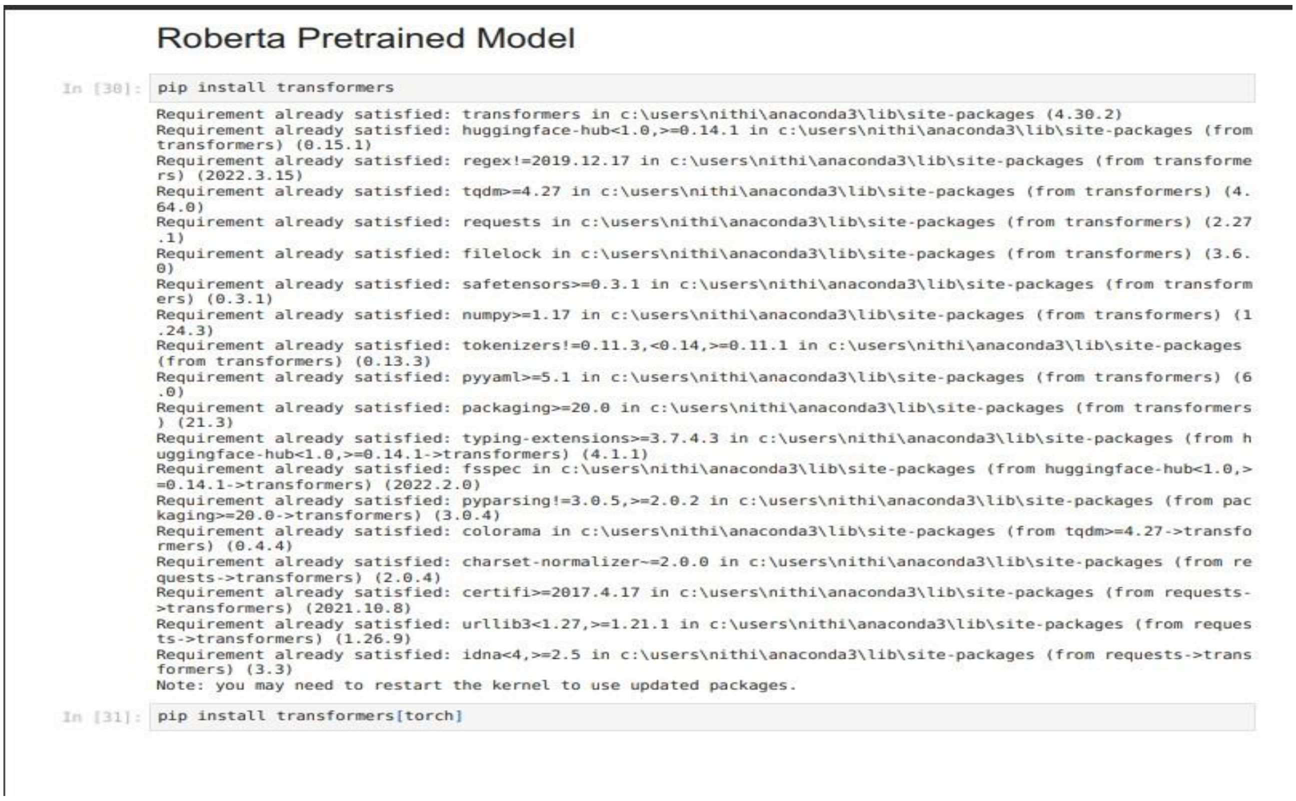
Architecture Tweaks:

- RoBERTa incorporates several architectural changes and training techniques compared to BERT, such as removing the next sentence prediction task, training with dynamic masking, and using larger batch sizes.

Performance:

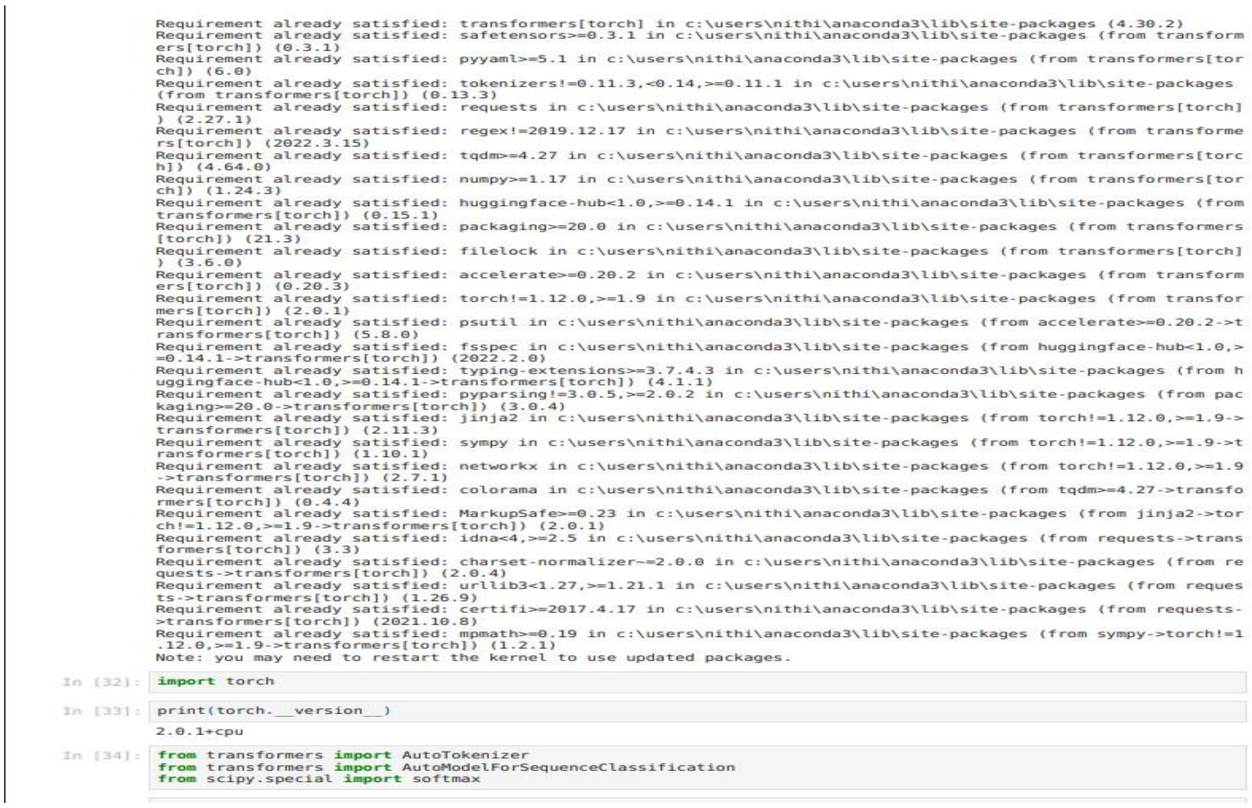
- RoBERTa has achieved state-of-the-art results on a wide range of NLP benchmarks, including tasks like text classification, question-answering, and text generation.

You can use pretrained RoBERTa models for various NLP tasks, such as text classification, sentiment analysis, named entity recognition, and machine translation. Hugging Face's Transformers library is a popular choice for accessing pretrained RoBERTa models in Python.



Roberta Pretrained Model

```
In [30]: pip install transformers

Requirement already satisfied: transformers in c:\users\nithi\anaconda3\lib\site-packages (4.30.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in c:\users\nithi\anaconda3\lib\site-packages (from
transformers) (0.15.1)
Requirement already satisfied: regex!=2019.12.17 in c:\users\nithi\anaconda3\lib\site-packages (from transforme
rs) (2022.3.15)
Requirement already satisfied: tqdm>=4.27 in c:\users\nithi\anaconda3\lib\site-packages (from transformers) (4.
64.0)
Requirement already satisfied: requests in c:\users\nithi\anaconda3\lib\site-packages (from transformers) (2.27
.1)
Requirement already satisfied: filelock in c:\users\nithi\anaconda3\lib\site-packages (from transformers) (3.6.
0)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\nithi\anaconda3\lib\site-packages (from transform
ers) (0.3.1)
Requirement already satisfied: numpy>=1.17 in c:\users\nithi\anaconda3\lib\site-packages (from transformers) (1
.24.3)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\nithi\anaconda3\lib\site-packages
(from transformers) (0.13.3)
Requirement already satisfied: pyyaml>=5.1 in c:\users\nithi\anaconda3\lib\site-packages (from transformers) (6
.0)
Requirement already satisfied: packaging>=20.0 in c:\users\nithi\anaconda3\lib\site-packages (from transformers
) (21.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\nithi\anaconda3\lib\site-packages (from h
uggingface-hub<1.0,>=0.14.1->transformers) (4.1.1)
Requirement already satisfied: fsspec in c:\users\nithi\anaconda3\lib\site-packages (from huggingface-hub<1.0,>
=0.14.1->transformers) (2022.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\nithi\anaconda3\lib\site-packages (from pac
kaging>=20.0->transformers) (3.0.4)
Requirement already satisfied: colorama in c:\users\nithi\anaconda3\lib\site-packages (from tqdm>=4.27->transfo
rmers) (0.4.4)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from re
quests->transformers) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\nithi\anaconda3\lib\site-packages (from requests-
>transformers) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\nithi\anaconda3\lib\site-packages (from reques
ts->transformers) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\nithi\anaconda3\lib\site-packages (from requests->trans
formers) (3.3)
Note: you may need to restart the kernel to use updated packages.

In [31]: pip install transformers[torch]
```



```
Requirement already satisfied: transformers[torch] in c:\users\nithi\anaconda3\lib\site-packages (4.30.2)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\nithi\anaconda3\lib\site-packages (from transform
ers[torch]) (0.3.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\nithi\anaconda3\lib\site-packages (from transformers[tor
ch]) (6.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\nithi\anaconda3\lib\site-packages
(from transformers[torch]) (0.13.3)
Requirement already satisfied: requests in c:\users\nithi\anaconda3\lib\site-packages (from transformers[torch]
) (2.27.1)
Requirement already satisfied: regex!=2019.12.17 in c:\users\nithi\anaconda3\lib\site-packages (from transforme
rs[torch]) (2022.3.15)
Requirement already satisfied: tqdm>=4.27 in c:\users\nithi\anaconda3\lib\site-packages (from transformers[torc
h]) (4.64.0)
Requirement already satisfied: numpy>=1.17 in c:\users\nithi\anaconda3\lib\site-packages (from transformers[tor
ch]) (1.24.3)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in c:\users\nithi\anaconda3\lib\site-packages (from
transformers[torch]) (0.15.1)
Requirement already satisfied: packaging>=20.0 in c:\users\nithi\anaconda3\lib\site-packages (from transformers
[torch]) (21.3)
Requirement already satisfied: filelock in c:\users\nithi\anaconda3\lib\site-packages (from transformers[torch]
) (3.6.0)
Requirement already satisfied: accelerate>=0.20.2 in c:\users\nithi\anaconda3\lib\site-packages (from transform
ers[torch]) (0.20.3)
Requirement already satisfied: torch!=1.12.0,>=1.9 in c:\users\nithi\anaconda3\lib\site-packages (from transfor
mers[torch]) (2.0.1)
Requirement already satisfied: psutil in c:\users\nithi\anaconda3\lib\site-packages (from accelerate>=0.20.2->t
ransformers[torch]) (5.8.0)
Requirement already satisfied: fsspec in c:\users\nithi\anaconda3\lib\site-packages (from huggingface-hub<1.0,>
=0.14.1->transformers[torch]) (2022.2.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\nithi\anaconda3\lib\site-packages (from h
uggingface-hub<1.0,>=0.14.1->transformers[torch]) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\nithi\anaconda3\lib\site-packages (from pac
kaging>=20.0->transformers[torch]) (3.0.4)
Requirement already satisfied: jinja2 in c:\users\nithi\anaconda3\lib\site-packages (from torch!=1.12.0,>=1.9->
transformers[torch]) (2.11.3)
Requirement already satisfied: sympy in c:\users\nithi\anaconda3\lib\site-packages (from torch!=1.12.0,>=1.9->t
ransformers[torch]) (1.10.1)
Requirement already satisfied: networkx in c:\users\nithi\anaconda3\lib\site-packages (from torch!=1.12.0,>=1.9
->transformers[torch]) (2.7.1)
Requirement already satisfied: colorama in c:\users\nithi\anaconda3\lib\site-packages (from tqdm>=4.27->transfo
rmers[torch]) (0.4.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\nithi\anaconda3\lib\site-packages (from jinja2->tor
ch!=1.12.0,>=1.9->transformers[torch]) (2.0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\nithi\anaconda3\lib\site-packages (from requests->trans
formers[torch]) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from re
quests->transformers[torch]) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\nithi\anaconda3\lib\site-packages (from reques
ts->transformers[torch]) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\nithi\anaconda3\lib\site-packages (from requests-
>transformers[torch]) (2021.10.8)
Requirement already satisfied: mpmath>=0.19 in c:\users\nithi\anaconda3\lib\site-packages (from sympy->torch!=1
.12.0,>=1.9->transformers[torch]) (1.2.1)
Note: you may need to restart the kernel to use updated packages.

In [32]: import torch

In [33]: print(torch.__version__)

         2.0.1+cpu

In [34]: from transformers import AutoTokenizer
         from transformers import AutoModelForSequenceClassification
         from scipy.special import softmax
```

```
In [35]: pip install emoji
```

```
Requirement already satisfied: emoji in c:\users\nithi\anaconda3\lib\site-packages (2.6.0)
Note: you may need to restart the kernel to use updated packages.
```

```
Requirement already satisfied: tensorflow in c:\users\nithi\anaconda3\lib\site-packages (2.13.0)
Requirement already satisfied: tensorflow-intel==2.13.0 in c:\users\nithi\anaconda3\lib\site-packages (from ten
sorflow) (2.13.0)
Requirement already satisfied: setuptools in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel=
=2.13.0->tensorflow) (61.2.0)
Collecting numpy<=1.24.3,>=1.22
  Using cached numpy-1.24.3-cp39-cp39-win_amd64.whl (14.9 MB)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (4.23.4)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-int
el==2.13.0->tensorflow) (1.12.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-in
tel==2.13.0->tensorflow) (1.4.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\nithi\anaconda3\lib\site-packag
es (from tensorflow-intel==2.13.0->tensorflow) (0.31.0)
Requirement already satisfied: flatbuffers>=23.1.21 in c:\users\nithi\anaconda3\lib\site-packages (from tensorf
low-intel==2.13.0->tensorflow) (23.5.26)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (1.56.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in c:\users\nithi\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (4.1.1)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-
intel==2.13.0->tensorflow) (2.3.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in c:\users\nithi\anaconda3\lib\site-packages
(from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-
intel==2.13.0->tensorflow) (16.0.6)
Requirement already satisfied: six>=1.12.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel
==2.13.0->tensorflow) (1.16.0)
Note: you may need to restart the kernel to use updated packages.Requirement already satisfied: astunparse>=1.6
.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel==2.13.0->tensorflow) (1.6.3)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel
==2.13.0->tensorflow) (3.6.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in c:\users\nithi\anaconda3\lib\site-packages (from tens
orflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: packaging in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel==
2.13.0->tensorflow) (21.3)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow
-intel==2.13.0->tensorflow) (3.3.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (2.13.1)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (0.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\nithi\anaconda3\lib\site-packages (from astunpars
e>=1.6.0->tensorflow-intel==2.13.0->tensorflow) (0.37.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\nithi\anaconda3\lib\site-packages (from tensor
board<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.22.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\nithi\anaconda3\lib\site-packa
ges (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.7.1)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\nithi\anaconda3\lib\site-packages (fr
om tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.0.0)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorboard<
2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.0.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorbo
ard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.27.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\nithi\anaconda3\lib\site-packages (from tensorboard<
2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.3.4)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from googl
e-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\nithi\anaconda3\lib\site-packages (from google
-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\nithi\anaconda3\lib\site-packages (from google-auth<3,
>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (4.7.2)
Requirement already satisfied: urllib3<2.0 in c:\users\nithi\anaconda3\lib\site-packages (from google-auth<3,>=
1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.26.9)
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This
behaviour is the source of the following dependency conflicts.
daal4py 2021.5.0 requires daal==2021.4.0, which is not installed.
tensorflow-intel 2.13.0 requires numpy<=1.24.3,>=1.22, but you have numpy 1.25.1 which is incompatible.
scipy 1.7.3 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.25.1 which is incompatible.
numba 0.55.1 requires numpy<1.22,>=1.18, but you have numpy 1.25.1 which is incompatible.
```

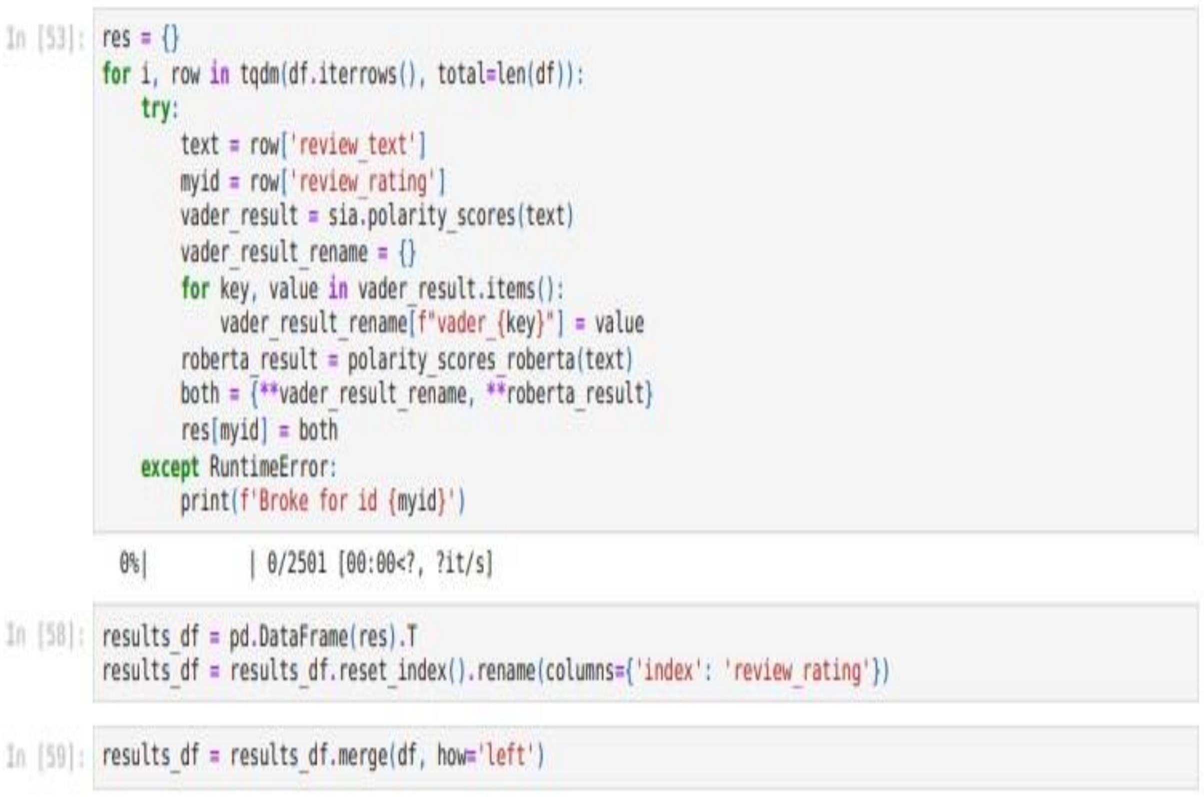
```
In [40]: import numpy as np
         import pandas as pd
         import sys
         import emoji
         import math
         import pickle
         from time import time
         from nltk.corpus import stopwords
         from nltk.stem import SnowballStemmer
         from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
         from sklearn.pipeline import Pipeline
         from sklearn.metrics import confusion_matrix
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.naive_bayes import GaussianNB, MultinomialNB
         from sklearn.linear_model import SGDClassifier
         from sklearn.cluster import KMeans
         from sklearn import tree
         from sklearn import svm
         from contextlib import contextmanager
         from sklearn.feature_extraction.text import TfidfVectorizer
         from scipy.sparse import hstack
         import time
         import re
         import string
         from scipy.sparse import csr_matrix
         from sklearn.preprocessing import MinMaxScaler
         import lightgbm as lgb
         from sklearn.model_selection import KFold
         from sklearn.metrics import roc_auc_score
         import gc
         from collections import defaultdict
         import os
         import psutil
         import os
```

```
In [41]: pip install keras
```

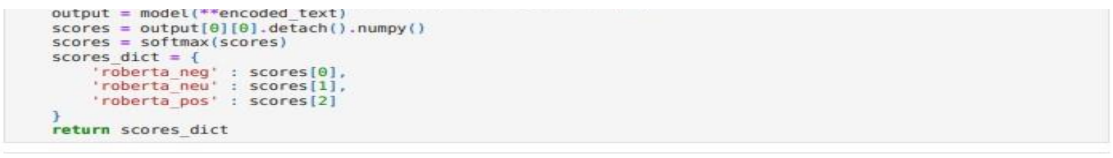
```
Requirement already satisfied: keras in c:\users\nithi\anaconda3\lib\site-packages (2.13.1)
Note: you may need to restart the kernel to use updated packages.
```

```
In [42]: pip install tensorflow
```

```
Requirement already satisfied: flatbuffers>=23.1.21 in c:\users\nithi\anaconda3\lib\site-packages (from tensorf
low-intel==2.13.0->tensorflow) (23.5.26)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\nithi\anaconda3\lib\site-packag
es (from tensorflow-intel==2.13.0->tensorflow) (0.31.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in c:\users\nithi\anaconda3\lib\site-packages
(from tensorflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-
intel==2.13.0->tensorflow) (16.0.6)
Requirement already satisfied: keras<2.14,>=2.13.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (2.13.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-in
tel==2.13.0->tensorflow) (1.4.0)
Requirement already satisfied: packaging in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-intel==
2.13.0->tensorflow) (21.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (1.56.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-int
el==2.13.0->tensorflow) (1.12.1)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow
-intel==2.13.0->tensorflow) (3.3.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorflow-
intel==2.13.0->tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in c:\users\nithi\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.13.0->tensorflow) (4.1.1)
Requirement already satisfied: tensorboard<2.14,>=2.13 in c:\users\nithi\anaconda3\lib\site-packages (from tens
orflow-intel==2.13.0->tensorflow) (2.13.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.13.0->tensorflow) (0.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\nithi\anaconda3\lib\site-packages (from astunpars
e>=1.6.0->tensorflow-intel==2.13.0->tensorflow) (0.37.1)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\nithi\anaconda3\lib\site-packages (fr
om tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.0.0)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\nithi\anaconda3\lib\site-packages (from tensorboard<
2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.0.3)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\nithi\anaconda3\lib\site-packa
ges (from tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.7.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\nithi\anaconda3\lib\site-packages (from tensor
board<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.22.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\nithi\anaconda3\lib\site-packages (from tensorboard<
2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.3.4)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\nithi\anaconda3\lib\site-packages (from tensorbo
ard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.27.1)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\nithi\anaconda3\lib\site-packages (from google-auth<3,
>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (4.7.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\nithi\anaconda3\lib\site-packages (from google
-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.2.8)
Requirement already satisfied: urllib3<2.0 in c:\users\nithi\anaconda3\lib\site-packages (from google-auth<3,>=
1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.26.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from googl
e-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (4.2.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\nithi\anaconda3\lib\site-packages (from goo
gle-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\nithi\anaconda3\lib\site-packages (from pyasn1-
modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.4.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\nithi\anaconda3\lib\site-packages (from requests<
3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\nithi\anaconda3\lib\site-packages (from requests<3,>=2.
21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from re
quests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.0.4)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\nithi\anaconda3\lib\site-packages (from requests-oau
thlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3
.2.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\nithi\anaconda3\lib\site-packages (from pac
kaging->tensorflow-intel==2.13.0->tensorflow) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```



```
In [53]: res = {}
         for i, row in tqdm(df.iterrows(), total=len(df)):
             try:
                 text = row['review_text']
                 myid = row['review_rating']
                 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%|          | 0/2501 [00:00<?, ?it/s]

In [58]: results_df = pd.DataFrame(res).T
         results_df = results_df.reset_index().rename(columns={'index': 'review_rating'})

In [59]: results_df = results_df.merge(df, how='left')
```



```
    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
```

9.6 Compare scores between models :

Comparing the performance of different models in sentiment analysis involves several key aspects and metrics. Sentiment analysis, also known as opinion mining, aims to determine the sentiment or emotional tone expressed in text data, typically classified as positive, negative, or neutral. Here are some common metrics and considerations for comparing the scores of different sentiment analysis models:

Accuracy:

- Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of correctly predicted sentiment labels to the total number of samples. However, accuracy alone may not be sufficient for imbalanced datasets.

Precision and Recall:

- Precision measures the proportion of true positive predictions out of all positive predictions made by the model. Recall (or sensitivity) measures the proportion of true positive predictions out of all actual positive instances. Models with higher precision and recall are better at correctly identifying the sentiment of interest.

F1-Score:

- The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance, especially when dealing with imbalanced datasets. A higher F1-score indicates a better-performing model.

Confusion Matrix:

- The confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, allowing you to assess the model's performance for each sentiment class individually.

ROC Curve and AUC:

- These metrics are commonly used for binary sentiment analysis tasks. The ROC (Receiver Operating Characteristic) curve plots the true positive rate (TPR) against the false positive rate (FPR) at various thresholds, while the AUC (Area Under the Curve) summarizes the ROC curve's performance. Higher AUC values indicate better model discrimination.

Macro and Micro-Averaged Metrics:

- For multi-class sentiment analysis, you can use macro-averaged and microaveraged precision, recall, and F1-score to assess the model's performance across multiple sentiment classes.

Cross-Validation:

- To ensure that the model's performance is not due to randomness in data splitting, you should perform k-fold cross-validation and evaluate the model's performance across multiple folds.

Domain and Data Specificity:

- Consider the domain and data used for training and testing. Some models might perform better in specific domains or on certain types of text data (e.g., social media, product reviews, news articles).

Computational Resources:

- Evaluate the computational resources required by each model. Some models may be more resource-intensive, making them less practical for real-time applications or deployment on resource-constrained devices.

Interpretability:

- Assess how interpretable and explainable the model's predictions are. In some cases, simpler models with good performance may be preferred over complex black-box models.

Fine-Tuning:

- Some models may require extensive hyperparameter tuning or fine-tuning on domain-specific data to achieve optimal performance. Take into account the effort and resources required for this process.

Real-World Testing:

- Finally, consider conducting real-world testing or A/B testing to assess how the models perform in actual applications and user scenarios.

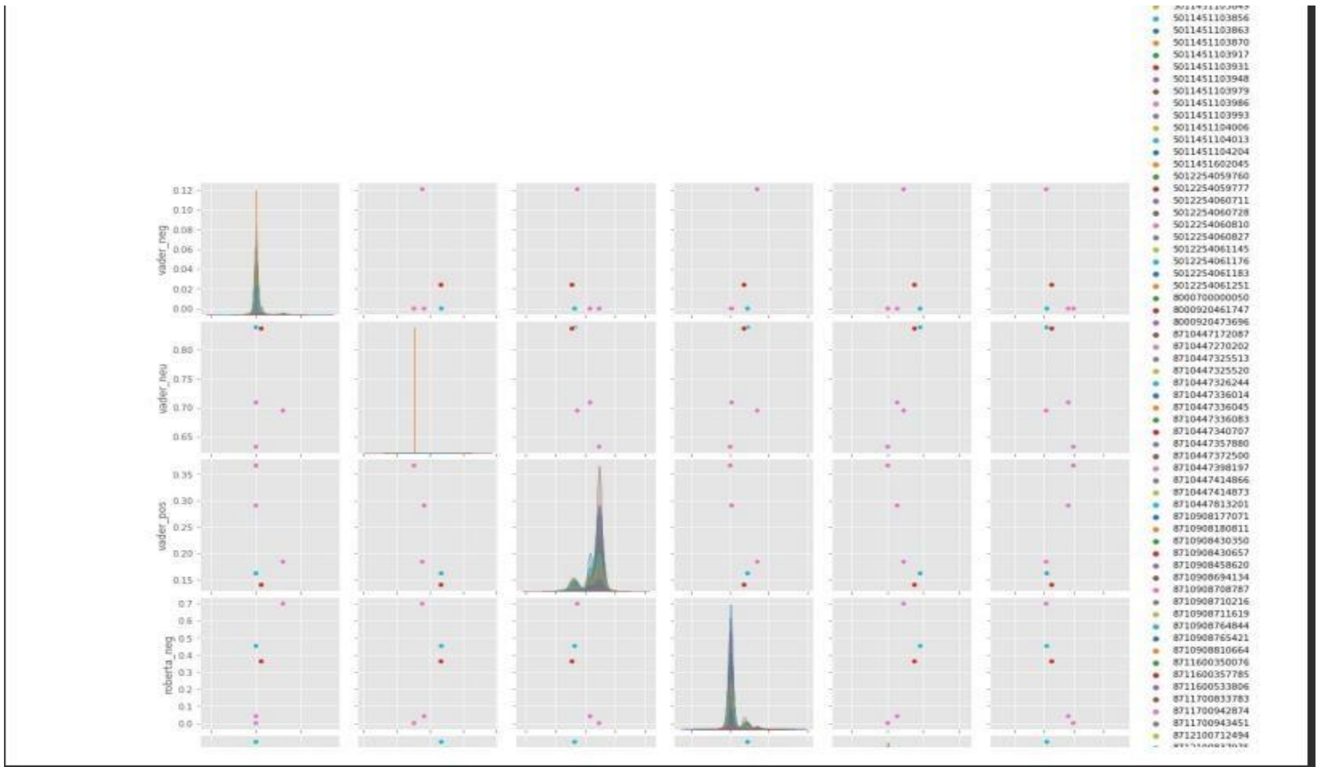
Compare Scores between models

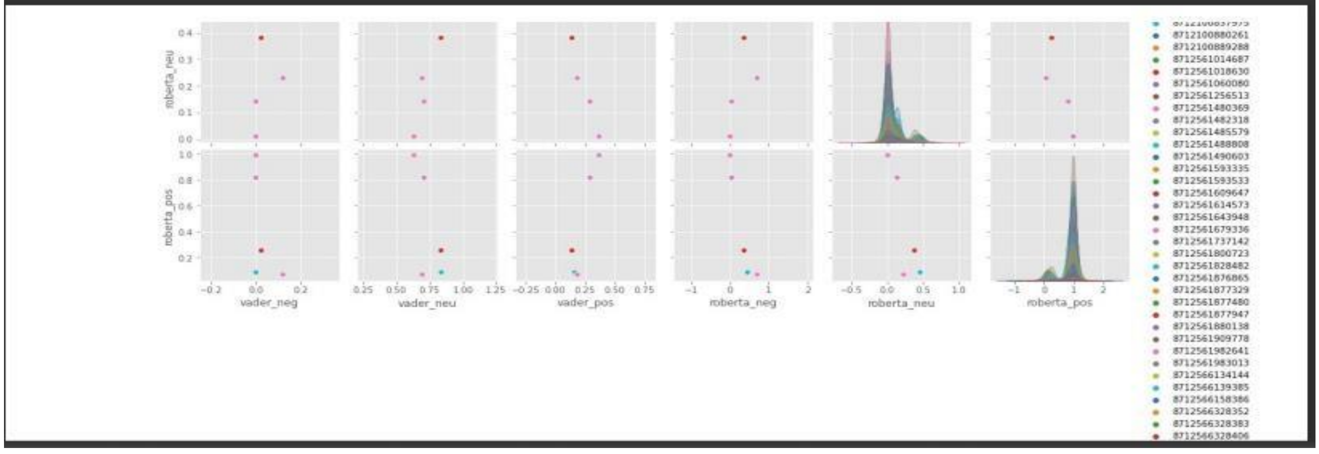
```
In [60]: results_df.columns
```

```
Out[60]: Index(['review_rating', 'vader_neg', 'vader_neu', 'vader_pos',
                'vader_compound', 'roberta_neg', 'roberta_neu', 'roberta_pos',
                'report_date', 'online_store', 'upc', 'retailer_product_code', 'brand',
                'category', 'sub_category', 'product_description', 'review_date',
                'review_title', 'review_text', 'is_competitor', 'manufacturer',
                'market', 'matched_keywords', 'time_of_publication', 'url',
                'review_type', 'parent_review', 'manufacturers_response', 'dimension1',
                'dimension2', 'dimension3', 'dimension4', 'dimension5', 'dimension6',
                'dimension7', 'dimension8', 'verified_purchase', 'helpful_review_count',
                'review_hash_id'],
               dtype='object')
```

```
In [64]: sns.pairplot(data=results_df,
                      vars=['vader_neg', 'vader_neu', 'vader_pos',
                            'roberta_neg', 'roberta_neu', 'roberta_pos'],
                     hue='upc',
                     palette='tab10')
```

```
Out[64]: <seaborn.axisgrid.PairGrid at 0x24494bdd7f0>
```





```
In [ ]: plt.show()
```

Review Examples

```
In [65]: results_df.query('upc == 42182658') \
             .sort_values('roberta_neu', ascending=False)['review_text'].values[0]
```

```
Out[65]: 'I have used Vaseline on my lips for well over 20 years and loved it. I buy a large jar of plain Vaseline and u
         se on my lips. I loved that there is really no smell or taste to it. BUT last week, I bought a new jar, put som
         e on, and it has a taste that I can only explain of tasting like black oil that goes into a car. It is terrible
         ! I2019ve tried using it for a week now, and just can2019t stand it any longer. I2019m guessing there2019s been
         a formula or ingredient supplier change? I hope you will bring back the old version. Very disappointed!'
```

```
In [67]: results_df.query('upc ==42182658 ') \
             .sort_values('vader_pos', ascending=False)['review_text'].values[0]
```

```
Out[67]: 'I came across an article to use Vaseline on acne scars and irritated skin, so I tried it out. Never would I th
         ink to do this, I would think that this would clog my pores. However it did the opposite, I have been using thi
         s for a month now and it is life changing! My scars have visibly faded! I have gone to so many dermatologist an
         d couldn2019t find any product that worked this well. This is a game changer my friends, please try it out!'
```

```
In [68]: results_df.query('upc== 8722700750185') \
             .sort_values('roberta_neg', ascending=False)['review_text'].values[0]
```

```
Out[68]: 'Like to make my own soups & casseroles. This product does very well for that.'
```

```
In [69]: results_df.query('upc == 8722700750185') \
             .sort_values('vader_neg', ascending=False)['review_text'].values[0]
```

```
Out[69]: 'Like to make my own soups & casseroles. This product does very well for that.'
```

9.7 The Transformers Pipeline:

The Transformers pipeline in sentiment analysis refers to the use of Transformerbased models, such as BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), and others, for performing sentiment analysis tasks. Transformers have revolutionized natural language processing (NLP) tasks, including sentiment analysis, due to their ability to capture contextual information effectively.

- Here is a step-by-step overview of a typical sentiment analysis pipeline using Transformers:

Data Collection:

- Gather a dataset of text documents or social media posts that you want to analyze for sentiment. This dataset should be labeled with sentiment categories (e.g., positive, negative, neutral).

Data Preprocessing:

- Preprocess the text data to make it suitable for input into the Transformer model. This preprocessing may include tasks such as tokenization (breaking text into words or subword units), lowercasing, and removing special characters or stop words.

Model Selection:

- Choose a pre-trained Transformer model that has been fine-tuned for sentiment analysis. Models like BERT, RoBERTa, or DistilBERT are popular choices. These models have already been trained on massive amounts of text data and can be fine-tuned on your specific sentiment analysis task.

Fine-Tuning:

- Fine-tune the selected Transformer model on your sentiment analysis dataset. During fine-tuning, the model learns to map input text to sentiment labels by adjusting its parameters based on your training data.

Model Training:

- Train the fine-tuned model using your preprocessed dataset. This involves feeding the text data into the model and updating its weights using techniques like backpropagation and gradient descent.

Evaluation:

- After training, evaluate the model's performance on a separate validation or test dataset to assess its accuracy, precision, recall, F1-score, and other relevant metrics.

Inference:

- Once the model is trained and evaluated, you can use it to make sentiment predictions on new, unseen text data. Simply input the text into the model, and it will provide predictions of the sentiment (e.g., positive, negative, neutral).

Post-processing:

- Depending on your specific application, you may need to post-process the model's predictions. For example, you might convert numerical scores into sentiment labels or apply additional logic to filter or categorize sentiments further.

Deployment:

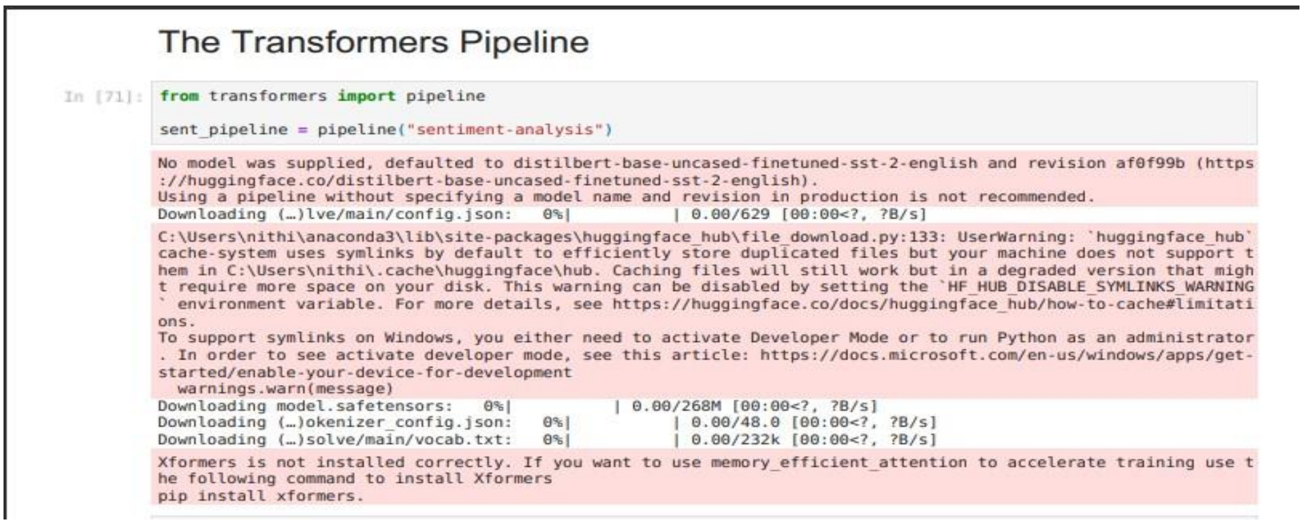
- Integrate the trained model into your application or workflow. You can deploy it as a web service, API, or as part of a larger NLP system.

Monitoring and Maintenance:

- Continuously monitor the model's performance in the production environment and periodically retrain it with new data to keep it up-to-date and accurate.

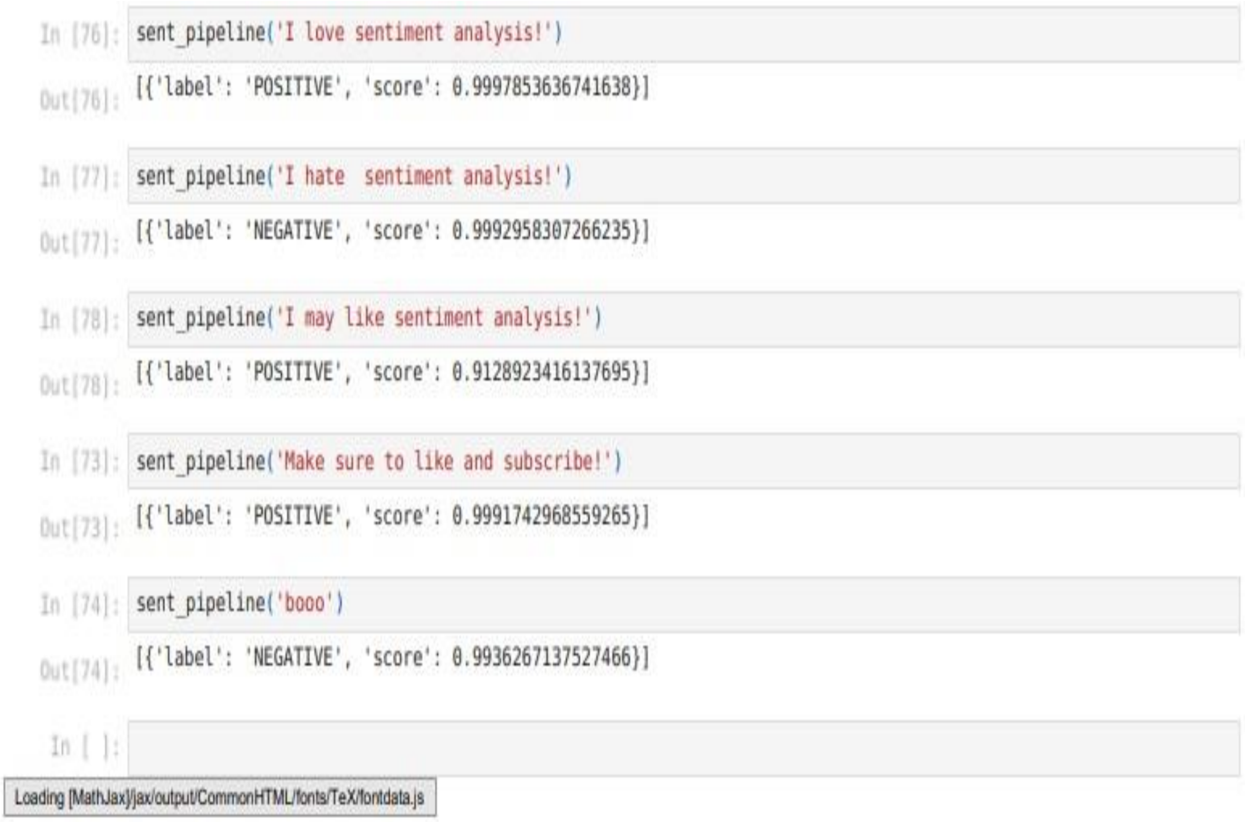
Transformers have greatly improved the accuracy of sentiment analysis by allowing models to understand the context and nuances of text.

They can handle long-range dependencies and capture subtle sentiment cues in language, making them a popular choice for sentiment analysis tasks in various domains, including social media monitoring, customer feedback analysis, and product reviews analysis.



The Transformers Pipeline

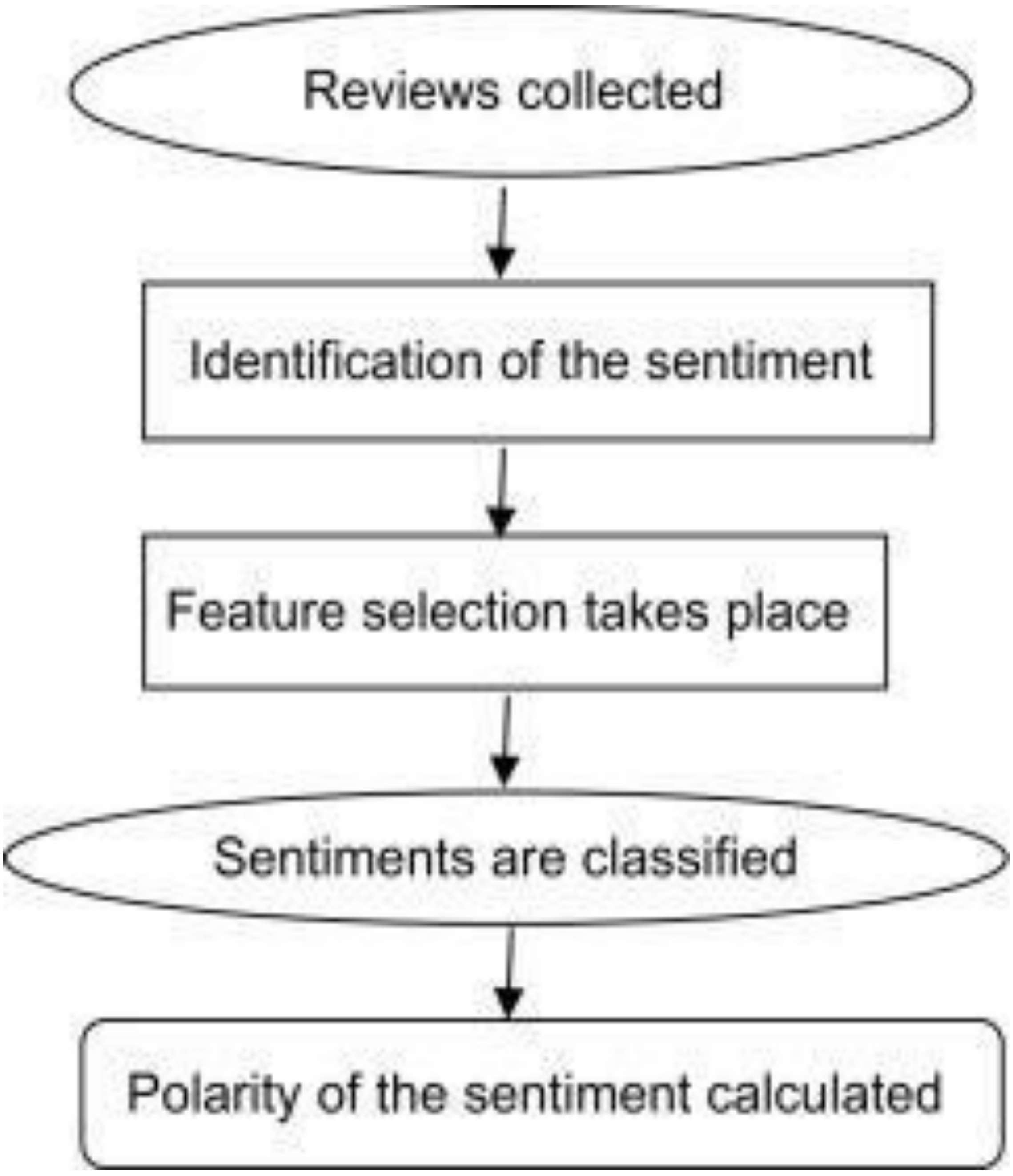
```
In [71]: from transformers import pipeline

         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 (…)lve/main/config.json:   0%|          | 0.00/629 [00:00<?, ?B/s]
C:\Users\nithi\anaconda3\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 t
hem in C:\Users\nithi\.cache\huggingface\hub. Caching files will still work but in a degraded version that migh
t 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#limitati
ons.
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 (…)okenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
Xformers is not installed correctly. If you want to use memory_efficient_attention to accelerate training use t
he following command to install Xformers
pip install xformers.
```



```
In [76]: sent_pipeline('I love sentiment analysis!')

Out[76]: [{'label': 'POSITIVE', 'score': 0.9997853636741638}]

In [77]: sent_pipeline('I hate  sentiment analysis!')

Out[77]: [{'label': 'NEGATIVE', 'score': 0.9992958307266235}]

In [78]: sent_pipeline('I may like sentiment analysis!')

Out[78]: [{'label': 'POSITIVE', 'score': 0.9128923416137695}]

In [73]: sent_pipeline('Make sure to like and subscribe!')

Out[73]: [{'label': 'POSITIVE', 'score': 0.9991742968559265}]

In [74]: sent_pipeline('booo')

Out[74]: [{'label': 'NEGATIVE', 'score': 0.9936267137527466}]

In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

10.DataFlow Diagram



Reviews collected
Identification of the sentiment
Feature selection takes place
Sentiments are classified
Polarity of the sentiment calculated

11.CONCLUSION

- Conclusion, sentiment analysis is a powerful tool with a wide range of applications across various industries, from marketing and customer service to finance and politics.

- Through the analysis of text data, sentiment analysis enables us to gain valuable insights into the emotional tone and opinions of individuals or groups.

- It is important to acknowledge that sentiment analysis is not without its challenges. Natural language is complex, and sentiment can be expressed in subtle and nuanced ways. Furthermore, sentiment analysis models may struggle with sarcasm, irony, and cultural context, making them imperfect but continually improving tools.

- As technology advances, sentiment analysis will likely become even more accurate and valuable in understanding human emotions and behaviors. However, ethical considerations must always be at the forefront of its development and implementation, ensuring that privacy and bias concerns are addressed to create more equitable and responsible solutions.

- In summary, sentiment analysis has the potential to revolutionize decision making processes, enhance customer experiences, and provide valuable insights into public opinion. As it evolves, it will continue to play a crucial role in shaping various aspects of our society and economy.

Reference:

https://nithin162002.github.io/sentiment_Analysis/sentiment%20analysis.pdf

THANK YOU