

# Predictive Power of Social Media Data

Devayani Jannagantti Shivaprakash

Submitted for the Degree of Master of Science in  
Data Science and Analytics



Department of Computer Science  
Royal Holloway University of London  
Egham, Surrey TW20 0EX, UK

December 18, 2022



## **Declaration**

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

**Word Count:** 9940

**Student Name:** Devayani Jannagantti Shivaprakash

**Date of Submission:** 19<sup>th</sup> December 2022

**Signature:** Devayani Jannagantti Shivaprakash

## Abstract

Sentiment Analysis is an important process of computation conducted by a lot of organizations to understand the public's opinion of their brand or products. This helps them understand market opportunities, feedback, and areas of improvement. It is an easy process that can be done manually but can become relatively hard or impossible if the size of the data becomes large, such is the case when dealing with tweets made on the social media platform Twitter. Big companies can even have to process millions of tweets directed at them using hashtags. In such scenario "Sentiment Analysis" needs to be automated on a large-scale data analysis platform. This project aims at conducting sentiment analysis of tweets using two tools based on lexicon-based approach: TextBlob, and Vader Sentiment. They will be used to predict sentiments of various Tweets extracted from a secondary source, Kaggle, into three labels: Positive, Neutral, and Negative based on a sentiment score. The data is loaded and processed using Pandas API on Spark which is a python library that allows using the user-friendly feature rich interface of Pandas to run workloads on the large-scale data analysis platform Spark. Functions written in python language can be used to utilize TextBlob and Vader and passed to Pandas API which will estimate and return polarity and compound scores which can be used as sentiment scores which is an automated tuning performed to estimate thresholds for predicting Positive, Neutral, and Negative labels from the sentiment score. This tuning is performed on the training set and validated using the validation set. The various scores resulting from TextBlob, and Vader are then compared to analyse their differences. Finally, TextBlob provides a subjectivity score which is then analysed to see how tweets can be filtered on the basis of how subjective or objective they are. The companion code is organized into multiple files according to what purpose they serve.

## **Acknowledgment**

The project I completed was one of the most unforgettable experiences in my life. As a result of my study, I now have better clarity and a broader range of analytical skills. I am grateful for everyone who have helped me during the project, and I would like to thank few people.

I specifically want to thank Prof. Daniel O’Keeffe my supervisor, without whom I would not have been able to complete this project. With his expert advice and proper guidance, I was able to complete my project.

I also want to thank Prof. Julien Lange for introducing me to the concepts of Large-Scale Data Analytics.

I would like to thank my parents, my sister without their love and support all my effort would be useless.

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	Background .....	1
1.2	Aims and Objectives .....	2
1.3	Scope .....	2
<b>2</b>	<b>Background Research.....</b>	<b>3</b>
2.1	Themes .....	4
2.1.1	Theoretical Aspect .....	4
2.1.2	Lexicon Based Approach.....	5
2.1.3	Vader Sentiment .....	5
2.1.4	Textblob Sentiment.....	6
2.2	Challenges and Disadvantages for Lexicon Based Approach .....	6
2.3	Literature Gap .....	6
2.4	Pandas API on Pyspark .....	7
<b>3</b>	<b>Methodology.....</b>	<b>9</b>
3.1	Approach of the Research .....	9
<b>4</b>	<b>Implementation Details and Running the Code.....</b>	<b>10</b>
4.1	Code Structure .....	10
4.2	Environment.....	10
4.3	Installing Dependencies .....	11
4.4	Importing the Dataset .....	12
4.5	Configuring plotly for VSCode.....	13
4.6	Suppressing Warnings.....	13
<b>5</b>	<b>Data Analysis.....</b>	<b>14</b>
5.1	Data Source and Description .....	14
5.2	Repeated Tweet IDs .....	14
5.3	Analysing the Sentiments.....	16
5.4	Analysing Entities .....	17
5.5	Overview of Sentiments .....	18
5.6	Irrelevant Data .....	19
5.7	Missing Values.....	19
<b>6</b>	<b>Parameter Tuning .....</b>	<b>22</b>

6.1	TextBlob .....	22
6.1.1	Filtering Data .....	22
6.1.2	Tuning Parameters .....	22
6.1.3	Best Parameters for TextBlob .....	23
6.2	Vader Sentiment .....	24
6.2.1	Filtering Data .....	24
6.2.2	Tuning Parameters .....	24
6.2.3	Best Parameters For Vader Sentiment .....	25
<b>7</b>	<b>Results and Analysis .....</b>	<b>26</b>
7.1.1	Predict Sentiment using Vader .....	28
7.1.2	Irrelevant Sentiments Vader .....	28
7.1.3	Predicting Sentiments using TextBlob .....	29
7.1.4	Irrelevant Sentiments TextBlob .....	29
7.1.5	Comparing Vader Compound with TextBlob Polarity .....	29
7.1.6	Working with Subjectivity .....	30
<b>8</b>	<b>Conclusion .....</b>	<b>33</b>
8.1	Further scope .....	33
<b>9</b>	<b>Self-Assessment .....</b>	<b>34</b>
<b>10</b>	<b>Professional Issues: The ethics of Data Collection and Storage .....</b>	<b>35</b>
10.1	Data Collection .....	35
10.2	Data Storage .....	35
	<b>References .....</b>	<b>36</b>

## Table of Figures

Figure 2-1 Importing Pandas API on Spark .....	8
Figure 3-1 Creating a Python Virtual Environment .....	11
Figure 3-2 Activating env in VSCode .....	11
Figure 3-3 Installing Dependencies manually .....	11
Figure 3-4 Installing dependencies through requirements.txt .....	12
Figure 3-5 Loading Dataset into a Pandas API on Pyspark DataFrame .....	12
Figure 3-6 Configuring plotly for VSCode .....	13
Figure 4-1 Distribution of number of times Tweet_ID is repeated .....	15
Figure 4-2 Tweet content of a sampled Tweet_ID .....	15
Figure 4-3 Pie Chart of Distribution of Sentiment Labels from the Dataset .....	16
Figure 4-4 Total Number of Unique IDs .....	17
Figure 4-5 Pie Chart of Distribution of Entities from the Data Set .....	17
Figure 4-6 Bar graph representing the number of individual sentiments for various entities(Scrollable in the notebook) .....	18
Figure 4-7 Calculating percentage of Irrelevant Sentiments .....	19
Figure 4-8 Finding Missing Values .....	20
Figure 4-9 Analysing Missing Value .....	20
Figure 5-1 Dropping Unnecessary records .....	22
Figure 5-2 Best Thresholds for TextBlob .....	23
Figure 5-3 Dropping Unnecessary Records .....	24
Figure 5-4 Best Thresholds for Vader Sentiment .....	25
Figure 6-1 Estimating Vader Compound .....	26
Figure 6-2 Verifying if all compound scores for a Tweet_ID are same .....	26
Figure 6-3 Estimating TextBlob Polarity .....	27
Figure 6-4 Estimating TextBlob Subjectivity .....	27
Figure 6-5 Verifying if all polarity scores for a Tweet_ID .....	28
Figure 6-6 Predicting Sentiments using Vader .....	28
Figure 6-7 Accuracy for Vader .....	28
Figure 6-8 Predicting Sentiments using TextBlob .....	29
Figure 6-9 Accuracy for TextBlob .....	29
Figure 6-10 Comparing Compound and Polarity scores .....	30
Figure 6-11 Differences between predictions from Vader and TextBlob ...	30
Figure 6-12 Histogram of distribution of subjectivity .....	31
Figure 6-13 Working with Subjectivity .....	31
Figure 6-14 Subjectivity Distribution for a sample entity Amazon .....	32



# 1 Introduction

Sentiment Analysis is also known as “opinion mining” or “emotion Artificial Intelligence”, it includes few methodical identification, extraction, evaluation and analysis of emotional responses and subjective data and comes under the domain of *Natural Language Processing* or (NLP). Social media has grown to be the most popular digital medium for people to communicate their own ideas and opinions on a daily basis. Furthermore, the predictive value of social media has been demonstrated by utilizing state of the art data analytics and programming languages such as Python, pyspark and so on. Additionally, data from TWITTER a social media platform is utilised to discover distinct types of sentiments that are often used by users to represent their thoughts and ideas on the online platform.

Sentiment Analysis is one of the methods used in "natural language processing" (NLP) to identify the kind of data and whether it is positive, neutral, or negative, and is commonly referred to as opinion mining. Here, a sentimental analysis of the Twitter dataset will be carried out in order to track the user's sentiments on the brand or tweet. Analysis of social media sentiment is helpful to solve the issues regarding their business, based on the feedback of the users and then to engage the users with the service more efficiently. The sentiment analysis on “Twitter” will allow the company in keeping a track of the feedback of their users that is being said by them about their services on the social media platform. In addition to that, this will be beneficial as it allows them to detect the users that are spreading negative influences and then take relevant actions on them.

## 1.1 Background

Every human activity is influenced by views since they have a significant impact on how we behave. Considering the opinions of others before making a decision mostly leads to better results. Businesses and organizations need to constantly learn what the public thinks of their products and services. People use a variety of online platforms for social media interaction, such as web-based social networking sites like Facebook and Twitter. Moreover, people from every ethnicity, sexual orientation, nation, and socioeconomic status use the internet to exchange experiences and opinions about almost every aspect of their life. In addition, many people use informal business sites to record options, convey feelings and get insights into their daily lives in addition to sending message, blogging, or leaving comments on corporate website (Ilhan, Kübler and Pauwels, 2018).

Twitter is a microblogging platform where users create “tweets” that are sent to their followers or to another user. In a given month in 2016, Twitter had more than 313 million dynamic users, including 100 million users every day. There are many different client origins, with 77% of them located outside of the United States and they generate well over 500 million tweets daily. In terms of overall activity in 2017, the Twitter website ranked 12th and responded to far more than 15

billion API requests daily. However, more than a million other websites use Twitter resources and APIs. Due to this significant development, Twitter has recently come under intense scrutiny because Tweets typically reflect opinions of users on divisive topics. Sentiment analysis and opinion mining are extremely difficult jobs in the social media setting, because of the vast amounts of data that both humans and algorithms produce.

In the past, two politicians were compared using real-time Twitter data that was taken from Twitter and retrieved using the Twitter-streaming API (API). Moreover, negative and positive scores were discovered using SentiWordNet and WordNet, two sentiment analysers. Furthermore, Negation handlings as well as word sequence disambiguation (WSD) were utilized to improve the accuracy of the model. The dataset used is Twitter dataset, and the data was retrieved from the online source Kaggle to make predictions about tweets on various entities. After gathering data for this project, automatic buzzer recognition was used to weed out pointless tweets, and the tweets were then sentimentally examined by dividing each tweet into many sub-tweets. Then, it used positive tweets related to each candidate to predict the outcome of the election and calculated sentiment polarity. Finally, it used mean absolute error (MAE) to evaluate the accuracy of the prediction and claimed that the Twitter-based prediction has been 0.61% more accurate than the similar type of surveys that are conducted traditionally.

## **1.2 Aims and Objectives**

- Perform Data Analysis on the dataset
- Estimate polarity and subjectivity scores using TextBlob
- Predict sentiments using polarity as sentiment scores
- Filter and analyse tweets using subjectivity scores
- Estimate compound scores using Vader Sentiment
- Predict sentiments using compound scores as sentiment scores
- Compare results from TextBlob and Vader

## **1.3 Scope**

There are numerous approaches that can be used to gain insights from tweets posted on Twitter. The scope of this project is restricted to performing sentiment analysis by labelling tweets with Positive, Neutral, and Negative sentiments using lexicon-based tools like TextBlob and Vader. For getting better accuracy for predictions the thresholds to classify tweets will be automatically tuned, and results are analysed.

## 2 Background Research

This section talks about the previous study about social media, sentiment analysis using pandas in Pyspark, lexicon-based approaches.

The social media platform makes it possible for businesses to collect data widely and cheaply. Facebook is a good example where more than two billion accounts are created every second. People use these platforms to express their emotion, thoughts and insights about an organization or product. In order to keep track of this predicting power of social media was introduced. According to (Raviya and Vennila 2021), people are engaged in virtual socialism as a result of social media's spectacular growth, which has resulted in a massive volume of textual and visual information. When the contents of status updates, tweets, and shared articles and tweets are taken into consideration, likes on other postings are a representation of people's online behavior. It is becoming impossible to predict a user's personality based on their digital footprints. Using user-generated textual material to reflect the personality on social media may be beneficial in a profile-based strategy. Researchers classified the qualities using machine learning methodologies such as supervised/unsupervised learning approaches and classification algorithms in order to predict personality traits computationally. Studies say that they have found a connection between personality and Facebook users, which is called as social networking services.

Sentiment Analysis is a component of natural language processing, a combination of text analysis, computational linguistics and much more towards systematic examine affective states from subjective data such as tweets. As cited by (Elzayady et al. 2018), the set of variables was found to have the greatest values for precision, recall and F1score after all the models were evaluated. As a result, this model was employed in the final trend analysis. Sentiment Analysis, commonly referred to as opinion mining and emotion AI, it gauges the strength of readers' views (Positive/Negative/Neutral) in unstructured text. As illustrated by (Alexopoulos et al. 2020), big Data refers to the enormous volume of both aggregation and analysis input which is huge and challenging to handle it using conventional methods. It has grown widely and with the use of this technique, it is now feasible to make sense of huge social media data in order to properly comprehend social interactions, product marketing efforts, and political events and reach more accurate conclusions.

The variety of uses for sentiment analysis has aided in its development. Sentiment analysis tools have made it possible to make sense of large amounts of social media data in order to fully understand social engagements, product marketing campaigns, and political events and make better judgments. As stated by (Jalilet al. 2020), the four articles chosen for this Special Issue address the use of sentiment analysis to enhance health insurance, comprehend AIDS patients, profile online shoppers, and identify cyber aggression. The diversity of viewpoints used in the articles throughout this special issue demonstrates the popularity and vitality of this subject. A fair overview of several of the major areas of study in this discipline is also provided in this Issue of Biomedical Engineering. Stock market

predictions can be done by using Twitter sentiment analysis, by finding the correlation between “public sentiment” and “market sentiment” according to (Mittal & Goel, 2012)

## 2.1 Themes

### 2.1.1 Theoretical Aspect

Three most popular techniques in Sentiment Analysis includes the hybrid methods, machine learning based techniques, and lexicon-based approach. As per the view of (Sasikanthet *al.*2020), these strategies, all entail a number of computational stages. The lexicon-based method establishes sentiment to divide posts on social media platforms into three groups: negative, neutral, positive. The ML-based technique, in contrast, is equally well-liked by academics, particularly when a lot of labelled information samples are accessible and employed in such a manner of supervised learning. This project is mainly focused on lexicon-based approach.

#### Text mining

Text mining is used to extract meaningful data from unrecognized textual material. As per the view of (Pradhan *et al.* 2022), there are two main stages to this procedure: The evaluation process describes the procedure of organizing material utilizing linguistic analytical techniques, including identifying words, and phrases, as well as the relationships between them in the grammar. It uses several techniques:

- The process of identifying the actual language used to write a text content is known as language identification.
- Tokenization seems to be the method of transforming a set of characters, including exclamation points, into words as well as phrases.
- Implementing filters, including deleting hollow words, is what filtering entails.
- Lemmatization seems to be the process of combining together some word's various conjugated forms by eliminating its plural, gender, as well as conjugations. Afterward, researchers examine them collectively.
- The technique of looking for text objects that fall into categories like people, dates, and locations is known as "named-entity recognition".

#### Opinion mining

Opinion mining involves employing text assessment to determine a text's sentimental orientation (positive level, negative level, or neutral level). As per the view of (Hasan *et al.* 2019), this is referred to a variety of catch-all titles, including sentiment analysis partiality and analysis of the position. Among its prospective uses is to monitor and comprehend public sentiment through social media

regarding a specific subject in a variety of fields, including politics, health, as well as marketing. For example, prospective customers can base their choice on ratings and reviews.

### **2.1.2 Lexicon Based Approach**

Lexicon based approach is a technique of semantic analysis, it is sorted into groups based on their perceived worth. Vocabulary-based approaches need a predetermined lexicon, whereas machine learning approaches automatically classify reviews and need training data. According to Sayed et al. (2022), the lexicon is a collection of terminology that is specific to a field or language. In order to overcome the drawbacks of these various strategies, Hybrid techniques were taken into consideration. One of the methodologies utilized in sentiment analysis is the lexicon-based approach. Using the sentiment orientation of lexicons, it determines the sentiment inclinations of the entire document or group of sentences. As stated by (Ortega et al. 2019) Negative, positive, or neutral semantic orientations are possible. Both manually and automatically generating the dictionaries of lexicons are options.

### **2.1.3 Vader Sentiment**

VADER (Valence Aware Dictionary for Sentiment Reasoning), a model for text sentiment analysis, is sensitive to both the polarity (positive/negative/neutral) and intensity (strong) of emotion. It can be used on unlabeled text data or dataset, and it is included in NLTK package. The VADER sentimental analysis uses a lexicon that converts lexical data into sentiment scores, which measures the intensity of an emotion. The text's sentiment score can be obtained by adding each word's intensity. Let's take a simple example from the website (Beri, 2020) to explain about positive, negative, and neutral words 'love,' 'enjoy,' 'glad,' and 'like' all express a positive sentiment. Also, VADER understands the core meaning of these terms, such as 'not', 'cannot' as a negative sentiment.

According to (Hutto, 2021) the positive, negative, and neutral scores are the ratios are the text proportions that fall into each category so that these should all sum up to 1 or close to it with float operation. To analyze the context and presentation on how the sentiment is conveyed for a given sentence these are the most useful matrices. For example, various writing styles can tell the positive and negative sentiment with varying proportions of neutral text. Some writing styles may reflect a preference for strongly flavored rhetoric, whereas other styles may use a large amount of neutral text while conveying a similar overall (compound) sentiment. Each word's valence score is added together, modified in accordance with the guidelines, and then normalized to fall between most extreme negative and most extreme positive to get the compound score. If you're looking for a single unidimensional measure of sentiment for a particular text, this metric is the most helpful. Can you only use compound scores? Answer is yes, because the other scores tell the context and presentation of the sentence but compound talks about the overall sentiment of the sentence.

#### **2.1.4 Textblob Sentiment**

According to (Shah, 2020) in Natural Language Processing, Textblob is a python library, and it actively uses NLTK (Natural Language Toolkit) to get its tasks. NLTK is a library which helps with a lot of lexicon resources and helps users to with classification, categorization, and other tasks. Textblob is an easy library which helps with complex analysis and operations on textual data.

A sentiment is defined by its sematic orientation and by the intensity of each word in the sentence in lexicon-based approach, it is a dictionary where it maps words to a sentiment score. A sentence is formed by of words, in here individual scores are assigned to all the words and the final sentiment is calculated by performing a pooling operation, one of them is by taking the average of all sentiments.

In Textblob the focus is on returning the polarity and subjectivity of a tweet or sentence. In polarity the lower polarity leads to the negative sentiment and higher polarity leads to positive sentiment. Few examples of semantic labels in textblob are detailed analysis, emoticons, exclamation points, emotions and so on. In subjectivity, if a sentence or a tweet contains a personal information rather than facts then the particular tweet has higher subjectivity. To calculate score of subjectivity, textblob also uses intensity and depending on the intensity it can be determined if a word can modify the next word. Adverbs in English can be taken as a good example here because they are used as modifiers.

### **2.2 Challenges and Disadvantages for Lexicon Based Approach**

The recommended lexicon-based approach is straightforward and powerful along with some shortcomings. According to (Ortega et al. 2019), lexicon-based approach was initially created for the sentiment classification of social media posts in a quick text message on social networking sites, where users typically exchange a brief opinion about the subject at hand. A lengthy phrase like publications and articles that cover a wide range of subjects, and they are concurrently mentioned. It deserves further investigation, including the extraction of topics, linked entities, and prior to performing sentiment analysis. Furthermore, the lexicon that can be employed strongly influences the proposed approach. As mentioned in the first phrase, the lexicon must be appropriately selected given the strongest sentiment on SentiWordNet nonetheless, Liu lexicon would be the one approach that performs the finest on the preview of sentiment analysis.

### **2.3 Literature Gap**

The most frequently used popular technique for this is “sentiment analysis”. It is classified as a computerized assessment that extracts views on the object of interest from the available content. Current methodologies for sentiment analysis capture views from material that is expressed in clear and syntactically acceptable language very effectively. However, there have been limitations found in the efficiency of sentiment analysis algorithms when dealing with informal data. These uses have led to a recent boom in industrial activity. As stated by (Kılınc

2019), applications for sentiment analysis have essentially appeared in every conceivable industry, from consumer goods to services, health, and financial services, to social gatherings and political elections. The key to understanding user needs, expectations, and customer base is now sentiment analysis.

Users of social media can add comments using slang, encompasses symbols, idioms, incorrect word usage, and caustic formulations. Social media data also suffer from the dimension issue, or high dimension nature of the data, which necessitates current understanding and feature extraction, improving classification accuracy. This paper provides a thorough description of sentiment analysis methodology based on current research. These papers serve as the foundation for identifying and discussing the key prospects for sentiment analysis heading forward. There are many uses for sentiment analysis on social networks like Twitter and Facebook, where it has developed into a potent tool for understanding user behaviours.

While studying the relevant papers on Sentiment Analysis on Twitter, it said that “VADER” Sentiment Analysis performs better than “Textblob” because emoticons may modify the meaning of a sentence. This makes “VADER” Sentiment Analysis more effective for articles from social networks and online sources. This is taken into consideration by “VADER” together with language, emphasis, and word choice in the context of the sentence. But in this project, both “VADER” and “Textblob” works equally good on the dataset used. As per the view of (Ortega *et al.*2019), the computations and data processing used in this study were carried out using a machine learning method using the Spark model on the Google cloud or any suitable platform. Few academics have studied sentiment analysis in a distributed large dataset, compared to the majority of stock prediction studies. In sentiment classification, algorithms like Logistic Regression and Naive Bayes are effective as well.

It has become vital to focus on improving the current sentiment analysis technologies in order to deal with the faulty and indirect languages used by netizens. As mentioned by (Sujatha and Radha 2021), the traditional sentiment analysis methods have made some progress in this area by using the right context data. However, the issues in natural language processing (NLP) are impeding sentiment analysis's effectiveness and precision. Deep learning models have been shown to offer a plausible solution to NLP's problems in recent years.

## **2.4 Pandas API on Pyspark**

Pandas API on Pyspark is an API that has been made available since the Apache Spark™ 3.2 release (Kwon, 2021). Pandas is a powerful library that is used by almost all data scientists working with python. It is easy to use and boosts a lot of helpful features for data analysis, pre-processing and so on. One of the core requirements of this project is to utilize a large-scale data engineering tool like Apache Spark. In this project Pandas API on Spark is used instead of using Apache directly because to utilize the benefits of Spark with the ease to use feature rich interface provided by the Pandas python package.

Pandas is the most common data analysis and manipulation tool preferred by data scientists when working with the Python Programming Language. Pandas provides a lot of useful features, but its performance is limited as it runs on a single machine. Pandas API on Spark allows Pandas to scale to multiple nodes by utilizing the distributed functionality provided by Spark while also improving single machine performance. This allows data scientists/analysts to learn only a single library for data analysis and manipulation, while enjoying the performance of big data processing tools.

This API is not only good for panda's users but also for developers that utilize PySpark because a lot of functionalities that are not easy to do PySpark directly are made easy using Pandas API on Spark. Some examples are plotting data directly from a PySpark data frame (Databricks, 2022) and analysing intermediate results in between tasks. Pandas API on Spark automatically creates optimized plans to map pandas to pyspark functionality. Developers can easily switch from Pandas to Pandas API on Spark simply by changing imports as described below:

```
1 # import pandas as pd
2 import pyspark.pandas as pd
```

Python

While coding this in pandas API on Spark it does not make the simple trick because it is not that effective nor provides the features provided by the pandas. During the development of the companion code for this project one such incompatibility was faced. The function named `read_tables` in the panda's library is used to read and return a Dataframe from a .txt file with data structured in a tabular format, by the function of the same name in Pandas API on Spark reads a Spark table and returns a Dataframe. Another issue with Pandas API on Spark is that it throws a lot of warnings which are difficult to suppress. A list of supported pandas API are provided on the official documentation (spark, 2022). To avoid such issues the companion code utilizes both Pandas and Pandas API on Spark while prioritizing the latter. Hence, both the libraries are imported as follows:

```
import pyspark.pandas as ps
import pandas as pd
```

Python

**Figure 2-1 Importing Pandas API on Spark**



## 3 Methodology

### 3.1 Approach of the Research

The approach of any form of research plays a very crucial role in setting off the project in accordance with all the objectives. There exist three fundamental approaches that form the very basis of this section and they are abductive approach, the inductive approach, and the deductive approach. In this project deductive approach is implemented. The respective method gets utilized when required for the purpose of gathering and analysing all the data in this regard. Methodology is a section tends to cater to the aforementioned factors in relation to the concerning requirements.

The “abductive approach” is the corresponding research with numerous “puzzles” along with “facts of surprise” and the respective procedure is premised upon the associated explanation for that matter. As the researchers come across an “empirical phenomenon” that could not be elucidated by the present set of theories, the two above-mentioned factors come up. The people concerned with the research aim to find the most plausible elucidation out of all the alternatives to elaborate the puzzles as well as surprising facts. In this context, both forms of reasoning such as “cognitive”, and “numerical” could very well be combined for the purpose of elaboration.

The second approach is “inductive approach”, it starts by gathering a significant quality of data that is related to the topic. A substantial quantity of the data gets collected and the researchers stops the process for the objective of getting an overall picture of the collected data. At this step the researchers try to find a pattern within the data, this helps them to construct a patient theory that would be able to describe all the patterns discovered. In this approach an array of observations is taken to execute the required process. The observations take the form of a general array of propositions regarding the experiences. Essentially, the movement happens from data to the theory, or rather from the specific to the common.

The last one and the approach followed in this project is “deductive approach”. Deductive approach is the reverse of inductive approach. It pertains to a “logical approach”, here the progression takes place from the general ideas to specific conclusions. Often it gets contrasted with the inductive approach, the inception takes the form of specific observations along with generating a general set of conclusions. This research is also utilized to conduct research in accordance with this topic.

## 4 Implementation Details and Running the Code

This section describes the implementation details of the code along with steps required to run it.

### 4.1 Code Structure

The code is structured into four files to facilitate organization, modularity, and to save computational time. Everything is implemented in the form of jupyter notebooks. These four notebooks are listed below in the sequence that they are recommended to be viewed. A detailed description of implementation of these files is present in the subsequent sections

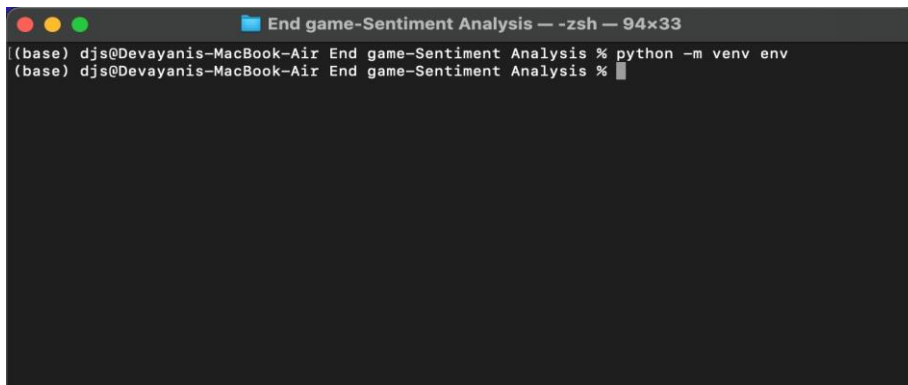
1. **data-analysis.ipynb**: Includes code related to data analysis of the dataset. It is discussed in detail in the Data Analysis Section.
2. **textblob\_tuning.ipynb**: Includes code related to finding the optimal threshold for predicting sentiments from polarity scores. It is discussed in detail in the Parameter Tuning Section.
3. **vader\_tuning.ipynb**: Includes code related to finding the optimal threshold for predicting sentiments from compound scores. It is discussed in detail in the Parameter Tuning Section.
4. **sentiment-analysis.ipynb**: Includes code related to performing sentiment analysis, analysing results, and exploring subjectivity(TextBlob). It is discussed in detail in the Results and Analysis section.

### 4.2 Environment

Google Collab and Visual Studio Code was the preferred editor of choice for developing and running the project. It has numerous features related to developing python code which makes it more convenient than working with a jupyter notebook in a browser.

Due to lack of computational resources the code was written on a personal MacOS and executed on an ubuntu server. The code is written in such a manner that it works seamlessly on cross-platforms.

The project was run in a python virtual environment rather than using Anaconda. It made things simple and easy while avoiding a learning curve. A virtual environment can be created easily by using the following command in the project directory.

A terminal window titled "End game-Sentiment Analysis -- zsh -- 94x33". The prompt is "(base) djs@Devayanis-MacBook-Air End game-Sentiment Analysis %". The command "python -m venv env" has been entered, and the prompt is now "(base) djs@Devayanis-MacBook-Air End game-Sentiment Analysis %".

```
(base) djs@Devayanis-MacBook-Air End game-Sentiment Analysis % python -m venv env
(base) djs@Devayanis-MacBook-Air End game-Sentiment Analysis %
```

Figure 4-1 Creating a Python Virtual Environment

This command creates a virtual environment in a folder named *env* in the current working directory. This environment is automatically picked up by Visual Studio Code but can be manually activated by running the following command:

The environment can be activated in the jupyter notebook by selecting the drop-down menu at the top right of the editor as shown below.

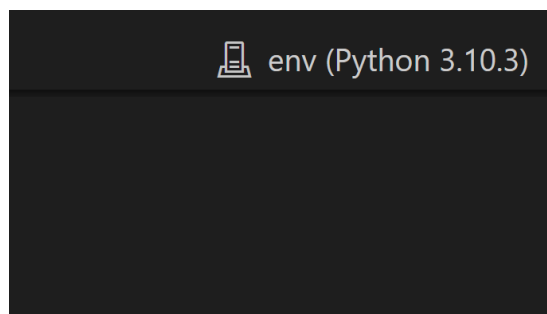
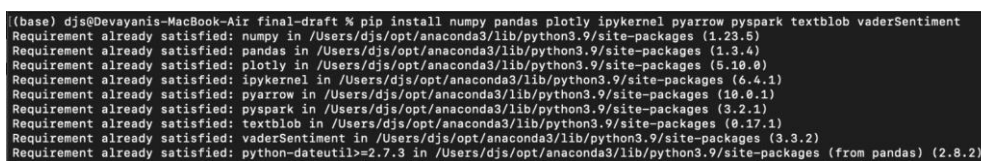


Figure 4-2 Activating env in VSCode

## 4.3 Installing Dependencies

Few third-party libraries were utilised for this project and required to run the project. These dependencies can be installed in multiple ways. This section describes two recommended ways of doing so.

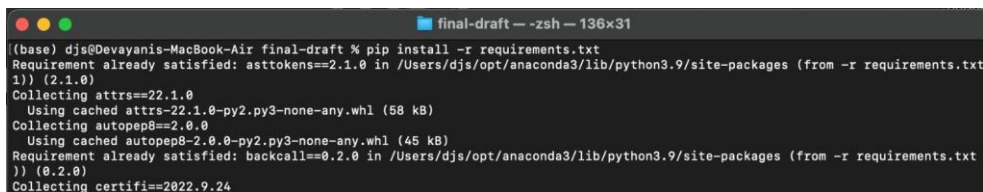
The first way is to install the dependencies manually by running the following command while the virtual environment is activated.

A terminal window showing the command "pip install numpy pandas plotly ipykernel pyarrow pyspark textblob vaderSentiment" being executed. The output shows that all requirements are already satisfied for the specified packages and versions.

```
(base) djs@Devayanis-MacBook-Air final-draft % pip install numpy pandas plotly ipykernel pyarrow pyspark textblob vaderSentiment
Requirement already satisfied: numpy in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (1.23.5)
Requirement already satisfied: pandas in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (1.3.4)
Requirement already satisfied: plotly in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (5.10.0)
Requirement already satisfied: ipykernel in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (6.4.1)
Requirement already satisfied: pyarrow in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (10.0.1)
Requirement already satisfied: pyspark in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (3.2.1)
Requirement already satisfied: textblob in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (0.17.1)
Requirement already satisfied: vaderSentiment in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (3.3.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (from pandas) (2.8.2)
```

Figure 4-3 Installing Dependencies manually

The second way is to install using requirements.txt file which will install all of the above dependencies while maintaining the exact versions used for this project.



```
(base) djs@Devayanis-MacBook-Air final-draft % pip install -r requirements.txt
Requirement already satisfied: asttokens==2.1.0 in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (from -r requirements.txt
1)) (2.1.0)
Collecting attrs==22.1.0
  Using cached attrs-22.1.0-py2.py3-none-any.whl (58 kB)
Collecting autopep8==2.0.0
  Using cached autopep8-2.0.0-py2.py3-none-any.whl (45 kB)
Requirement already satisfied: backcall==0.2.0 in /Users/djs/opt/anaconda3/lib/python3.9/site-packages (from -r requirements.txt
1)) (0.2.0)
Collecting certifi==2022.9.24
```

Figure 4-4 Installing dependencies through requirements.txt

The first way was followed while developing the project [reference: (Sloria, 2013) (cjhutto, 2014)], and the second way is followed while transferring code between the machines.

Apart from the python related dependencies a Java Runtime Environment is also required to run the project. Openjdk11 was installed for this project, and the required steps vary from operating systems and package managers. Hence, the steps have not been mentioned in this section.

## 4.4 Importing the Dataset

Importing the dataset is a common step that is carried out in all the files. Pandas API on Spark is used to process the dataset. Unfortunately, it lacked necessary features to import the Twitter Sentiment Analysis dataset. This is because the *Tweet\_Content* field of the dataset sometimes span over multiple lines. The `read_csv` method could not properly load the dataset either from Pandas API on Spark as well as vanilla pandas. The `read_table` method of vanilla pandas did the trick whereas this function was different in Pandas API on Spark. In the latter this function corresponds to reading data from a spark table.

To solve this issue vanilla pandas was used to load the dataset which was then passed on to the pyspark api as shown below:



```
import pyspark.pandas as ps
import pandas as pd

pdf_train = pd.read_table("./twitter_training.csv", names=names, index_col="Tweet_ID", sep=",")
pdf_valid = pd.read_table("./twitter_validation.csv", names=names, index_col="Tweet_ID", sep=",")


df = ps.concat([
    ps.from_pandas(pdf_train),
    ps.from_pandas(pdf_valid)
])
```

Figure 4-5 Loading Dataset into a Pandas API on Pyspark DataFrame

## 4.5 Configuring plotly for VSCode

Pandas API on Pyspark uses plotly as the default backend for plotting. This backend works out of box for a jupyter notebook running on the browser but for VSCode it has to be configured as shown below:

```
# VSCode was used for developing and testing this notebook
# The following code is for plotly figure to render successfully on VSCode
import plotly.io as pio
pio.renderers.default = "vscode"
```



**Figure 4-6 Configuring plotly for VSCode**

This is another step that is common for all the jupyter notebooks submitted for this project.

## 4.6 Suppressing Warnings

As Pandas API on Spark is in its initial phase there are some caveats while using it. One of these things is not being able to suppress warnings. While analysing data this API throws a lot of warnings which are harmless in nature, but the sheer number of them makes it harder to focus on the actual results. A lot of research was conducted on how to suppress them, but they did not turn out to be fruitful. Hence, the current code contains a lot of warnings alongside the results.

## 5 Data Analysis

The first step towards any successful Data Science project is to carry out preliminary data analysis on the underlying dataset. This study has been performed in the **data-analysis.ipynb** and its results have been summarized in this section.

### 5.1 Data Source and Description

The dataset used in this project was taken from the platform “Kaggle”, reference: ([https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis?select=twitter\\_training.csv](https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis?select=twitter_training.csv)). There were two datasets taken from this platform, `twitter_training.csv` and `twitter_validation.csv`, both of them consists of 4 columns, and have ‘74682’ rows and ‘1000’ rows respectively. These two datasets were combined into one dataset, the columns present in the combined dataset defines the “Tweet\_ID”, “Entity”, “Sentiment”, “Tweet\_Content” where each column have unique values. As observed in the dataset the column named “Tweet\_ID” corresponds to a unique ID of a tweet, “Entity” is the entity that the tweet is tweeted for, “Sentiment” is the predefined sentiment label of the tweet present in the column “Tweet\_Content”.

*This dataset has been licensed under CC0 1.0 Universal (CC0 1.0) which means that there is no copyright on the dataset. “It is allowed to copy, modify, distribute, and perform the work, even for commercial purposes.” (Creative Commons, 2022)*

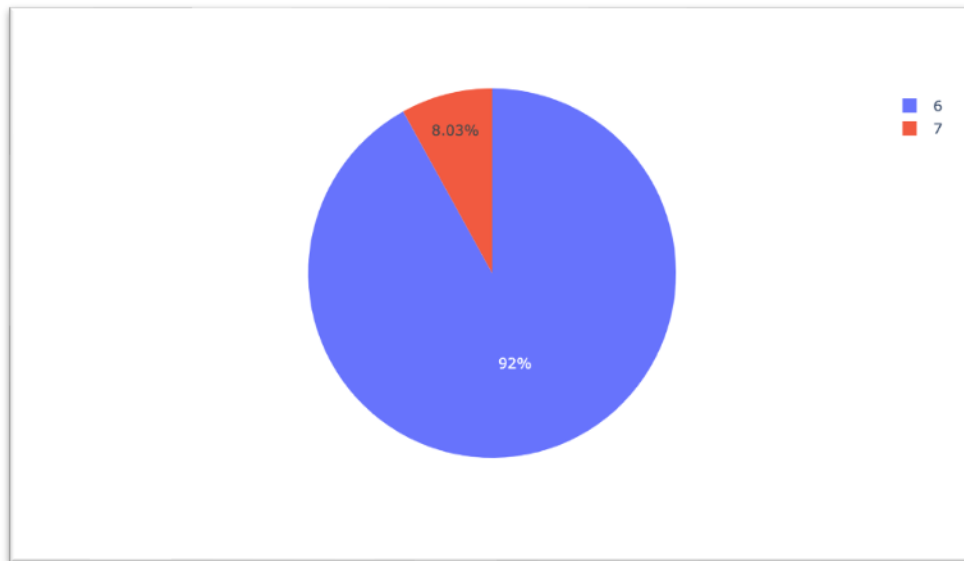
### 5.2 Repeated Tweet IDs

Initially the *Tweet\_ID* column was taken as the index column as usually an ID column is fit to serve as index. After doing so it was realized that the column did not contain unique values.

The following code was executed to check the distribution of number of repeated IDs

```
counts = df.index.value_counts()
print(counts.value_counts())
counts.value_counts().plot.pie()
```

Python



**Figure 5-1 Distribution of number of times Tweet\_ID is repeated**

From the graph it can be seen that there is not even a single *Tweet\_ID* which is unique. Most of the *Tweet\_ID* have been repeated 6 times (92%) and some 7 times(8%)

Multiple tweets were sampled manually to understand the reason behind repeated *Tweet\_IDs* and one such example is shown below:

```
for tweet in df["Tweet_Content"][350].to_numpy():
    print(tweet)
```

Python

```
I played this interesting quiz on Amazon - Try your luck for a chance to win exciting rewards amazon.in/game/share/g8M...
ve played this interesting quiz on Amazon - Try your luck for a chance to win exciting rewards amazon.in / game / share / g8M...
I played this interesting quiz on Amazon - Try your luck for a chance to win exciting rewards amazon.in / game / share / g8M...
I played this interesting lottery on Amazon - Try your luck earn a card to win exciting rewards amazon.in/game/share/g8M...
I also played this interesting game quiz on Amazon - Try your luck today for finding a chance to help win exciting rewards amazon. in / game / share /
and g8M _
I played this interesting rewards game Amazon - so good luck for a day to win exciting rewards amazon.in/game/share/g8M...
```

**Figure 5-2 Tweet content of a sampled Tweet\_ID**

From the above output we can make a hypothesis that the tweets corresponding to single *Tweet\_ID* are actually minor edits for the tweet. This leads to an interesting opportunity. These edits can later be analysed in the **sentiment-analysis.ipynb** to check if minor edits can lead to a different sentiment being predicted. After this point *Tweet\_IDs* were no longer used as the index for the dataframe.

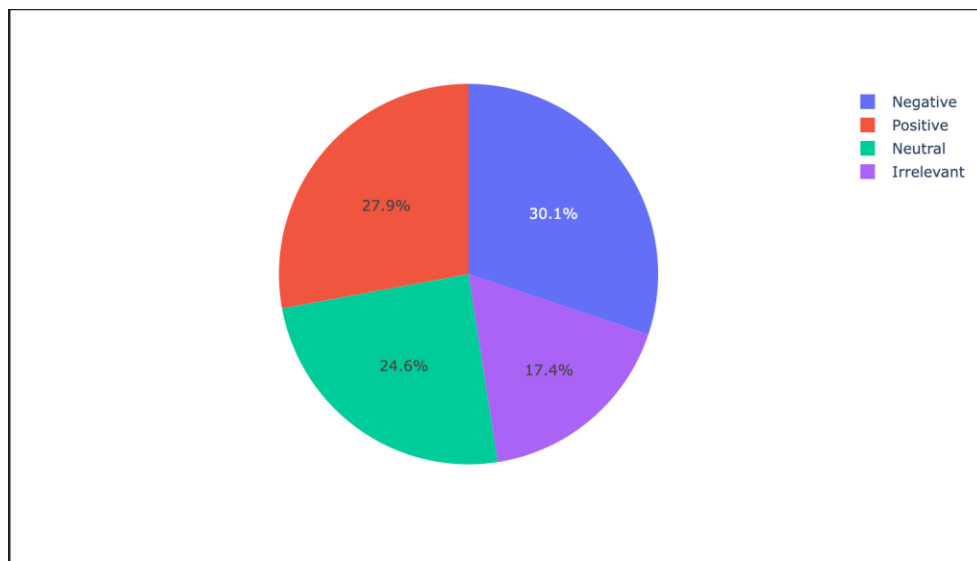
## 5.3 Analysing the Sentiments

This subsection focusses on analysing the Sentiment feature provided in the dataset. This is actually the label that are supposed to be predicted in this project.

The following code allows us to visualise the distribution of sentiments:

```
print(df["Sentiment"].value_counts())
df["Sentiment"].value_counts().plot.pie()
```

Python



**Figure 5-3 Pie Chart of Distribution of Sentiment Labels from the Dataset**

There are “22808” negative sentiments, “21109” positive sentiments, “18603” neutral sentiments and “13162” irrelevant sentiments. The irrelevant sentiments is tagged on tweets which do not relate to the corresponding entity.

After finding the sentiments of the dataset another interesting thing to observe here is to check if all edits for the same Tweet\_ID have the same sentiment. The steps below can be followed:

1. Group the records by `Tweet\_ID`
2. Retrieve the `Sentiment` column of the group
3. Find out the unique sentiments in the group. If there is only a single sentiment for all the edits, then only one unique sentiment will be present for each group
4. Applying the len function to calculate number of unique sentiments for the group
5. Finding groups for which number of unique sentiments is not equal to 1.
6. Step 5 will result in True and False values which can be then summed to find out how many groups don't have a single unique sentiment
7. If the result is 0 then all the edits have the same sentiment.



```
df.groupby(["Tweet_ID"])["Sentiment"].unique().apply(len).ne(1).sum()
```

Python

The above code resulted in a value of 0 hinting that the predefined labels corresponding to each Tweet\_ID are the same.

## 5.4 Analysing Entities

The Entity column represents the entity for which the tweet content is for. The sentiment provided in the 'Sentiment' column, or that analysed by the lexicon-based approaches performed in the 'sentiment-analysis.ipynb' notebook can be used to analyse a generic public opinion on an entity.

This section focusses on analysing different entities present in the dataset, and the next section analyses the sentiments for these entities.

```
counts = df["Entity"].value_counts()

print("Number of Unique Entities", len(counts))
# counts
```

Python

Figure 5-4 Total Number of Unique IDs

The total number of unique entities are 32.

```
df["Entity"].value_counts().plot.pie()
```

Python

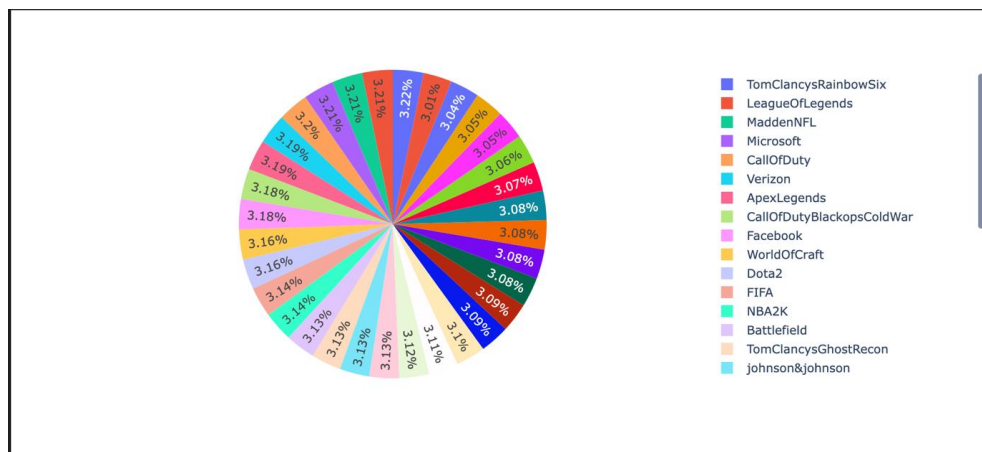


Figure 5-5 Pie Chart of Distribution of Entities from the Data Set

The observation by looking at the above pie chart is that the dataset contains nearly equal number of tweets for each entity. The entity in the dataset is either about brands or video games.

## 5.5 Overview of Sentiments

The following step can be considered in one way to know how an organization can get a brief overview of the public's opinion towards them or their product.

This code output gives us what are the sentiments of the entities in the dataset with the count of each sentiment. The below code gives the pictorial representation of the count of each sentiment.

```
counts = df.groupby(["Entity"])[Sentiment].value_counts().sort_index().to_frame()

counts.columns = ["Count"]
counts

counts = counts.unstack()["Count"]

# counts.head()

counts.plot.bar()
```

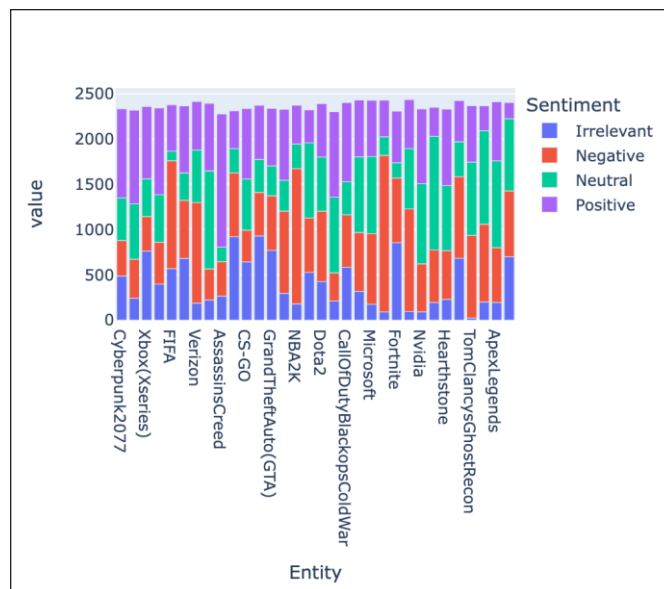


Figure 5-6 Bar graph representing the number of individual sentiments for various entities(Scrollable in the notebook)

## 5.6 Irrelevant Data

After observing multiple tweet contents whose sentiment was “Irrelevant” it can be seen that these tweets are related to the entity. By looking at this a question arises “weather to remove this column from sentiment analysis or keep them in the dataset?

```
df[df["Sentiment"]=="Irrelevant"].head(10)
```

Python

```
(df["Sentiment"]=="Irrelevant").sum()/len(df)
```

Python

Figure 5-7 Calculating percentage of Irrelevant Sentiments

According to the output of the above code the “Irrelevant” sentiment is around 17% of the total rows which is a huge number. These rows can be ignored for computing the accuracy of “Vader Sentiment” and “Textblob” but can be used to compare compound and polarity scores estimated by these two tools.

## 5.7 Missing Values

This step is to find out if we have any missing values in any columns present in the dataset.

```
df[df.isna()["Tweet_ID"]]
```

Python

Tweet_ID	Entity	Sentiment	Tweet_Content
----------	--------	-----------	---------------

```
df[df.isna()["Entity"]]
```

Python

Tweet_ID	Entity	Sentiment	Tweet_Content
----------	--------	-----------	---------------

```
df[df.isna()["Sentiment"]]
```

Python

Tweet_ID	Entity	Sentiment	Tweet_Content
----------	--------	-----------	---------------

```
df[df.isna()["Tweet_Content"]].head()
```

Python

	Tweet_ID	Entity	Sentiment	Tweet_Content
96	16	Amazon	Neutral	None
97	16	Amazon	Neutral	None
98	16	Amazon	Neutral	None
211	37	Amazon	Neutral	None
212	37	Amazon	Neutral	None

Figure 5-8 Finding Missing Values

From the above code cells image, we can see that none of them have missing values except “Tweet\_Content”. An example of Tweet\_Content for a single Tweet\_ID with missing values.

```
df[df["Tweet_ID"]==16]
# Raw Data
# 16,Amazon,Neutral,
# 16,Amazon,Neutral,It is not the first time that the EU Commission has taken such a step.
# 16,Amazon,Neutral,"At the same time, despite the fact that there are currently some 100 million people living i
# 16,Amazon,Neutral,
# 16,Amazon,Neutral,
# 16,Amazon,Neutral,
```

	Tweet_ID	Entity	Sentiment	Tweet_Content
93	16	Amazon	Neutral	
94	16	Amazon	Neutral	It is not the first time that the EU Commissio...
95	16	Amazon	Neutral	At the same time, despite the fact that there ...
96	16	Amazon	Neutral	None
97	16	Amazon	Neutral	None
98	16	Amazon	Neutral	None

Figure 5-9 Analysing Missing Value

Now we can understand that the missing values are present in “Tweet\_Content” column because no content exist in some of the edits, and it still shared the “Tweet\_ID” from “Tweet\_Content” that has non-empty data. By this understanding we can say that it is safe to drop these records for sentiment analysis.

```
missing_count = df.isna()["Tweet_Content"].sum()
print("Records that will be dropped because of missing values: ",missing_count)

print(f"This is {missing_count/len(df):0.2%} of the total data")
```

This is 0.91% of the total data, and hence these missing values can be dropped.

## 6 Parameter Tuning

TextBlob and VaderSentiment estimate polarity and compound scores respectively. They do not predict sentiments directly. These scores can be taken as sentiment scores where a higher value towards 1 indicates a positive sentiment, and a lower value towards -1 indicates a negative sentiment. A range of values in the middle of the range between -1 and 1 can be considered as a neutral sentiment. There is no predefined rule on defining the ranges where a sentiment can be considered positive, neutral, or negative.

For parameter tuning the training set was used. But the output was tested on the complete dataset

### 6.1 TextBlob

In TextBlob we try to output a “polarity score” for each tweet rather than predicting a sentiment. The selection of threshold is up to the developer choice for which tweet they want to tag positive, negative, or neutral. This notebook focusses on finding the best thresholds for which a tweet is successfully predicted as its actual sentiment label.

The code corresponding to this section can be found in the file `textblob_tuning.ipynb`

#### 6.1.1 Filtering Data

```
# As discussed in the data analysis notebook irrelevant tweets can be ignored
# for tuning parameters
df = df[df["Sentiment"]!="Irrelevant"]
```

Python

```
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)
```

Python

Figure 6-1 Dropping Unnecessary records

#### 6.1.2 Tuning Parameters

The `apply` method of a pandas data frame is used to count correct predictions. This method takes only one argument which is a function which accepts one parameter. For tuning different values of `n` and `p` are to be used. Hence, a wrapper function is written below which can allow a for loop to change values of `n` and `p` while still passing a function that takes a single argument to the `apply` method.

Here, `p` is the polarity value above which a tweet is tagged as positive. A polarity value between `n` and `p` is tagged as neutral. A polarity value below `n` is tagged as negative.

```
def getSentimentTextBlob(n=-0.5,p=0.5):
    def getSentiment(line):
        analysis = TextBlob(line)
        if analysis.sentiment.polarity>=p:
            return "Positive"
        elif analysis.sentiment.polarity>= n:
            return "Neutral"
        return "Negative"
    return getSentiment
```

Python

```
positive_cut = None
neutral_cut = None
best_score = 0
grid = np.linspace(-1,1,21)
grid
```

Python

```
array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ,
        0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ])
```

```
for p in range(len(grid)-2,1,-1):
    for n in range(1,p):
        sentiments = df[label].apply(getSentimentTextBlob(round(grid[n],1),round(grid[p],1)))
        score = (sentiments == df[label]).sum()/len(df)
        if score > best_score:
            positive_cut = p
            neutral_cut = n
            best_score = score
```

Python

```
print(f"Positive Cut: {round(grid[positive_cut],1)}, Neutral Cut: {round(grid[neutral_cut],1)}, Best Score: {best_score}")
```

Python

Positive Cut: 0.2, Neutral Cut: -0.2, Best Score: 1.0

**Figure 6-2 Best Thresholds for TextBlob**

### 6.1.3 Best Parameters for TextBlob

The dataset was loaded in a Pandas API on Spark dataset, which was filtered by removing missing values, duplicates, and records with `Irrelevant` sentiment. Then parameters for positive and neutral cuts were tuned by using a grid of values between -1 and +1. The parameters were tuned on the basis of finding the best accuracy, and the resultant cuts were estimated as follows:

Positive Sentiment:  $1 \geq \text{Polarity} \geq 0.2$   
 Neutral Sentiment:  $0.2 > \text{Polarity} \geq -0.2$   
 Negative Sentiment:  $-0.2 > \text{Polarity} \geq -1$

## 6.2 Vader Sentiment

In Vader we try to output a “compound score” for each tweet rather than predicting a sentiment. The selection of threshold is up to the developer choice for which tweet they want to tag positive, negative, or neutral. This notebook focusses on finding the best thresholds for which a tweet is successfully predicted as its actual sentiment label.

The code corresponding to this section can be found in the `vader_tuning.ipynb` notebook.

### 6.2.1 Filtering Data

```
# As discussed in the data analysis notebook irrelevant tweets can be ignored
# for tuning parameters
df = df[df["Sentiment"]!="Irrelevant"]

df.drop_duplicates(inplace=True)
df.dropna(inplace=True)
```

Python

Python

Figure 6-3 Dropping Unnecessary Records

### 6.2.2 Tuning Parameters

```
analyzer = SentimentIntensityAnalyzer()
```

Python

The `apply` method of a pandas DataFrame is used to count correct predictions. This method takes only one argument which is a function which accepts one parameter. For tuning different values of `n` and `p` are to be used. Hence, a wrapper function is written below which can allow a for loop to change values of `n` and `p` while still passing a function that takes a single argument to the `apply` method.

Here, `p` is the compound value above which a tweet is tagged as positive. A compound value between `n` and `p` is tagged as neutral. A compound value below `n` is tagged as negative.



```

def getSentimentVader(n=-0.5,p=0.5):
    def getSentiment(line):
        scores = analyzer.polarity_scores(line)
        if scores['compound'] >= p:
            return "Positive"
        elif scores['compound'] > n:
            return "Neutral"
        return "Negative"
    return getSentiment

positive_cut = None
neutral_cut = None
best_score = 0
grid = np.linspace(-1,1,21)
grid

array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ,
        0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ])

for p in range(len(grid)-2,1,-1):
    for n in range(1,p):
        sentiments = df[label].apply(getSentimentVader(grid[n],grid[p]))
        score = (sentiments == df[label]).sum()/len(df)
        # print(f"p:{grid[p]}, n:{grid[n]}, score:{score}")
        if score > best_score:
            positive_cut = p
            neutral_cut = n
            best_score = score
            # print(f"p:{grid[p]}, n:{grid[n]}, score:{best_score}")

print(f"Positive Cut: {grid[positive_cut]}, Neutral Cut: {grid[neutral_cut]}, Best Score: {best_score}")

Positive Cut: 0.5, Neutral Cut: -0.5, Best Score: 1.0

```

**Figure 6-4 Best Thresholds for Vader Sentiment**

### 6.2.3 Best Parameters For Vader Sentiment

The dataset was loaded in a Pandas API on Spark dataset, which was filtered by removing missing values, duplicates, and records with 'Irrelevant' sentiment. Then parameters for positive and neutral cuts were tuned by using a grid of values between -1 and +1. The parameters were tuned on the basis of finding the best accuracy, and the resultant cuts were estimated as follows:

Positive Sentiment: 1 >= Compound >= 0.5  
 Neutral Sentiment: 0.5 > Compound >= -0.5  
 Negative Sentiment: -0.5 > Compound >= -1

## 7 Results and Analysis

Best cuts for predicting sentiments using TextBlob and Vader were estimated in the notebooks `textblob\_tuning.ipynb` and `vader\_tuning.ipynb` respectively. This notebook uses those cuts to predict and analyze the sentiments as estimated by the models. Apart from that this notebook also discusses how the subjectivity scores can be used to filter tweets on the basis of how subjective or objective they are.

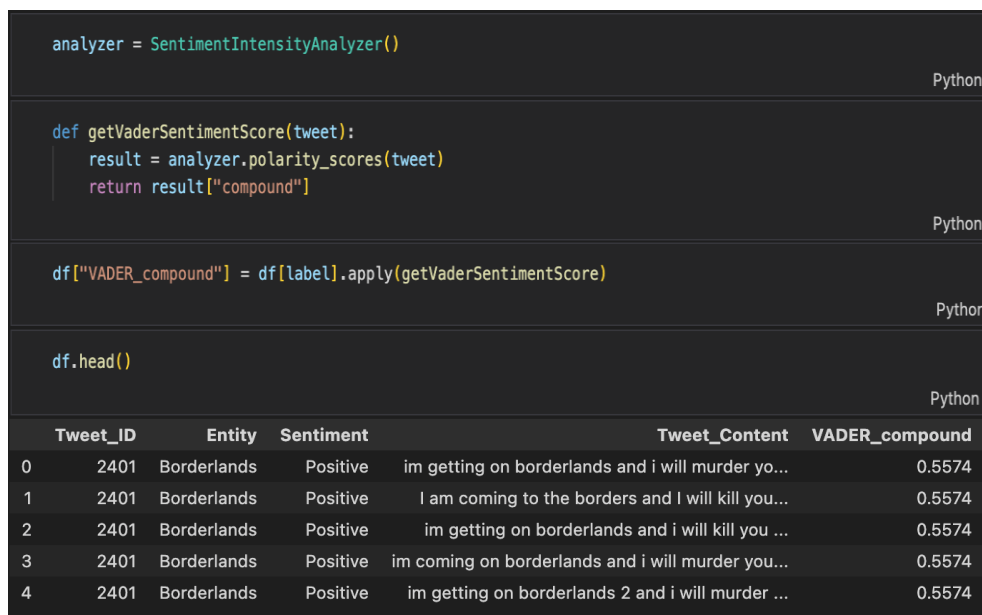


Figure 7-1 Estimating Vader Compound

As done in the `data-analysis.ipynb` notebook, the following code checks if the compound score estimated by Vader is the same for all edits of a tweet corresponding to a single `Tweet\_ID`.

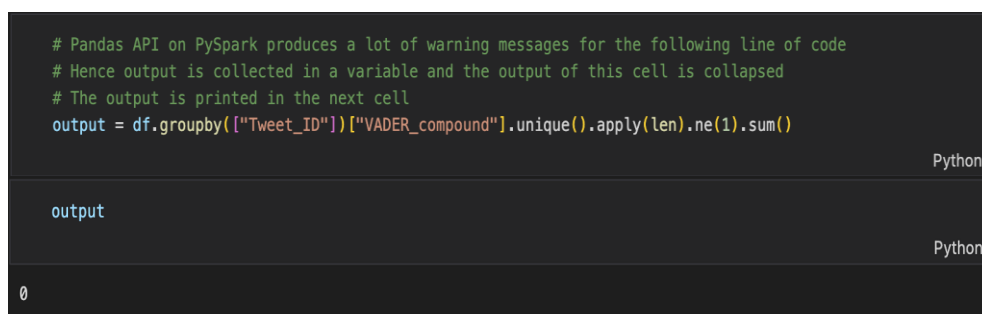


Figure 7-2 Verifying if all compound scores for a Tweet\_ID are same

As we can see that the above output is 0 hence, we can conclude that all the edits of a single "Tweet\_ID" received the same compound score from Vader.

```
def getTextBlobPolarity(line):
    analyzer = TextBlob(line)
    return analyzer.sentiment.polarity
```

Python

```
df["TEXTBLOB_polarity"] = df[label].apply(getTextBlobPolarity)
```

Python

```
df.head()
```

Python

	Tweet_ID	Entity	Sentiment	Tweet_Content	VADER_compound	TEXTBLOB_polarity
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	0.5574	0.227273
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	0.5574	0.227273
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	0.5574	0.227273
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	0.5574	0.227273
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	0.5574	0.227273

**Figure 7-3 Estimating TextBlob Polarity**

By looking at the above output for all the edits for the same Tweet\_ID seems to have same score.

```
output = df.groupby(["Tweet_ID"])["TEXTBLOB_polarity"].unique().apply(len).ne(1).sum()
```

Python

```
output
```

Python

```
0
```

```
def getTextBlobSubjectivity(line):
    analyzer = TextBlob(line)
    return analyzer.sentiment.subjectivity
```

Python

```
df["TEXTBLOB_subjectivity"] = df[label].apply(getTextBlobSubjectivity)
```

Python

**Figure 7-4 Estimating TextBlob Subjectivity**

The output for all the edits for the same Tweet\_ID in the subjectivity seems to have same score.

```
output = df.groupby(["Tweet_ID"])["TEXTBLOB_subjectivity"].unique().apply(len).ne(1).sum()
```

Python

```
output
```

```
0
```

Python

Figure 7-5 Verifying if all polarity scores for a Tweet\_ID

### 7.1.1 Predict Sentiment using Vader

The following code is different from what was written in `vader\_tuning.ipynb`. Here, the thresholds used for predicting sentiment are hard coded values that have been tuned in the aforementioned notebook.

```
def getVaderSentiment(score):
    if score>=0.5:
        return "Positive"
    elif score >=-0.5:
        return "Neutral"
    return "Negative"
```

Python

```
df["VADER_sentiment"] = df["VADER_compound"].apply(getVaderSentiment)
```

Python

```
df.head()
```

Python

	Tweet_ID	Entity	Sentiment	Tweet_Content	VADER_compound	TEXTBLOB_polarity	TEXTBLOB_subjectivity	VADER_sentiment
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	0.5574	0.227273	0.545455	Positive
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	0.5574	0.227273	0.545455	Positive
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	0.5574	0.227273	0.545455	Positive
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	0.5574	0.227273	0.545455	Positive
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	0.5574	0.227273	0.545455	Positive

Figure 7-6 Predicting Sentiments using Vader

### 7.1.2 Irrelevant Sentiments Vader

This step is to check the accuracy for rows whose sentiment is not Irrelevant.

```
not_irrelevant = df[df[label]!="Irrelevant"]
output = ((not_irrelevant[label] == not_irrelevant["VADER_sentiment"]).sum())/len(not_irrelevant)
```

Python

```
output
```

Python

```
1.0
```

Figure 7-7 Accuracy for Vader

### 7.1.3 Predicting Sentiments using TextBlob

```
def getTextBlobSentiment(polarity):  
    if polarity>=0.2:  
        return "Positive"  
    elif polarity >=-0.2:  
        return "Neutral"  
    return "Negative"  
  
df["TEXTBLOB_sentiment"] = df["TEXTBLOB_polarity"].apply(getTextBlobSentiment)
```

Python

Python

Figure 7-8 Predicting Sentiments using TextBlob

### 7.1.4 Irrelevant Sentiments TextBlob

```
not_irrelevant = df[df[label]!="Irrelevant"]  
output = ((not_irrelevant[label] == not_irrelevant["TEXTBLOB_sentiment"]).sum())/len(not_irrelevant)
```

Python

output

Python

1.0

Figure 7-9 Accuracy for TextBlob

Textblob also predicts all the labels accurately.

```
df.head()
```

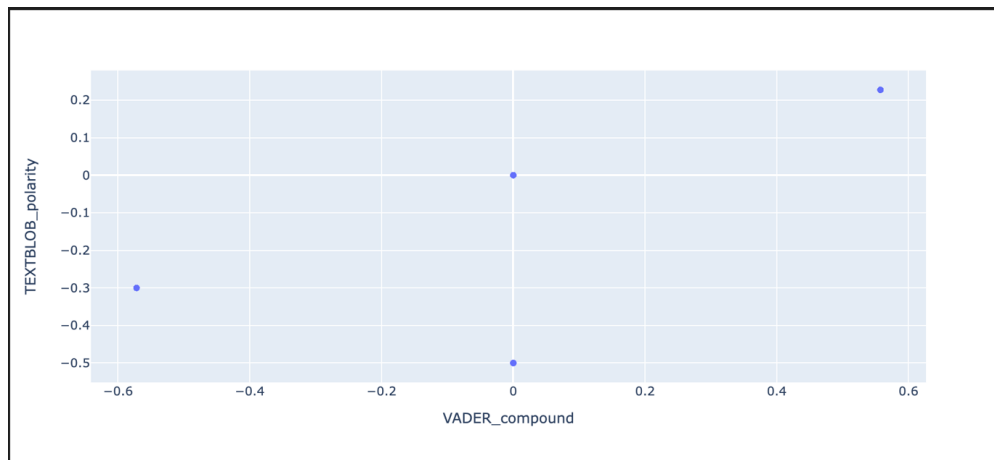
Python

	Tweet_ID	Entity	Sentiment	Tweet_Content	VADER_compound	TEXTBLOB_polarity	TEXTBLOB_subjectivity	VADER_sentiment	TEXTBLOB_sentiment
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...	0.5574	0.227273	0.545455	Positive	Positive
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...	0.5574	0.227273	0.545455	Positive	Positive
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...	0.5574	0.227273	0.545455	Positive	Positive
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...	0.5574	0.227273	0.545455	Positive	Positive
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...	0.5574	0.227273	0.545455	Positive	Positive

### 7.1.5 Comparing Vader Compound with TextBlob Polarity

```
df.plot.scatter(x="VADER_compound",y="TEXTBLOB_polarity")
```

Python



**Figure 7-10 Comparing Compound and Polarity scores**

From the scatter plot we can observe that there are four unique values estimated for “TextBlob polarity” and three unique values estimated for “VADER compound”.

For 0 value of “VADER compound”, “TEXTBLOB polarity” seems to have two values 0, and -0.5. This is counter intuitive as this will result in different sentiments, but both Vader and Textblob have seem to predict all sentiments accurately for rows not having ‘Irrelevant’ sentiment.

```
df_differ = df[df["VADER_sentiment"] != df["TEXTBLOB_sentiment"]]
length = len(df_differ)
only_irrelevant = (df_differ["Sentiment"].unique().to_list() == ["Irrelevant"])

print(f"Number of rows for which vader and textblob have different sentiments {length}")
print(f"Do these rows only contain the sentiment 'Irrelevant'? {only_irrelevant}")
```

Python

```
print(f"Number of rows for which vader and textblob have different sentiments {length}")
print(f"Do these rows only contain the sentiment 'Irrelevant'? {only_irrelevant}")
```

Python

Number of rows for which vader and textblob have different sentiments 13047  
Do these rows only contain the sentiment 'Irrelevant'? True

**Figure 7-11 Differences between predictions from Vader and TextBlob**

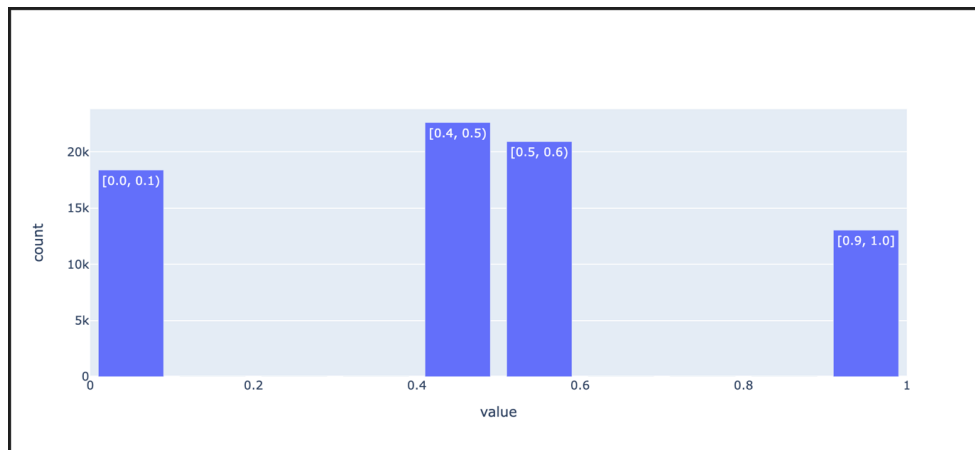
### 7.1.6 Working with Subjectivity

Subjectivity is not a field to predict sentiments. It rather seems like a value that can be used to filter sentiments by an organization. An organization might want to view tweets or their sentiments on the basis of how objective or subjective a tweet is.

Let us plot a histogram to understand the subjectivity of tweets.

```
df["TEXTBLOB_subjectivity"].hist()
```

Python



**Figure 7-12 Histogram of distribution of subjectivity**

Subjectivity can be used to find the tweets where user have expressed their personal feelings as shown below,

```
for tweet in df[df["TEXTBlob_subjectivity"]>0.5].sample(frac=0.01).head()["Tweet_Content"].to_numpy():
    print(tweet)
```

Python

Or tweets can be filtered on the basis of how objective they are as shown below,

```
for tweet in df[df["TEXTBlob_subjectivity"]<0.5].sample(frac=0.01).head()["Tweet_Content"].to_numpy():
    print(tweet)
```

Python

The cut for subjectivity can be reduced even further to filter more objective tweets

```
for tweet in df[df["TEXTBlob_subjectivity"]<0.1].sample(frac=0.01).head()["Tweet_Content"].to_numpy():
    print(tweet)
```

Python

Objective tweets can also be filtered for a specific entity as shown below,

```
entity = "Amazon"
df_entity = df[df["Entity"]==entity]
for tweet in df_entity[df_entity["TEXTBlob_subjectivity"]<0.1].sample(frac=0.01).head()["Tweet_Content"].to_numpy():
    print(f"Tweet for Entity: {entity}")
    print(tweet)
```

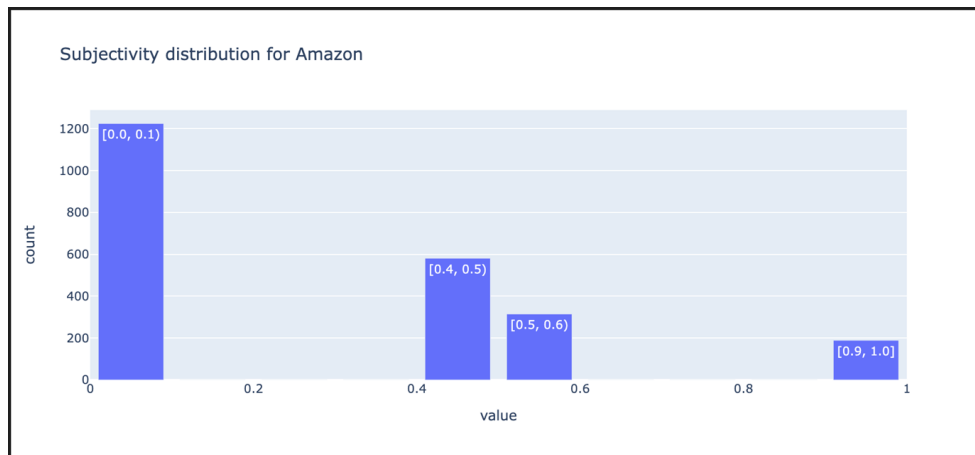
Python

**Figure 7-13 Working with Subjectivity**

A histogram showing different levels of subjectivity for a particular entity can also be plotted as shown below

```
df_entity["TEXTBLOB_subjectivity"].hist(title=f"Subjectivity distribution for {entity}")
```

Python



**Figure 7-14 Subjectivity Distribution for a sample entity Amazon**

The same can be configured for any level of subjectivity by just modifying the cuts.

The twitter sentiment analysis dataset was loaded in a Pandas API on Spark dataframe. Using the cut's estimated sentiments were predicted. The predicted sentiments had a 100% accuracy if not considering Irrelevant sentiments. As discussed in the "data\_analysis.ipynb" notebook the scores estimated for various tweet edits turned out to be the same. TextBlob and Vader predicted the same sentiments except for tweets labelled with 'Irrelevant' sentiment where the sentiments predicted were a bit different. Finally, subjectivity scores estimated by TextBlob were used to filter tweets for better analysis of subjective and objective tweets.



## 8 Conclusion

This project presented two tools that can be used for Sentiment Analysis: TextBlob and Vader Sentiment. The underlying dataset encompassed of thousands of tweets parsed into columns. The dataset contained predefined sentiments which were taken as labels to measure the performance of the tools. Both these models returned an accuracy of 100% when trying to predict Positive, Neutral, and Negative as labels for Sentiment Analysis.

TextBlob and Vader Sentiment do not predict sentiments on their own, instead they estimate polarity and compound scores which can be taken as sentiment scores. These scores have a value between -1 and +1 with a score leaning towards +1 indicates a positive sentiment, and value leaning towards -1 indicates a negative sentiment. Values around 0 can be considered Neutral Sentiments. Using the dataset three sets of ranges were tuned for the sentiments. A sentiment score that falls within these ranges indicates which sentiment label is to be predicted. This tuning is the main cause of getting a 100% accuracy rate.

Furthermore, few code blocks were proposed in the **4 Data Analysis** section. The result of these code blocks produces tables and charts that can be used by an organization to get an overview of the sentiments for their brand or products. Even in Section **6 Results and Analysis**, subjectivity scores estimated by TextBlob was utilized to filter tweets on the basis of how subjective or objective they are.

### 8.1 Further scope

Sentiment Analysis have a lot of scope in the future since large volume of data is generated by the social media platforms day by day (Villavicencio et al. 2021). More algorithms will be discovered and employed for the sentiment analysis of Twitter data and the process of extracting the sentiments of the users from the dataset will become more complex. Twitter and other social networking platforms will play a most significant role in the future and communication will revolve around these platforms. There will be need for specialists and data analysts for the sentiment analysis of social media datasets. Further research will be required to increase the accuracy of the models for sentiment analysis and to develop alternative ways of performing sentiment analysis. For further implementation on sentiment analysis, BERT and hugging face techniques can be used.

## 9 Self-Assessment

This project has been a learning journey which I am going to remember for the years to come. I started the project with very little knowledge and awareness of my project, and I finished it with experience that has made me very confident in life. This is evident from the fact that the initial code I had differs heavily from the one I am submitting in terms of organisation and best practices.

I am very happy with how the project turned out to be. The only regret I had is that I could not include BERT for sentiment analysis in the final draft. I spent a lot of time learning and experimenting with BERT, but it wasn't enough to get publishable results. Nevertheless, I am content with my capability to manage time and submit a project which is complete in its own sense rather than a project with lot of incomplete features.

I faced a lot of unexpected hurdles while working on this project which caused a lot of stress and challenged my capability to maintain my mental ability. I am extremely proud of myself as I stayed strong throughout, while managing my time and priorities and submitted a working copy of my project.

I went through a lot of issues like lack of relevant resources and hardware capacity. As I started with limited knowledge, I searched the internet for relevant resources but could not find ones with the exact set of tools that I required. For example, to understand how to use pyspark for sentiment analysis with TextBlob or Vader sentiment, I found disjointed resource on how to use pyspark and how to use the other tools separately. Hence, I spent a lot of time practicing the use of Logistic Regression with pyspark to get comfortable with pyspark. I wrote a lot of code which unfortunately did not make it to the final draft. It isn't until very late that I stumbled upon Pandas API on Pyspark which got me better approach. I had to start from scratch, but it was worth it with some limitations.

In the end I learned new concepts while implementing what I'd learned through my programme. Now, I am comfortable in planning projects, managing my time, setting up priorities and expectations, and delivering results. I have grown a lot over this period personally and professionally and cannot wait to apply my skills in my future career.

## **10 Professional Issues: The ethics of Data Collection and Storage**

This project was developed while considering ethical issues. The dataset used is deprived of any personally identifiable information, and the Tweet\_ID column doesn't correspond to an official tweet id which is a 32-bit to 64-bit identifier (Twitter, 2022). Along with it there are multiple professional issues related to data collection, and storage that data scientists and machine learning engineers should consider while designing a Sentiment Analysis solution.

### **10.1 Data Collection**

This project utilized tweets retrieved from a secondary source, but sentiment analysis can be performed on any form of text that an individual posts online. This can be from any social media platforms, reviews on retail websites, review websites, or maybe some forum. Normally to post on these forums the user consents for their information to be collected, stored, and processed. But it is not a general notion that their opinions will be automatically processed to predict their sentiments about a particular entity. The term that is being pointed out is "Informed Consent" that whether a user was made aware that their opinions would be mined or not. This is a complicated task because there are various other professional issues that accompany during the collection of data.

### **10.2 Data Storage**

The primary idea to consider here is whether sentiments can be considered private data. If yes, then is it allowed for an organization to store the data without consent. If personally identifiable information is attached to sentiments, will the organization try to target a group of people sharing a sentiment to manipulate their views.

## References

Sayed, A. A., Abdallah, M. M., Zaki, A. M. & Ahmed, A. A., 2020. *Big Data analysis using a metaheuristic algorithm: Twitter as Case Study*. Egypt, IEEE.

K, R. & Vennila, D. M., 2021. An Implementation of Hybrid Enhanced Sentiment Analysis System using Spark ML Pipeline: A Big Data Analytics Framework. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, Vol. 12, No. 5, 2021, 12(5), pp. 323-329.

Ortega, P., Arcila, D. & García, P., 2019. *Predictive sentiment analysis of messages for Journalistic Purposes: Real-time classification of tweets based on Machine Learning*. Dublin, University College Dublin..

Beri, A., 2020. *towardsdatascience*. [Online]  
Available at: [https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664#:~:text=VADER%20\(%20Valence%20Aware%20Dictionary%20for,intensity%20\(strength\)%20of%20emotion](https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664#:~:text=VADER%20(%20Valence%20Aware%20Dictionary%20for,intensity%20(strength)%20of%20emotion).  
[Accessed 28 October 2022].

Databricks, 2022. *Databricks*. [Online]  
Available at: <https://docs.databricks.com/pandas/pandas-on-spark.html>  
[Accessed 25 November 2022].

Hutto, C., 2021. *Github*. [Online]  
Available at: <https://github.com/cjhutto/vaderSentiment#about-the-scoring>  
[Accessed 20 October 2022].

Kwon, H. & Meng, X., 2021. *databricks*. [Online]  
Available at: <https://www.databricks.com/blog/2021/10/04/pandas-api-on-upcoming-apache-spark-3-2.html>  
[Accessed 25 November 2022].

Shah, P., 2020. *towardsdatascience*. [Online]  
Available at: <https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524>  
[Accessed 2021].

Spark, 2022. *Apache spark*. [Online]  
Available at:  
[https://spark.apache.org/docs/latest/api/python/user\\_guide/pandas\\_on\\_spark/supported\\_pandas\\_api.html#dataframe-api](https://spark.apache.org/docs/latest/api/python/user_guide/pandas_on_spark/supported_pandas_api.html#dataframe-api)  
[Accessed 25 Nov 2022].

Twitter, 2022. *Twitter IDs*. [Online]  
Available at: <https://developer.twitter.com/en/docs/twitter-ids>  
[Accessed 25 Nov 2022].

Schuller, B., Ganascia, J.G. and Devillers, L., 2016. Multimodal sentiment analysis in the wild: Ethical considerations on data collection, annotation, and exploitation. In *Proceedings of the 1st International Workshop on ETHics In Corpus Collection, Annotation and Application (ETHI-CA '20 \$2016)*, satellite of the 10th Language Resources and Evaluation Conference (LREC 2016) (2016) (pp. 29-34).

Kılınç, D., 2019. A spark-based big data analysis framework for real-time sentiment prediction on streaming data. *Software: Practice and Experience*, 49(9), pp. 1352-1364.

Sujatha, E. and Radha, R., 2021. A Hybrid of Proposed Filtration and Feature Selections to Enhance the Model Performance. *Indian Journal of Science and Technology*, 14(24), pp.2039-2050.

Mittal, A. & Goel, A., 2012. *Stock Prediction Using Twitter Sentiment Analysis*, Stanford: Stanford University.

Elzayady, H., Badran, K.M. and Salama, G.I., 2018, December. Sentiment analysis on twitter data using apache spark framework. In 2018 13th International Conference on Computer Engineering and Systems (ICCES) (pp. 171-176). IEEE.

Alexopoulos, A., Drakopoulos, G., Kanavos, A., Sioutas, S. and Vonitsanos, G., 2020, September. Parametric evaluation of collaborative filtering over apache spark. In 2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM) (pp. 1-8). IEEE.

Jalil, S.A., Vikas, S.N., Madhusudan, M.S. and Jangid, P., 2020. Social Media Sentiment Analysis Using Big data. Department of Information Technology, p.174.

Ilhan, B.E., Kübler, R.V. and Pauwels, K.H., 2018. Battle of the brand fans: Impact of brand attack and defense on social media. *Journal of Interactive Marketing*, 43, pp.33-51.

Adwan, O., Al-Tawil, M., Huneiti, A., Shahin, R., Zayed, A.A. and Al-Dibsi, R., 2020. Twitter sentiment analysis approaches: A survey. *International Journal of Emerging Technologies in Learning (iJET)*, 15(15), pp.79-93.

Pota, M., Ventura, M., Catelli, R. and Esposito, M., 2020. An effective BERT-based pipeline for Twitter sentiment analysis: A case study in Italian. *Sensors*, 21(1), p.133.

Jalil, S.A., Vikas, S.N., Madhusudan, M.S. and Jangid, P., 2020. Social Media Sentiment Analysis Using Big data. Department of Information Technology, p.174.

Sasikanth, K.V.K., Samatha, K., Deshai, N., Sekhar, B.V.D.S. and Venkatramana, S., 2020. Effective Sentiment Analysis on Twitter with Apache Spark. *International Journal of Industrial Engineering & Production Research*, 31(3), pp.343-350.

Pradhan, S., Kasetwar, A. and Badjate, S. (2022) "Spatial and Sentimental Analysis for Trending Information Technology", *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology*, 14(01 SPL), pp. 93-98. doi: 10.18090/samriddhi.v14spli01.18.

Hasan, R.A., Alhayali, R.A.I., Zaki, N.D. and Ali, A.H., 2019. An adaptive clustering and classification algorithm for Twitter data streaming in Apache Spark. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(6), pp.3086-3099.

Zad, S., Heidari, M., Jones, J.H. and Uzuner, O., 2021, May. A survey on concept-level sentiment analysis techniques of textual data. In *2021 IEEE World AI IoT Congress (AllIoT)* (pp. 0285-0291). IEEE.

Ardakani, S.P., Zhou, C., Wu, X., Ma, Y. and Che, J., 2021, December. A Data-driven Affective Text Classification Analysis. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 199-204). IEEE.

Babu, N.V. and Rawther, F.A., 2019. User Behavior Analysis on Social Media data using Sentiment Analysis or Opinion Mining. *International Research Journal of Engineering and Technology (IRJET)*, 6(6).

Chuanming, Y., Sai, Y., Feng, W. and Lu, A., 2019. Research on Scale Adaptation of Text Sentiment Analysis Algorithm in Big Data Environment: Using Twitter as Data Source. *Library and Information Service*, 63(4), p.101.

cjhutto, 2014. *Github*. [Online]  
Available at: <https://github.com/cjhutto/vaderSentiment>  
[Accessed October 2022].

Sloria, 2013. *Github*. [Online]  
Available at: <https://github.com/sloria/TextBlob>  
[Accessed October 2022].

