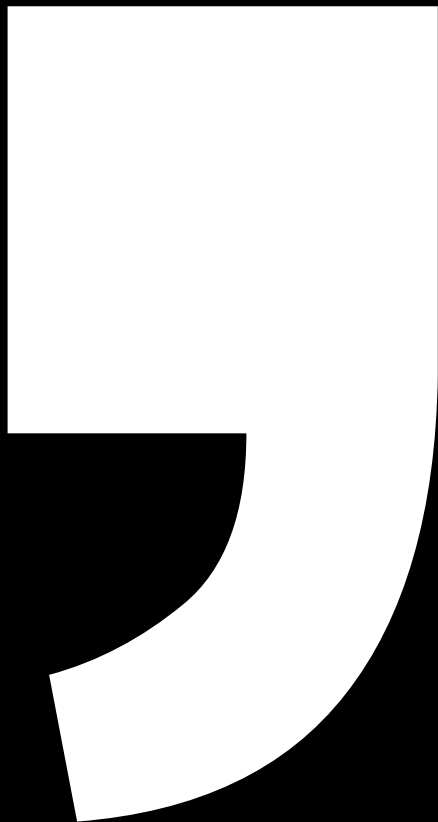


# IMDB Sentiment Analysis

Mohamed Arbi Wérghi  
Ibrahim Doghri



# Table of Contents

**01**

Introduction

**03**

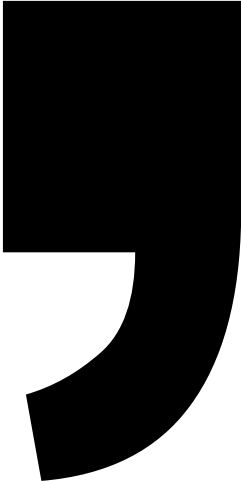
Process

**02**

Results

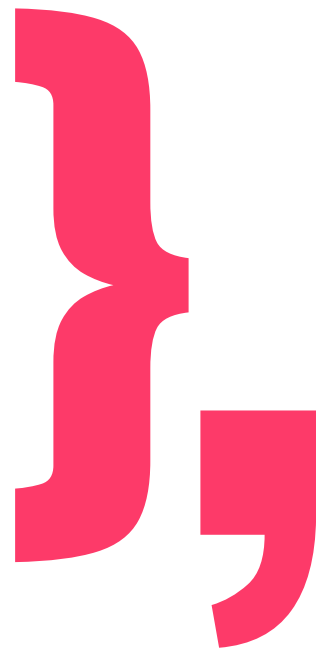
**04**

Conclusion





# **Introduction**



# What is Beautiful Soup

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

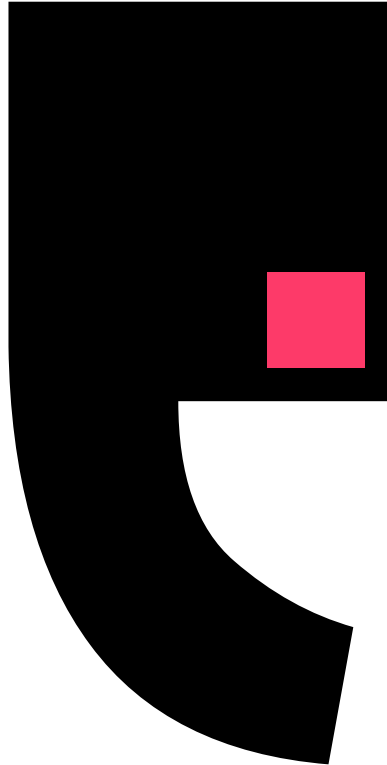


# What is NLP

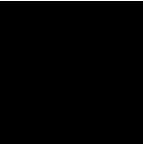
Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.



# Our Project



The project is made up of three parts. The first one is scraping the data from IMDB in which we used the **Beautiful Soup** library. We then clean and reshape the data as it is necessary to be able to visualize it and process it which is the third and last step. We used the **"Multinomial Naive Bayes"** and **"Vader"** Models to conduct the sentiment analysis as well as **"Seaborn"** and **"Matplotlib"** to visualize it

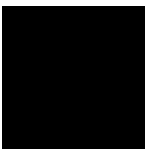
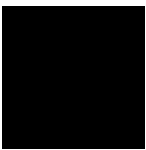


{

**Results**

},

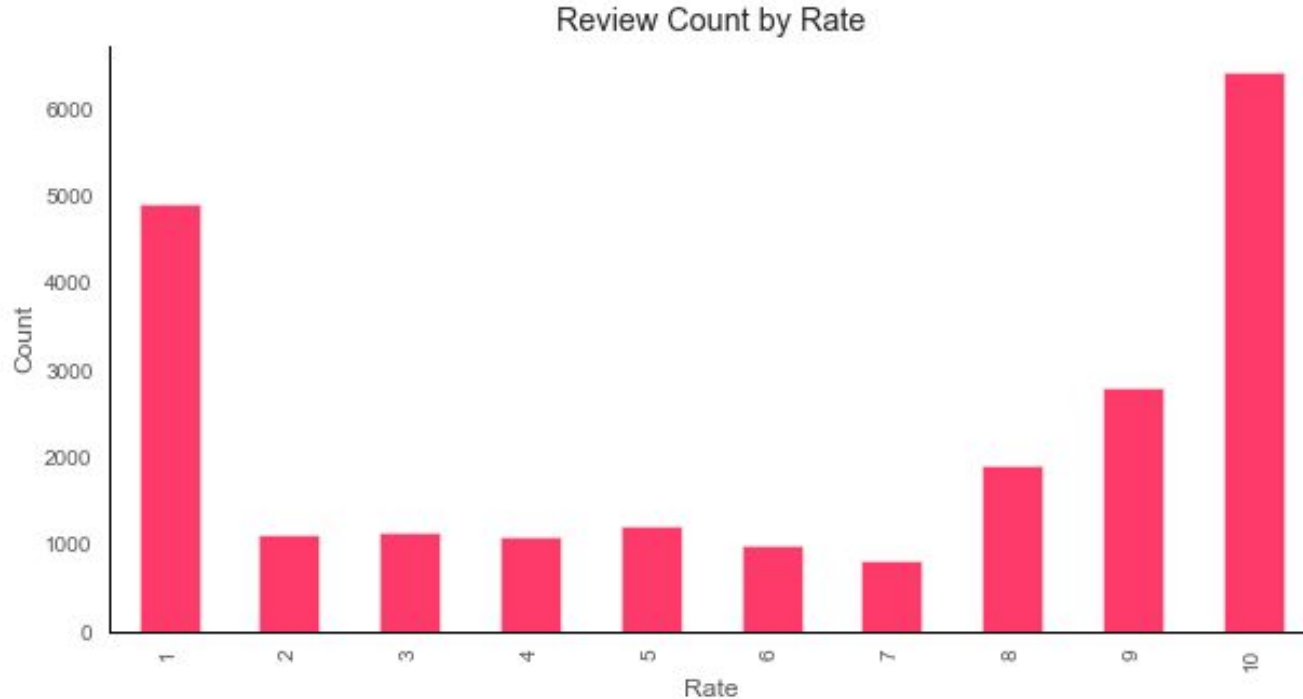
# Multinomial Naïve Bayes



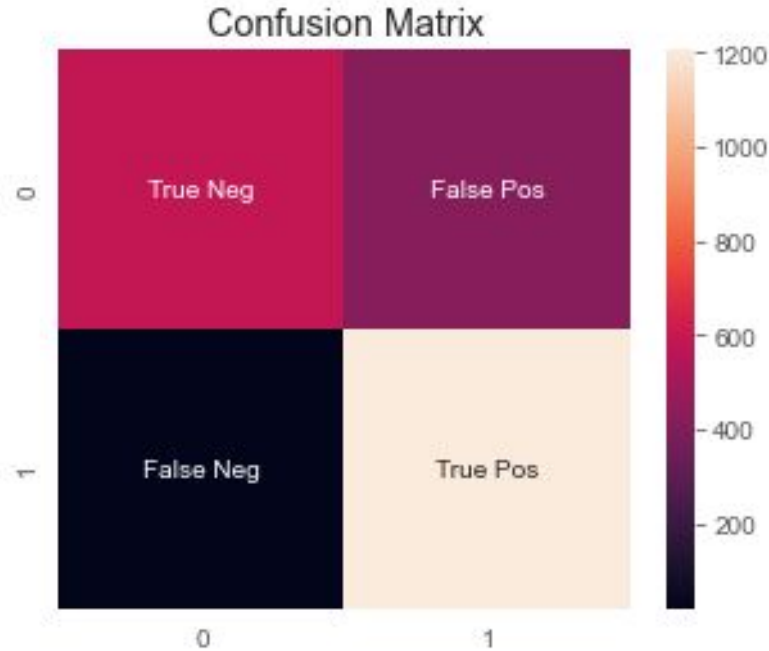
Title	Precision	Recall	F-1 Score	Support
Negative review	0.97	0.58	0.72	1010
Positive review	0.74	0.98	0.84	1224
Accuracy			0.80	2234
Macro Avg	0.85	0.78	0.78	2234
Weighted Avg	0.84	0.80	0.97	2234



# Multinomial Naïve Bayes



# Multinomial Naïve Bayes

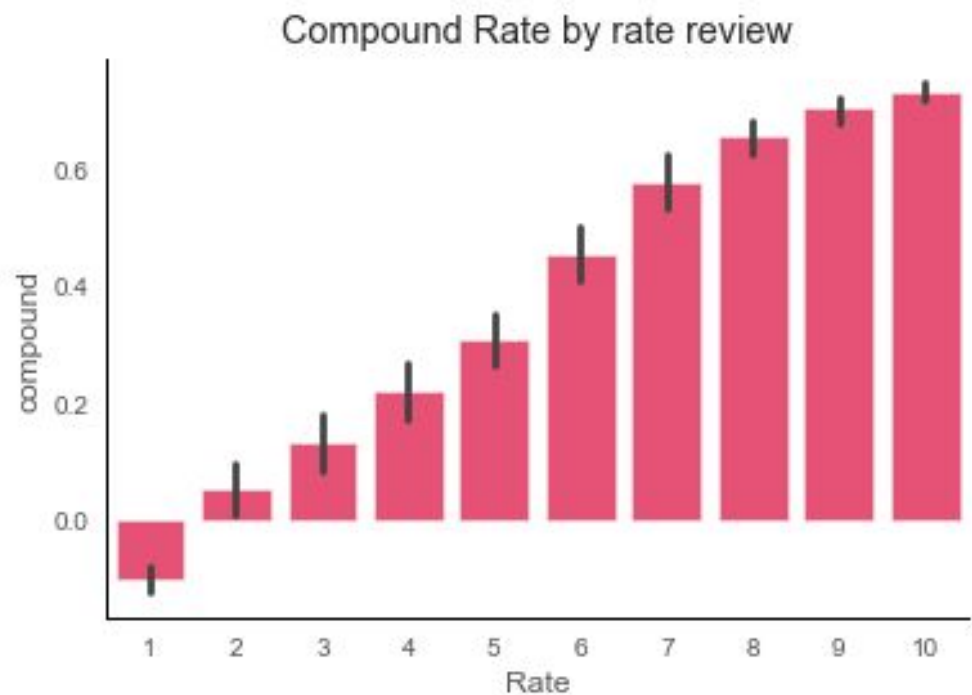


# VADER

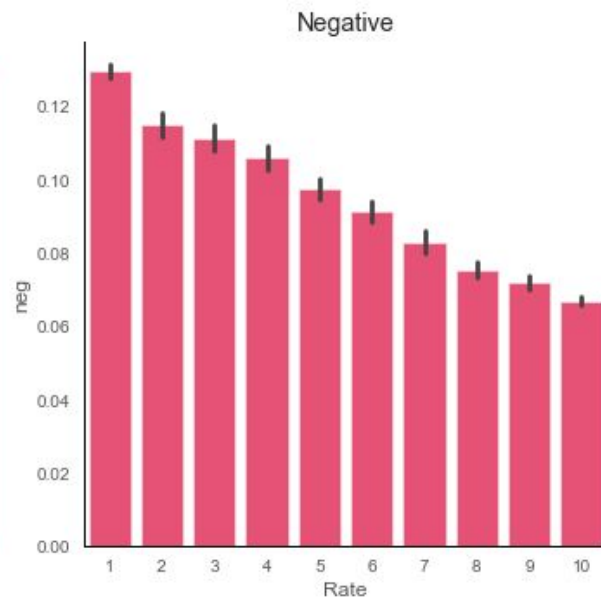
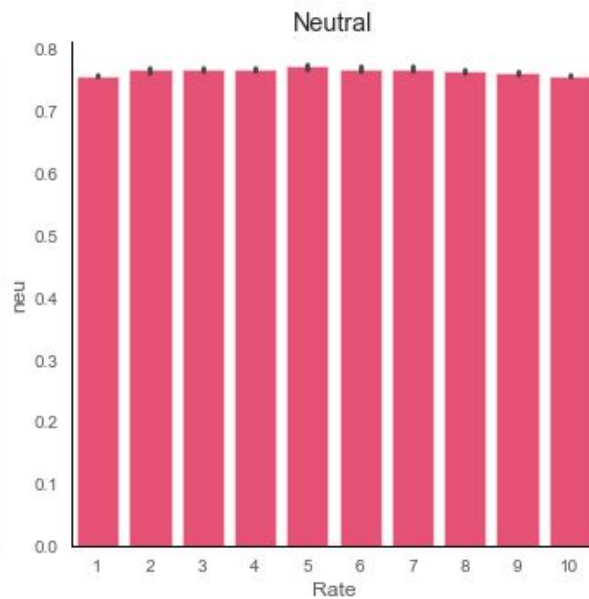
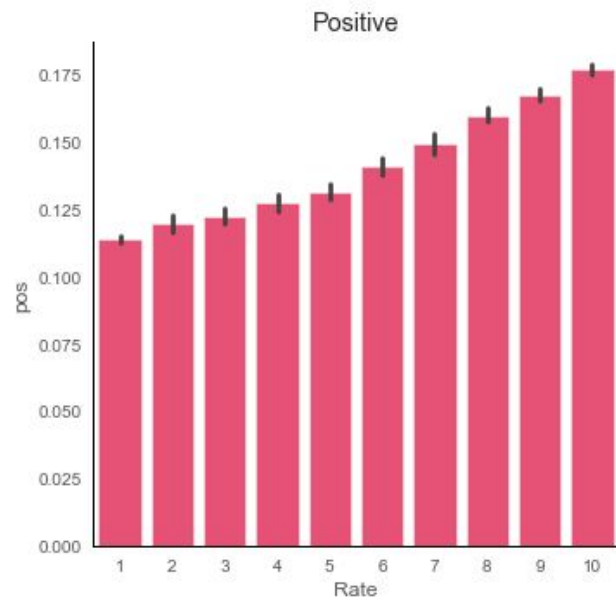


	id	Negative	Neutral	Positive	Compound	Rate	Review
0	1	0.143	0.674	0.182	0.8353	6	This was great the first time I watched it, bu...
1	2	0.058	0.776	0.167	0.9945	9	James Whale is, for good reason, most famous f...
2	3	0.081	0.755	0.164	0.7500	10	This is definitely one of the best horror/sci-...
3	4	0.015	0.820	0.165	0.9377	8	I actually saw The Invisible Man (1933) shortl...
4	5	0.132	0.718	0.149	0.5742	10	Spoiler ahead - a well known one though.It was...

# VADER

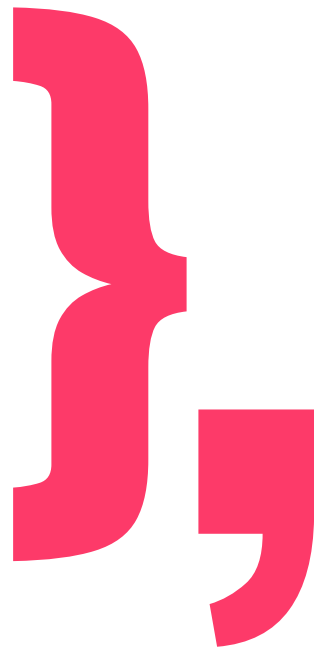


# VADER



A large, thick, pink opening curly brace on the left side of the image.

**Process**

A large, thick, pink closing curly brace followed by a comma on the right side of the image.

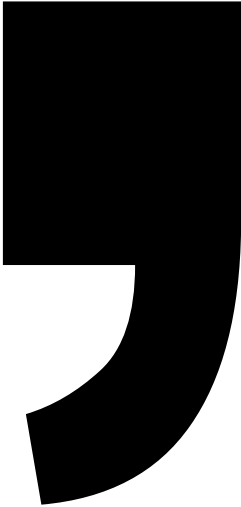


# **Web Scraping**

# Scraping The Movie Details

Scraping Data off of IMDB is straightforward already since there are no anti-scraping precautions being taken, and “**Beautiful Soup**” makes it even simpler. We scraped the data of **2000** movies each has multiple reviews  
~~~~~**250** of which resulted in getting **324,290** reviews.

Though, for the data to be cleaner we restored to storing the reviews separately with only the necessary/ needed data for the analysis.





# Parsing Movie Listing Pages



Web-scraping- - scrapingMoreMovies.py

```
1 start = 1
2 # for start in range(1,1001):
3 while start <= 8900:
4     url = f"https://www.imdb.com/search/title/?title_type=feature&user_rating=1.0,10.0&genres=sci-fi&count=250&start={start}&ref_=adv_nxt"
5     print(getTime()+':\t\x1b[4;36;40m' + url + '\x1b[0m')
6
7     start += 250
```

# Scraping Movies

Web-scraping - scrapingMoreMovies.py

```
1
2 movies = soup.find('div', class_="listner-list").find_all('div', class_="listner-item mode-advanced")
3 for movie in movies:
4     name = movie.find('h3', class_="listner-item-header").a.text
5     if (name in scrapedMovieNames):
6         print(getTime()+':\t\x1b[6;35;40mAlready Scraped:\t\t\t' + name + '...\x1b[0m') # purple on black
7         continue
8     print(getTime()+':\t\x1b[6;32;40m' + name + '\x1b[0m', name in scrapedMovieNames)# green on black
9
10    movieId= movie.find('h3', class_="listner-item-header").a.get('href').split("/")[2]
11    reviewLink=f"https://www.imdb.com/title/{movieId}/reviews?spoiler=hide&sort=userRating&dir=desc&ratingFilter=0"
12    source = requests.get(reviewLink)
13    source.raise_for_status()
14    soup = BeautifulSoup(source.text, 'html.parser')
15
16    scrapeReviews(reviewLink).to_csv("IMDB_reviews.csv", mode="a", index=False, header=headers)
17    headers=False
18    ## Movie details scraping
```

# Parsing and Scraping Reviews

```
Web-scraping- - scrapingMoreMovies.py

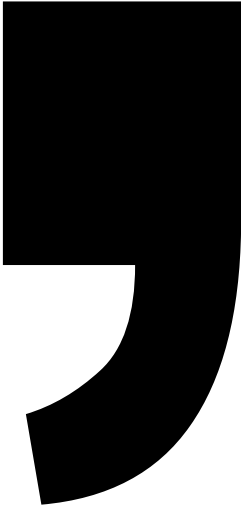
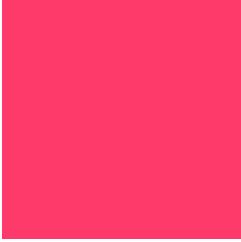
1 def scrapeReviews(url):
2     df= pd.DataFrame(columns=["movie", 'user', "rate", "title", "review", "url"])
3     # url = "https://www.imdb.com/title/tt9114286/reviews?spoiler=hide&sort=userRating&dir=desc&ratingFilter=0"
4     path= ""
5     while True:
6         source = requests.get(url)
7         source.raise_for_status()
8         soup = BeautifulSoup(source.text, 'html.parser')
9         dom = etree.HTML(str(soup))
10        if path == "":
11            path= dom.xpath('//div[@class="load-more-data"]/@data-ajaxurl')[0]
12            movieName=dom.xpath('//h3[@itemprop="name"]/a/text())[0]
13
14            user = dom.xpath('//span[@class="display-name-link"]/a/text()')
15            reviewRate = dom.xpath('//span[@class="point-scale"]/preceding-sibling::span/text()')
16            reviewTitle = dom.xpath('//a[@class="title"]/text()')
17            reviewContent = [revC.text for revC in dom.xpath('//div[contains(@class,"text show-more__control")]')]
18            reviewLink= [f"https://www.imdb.com{x}" for x in dom.xpath('//a[@class="title"]/@href')]
19
20        for a in range(0,len(reviewLink)-len(reviewRate)):
21            reviewRate.extend(["null"])
22
23        try:
24            paginationKey = dom.xpath('//div[@class="load-more-data"]/@data-key')[0]
25        except IndexError:
26            print(getTime()+':\t\\x1b[4;32;41m Finished ' + movieName + '\\x1b[0m')
27            break
28        url= f"https://www.imdb.com{path}&paginationKey={paginationKey}"
29        # print(movieName,url)
30        dff = pd.DataFrame({
31            "movie":movieName,
32            "user":user,
33            "rate":reviewRate,
34            "title":reviewTitle,
35            "review":reviewContent,
36            "url":reviewLink
37        })
38        df= pd.concat([df,dff], ignore_index=True,sort=False)
39    return df
```



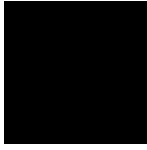
# **Data Visualisation**

# Data Visualisation For Cleaning

Since the end goal was to conduct the sentiment analysis we used a combination of graphs to better and faster understand the data we have to work with.



# Exploring Null Values



Title

Username

Rate

Review

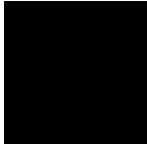
Content

1

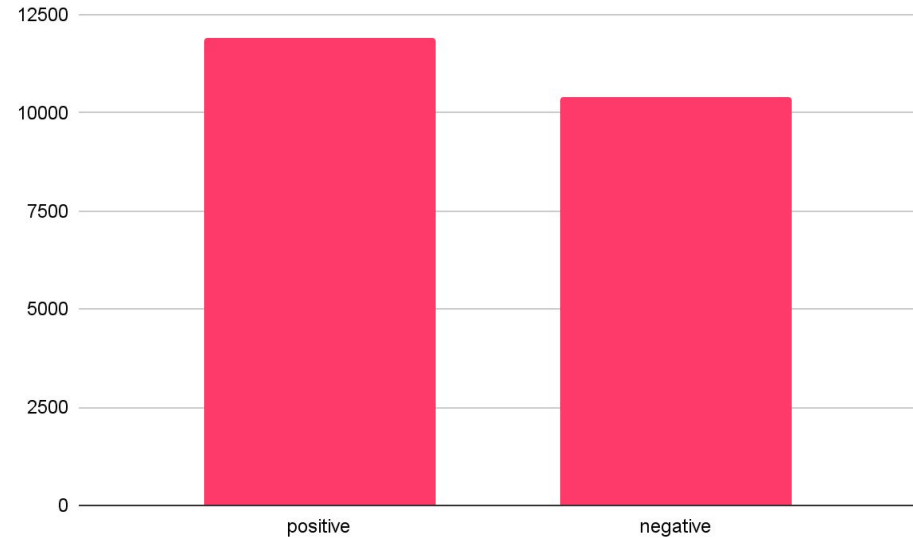
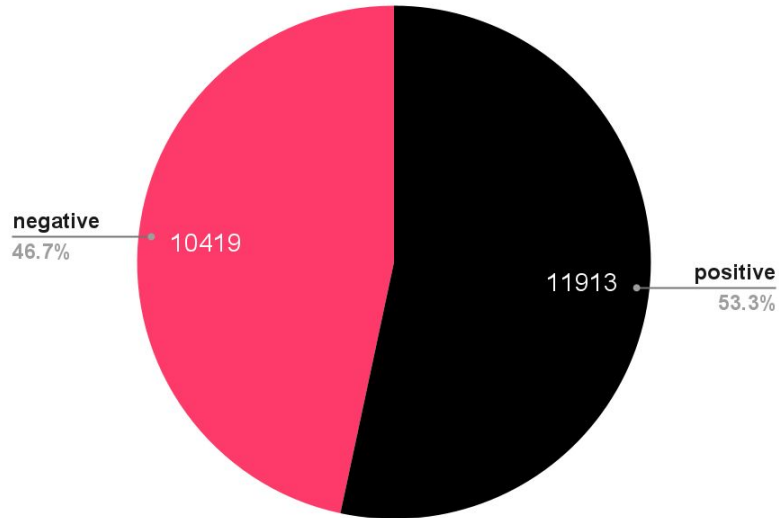


22342

# Understanding the Distribution of Sentiments



Distribution of Sentiment





# **Sentiment Analysis**



# Why is Sentiment Analysis difficult to perform?

A text may contain multiple sentiments all at once. For instance, “The acting was good , but the movie could have been better”

The above sentence consists of two polarities!!!



# F1 Score

$$F1\ Score = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)}$$

We use the harmonic mean (f1 score) instead of a simple average because it punishes extreme values.

**Precision:** Correct positive predictions relative to total positive predictions

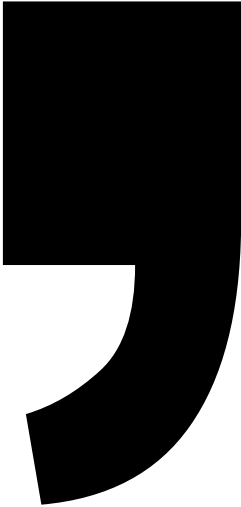
**Recall:** Correct positive predictions relative to total actual positives

# Multinomial Naïve Bayes Model

The multinomial naïve Bayes is widely used for assigning documents to classes based on the statistical analysis of their contents.

It provides an alternative to the "heavy" AI-based semantic analysis and drastically simplifies textual data classification.

It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.



# Multinomial Naïve Bayes Model

$$P(A|B) = P(A) * P(B|A)/P(B)$$

Where we are calculating the probability of class A when predictor B is already provided.

$P(B)$  = prior probability of B

$P(A)$  = prior probability of class A

$P(B|A)$  = occurrence of predictor B given class A probability

This formula helps in calculating the probability of the tags in the text.



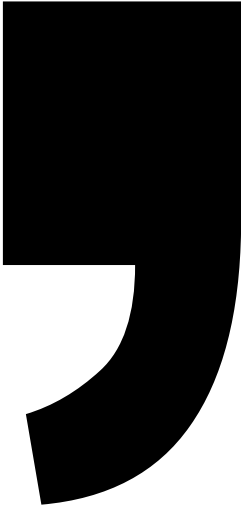
# VADER

(Valence Aware Dictionary for Emotional 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 sentiment 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.



# VADER

For example:

Words like **'love'**, **'enjoy'**, **'happy'**, **'like'** all convey a positive sentiment. It also understands the emphasis of capitalization and punctuation, such as **"ENJOY"**.

VADER is intelligent enough to understand the basic context of these words, such as "did not love" as a negative statement.

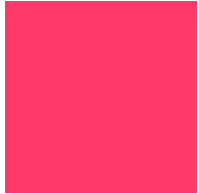




**Conclusion**



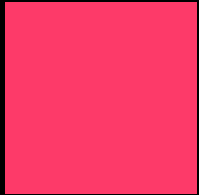
# Sources



What's Beautiful Soup



What's NLP



Multinomial Naïve Bayes



Vader





# Thank You

**GitHub**

Repo is private until nov 24

