INDIAN INSTITUTE OF TECHNOLOGY KANPUR

DEPARTMENT OF MATHEMATICS & STATISTICS

Data Science Lab - 2 (MTH-209A)

"Beyond the Surface: Deep Dive into Social Media with Machine Learning,Time Series & Text Analysis"

SUPERVISOR:

Dr. Subhajit Dutta

Submitted By:

Souhardya Mitra	(Roll. No. 231080090)
Soumen Konai	(Roll. No. 231080091)
Arnab Das	(Roll. No. 231080020)
Anirul Islam	(Roll. No.231080013)
Satyajit Das	(Roll. No. 220982)

Academic Year

2024

Abstract

This project delves into the multifaceted realm of social media data analysis, aiming to uncover hidden insights and trends that drive impactful decision-making. Our approach encompasses various methodologies and techniques, including exploratory data analysis (EDA) to navigate through the intricacies of social media data, unveiling patterns, trends, and hidden gems that lie within. We employ machine learning algorithms for sentiment analysis to classify social media posts based on sentiment (positive, negative, neutral), providing valuable insights into public perception and emotional responses. Predictive modeling techniques are utilized to develop models that forecast engagement metrics such as retweet count, reply count, like count, and quote count, aiding in strategic content planning. Furthermore, our exploration includes a diverse range of machine learning algorithms such as Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Logistic Regression, Naive Bayes, Random Forest, and Decision Tree, assessing their effectiveness in analyzing social media data and extracting meaningful insights. Through time series analysis and forecasting, we examine temporal patterns, peak engagement times, recurring themes, and emerging trends in social media activity, enabling proactive decision-making based on anticipated user behavior changes. This aspect of the project provides valuable insights into the dynamics of social media engagement over time, guiding strategic initiatives and enhancing decision-making processes.

Acknowledgment

Our journey in accomplishing this project has been enriched by the support and guidance of many individuals, to whom we are deeply grateful. We extend our heartfelt appreciation to Dr. Subhajit Dutta, from the Department of Mathematics and Statistics at IIT Kanpur, for entrusting us with this project.

This experience has not only been a tremendous learning opportunity but has also provided us with a practical application of the theoretical knowledge acquired during our academic journey.

We would also like to express our gratitude to our friends for their unwavering support throughout this endeavor. Their encouragement has been instrumental in enabling us to complete the project within the stipulated timeframe.

Souhardya Mitra

Soumen Konai

Arnab Das

Anirul Islam

Satyajit Das

Date: 06.04.2024

Contents

1 Introduction

In today's digital age, social media platforms have become invaluable sources of information and communication. With the vast amounts of data generated through social media interactions, businesses and researchers alike have the opportunity to gain insights into public sentiment, trends, and preferences. In this project, we embark on a journey to harness the power of machine learning techniques to analyze social media data, spanning text, time series, and beyond.

Our dataset comprises a diverse array of columns capturing various aspects of social media engagement, including metrics like retweet count, reply count, like count, and quote count, as well as textual features such as status text and emotional sentiment indicators. Additionally, temporal attributes such as date, time, and day provide valuable context for understanding the dynamics of social media activity.

We aim to explore a range of machine learning algorithms, including Support **Vector Machine (SVM), k-Nearest Neighbors (kNN), Logistic Regression, Naive Bayes, Random Forest, and Decision Tree,** to extract meaningful insights from the data. From our Status.text column after calculating some intersting metrices from the text will do some **Time Series** analysis to observe the pattern of sentiments day by day.

2 Source of the Data

We have collected the data from kaggle. published by DMO. The dataset consists of 21677 tweets posted from 25th March 2019 to 31st January 2022 by 23 Destination marketing organizations (DMO). The tweets were collected to study the social media content strategy of DMOs. Specifically, the research attempted to study how the linguistic features in social media content strategies of DMOs change during the pre-Covid, lockdown, and post-lockdown phases and their impact on social media user engagement. Accordingly, the data constitutes key variables like confidence and positive engagement expressed in a tweet, the amount of cognitive content embedded in a tweet, the media type, the number of hashtags, mentions, and word count of a tweet. The number of likes and retweets associated with each tweet is also captured. The dataset also captures each tweet's date and time stamp.

3 Data Description

Column Name	**Description**
`Status text`	This column contains the actual text of the status update or message posted by a user on a social media platform. It could include a wide range of content, such as thoughts, opinions, announcements, or links to other content.
`State`	This column represents the geographical location associated with the user who posted the status update. It is the state in a country, indicating where the user is based or where the event described in the status update occurred.
`Retweet count`	This column indicates the number of times the status update has been shared or retweeted by other users on the platform. Retweets are a common way for users to share interesting or noteworthy content with their own followers.
`Reply count`	This column represents the number of responses or comments the status update has received from other users. Replies often indicate engagement and interaction with the original post.
`Like count`	This column indicates the number of likes or favorites the status update has received from other users. Likes are a form of positive feedback and can indicate agreement or approval of the content.
`Quote count`	This column signifies the number of times the status update has been quoted by other users. Quoting allows users to share the original content while adding their own commentary or context.

Buzz	In the context of a tweet, ”buzz” refers to the level of attention, discussion, or activity surrounding a particular topic or tweet. It signifies the amount of engagement, retweets, likes, and overall interest generated by the tweet within the Twitter community..
Day	This column likely represents the date on which the status update was posted. It could help analyze trends in posting behavior over time.
Time	This column represents the time of day at which the status update was posted. It could be used to analyze posting patterns and engagement levels at different times of the day.
Followers	This column indicates the number of followers or subscribers the user who posted the status update has on the social media platform. It provides insight into the reach and potential audience of the status update.
Vividness	In a tweet, ”vividness” means the degree to which the language and content paint a clear, detailed, and engaging picture in the reader’s mind.. It could be calculated based on factors like the use of images, videos, or emotive language in the text.
WC(Word Count)	This column represents the word count of the status update. It provides a quantitative measure of the length of the text and can be used to analyze the complexity or depth of the content.
Clout	In the context of a tweet, ”clout” refers to the influence, power, or status that a user holds within the Twitter community. It pertains to the ability of a user to garner attention, engagement, and followers, often through their reputation, expertise, or popularity on the platform..
Cognition	In the context of a tweet, ”cognition” refers to the mental processes involved in understanding, interpreting, and responding to the content of the tweet. It encompasses how users perceive, process, and make sense of the information presented in the tweet, including aspects like comprehension, memory, and reasoning..
Sentiment Type	These column to represent the type of the sentiment within a text.Sentiment Type can be positive,negative,neutral.If the total of positive and negative sentiment is positive,negative,zero then it will be categorised as Positive,Negative &Neutral Sentiment Type respectively.

4 Motive of the project

Our objectives for this project are multifaceted. Here the objectives of our project are given below:

- **EDA:** Exploratory Data Analysis (EDA) is the art of uncovering hidden gems within data, transforming raw information into actionable insights. It's like exploring a treasure map, where each data point is a clue leading to a deeper understanding of the story behind the numbers. Through EDA, we illuminate patterns, unveil trends, and unravel mysteries, empowering us to make informed decisions and unlock the full potential of data-driven solutions. So, at first we will EDA to understand and visualize the whole data.
- **Sentiment Analysis:** Utilize machine learning algorithms to classify social media posts based on sentiment (positive, negative, neutral), providing insights into public perception and emotional responses.
- **Predictive Modeling:** Develop models to forecast engagement metrics such as retweet count, reply count, like count, and quote count based on the textual content of posts and other contextual features, aiding in predicting the popularity and impact of social media content.
- **Algorithm Exploration:** Explore a variety of machine learning algorithms including Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Logistic Regression, Naive Bayes, Random Forest, and Decision Tree, to assess their effectiveness in analyzing social media data and extracting meaningful insights.
- **Time Series Analysis & Forecasting:** Examine temporal patterns and trends in social media activity through time series analysis, identifying peak times of engagement, recurring themes, and emerging trends to inform marketing strategies and content scheduling. Apply time series forecasting model to predict future trends and patterns in social media engagement metrics, enabling proactive decision-making and strategy formulation based on anticipated changes in user behavior over time.
- **Actionable Insights:** Extract actionable insights from the data to inform marketing strategies, product development decisions, and public opinion analysis, demonstrating the value of machine learning in extracting actionable insights from social media data.

“Exploratory Data Analysis (EDA)”

1 State

1.1 Frequency plot of number of tweets statewise

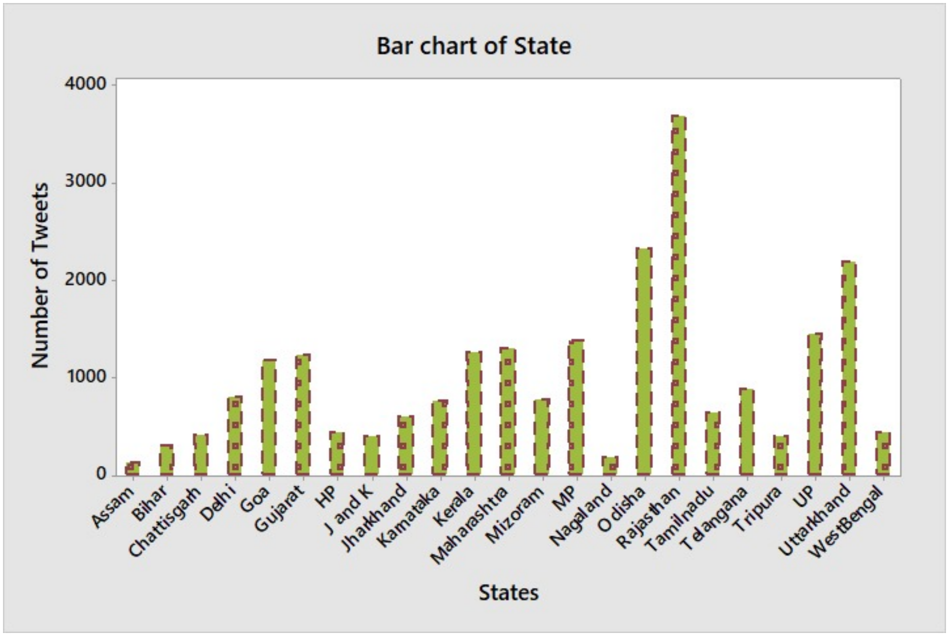
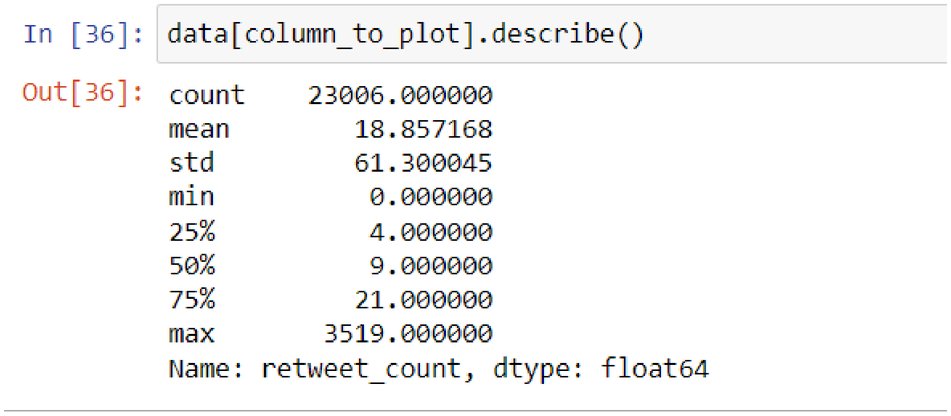


Figure 1: Bar plot of statewise tweets

The frequency bar plot shown in Figure 1 illustrates the distribution of tweets statewise. From the plot, we can observe that nearly 3600 tweets are made from Rajasthan holding the most number of tweets as a state and 122 tweets are uploaded from Assam, which is lowest. Bihar, Tripura and Nagaland also has relatively lower number of tweets and Odisha and Uttarakhand has relatively higher number of tweets.

2 Retweet Counts

2.1 Data summary



```
In [36]:  data[column_to_plot].describe()

Out[36]:  count    23006.000000
          mean        18.857168
          std         61.300045
          min          0.000000
          25%          4.000000
          50%          9.000000
          75%         21.000000
          max       3519.000000
          Name: retweet_count, dtype: float64
```

Figure 2: Summary of Retweet Count

The summary shown in Figure 2 tells us that retweet counts are spread with minimum value of 0 and a maximum value of 3519. It also shows that 50

2.2 Frequency plot of Retweet Counts

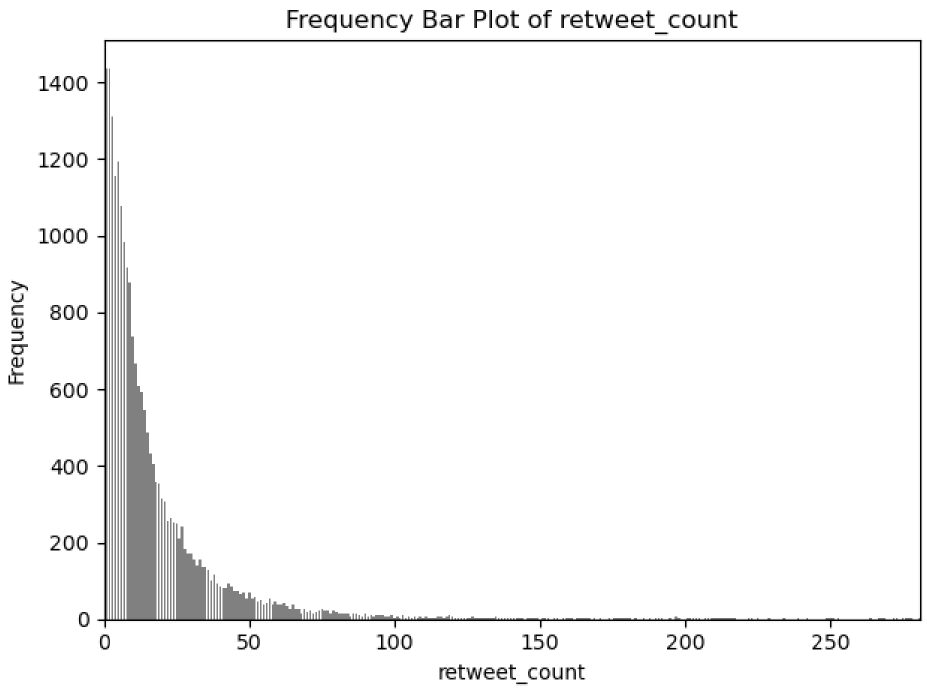


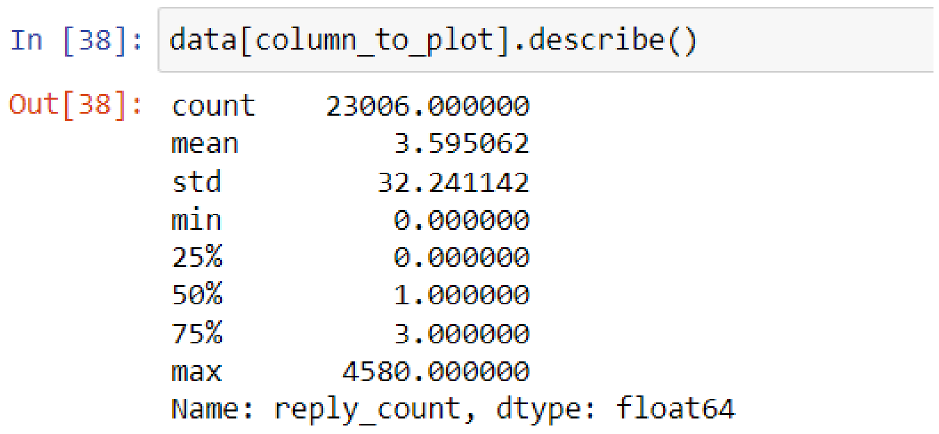
Figure 3: Frequency Bar Plot of Retweet Count

The frequency bar plot shown in Figure 3 illustrates the distribution of retweet counts. From the plot, we can observe that the most common retweet count ranges from 0 to 8, with fewer occurrences of higher retweet counts. This suggests that the majority of tweets in the dataset receive a relatively low number of retweets.

3 Reply Count

3.1 Data summary

The summary shown in Figure 4 tells us that reply counts are spread with minimum value of 0 and a maximum value of 4580. It also shows that 75



```
In [38]: data[column_to_plot].describe()

Out[38]: count    23006.000000
         mean         3.595062
         std         32.241142
         min          0.000000
         25%          0.000000
         50%          1.000000
         75%          3.000000
         max       4580.000000
         Name: reply_count, dtype: float64
```

Figure 4: Summary of Reply Count

3.2 Frequency plot of Reply Counts

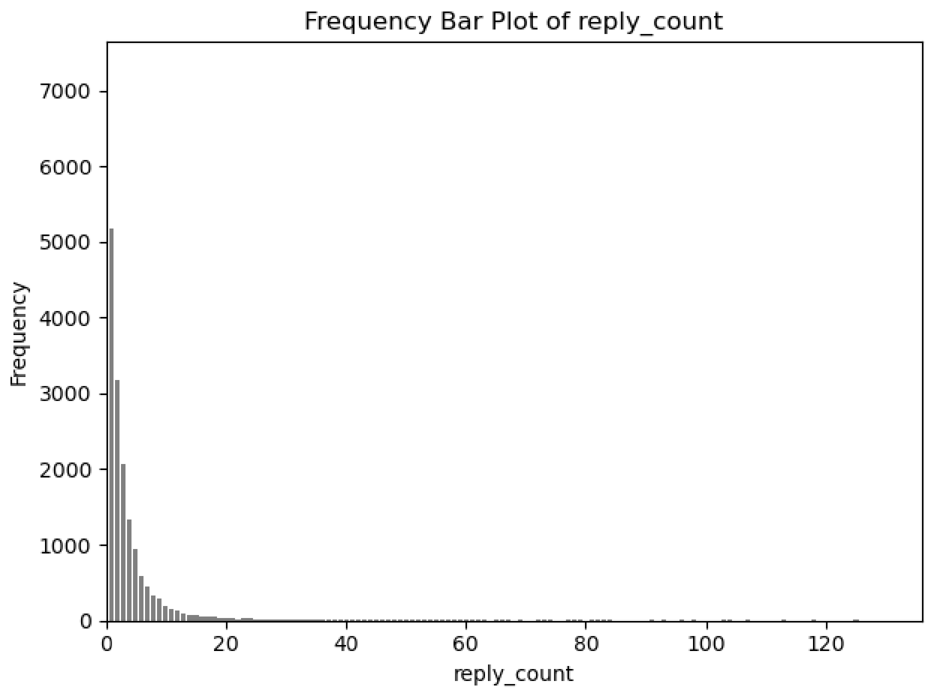
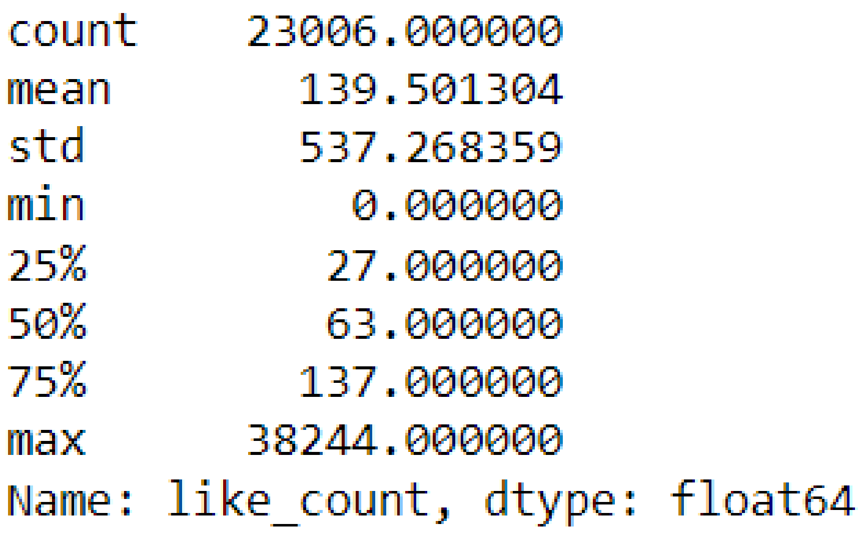


Figure 5: Frequency Bar Plot of Reply Count

The frequency bar plot shown in Figure 5 illustrates the distribution of reply counts. From the plot, we can observe that the most common reply count ranges from 0 to 1, with fewer occurrences of higher reply counts. This suggests that the mostly all tweets in the dataset receive a very low number of replies.

4 Like Count

4.1 Data summary



```
count    23006.000000
mean       139.501304
std        537.268359
min          0.000000
25%         27.000000
50%         63.000000
75%        137.000000
max      38244.000000
Name: like_count, dtype: float64
```

Figure 6: Summary of Like Count

The summary shown in Figure 6 tells us that like counts are spread with minimum value of 0 and a maximum value of 38244. It also shows that 75% of the data lies between 0 to 137, which indicates that the data is highly positively skewed data. Also the median of the data is 63 which tells us the tweets has relatively low numbers of likes.

4.2 Frequency plot of Like Counts

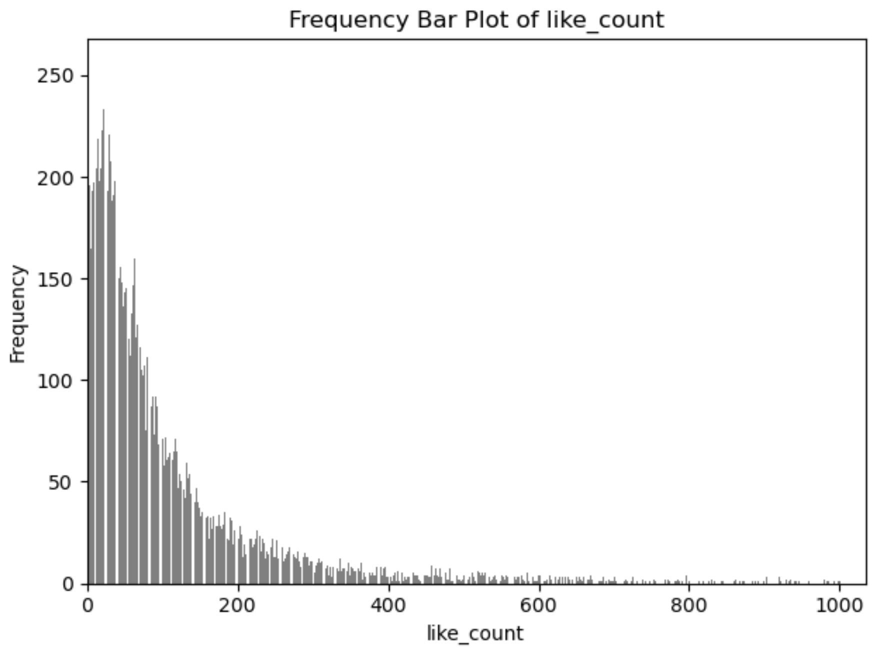
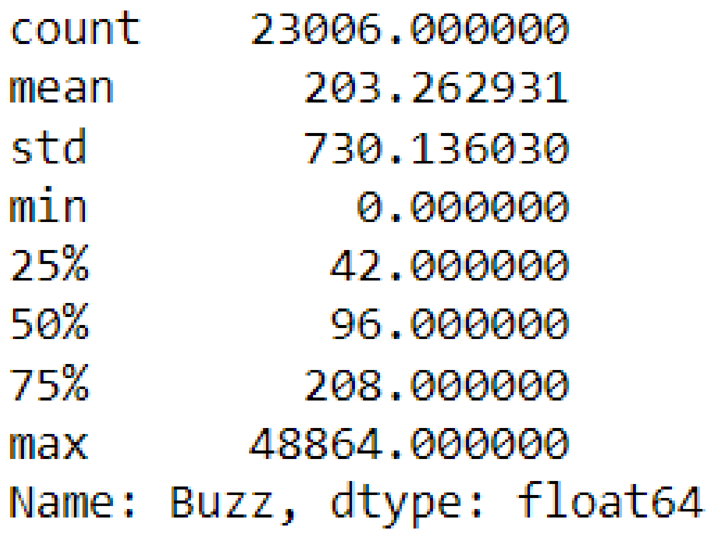


Figure 7: Frequency Bar Plot of Like Count

The frequency bar plot shown in Figure 7 illustrates the distribution of like counts. From the plot, we can observe that the most common like count ranges from 0 to 100, with fewer occurrences of higher like counts. This suggests that the mostly all tweets in the dataset receive a moderately low number of likes.

5 Buzz Count

5.1 Data summary



```
count    23006.000000
mean       203.262931
std        730.136030
min          0.000000
25%         42.000000
50%         96.000000
75%        208.000000
max      48864.000000
Name: Buzz, dtype: float64
```

Figure 8: Summary of Like Count

The summary shown in Figure 8 tells us that buzz counts are spread with minimum value of 0 and a maximum value of 48864. It also shows that 75

5.2 Frequency plot of Buzz

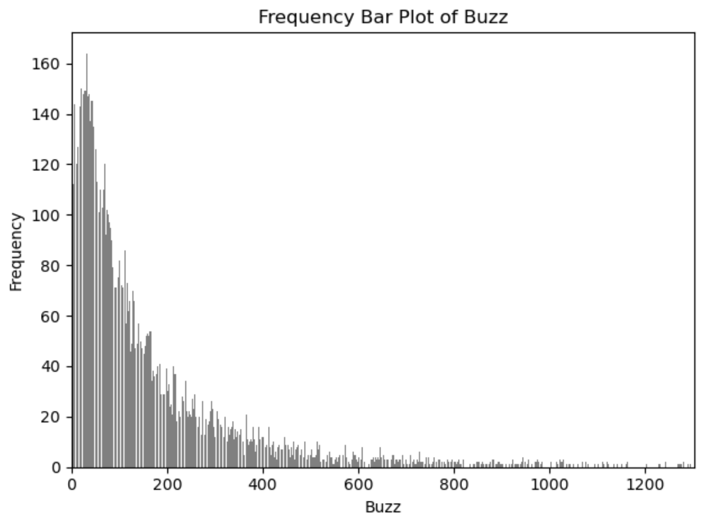


Figure 9: Frequency Bar Plot of Reply Count

The frequency bar plot shown in Figure 9 illustrates the distribution of buzz counts. From the plot, we can observe that the most common buzz count ranges from 0 to 96, with fewer occurrences of higher buzz counts.

6 Day type

6.1 Frequency bar plot of Day type

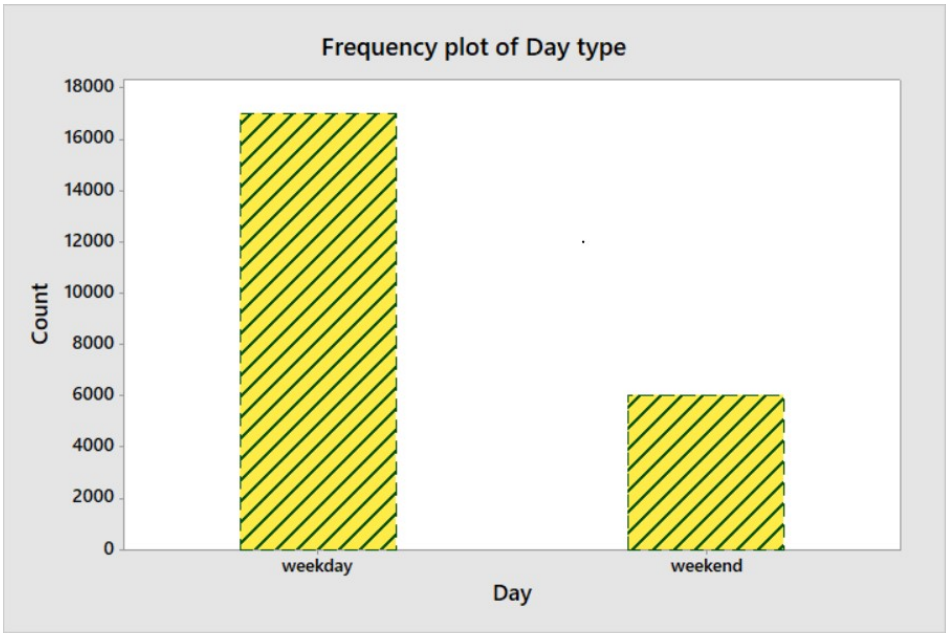


Figure 10: Frequency Bar Plot of Day type

The frequency bar plot shown in Figure 10 illustrates the distribution of Day. From the plot, we can observe that nearly 16990 tweets are made on weekdays and 6016 tweets are made on weekends. So it can be interpreted that these tweets are Business Promotions, News Updates, Professional Content, Events and Announcements, Community Engagement, Marketing Campaigns, Educational Content and Trending Topics.

7 Time

7.1 Frequency bar plot of Time type

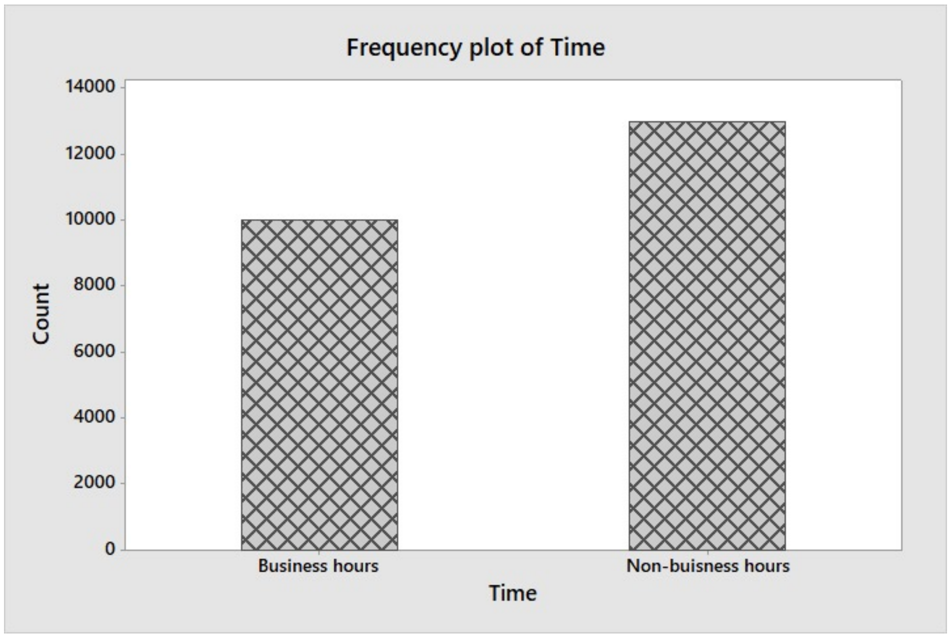


Figure 11: Frequency Bar Plot of Time type

The frequency bar plot shown in Figure 11 illustrates the distribution of Time of uploading the tweets. From the plot, we can observe that nearly 10000 tweets are made during business hours and 13000 tweets are made during non-business hours. So it can be interpreted that most tweets are from working professionals and uploaded during non-business hour.

8 Followers

8.1 Statewise folloers Bar plot

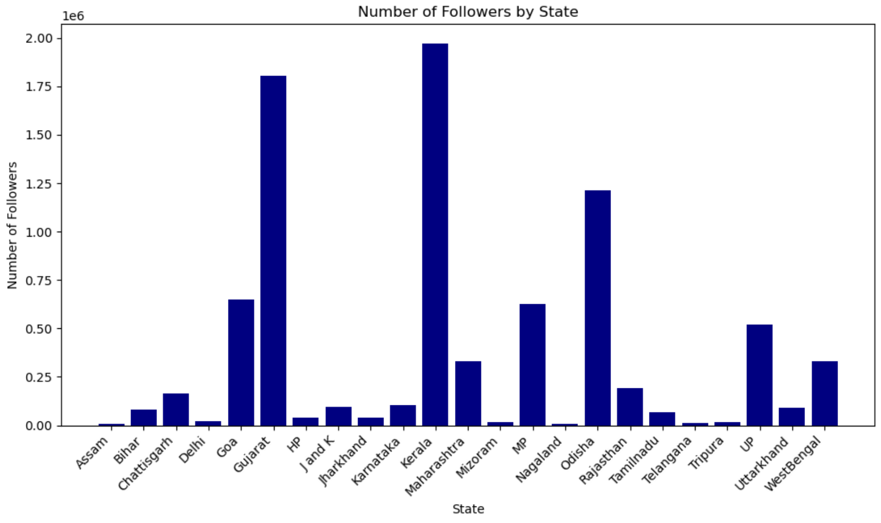


Figure 12: Bar Plot of Followers

The bar plot shown in Figure 12 illustrates the distribution of followers in that particular state. From the plot we can see that Assam, Nagaland, Telengana, Tripura, Delhi has very low number of followers compared to Kerala, Gujarat, Odisha.

9 Vividness

9.1 Pie-chart for vividness of tweets

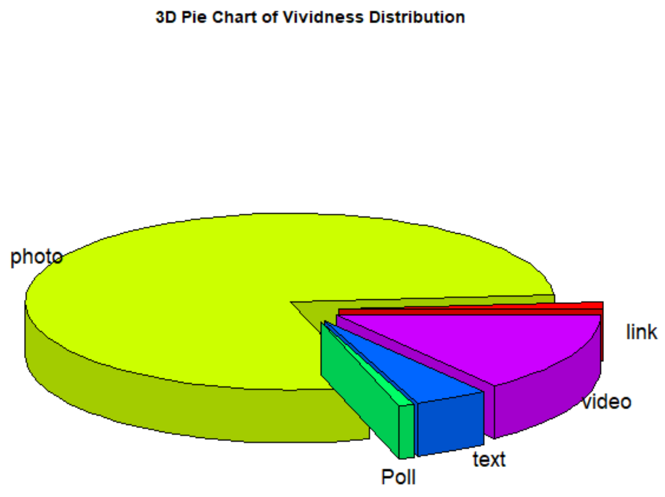


Figure 13: Pie chart for Vividness

The 3D pie plot shown in Figure 13 illustrates the distribution of vividness in tweets. From the plot we can see that most number of tweets contain photos and Poll and Links are keeping very lower presence in the tweets compared to Photos, Videos and Texts.

10 Word Count

10.1 Scatter plot of Tweets corresponding to word counts

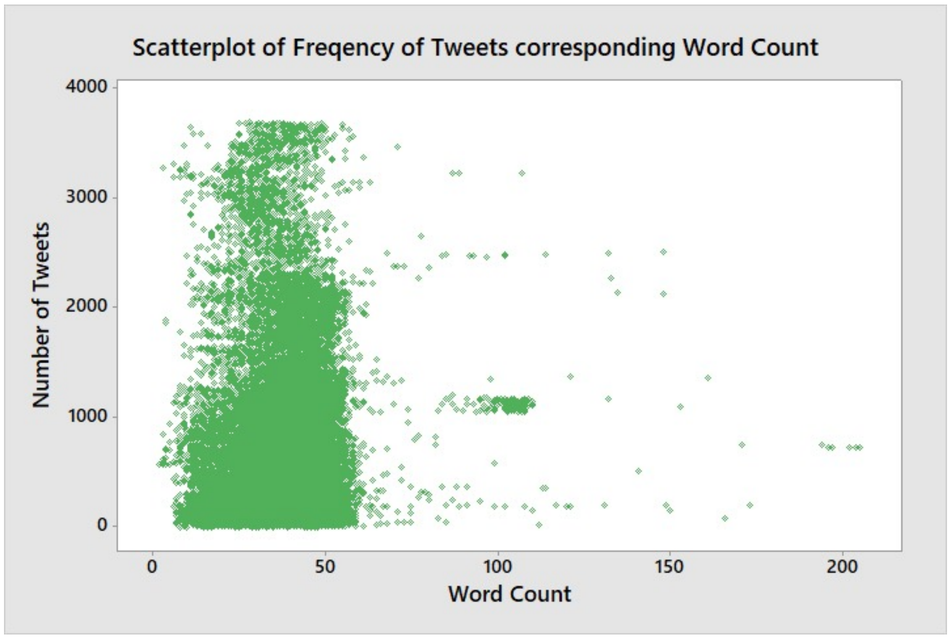


Figure 14: Scatterplot of Tweets vs Word count

The scatterplot shown in Figure 14 illustrates the distribution of tweets for different values of Word count. From the plot we can see that most number of tweets contain lesser than 60 word counts. Even very few, like 5-6 tweets have word count near about 200.

11 Clout

11.1 Histogram of Clout

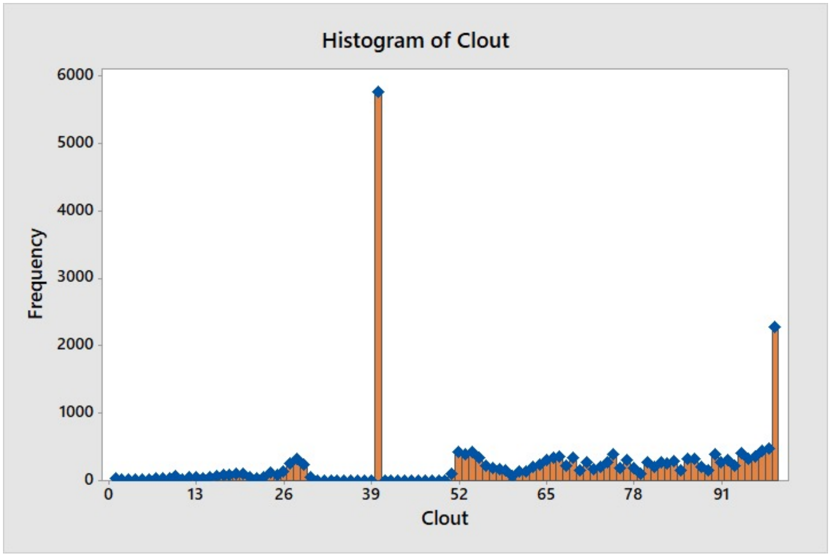


Figure 15: Histogram of Clout

A histogram of "clout" in Figure 15 provides a visual representation of the distribution of this variable. It reveals patterns of central tendency and outliers. Analyzing the histogram we can see maximum number of clout happened around 40. Clout indicates the influence, power, or importance of a context. Here we have a moderately high frequency of clout score 99 which indicates our dataset contains moderately high volume of important tweets.

12 Cognition

12.1 Histogram of Cognition

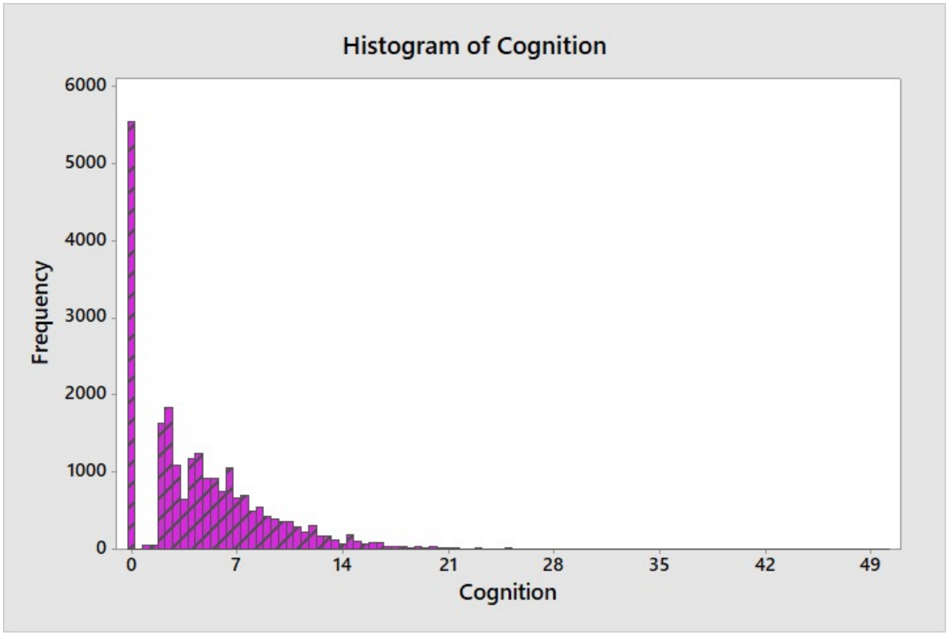


Figure 16: Histogram of Cognition

A histogram of "cognition" in Figure 16 provides a visual representation of the distribution of this variable. Analyzing the histogram we can see maximum number of cognition occurred around 0. Also the highly positively skewed Cognition indicates low amount of mental processes involved in understanding and interpreting the sentiment expressed in tweets.

13 Emotion

13.1 Plot of emotion, emotion positive and emotion negative

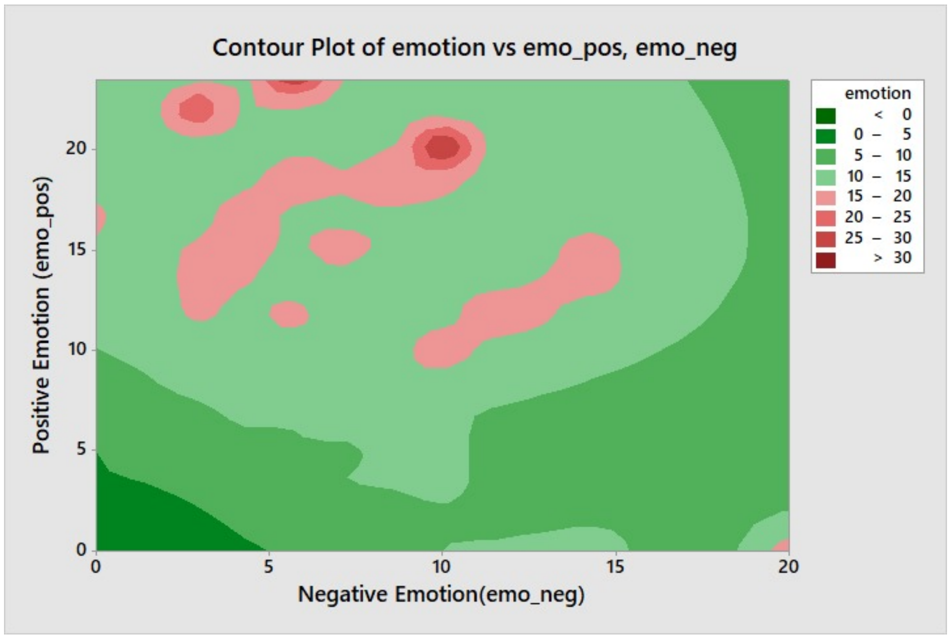


Figure 17: Contour plot of Emotion, Positive emotion and Negative emotion

The contour plot of ”Emotion”, with ”Emotion Positive” and ”Emotion Negative” as the x and y variables in Figure 17, respectively, and ”emotion” as the z variable, provides a visual representation of the relationship between positive and negative emotions and the overall emotional intensity or valence. It shows the intensity, valence, balance, distribution, and interactions of emotions across the spectrum of positive and negative emotions. The plot tells us that higher values of positive emotion is well associated with overall emotion and most of the cases overall emotion is around 10-15.

14 Sentiment Type

14.1 Bar plot of Sentiment Type

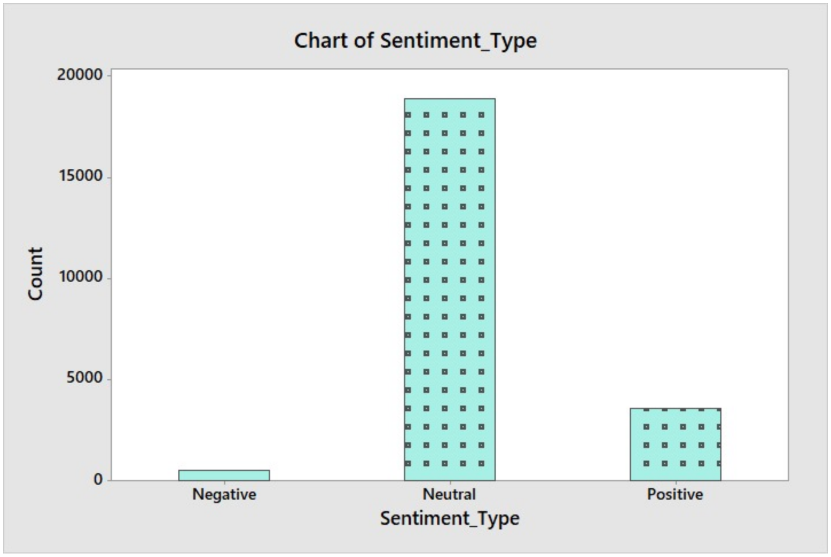


Figure 18: Frequency bar plot of Sentiment Type

The dominance of neutral sentiments in my dataset from Figure 18 suggests that sentiment expressions are evenly distributed, with most of the content neither strongly positive nor negative. This implies that my data covers a diverse range of topics or perspectives without a clear bias towards emotional language. While this objectivity may be beneficial for conveying factual information, it presents challenges for sentiment analysis algorithms, which may struggle to accurately classify neutral sentiments.

15 Cognition and Emotion vs Sentiment

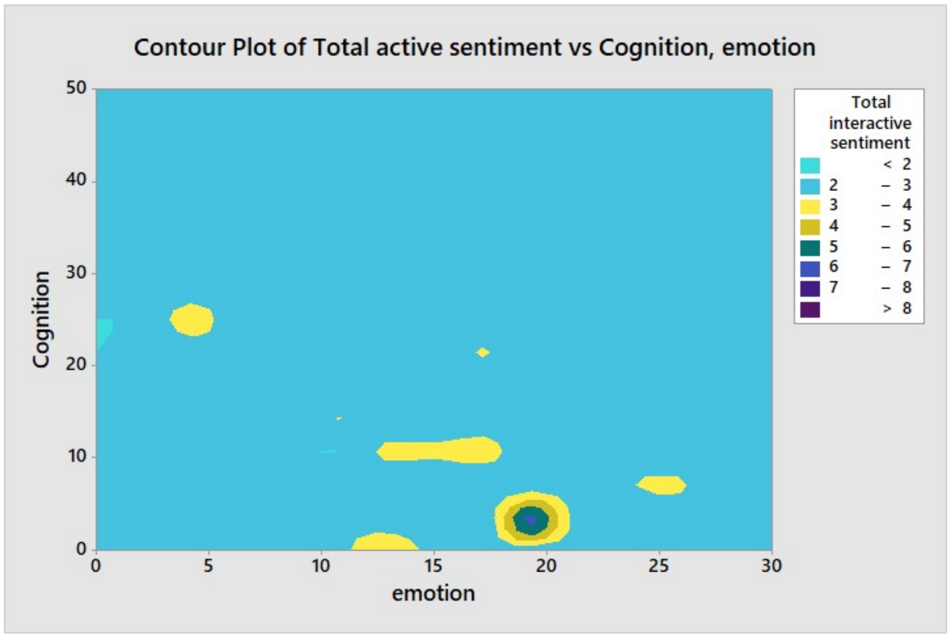


Figure 19: Contour plot of Cognition, Emotion and Sentiment

The contour plot in Figure 19 of cognition and emotion with ”Cognition” as the y variable, ”emotion” as the x variable, and ”sentiment” as the z variable, provides a visual representation of the relationship between cognitive processes, emotional states, and sentiment expression. It illustrates how variations in cognitive factors such as perception, attention, and reasoning, as well as emotional factors such as arousal and valence, influence the overall sentiment conveyed in a dataset. The contour lines depict regions of different sentiment intensities, revealing patterns such as clusters of positive or negative sentiment associated with specific combinations of cognitive and emotional states. The plot shows that Sentiment (Sum of Absolute values of positive and negative sentiments) is majorly associated with higher emotion and lower cognition associated tweets.

“Twitter Sentiment Analysis Using Naive Bayes Model”

Twitter Sentiment Analysis Using Naive Bayes Model

April 5, 2024

1 Introduction

This section of the report provides an overview of the application of Naive Bayes model in analyzing sentiment on Twitter. The analysis aims to categorize tweets as positive, negative, or neutral based on the sentiment expressed in the text. Naive Bayes is chosen as the classification model due to its efficiency and effectiveness in handling large volumes of text data. The use of sentiment analysis on social media platforms like Twitter has gained significant attention in recent years. Understanding the sentiment of tweets can provide valuable insights for various purposes such as brand monitoring, public opinion analysis, and market research. In this report, we delve into the methodology of applying the Naive Bayes model to Twitter sentiment analysis, discuss the preprocessing steps involved, feature extraction techniques, and conclude with the significance of this approach in analyzing twitter data.

2 Methodology

2.1 Data Collection

Data was collected from the website Mendeley.com

2.2 Data description

The dataset includes 21,677 tweets shared by 23 Destination Marketing Organizations (DMOs) from March 25, 2019, to January 31, 2022. These tweets were collected to examine how DMOs' social media content strategies evolved over time, particularly before, during, and after the COVID-19 pandemic lockdowns. The study aims to understand how changes in language usage in DMOs' social media content impacted user engagement. The dataset comprises several key variables, including the tone of tweets (confidence and positive engagement), the level of cognitive content, media type used, number of hashtags and mentions, and tweet length. Additionally, data on likes and retweets for each tweet are recorded. To categorize tweets into different phases (pre-COVID, during lockdown, and post-COVID), timestamps were utilized, aligning with the lockdown phases declared by the Government of India.

2.3 Preprocessing

Sentiment analysis aims to assess the overall sentiment expressed in a text. Before analyzing sentiment, the text undergoes preprocessing. This process includes eliminating unnecessary words, breaking the text into individual units (tokenization), removing common words that do not carry much meaning (stopwords), and reducing words to their base form (stemming). Each word in the text is associated with a sentiment. The sentiment of each word is determined, and then the total sentiment of the text is calculated. If the total sentiment is positive (greater than zero), the overall sentiment is considered positive. Conversely, if the total sentiment is negative (less than zero), the overall sentiment is negative. If the total sentiment is zero, the overall sentiment is neutral.

2.3.1 Removing Irrelevant Words

- Irrelevant symbols, tags like URLs, and punctuation marks are removed from the text.

2.3.2 Tokenization

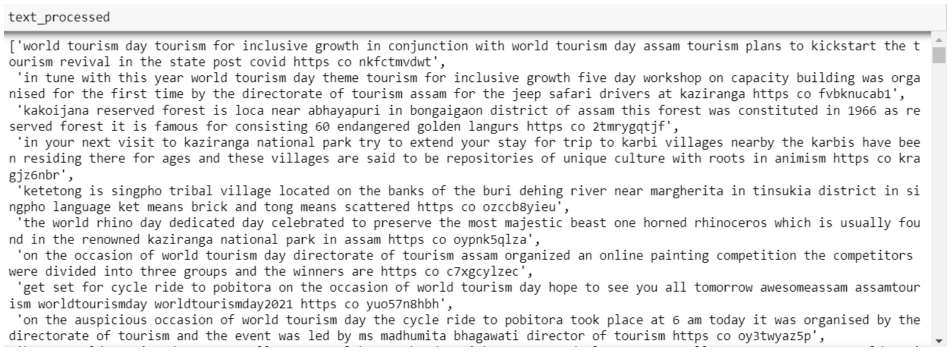
- Tokenization involves breaking down the text into smaller units such as words or terms, known as tokens. This processes text units are shown in Figurefig:process

2.3.3 Removing Stopwords

- Common words in English, known as stopwords, are removed from the text as they do not contribute significantly to the meaning of the sentence.

2.3.4 Stemming

- Stemming is applied to reduce words to their root form by removing prefixes and suffixes, ensuring consistency in sentiment analysis.



```
text_processed
['world tourism day tourism for inclusive growth in conjunction with world tourism day assam tourism plans to kickstart the t
ourism revival in the state post covid https co nkfctmvdwt',
 'in tune with this year world tourism day theme tourism for inclusive growth five day workshop on capacity building was orga
nised for the first time by the directorate of tourism assam for the jeep safari drivers at kaziranga https co fvbknucab1',
 'kakoijana reserved forest is loca near abhayapuri in bongaigaon district of assam this forest was constituted in 1966 as re
served forest it is famous for consisting 60 endangered golden langurs https co 2tmrygqtjf',
 'in your next visit to kaziranga national park try to extend your stay for trip to karbi villages nearby the karbis have bee
n residing there for ages and these villages are said to be repositories of unique culture with roots in animism https co kra
gjz6nbr',
 'ketetong is singpho tribal village located on the banks of the buri dehing river near margherita in tinsukia district in si
ngpho language ket means brick and tong means scattered https co ozccb8yieu',
 'the world rhino day dedicated day celebrated to preserve the most majestic beast one horned rhinoceros which is usually fou
nd in the renowned kaziranga national park in assam https co oypnk5qlza',
 'on the occasion of world tourism day directorate of tourism assam organized an online painting competition the competitors
were divided into three groups and the winners are https co c7xgcylzec',
 'get set for cycle ride to pobitora on the occasion of world tourism day hope to see you all tomorrow awesomeassam assamtour
ism worldtourismday worldtourismday2021 https co yuo57n8hbh',
 'on the auspicious occasion of world tourism day the cycle ride to pobitora took place at 6 am today it was organised by the
directorate of tourism and the event was led by ms madhumita bhagawati director of tourism https co oy3twyaz5p',
```

Figure 1: Processed text units

2.4 Feature Extraction

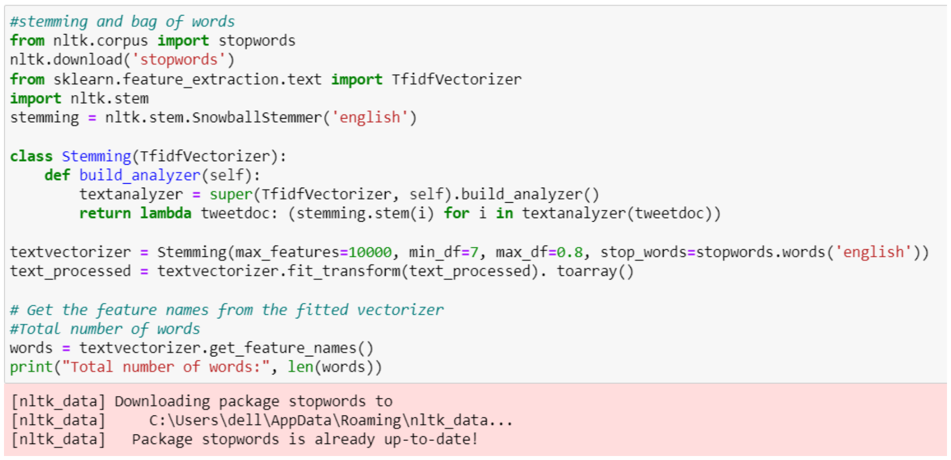
Stemming is a process aimed at reducing each word to its root form. One common algorithm for stemming is the Snowball Stemmer, which is supported by NLTK (Natural Language Toolkit). To handle preprocessing and tokenization of words, the build_analyzer method is used. A lambda function is created within this method, which utilizes the stem() function from the Snowball Stemmer to stem the words.

After stemming, the words are converted into a vector representation known as the Bag of Words. This creates a dictionary of all words present in the document, capturing their occurrences. The importance of each word is then determined using the Term Frequency-Inverse Document Frequency (TF-IDF) model.

TF-IDF measures the importance of a word based on its frequency in the document compared to its frequency across all documents. In Python, TF-IDF values can be calculated using the TfidfVectorizer() function from the "scikit-learn" library.

In the TfidfVectorizer() function, the max_features parameter is set to 10000, indicating that only the 10000 most frequently occurring words are used to create the Bag of Words. The max_df parameter is set to an 80% threshold, meaning words occurring in 80% of the documents are included. Similarly, the min_df parameter is set to a threshold value of 7, ensuring that words occurring in at least eight documents are included in the Bag of Words.

Additionally, predefined stopwords in English are utilized as a reference to remove common stopwords from the Bag of Words.



```
#stemming and bag of words
from nltk.corpus import stopwords
nltk.download('stopwords')
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk.stem
stemming = nltk.stem.SnowballStemmer('english')

class Stemming(TfidfVectorizer):
    def build_analyzer(self):
        textanalyzer = super(TfidfVectorizer, self).build_analyzer()
        return lambda tweetdoc: (stemming.stem(i) for i in textanalyzer(tweetdoc))

textvectorizer = Stemming(max_features=10000, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
text_processed = textvectorizer.fit_transform(text_processed). toarray()

# Get the feature names from the fitted vectorizer
#Total number of words
words = textvectorizer.get_feature_names()
print("Total number of words:", len(words))
```

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

Figure 2: Stemming and bag of words code chunk

2.5 Naive Bayes Classification

2.5.1 Feature Representation

The feature representation involves converting the preprocessed text data into a numerical format suitable for machine learning algorithms. In the case of Twitter sentiment analysis using Naive Bayes, this typically involves creating a bag of words representation. Let n denote the total number of unique words in the entire dataset. Each tweet i is represented as a feature vector $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{in})$, where x_{ij} represents the frequency of occurrence of the j-th word in the i-th tweet. The feature vectors for all tweets are then stacked to form a feature matrix $\mathbf{X}$, where each row corresponds to a tweet and each column corresponds to a word.

2.5.2 Training

During the training phase, the Naive Bayes classifier estimates the probabilities of each word occurring in tweets belonging to each sentiment class (positive, negative, or neutral). Let N_c denote the total number of tweets in the training set belonging to class c, $N_{c,j}$ denote the number of times word j occurs in tweets of class c, and V denote the size of the vocabulary (total number of unique words). The probability $P(x_{ij}|c)$ of word j occurring in tweets of class c is estimated as:

$$P(x_{ij}|c) = \frac{N_{c,j} + 1}{N_c + V}$$

This estimation accounts for Laplace smoothing to avoid zero probabilities.

2.5.3 Classification

To classify a new tweet, the Naive Bayes classifier calculates the posterior probability of each sentiment class given the feature vector of the tweet using Bayes' theorem:

$$P(c|\mathbf{x}) = \frac{P(c)\prod_{j=1}^{n} P(x_{ij}|c)}{\sum_{c'} P(c')\prod_{j=1}^{n} P(x_{ij}|c')}$$

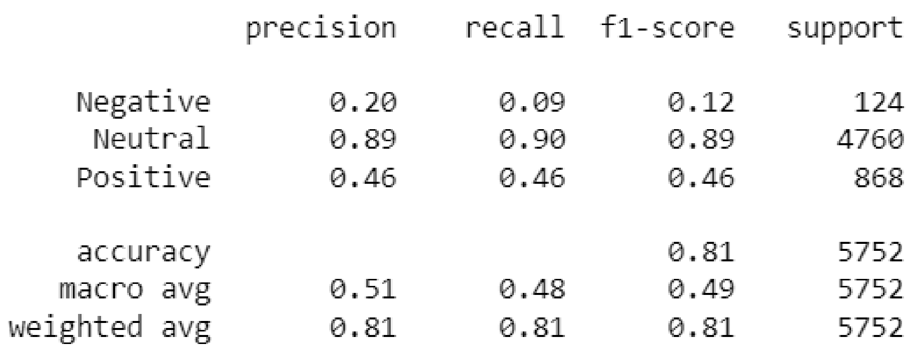
where:

- $P(c)$ is the prior probability of class c,
- $P(x_{ij}|c)$ is the likelihood of word j given class c,
- The denominator is a normalization factor ensuring that the probabilities sum up to 1 over all classes.

The class with the highest posterior probability is assigned to the tweet.

2.6 Evaluation

- The performance of the Naive Bayes classification model is evaluated using standard evaluation metrics such as accuracy, precision, recall, and F1-score.
- This evaluation involves comparing the predicted sentiment labels of tweets with the true labels in the test dataset.
- For visualization, evaluation and interpretation purpose three meaningful visualizations are given below



```
              precision    recall  f1-score   support

    Negative       0.20      0.09      0.12       124
     Neutral       0.89      0.90      0.89      4760
    Positive       0.46      0.46      0.46       868

    accuracy                           0.81      5752
   macro avg       0.51      0.48      0.49      5752
weighted avg       0.81      0.81      0.81      5752
```

Figure 3: Naive Bayes accuracy matrix

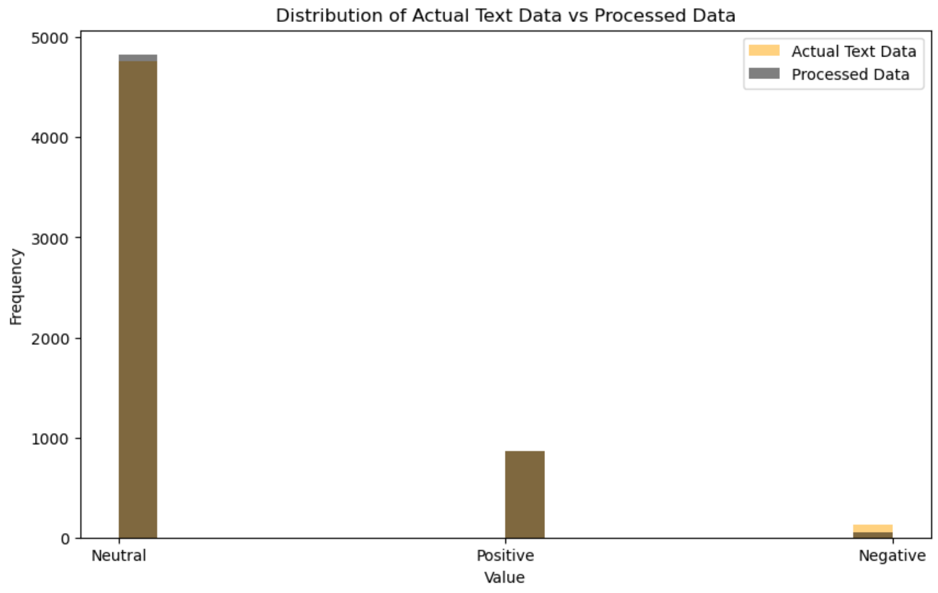


Figure 4: Naive Bayes accuracy matrix

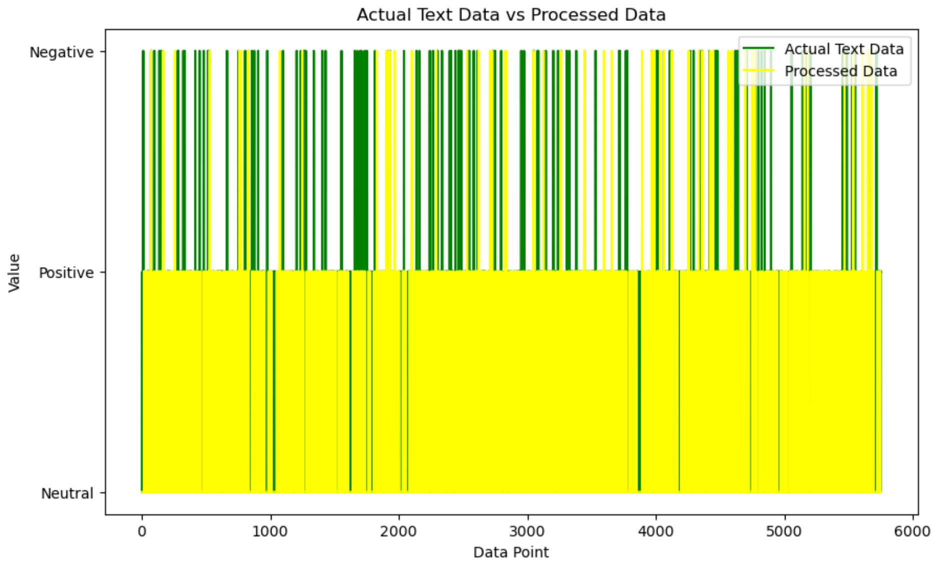


Figure 5: Naive Bayes accuracy matrix

2.7 Results Interpretation

- Here in Figure 3, our **Naive Bayes** model shows us overall accuracy is **81 percent**, which is quite good for classifying sentiments. Support of negative sentiments is too low and that may be one of the major reason for reduced precision.

- Now Figure 4 shows that histograms of all the sentiments are well overlapped but the frequency of positive and negative sentiments are too low to interpret in a directive way. Again the histograms exhibit similar shapes and distributions, it suggests that the processing steps have preserved the overall characteristics of the original text data.

- Finally line diagram in Figure 5 shows that actual textual sentiments are quite good fit for both positive and neutral sentiments but negative sentiments are lacking a bit to fit, which is indicated by the frequency of green lines in the diagram.

- There are notable deviations between the actual text data and the processed data in negative sentiment part in Figure 5. It suggests that the Naive Bayes model may have introduced alterations or biases during processing. Such discrepancies could indicate issues with feature extraction and encoding.

2.8 Limitations

2.8.1 Assumption of Conditional Independence

The Naive Bayes model assumes that the features (words) are conditionally independent given the class label. However, in natural language, words often exhibit dependencies and correlations, leading to oversimplified modeling.

2.8.2 Sensitivity to Feature Quality

Since the model relies heavily on the quality of features (words), it may perform poorly if the features do not adequately capture the semantics or context of the text. This is especially problematic when dealing with noisy or ambiguous text data.

2.8.3 Lack of Contextual Understanding

Naive Bayes treats each word independently, disregarding the sequential and contextual information present in the text. As a result, the model may struggle to understand nuances, sarcasm, negation, or sentiment expressed through word combinations or phrases.

2.8.4 Vulnerability to Overfitting

Despite its simplicity, Naive Bayes can still suffer from overfitting, especially when dealing with high-dimensional feature spaces or when the training data is insufficient or noisy. Which is the case here for negative sentiment.

3 Conclusion

Naive Bayes is a popular algorithm for sentiment analysis on Twitter data due to its simplicity and effectiveness. In our study, we utilized Naive Bayes to classify tweets into positive, negative, or neutral sentiments based on the words used in the text. The algorithm assumes that the presence of each word in a tweet is independent of the presence of other words, hence the term "naive". Despite this simplification, Naive Bayes often performs well in practice, particularly for text classification tasks like sentiment analysis.

Our results indicate that Naive Bayes achieved promising accuracy rate(**81 percent**) in classifying sentiment in Twitter data. By leveraging a labeled dataset for training, the algorithm learned to identify patterns and associations between words and sentiment labels. However, we observed challenges in handling noisy or ambiguous language commonly found in tweets, such as slang, abbreviations, and misspellings. As our dataset containing mainly tourism advertising and professsional tweets, we lack of negative sentiment in our tweets causing negative sentiment precision droped at 20 percent.

Furthermore, Naive Bayes demonstrated a tendency to classify tweets with neutral sentiments accurately, as they often contain common words with less emotional connotations.

In conclusion, Naive Bayes presents a viable approach for sentiment analysis on Twitter data, offering a balance between simplicity and effectiveness. While it excels in classifying tweets with neutral sentiments, further refinement and feature extraction engineering may be necessary to improve its performance on tweets with more nuanced or complex emotional expressions. For that purpose we studied various Machine learning classification algorithms with more number of predictors to neutralize the difficulties faced while Naive Bayes modelling.

“Different Machine Learning Models to Predict Sentiment Type”

1 Detailed Description of Columns

1.1 Column Descriptions

Column Name	**Description**
`Status text`	This column contains the actual text of the status update or message posted by a user on a social media platform. It could include a wide range of content, such as thoughts, opinions, announcements, or links to other content.
`State`	This column represents the geographical location associated with the user who posted the status update. It is the state in a country, indicating where the user is based or where the event described in the status update occurred.
`Retweet count`	This column indicates the number of times the status update has been shared or retweeted by other users on the platform. Retweets are a common way for users to share interesting or noteworthy content with their own followers.
`Reply count`	This column represents the number of responses or comments the status update has received from other users. Replies often indicate engagement and interaction with the original post.
`Like count`	This column indicates the number of likes or favorites the status update has received from other users. Likes are a form of positive feedback and can indicate agreement or approval of the content.
`Quote count`	This column signifies the number of times the status update has been quoted by other users. Quoting allows users to share the original content while adding their own commentary or context.
`Buzz`	In the context of a tweet, ”buzz” refers to the level of attention, discussion, or activity surrounding a particular topic or tweet. It signifies the amount of engagement, retweets, likes, and overall interest generated by the tweet within the Twitter community..
`Day`	This column likely represents the date on which the status update was posted. It could help analyze trends in posting behavior over time.
`Time`	This column represents the time of day at which the status update was posted. It could be used to analyze posting patterns and engagement levels at different times of the day.
`Followers`	This column indicates the number of followers or subscribers the user who posted the status update has on the social media platform. It provides insight into the reach and potential audience of the status update.
`Vividness`	In a tweet, ”vividness” means the degree to which the language and content paint a clear, detailed, and engaging picture in the reader’s mind.. It could be calculated based on factors like the use of images, videos, or emotive language in the text.
`WC(Word Count)`	This column represents the word count of the status update. It provides a quantitative measure of the length of the text and can be used to analyze the complexity or depth of the content.

`Clout`	In the context of a tweet, ”clout” refers to the influence, power, or status that a user holds within the Twitter community. It pertains to the ability of a user to garner attention, engagement, and followers, often through their reputation, expertise, or popularity on the platform..
`Cognition`	In the context of a tweet, ”cognition” refers to the mental processes involved in understanding, interpreting, and responding to the content of the tweet. It encompasses how users perceive, process, and make sense of the information presented in the tweet, including aspects like comprehension, memory, and reasoning..
`Sentiment Type`	These column to represent the type of the sentiment within a text.Sentiment Type can be positive,negative,neutral.If the total of positive and negative sentiment is positive,negative,zero then it will be categorised as Positive,Negative &Neutral Sentiment Type respectively.

1.2 Engagement Ratio Feature

The `engagement_ratio` feature has been calculated as the ratio of the total engagement (sum of retweet count, reply count, and like count) to the number of followers of the user who posted the status update. Mathematically, it is represented as:

$$\text{engagement\_ratio} = \frac{\text{retweet\_count} + \text{reply\_count} + \text{like\_count}}{\text{Followers}}$$

where:

- retweet_count represents the number of times the status update has been retweeted.
- reply_count represents the number of replies or comments the status update has received.
- like_count represents the number of likes or favorites the status update has received.
- Followers represents the number of followers or subscribers the user who posted the status update has on the social media platform.

1.3 Classification Performance Matrix

In a classification problem, we often use a confusion matrix to evaluate the performance of a model. Let's define a confusion matrix for a binary classification problem:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

where:

- TP (True Positive): The number of instances that are actually positive and are predicted by the model as positive.
- TN (True Negative): The number of instances that are actually negative and are predicted by the model as negative.

- FP (False Positive): The number of instances that are actually negative but are predicted by the model as positive (Type I error).
- FN (False Negative): The number of instances that are actually positive but are predicted by the model as negative (Type II error).

1.3.1 Precision

Precision measures the accuracy of positive predictions. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the classifier:

$$\text{Precision} = \frac{TP}{TP + FP}$$

1.3.2 Recall

Recall measures the ability of the classifier to find all the positive samples. It is defined as the ratio of true positive predictions to the total number of actual positive instances:

$$\text{Recall} = \frac{TP}{TP + FN}$$

1.3.3 F1-Score

F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is calculated as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.3.4 Accuracy

Accuracy measures the overall correctness of the model:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

It represents the ratio of correctly classified instances to the total instances.

1.3.5 Macro Average

Macro average calculates the metric independently for each class and then takes the average across all classes. For precision, recall, and F1-score:

$$\text{Macro Average} = \frac{1}{N} \sum_{i=1}^{N} \text{Metric}_i$$

where N is the number of classes.

1.3.6 Weighted Average

Weighted average calculates the metric for each class and then takes the weighted average, where the weight is the support (the number of true instances for each label). For precision, recall, and F1-score:

$$\text{Weighted Average} = \frac{\sum_{i=1}^{N} \text{Metric}i \times \text{Support}_i}{\sum i = 1^N \text{Support}_i}$$

where N is the number of classes.

2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a powerful dimensionality reduction technique widely used in data analysis and machine learning. It aims to transform high-dimensional data into a lower-dimensional space while preserving as much of the variability in the original data as possible.

2.1 Purpose

The primary purpose of PCA is to simplify complex datasets by reducing the number of features (or dimensions) while retaining the most important information. By reducing the dimensionality, PCA can help in visualizing data, identifying patterns, and improving computational efficiency in subsequent analysis tasks.

2.2 Methodology

PCA works by finding the orthogonal axes (principal components) along which the variance of the data is maximized. Let $\mathbf{X}$ be the data matrix with n observations and p features. The steps involved in PCA are as follows:

2.2.1 Standardization

Standardize the data to have a mean of 0 and a standard deviation of 1 (optional but recommended) to ensure that all features contribute equally to the analysis.

$$\mathbf{X}^* = \frac{\mathbf{X} - \mu}{\sigma}$$

where μ is the mean vector and σ is the standard deviation vector.

2.2.2 Covariance Matrix

Compute the covariance matrix $\mathbf{C}$ of the standardized data:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

where $\mathbf{X}^{*T}$ denotes the transpose of the standardized data matrix.

2.2.3 Eigendecomposition

Perform eigendecomposition on the covariance matrix $\mathbf{C}$ to obtain the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p$. Sort the eigenvalues in descending order.

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

2.2.4 Select Principal Components

Select the top k eigenvectors corresponding to the largest eigenvalues to form the principal components matrix $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$.

2.2.5 Projection

Project the original data onto the new lower-dimensional space defined by the selected principal components:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}^* \mathbf{V}_k$$

The projected data matrix $\mathbf{X}_{\text{proj}}$ contains the observations represented in terms of the principal components.

2.3 Application

PCA has various applications in data analysis, including:

- Data visualization: PCA can help visualize high-dimensional data in two or three dimensions, making it easier to identify patterns and clusters.
- Feature extraction: PCA can be used to extract the most important features from a dataset, reducing noise and redundancy.
- Compression: PCA can compress data by representing it in a lower-dimensional space, reducing storage and computational requirements.
- Preprocessing: PCA can be used as a preprocessing step before applying other machine learning algorithms to improve their performance and efficiency.

2.4 Brief Description

In this report, we have provided a comprehensive overview of Principal Component Analysis (PCA), detailing its methodology, purpose, and applications. We have discussed the steps involved in PCA, including standardization, computation of the covariance matrix, eigendecomposition, selection of principal components, and projection of the data onto the principal components. PCA is a fundamental technique in dimensionality reduction and plays a crucial role in simplifying complex datasets, visualizing data, and improving computational efficiency in various data analysis tasks. This documentation aims to serve as a comprehensive reference guide for individuals seeking to apply PCA in their data analysis projects.

2.4.1 Observation of Negligible Correlation in Principal Components (PCs

The observation of negligible correlation between principal components (PCs) is indicative of their orthogonality and linear independence. This property is highly desirable in Principal Component Analysis (PCA) as it suggests that each principal component provides unique and independent information about the data.

2.4.2 Multicollinearity Effect and Orthogonality

Multicollinearity Effect:

Multicollinearity occurs when independent variables in a regression model are highly correlated with each other. This can lead to unstable coefficient estimates and difficulties in interpreting the model.

Orthogonality in PCA:

In PCA, principal components are constructed to be orthogonal to each other, meaning they are uncorrelated. This ensures that each component captures unique variance in the data, without redundancy.

2.4.3 Importance of Orthogonality in PCA

Independent Information:

When principal components are orthogonal, they provide independent information about the data. This means that each component explains a distinct aspect of the variability in the dataset.

Reduced Redundancy:

Orthogonal principal components reduce redundancy in the information captured by the components. They focus on different directions of variance in the data, allowing for more efficient dimensionality reduction.

2.4.4 Justification from Correlation Coefficients

The observation of very low correlation coefficients between different principal components in the correlation matrix supports the notion of orthogonality. Low correlation coefficients indicate that the principal components are nearly independent of each other, providing justification for their use as independent sources of information.

2.4.5 Practical Implications

Improved Interpretability:

Orthogonal principal components facilitate easier interpretation of the data structure, as each component captures distinct patterns without overlapping information.

Enhanced Model Stability:

The absence of multicollinearity among principal components improves the stability and reliability of models built using these components.

2.4.6 Conclusion

In summary, the negligible correlation coefficients between principal components in PCA indicate their orthogonality and independence. This property is valuable as it ensures that each component

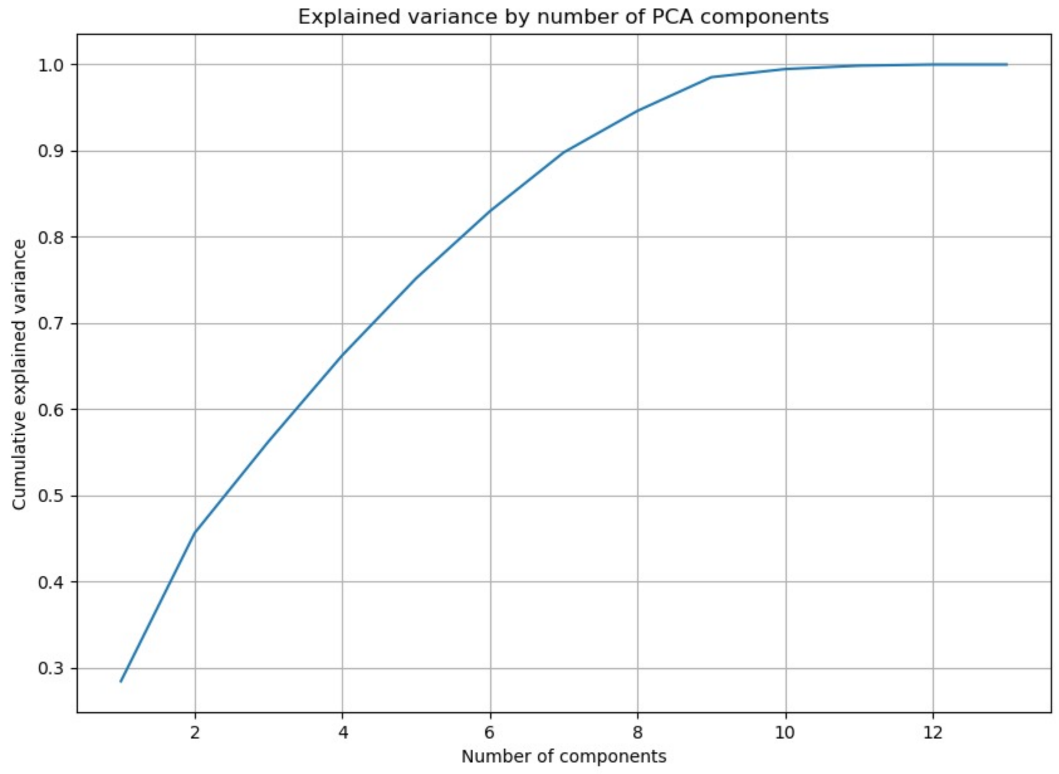


Figure 1: Changes in Variation with respect to the PCA components

contributes unique information to the analysis, leading to improved interpretability and model stability. The observation of low correlations provides justification for the use of principal components as independent sources of information in dimensionality reduction and data analysis tasks.

1. **Principal Component 1 (PC1)**:

   PC1 is the first Principal component that explains 28.420% of the total variability.

2. **Principal Component 2 (PC2)**:

   PC2 is the second Principal component that explains 17.214% of the total variability.

3. **Principal Component 3 (PC3)**:

   PC3 is the third Principal component that explains 10.619% of the total variability.

4. **Principal Component 4 (PC4)**:

   PC4 is the fourth Principal component that explains 10.029% of the total variability.

5. **Principal Component 5 (PC5)**:
   PC5 is the fifth Principal component that explains 8.928% of the total variability.
6. **Principal Component 6 (PC6)**:
   PC6 is the sixth Principal component that explains 7.789% of the total variability.
7. **Principal Component 7 (PC7)**:
   PC7 is the Seventh Principal component that explains 6.823% of the total variability.
8. **Principal Component 8 (PC8)**:
   PC8 is the eighth Principal component that explains 4.825% of the total variability.
9. **Principal Component 9 (PC9)**:
   PC9 is the last component that explains 3.906% of the total variability.

These interpretations provide valuable insights into the underlying patterns and characteristics represented by each principal component that helps us to understand the driving variability in the dataset.

2.4.7 Highlight Correlations

We generated a heatmap of the correlation matrix for the Prinipal components as shown in Figure2. The correlation coefficient between two different principal components are almost negligible that means we are able to deduct the dependencies of the explanatory variables.Actually PCA removes the multicollinearity effect from the dataset.Now it can be assumed that all the pca's are now linearly independent,so we can actually interpret all of the components clearly.

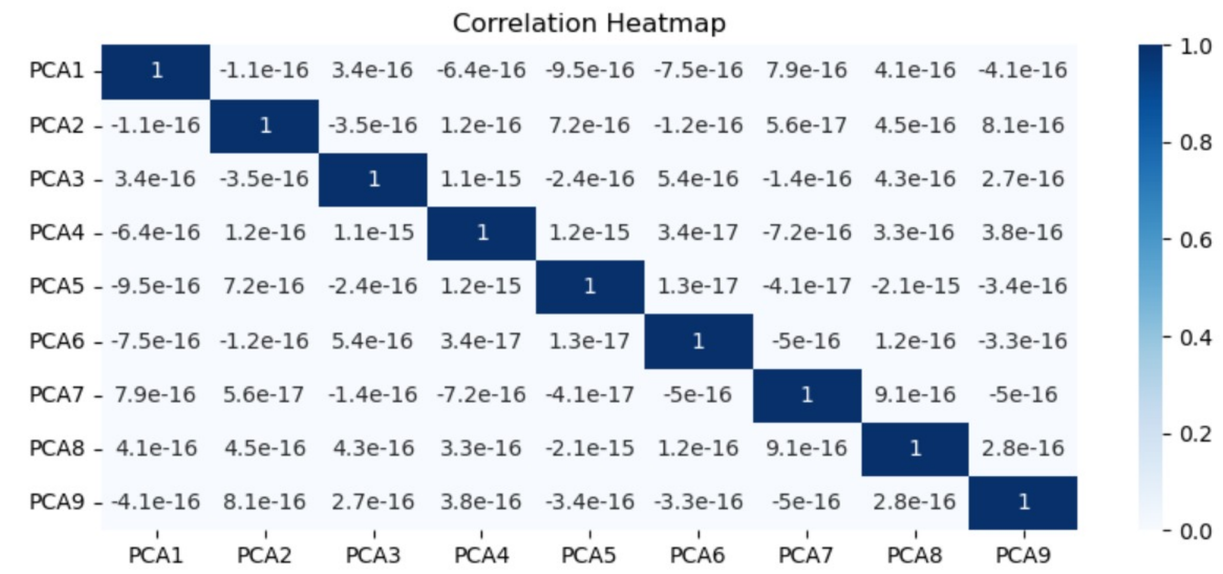


Figure 2: Heatmap Showing correlation between the Principal components

3 Logistic Regression

3.1 When the response variable has binary category0,1 / yes,no etc.

3.1.1 Introduction to Logistic Regression

Logistic regression is a statistical method used for modeling the relationship between a binary dependent variable and one or more independent variables. It is widely employed in various fields such as medicine, economics, social sciences, and machine learning.

Unlike linear regression, which predicts continuous outcomes, logistic regression is specifically designed for binary classification problems. In other words, it is used when the dependent variable is categorical and has only two possible outcomes (e.g., yes/no, 1/0, pass/fail).

The primary objective of logistic regression is to estimate the probability that a given observation belongs to a particular category or class based on the values of the independent variables. This makes logistic regression a powerful tool for predicting binary outcomes and understanding the factors that influence them.

3.1.2 Model Evaluation

After estimating the parameters, statistical tests can be conducted to assess the significance of each coefficient, and measures such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) can be used to evaluate the overall goodness-of-fit of the model. Additionally, the performance of the model can be evaluated using metrics such as accuracy, precision, recall, and ROC curve analysis.

3.1.3 Purpose

1. To model the relationship between a binary dependent variable and multiple independent variables.
    - Multiple logistic regression is used when we want to understand how the probability of a binary outcome (such as success/failure, yes/no) is affected by two or more independent variables.
    - It allows us to quantify the impact of each predictor variable on the probability of the event occurring.
2. To predict the probability of a particular outcome based on the values of predictor variables.
    - Once the model is built, it can be used to predict the probability of the dependent variable being in a particular category given the values of the independent variables.

3.1.4 Mathematical Background

logistic regression is a statistical method used to model the relationship between a binary dependent variable and multiple independent variables. The logistic regression model is based on the logistic function, also known as the sigmoid function, which maps any real-valued number to the range between 0 and 1, making it suitable for modeling probabilities. The logistic function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The logistic regression equation for multiple predictors is formulated as follows:

$$p(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k)}}$$

where:

- $p(Y = 1|X)$ is the probability of the dependent variable being 1 given the values of the independent variables X.
- $\beta_0, \beta_1, \ldots, \beta_k$ are the coefficients representing the strength and direction of the relationship between each independent variable and the log-odds of the dependent variable.
- $X_1, X_2, \ldots, X_k$ are the independent variables.

The logistic regression model assumes that the log-odds of the dependent variable being in category 1 (success) versus category 0 (failure) is a linear combination of the independent variables. Mathematically, this can be expressed as:

$$\log\left(\frac{p(Y = 1|X)}{1 - p(Y = 1|X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

This equation is known as the logit transformation, where the log odds of the probability is transformed linearly in terms of the independent variables.

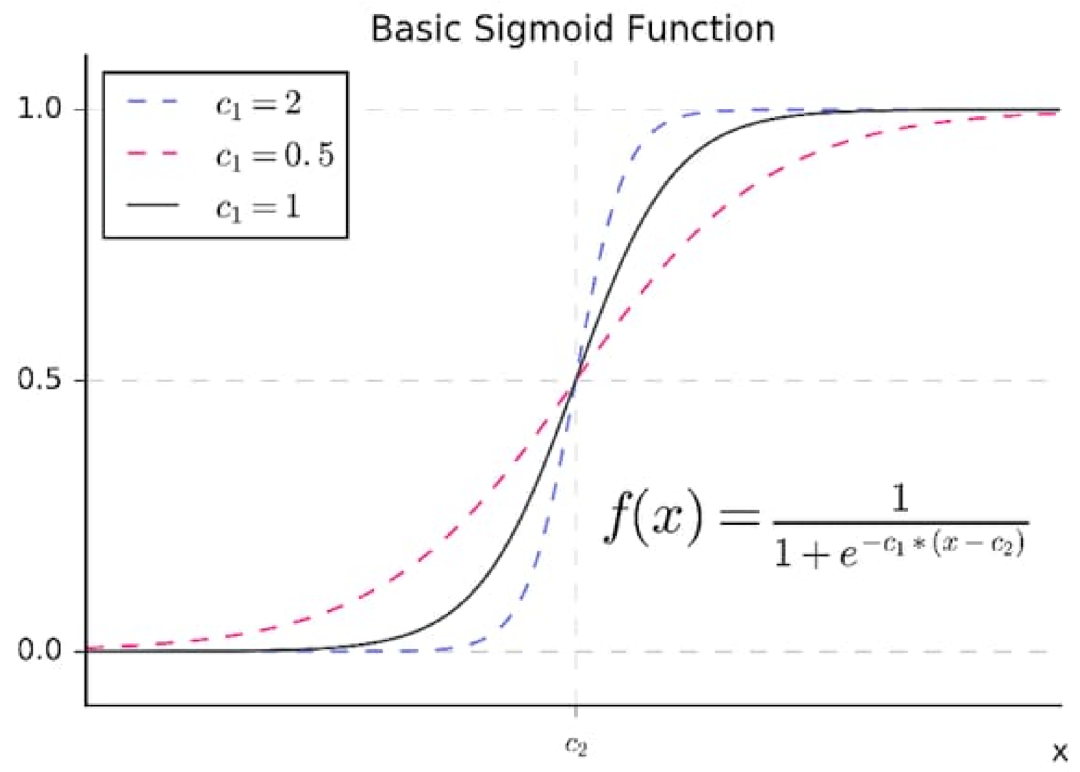


Figure 3: Sigmoid Curve

The logistic regression model is typically fitted using Maximum Likelihood Estimation (MLE), which involves maximizing the likelihood function. The likelihood function for logistic regression is derived from the probability mass function of the Bernoulli distribution. Given n observations (Y_i, X_i), the likelihood function is defined as the product of the probabilities of observing the given outcomes given the predictor variables and model parameters:

$$L(\beta_0, \beta_1, \ldots, \beta_k) = \prod_{i=1}^{n} p(Y_i | X_i; \beta_0, \beta_1, \ldots, \beta_k)$$

where $p(Y_i | X_i; \beta_0, \beta_1, \ldots, \beta_k)$ is the probability of observing the outcome Y_i given the predictor variables X_i and model parameters $\beta_0, \beta_1, \ldots, \beta_k$.

The log-likelihood function is then obtained by taking the natural logarithm of the likelihood function:

$$\ell(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=1}^{n} (Y_i \log(p_i) + (1 - Y_i) \log(1 - p_i))$$

where $p_i = p(Y_i = 1|X_i; \beta_0, \beta_1, \ldots, \beta_k)$ is the predicted probability of the i-th observation being in category 1.

The goal of logistic regression is to find the values of $\beta_0, \beta_1, \ldots, \beta_k$ that maximize the log-likelihood function. This optimization problem is typically solved using numerical optimization algorithms such as gradient descent or Newton-Raphson method.

After estimating the parameters, statistical tests can be conducted to assess the significance of each coefficient, and measures such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) can be used to evaluate the overall goodness-of-fit of the model.

3.1.5 Methodology

(a) Collect data with a binary dependent variable and multiple independent variables.

- Ensure that the dependent variable represents a binary outcome, such as presence/absence, success/failure, etc.
- Collect data on multiple independent variables that could potentially influence the outcome.

(b) Fit the logistic regression model using maximum likelihood estimation.

- Maximum Likelihood Estimation (MLE) is a statistical method used to estimate the parameters of a model by maximizing the likelihood function.
- In logistic regression, the likelihood function is defined as the product of the probabilities of observing the given outcomes (binary responses) given the predictor variables and model parameters.
- The log-likelihood function for multiple logistic regression is:

$$\ell(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=1}^{n} (y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

  where y_i is the observed outcome for the i-th observation, p_i is the predicted probability of the outcome being 1 for the i-th observation, and n is the total number of observations.
- The goal is to find the values of $\beta_0, \beta_1, \ldots, \beta_k$ that maximize the log-likelihood function.
- This optimization problem is typically solved using numerical optimization algorithms such as gradient descent or Newton-Raphson method.

(c) Assess the significance of coefficients and goodness-of-fit of the model.

- After estimating the parameters, statistical tests (e.g., Wald test, likelihood ratio test) can be conducted to assess the significance of each coefficient.
- Additionally, measures such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) can be used to evaluate the overall goodness-of-fit of the model.

(d) Validate the model using techniques like cross-validation.

- Split the data into training and testing sets.

- Fit the model on the training data and evaluate its performance on the testing data.
- Use techniques like k-fold cross-validation to assess the model's stability and generalization ability.

3.1.6 Analysis

(a) Estimate the coefficients $\beta_0, \beta_1, \ldots, \beta_k$.

- Use statistical software (e.g., R, Python) to estimate the coefficients of the logistic regression model based on the available data.

(b) Interpret the coefficients to understand the effect of each independent variable on the probability of the dependent variable.

- Positive coefficients indicate that an increase in the independent variable is associated with an increase in the log-odds (or probability) of the dependent variable.
- Negative coefficients indicate the opposite relationship.
- The magnitude of the coefficient represents the strength of the association.

(c) Use the model to predict the probability of the dependent variable being 1 for new observations.

- Once the model is built and validated, it can be used to predict the probability of the dependent variable being in a particular category for new observations with known values of the independent variables.

3.2 Multinomial Logistic Regression(When the response variable has more than two categories present)

3.2.1 Introduction

Multinomial logistic regression is an extension of binary logistic regression that allows for the prediction of categorical outcomes with more than two categories. It is commonly used when the dependent variable has multiple unordered categories. Unlike binary logistic regression, which is used for binary classification tasks, multinomial logistic regression can handle multiple classes simultaneously.

3.2.2 Overview

In multinomial logistic regression, the dependent variable Y can take on K different categories. The goal is to model the probabilities of each category given the values of the independent variables $X_1, X_2, \ldots, X_k$. The model estimates the probability of each category relative to a reference category, which is typically chosen arbitrarily. The probabilities for all categories sum up to 1 for each observation.

3.2.3 Mathematical Formulation

The multinomial logistic regression model is formulated using the softmax function, which generalizes the logistic function for multiple categories. The softmax function is defined as follows:

$$P(Y=k|X) = \frac{e^{\beta_{0k}+\beta_{1k}X_1+\ldots+\beta_{pk}X_p}}{\sum_{j=1}^{K} e^{\beta_{0j}+\beta_{1j}X_1+\ldots+\beta_{pj}X_p}}$$

where:

- $P(Y=k|X)$ is the probability of the dependent variable being in category k given the values of the independent variables $X_1, X_2, \ldots, X_p$.
- $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ are the coefficients associated with category k.
- K is the total number of categories.

The softmax function ensures that the predicted probabilities sum up to 1 across all categories for each observation.

3.2.4 Parameter Estimation

Similar to binary logistic regression, the parameters of the multinomial logistic regression model are estimated using Maximum Likelihood Estimation (MLE). The likelihood function for multinomial logistic regression is the product of the probabilities of observing the given outcomes given the predictor variables and model parameters.

The log-likelihood function is then obtained by taking the natural logarithm of the likelihood function. The goal is to find the values of $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ that maximize the log-likelihood function. This optimization problem is typically solved using numerical optimization algorithms such as gradient descent or Newton-Raphson method.

3.2.5 Advantages of Multinomial Logistic Regression

- Allows for the prediction of categorical outcomes with more than two categories.
- Provides interpretable coefficients representing the effect of each independent variable on the probability of each category relative to the reference category.
- Can handle multicollinearity among independent variables.

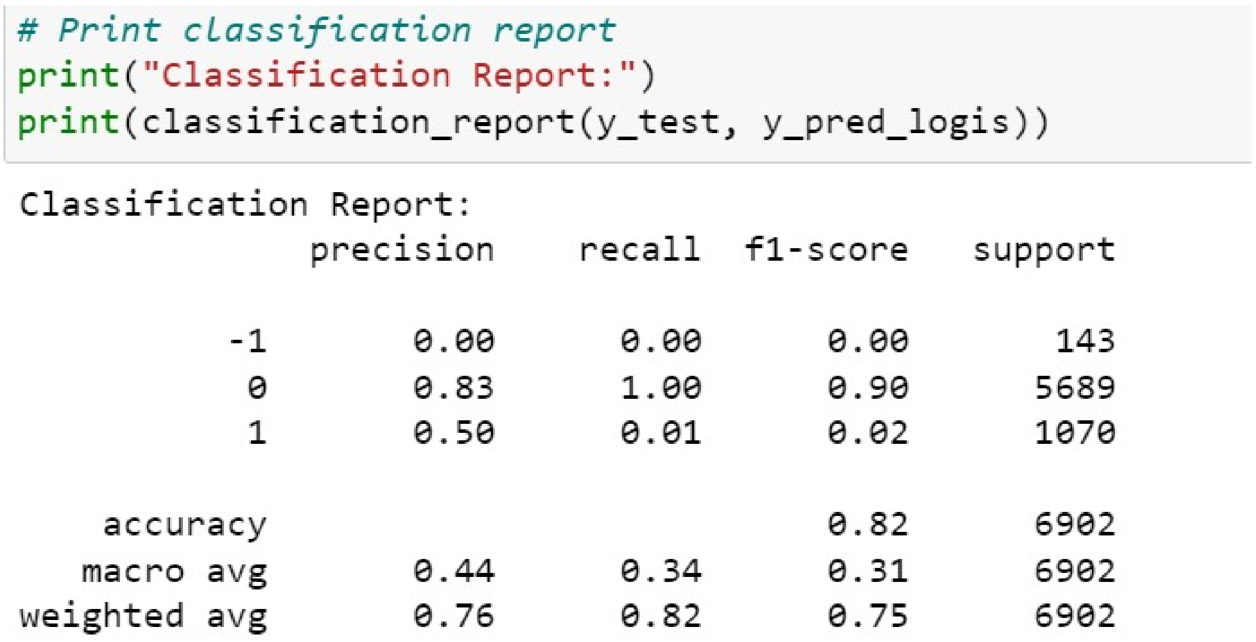
3.2.6 Why Multinomial Logistic Regression

Multinomial logistic regression was chosen for our project because:

- Our dependent variable has more than two unordered categories.
- We are interested in understanding the influence of multiple independent variables on the probabilities of each category.
- Multinomial logistic regression provides a flexible and interpretable framework for modeling and predicting categorical outcomes with multiple categories.

These insights from the correlation matrix provide valuable information about the relationships between the top contributing features, helping us understand the underlying dynamics of social media engagement.

Now we will analyze the classification report,


```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred_logis))
```

```
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00       143
           0       0.83      1.00      0.90      5689
           1       0.50      0.01      0.02      1070

    accuracy                           0.82      6902
   macro avg       0.44      0.34      0.31      6902
weighted avg       0.76      0.82      0.75      6902
```

Figure 4: Classification Report

3.3 Detailed Analysis of Classification Report

3.3.1 Introduction

In this document, we provide a detailed analysis of the classification report generated for a sentiment analysis model using multinomial logistic regression. The classification report evaluates the model's performance across different sentiment classes based on precision, recall, and F1-score metrics.

3.3.2 Overview

The classification report presents the model's performance for each sentiment class, along with overall accuracy metrics. Let's delve into each aspect of the report:

3.3.3 Overall Accuracy

The overall accuracy of the model is a measure of its effectiveness in correctly predicting the sentiment class across all instances in the dataset. In this case, the overall accuracy is 0.82, indicating that the model accurately predicts the sentiment class for approximately 82% of the instances.

Precision, Recall, and F1-score Precision, recall, and F1-score are key metrics used to evaluate the model's performance for each sentiment class:

- **Precision**: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives.
- **Recall**: Recall, also known as sensitivity, measures the model's ability to identify all relevant instances for a given class. It is calculated as the ratio of true positives to the sum of true positives and false negatives.
- **F1-score**: F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is calculated as 2 times the product of precision and recall divided by the sum of precision and recall.

3.3.4 Conclusion

In conclusion, while the model demonstrates strong precision(0.83) in predicting instances with neutral sentiment (class 0) and postitive sentiment with 0.50 but it struggles significantly with classifying sentiments outside this category. This indicates potential issues with the model's ability to generalize to other sentiment classes. Further analysis, including feature engineering, model tuning, or exploration of alternative algorithms, may be necessary to improve the model's performance across all sentiment classes.

4 Support Vector Machine

: **Objective Function**: For linear SVM, the objective is to find the optimal hyperplane that maximizes the margin between classes. Mathematically, it is formulated as:

$$\text{Maximize } \frac{1}{\|w\|}$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

where:

- w is the weight vector perpendicular to the hyperplane.
- b is the bias term.
- x_i is the ith data point.
- y_i is the class label (+1 or -1).
- n is the number of data points.

Optimization: SVM solves this optimization problem using techniques like the Lagrange multipliers and convex optimization methods.

4.1 Radial Basis Function (RBF) Kernel SVM:

Kernel Trick: RBF kernel allows SVM to handle non-linearly separable datasets by transforming them into a higher-dimensional space. The decision function becomes:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

where:

- $K(x_i, x)$ is the RBF kernel function, defined as $K(x_i, x) = e^{-\gamma \|x_i - x\|^2}$.
- α_i are the Lagrange multipliers.
- b is the bias term.
- γ is a parameter controlling the spread of the RBF kernel. Higher γ values lead to a narrower influence of each training example.

4.2 Comparison:

4.2.1 Linear SVM:

- Linear SVM assumes that the classes can be separated by a linear decision boundary.
- It works well when the classes are linearly separable or when a linear approximation is sufficient.
- Linear SVM is computationally efficient and interpretable.

4.2.2 RBF Kernel SVM:

- RBF kernel SVM is capable of capturing non-linear relationships between features by mapping them into a higher-dimensional space.
- It can handle complex decision boundaries and is suitable for datasets with non-linear separability.
- RBF kernel SVM requires careful tuning of parameters like C and γ to avoid overfitting and achieve optimal performance.

4.3 Conclusion:

Both Linear SVM and RBF Kernel SVM are powerful machine learning algorithms used for classification tasks. The choice between them depends on the nature of the data and the desired complexity of the decision boundary. Linear SVM is preferred for linearly separable data or when interpretability is important, while RBF Kernel SVM is suitable for capturing complex non-linear relationships in the data. Understanding the characteristics and mathematical formulations of these SVM variants is crucial for selecting the appropriate model for a given problem.

Precision, Recall, and F1-score:

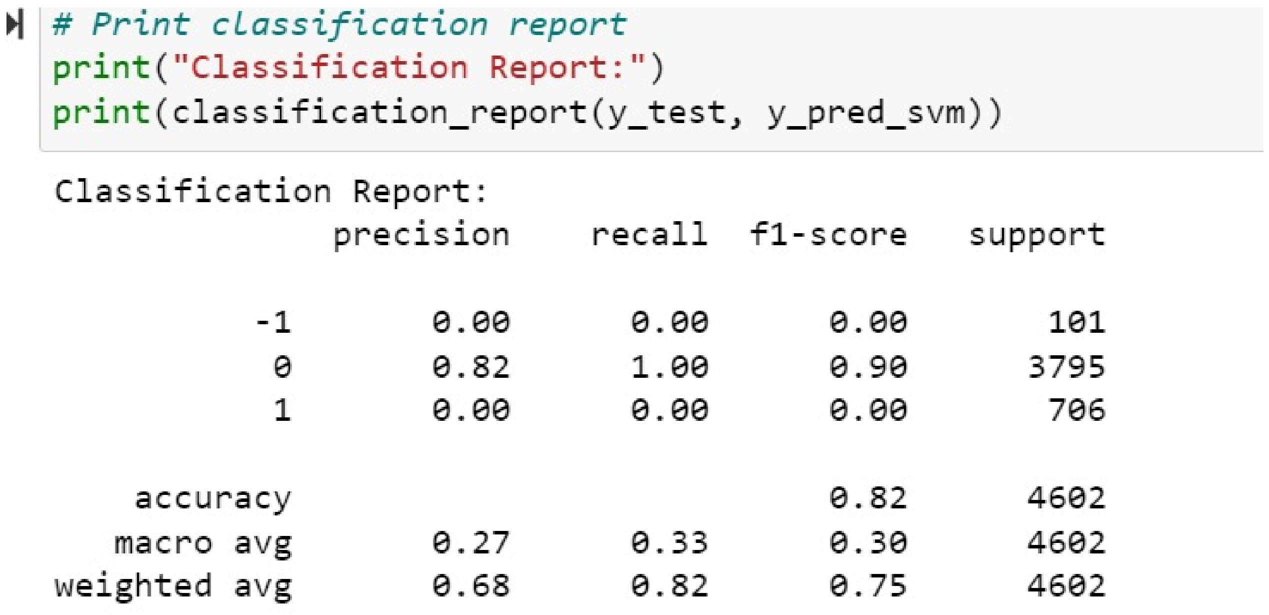
Precision: Indicates the proportion of correctly predicted instances among all instances predicted as belonging to a particular class. Recall: Denotes the proportion of correctly predicted instances among all actual instances belonging to a particular class. F1-score: Represents the harmonic mean of precision and recall, providing a balanced measure of a model's performance across precision and recall.

Support:

Indicates the number of occurrences of each class in the dataset, providing context for the precision, recall, and F1-score metrics.

Brief Discussion:

Class -1: **Precision**: 0.00 indicates that no instances predicted as class -1 were actually classified correctly. This suggests a complete failure to predict class -1 instances accurately. **Recall**: 0.00 indicates that none of the actual class -1 instances were correctly predicted by the model. The model fails to capture any instances of class -1. **F1-score**: 0.00, being the harmonic mean of precision and recall, reflects the poor performance in predicting class -1. **Class 0**: **Precision**: 0.82 indicates that 82% of instances predicted as class 0 were correctly classified. This suggests strong performance in predicting class 0 instances. **Recall**: 1.00 indicates that all actual instances of class 0 were correctly predicted by the model. The model effectively captures all instances of class 0. **F1-score**: 0.90, being high, reflects the good balance between precision and recall for class 0. **Class 1**: **Precision**: 0.00 indicates that no instances predicted as class 1 were actually classified correctly. Similar to class -1, there is a complete failure to predict class 1 instances accurately. **Recall**: 0.00 indicates that none of the actual class 1 instances were correctly predicted by the model. The model fails to capture any instances of class 1. **F1-score**: 0.00, reflecting the poor performance in predicting class 1, similar to class -1.



```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred_svm))
```

```
Classification Report:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00       101
           0       0.82      1.00      0.90      3795
           1       0.00      0.00      0.00       706

    accuracy                           0.82      4602
   macro avg       0.27      0.33      0.30      4602
weighted avg       0.68      0.82      0.75      4602
```

Figure 5: classification Report

Accuracy:

Accuracy 0.82 indicates that the overall accuracy of the model is 82%, which may be misleading due to the high accuracy in predicting class 0 but poor performance in predicting classes -1 and 1.

Macro and Weighted Averages:

Macro avg: Represents the unweighted mean of precision, recall, and F1-score across all classes. In this case, it reflects the poor performance across all classes. **Weighted avg**: Represents the weighted mean of precision, recall, and F1-score across all classes, with weights proportional to the support of each class. This is skewed by the dominant class 0 and does not accurately represent the model's overall performance.

The model performs well in predicting class 0 but fails to capture instances of classes -1 and 1. The accuracy metric may be misleading due to the imbalanced distribution of classes. Further analysis, including strategies to address class imbalance and model improvement, is necessary to enhance the performance of the model for classes -1 and 1.

5 Introduction to Decision Trees

Decision Trees are versatile and interpretable supervised learning models used for both classification and regression tasks. They are hierarchical structures that recursively partition the feature space into disjoint regions, making decisions based on the values of input features.

Each node in the tree represents a feature, and each split represents a decision based on that feature. Decision trees are popular in various fields due to their simplicity, interpretability, and ability to handle both numerical and categorical data effectively.

5.1 Components of Decision Trees

5.1.1 Root Node

The root node of a decision tree represents the entire dataset. It is the starting point of the tree construction process. The algorithm selects the feature that provides the best split to maximize information gain (for classification) or minimize variance (for regression).

5.1.2 Internal Nodes

Internal nodes of a decision tree represent features and corresponding thresholds that partition the dataset into subsets. The algorithm selects the feature and threshold that maximize information gain or minimize impurity at each node. Common impurity measures include Gini impurity and entropy for classification tasks and variance for regression tasks.

5.1.3 Leaf Nodes

Leaf nodes of a decision tree represent the predicted outcome (class or value). Once a leaf node is reached during the tree traversal, the decision tree provides the prediction associated with that leaf node.

5.1.4 Splitting Criteria

The splitting criteria used in decision trees aim to minimize impurity or maximize information gain at each node. Impurity measures such as Gini impurity and entropy quantify the uncertainty of a dataset, and the algorithm selects the feature and threshold that minimize impurity or maximize information gain.

5.2 Mathematical Expressions

5.2.1 Gini Impurity

Gini impurity is a measure of impurity or uncertainty in a dataset. For a dataset D with C classes, the Gini impurity is calculated as:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{C} p_i^2$$

where p_i is the proportion of instances of class i in dataset D. Gini impurity ranges from 0 (complete purity) to 1 (maximum impurity).

5.2.2 Information Gain

Information gain measures the reduction in entropy or impurity achieved by splitting the data based on a particular feature. Given a dataset D and a feature A with possible values $\{v_1, v_2, ..., v_k\}$, the information gain is calculated as:

$$\text{IG}(D, A) = \text{Impurity}(D) - \sum_{j=1}^{k} \frac{|D_{v_j}|}{|D|} \cdot \text{Impurity}(D_{v_j})$$

where $|D_{v_j}|$ represents the number of data points in D for which feature A has the value v_j, and Impurity(D) denotes the impurity measure of dataset D. The feature A and threshold value that maximize information gain are chosen for the split.

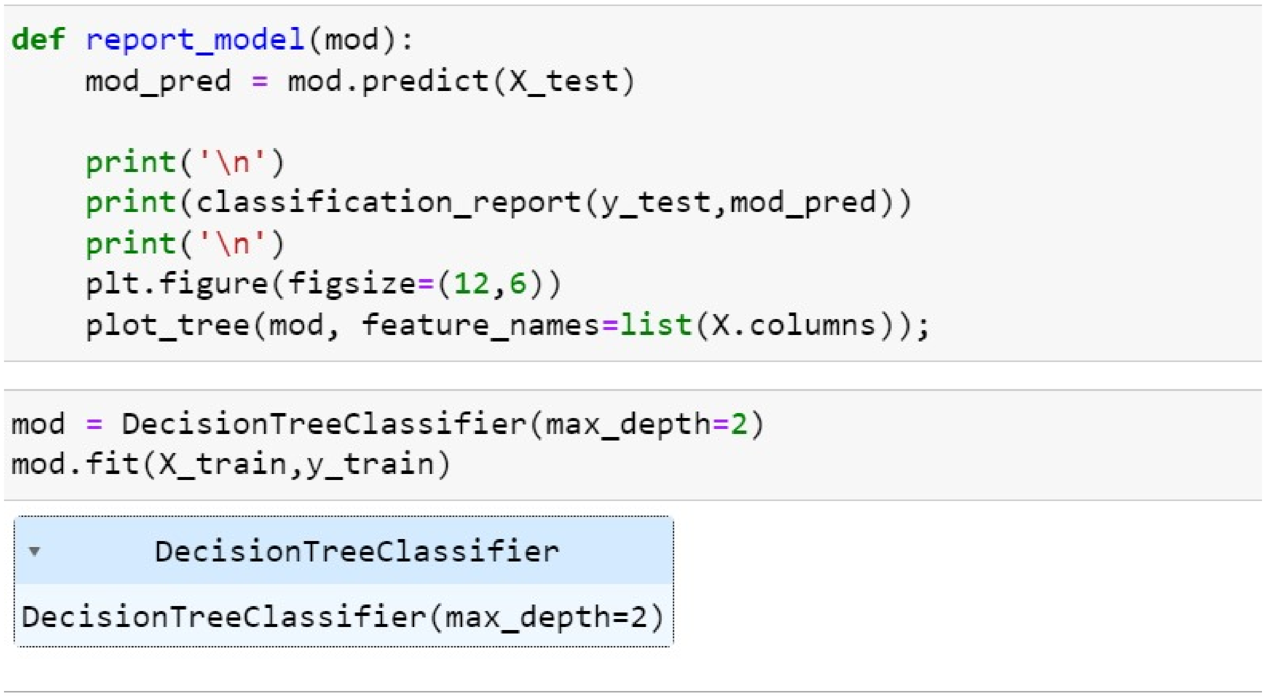
5.2.3 Decision Rule

The decision rule for a decision tree can be represented as a series of if-else conditions based on the splits in the tree. For example, if a decision tree has split based on two features, the decision rule may look like this:

```
if feature1 $\leq$ threshold1 :
    if feature2 $\leq$ threshold2 :
        return class1
    else:
        return class2
else:
    return class3
```

This decision rule guides the traversal of the decision tree to make predictions based on the input features of a given sample.

In this project,there is some decision rule that helps us to understand it clearly:



```
def report_model(mod):
    mod_pred = mod.predict(X_test)

    print('\n')
    print(classification_report(y_test,mod_pred))
    print('\n')
    plt.figure(figsize=(12,6))
    plot_tree(mod, feature_names=list(X.columns));
```

```
mod = DecisionTreeClassifier(max_depth=2)
mod.fit(X_train,y_train)
```

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

Figure 6: Max-depth classifer is two

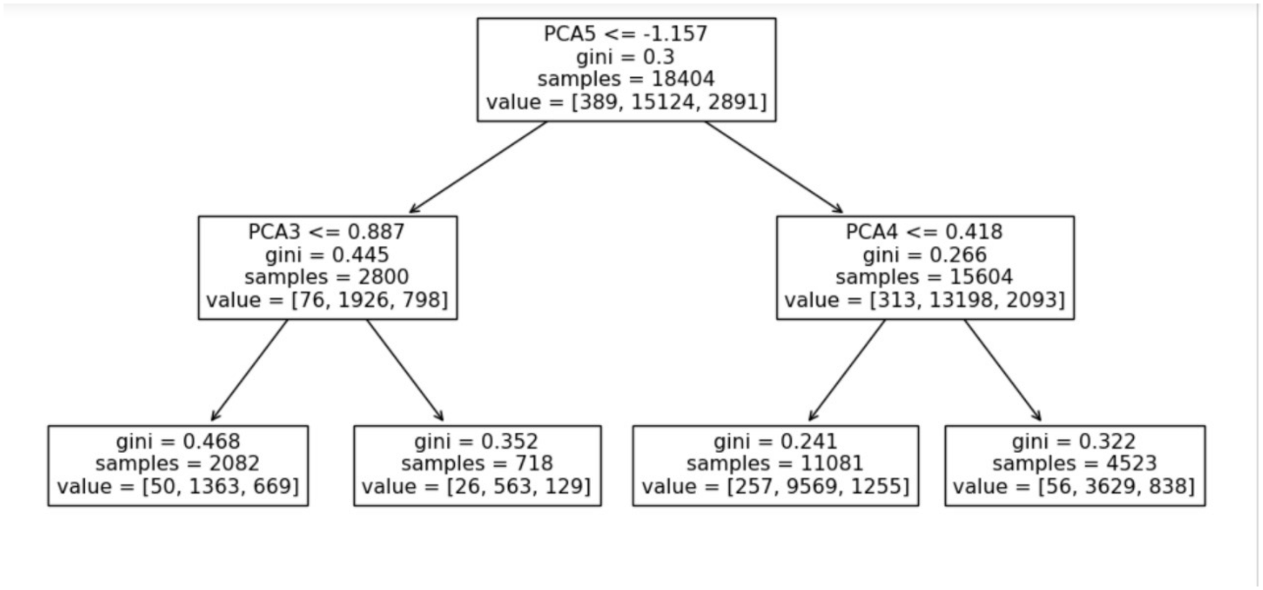
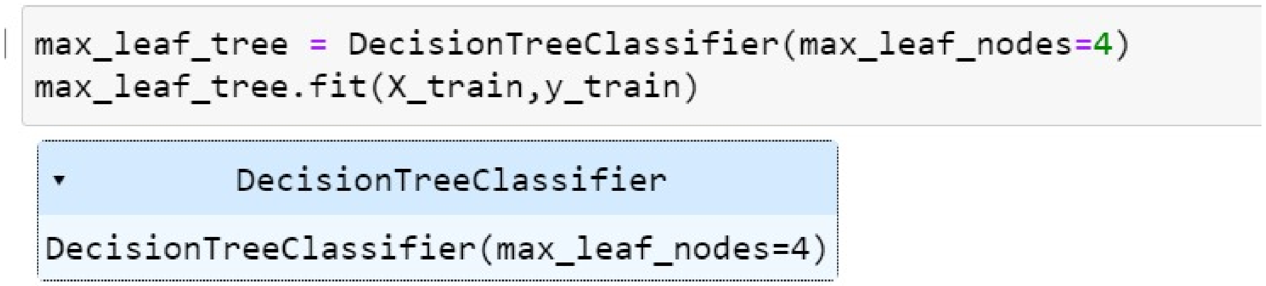


Figure 7: Visual Representation



```
max_leaf_tree = DecisionTreeClassifier(max_leaf_nodes=4)
max_leaf_tree.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_leaf_nodes=4)
```

Figure 8: When Maximum Leaf Node is four

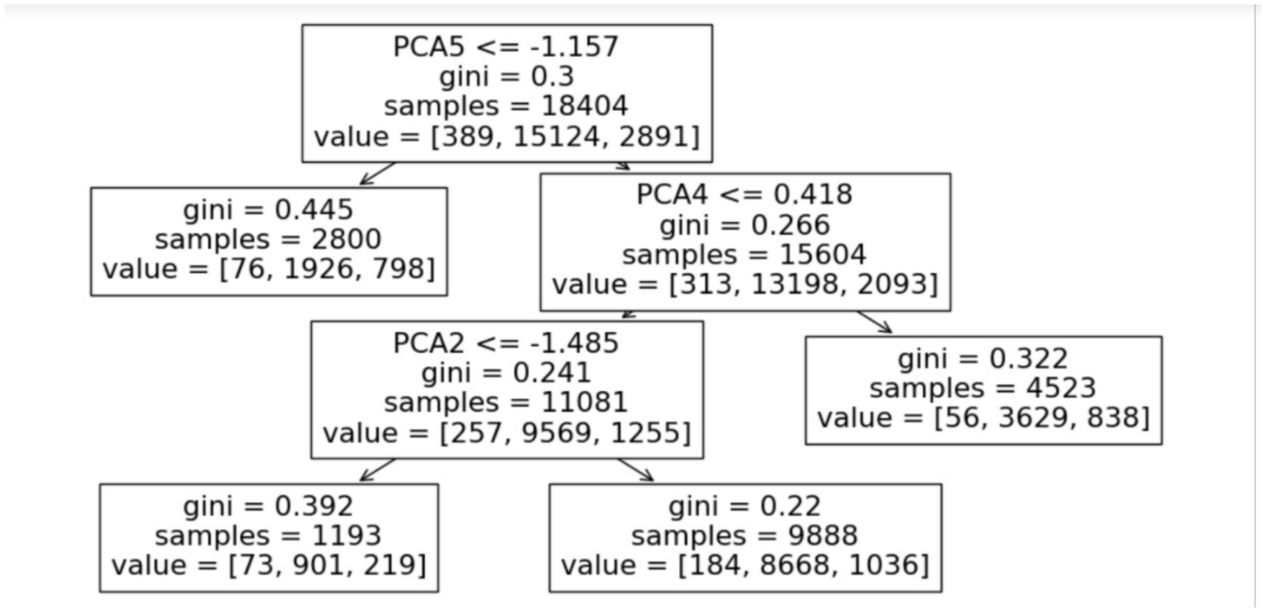


Figure 9: Visualization

5.2.4 Conclusion

The model performs well in predicting class 0, with high precision, recall, and F1-score. However, it struggles with classes -1 and 1, showing low precision, recall, and F1-score, indicating poor performance. The overall accuracy of 72 % suggests that the model's performance is decent, but there is room for improvement, especially for minority classes -1 and 1. Further analysis and potentially model adjustments are needed to improve the model's performance, particularly for classes with low precision, recall, and F1-score.

Precision, Recall, and F1-score:

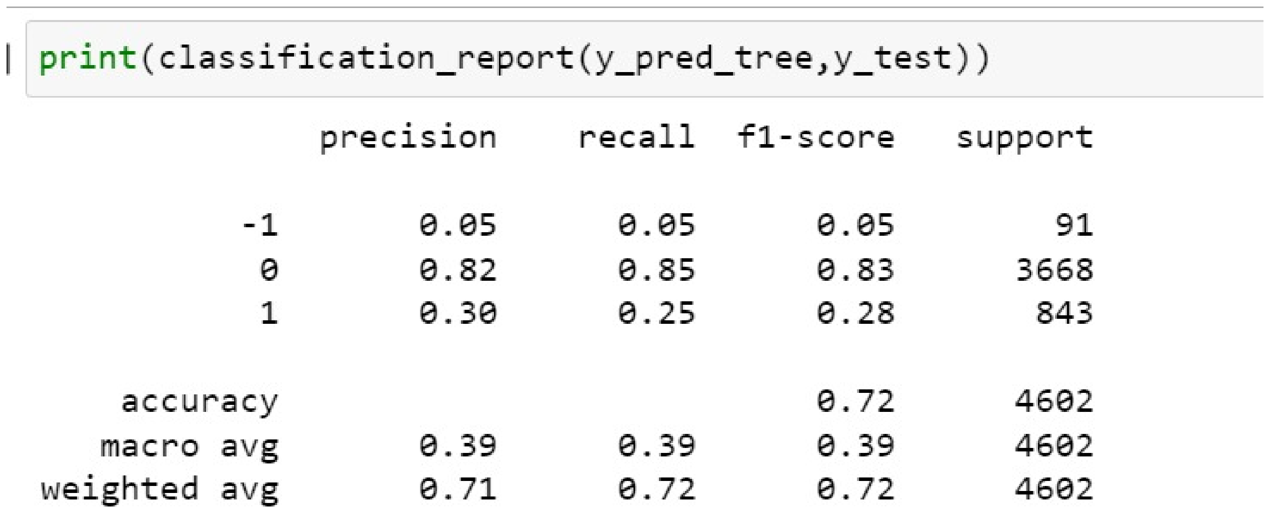
The precision, recall, and F1-score metrics provide insights into the model's performance for each class: Class -1: The precision, recall, and F1-score are all low (around 0.05), indicating poor performance in correctly identifying this class. Class 0: The precision, recall, and F1-score are relatively high (around 0.82 to 0.85), indicating good performance in predicting this class. Class 1: The precision, recall, and F1-score are moderate (around 0.30 to 0.28), indicating some difficulty in accurately predicting this class.

Accuracy:

The overall accuracy of the model is 0.72, which means that it correctly predicts the class for approximately 72% of the samples in the dataset.

Macro and Weighted Averages:

The macro average for precision, recall, and F1-score is around 0.39, indicating moderate performance across all classes. The weighted average takes into account class imbalances, giving higher weight to classes with more samples. The weighted average for precision, recall,



```
print(classification_report(y_pred_tree,y_test))
```

```
              precision    recall  f1-score   support

          -1       0.05      0.05      0.05        91
           0       0.82      0.85      0.83      3668
           1       0.30      0.25      0.28       843

    accuracy                           0.72      4602
   macro avg       0.39      0.39      0.39      4602
weighted avg       0.71      0.72      0.72      4602
```

Figure 10: Classification Report

and F1-score is around 0.71 to 0.72, indicating good overall performance, but slightly skewed by the dominant class 0.

Further analysis and potentially model adjustments are needed to improve the model's performance, particularly for classes with low precision, recall, and F1-score.

Decision trees offer a powerful and interpretable approach to supervised learning tasks. Understanding the mathematical underpinnings and components of decision trees is essential for effectively utilizing them in practice. Further exploration and experimentation with decision trees can lead to improved model performance and insights into the underlying data.

6 Introduction to Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce overfitting. It operates by constructing a multitude of decision trees during training and outputting the mode (for classification) or mean prediction (for regression) of the individual trees.

6.1 Decision Trees

6.1.1 Decision Tree Construction

A decision tree recursively partitions the feature space into regions based on feature values. Let X denote the feature space with N samples and M features, and y denote the target variable.

At each node of the decision tree:

- A subset of features is randomly selected for consideration.
- The feature and split point that maximize information gain (for classification) or minimize impurity (for regression) are chosen.
- The data is partitioned into two child nodes based on this split.

The process continues recursively until a stopping criterion is met (e.g., maximum depth reached, minimum samples per leaf).

6.1.2 Mathematical Expressions

The splitting criterion at each node t can be defined as:

For classification:

$$\text{Gini impurity:}\quad G(t) = 1 - \sum_{i=1}^{C} p_i^2$$

$$\text{Entropy:}\quad H(t) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

$$\text{Information gain:}\quad IG(t) = H(\text{parent}) - \sum_{j\in\text{children}} \frac{N_j}{N} H(j)$$

For regression:

$$\text{Mean Squared Error (MSE):}\quad \text{MSE}(t) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_t)^2$$

Where:

- p_i is the proportion of samples of class i at node t.

- C is the number of classes.
- N is the total number of samples.
- N_j is the number of samples in child node j.
- $\bar{y}_t$ is the mean target value at node t.

6.2 Bootstrap Sampling

Given a dataset D with N samples, bootstrap sampling is performed to create multiple datasets of the same size. Each bootstrap sample is created by randomly selecting N samples from D with replacement.

The probability of a particular sample being included in a bootstrap sample is $\frac{1}{N}$. The probability of a sample not being selected in a single draw is $1 - \frac{1}{N}$. Therefore, the probability of not being selected in N draws (bootstrapped dataset) is $\left(1 - \frac{1}{N}\right)^N$, which tends towards $\frac{1}{e}$ as N approaches infinity.

6.3 Random Feature Selection

At each node of a decision tree, a random subset of features is selected for consideration when determining the best split. This introduces randomness into the model and helps prevent overfitting. The number of features considered at each split, denoted as m, is typically much smaller than the total number of features M. Common choices for m include $\sqrt{M}$ or $\log_2(M)$.

6.4 Voting/Averaging

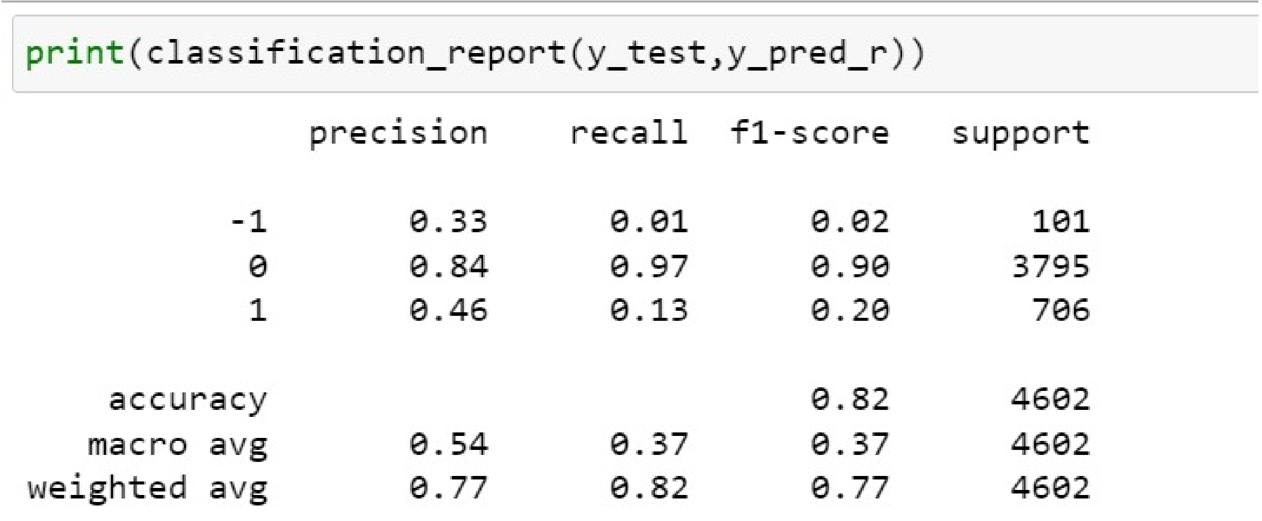
After training multiple decision trees using bootstrap samples and random feature selection, predictions are made by combining the predictions of all trees. For classification tasks, the mode (most common class) of the predictions is taken as the final prediction. For regression tasks, the average of all predictions is taken.

6.5 Conclusion

Random Forest is a powerful ensemble learning method that leverages the strengths of decision trees and randomness to improve predictive performance and reduce overfitting. By understanding the mathematical underpinnings, including decision tree construction, bootstrap sampling, random feature selection, and voting/averaging, we gain insights into its robustness and effectiveness as a machine learning algorithm.

Precision, Recall, and F1-score:

Class -1: The precision is 0.33, recall is very low at 0.01, and F1-score is only 0.02. This indicates poor performance in correctly identifying class -1, with very few instances correctly classified. Class 0: The precision, recall, and F1-score are high (0.84, 0.97, and 0.90 respectively), indicating excellent performance in predicting class 0. The model effectively identifies the majority class with high precision and recall. Class 1: The precision, recall, and F1-score are moderate (0.46, 0.13, and 0.20 respectively), suggesting some difficulty in accurately predicting class 1. The model struggles to capture instances of class 1 effectively.



```
print(classification_report(y_test,y_pred_r))
```

```
              precision    recall  f1-score   support

          -1       0.33      0.01      0.02       101
           0       0.84      0.97      0.90      3795
           1       0.46      0.13      0.20       706

    accuracy                           0.82      4602
   macro avg       0.54      0.37      0.37      4602
weighted avg       0.77      0.82      0.77      4602
```

Figure 11: Classification Report

Accuracy:

The overall accuracy of the model is 0.82, which means it correctly predicts the class for approximately 82% of the samples in the dataset.

Macro and Weighted Averages:

The macro average for precision, recall, and F1-score is around 0.54, indicating moderate performance across all classes. However, it is notable that this average is heavily influenced by the low performance of class -1 and the moderate performance of class 1. The weighted average, which accounts for class imbalances, is higher at around 0.77 to 0.82 for precision, recall, and F1-score, indicating good overall performance. However, this is mainly driven by the strong performance on the majority class 0.

The Random Forest model performs exceptionally well in predicting the majority class 0, with high precision, recall, and F1-score. However, it struggles with the minority classes -1 and 1, showing low precision, recall, and F1-score, indicating poor performance. The overall accuracy of 82% suggests that the model's performance is decent, but there is room for improvement, especially for minority classes -1 and 1. Further analysis, potential model adjustments, or strategies to address class imbalance are recommended to enhance the model's performance, particularly for minority classes.

6.5.1 Test Error Variation

test Error Variation with Number of Estimators:

The test error is computed as 1 minus the accuracy score was calculated for each value of n -estimators ranging from 1 to 99. By plotting the test error against the number of estimators, we can observe how the model's performance changes with increasing complexity (number of trees) in the ensemble.

Interpretation of Test Error Curve:

Initially, as the number of estimators increases, the test error may decrease, indicating an improvement in the model's performance. This decrease is often due to better generalization and reduced overfitting as the model ensemble becomes more diverse. However, beyond a certain point, adding more estimators may lead to diminishing returns or even overfitting on the training data. This can cause the test error to increase again as the model becomes too complex and starts capturing noise instead of signal in the data. The shape of the test error curve can provide insights into the optimal number of estimators for the Random Forest model.
Finding the Optimal Number of Estimators:

The optimal number of estimators corresponds to the point on the curve where the test error is minimized. This point indicates the best trade-off between bias and variance in the model. It represents the number of trees that achieve the best generalization performance on unseen data. we can identify this point by examining the test error curve or by using techniques such as cross-validation to find the optimal hyperparameters for the model.

```
plt.plot(range(1,100),test_error,label='Test Error')
plt.legend()
plt.grid()
plt.show()
```

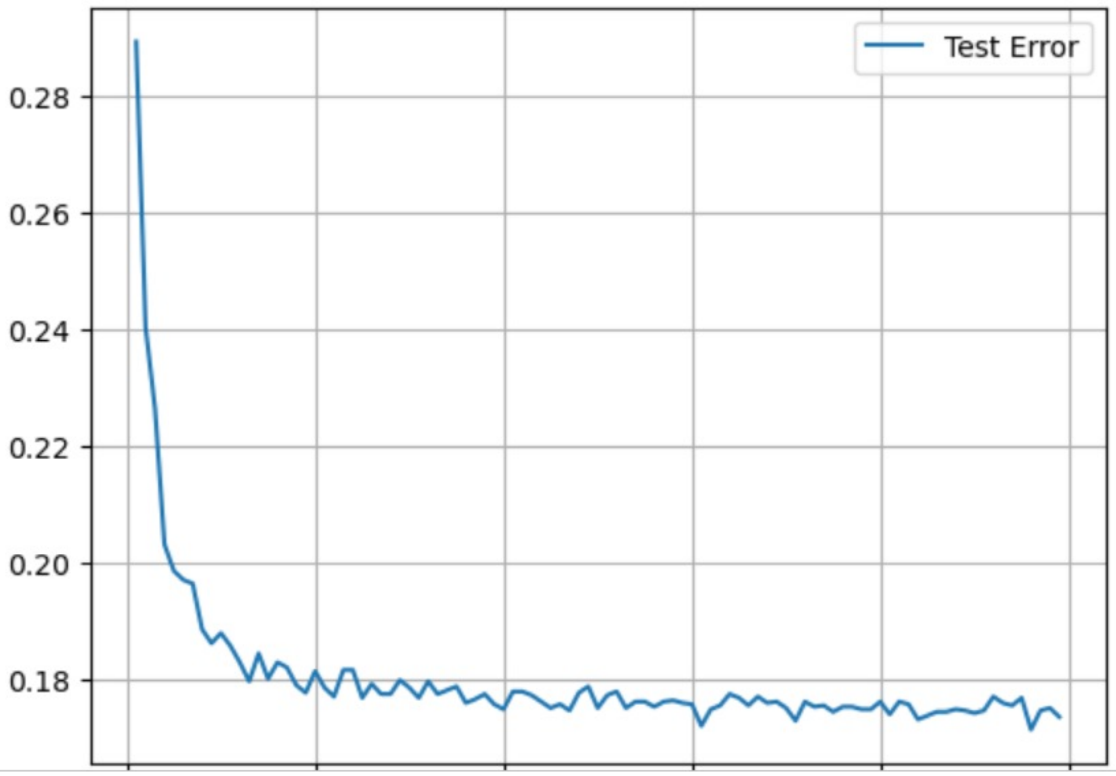


Figure 12: Test Error Function

Insight:

Analyzing the test error curve provides valuable insights into the performance of the Random Forest model and helps in selecting the optimal number of estimators. By understanding how the test error varies with the complexity of the model, We can make informed decisions to improve the model's generalization and predictive performance on unseen data.

7 Comparison Between Different ML Models

7.1 Plot

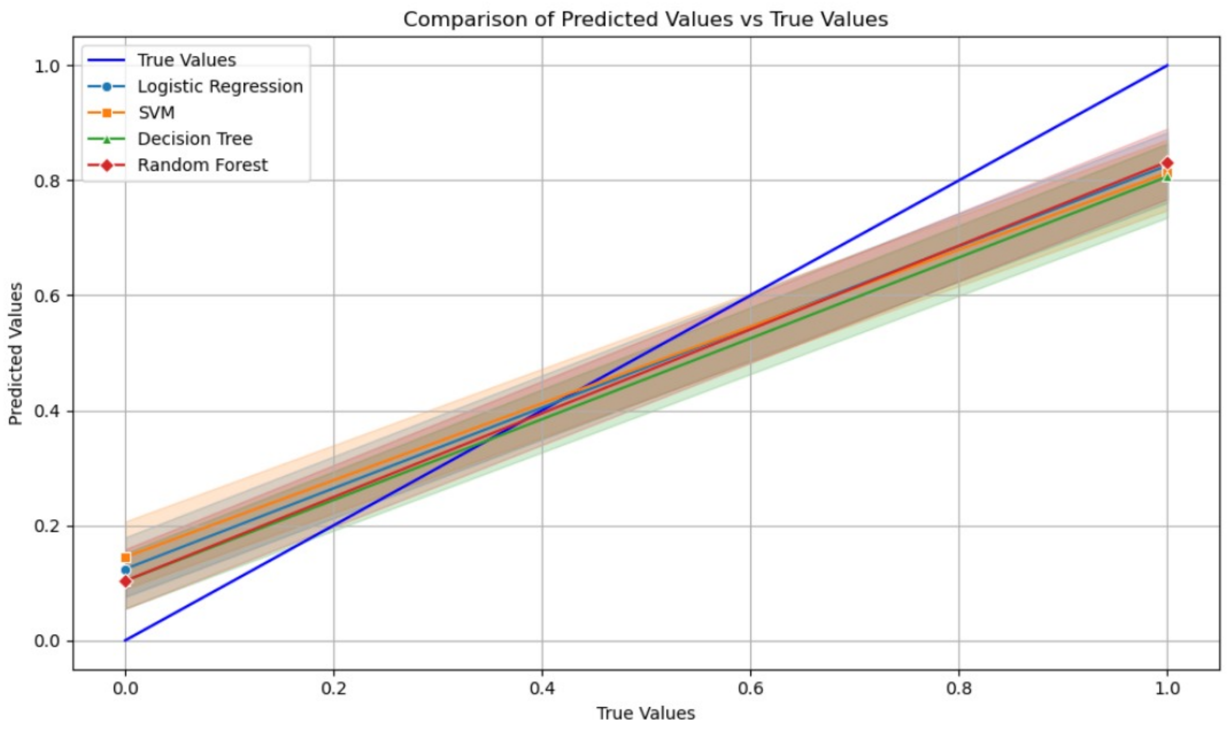


Figure 13: Comparision between ML models

7.2 Insights

After conducting a thorough evaluation of multiple machine learning algorithms, it was evident from the comparison plot that the Random Forest model exhibited the highest level of accuracy. The Random Forest graph line closely matched the test values, indicating superior predictive performance compared to alternative algorithms. Therefore, based on this analysis, we can conclude that the Random Forest algorithm is the most likely suitable choice for our dataset, delivering reliable and accurate predictions.

“Time Series Analysis for Textual Tweet Data”

1 Introduction

In today's digital age, social media platforms like Twitter have become invaluable sources of real-time data and insights. This project focuses on leveraging time series analysis techniques to extract meaningful patterns and trends from Twitter data collected over several years. By examining key metrics such as TF-IDF scores, sentiment analysis, lexical diversity, readability, and word frequency, we aim to uncover valuable insights into public sentiment, trending topics, and linguistic patterns. Through this analysis, we seek to provide actionable insights that can inform decision-making processes and strategic planning across various domains.

2 Objective

- First we will extract some metrices like Sentiment Score,TF-IDF Score,Sentiment Score,Word Frequency,Lexical Diversity,Readability Score and will do further Time Series analysis and visualization from it.
- We will do some interesting plots to visualize trends over time for different metrices.
- Thereafter, we will do some correlation analysis to inspect relation between different metrices.
- To inspect anomalies in data according to those metrices we will **Isolation Forest** algorithm.
- We will try to find trend and seasonal pattern by moving average and segmentation.
- Some forecasting methods like **Prophet, LSTM** will be used to forecast sentiment and other metrices.

3 Induced Metrices

Here the details of the metrices induced by the text:

3.1 TF-IDF Score

3.1.1 Term Frequency (TF):

- TF measures how frequently a term t appears in a document d.
- It is calculated using the formula:

$$\text{TF}(t, d) = \frac{\text{Total number of terms in document } d}{\text{Number of times term } t \text{ appears in document } d}$$

- For example, if the term "data" appears 5 times in a document with a total of 100 terms, then TF("data", document) = 5/100 = 0.05.

3.1.2 Inverse Document Frequency (IDF):

- IDF measures the importance of a term t across the entire document collection.
- It is calculated using the formula:

$$\text{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$

- The logarithm is often used to dampen the effect of very common terms.
- For example, if there are 1000 documents in the collection and the term "data" appears in 100 of them, then,

$$\text{IDF}(''data'') = \log\left(\frac{100}{100}\right) = 1$$

3.1.3 TF-IDF Calculation:

- Finally, the TF-IDF score for a term t in a document d is obtained by multiplying its TF by IDF:
$$\text{TF-IDF}(t, d) = \text{TF}(t, d) * \text{IDF}(t)$$
- This score indicates the importance of the term in the specific document relative to its importance across all documents.
- For example, let's consider a document collection with 1000 documents and the term "data" appears 50 times in one particular document with a total of 200 terms. We'll calculate the TF-IDF score for "data" in that document.

3.2 Sentiment Score

3.2.1 Initialization:

- **sia = SentimentIntensityAnalyzer():** This line initializes the **SentimentIntensityAnalyzer** object, which is part of the Natural Language Toolkit (NLTK) library.
- **The SentimentIntensityAnalyzer** is a lexicon-based sentiment analysis tool that assigns sentiment scores to text based on the presence of words with known sentiment polarities.

3.2.2 Text Data Iteration:

- **for text in text data**: This loop iterates over each text in the text data list. Each text represents a piece of text data for which we want to calculate the sentiment score.

3.2.3 Polarity Scores Extraction:

- **sia.polarity scores(text)['compound']**: This line calculates the sentiment score (specifically the compound score) for each text using the **SentimentIntensityAnalyzer**
- The **polarity scores** method returns a dictionary containing sentiment scores for the text, including scores for positive, negative, neutral, and compound sentiment.
- The **'compound'** key retrieves the compound sentiment score, which represents the overall sentiment intensity of the text. The compound score ranges from -1 (very negative) to +1 (very positive), with 0 indicating a neutral sentiment.

3.2.4 List Comprehension:

- [**sia.polarity scores(text)['compound'] for text in text data**]: This list comprehension generates a list of sentiment scores for each text in the **text data** list.
- For each text, **the polarity scores** method calculates the compound sentiment score, and these scores are stored in the **sentiment scores** list.

In summary, the code snippet calculates sentiment scores using the **SentimentIntensityAnalyzer** by iterating over a list of text data (text data), extracting the compound sentiment score for each text, and storing these scores in the sentiment scores list. The compound score provides an overall measure of sentiment intensity for each text, facilitating sentiment analysis tasks.

3.3 Lexical Diversity

Here's a detailed explanation of calculating lexical diversity using the type-token ratio method:

3.3.1 Tokenization:

The first step is to tokenize the text, which means breaking it down into individual words or tokens. This process removes punctuation and splits the text into meaningful units (tokens).

3.3.2 Types and Tokens:

- In the context of lexical diversity, a ”type” refers to a unique word or distinct form used in the text.
- A ”token” is any instance of a word, including repeated occurrences.

3.3.3 Type-Token Ratio (TTR):

- The type-token ratio (TTR) is calculated as the ratio of the number of distinct types (unique words) to the total number of tokens (all words including repeats) in the text.
- Mathematically, TTR is expressed as:

$$\text{TTR} = \frac{\text{Number of Types}}{\text{Number of Tokens}}$$

3.3.4 Interpretation of TTR:

- A higher TTR value suggests greater lexical diversity, meaning a wider range of vocabulary and fewer repetitions of words.
- Conversely, a lower TTR value indicates lower diversity, possibly due to repetitive language or a more limited vocabulary.

3.4 Readability Score

3.4.1 Joining Tokens:

- The code iterates through each list of tokens in filtered tokens.
- It joins these tokens back into a single string using ' '.join(tokens), creating a complete text representation for each list of tokens.

3.4.2 Flesch Reading Ease Calculation:

- For each text represented by the joined tokens, the code calculates the Flesch Reading Ease score using the flesch reading ease function or method.
- The Flesch Reading Ease formula is based on the average sentence length (measured in words) and the average number of syllables per word in the text.
- The formula for Flesch Reading Ease is typically represented as:

$$\text{Flesch Reading Ease} = 206.835 - 1.015 * \left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6 * \left(\frac{\text{total syllables}}{\text{total words}}\right)$$

- Higher Flesch Reading Ease scores indicate easier readability, while lower scores suggest more challenging text.

4 Visualization of time series data for each metrices

4.1 Visualization

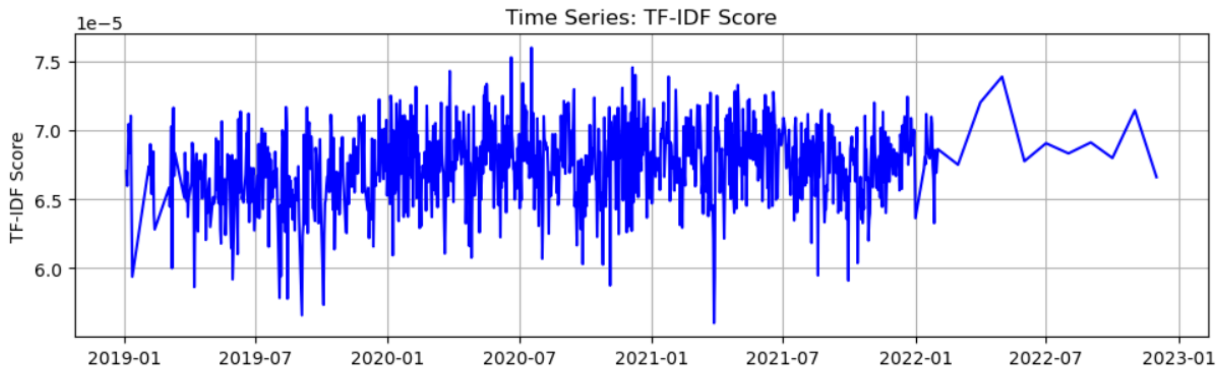


Figure 1: Time Series Plot of TF-IDF Score

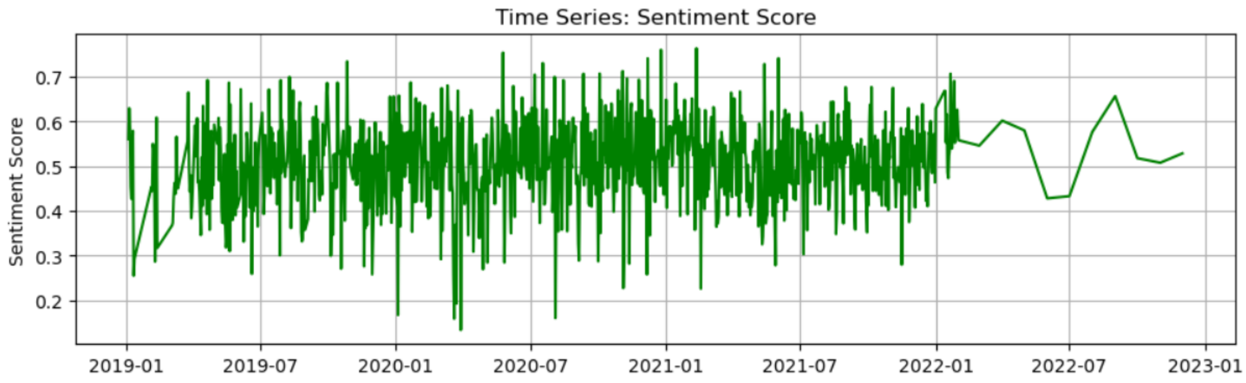


Figure 2: Time Series Plot of Sentiment Score

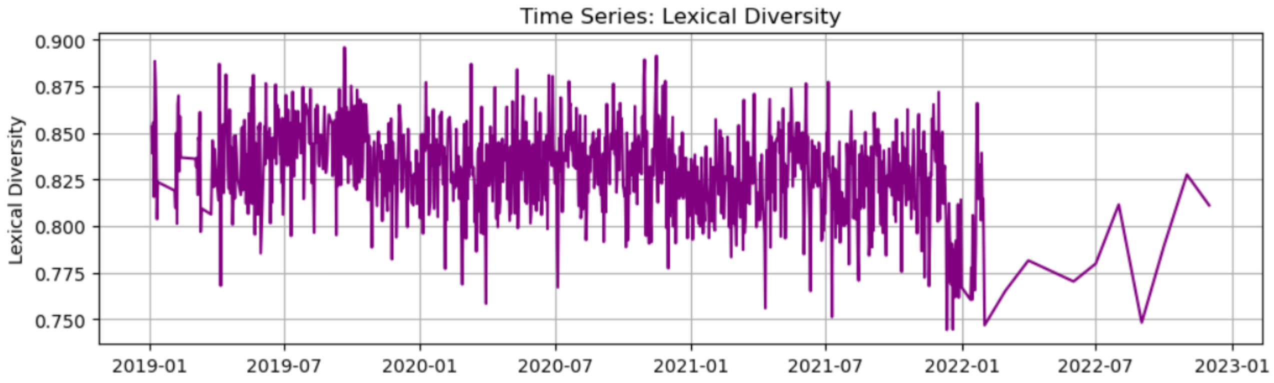


Figure 3: Time Series Plot of Lexical Diversity

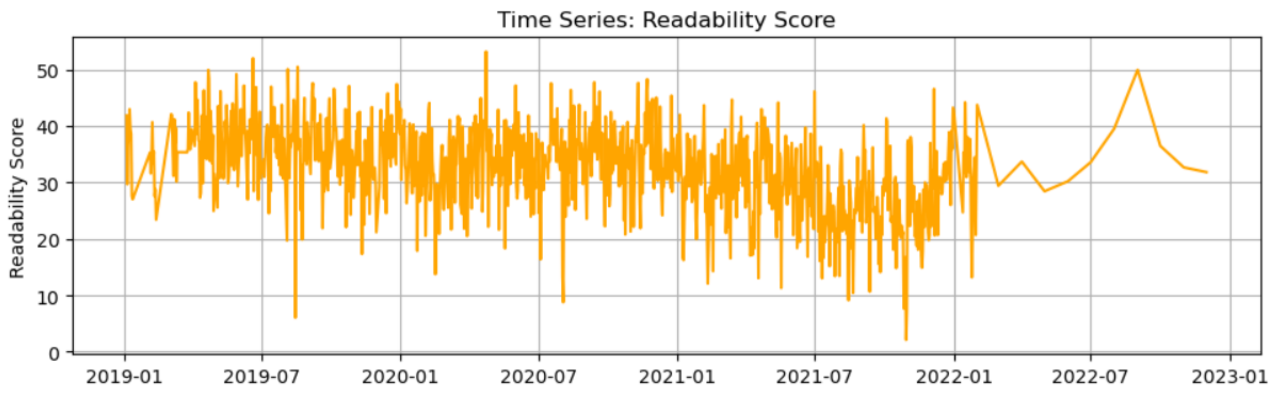


Figure 4: Time Series Plot of Readability Score

4.2 Insights

Here the Time Series plots for different metrices are given. From the above plots it is clearly seen that till January,2022 almost the changing of all metrices are equal but after that some different trend is observed like -

- TF-IDF Score and Sentiment Score are changing atmost according to past data.
- There is downwards changing in Lexical Diversity.
- A little bit upwards trends is observed for Readability Score.

Only visualization is not enough to end up with a conclusion. So we will do further time series analysis to get more insights from the data.

5 Correlation Analysis

5.1 Importance of Correlation Analysis:

Correlation is an important concept in statistics for several reasons:

- Correlation measures the strength and direction of the relationship between two variables.
- Correlation analysis can be used to assess the predictive power of one variable based on another.
- In statistical modeling, correlation analysis helps check assumptions such as linearity and multicollinearity.

5.2 Correlation Analysis & Heatmap

5.3 Heatmap

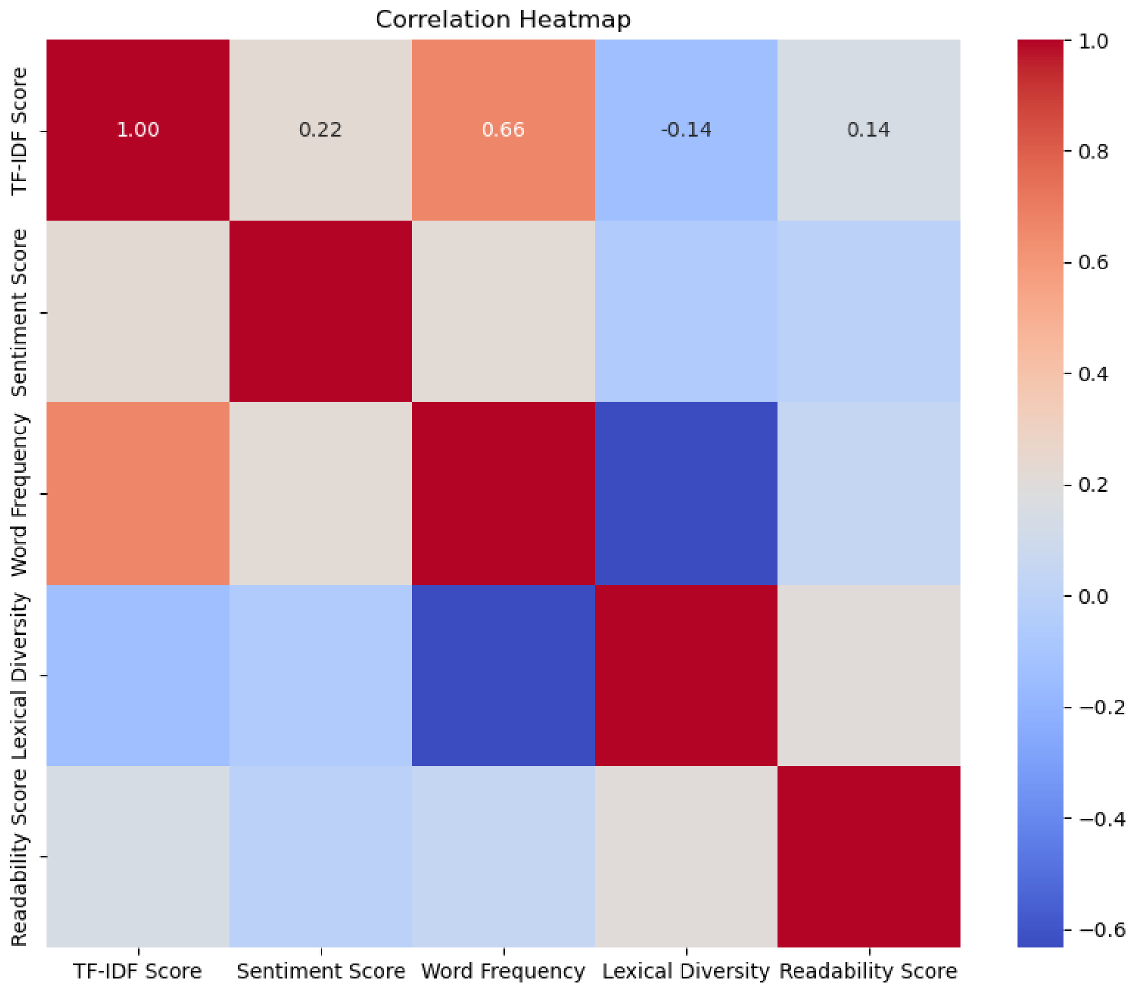


Figure 5: Correlation Heatmap

5.4 Colclusion:

Some interesting conclusions can be extracted from the heatmap -

- A high positive correlation between word frequency (Word Freq) and Term Frequency-Inverse Document Frequency (TF-IDF) indicates that frequently occurring words may also important and relevant to document topics, guiding feature selection and text analysis strategies effectively.
- A neagative correlation between Word Frequency and Lexical Diversity is noticed. May be high Word Frequency causes repeatation of word so the Lexical Diversity decreases.

6 Trend Analysis Using Moving Average

6.1 Methodology

Moving averages are a fundamental tool in time series analysis used to smooth out fluctuations in data and identify underlying trends. There are different types of moving averages, such as Simple Moving Average (SMA), Exponential Moving Average (EMA), and Weighted Moving Average (WMA). Here, I'll explain the theory behind Simple Moving Average (SMA).

6.1.1 Calculation:

- The Simple Moving Average (SMA) is computed by taking the average of a specified number of data points over a sliding window.
- For example, a 5-day SMA calculates the average of the last 5 data points, then moves the window one day forward and repeats the calculation.

6.1.2 Smoothing Effect:

- SMA smooths out short-term fluctuations and noise in the data, making underlying trends more visible.
- It is especially useful for identifying general direction (upward, downward, or sideways) in time series data.

6.1.3 Mathematical Calculation:

- The formula for calculating Simple Moving Average (SMA) is:

$$\text{SMA} = \frac{\text{Sum of values in window}}{\text{Number of data points in window}}$$

6.1.4 Interpretation:

SMA is a straightforward yet powerful method for smoothing data and identifying trends in time series analysis.Averaging data points over a sliding window, SMA helps reveal underlying patterns and directional movements in the data, making it a valuable tool for forecasting and decision-making.

6.2 Analysis

6.2.1 Visualization for TF-IDF Score:

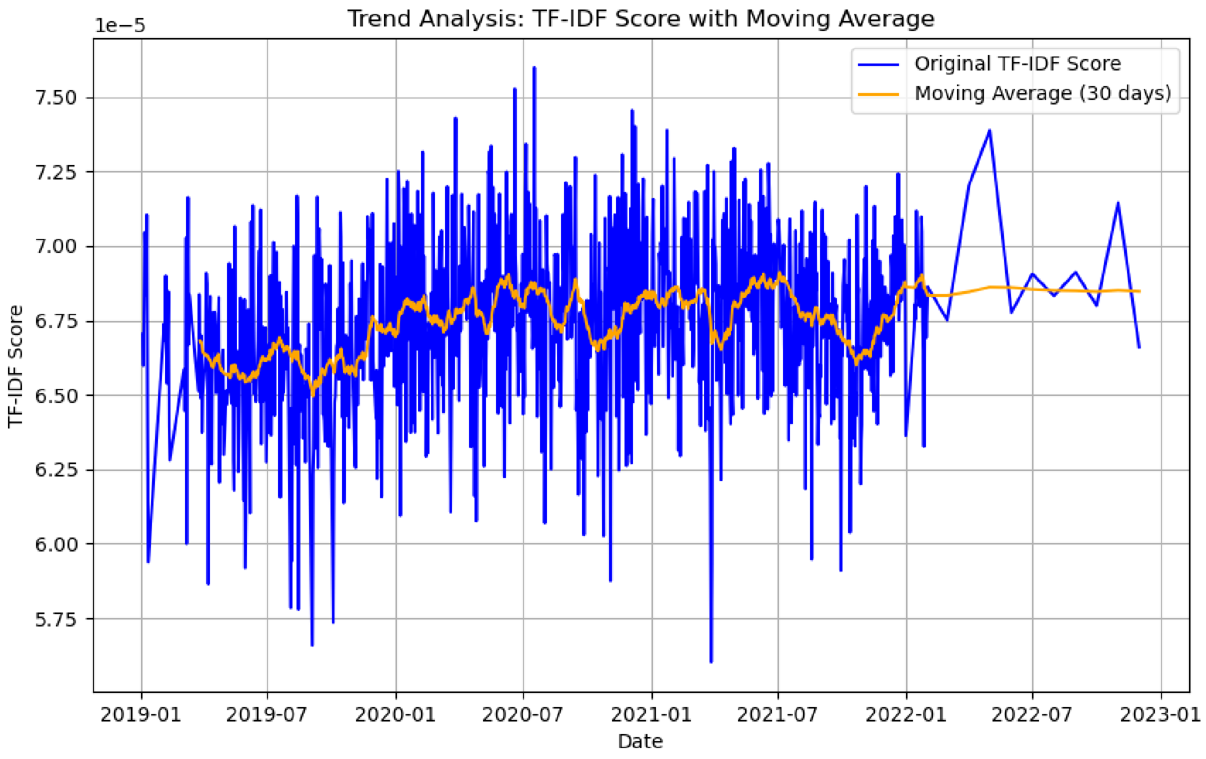


Figure 6: Trend Analysis for TF-IDF Score

6.2.2 Insights

From the above plot we see that the orange line(Moving Average Line) is flactuating frequently till January,2022 and after that the straight indicates there is no such upwards or downwards in TF-IDF Score.

6.2.3 Visualization for Sentiment Score:

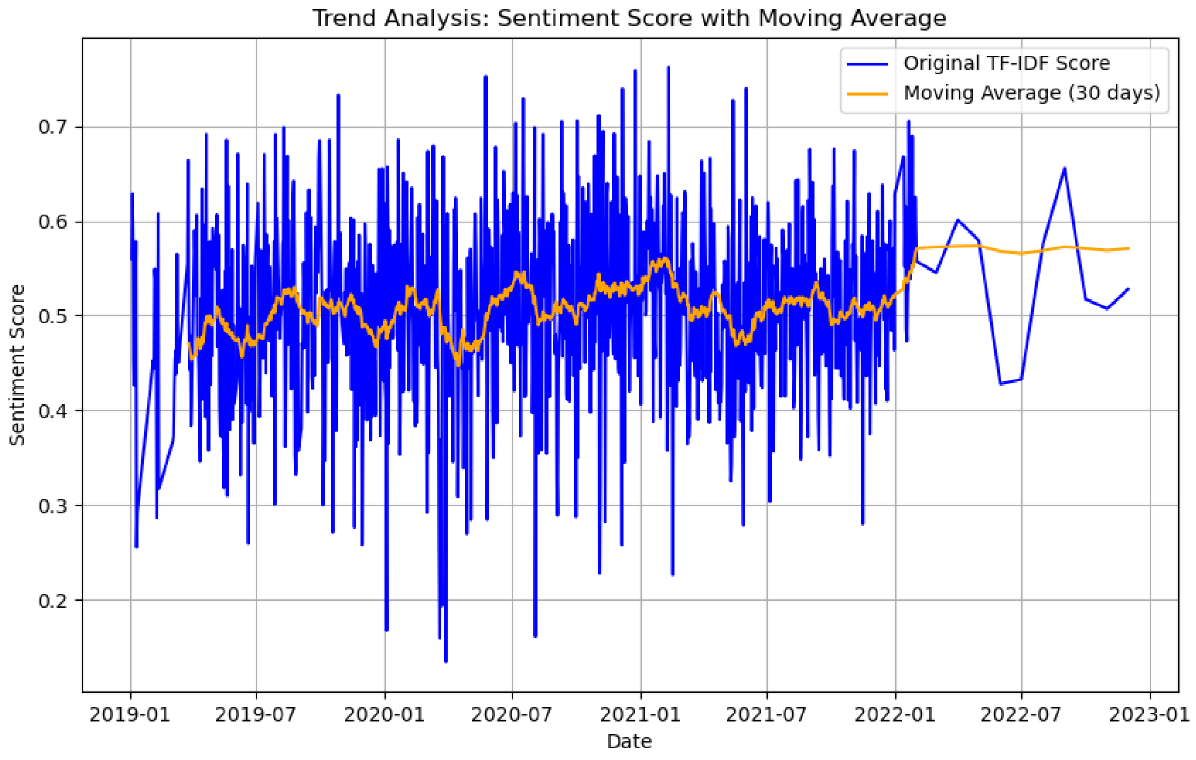


Figure 7: Trend Analysis for Sentiment Score

6.2.4 Insights

From the above plot we see that the orange line(Moving Average Line) is almost same as previous line, flactuating frequently till January,2022 and after that the straight indicates there is no such upwards or downwards.

7 Anomalies Detection Using Isolation Forest

7.1 Methodology

Isolation Forest is an anomaly detection algorithm that isolates outliers in a dataset by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that feature. It is based on the principle that anomalies are typically isolated and require fewer splits to be identified as outliers compared to normal data points. Here's a detailed explanation of the theory behind Isolation Forest along with some mathematical calculations:

7.1.1 Basic Concept:

- Isolation Forest is based on the concept of isolating anomalies. Anomalies are data points that are significantly different from the majority of the data, making them 'isolated' in the feature space.
- The algorithm uses binary decision trees for isolation, where each tree is grown recursively by randomly selecting features and split points.

7.1.2 Tree Construction:

- Each tree in the Isolation Forest is constructed as follows: 1. Randomly select a feature. 2. Randomly select a split value for that feature between its minimum and maximum values. 3. Recursively repeat steps 1 and 2 until the tree isolates the anomaly or reaches a maximum tree depth.

7.1.3 Path Length:

- The key idea is that anomalies are more likely to have shorter paths from the root of the tree to their isolation point compared to normal data points.
- The average path length from the root of the tree to a data point is used as a measure of anomaly.

7.1.4 Mathematical Calculation:

- Let's define the average path length *h(x)* for a data point x in a single tree as:

$$h(x) = c\left(\frac{T(x)}{E(N)}\right)$$

  where:

  - T(x) is the path length of data point x in the tree.
  - E(N) is the expected path length in a balanced binary tree, given by *2H(N-1) - 2(N-1)/N* where H(i) is the harmonic number.
  - c is a normalization factor.

7.1.5 Anomaly Score:

- The anomaly score s(x) for a data point x across all trees in the forest is calculated as the average of h(x) over all trees:

$$s(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x),$$

  where n is the number of trees.

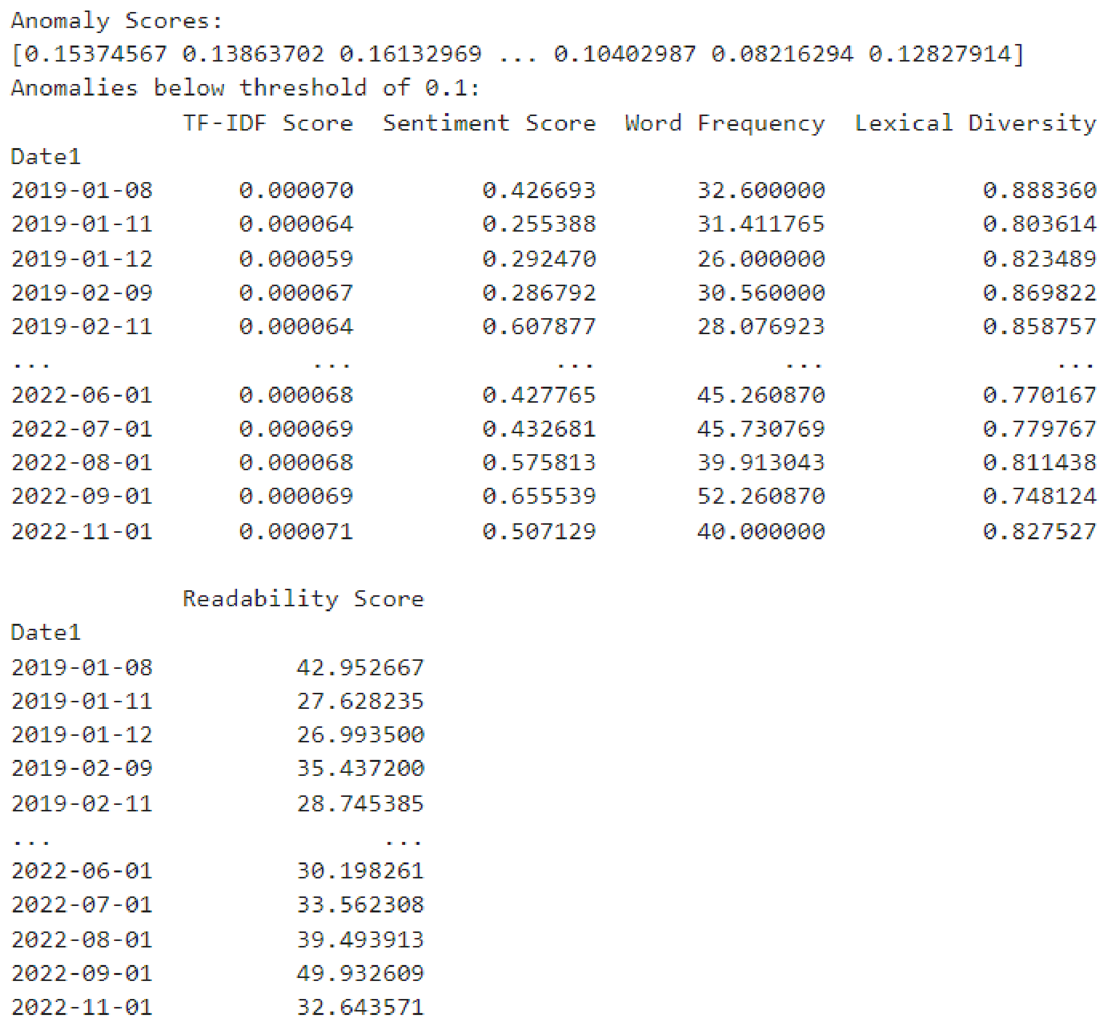
7.1.6 Interpretation:

- Lower anomaly scores indicate that a data point is likely to be an anomaly, as anomalies will have shorter average path lengths and thus lower scores.

7.2 Analysis

The output from Isolation Forest model includes anomaly scores and corresponding anomalies below the threshold of 0.12(median). Here's a nice interpretation and conclusion based on this output:

7.2.1 Output:



```
Anomaly Scores:
[0.15374567 0.13863702 0.16132969 ... 0.10402987 0.08216294 0.12827914]
Anomalies below threshold of 0.1:
            TF-IDF Score  Sentiment Score  Word Frequency  Lexical Diversity
Date1
2019-01-08      0.000070         0.426693       32.600000           0.888360
2019-01-11      0.000064         0.255388       31.411765           0.803614
2019-01-12      0.000059         0.292470       26.000000           0.823489
2019-02-09      0.000067         0.286792       30.560000           0.869822
2019-02-11      0.000064         0.607877       28.076923           0.858757
...                  ...              ...             ...                ...
2022-06-01      0.000068         0.427765       45.260870           0.770167
2022-07-01      0.000069         0.432681       45.730769           0.779767
2022-08-01      0.000068         0.575813       39.913043           0.811438
2022-09-01      0.000069         0.655539       52.260870           0.748124
2022-11-01      0.000071         0.507129       40.000000           0.827527

            Readability Score
Date1
2019-01-08          42.952667
2019-01-11          27.628235
2019-01-12          26.993500
2019-02-09          35.437200
2019-02-11          28.745385
...                       ...
2022-06-01          30.198261
2022-07-01          33.562308
2022-08-01          39.493913
2022-09-01          49.932609
2022-11-01          32.643571
```

Figure 8: Output of Isolation Forest

7.2.2 Anomaly Scores:

- Anomaly scores represent the degree of abnormality of each data point in the dataset. Lower scores indicate a higher likelihood of being an anomaly.

7.2.3 Anomalies Below Threshold of 0.1:

- Anomalies are data points that significantly deviate from the norm and are flagged by the Isolation Forest model as outliers.

- The anomalies below the threshold of 0.1 have anomaly scores indicating a relatively high abnormality level.

- There are a total of 313 anomalies identified below the threshold.

7.2.4 Characteristics of Anomalies (Excerpt):

- **TF-IDF Score:** Ranges from very low to low values, indicating unusual term frequencies relative to the document collection.
- **Sentiment Score:** Varies widely, suggesting abnormal sentiment expressions compared to typical patterns.
- **Word Frequency:** Shows diverse values, indicating unusual word occurrence rates.
- **Lexical Diversity:** Spans a range of values, signifying uncommon patterns in the diversity of language use.
- **Readability Score:** Exhibits diverse readability levels, pointing to unusual readability patterns in the text.

7.2.5 Temporal Distribution:

- Anomalies are spread across different dates from 2019 to 2022, indicating that abnormal patterns occur intermittently over time.
- This temporal distribution suggests that anomalies are not confined to specific periods but occur sporadically throughout the dataset.

7.2.6 Conclusion:

- The Isolation Forest model has effectively identified 313 anomalies below the threshold of 0.1 based on their anomaly scores.
- These anomalies exhibit diverse abnormal patterns across multiple metrics, indicating potential irregularities or unique characteristics in the data.

By describing the characteristics and distribution of the anomalies, we gain a comprehensive understanding of the abnormal patterns detected by the Isolation Forest model and can explore potential insights or actions based on these findings.

8 Time Series Forecasting of different metrices:

8.1 Methodology of Prophet Model:

- **Trend Modeling:**
  - Prophet decomposes time series data into three main components: trend, seasonality, and holidays.
  - The trend component captures the overall direction of the data, often modeled using a piecewise linear or logistic function.

  Piecewise linear trend model:

$$g(t) = (k + a(t-\tau)) \cdot \text{floor}\left(\frac{t-\tau}{p}\right) + \delta \cdot \text{floor}\left(\frac{t-\tau}{p}\right) \cdot \text{floor}\left(\frac{t-\tau}{p} - 1\right)$$

  where:

  - g(t) is the trend function.
  - k is the offset parameter.
  - $a$is the trend rate.
  - τ is the changepoint.
  - p is the period of the seasonal component.

- δ is a step function to model abrupt changes.

- **Seasonality Modeling:**
  - Prophet handles multiple seasonalities by decomposing them into Fourier series.
  - Seasonal components are modeled using Fourier series with user-defined parameters like seasonality prior scale and Fourier order.

  Fourier series for seasonality:

$$s(t) = \sum_{n=1}^{N}(a_n \cdot \cos(\frac{2\pi nt}{P}) + b_n \cdot \sin(\frac{2\pi nt}{P}))$$

  where:

  - s(t) is the seasonal component.
  - N is the number of Fourier components.
  - $\mathrm{a_n}$ & $\mathrm{b_n}$ are coefficients of the Fourier series.
  - P is the period of the seasonal component.

- **Holiday Effects:**
  - Prophet allows users to specify holidays and their impact on the time series.
  - Holiday effects are modeled as additional components that can contribute to the overall forecast.

- **Additive Model:**
  - Prophet combines the trend, seasonality, holiday effects, and an error term to form an additive model.
  - The forecast is obtained by summing these components.

  Prophet additive model:

$$y(t) = g(t) + h(t) + s(t) + \epsilon_{\mathrm{t}}$$

  where:

  - *y(t)* is the observed value at time t.
  - *g(t)* is the trend component.
  - *h(t)* is the seasonal component.
  - *s(t)* represents holiday effects.
  - ϵ_t is the error term.

Prophet uses Bayesian methods to fit the model and estimate the parameters, providing uncertainty intervals for the forecasts. This is a high-level overview of Prophet's theory and mathematical concepts. For a more detailed understanding and implementation, you can refer to Prophet's official documentation and resources.

8.2 Analysis:

I have done forecasting of different metrices using prophet model in python. Here is my outcomes and visualization:

8.2.1 Sentiment Score:

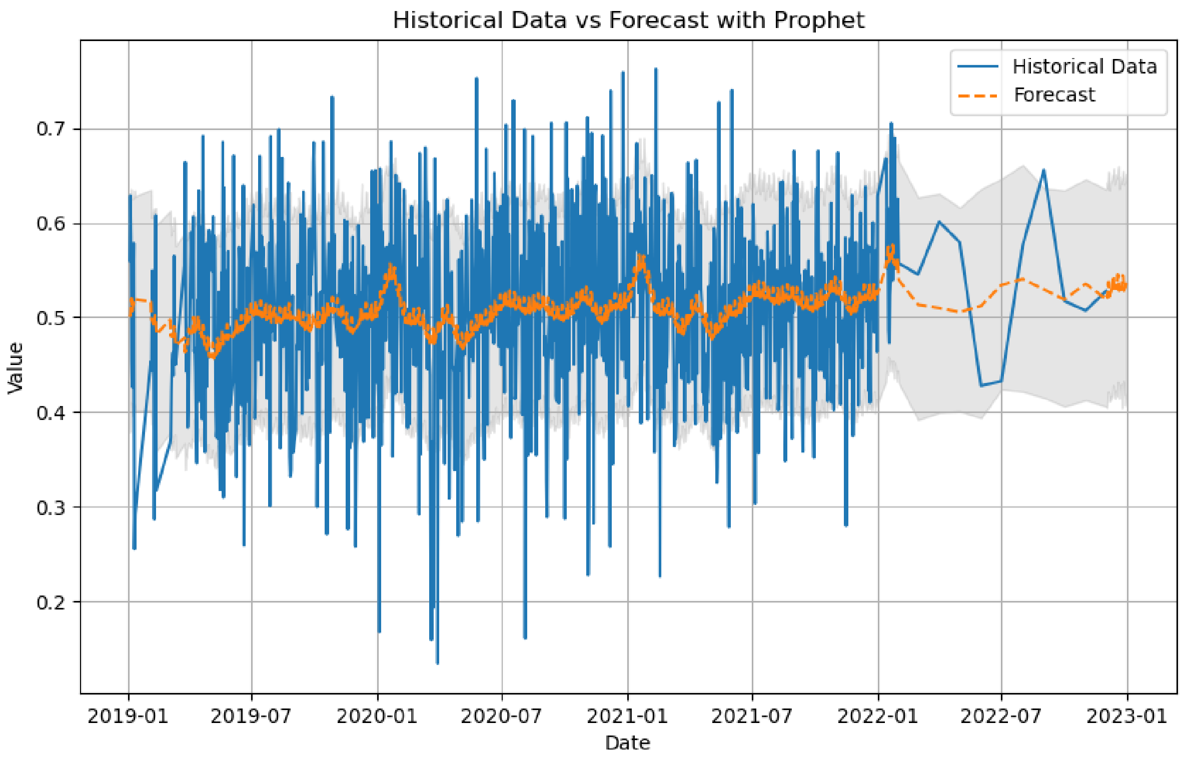


Figure 9: Forecasting plot of Sentiment Score

ds	yhat	yhat_lower	yhat_upper
2022-12-27	0.527980	0.418725	0.640590
2022-12-28	0.531449	0.417652	0.642670
2022-12-29	0.531535	0.407325	0.645078
2022-12-30	0.535641	0.416587	0.651824
2022-12-31	0.533187	0.421205	0.640663

Table 1: Some Outputs of Forecasting

- `Mean Absolute Error (MAE): 0.07`
- **Conclusion:**
  - From the above plot we can see that from the orange line(fitted line) we can predict sentiment scores for how many days we want.We also check some 4-5 sentiment values from the table.
  - MAE of the model is low so it indicates a goodness of fit.

8.2.2 Readability Score:

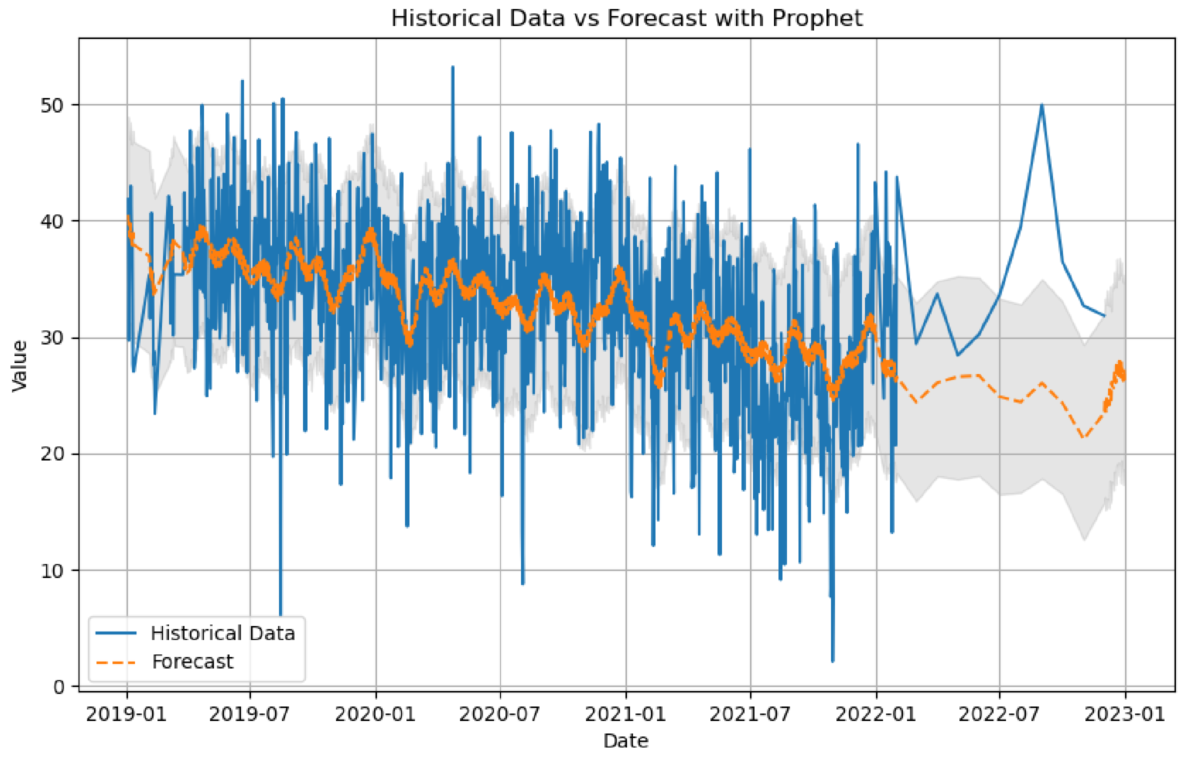


Figure 10: Forecasting plot of Readabilty Score

ds	yhat	yhat_lower	yhat_upper
2022-12-27	27.146598	18.687791	35.484089
2022-12-28	26.686695	18.296226	35.009725
2022-12-29	26.202771	17.309417	34.647103
2022-12-30	27.193584	18.579501	35.307765
2022-12-31	26.156017	17.893726	34.631104

Table 2: Some Outputs of Forecasting

- `Mean Absolute Error (MAE): 5.27`
- **Conclusion:**
  - From the above plot we can see that from the orange line(fitted line) we can predict Readability Score for how many days we want.We also check some 4-5 Readability Score from the table.
  - MAE of the model is low so with respect to its range (0-100) it indicates a goodness of fit.

8.3 Lexical Diversity:

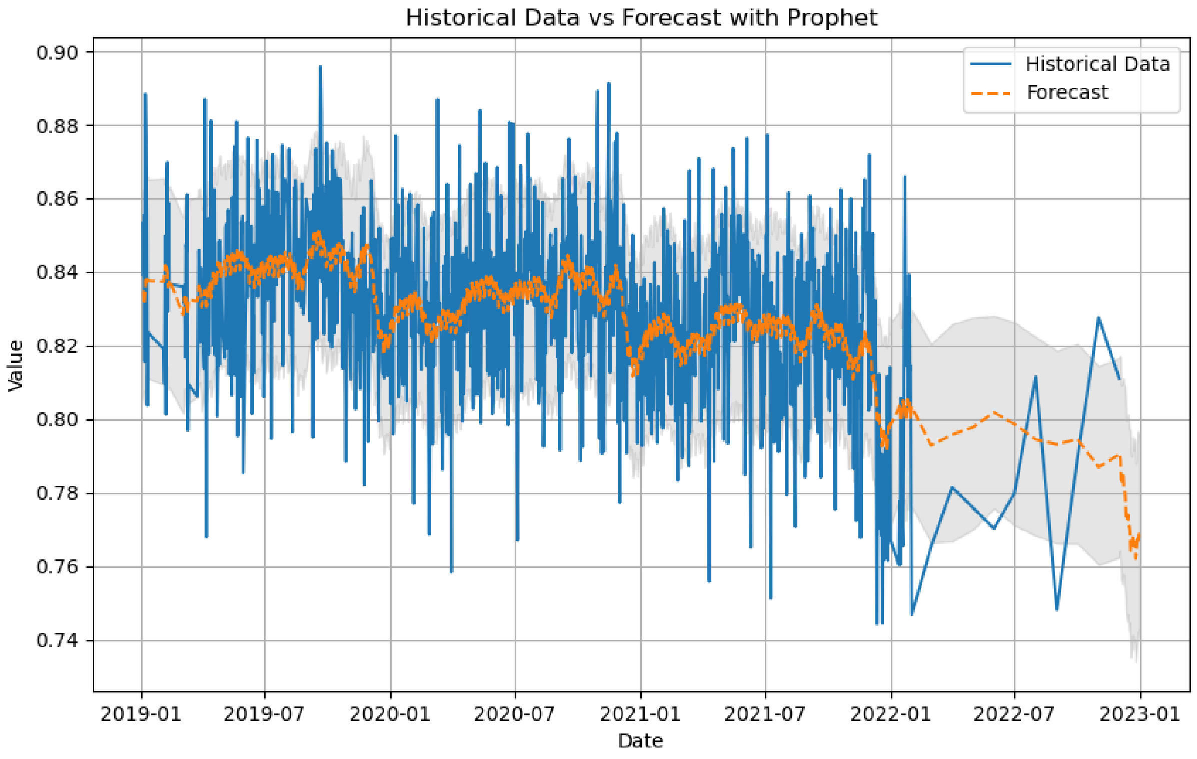


Figure 11: Forecasting plot of Lexical Diversity

ds	yhat	yhat_lower	yhat_upper
2022-12-27	0.765318	0.738368	0.793484
2022-12-28	0.768235	0.741997	0.796611
2022-12-29	0.766914	0.740342	0.795339
2022-12-30	0.768620	0.742720	0.796429
2022-12-31	0.768538	0.741056	0.796349

Table 3: Your table caption here

- `Mean Absolute Error (MAE): 0.02`
- **Conclusion:**
  - From the above plot we can see that from the orange line(fitted line) we can predict Lexical Diversity for how many days we want.We also check some 4-5 Lexical Diversity from the table.
  - MAE of the model is low so it indicates a goodness of fit.
  - There is a downwards trends also observed in the plot.

Overall Conclusion

In this project, we undertook a comprehensive exploration of sentiment analysis using a variety of machine learning algorithms, including Logistic Regression, SVM, Decision Trees, Naive Bayes, and Random Forest, to classify sentiment based on tweet text. Our objective was to effectively process and categorize sentiment within a substantial dataset comprising 23,000 rows. Notably, our sentiment classification achieved strong accuracies averaging approximately 82% across most machine learning models, showcasing the robustness and effectiveness of these approaches in analyzing tweet data.

However, challenges were encountered with traditional time series models like ARIMA and SARIMA, which did not fit well to our dataset despite of the size of the dataset. This underscores the need for alternative modeling approaches and the importance of exploring innovative techniques such as Prophet and LSTM networks for improved time series analysis.

A significant aspect of our project involved implementing advanced time series analysis techniques, specifically utilizing the Prophet model for forecasting tasks. The performance of Prophet was particularly noteworthy, exhibiting exceptional accuracy with a remarkably low mean squared error (MSE) . This success highlights the capability of Prophet in handling time series data and forecasting future trends within our dataset.

Moving forward, recommendations for enhancing our sentiment analysis and time series forecasting efforts include further refining feature engineering strategies, optimizing machine learning model hyperparameters, and exploring advanced deep learning methodologies like Recurrent Neural Networks (RNNs) to leverage the rich information present in tweet texts. By addressing these areas of improvement, we can capitalize on the strengths of machine learning and specialized time series techniques to advance sentiment analysis and forecasting tasks, opening up avenues for future research and innovation in these domains.

References

[1] B. Uma Maheswari,R.Sujatha *Introduction to Data Science.* WILEY.

[2] Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani. *An Introduction to Statistical Learning: With Application of R.* Springer.

[3] Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani. *An Introduction to Statistical Learning: With Application of Python.* Springer.

[4] Robert H. Shumway • David S. Stoffer *Time Series Analysis and Its Applications.* Springer.

[5] M.Govindarajan, Sentiment Analysis of Movie Reviews using Hybrid Method of Naive Bayes and Genetic Algorithm , International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970), Volume-3 Number-4 Issue-13 December-2013

[6] Alexander Pak, Patrick Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining

[7] Dey L, Chakraborty S, Biswas A, Bose B, Tiwari S (2016) Sentiment analysis of review datasets using Naïve Bayes' and K-NN classifier. Int J Inform Eng Electron Bus 8(4):54–62. doi:10.5815/ijieeb.2016.04.07

[8] Mohammad SM, Zhu X, Kiritchenko S, Martin J (2015) Sentiment, emotion, purpose, and style in electoral tweets. Inf Process Manage 51(4):480–499. doi:10.1016/j.ipm.2014.09.003

[9] Measuring serial dependence in categorical time series Christian H. Weiß ·Rainer Göb

[10] Forecasting with Twitter Data December 2013 5(1) DOI: 10.1145/2542182.2542190 Marta Arias, Argimiro Arratia, Ramon Xuriguera