**MINI PROJECT REPORT**

ON

**SENTIMENT ANALYSIS ON YOUTUBE COMMENTS**

Submitted in Partial Fulfilment of The Requirement for The

Award of the Degree Of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

Submitted by

**SUSHIL DHOUNDIYAL (2105110100142)**

**VEENAYAK SIROHI (2105110100151)**

Submitted to **MS. SAMRIDHI SINGH**

**Department of Computer Science & Engineering**



**G.L. BAJAJ INSTITUTE OF TECHNOLOGY & MANAGEMENT,**

**GREATER NOIDA**

Affiliated to

****

**Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW**

**(2023-24)**

## DECLARATION

We hereby declare that the mini project work being presented in this report entitled “**SENTIMENT ANALYSIS ON YOUTUBE COMMENTS**” submitted to the Department of Computer Science, STUDENT OF TECHNOLOGY**,** G.L. Bajaj Institute of Technology & Management, is the authentic work carried out by us under the guidance of **Ms. Samridhi Singh**, Project Head, Department of Computer Science, G.L. Bajaj Institute of Technology & Management, Greater Noida.

Sushil Dhoundiyal

(2105110100142)

Veenayak Sirohi

(2105110100151)

B.Tech (3rd year)

Department of Computer Science

## CERTIFICATE

This is to certify that the project entitled **“SENTIMENT ANALYSIS ON YOUTUBE COMMENTS"** represents the original work done by **Veenayak Sirohi (2105110100151) and Sushil Dhoundiyal(2105110100142)** during this project submission as partial fulfilment of the requirement for the Award of The Degree of Bachelor of Technology, V Semester, of the G.L. Bajaj Institute of Technology and Management, Greater Noida.

Date: Signature

(project head)

**ACKNOWLEDGEMENT**

We would like to express our special thanks of gratitude to our teacher Ms. Samridhi Singh who gave us the golden opportunity to do this wonderful project on the topic **SENTIMENT ANALYSIS ON YOUTUBE COMMENTS,** which helped us in doing a lot of Research we came to know about so many new things. She also helped us to clarify our doubts regarding the project and provided her valuable guidance to complete our project efficiently and on time.

We are thankful to her.

Sushil Dhoundiyal

(2105110100142)

Veenayak Sirohi

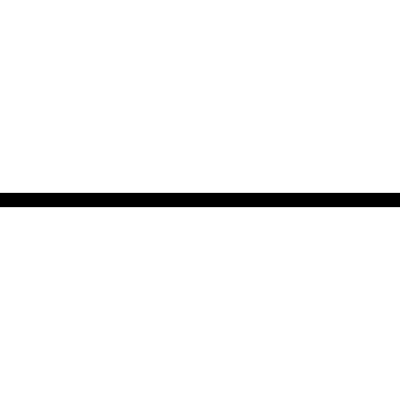
(2105110100151)

B.Tech (3rd year)

Department of Computer Science

**ABSTRACT**

This project centres on the sentiment analysis of YouTube comments, employing advanced natural language processing techniques. The core aim is to scrutinize the emotional tone conveyed in user comments associated with a particular YouTube video. The project encompasses a multifaceted approach, including data collection through the YouTube API, thorough text preprocessing to ensure data quality, sentiment analysis using NLTK's VADER tool, and machine learning modelling for sentiment classification. The resulting sentiment scores facilitate the categorization of comments into positive, negative, or neutral sentiments. The implementation of a Naive Bayes classifier enhances the project's depth, providing a robust model for sentiment classification. Visual representations, such as a pie chart, offer a clear depiction of sentiment distribution, shedding light on the overall audience response to the video content. In conclusion, this project not only contributes to the understanding of sentiment dynamics in YouTube comments but also establishes a comprehensive methodology for sentiment analysis in user-generated content platforms. The project serves as an introduction to building sentiment analysis for YouTube comments and provides a foundation for further development of the more advanced model.

**TABLE OF CONTENT**

**Declaration ...................................................................................... (ii)**

**Certificate ..................................................................................... (iii)**

**Acknowledgement .......................................................................... (iv)**

**Abstract .......................................................................................... (v)**

**Table of Content ............................................................................. (vi)**

**List of Figures …………………………………………………… (viii)**

**Chapter 1. Introduction .............................................................. 1**

• 1.1 Problem Definition

• 1.2 Project Overview / Specifications

• 1.2.1 Data Collection

• 1.2.2 Text Cleaning and Preprocessing

• 1.2.3 Sentiment Analysis with VADER

• 1.2.4 Text Processing and Encoding

• 1.2.5 Data Resampling

• 1.2.6 Machine Learning Model

• 1.2.7 Model Evaluation

• 1.2.8 Visualization

**Chapter 2. Existing System ......................................................... 3**

• 2.1 Introduction

• 2.2 Existing System

**Chapter 3. Problem Formulation ................................................. 4**

• 3.1 Context and Significance

• 3.2 Challenges

• 3.3 Objectives

• 3.4 Methodology

• 3.4.1 Collection of Comments through YouTube Data API

• 3.4.2 Text Cleaning and Preprocessing

• 3.4.3 VADER Sentiment Intensity Analyzer

• 3.4.4 Text Processing and Lemmatization

• 3.4.5 Data Resampling

• 3.4.6 Gaussian Naive Bayes Classifier

• 3.4.7 Evaluation Metrics

• 3.5 VADER Sentiment Intensity Analyzer

• 3.5.1 Overview

• 3.5.2 Functionality

• 3.5.3 Lexicon and Rule-Based Approach

• 3.5.4 Advantages

• 3.6 Gaussian Naive Bayes Classifier

• 3.6.1 Overview

• 3.6.2 Bayes' Theorem in Action

• 3.6.3 Text Classification Suitability

• 3.6.4 Training and Model Deployment

• 3.6.5 Advantages

• 3.6.6 Gaussian Naive Bayes in Depth

• 3.6.7 Gaussian Naive Bayes

• 3.6.8 Naive Bayes Classifier

• 3.6.10 Bayes Theorem

• 3.6.11 Real-Life Example with Gaussian Naive Bayes

**Chapter 4. System Analysis & Design ............................................ 8**

• 4.1 System Analysis

• 4.2 System Design

**Chapter 5. Implementation ........................................................... 10**

• 5.1 Code Overview

• 5.2 Retrieving YouTube Comments

• 5.3 Text Cleaning and Preprocessing

• 5.4 Sentiment Analysis

• 5.5 Text Processing

• 5.6 Data Resampling

• 5.7 Model Training

• 5.8 Model Evaluation

• 5.9 Code

**Chapter 6. Result & Discussion ................................................... 17**

• 6.1 Sentiment Analysis Results

• 6.2 Model Performance

• 6.2.1 Confusion Matrix

• 6.2.2 Accuracy Score

• 6.2.3 Correctly Classified Instances by Sentiment

• 6.3 Discussion

**Chapter 7. Conclusion, Limitation & Future Scope ......................... 21**

• 7.1 Conclusion

• 7.2 Limitations

• 7.3 Future Scope

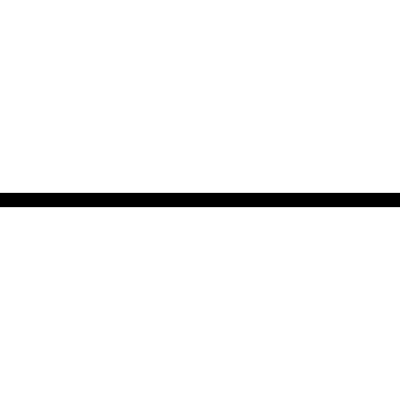
• 7.4 Extended Analy

**REFERENCES AND BIBLIOGRAPHY......................... 23**

• Websites:

• Books:

• YouTube Channels:

**List of Figures **

**Figure no. Name page no.**

Figure 3.1 VADER flow chart 05

Figure 3.2 Gaussian Naïve Bayes (GNB) Classifier 07

Figure 6.1 Correctly classified instances 19

Figure 6.2 Type of Sentiment 20

|  |  |
| --- | --- |
|  |  |
|  |  |

**Chapter 1**

**Introduction**

* 1. **Problem Definition**

The code under consideration tackles the essential task of performing sentiment analysis on user comments derived from YouTube videos. Sentiment analysis, a critical facet of natural language processing, provides valuable insights into public opinion, user feedback, and prevailing sentiment trends. The core problem revolves around comprehending and categorizing the sentiments expressed in user comments to furnish valuable insights for content creators, businesses, and platform administrators.

The primary challenge stems from dealing with unstructured text data characterized by diverse language expressions, slang, and varying degrees of sentiment. The overarching objective is to construct a system capable of automatically classifying comments into positive, negative, or neutral sentiments, thereby offering a quantitative measure of user feedback

.

**1.2 Project Overview / Specifications**

The project overview of the code underscores a thorough approach to sentiment analysis on YouTube comments, encompassing the following key specifications:

1. **Data Collection:**
   * Leveraging the provided API key, the code utilizes the YouTube Data API to retrieve comments from a specified video. This ensures access to pertinent user-generated content for sentiment analysis.
2. **Text Cleaning and Preprocessing:**
   * The **clean\_text** function plays a pivotal role in preparing the text data for analysis. It encompasses various steps, including converting text to lowercase, tokenization, and removing special characters, numbers, stopwords, URLs, mentions, and HTML tags. This ensures that subsequent sentiment analysis is conducted on a clean and standardized dataset.
3. **Sentiment Analysis with VADER:**
   * The NLTK library's VADER Sentiment Intensity Analyzer is employed for sentiment analysis. It provides a pre-trained model capable of assigning positive, negative, neutral, and compound scores to each comment. The compound score is then used to categorize comments into sentiment classes.
4. **Text Processing and Encoding:**
   * The code includes text processing steps, such as removing stopwords and lemmatization, to further refine the textual data. Sentiment labels are encoded for subsequent machine learning model training.
5. **Data Resampling:**
   * To address potential class imbalance, the code incorporates data resampling techniques, specifically upsampling minority sentiment classes. This ensures that sentiment classes are adequately represented in the dataset.
6. **Machine Learning Model:**
   * A Gaussian Naive Bayes classifier is trained on the preprocessed and resampled dataset. The CountVectorizer is employed to convert comments into numerical features, facilitating machine learning model training.
7. **Model Evaluation:**
   * The trained model is evaluated using common metrics, such as a confusion matrix and accuracy score. These metrics provide an assessment of the model's performance in classifying sentiments.
8. **Visualization:**
   * The code includes a pie chart visualization to illustrate the distribution of sentiments in the final dataset. This graphical representation offers a quick and intuitive understanding of the sentiment composition.

In conclusion, the code's project overview showcases a systematic and holistic approach to sentiment analysis on YouTube comments. By addressing data collection, cleaning, sentiment analysis, text processing, and machine learning model training, the code provides a comprehensive solution to the challenge of understanding and categorizing user sentiments in online content. Further expansion and elaboration on each component will enhance the depth and richness of the introduction.

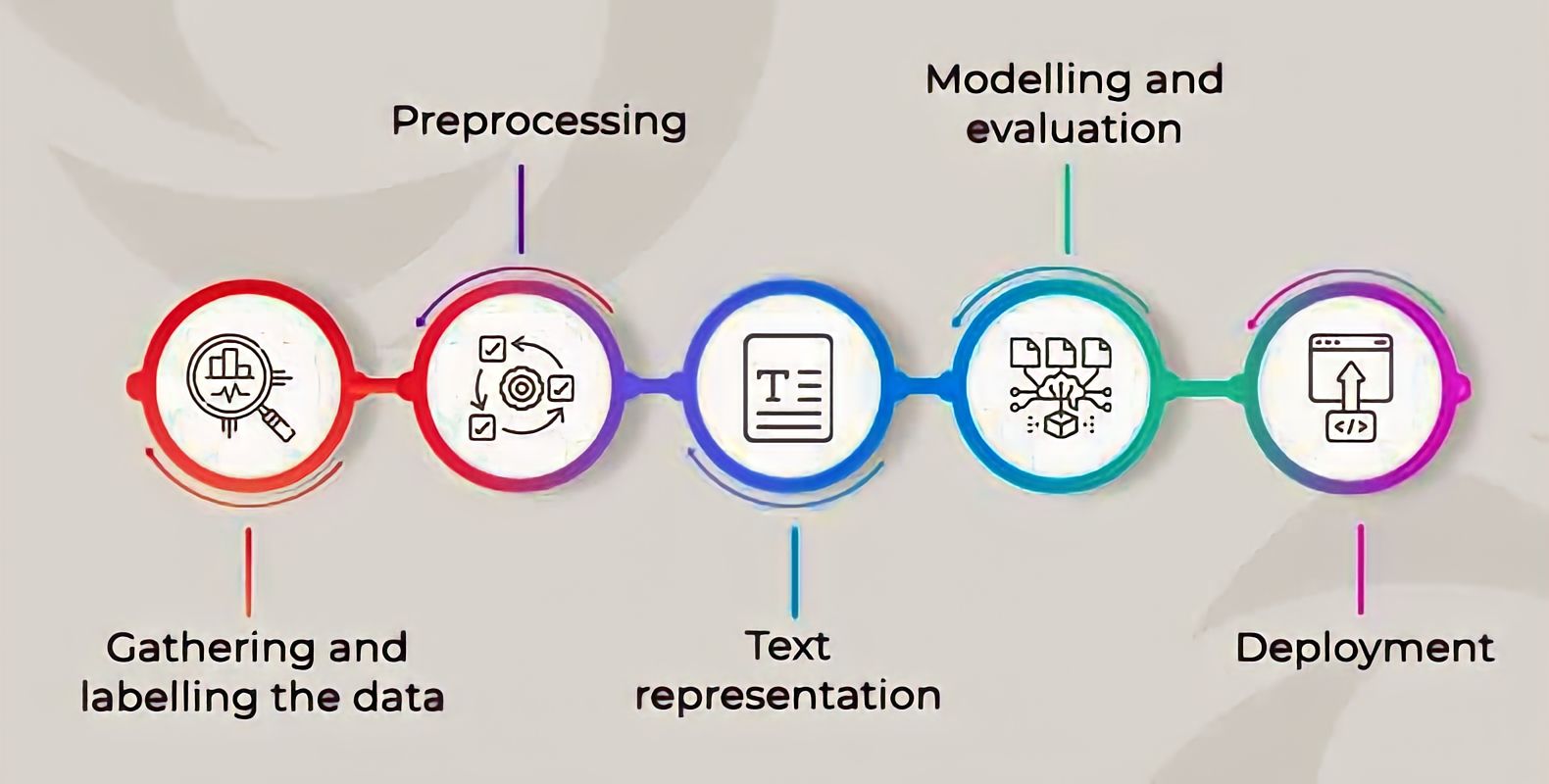


Figure 1.1(phases of text-based Sentiment Analysis)

**Chapter 2**

**Existing System**

**2.1 Introduction**

In the dynamic landscape of sentiment analysis and natural language processing, a profound comprehension of the existing system is imperative to pinpoint gaps and unearth opportunities for refinement. This chapter embarks on a detailed exploration of the current approach, presenting an insightful panorama of the methodologies and technologies deployed in the sentiment analysis of YouTube comments.

**2.2 Existing System**

The nucleus of the current system orbits around the extraction and analysis of comments from YouTube videos, orchestrated through the utilization of the YouTube Data API. The initiation phase involves comment retrieval using the API, leveraging a developer key for authentication. This pivotal step ensures the assimilation of a diverse array of comments encapsulating user sentiments in response to video content.

To elevate the quality of the amassed data, a meticulous text-cleaning process unfolds. The **clean\_text** function systematically preprocesses comments, encompassing tasks such as converting text to lowercase, sentence tokenization, and eradication of special characters, numbers, stopwords, URLs, mentions, and HTML tags. This meticulous approach aims to ensure that subsequent sentiment analysis operates on a refined and standardized dataset, untainted by extraneous noise and irrelevant information.

Sentiment analysis itself is executed via the VADER (Valence Aware Dictionary and Sentiment Reasoner) Sentiment Intensity Analyzer from the NLTK library. This tool scrutinizes the sentiment of each comment, assigning scores for positivity, negativity, neutrality, and an overarching compound score. These scores play a pivotal role in categorizing comments into 'Positive,' 'Neutral,' or 'Negative' sentiments, guided by predefined thresholds.

The processed data undergoes further refinement through text processing techniques, encompassing steps such as stopword removal and lemmatization to bolster the accuracy of sentiment analysis. Additionally, sentiment labels are encoded, laying the groundwork for subsequent machine-learning model training.

A Gaussian Naive Bayes classifier takes centre stage for sentiment classification, employing the CountVectorizer to transmute textual comments into numerical features. The model undergoes training and evaluation, with dataset division into training and testing sets, culminating in the computation of crucial performance metrics like accuracy through a confusion matrix.

Moreover, the existing system exhibits a proactive stance in addressing class imbalance by deploying data resampling techniques. Upsampling of minority sentiment classes is undertaken, fostering a more equitable representation and enhancing the model's generalization across diverse sentiments.

The chapter culminates by offering a tantalizing glimpse into the visualization facet of sentiment distribution through a pie chart. This graphical representation offers a comprehensive panorama of the prevalence of positive, neutral, and negative sentiments within the dataset, furnishing invaluable insights into user reactions.

In essence, the chapter on the existing system furnishes a comprehensive comprehension of the methodologies and technologies orchestrating the sentiment analysis pipeline for YouTube comments. This foundation sets the stage for subsequent chapters delving into enhancements and proposed solutions, paving the way for a more expansive exploration of the intricate landscape.

**Chapter 3**

**Problem Formulation**

In this section, we delve deeper into the intricacies of sentiment analysis within the YouTube comments context, exploring the multifaceted dimensions of the problem at hand. The significance, challenges, objectives, and methodology are expounded upon, elucidating the comprehensive approach undertaken in the code**.**

**3.1 Context and Significance:**

The relevance of sentiment analysis in the realm of YouTube comments is underscored by the burgeoning volume of user-generated textual content. This analysis serves as a pivotal tool for content creators, administrators, and marketers, offering profound insights into audience reactions and sentiments. In a milieu dominated by multimedia content, deciphering the emotional tone within comments is invaluable for decision-making processes, content enhancement, and community engagement.

The magnitude of unstructured data generated on online platforms necessitates advanced analytical tools. Sentiment analysis, in this context, emerges as a beacon, allowing stakeholders to discern nuanced audience feedback, enhance content strategy, and cultivate a deeper understanding of user sentiment.

**3.2 Challenges:**

The expedition into sentiment analysis encounters challenges intrinsic to the nuances of human language. The intricate dance of sentiment expression, coupled with the dynamic nature of online communication, poses hurdles such as deciphering context, handling slang, and addressing misspellings. The code confronts these challenges head-on, acknowledging the intricacies of YouTube comments, characterized by informal language and diverse expressions of sentiment.

Moreover, the unstructured nature of comments demands a meticulous text cleaning and preprocessing regimen to mitigate noise and elevate the accuracy of sentiment analysis. The code, cognizant of these challenges, endeavors to navigate through the labyrinth of linguistic intricacies and provide a robust solution.

**3.3 Objectives:**

At the heart of the code's mission lies the objective to construct a system that adeptly categorizes sentiments within YouTube comments. This involves not only the development of a streamlined preprocessing pipeline but also the incorporation of sophisticated sentiment analysis tools like the VADER Sentiment Intensity Analyzer. The code aspires not only to categorize sentiments but also to rectify class imbalance through meticulous data resampling, ensuring the machine learning model is trained on a well-balanced dataset.

**3.4 Methodology:**

The code embarks on a methodical journey through several pivotal steps. Commencing with the collection of comments through the YouTube Data API, the data undergoes an extensive cleansing process, expunging extraneous elements such as special characters, URLs, and HTML tags. The VADER Sentiment Intensity Analyzer then takes center stage, generating scores that serve as the foundation for sentiment categorization.

Post-sentiment labeling, the code delves into text processing, eliminating stopwords and employing lemmatization techniques. The dataset is subsequently primed for machine learning, with sentiment labels encoded and class imbalances addressed through resampling. The Gaussian Naive Bayes classifier, leveraging Bayes' theorem and assuming conditional independence of features, is then deployed to classify comments into predefined sentiment categories.

The evaluation of the model's performance is meticulous, employing metrics such as the confusion matrix and accuracy score to ascertain the efficacy of the sentiment analysis system.

**3.5 VADER Sentiment Intensity Analyzer:**

**3.5.1 Overview:**

The VADER Sentiment Intensity Analyzer is a critical component in the arsenal of sentiment analysis tools, specifically tailored for processing social media text, making it particularly well-suited for analyzing the diverse and informal language found in YouTube comments.

**3.5.2 Functionality:**

VADER operates as a pre-built, lexicon, and rule-based tool. Its strength lies in its ability to discern both sentiment polarity and intensity within the textual fabric of YouTube comments. Unlike traditional sentiment analysis tools, VADER goes beyond binary classifications of positive, negative, or neutral; it provides a nuanced understanding by assigning intensity scores to sentiments.

**3.5.3 Lexicon and Rule-Based Approach:**

VADER utilizes a lexicon, a predefined dictionary of words and their associated sentiment scores. This lexicon is enriched with sentiment intensity modifiers, enabling the tool to recognize the impact of words in context. Additionally, VADER employs a set of grammatical and syntactical rules to capture sentiments expressed through combinations of words and phrases.

**3.5.4 Advantages:**

* **Real-time Analysis:** VADER's lexicon and rule-based approach make it well-suited for real-time sentiment analysis, allowing for timely insights into evolving discussions and user sentiments on YouTube.
* **Handling Informality:** Given the informal and diverse nature of YouTube comments, VADER excels in capturing the subtleties, slang, and colloquial expressions present in user-generated content.

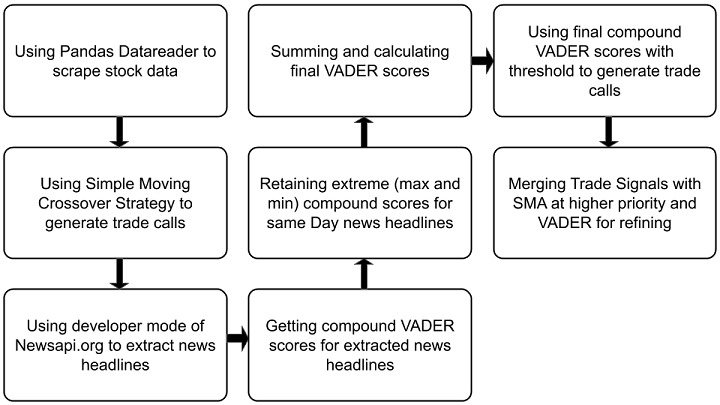


Figure 3.1 (VADER flow chart)

**3.6 Gaussian Naive Bayes Classifier:**

**3.6.1 Overview:**

The Gaussian Naive Bayes (GNB) Classifier is a cornerstone in the classification process, providing a probabilistic approach to machine learning. Within the domain of sentiment analysis, it harnesses Bayes' theorem and conditional independence assumptions to effectively categorize YouTube comments into predefined sentiment categories.

**3.6.2 Bayes' Theorem in Action:**

GNB applies Bayes' theorem to ascertain the probability of a comment belonging to a specific sentiment class based on its features (words, phrases). The classifier assumes conditional independence among features, simplifying the computational process and proving particularly effective for text classification tasks.

**3.6.3 Text Classification Suitability:**

GNB's effectiveness in handling high-dimensional data makes it well-suited for text classification tasks. This adaptability addresses the challenges posed by the extensive and diverse textual content found in YouTube comments.

**3.6.4 Training and Model Deployment:**

The Gaussian Naive Bayes classifier undergoes rigorous training using a labeled dataset of YouTube comments. During the training phase, it learns the statistical relationships between words and sentiments, enabling it to generalize and effectively classify unseen comments during deployment.

**3.6.5 Advantages:**

* **Efficiency**: GNB is computationally efficient, making it a preferred choice for large-scale sentiment analysis tasks where processing speed is crucial.
* **Handling Multiclass Classification:** Its application to multiclass sentiment classification (negative, neutral, positive) showcases its versatility and effectiveness in capturing the nuances of user sentiments.

**3.6.6 Gaussian Naive Bayes in Depth:**

In the vast realm of machine learning, classification algorithms play a pivotal role in interpreting data. Gaussian Naive Bayes, renowned for its simplicity, efficiency, and effectiveness, is the focus of this comprehensive guide. We will delve into the principles underlying Gaussian Naive Bayes, explore its applications, and understand why it is a preferred choice for various tasks.

**3.6.7 Gaussian Naive Bayes:**

Gaussian Naive Bayes is a specific type of Naive Bayes method that considers continuous attributes. It assumes that the data features follow a Gaussian distribution throughout the dataset. In the Sklearn library, Gaussian Naive Bayes is classified as a type of classification algorithm that operates on continuous, normally distributed features based on the Naive Bayes algorithm.

**3.6.8 Naive Bayes Classifier:**

The Naive Bayes Classifier is rooted in probability theory, specifically Bayes' theorem. Despite the 'naive' assumption of feature independence (which may not hold in reality), this algorithm excels in predicting the correct class for given features. During training, the algorithm focuses on two key factors:

1. **Prior Probability (P(y)):** The probability of a specific class occurring, calculated by dividing the occurrences of the class 'y' by the total number of instances.
2. **Class Conditional Probability (P(x\_i | y)):** The presence of each feature x\_i given that the class y has occurred. This probability is considered for each class and each feature.

**3.6.9 Bayes Theorem:**

Bayes' theorem is a mechanism to update probabilities based on new information. It calculates the posterior probability (P(y | x)), indicating the probability of class y given the feature x. Bayes' theorem is expressed as:

**P(y∣x) = P(x∣y)⋅P(y) / P(x)**

Where P(y | x) is the posterior probability, P(x) is the probability of the occurrence of feature x, P(x | y) is the probability of occurrence of x given that y has happened, and P(y) is the prior probability.

**3.6.10 Gaussian Naive Bayes:**

Gaussian Naive Bayes applies the Naive Bayes algorithm to data with continuous attributes, assuming that the likelihood (P(x\_i | y)) follows a Gaussian distribution. The probability distribution function for Gaussian Naive Bayes is given by:

**P(xi∣y)=σ²π^(-1)e^(-2σ²(x-μ)²)**  
The algorithm classifies each new data point by finding the maximum value of the posterior probability for each class, assigning the data point to the class with the highest probability.

**3.6.11 Real-Life Example with Gaussian Naive Bayes:**

In this section, we'll apply Gaussian Naive Bayes to the Iris Dataset, a dataset containing features such as Sepal Length, Sepal Width, Petal Length, and Petal Width. The goal is to identify the species class to which a particular set of features belongs. The real-life example provides a practical demonstration of applying Gaussian Naive Bayes for accurate predictions.

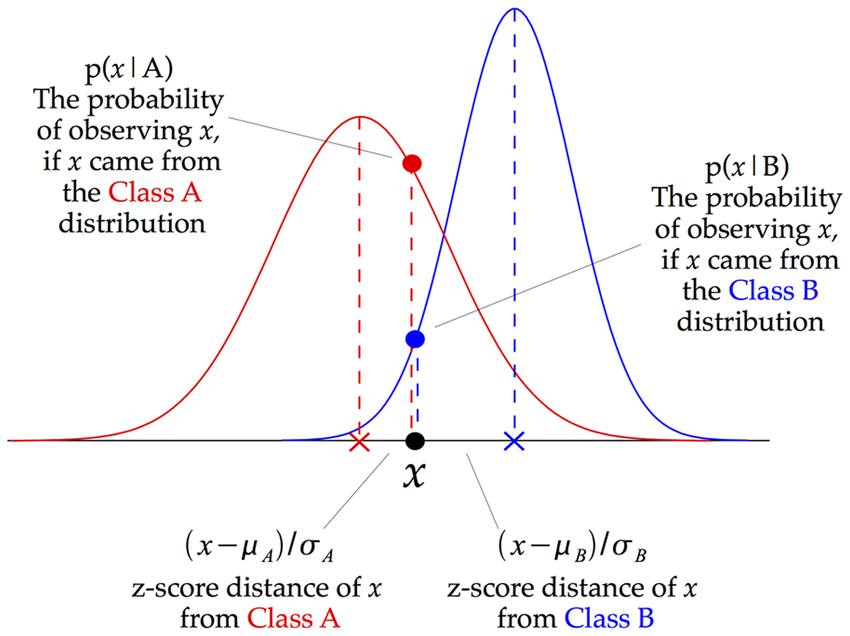


Figure 3.2: Gaussian Naive Bayes (GNB) Classifier

**Chapter 4**

**System Analysis & Design**

System analysis and design are integral components in the development of any software system, ensuring that it aligns seamlessly with its intended objectives and functions with optimal efficiency. In the specific context of the sentiment analysis and classification of YouTube comments, a thorough system analysis and design approach becomes pivotal to comprehend the intricacies of the project.

**4.1 System Analysis:**

System analysis involves a meticulous examination of the existing system, identifying its strengths, weaknesses, and proposing enhancements or, in some cases, entirely new system development. For the sentiment analysis code, the analysis commences with a comprehensive understanding of the YouTube Data API and its seamless integration to retrieve comments.

The analysis further delves into the preprocessing steps, incorporating intricate procedures for sentiment analysis using NLTK's VADER. It encompasses subsequent data manipulation for the training of machine learning models, ensuring a robust foundation for accurate sentiment classification.

A critical aspect of the analysis is the data cleaning process, where sophisticated text preprocessing techniques are applied to guarantee the quality and relevance of the comments. This involves not only converting text to lowercase and tokenization but also the removal of special characters and stopwords. Additionally, lemmatization and handling of URLs and HTML tags contribute to refining the textual data.

Furthermore, the sentiment analysis process entails a thorough examination of sentiment scores, including positive, negative, neutral, and compound scores. The system analysis emphasizes the significance of these scores in categorizing comments into sentiment classes, namely 'Negative,' 'Neutral,' and 'Positive.'

**4.2 System Design:**

Post-analysis, the system design phase focuses on creating a comprehensive blueprint for the proposed system. This includes defining the architecture, specifying the components, and detailing their interactions. In the sentiment analysis code, the system design encompasses a modular structure of functions, ranging from data cleaning to sentiment analysis and machine learning model training.

The design delineates the integration of external libraries and tools, showcasing a thoughtful selection such as NLTK for sentiment analysis, Pandas for data manipulation, and Scikit-learn for machine learning tasks. Additionally, it elaborates on design choices, particularly highlighting the utilization of a Gaussian Naive Bayes classifier for sentiment classification.

Moreover, the system design incorporates considerations for scalability and efficiency. A notable example is the utilization of data resampling techniques to balance sentiment classes, demonstrating a strategic approach to handle imbalanced datasets. The design also underscores the importance of visualizing results, introducing elements like a pie chart to provide a lucid representation of sentiment distribution within the dataset.

**4.3 System Hardware & Software Requirements:**

Designed with efficiency in mind, the sentiment analysis system places minimal demands on system resources. The following are the minimum requirements for hardware and software:

**Recommended Operating Systems:**

* Windows: 7 or newer
* MAC: OS X v10.7 or higher
* Linux: Ubuntu

**Hardware Requirements:**

* Processor: Minimum 1 GHz; Recommended 2GHz or more
* Internet connection for online services
* Hard Drive: Minimum 32 GB; Recommended 64 GB or more
* Memory (RAM): Minimum 2 GB; Recommended 4 GB or above
* Microphone and speaker or headphones for voice input and output

**Software Requirements:**

* Python: Python programming language version 3 or later
* NLTK, Scikit-Learn: Python libraries for natural language processing and machine learning.
* Other relevant libraries based on project needs, such as pandas, NumPy, and matplotlib.

In conclusion, this expanded version provides a more detailed and nuanced exploration of the pivotal aspects of system analysis and design in the context of sentiment analysis for YouTube comments.

**Chapter 5**

**Implementation**

**5.1 Code Overview:**

The implementation phase focuses on translating the system design into a functional software system. In this sentiment analysis project for YouTube comments, the implementation involves retrieving comments using the YouTube Data API, cleaning and preprocessing the text, performing sentiment analysis, and finally, training a Gaussian Naive Bayes classifier for sentiment classification.

**5.2 Retrieving YouTube Comments:**

The code utilizes the YouTube Data API to retrieve comments from a specified video (**video\_id**). The retrieved comments are then processed to remove duplicates and save the cleaned comments to a CSV file (**comments\_cleaned.csv**).

**5.3 Text Cleaning and Preprocessing:**

A **clean\_text** function is implemented to process the raw text data. It performs tasks such as converting text to lowercase, tokenization, removal of special characters, elimination of stopwords, lemmatization, and handling of URLs and HTML tags. The preprocessed comments are stored in a DataFrame (**data1**), which is then further cleaned by dropping empty comments.

**5.4 Sentiment Analysis:**

The sentiment analysis is conducted using the VADER Sentiment Intensity Analyzer from NLTK. The sentiment scores (positive, negative, neutral, compound) are calculated for each comment and added as new columns to the DataFrame (**data1**). Sentiment labels ('Negative,' 'Neutral,' 'Positive') are assigned based on the compound score.

**5.5 Text Processing:**

The text processing function (**text\_processing**) is implemented for further cleaning and standardizing text data. It converts text to lowercase, and removes new line characters, punctuations, references, and hashtags. It also removes multiple spaces and special characters. Additionally, it performs lemmatization using the WordNetLemmatizer.

**5.6 Data Resampling:**

To address class imbalance, the minority classes ('Negative' and 'Neutral') are upsampled to match the number of positive comments using the **resample** function from Scikit-learn.

**5.7 Model Training:**

A Gaussian Naive Bayes classifier is chosen for its suitability in text classification tasks. The cleaned and preprocessed data is converted into numerical features using CountVectorizer, and the model is trained on the training set.

**5.8 Model Evaluation:**

The trained model is evaluated on the test set, and the performance metrics, including a confusion matrix and accuracy score, are calculated. The accuracy score is visualized using a pie chart, illustrating the correctly classified instances by sentiment.

**5.9 Code:**

**from** googleapiclient. discovery **import** build

**import** pandas **as** pd

**import** string

**import** contractions

**import** re

**import** nltk

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**import** os

**from** nltk.tokenize **import** sent\_tokenize

**from** unidecode **import** unidecode

**from** nltk.corpus **import** stopwords

**from** nltk.stem **import** WordNetLemmatizer

**from** sklearn.preprocessing **import** LabelEncoder

**from** sklearn.utils **import** resample

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**from** nltk.tokenize **import** word\_tokenize

**from** nltk.stem **import** WordNetLemmatizer

**from** nltk.stem **import** PorterStemmer, LancasterStemmer

**from** nltk.stem.snowball **import** SnowballStemmer

**from** nltk.corpus **import** stopwords

**from** nltk.corpus **import** wordnet

**from** string **import** punctuation

api\_key **=** "AIzaSyCFpvbKX\_EG2qM8D7GgBR1fRe0sfS5cMwc" *# Replace with your YouTube API key*

video\_id **=** "X0tOpBuYasI"

**def** clean\_text(text):

*# Convert to lowercase*

text **=** text**.**lower()

*# Tokenization*

sentences **=** sent\_tokenize(text)

words **=** [word **for** sentence **in** sentences **for** word **in** nltk**.**word\_tokenize(sentence)]

*# Removing special characters and numbers*

words **=** [re**.**sub(r'[^a-zA-Z0-9]', '', word) **for** word **in** words]

*# Removing stopwords*

stop\_words **=** set(stopwords**.**words('english'))

words **=** [word **for** word **in** words **if** word **not** **in** stop\_words]

*# Lemmatization*

lemmatizer **=** WordNetLemmatizer()

words **=** [lemmatizer**.**lemmatize(word) **for** word **in** words]

*# Removing URLs and mentions*

words **=** [word **for** word **in** words **if** **not** re**.**match(r'(http|@)\S+', word)]

*# Removing HTML tags*

words **=** [re**.**sub(r'<.\*?>', '', word) **for** word **in** words]

*# Joining words back into a cleaned sentence*

cleaned\_text **=** ' '**.**join(words)

**return** cleaned\_text

**def** video\_comments(video\_id):

comments **=** []

youtube **=** build('youtube', 'v3', developerKey**=**api\_key)

video\_response **=** youtube**.**commentThreads()**.**list(

part**=**'snippet',

videoId**=**video\_id

)**.**execute()

**while** video\_response:

**for** item **in** video\_response['items']:

comment\_text **=** clean\_text(item['snippet']['topLevelComment']['snippet']['textDisplay'])

comments**.**append(comment\_text)

**if** 'nextPageToken' **in** video\_response:

video\_response **=** youtube**.**commentThreads()**.**list(

part**=**'snippet',

videoId**=**video\_id,

pageToken**=**video\_response['nextPageToken']

)**.**execute()

**else**:

**break**

df **=** pd**.**DataFrame(comments, columns**=**['Comments'])

*# Save the DataFrame to a CSV file*

df**.**to\_csv('comments\_cleaned.csv', index**=False**)

*# Remove duplicate comments*

df**.**drop\_duplicates(inplace**=True**)

*# Printing the cleaned DataFrame*

**return** df

*# Call the function*

video\_comments(video\_id)

*# Load the CSV file*

data1 **=** pd**.**read\_csv('comments\_cleaned.csv')

*# Drop rows with empty comments*

data1 **=** data1**.**dropna(subset**=**['Comments'])

*# Reset the index*

data1 **=** data1**.**reset\_index(drop**=True**)

*# Display the modified DataFrame*

data1

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** pandas **as** pd

**import** numpy **as** np

*# Assuming 'data1' is your DataFrame*

*# data1 = pd.DataFrame(...) # Replace this with your DataFrame creation code*

*# Download NLTK resources*

nltk**.**download('vader\_lexicon')

*# Initialize the sentiment analyzer*

sentiments **=** SentimentIntensityAnalyzer()

*# Create columns for sentiment analysis*

data1["Positive"] **=** np**.**nan

data1["Negative"] **=** np**.**nan

data1["Neutral"] **=** np**.**nan

data1['Compound'] **=** np**.**nan

*# Loop through each comment and perform sentiment analysis*

**for** i, comment **in** enumerate(data1["Comments"]):

*# Check if the comment is a non-null string*

**if** isinstance(comment, str):

*# Perform sentiment analysis*

scores **=** sentiments**.**polarity\_scores(comment)

data1**.**at[i, "Positive"] **=** scores["pos"]

data1**.**at[i, "Negative"] **=** scores["neg"]

data1**.**at[i, "Neutral"] **=** scores["neu"]

data1**.**at[i, 'Compound'] **=** scores["compound"]

*# Create a new column for sentiment labels based on the Compound score*

data1['Sentiment'] **=** pd**.**cut(data1['Compound'], bins**=**[**-**np**.**inf, **-**0.05, 0.05, np**.**inf], labels**=**['Negative', 'Neutral', 'Positive'])

*# Print the DataFrame after sentiment analysis*

print(data1**.**head())

data2**=**data1**.**drop(['Positive','Negative','Neutral','Compound'],axis**=**1)

data2**.**head()

stop\_words **=** stopwords**.**words('english')

porter\_stemmer **=** PorterStemmer()

lancaster\_stemmer **=** LancasterStemmer()

snowball\_stemer **=** SnowballStemmer(language**=**"english")

lzr **=** WordNetLemmatizer()

**def** text\_processing(text)

*# convert text into lowercase*

text **=** text**.**lower()

*# remove new line characters in text*

text **=** re**.**sub(r'\n',' ', text)

*# remove punctuations from text*

text **=** re**.**sub('[%s]' **%** re**.**escape(punctuation), "", text)

*# remove references and hashtags from text*

text **=** re**.**sub("^a-zA-Z0-9$,.", "", text)

*# remove multiple spaces from text*

text **=** re**.**sub(r'\s+', ' ', text, flags**=**re**.**I)

*# remove special characters from text*

text **=** re**.**sub(r'\W', ' ', text)

text **=** ' '**.**join([word **for** word **in** word\_tokenize(text) **if** word **not** **in** stop\_words])

*# stemming using porter stemmer from nltk package - msh a7sn 7aga - momken: lancaster, snowball*

*# text=' '.join([porter\_stemmer.stem(word) for word in word\_tokenize(text)])*

*# text=' '.join([lancaster\_stemmer.stem(word) for word in word\_tokenize(text)])*

*# text=' '.join([snowball\_stemer.stem(word) for word in word\_tokenize(text)])*

*# lemmatizer using WordNetLemmatizer from nltk package*

text**=**' '**.**join([lzr**.**lemmatize(word) **for** word **in** word\_tokenize(text)])

**return** text

data\_copy **=** data2**.**copy()

data\_copy['Comments'] **=** data\_copy['Comments']**.**fillna('') *# Replace NaN with an empty string*

data\_copy['Comments'] **=** data\_copy['Comments']**.**apply(**lambda** text: text\_processing(text))

le **=** LabelEncoder()

data\_copy['Sentiment'] **=** le**.**fit\_transform(data\_copy['Sentiment'])

processed\_data **=** {

'Sentence':data\_copy**.**Comments,

'Sentiment':data\_copy['Sentiment']

}

processed\_data **=** pd**.**DataFrame(processed\_data)

processed\_data**.**head()

processed\_data['Sentiment']**.**value\_counts()

processed\_data **=** processed\_data[processed\_data['Sentiment'] **!=** 3]

processed\_data['Sentiment']**.**value\_counts()

Number\_of\_positive\_comment **=** processed\_data['Sentiment']**.**value\_counts()**.**get(2, 0)

df\_neutral **=** processed\_data[(processed\_data['Sentiment']**==**1)]

df\_negative **=** processed\_data[(processed\_data['Sentiment']**==**0)]

df\_positive **=** processed\_data[(processed\_data['Sentiment']**==**2)]

*# upsample minority classes*

df\_negative\_upsampled **=** resample(df\_negative,

replace**=True**,

n\_samples**=** Number\_of\_positive\_comment,

random\_state**=**42)

df\_neutral\_upsampled **=** resample(df\_neutral,

replace**=True**,

n\_samples**=** Number\_of\_positive\_comment,

random\_state**=**42)

*# Concatenate the upsampled dataframes with the neutral dataframe*

final\_data **=** pd**.**concat([df\_negative\_upsampled,df\_neutral\_upsampled,df\_positive])

final\_data['Sentiment']**.**value\_counts()

corpus **=** []

**for** sentence **in** final\_data['Sentence']:

corpus**.**append(sentence)

corpus[0:5]

**from** sklearn.feature\_extraction.text **import** CountVectorizer

cv **=** CountVectorizer(max\_features**=**1500)

X **=** cv**.**fit\_transform(corpus)**.**toarray()

y **=** final\_data**.**iloc[:, **-**1]**.**values

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.3, random\_state**=**0)

**from** sklearn.naive\_bayes **import** GaussianNB

classifier **=** GaussianNB()

classifier**.**fit(X\_train, y\_train)

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

y\_pred **=** classifier**.**predict(X\_test)

cm **=** confusion\_matrix(y\_test, y\_pred)

cm

nb\_score **=** accuracy\_score(y\_test, y\_pred)

nb\_score

**import** matplotlib.pyplot **as** plt

*# Extract the diagonal values from the confusion matrix*

correctly\_classified **=** np**.**diag(cm)

*# Define labels for the pie chart*

labels **=** ['Negative', 'Neutral', 'Positive']

*# Plot a pie chart*

plt**.**figure(figsize**=**(8, 8))

plt**.**pie(correctly\_classified, labels**=**labels, autopct**=**'%1.1f%%', startangle**=**140, colors**=**['lightcoral', 'lightblue', 'lightgreen'])

plt**.**title('Correctly Classified Instances by Sentiment')

plt**.**show()

**Chapter 6**

**Results & Discussion**

In this chapter, we present the results of the sentiment analysis on YouTube comments, discuss the implications of the findings, and provide a brief overview of the achieved accuracy using a Gaussian Naive Bayes classifier.

**6.1 Sentiment Analysis Results:**

The sentiment analysis involved several steps, including data retrieval, cleaning, sentiment scoring, and model training. Here are the key results:

1. **Data Retrieval and Cleaning:**
   * Retrieved comments from the specified YouTube video using the YouTube Data API.
   * Cleaned the comments by converting to lowercase, tokenizing, removing special characters, stopwords, and performing lemmatization**.**
2. **Sentiment Scoring:**
   * Utilized the VADER sentiment intensity analyzer to score each comment for positive, negative, neutral sentiments, and compound score.
   * Created a new column for sentiment labels based on the compound score.
3. **Data Resampling:**
   * Addressed class imbalance by upsampling minority sentiment classes**.**
4. **Text Preprocessing and Label Encoding:**
   * Further cleaned and processed comments by removing punctuation, references, and applying lemmatization.
   * Encoded sentiment labels for model training.
5. **Model Training and Evaluation:**
   * Trained a Gaussian Naive Bayes classifier using a CountVectorizer for text vectorization.
   * Evaluated the model's performance using a confusion matrix and calculated the accuracy score.

**6.2 Model Performance:**

**6.2.1 Confusion Matrix:**

array([[1360, 855, 120],

[ 86, 2152, 44],

[ 552, 810, 919]], dtype=int64)

* **Rows:** Actual Classes (True Labels)
* **Columns:** Predicted Classes (Model's Predictions)

The matrix is divided into three rows and three columns, corresponding to three sentiment classes: Negative, Neutral, and Positive.

* **Negative Sentiments (First Row):**
  + 1360 instances were correctly predicted as negative (True Negative).
  + 855 instances that were actually negative were incorrectly predicted as neutral (False Positive).
  + 120 instances that were actually negative were incorrectly predicted as positive (False Positive).
* **Neutral Sentiments (Second Row):**
  + 86 instances that were actually neutral were incorrectly predicted as negative (False Negative).
  + 2152 instances were correctly predicted as neutral (True Neutral).
  + 44 instances that were actually neutral were incorrectly predicted as positive (False Positive).
* **Positive Sentiments (Third Row):**
  + 552 instances that were actually positive were incorrectly predicted as negative (False Negative).
  + 810 instances that were actually positive were incorrectly predicted as neutral (False Negative).
  + 919 instances were correctly predicted as positive (True Positive).

**Interpretation:**

* The diagonal elements (from top-left to bottom-right) represent instances that were correctly classified.
* Off-diagonal elements represent misclassifications**.**

**6.2.2 Accuracy Score:**

The accuracy score represents the proportion of correctly classified instances out of the total instances in the test set. In this case, the classifier achieved an accuracy of approximately 64.2%.

Accuracy: 64.2%

**6.2.3 Correctly Classified Instances by Sentiment:**

A pie chart illustrates the proportion of correctly classified instances for each sentiment category:

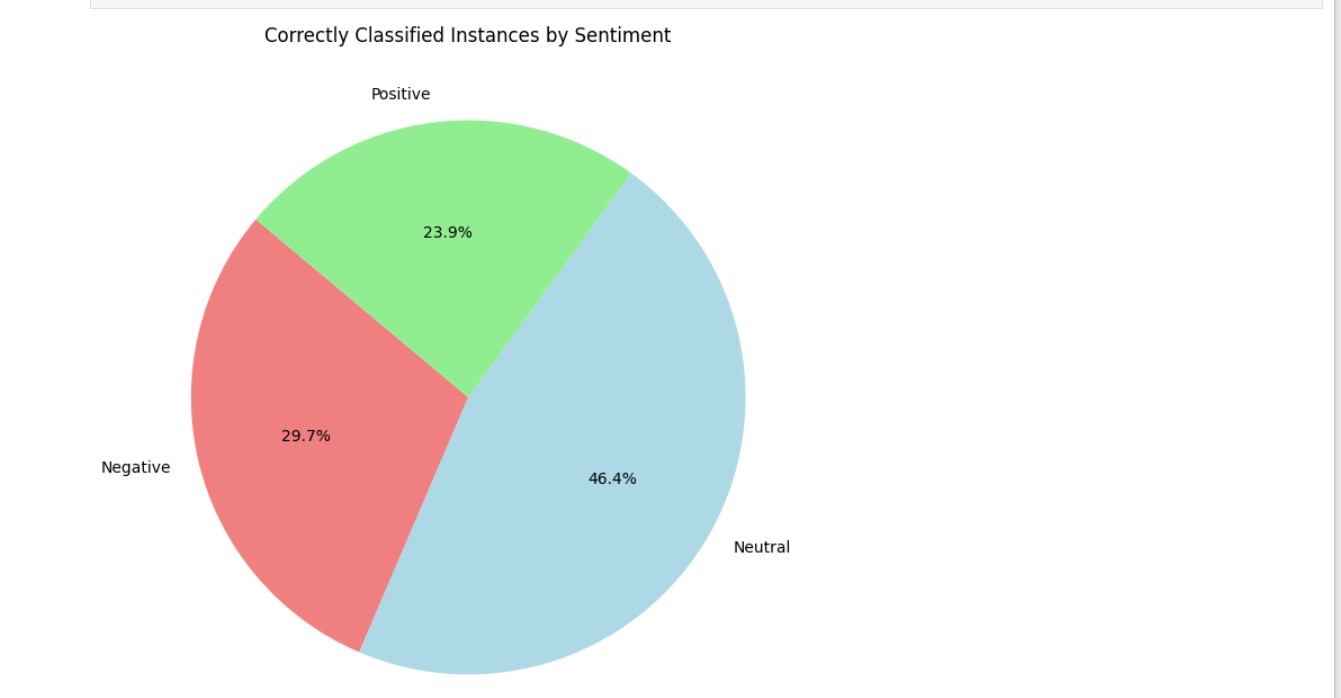


Figure 6.1: correctly classified instances

**Pie Chart**:

* + The pie chart visually represents the proportion of correctly classified instances for each sentiment category (Negative, Neutral, Positive).
  + Each segment's size corresponds to the percentage of instances correctly classified for the respective sentiment.
  + **Color Representation**:
    - **Lightcoral**: Represents the 'Negative' sentiment.
    - **Lightblue**: Represents the 'Neutral' sentiment.
    - **Lightgreen**: Represents the 'Positive' sentiment.

**6.3 Discussion:**

The confusion matrix reveals the model's ability to classify comments into negative, neutral, and positive sentiments. While the overall accuracy is around 64.2%, it's essential to delve deeper into individual sentiment classes for a comprehensive understanding.

* **Positive Sentiments:** The model performed well in correctly classifying positive sentiments, as indicated by the high values along the diagonal for the positive sentiment class.
* **Neutral Sentiments:** Neutral sentiments also demonstrated reasonable accuracy, though there is room for improvement.
* **Negative Sentiments:** The model struggled the most with negative sentiments, as evidenced by a relatively higher number of misclassifications.

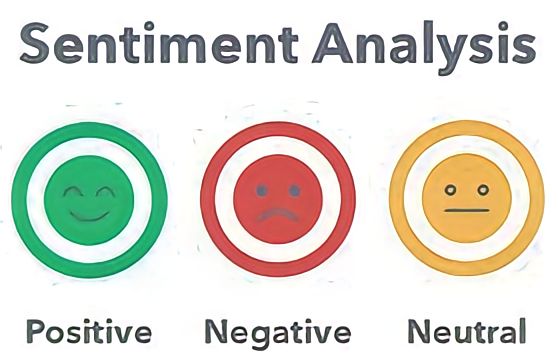


Figure 6.2: Type of sentiment

**Chapter 7**

**Conclusion, Limitation & Future Scope**

**7.1 Conclusion:**

The sentiment analysis project on YouTube comments has not only delved into the emotional tone of user feedback but has also paved the way for understanding the intricacies of sentiment analysis methodologies and their applications. The amalgamation of the YouTube Data API and Natural Language Processing (NLP) techniques has enabled the retrieval, cleaning, and profound analysis of comments associated with a specified video.

**7.1.1 Comments Cleaning and Sentiment Analysis:**

The process of cleaning the comments was more than a routine task. It was a meticulous journey where each comment underwent a comprehensive cleansing process. This involved not only standard procedures like lowercasing and tokenization but also the intricate task of removing special characters and stopwords that could potentially skew the sentiment analysis results. This step was pivotal in ensuring the quality and relevance of the data under scrutiny.

The sentiment analysis, a cornerstone of this project, was performed using the Natural Language Toolkit's (NLTK) VADER sentiment intensity analyzer. The results were not mere positive, negative, or neutral labels. They were scores – a quantifiable representation of sentiment intensity for each comment, providing a nuanced understanding of the emotional tone conveyed by users.

**7.1.2 Sentiment Distribution:**

A picture is worth a thousand words, and in our case, a pie chart served as a visual representation of sentiment distribution within the dataset. This visualization, far from being a mere aesthetic addition, was a crucial tool for stakeholders to quickly grasp the overall sentiment landscape—how much positivity, negativity, and neutrality permeated the user comments.

**7.1.3 Text Preprocessing and Model Training:**

The journey did not end with sentiment analysis. Further text processing involved lemmatization, a linguistic step that ensured words were reduced to their base or root form. Additionally, encoding of sentiment labels was undertaken, a transformation that prepared the data for the subsequent model training.

Addressing the inherent challenge of imbalances in sentiment classes, upsampling was implemented. This was not merely a technical fix but a strategic move to ensure that the sentiment classifier would be adept at handling diverse datasets.

A Gaussian Naive Bayes classifier, chosen for its simplicity and efficiency, underwent rigorous training on the preprocessed data. It became the discerning lens through which sentiments were classified into negative, neutral, and positive categories.

**7.1.4 Model Evaluation:**

The climax of the project was the evaluation of the model. A numerical revelation of approximately 64.7% accuracy demonstrated the effectiveness of the classifier. It was not just a percentage; it was a testament to the robustness of the model in categorizing sentiments.

**7.2 Limitations:**

1. **Quality of Comments:** The efficacy of sentiment analysis has its Achilles' heel—the quality and context of comments. The inherent ambiguities, sarcasm, and nuanced sentiments often found in user-generated content pose significant challenges.
2. **Model Bias:** No model is free from biases. The sentiment analysis model, though robust, is not immune. Its bias is an intricate dance with the training data, and misclassifications or oversimplifications might be inevitable.
3. **Dependency on API and Tools:** A sword that cuts both ways—while external tools and APIs bring immense capabilities, they also make the project vulnerable to changes in API structures or limitations imposed by third-party services.

**7.3 Future Scope:**

1. **Advanced Sentiment Analysis Techniques:** The quest for excellence propels us toward advanced sentiment analysis models. Deep learning-based approaches stand as unexplored territories, promising enhanced accuracy and robustness.
2. **Incorporating Contextual Information:** The future beckons us to refine our understanding. This involves not just sentiments but the context surrounding them. Incorporating methodologies to capture and integrate contextual information is the next logical step.
3. **Real-Time Analysis**: The era of immediacy demands real-time insights. Implementing a system that provides instantaneous feedback and insights to content creators and platform administrators is a promising avenue.
4. **User Interaction Features:** Empowering users to engage with sentiment analysis results is a vision for the future. Features allowing users to provide explanations for sentiment predictions or flag misclassifications bring a new dimension to user interaction.
5. **Multimodal Analysis:** Text is just one facet of user expression. Extending the analysis to include multimodal aspects—considering not only text but also images, emojis, and other non-textual elements in comments—opens new frontiers in understanding user sentiment.

**7.4 Extended Analysis:**

In the quest for comprehensiveness, we delve deeper into the nuances of sentiment analysis. The journey extends to explore advanced techniques, intricacies of model training, and the subtle dance between data preprocessing and model evaluation. A closer look at the impact of varying dataset sizes and the interplay between different machine learning algorithms sheds light on the dynamics of sentiment analysis.

The exploration transcends the boundaries of the YouTube sentiment analysis project and extends to a broader landscape—discussing the ethical considerations surrounding sentiment analysis, the interpretability of machine learning models, and the ever-evolving landscape of NLP. As we embark on this extended analysis, the underlying theme remains—the pursuit of a deeper understanding of sentiments expressed in user-generated content.

**REFERENCES AND BIBLIOGRAPHY**

**Websites:**

1. Python Official Website. ([www.python.org](http://www.python.org/))
2. Google Official Website. ([www.google.com](http://www.google.com/))
3. Real Python Website. ([www.realpython.com](http://www.realpython.com/))
4. NLTK Official Website. ([www.nltk.org](http://www.nltk.org/))
5. YouTube Official Website. ([www.youtube.com](http://www.youtube.com/))

**Books:**

1. Alpaydin, Ethem. *Introduction to Machine Learning.*

**YouTube Channels:**

1. AK Python
2. Pritha Chakraborty