**Nagarjuna College of Information Technology**

**Sankhamul, Kathmandu**

**Department of Computer Science and Information Technology**

****

**Project Report on**

**“Youtube Video Sentiment Analysis”**

**Submitted By:**

Anish Basnet - 20883/075

Ganesh Tiwari - 20885/075

Madan Sunwar - 20889/075

**Submitted To:**

Department of Computer Science and Information Technology Nagarjuna College of Information Technology

**Supervisor**

Ramesh Singh Saud

**Table of Content**

**Chapter 1 : Introduction 1**

**1.1. Introduction 1**

**1.2. Problem Statement 1**

**1.3. Objectives 2**

**1.4. Scope and Limitations 2**

**1.5. Development Methodology 2**

**1.6. Report Organization 4**

**Chapter 2 : Background Study and Literature Review 5**

**2.1. Background Study 5**

**2.2. Literature Review 6**

**Chapter 3 : System Analysis8**

**3.1. Requirement Analysis 8**

**3.1.1. Functional Requirement 8**

**3.1.2. Non Functional Requirement 9**

**3.2. Feasibility Analysis 9**

**3.2.1. Technical 9**

**3.2.2. Operational 9**

**3.2.3. Economic 9**

**3.2.4. Schedule (Gantt chart) 10**

**3.3. Analysis 10**

**3.3.1. Object Modeling Using Class and Object Diagram 11**

**3.3.2. Dynamic Modelling Using State and Sequence Diagram 13**

**3.3.3. Process Modeling Using Activity Diagram 15**

**Chapter 4 : System Design 16**

**4.1. Design 16**

**4.1.1. Refined Sequence and Activity Diagram 16**

**4.1.2. Component Diagram 18**

**4.1.3. Deployment Diagram 19**

**4.2. Algorithm Details 19**

**Chapter 5 : Implementation and Testing26**

**5.1. Implementation 26**

**5.1.1. Tools Used26**

**5.1.2. Implementation Details of Modules27**

**5.2. Testing 28**

**5.2.1. Test Cases for Unit Testing29**

**5.2.2. Test Cases for System Testing 31**

**5.3. Result Analysis 31**

**Chapter 6 : Conclusion 32**

**6.1. Conclusion 32**

**6.2. Future Recommendations 32**

# 

# **CHAPTER 1**

# **INTRODUCTION**

**1.1 Introduction**

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Generally speaking, sentiment analysis aims to determine the attitude of a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be their judgement towards the certain things. Sentimental analysis is additionally referred as opinion mining that means to find out or identify the positive, negative, views, attitudes, impressions, emotions, and feelings indicated in the text.

In this work, we will collect the data from the you-tube comments of the public and measures the attitude of the user towards the aspects of a video which they describe in a text and then give rating to the video on basis of user comments. Sentiment analysis is useful for quickly gaining the whole idea by using large number of text data and it will be helpful to understand the user’s opinion. YouTube provides many social mechanisms to gauge user opinion and views a few videos by means of voting, rating, favorites, sharing and negative comments, etc. Text analytic is that the analysis of “unstructured” data contained in natural language text using various methods machine learning tools, and techniques. Text analysis offers a very low-cost method to gauge public opinion.

We have developed a system based on Naive-Bayes classification algorithm which is based on principle of Bayes theorem which plays vital role to determine polarity of a sentence. To improve our system performance, we have used different features selection techniques like tokenization, stop words removal, lower casing the comments and punctuation removal.

**1.2 Problem Statement**

Generally, users watch YouTube videos on the basis of like and views, but the videos might not have a positive opinion from the users. The actual opinion/comment of the users may not be justified by the likes and views videos get. This has proven to be a challenge to identify the response of users about the video.

**1.3 Objectives**

The main objective of the system is:

* To scrap the YouTube comments and analyze the sentiment of the users about the video checking the polarity using Naive Bayes theorem.

## **1.4 Scope and Limitations**

### **1.4.1 Scope**

Sentiment analysis System aim helps the content creators as it solves the problems of analyzing each comments for response of users. This system can be used as an online platform for proper review of the youtube comments.

### **1.4.2 Limitation**

The limitations of the Sentiment Analysis System are as follows:

* The system can’t generate revenue for now.
* It is difficult to process the text which have multiple sentiment.
* With the availability of limited data set some result might not be satisfactory.

An infrastructural barrier such as internet connectivity, electrical connection, etc. might interfere with how the system will work.

## **1.5 Development Methodology**

Before the designing and implementation phase, it is very important to perform a software development methodology that refers to structured processes involved when working on a project. It provides a platform for developers to work together more efficiently as a team. There are various types of SDLC (Software Development Life Cycle) process such as waterfall model, iterative model, spiral model, etc.

The project was developed using a waterfall model i.e., the traditional approach SDLC model. It is a model that emphasizes a linear progression from beginning to end of a project. The Waterfall methodology follows a chronological process and works based on fixed dates, requirements, and outcomes. With this method, the team members also tend to work on one phase at a time and other phase doesn’t begins unless the previous phase id finished. As objectives and requirements are cleared from the beginning, each team members knows what must be done when, and can efficiently can plan their schedule for the duration of the project. The phases do not overlap with each other and hence the outcome of current phase acts as an input for the next phase respectively. The waterfall model can be depicting as:

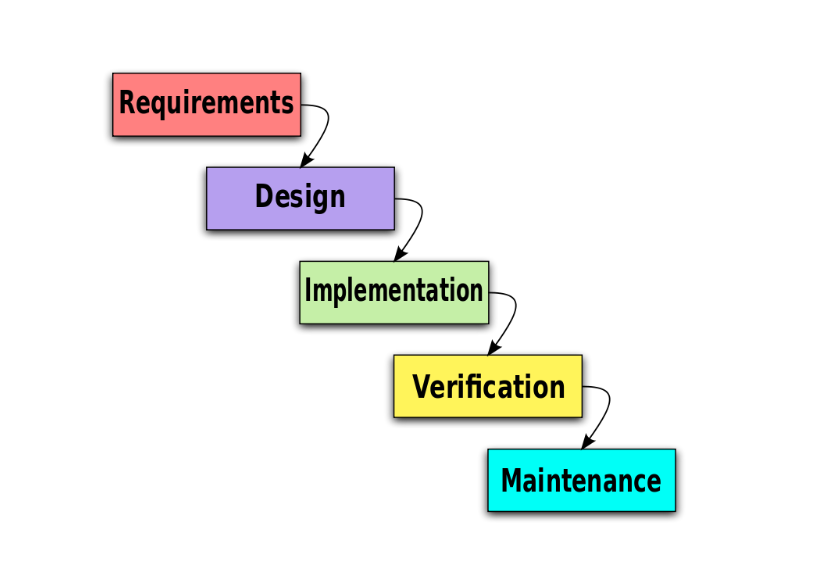


Figure 1.5: Waterfall model

## **1.6 Report Organization**

This report is divided into six chapters. Each chapter is further divided into different headings.

* **Chapter 1** gives an introduction to Sentiment Analysis System. The problem definition, objectives, scopes, and limitations of this system are discussed here.
* **Chapter 2** focuses on the background study. It contains a literature review section where the research works done in the field of the Sentiment analysis.
* **Chapter 3** focuses on the analysis part. It contains a requirement analysis, feasibility analysis and diagrams like class diagrams, sequence diagram, activity diagram and flowchart.
* **Chapter 4** discusses in detail the design of the system. This chapter also discusses databases design, interface design, and flowchart of the system built. This chapter also includes feasibility study, requirement analysis, and diagrams like ER, DFD, Flowchart
* **Chapter 5** gives information about the implementation and testing process. It discusses how the system is implemented and what tools and software are used to implement this system. The testing process is also included in detail in this chapter.
* **Chapter 6** includes the conclusion of the whole project. This chapter shows major achievements in the system and also shows how it can be enhanced later in the future

# **CHAPTER 2**

## **BACKGROUND STUDY AND LITERATURE REVIEW**

Sentiment analysis is the measure of people’s opinions on the level of agreement on a specific topic, a product, or a service, or even elections. Two approaches had been employed to study the sentiment analysis: natural language processing, and machine learning algorithms.

The growing phenomena of platforms such as: Youtube, Facebook, Twitter, Linkedin, and Instagram, with each one has its own characteristics and its usages, are constantly affecting out societies. Twitter, for example, is considered as a social network where everyone in the network has a reciprocated relationship with another one in the same network. The relationship in this case is undirected. Conversely, in Twitter everyone in the network does not necessarily have a reciprocated relationship with others. In this case, the relationship is either directed or undirected.

In 2020, Xu et al. [1] have introduced a NB method for multi-domain and large-scale E-commerce platform product review classification of sentiment. Consequently, the parameter evaluation method was extended in NB for continuous learning fashion. Later, for fine-tuning the learned distribution on the basis of three types of assumptions, many ways were introduced for acquiring the best performance. The results have shown that the suggested model has high accuracy in Amazon product and movie review sentiment datasets. In 2019, Afzaal et al. [2] have recommended a novel approach of aspect-based sentiment classification, which recognized the features in a precise manner and attained the best classification accuracy. Moreover, the scheme was developed as a mobile application, which assisted the tourists in identifying the best hotel in the town, and the proposed model was analyzed using the real-world data sets. The results have shown that the presented model was effective in both recognition as well as classification.

However, consider that emotions are the number one factor in making purchasing decisions. With so many consumers sharing their thoughts and feelings on social media, it quite literally pays for brands to have a pulse on how their products make people feel [3].

## **2.1 Background**

The growth of social media (e.g., blogs, forum discussions, reviews, micro-blogs) in the last decade, has substantially changed the web context to the extent that nowadays billions of people all around the globe are freely allowed to conduct many activities such as interacting, sharing, posting and manipulating contents on social media and the internet as general. This has enabled people to connect and interact with each other anytime without geographical boundaries. Sentiment analysis also known opinion mining is one class of computational techniques which automatically extracts and summarizes the opinions of such immense volume of data which the average human reader is unable to process [1]. This refers to identifying and classifying the sentiments that are expressed in the textsource. It is a powerful marketing tool that enables product managers to understand customer emotions in their marketing campaigns which is an important factor when it comes to product and brand recognition, customer loyalty and satisfaction, advertising and promotion's success, and product acceptance [2].

Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of the people about a variety of topics and for this main purpose sentiment analysis is effective as we are able to separate the positive comments from negative ones. That means searching for relevant terms which highlight customer sentiment.

Some sentiment terms are relatively straightforward and others might be specific to a particular industry. Either way, the sentiment terms need to be divided into positive and negative terms. Here is an example of what some of those terms might look like for a sentiment search.

* **Positive sentiments:**best, love, high-five, amazing, perfect, thanks
* **Negative sentiments:**worst, hate, ugh, disappointed, bad, avoid

## **2.2 Literature Review**

### **2.2.1 Similar System Study**

Lexalytics: The cloud-based Lexalytics offering performs as most sentiment analysis software does: It uses natural language processing to parse a customer's message, then performs sentiment analysis on the result to uncover the customer's underlying intention [3].

Talkwalker quick search: The sentiment analysis tool Quick Search is part of the larger suite Talkwalker, a full customer service platform. Quick Search is an integrated tool that aggregates mentions, likes, comments and other data that give insight into social media marketing efforts [3].

Social Searcher: Social Searcher is an inexpensive, social media-based toolkit that offers a simple up-or-down summary of positive sentiment and negative sentiment, based on entered keywords and hashtags. It breaks down its results by social media platform, apps and channels [3].

# **CHAPTER 3**

# **SYSTEM ANALYSIS**

## **3.1 System Analysis**

System analysis is the process of understanding and comparing the functional impacts of sub-systems to the total system. It identifies the problems and organizes the facts and details of the system.

### **3.1.1 Requirement Analysis**

As the system design evolves, requirements analysis activities support allocation and derivation of requirements down to the system elements representing the lowest level of the design. The functional and non-functional requirements are necessary to analyze the system requirements before developing and implementing.

### **3.1.2 Functional Requirements**

The functional requirements specify the documentation of the system and activities that a system must be able to perform.

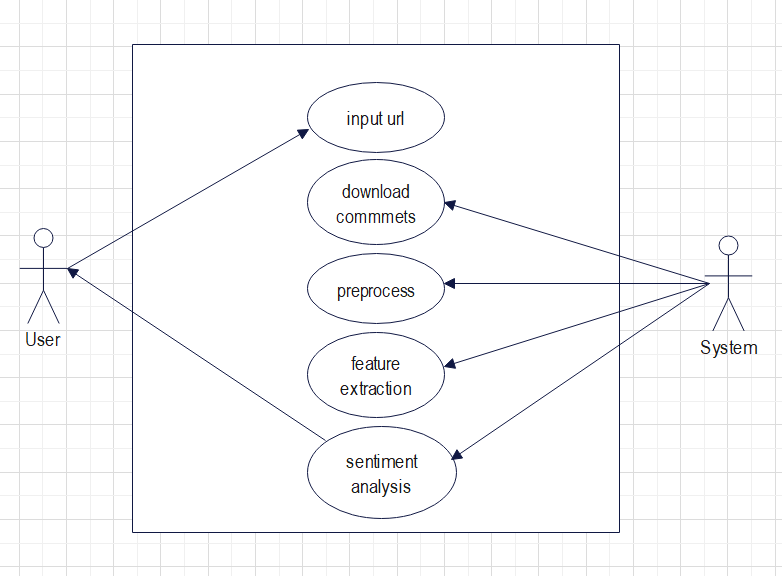


Figure 3.1.2 : Use Case Diagram

### **3.1.2 Non-functional Requirements**

Non-functional requirements are those type of requirements which is not directly concerned with the system functionality but in absence of it reduce the quality of the system process. The non-functional requirements in the context of the Sentiment Analysis System follow an interactive interface along with good performance providing scalability and reliability with maintainability.

## **3.2 Feasibility Study**

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats as presented by the environment, theresources required to carry through, and ultimately the prospects for success.

### **3.2.1 Economic Feasibility**

It is used for the evaluation of the effectiveness of the system. The system is economically feasible as existing tools and software are being used for developing and deploying the system is open source.

### 

### **3.2.2 Operational Feasibility**

It is concerned with the operating capabilities of the system. For the efficient operation, the reports and classification provided by our project (Sentiment Analysis) can help decision makers to have proper decisions. Result generated from the system are easier to read and understand. Hence, the system is feasible operationally.

### **3.2.3 Technical Feasibility**

It is a web-based application that uses Flask as front end and Python Technology as a back end. It will provide technical supports such as developing software, tools and, framework as freely available as open-source making it technically feasible.

### **3.2.4 Schedule Feasibility**

A picture containing graphical user interface

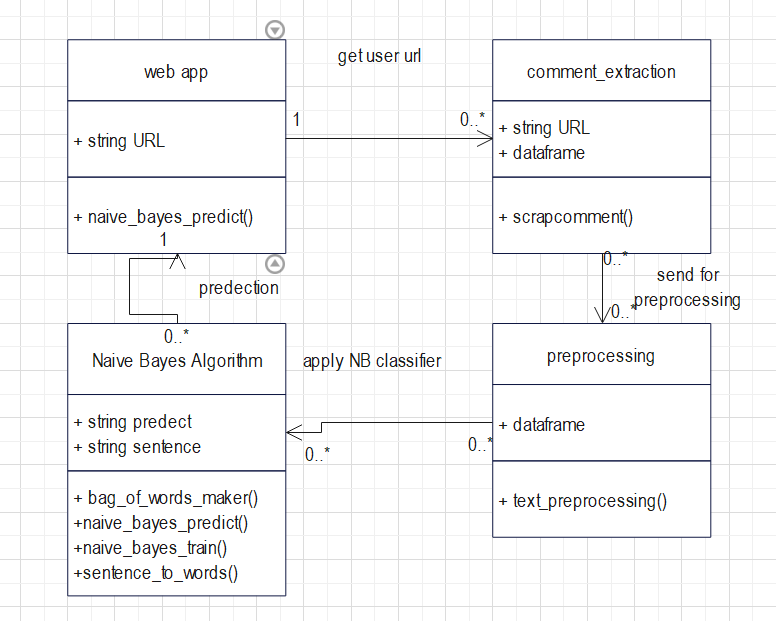
Description automatically generated

Figure 3.2.4: Working Schedule

## **3.3 Analysis**

### **3.3.1 Object Modeling using Class and Object Diagram**

#### **3.3.1.1 Class Diagram**

: Figure 3.3.1:Class Diagram

The class diagram is the building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application. The class diagram above shows the structure of the sentiment analysis system with the dependencies.

In above class diagram user gives URL through web app. The web app extracts comment and pre-process data then apply naive\_bayes\_predict() to predict the sentiment.

#### **3.3.1.2 Object Diagram**

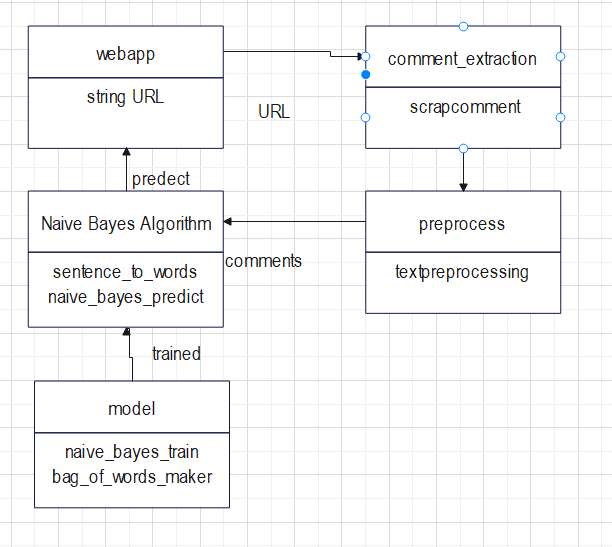


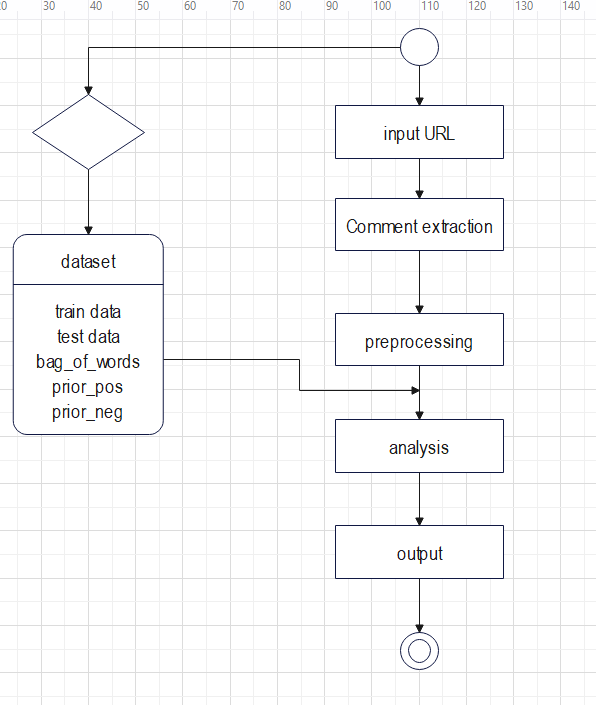
Figure3.3.1.2: Object Diagram

An object diagram is a graph instance including object and data values. Object diagram is an instance of class diagram. In above object diagram the URL given by the user is accessed by scrapcomment and preprocessed by text processor then predicted by trained NB model.

### **3.3.2 Dynamic modeling using state and sequence diagram**

A data flow diagram is a graphical representation of the flow of data through the system. It provides the information about the output and input of each entity and process itself.

#### **3.3.2.1 State Diagram**



3.3.2.1: State diagram

A state diagram is a graphical representation of a state machine which shows the behavioral model consisting of states, states transaction and action. In above state diagram dataset is split into train and test data and then bag\_of\_words prior\_pos prior\_neg are generated and connected to the analysis. In above input is a one state which is then preprocessed analyze and the output is acquired according to analysis.

#### **3.3.2.2 Sequence Diagram**

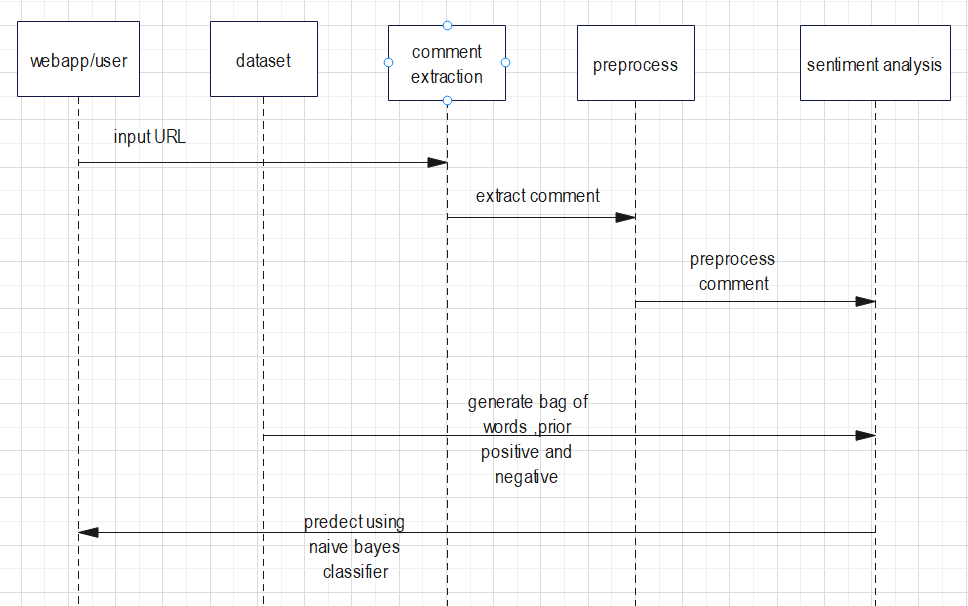


Figure 3.3.2.2: Sequence diagram

A sequence diagram is a structured in such a way that in represents the time line which begins at the top and descends gradually to mark the sequence of interaction. In above sequence diagram the first thing that took place is text input URL by user then the valid text dataset generates bag of words and prior positive and negative then comment is extracted then is preprocessed and sent to analyzer then it returns the sentiment of the text to the user.

### **3.3.3 Process Modeling with Activity Diagram**

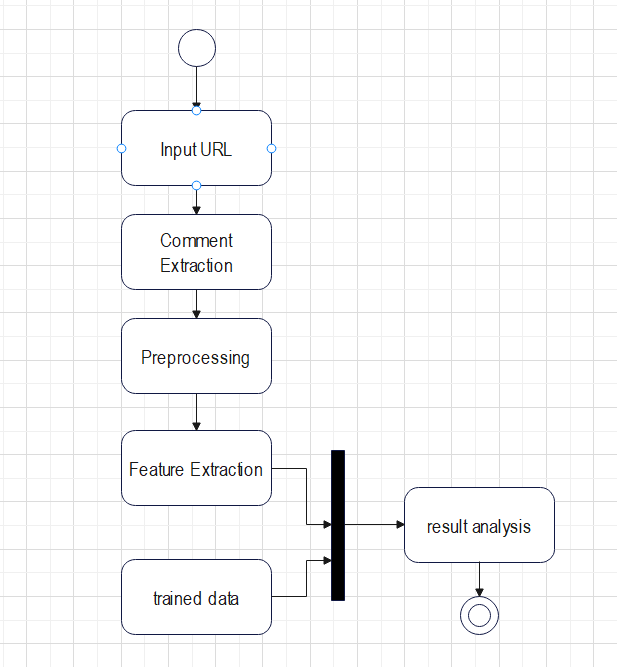


Figure 3.3.3: Activity Diagram

AN activity diagram is a behavioral diagram which portrays the control flow from start point to finish point showing the various decision paths that exists while the activity are executed.

# **CHAPTER 4**

# **SYSTEM DESIGN**

## **4.1 Design**

The phase consists of diagrams and design that helps to impart knowledge about the overall flow of the system. The detail entailment to the system is described as:

### **4.1.1**

### **Refined Sequence Diagram**

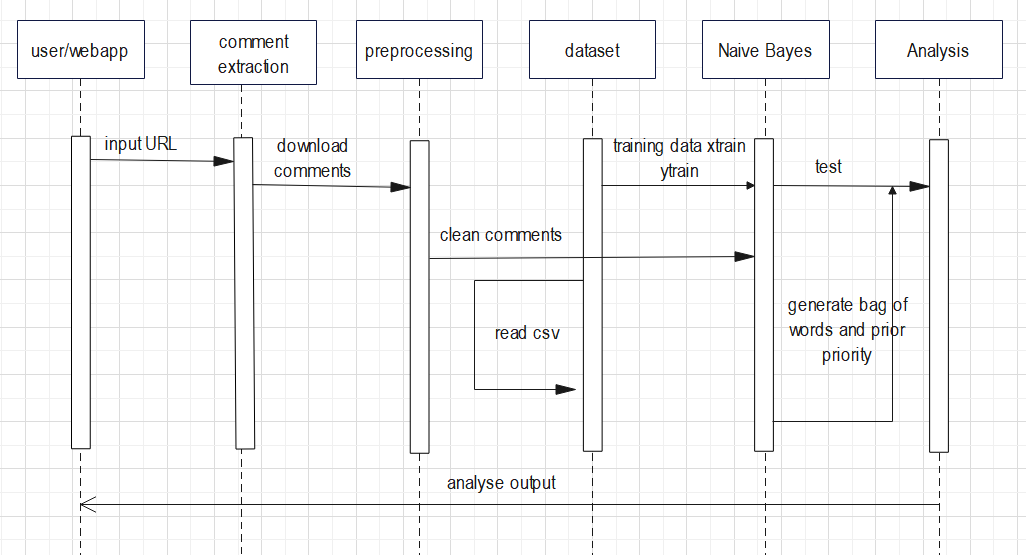


Figure 4.1.1: Refined Sequence Diagram

In above figure we built the complete system sequence. The input URL provides comments which are then extracted and dispatched to the dataset for training our model which reads the csv file. After reading the csv file then training and testing dataset is return to Naive Bayes classifier where Naive Bayes algorithm trained the model and generated prior probability and bag of words preserve it by pickling for further use. Then after result is analyzed and sent to the user with respective sentiment.

### **Refined Activity Diagram**

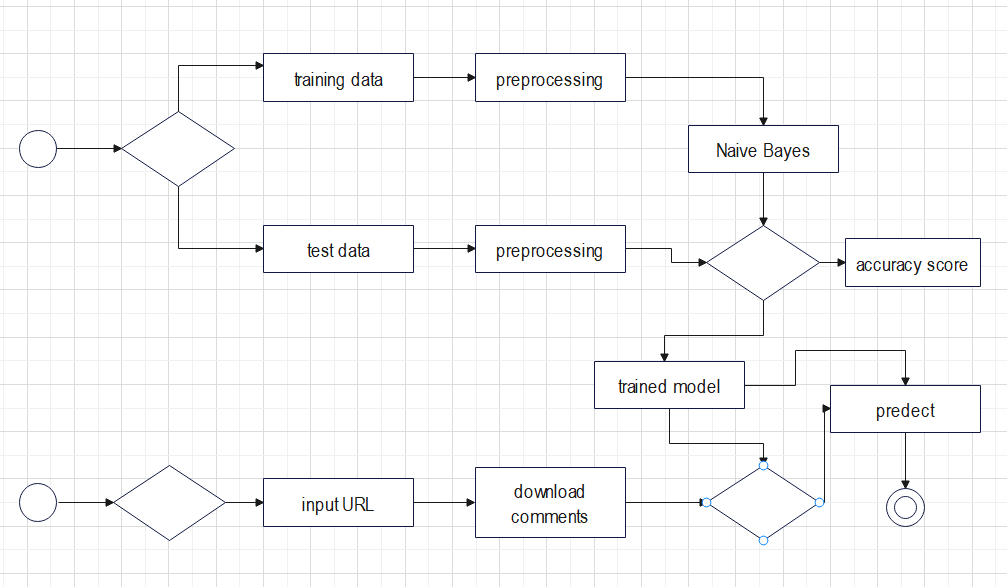


Figure 4.1.1: Revised Activity Diagram

The above activity diagram shows the detail control flow from the start to the finished point while the activity being executed.

### **4.1.2 Component Diagram**

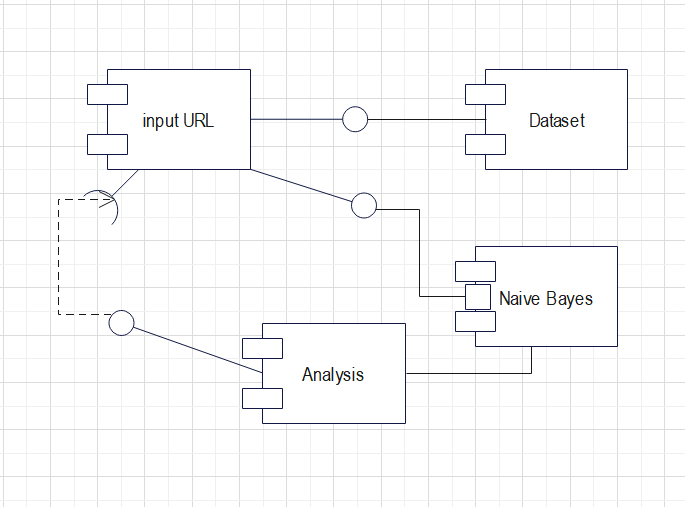


Figure 4.1.4: Component diagram

A component diagram is a collection of vertices and arcs and commonly contains components, interfaces, dependencies, aggregation etc. it may also contain note and constraints.

### **4.1.3 Deployment Diagram**

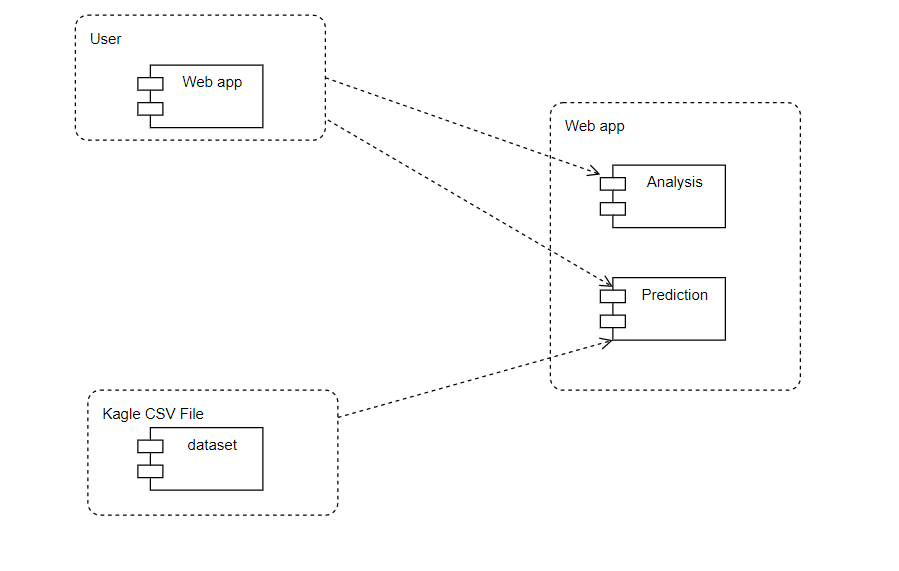


Figure 4.1.5: Deployment Diagram

A deployment diagram is used to visualize topology of a physical component of a system where the software components of the system are deployed it is also usefull to show the execution architecture of a system.

## **4.2 Algorithm Details**

### **4.2.1 Naive Bayes Algorithm**

For this project, Naive Bayes classifier have been used. This classifier works on the principles of Bayes Theorem. Naive Bayes is a supervised machine learning algorithm based on Bayes’ theorem. Bayes' theorem is defined mathematically as the following equation:

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

* A is the polarity of statement depending on what probability we are calculating. (i.e., positive or negative)
* B is the statement for which we are calculating the probability of it being positive or negative.
* A and B are independent events (i.e., the probability of the outcome of event A does not depend on the probability of the outcome of event B).
* P is the probability.
* P(A|B) represents the probability of event A happening given that B is true.
* P(B|A) represents the probability of event B happening given that A is true.
* P(A) and P(B) are the probabilities of observing A and B without any prior conditions. These are referred to as prior probabilities.

For sentiment analysis we use formula derrrived from above Naive Bayes theorem ,After tokanization of sentence we get words represented as wi .

P(A|B) = P(A) \* Π P( wi | A) …..1

where i belongs to positions of tokanized words

**TRAINING THE MODEL**

**1. Calculate the Prior Probability:** We will first find the number of documents belonging to each class . Finding the percentage of the documents in each class will give us the required prior probability.

Let’s assume the number of documents in class Positive is Np.

Total number of documents is assumed to be total.

So , P(Positive) = Np / total which is similar for Negative class.

1. **Calculate the Likelihood Probability:** Our main goal is to find the fraction of times the word wi appears among all words in all documents of class Positive. We first concatenate all documents with category Positive and Negative into one big text ‘bag’ which is Bag of words. We will however face a very unique problem at this point . If the word doesn’t belong to the given class then overall probability will be zero. To combat this problem we will introduce an add-on , Laplace Smoothing Coefficient , to both the numerator and the denominator . Our equation will be modified as follows:

P( wi | Positive) = (count( wi | Positive) + a) / (bag(Positive) + length(bag) \* a)

P( wi | Negative) = (count( wi | Negative) + a) / (bag(Negative) + length(bag) \* a)

* P( wi | Positive) is word being positive
* P( wi | Negative) is word being negative
* count( wi | Positive) count of word wi in bag which are positive
* count( wi | Negative) count of word wi in bag which are negative
* bag(Positive) is length of positive word in bag
* bag(Negative) is length of negative word in bag
* length(bag) is length of bag
* a is the Laplace smoothing coefficient. We usually consider its value to be 1.

After calculating all the individual word probabilities and applying Laplace smoothing to each of them, we finally create a bag of words with each words negative and positive probability. Now for calculating overall probability of a statement we tokanize it first and use equation Naive Bayes theorem using our bag of words. After that, all we do is compare which class has the greatest probability; this class is the output of the classifier.

The input to this algorithm is the statement ,bag of words and prior probabilities of each class and it classifies the sentiment of sentence as output

**ALGORITHM STEPS:**

* STEP 1: START
* STEP 2: Calculate prior probability of each class label (positive or negative).
* STEP 3: Calculate conditional probability with each feature for each label.
* STEP 4: Multiply the conditional probability for each label.
* STEP 5: Multiply the label probability with STEP 4.
* STEP 6: The label with highest probability is taken as sentiment.
* STEP 7: STOP

**Algorithm that implements Naive Bayes is:**

def naive\_bayes\_predict(X, bag, prior\_pos, prior\_neg):

Y = []

for i in range(len(X)):

k\_pos = 1

k\_neg = 1

p = sentence\_to\_words(X[i])

for k in range(len(bag)):

for word in p:

if word == bag['index'][k]:

k\_pos = k\_pos \* bag['sent=positive'][k] #product of likelihood prob given in vocabulary

k\_neg = k\_neg \* bag['sent=negative'][k]

nb = [prior\_neg \* k\_neg, prior\_pos \* k\_pos] # multiply each likelihood prob with the prior prob

Y.append(np.argmax(nb))

return Y

def naive\_bayes\_train(X, Y, a=0.000001):#a is laplace smoothing to avoid overall prob being zero

n\_length = len(X)

n\_class\_pos = len(Y[Y == 1])

n\_class\_neg = len(Y[Y == 0])

prior\_pos = n\_class\_pos / n\_length # prior probability for class

prior\_neg = n\_class\_neg / n\_length #prior probability for class

(n, p, bag) = bag\_of\_words\_maker(X, Y)

pr = {}

for i in range(len(bag)): #evaluating the likelihood prob for each word given a class

p\_pos = (bag['count\_pos'][i] + a) / (p + len(bag) \* a)

p\_neg = (bag['count\_neg'][i] + a) / (n + len(bag) \* a)

pr[bag['index'][i]] = [p\_pos, p\_neg]

pr = pd.DataFrame(pr).T

pr.columns = ['sent=positive', 'sent=negative']

pr = pr.reset\_index()

return (prior\_pos, prior\_neg, pr)

**Bag of words maker:**

def bag\_of\_words\_maker(X, Y):

bag\_dict\_binary\_NB\_pos = {} #keeping track of the positive class words

bag\_dict\_binary\_NB\_neg = {} #keeping track of the negative class words

for i in range(len(X)):

p = sentence\_to\_words(X[i])

sent = Y[i]

x\_pos = {}

x\_neg = {} #we intialize the dict every iteration so that it does not consider repititions .(Binary NB)

if sent == 1:

for word in p:

if word in x\_pos.keys():

x\_pos[word] = [x\_pos[word][0] + 1, x\_pos[word][1]]

else:

x\_pos[word] = [1, sent]

for key in x\_pos.keys():

if key in bag\_dict\_binary\_NB\_pos.keys():

bag\_dict\_binary\_NB\_pos[key] = \

[bag\_dict\_binary\_NB\_pos[key][0] + 1,

bag\_dict\_binary\_NB\_pos[key][1]]

else:

bag\_dict\_binary\_NB\_pos[key] = [1, sent] #storing it in the final dict

if sent == 0:

for word in p:

if word in x\_neg.keys():

x\_neg[word] = [x\_neg[word][0] + 1, x\_neg[word][1]]

else:

x\_neg[word] = [1, sent]

for key in x\_neg.keys():

if key in bag\_dict\_binary\_NB\_neg.keys():

bag\_dict\_binary\_NB\_neg[key] = \

[bag\_dict\_binary\_NB\_neg[key][0] + 1,

bag\_dict\_binary\_NB\_neg[key][1]]

else:

bag\_dict\_binary\_NB\_neg[key] = [1, sent]

# print(bag\_dict\_multi.keys())

# returns the dataframe containg word count in each sentiment

neg\_bag = pd.DataFrame(bag\_dict\_binary\_NB\_neg).T

pos\_bag = pd.DataFrame(bag\_dict\_binary\_NB\_pos).T

neg\_bag.columns = ['count\_neg', 'sentiment\_neg']

pos\_bag.columns = ['count\_pos', 'sentiment\_pos']

neg\_bag = neg\_bag.reset\_index()

pos\_bag = pos\_bag.reset\_index()

n = len(neg\_bag)

p = len(pos\_bag)

bag\_of\_words = pd.merge(neg\_bag, pos\_bag, on=['index'], how='outer')

bag\_of\_words['count\_neg'] = bag\_of\_words['count\_neg'].fillna(0)

bag\_of\_words['count\_pos'] = bag\_of\_words['count\_pos'].fillna(0)

bag\_of\_words['sentiment\_neg'] = bag\_of\_words['sentiment\_neg'].fillna(0)

bag\_of\_words['sentiment\_pos'] = bag\_of\_words['sentiment\_pos'].fillna(1)

return (n, p, bag\_of\_words)

# **CHAPTER 5**

# **IMPLEMENTATION AND TESTING**

## 

## **5.1 Implementation**

System implementation is mainly concerned with the building of a properly working system. The classification of the youtube comments is performed using the Naive Bayes algorithm implemented using ML.

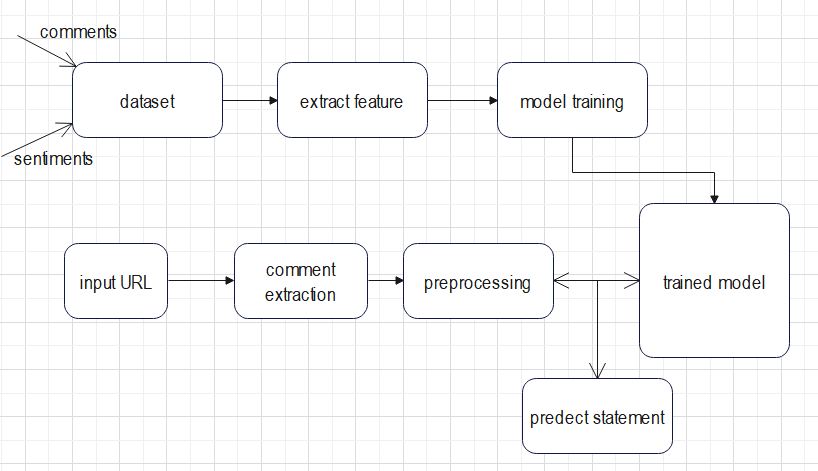


Figure 5.1: Implementation of Sentiment Analysis System

## **5.1.1 Tools Used**

### **5.1.1.1 Flask**

Flask is used to design the interface of the system to make the interface attractive. It is easy, fast and we could use pickled model so flask was used.

### **5.1.1.2 Python**

Python is used for building a model that can be used to classify and recommend different music based on the genre for the users. Machine learning algorithms are implemented in Jupyter Notebooks such as Naive Bayes.

### **5.1.1.3 Visual Studio Code**

It is a most used IDE in any Operating System including Windows, Linux and MacOS to code any programming language that has extensions to execute any program. Further, it has debugging, snippet and Git push and clone benefit.

### **5.1.1.4 MS Word**

In this project we have used MS word for the documentation of report.

### **5.1.1.5 Wondershare Edrawmax**

In this project we have used this application to draw the flowchart, DFD, ER diagram and other kind of structure.

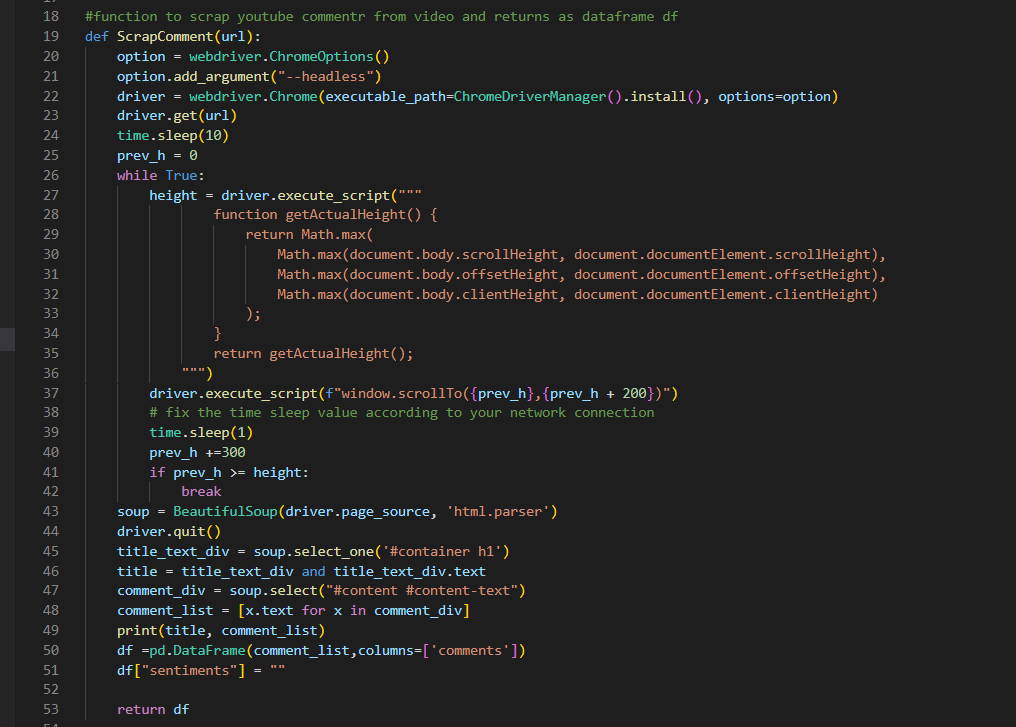
**5.1.1.6 Html and CSS**

HTML and CSS have been used to design the user interface of the system.

**5.1.2 Implementation Detail of Modules:**

**5.2.1.1 Comment extraction module**

In comment extraction we simply extract the comments. Beautifulsoup 4, webdriver manager and selenium were used for comment extraction.



**5.2.1.2 Preprocessing module**

In preprocessing modules, there is a lot of cleaning and preprocessing that needs to be done before we get into feature extraction part. In short in this module, we have to lowercase the comments, remove stop words, punctuation marks, remove URLs and unwanted characters from the tweet and lemmatization of words.

#importing string,nltk and re for text preprocessing

import string

from nltk import pos\_tag

from nltk.corpus import wordnet

from nltk.stem import WordNetLemmatizer

import re

#function that processes the comments and returns clean\_comments as df

def text\_preprocessing(data):

    #convert to lowercase

    data['clean\_comments'] = data['comments'].str.lower()

    #removing empty columns

    data['clean\_comments'].replace('', np.nan, inplace=True)

    data.dropna(subset=['clean\_comments'], inplace=True)

    #removing urls and html tags

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: re.sub(r'https?://\S+www\.\S+', ' ', x))

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: re.sub(r'<.\*?>', ' ', x))

    #removing punctuations

    punctuation = string.punctuation

    data['clean\_comments'] = data['clean\_comments'].astype(str)

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: x.translate(str.maketrans('','',punctuation)))

    #removing stopwords

    STOPWORDS = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',

       "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',

       'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',

       'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',

       'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom',

       'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are',

       'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',

       'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and',

       'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at',

       'by', 'for', 'with', 'about', 'against', 'between', 'into',

       'through', 'during', 'before', 'after', 'above', 'below', 'to',

       'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',

       'again', 'further', 'then', 'once', 'here', 'there', 'when',

       'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',

       'most', 'other', 'some', 'such', 'nor', 'only', 'own', 'same',

       'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',

       'don', 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',

       've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'doesn',

       "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't",

       'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't",

       'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",

       'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',

       "wouldn't",'another','other']

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: " ".join([w for w in x.split() if w not in STOPWORDS]))

    #removing special characters

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: re.sub('[^a-zA-Z0-9]', ' ', x))

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: re.sub('\s+', ' ', x))

    #stemization

    #ps = PorterStemmer()

    #data['clean\_comments'] = data['clean\_comments'].apply(lambda x: " ".join([ps.stem(w) for w in x.split()]))

    #lemmatization and pos tagging

    lemmatizer = WordNetLemmatizer()

    wordnet\_map = {"N":wordnet.NOUN,"V":wordnet.VERB,"J":wordnet.ADJ,"R":wordnet.ADV}

    data['clean\_comments'] = data['clean\_comments'].apply(lambda x: " ".join([lemmatizer.lemmatize(w,wordnet\_map.get(pos[0],wordnet.NOUN)) for w, pos in pos\_tag(x.split())]))

    #removing empty columns

    data['clean\_comments'].replace('', np.nan, inplace=True)

    data.dropna(subset=['clean\_comments'], inplace=True)

    return data

### **5.2.1.3 Classifying module**

The classifying module is needed to classify the given text into the respective sentiment. User input a text along with its extracted features filling up the form. The module then predicts the sentiment of the text and displays it to the user.

def naive\_bayes\_predict(X, bag, prior\_pos, prior\_neg):

    Y = []

    for i in range(len(X)):

        k\_pos = 1

        k\_neg = 1

        p = sentence\_to\_words(X[i])

        for k in range(len(bag)):

            for word in p:

                if word == bag['index'][k]:

                    k\_pos = k\_pos \* bag['sent=positive'][k] #product of likelihood prob given the word is present in vocabulary

                    k\_neg = k\_neg \* bag['sent=negative'][k]

        nb = [prior\_neg \* k\_neg, prior\_pos \* k\_pos] # multiply each likelihood prob with the prior prob

        Y.append(np.argmax(nb))

    return Y

## 

## **5.2 Testing**

The purpose of testing is to detect software failures so that the defects may be discovered and corrected accordingly. Testing is the process of execution to find any bugs or errors in software. Properly tested software product ensures reliability, security, and high performance.

We have performed various types of testing to check the overall functionality of the proposed system by analyzing different modules to achieve proper accuracy and performance. The various types are explained below:

## **Unit Testing**

Each module of the system was checked for correctness to meet the expected result. Both the user registration module and the classifying module were tested separately so that they work correctly in generating the desired output to the system.

## **b. Integration Testing**

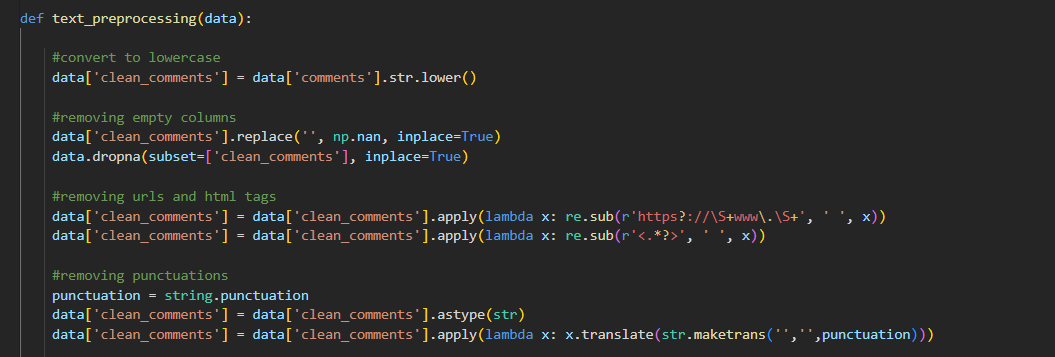
The testing was performed to check the reliability and stability of the whole web application after we integrate the modules. During testing the algorithm implementation of the Naive Bayes algorithm in python and the web application built in flask was integrated to find out if they work properly or not together.

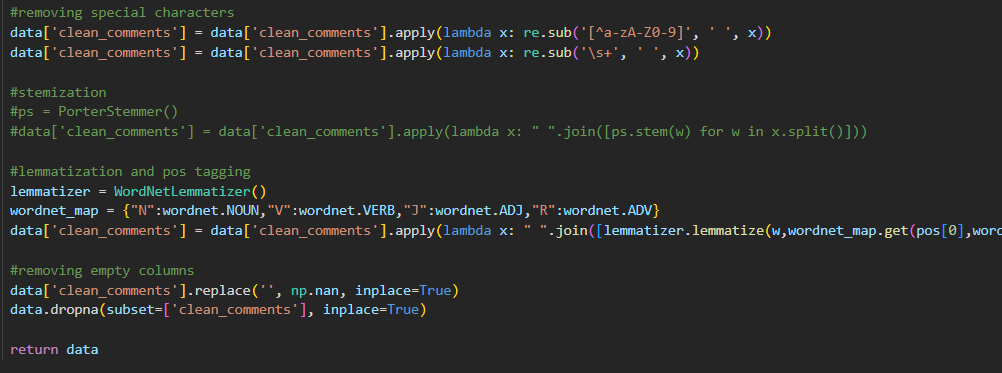
## **Test Cases**

Various test cases were designed to test the overall functioning of various functional units of the system and they are mentioned below.

## **5.2.1 Test cases for Unit Testing**

For the unit testing we have tested each module of the code like preprocessing, training, and classification through Naive Bayes classifier





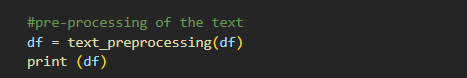
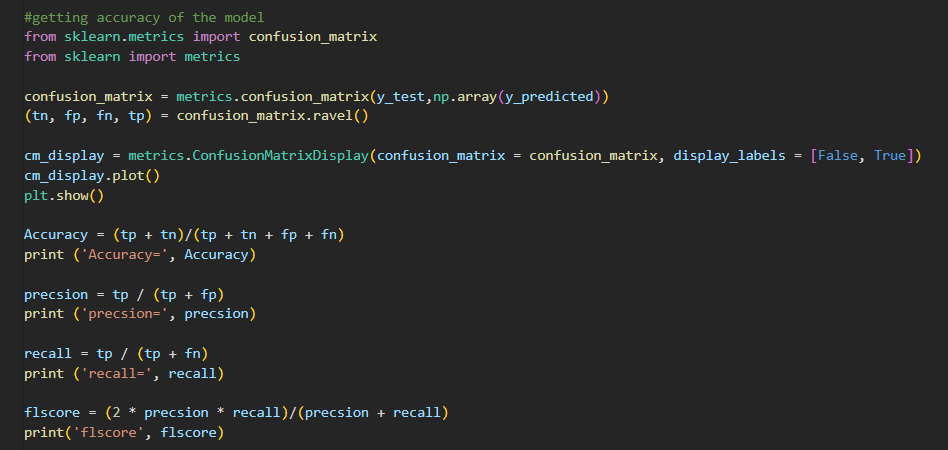


Fig 5.4.1: Preprocessing testing



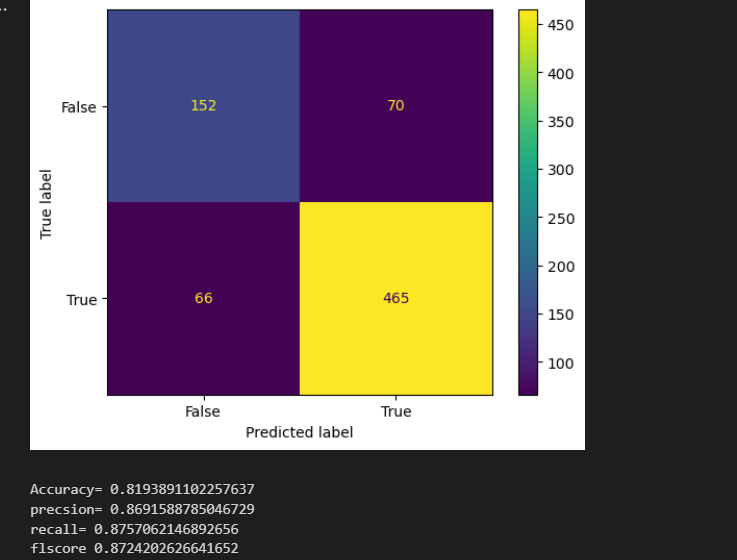


Figure 5.4.1: Naive Bayes Classifier Testing

## **5.2.2 Test cases for Integration Testing**

## 

## 

## The system generates the result as below on providing the input to URL field.



## **5.3 Result Analysis**

The result is analyzed based on the attributes of the text such as words like love, good, hate, bad etc. These are the highly corelated attributes that defines the genre of a specific text. The Naive Bayes algorithm takes the attribute and classify the music into its respective genres with an accuracy of around 81.93% as predected by sklearn confusion matrix.

# **CHAPTER 6**

# **CONCLUSION AND FUTURE RECOMMENDATION**

## **6.1 Conclusion**

Hence, we have developed an application named as “Youtube Sentiment Analysis System”. The project fulfills its objective and is able to classify the given text into respective sentiment with an acceptable accuracy. Large number of Youtube dataset are preprocessed using different kind of module and processed it with Naive Bayes algorithm.

## **6.2 Future Recommendation**

Naive Bayes algorithm was used to classify the text provided that required parameters were given already. It is always an overhead to calculate the required parameters or attribute manually before classifying the text. As large number of texts should be classified on a daily basis and such process is not a great choice. Also, only two parameters are considered to classify the given text into respective sentiment which is not sufficient enough and comment can be also considered as neutral. The performance of the algorithm can be improved by taking more parameters and improving data preprocessing techniques like cross fold validation to identify specific attributes.