PUSAT SAINS MATEMATIK
قوستا ساءيناس ماتيماتيک

BSD3523:
MACHINE LEARNING

GROUP PROJECT REPORT
YOUTUBE COMMENTS ANALYSIS

BRANDYT

Authored by: NIK NUR AIN BINTI NIK JID (SD20022)
CONNIE LEE WAI YAN (SD20016)
YAP KAH HUI (SD20018)
SITI NOR BALQIS BINTI JAFAAR (SD20043)

TABLE OF CONTENT

1.0 INTRODUCTION

1.1 BACKGROUND OF COMPANY

Figure 1.1 Company Logo

BrandYT is a software and startup company that was established in November 2022 and founded by 4 people that specialize in data analytics and has extensive experience in the field. BrandYT is located and based in the industrialized area of Kuala Lumpur. BrandYT is a social media monitoring and analytics company that helps businesses understand their online presence and track the conversations around their brand. BrandYT also provides sentiment analysis of social media such as Twitter, Instagram, YouTube, etc to identify key influencers and track performance metrics such as reach and engagement. Because BrandYT is only a startup company, the founders have decided to focus and polish their skills in the field of YouTube content analysis. Hence, the main service provided by BrandYT is to analyze YouTube content and comments. In the sentiment analysis, BrandYT can determine whether the comments of the YouTube video are positive, negative, or neutral, to provide insight into the overall sentiment around the video. The data collected by BrandYT can be exported in various formats, including CSV, Excel, and PDF that enables the clients to share and view the data with their teams.

1.2 FOUNDERS

The company assigns responsibilities that each person must accomplish in order to keep the company running and to produce profits. Below are the organization chart of BrandYT:

NIK
CEO / DATA ANALYST

CONNIE
COO / ML ENGINEER

BALQIS
SOFTWARE DEVELOPER

YAP
DATA ANALYST

2.0 PROJECT DESCRIPTION

2.1 BACKGROUND OF THE PROJECT

In this project, we will collect data from public YouTube comments and measure the user's attitudes toward aspects of a video that they describe in a text using Taylor Swift songs. Sentiment analysis is useful for quickly gaining the overall idea by analysing a large amount of text data, and it will aid in understanding the user's opinion of her songs. Sentimental analysis, also known as opinion mining, is the process of determining or identifying the positive, negative, and neutral opinions, views, attitudes, impressions, emotions, and feelings expressed in a text.

According to current YouTube usage statistics (2023, January 10), there are more than 70 thousand unique users viewing Taylor Swift's song content at the time of this study. YouTube offers numerous social mechanisms for gauging user opinion and views on a few Taylor Swift videos, such as voting, rating, favourites, sharing, and negative comments, among others. It's worth noting that YouTube offers more than just video sharing. In addition to uploading and viewing videos, users can subscribe to video channels and interact with other users via comments.

YouTube is primarily made up of implicit and explicit user-to-user interaction. The YouTube social network's user-to-user social aspect has been cited as a key differentiating factor when compared to other traditional content providers. The analysis of "unstructured" data contained in natural language text using various methods, machine learning tools, and techniques is known as text analytics. Text analysis is a low-cost method for gauging public opinion.

We created a system that is based on machine learning algorithms. The classification algorithms' accuracy has been evaluated using various evaluation measures such as F-Score and Accuracy score to determine the classification system's correctness. To improve the performance of our system, we used various feature selection techniques such as stemming, tokenization, stop words, emoji removal, and punctuation removal.

2.2 PROBLEM STATEMENT

Anti-Hero by Taylor-Swift was a depression song inspired by Taylor's nightmares and self-hatred issues. Taylor said that this song examines her mental insecurities in depth, detailing the things she hates about herself, and her struggle with 'not feeling like a person'. Taylor wants to show to her audience that she is far from perfect.

The song that has been released on October 21, 2022 has been getting so many criticisms by the audience and one of the scenes in the music video has led to Taylor being accused of fatphobia. Fatphobia is a form of discrimination and prejudice against individuals who are considered to be overweight or obese.

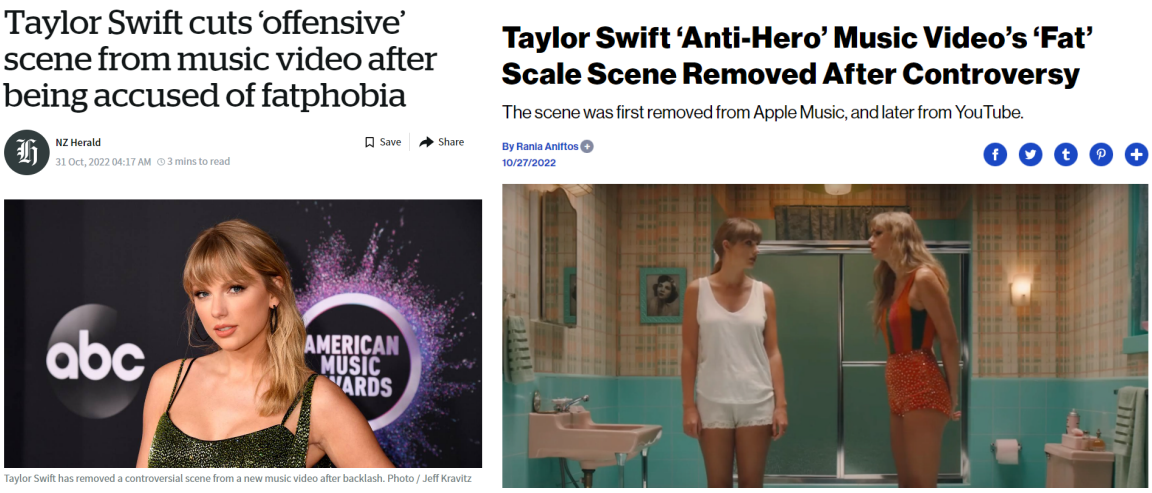


Figure 2.1 Controversy News of Taylor Swift's Anti-Hero

Figure 2.2 Deleted Scene

After being accused of fatphobia by several health experts, Taylor Swift has edited out the scene that led to controversy in the Anti-Hero music video and released the song again as shown in Figure 2.1 and Figure 2.2 above.

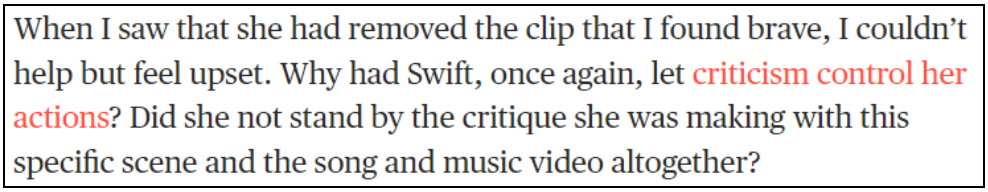


Figure 2.3 Upset Fans

Source: Georgi, M. (2022). NBC News

However, even after the scene has been deleted, there are some people that weren't satisfied with Taylor's action by deleting the scene. Taylor has been questioned why she has 'once again, let criticism control her actions' and why 'did she not stand by the critique she was making with this scene'. Hence, to see the audience's satisfaction towards the edited music video, this study has been conducted to see what kind of sentiment in the comments section of the video.

2.3 OBJECTIVES

The aim of this study is youtube comments analysis on Taylor-Swift song which is Anti-Hero. To achieve this goal, three (3) objectives need to be accomplished as follows:

1. To perform the sentiment analysis on YouTube comments for Taylor Swift's Anti-Hero music video.
2. To analyze which sentiment's category are the most in the comments section of Taylor Swift's Anti-Hero to spot their trends and seasonality.
3. To see the best machine learning models for the comment's sentiment prediction.

2.4 SCOPE

The focus of this project is to analyze the comment section for Taylor Swift's music video which is Anti-Hero. All the comments about this video from YouTube will be scrapped to use as a dataset for this project. The process of getting the dataset will start from the YouTube API service using the coding that has been provided by YouTube. The dataset will process using the sentiment analysis method through python.

2.5 LIMITATION

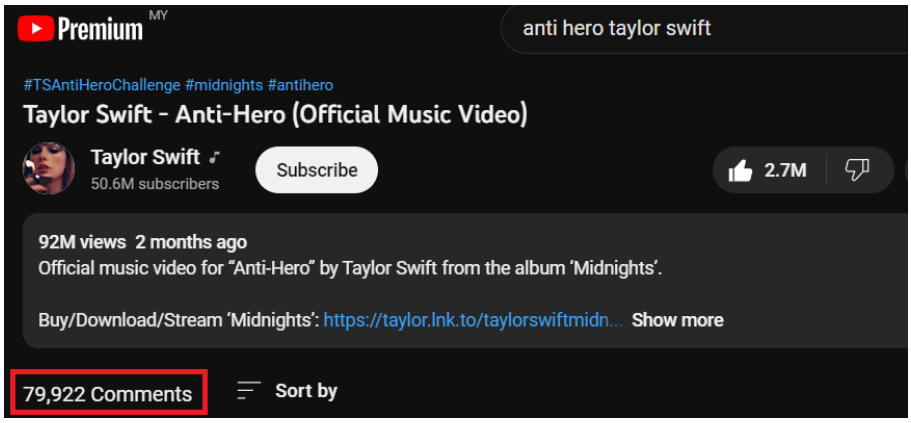
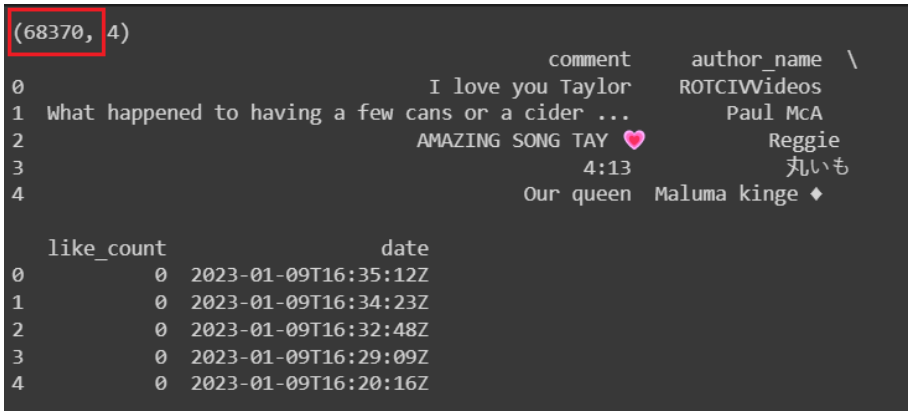


Figure 2.4 Total Comments on YouTube



```
(68370, 4)
                                                comment     author_name  \
0                                     I love you Taylor    ROTCIVVideos
1  What happened to having a few cans or a cider ...         Paul McA
2                                    AMAZING SONG TAY 💗          Reggie
3                                                  4:13           丸いも
4                                           Our queen  Maluma kinge ♦

   like_count                 date
0           0  2023-01-09T16:35:12Z
1           0  2023-01-09T16:34:23Z
2           0  2023-01-09T16:32:48Z
3           0  2023-01-09T16:29:09Z
4           0  2023-01-09T16:20:16Z
```

Figure 2.5 Total Comments Scrapped

The limitation of this project comes from the scope of the project. As we can see from Figure 2.4 and Figure 2.5 above, the total comments on YouTube at the time comments were scrapped are 79,922. Meanwhile, the total number of comments that have been successfully scrapped to make as the dataset is 68,370 only. Not all comments on YouTube can be scrapped as a dataset. This is because 79,922 comments on YouTube are included the replies for some of the comments as well.

The dataset has been limited to scrap the comments without a reply. This is because the comments and the replies fall under two different attributes. It will be displayed in the dataset under two different columns. This situation will create too many null data in the dataset as not all the comments have reply. Because the data type for the comments is a string, the imputation of mean, mode, and median cannot be used to treat the null values. The only way to counter this problem is to drop the rows that have null values. This will make the dataset will left with only the comments that have a reply which is only 11,552 comments (79,922 - 68370). With a

minority number of comments that have been scrapped, the pattern and analysis for the comments will be limited. Hence, for this project, only the comments without a reply will be selected.

There is a limitation in training the models. The computers of our team members have 8GB of memory, which is not enough to handle text classification of such a huge dataset. It occurs when convert the comment attribute to vector. Jupyter crashes while fetching features due to insufficient RAM. For avoid this issues, the maximum features is defined as 1500. This may lead to undiscovered potential features.

```
[73] from sklearn.feature_extraction.text import CountVectorizer
     cv = CountVectorizer(max_features=1500)
     X = cv.fit_transform(comment).toarray()
     y = final_data["Sentiment"].values
```

Figure 2.6 Maximum features

3.0 PROJECT PIPELINE

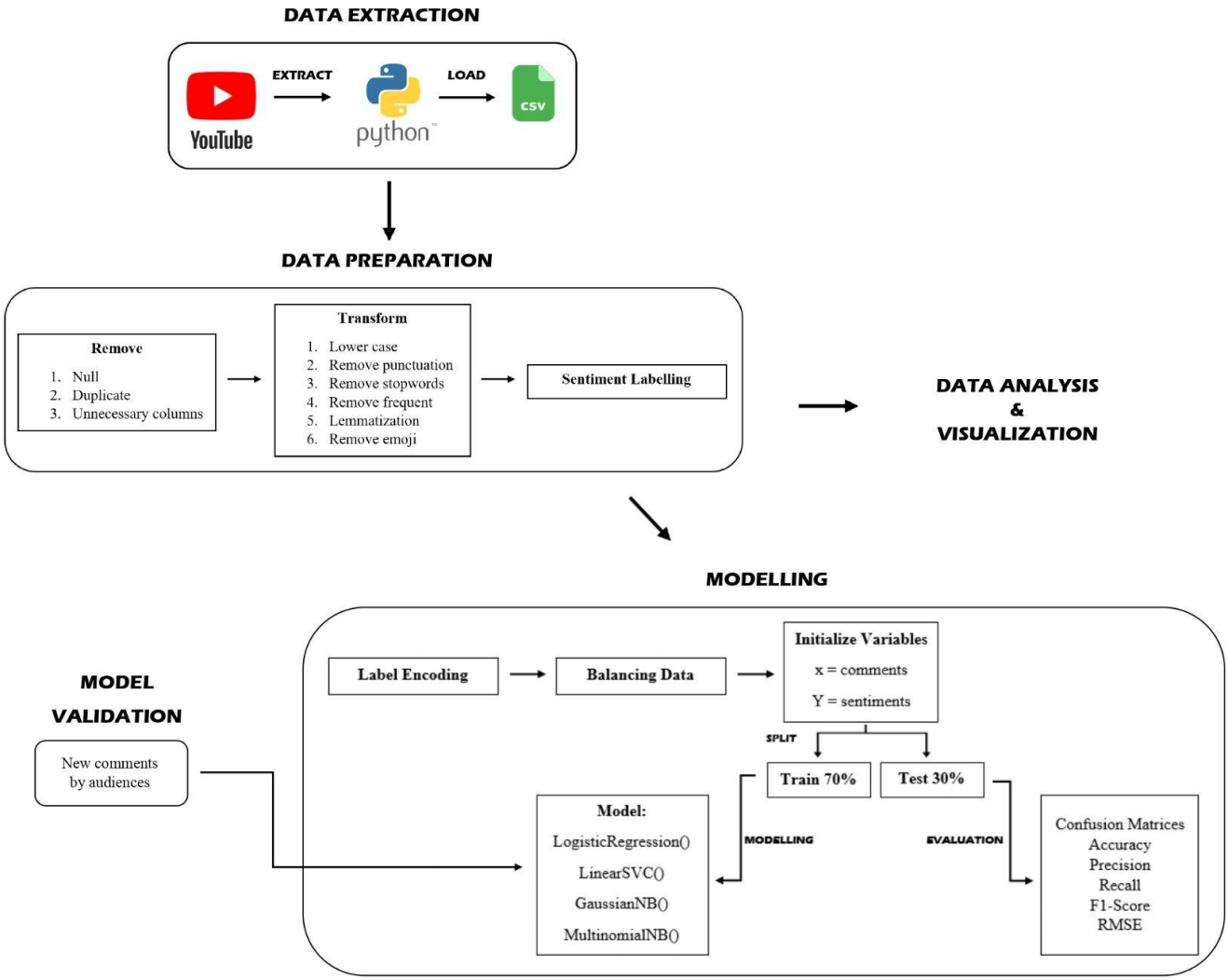


Figure above shows the overview of the pipeline for this project.

4.0 DATASET

4.1 DATASET OVERVIEW

4.1.1 DATA DESCRIPTION

The dataset that have been used for this project is the comments for the Taylor Swift's music video which is Anti-Hero. This data consists of one table only which is the scrapped comments. There are five columns in this dataset as shown in table 4.1.1 below:

Table 4.1.1 Dataset Table

Variable	Data Type	Description
comment	String	The comments of the video.
author_name	String	Name of the person that write the comment.
like_count	Integer	Number of like for that comment.
date	String	Time and date when the comment was written.
just_date	String	Time and date when the comment was scrapped.

4.1.2 DATA SCHEMA

```
anti = pd.read_csv('Antihero Comments.csv')
anti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68370 entries, 0 to 68369
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   comment      68340 non-null  object
 1   author_name  68366 non-null  object
 2   like_count   68370 non-null  int64 
 3   date         68370 non-null  object
 4   just_date    68370 non-null  object
dtypes: int64(1), object(4)
memory usage: 2.6+ MB
```

Figure 4.1.1 Dataset Schema

Based on figure 4.1.1 above, the data schema for this dataset consists of 5 attributes or columns. The data type for this data frame are string and integer. Four columns have the string data type and one column have the integer data type. There exist null values in some of the columns.

4.2 DATA PREPARATION

As mentioned above, the dataset for this project was scrapped from video's comment on YouTube. Before scrapping the data, the most important thing to remember is we need to use only one Google account for all platforms we will use. This is because all the platforms will be linked to each other in this process. The platforms involved are YouTube, Google Developer Console, and YouTube Data API.

4.2.1 YOUTUBE

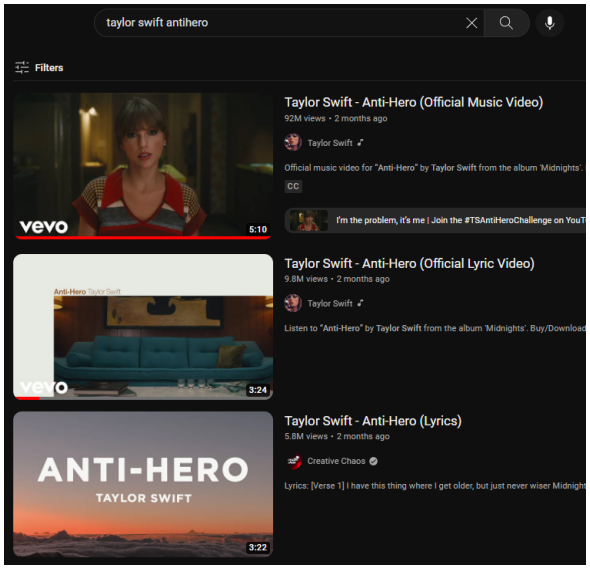


Figure 4.2.1 Anti-Hero's video

The first step before starting the process of getting the data, we need to identify which video we want to analyze. As shown in figure 4.2.1, when we search for 'taylor swift antihero', the results will show many videos that relate to that search word such as official music video and official lyric video by Taylor, lyrics video, remix video etc by Swifties. Hence, for this project, we choose an official music video by Taylor as the data source for the project. Next, after identify the data source video, we need to get the ID for the video from the video's link.

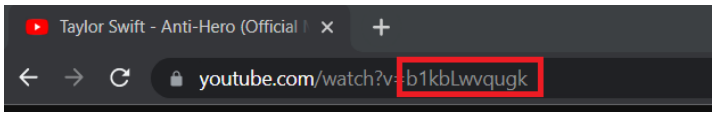


Figure 4.2.2 Video's ID

4.2.2 GOOGLE DEVELOPER CONSOLE

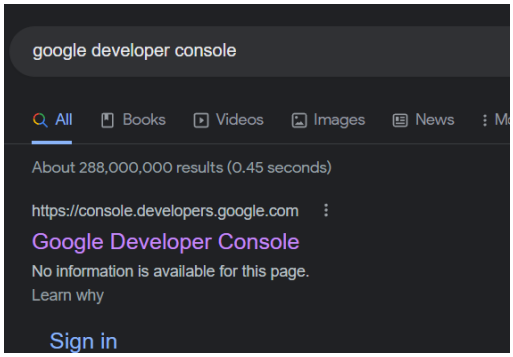


Figure 4.2.3 Google Developer

Search for the 'google developer console' and click the first link shown. As for the first time user, you need to sign up first with a Google account and agree with the terms of service before continue.

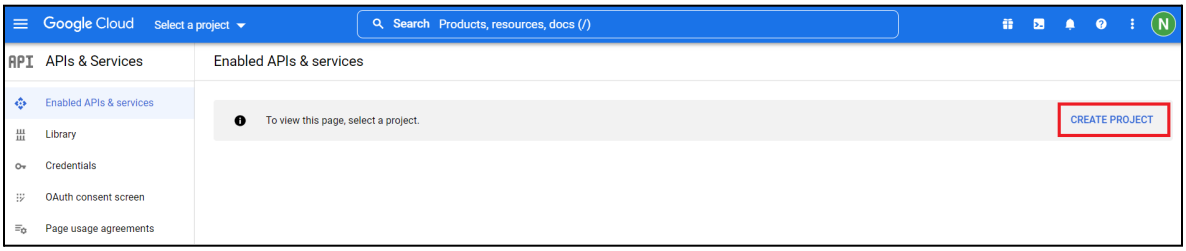


Figure 4.2.4 Google Developer homepage

The Google Developer homepage will be shown as figure 4.2.4 above. Click 'CREATE PROJECT' to create your project.

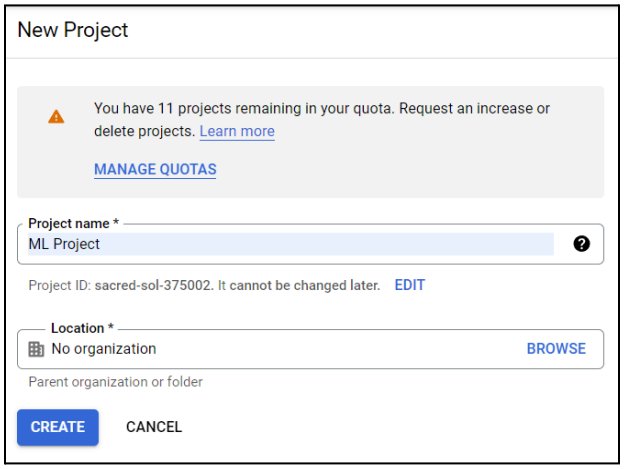


Figure 4.2.5 Create new project

Create a project name and click 'CREATE' button.

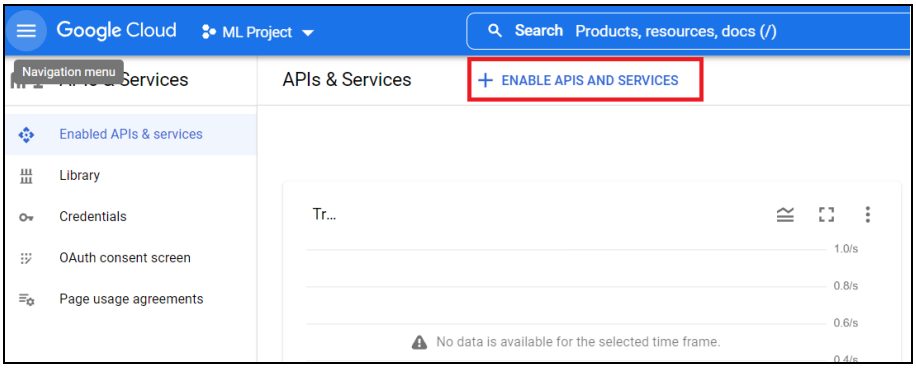


Figure 4.2.6 Enable API button

Click 'ENABLE APIS AND SERVICES' to open the API Libary.

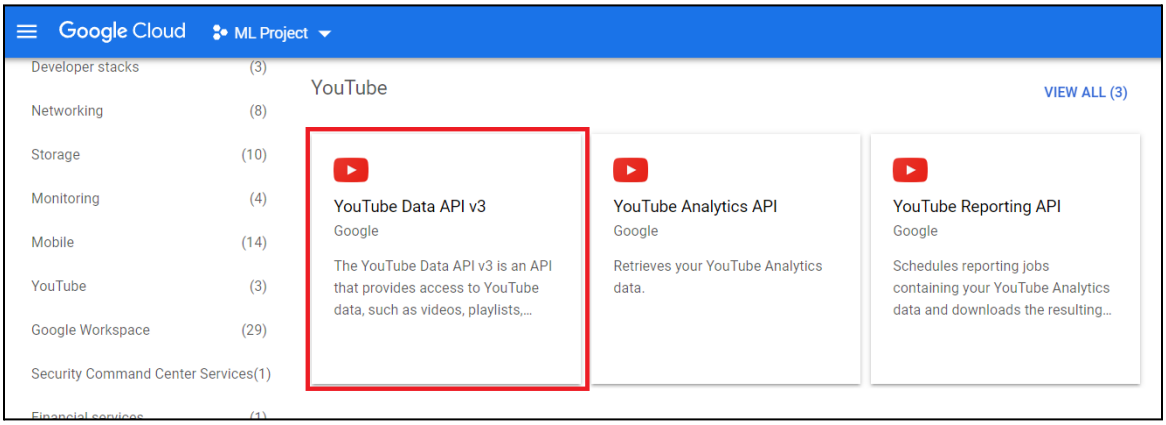


Figure 4.2.7 YouTube API list

Scroll down the API library until you find the YouTube API list. Choose 'YouTube Data API v3' because this is the API that will provide us access for the comments on the video.

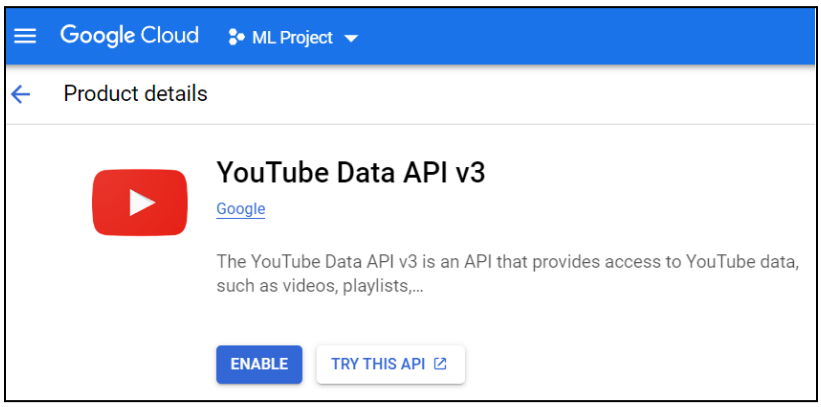


Figure 4.2.8 YouTube Data API v3

Click 'ENABLE' button to enable access of this API for the google account used.

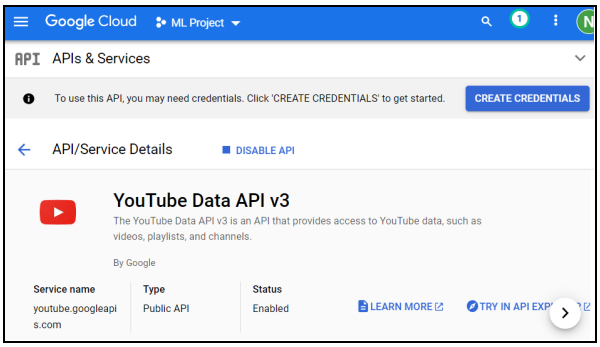


Figure 4.2.9 Credentials button

Click 'CREATE CREDENTIALS' button to allows us to get the credentials for this API.

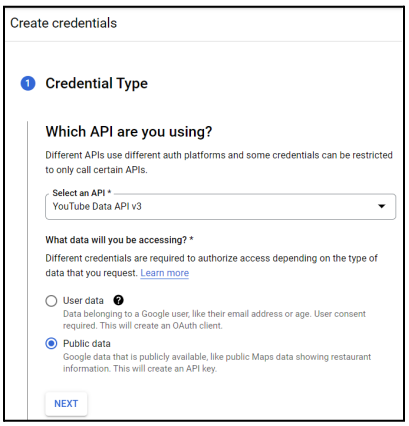


Figure 4.2.10 Credential type

Choose 'Public Data' for the credential type because the owner for the account of the video that we want to scrap is Taylor Swift not us. Hence, we need to get a developer key to access other user accounts.

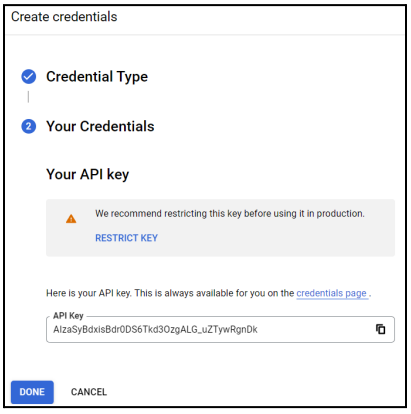


Figure 4.2.11 API key

After that, the API key will be shown. Save it to be use as the developer key in the coding later.

4.2.3 YOUTUBE DATA API

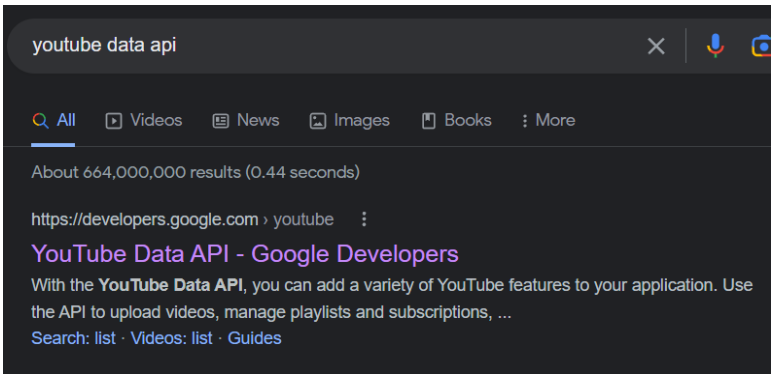


Figure 4.2.12 YouTube Data API

Search for the 'youtube data api' and click the first link shown. As for the first time user, you need to sign up first with the same Google account we used before.

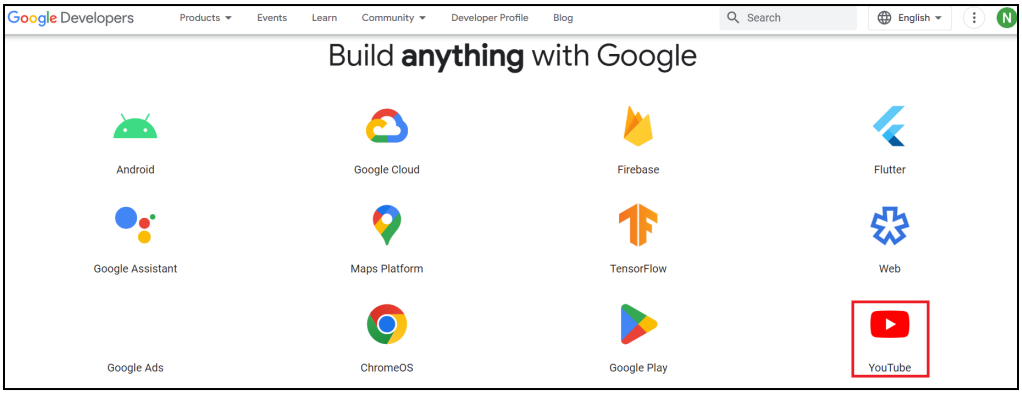


Figure 4.2.13 Developer product list

From the developer product list, click on the YouTube icon to proceed to the YouTube developer homepage.

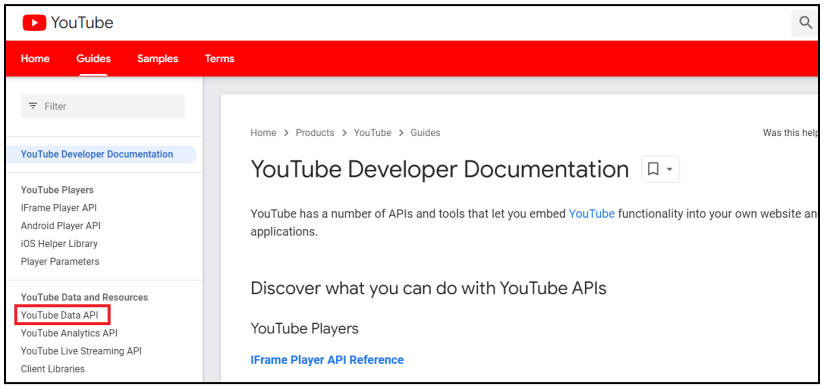


Figure 4.2.14 YouTube developer homepage

At the homepage, click YouTube Data API under Guides pane.

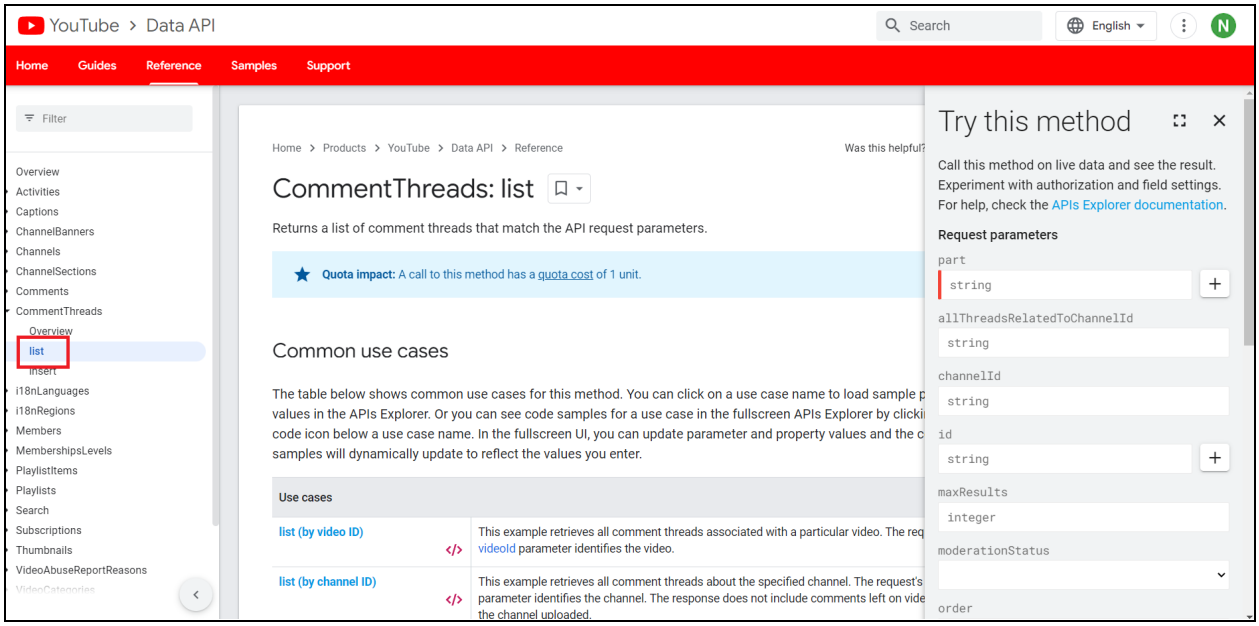


Figure 4.2.15 CommentThreads: list

After that, under the Reference pane, open the CommentThreads: list to get started with the extracting the comments.

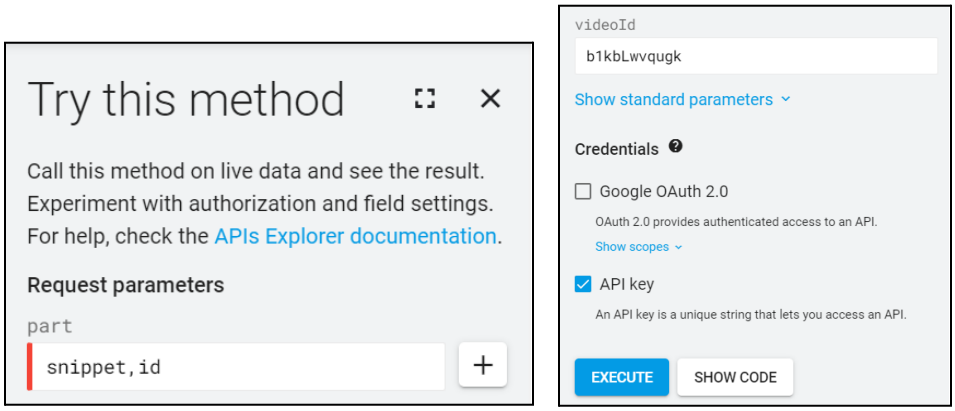


Figure 4.2.16 Part and videoId parameter

Under 'Try this method' section, put snippet and id as the parameter for the part. Function for snippet is to extract the comments and function for id is want to extract the author of the comment. After that, put the videoId of Taylor Swift Anti-Hero video under the videoId section. Choose API key as the credentials and click 'EXECUTE' button to print the output.

Figure 4.2.17 Extracted comments

The extracted comments will be shown with the 20 comments per page. Click the 'SHOW CODE' button to see the coding behind the output.

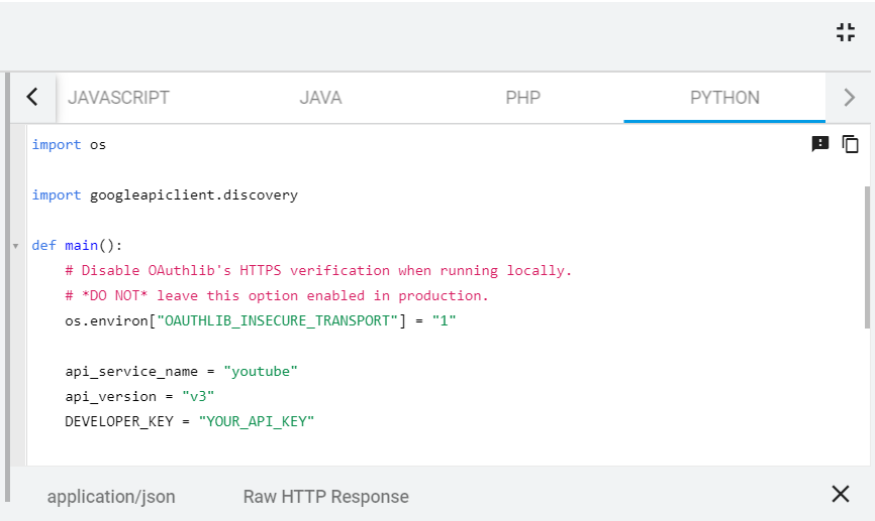


Figure 4.2.18 Coding of the output

The coding behind the output will be shown. Google developer has provided the coding in variety of programming language. For this project, we used python language to do the code. Hence, copy the coding provided in the python language and modified the code to extract the comments into csv file.

4.2.4 EXTRACT TO CSV

Copy the coding provided by Google developer and paste to the Jupyter Notebook to be modified.

```
#developer key
key = 'AIzaSyAe98RAU1Hp6t1cJg9lfX3HEKZsDPpaByA'
#video id (Taylor Swift - Anti-Hero)
videoId = 'b1kbLwvqugk'

#build youtube service
def build_service():
    YOUTUBE_API_SERVICE_NAME = "youtube"
    YOUTUBE_API_VERSION = "v3"
    return build(YOUTUBE_API_SERVICE_NAME,
                 YOUTUBE_API_VERSION,
                 developerKey=key)
```

Figure 4.2.19 Build service

Build the YouTube service function to call the YouTube Data API as the client. Define the API key, videoId, YouTube API service and version with the information that we have gather before.

```
#configure function parameters for required variables to pass to service
def get_comments(part='snippet',
                 maxResults=100,
                 textFormat='plainText',
                 order='time',
                 videoId=videoId,
                 csv_filename="antihero_comments"
                 ):

    #create empty lists to store desired information
    comments, authornames, likesCount, dates = [], [], [], []

    #build our service from path to apikey
    service = build_service()

    #make an API call using our service
    response = service.commentThreads().list(
        part=part,
        maxResults=maxResults,
        textFormat='plainText',
        order=order,
        videoId=videoId
    ).execute()

    while response: # this loop will continue to run until you max out your quota

        for item in response['items']:
            #index item for desired data features
            comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
            like_count = item['snippet']['topLevelComment']['snippet']['likeCount']
            authorname = item['snippet']['topLevelComment']['snippet']['authorDisplayName']
            date = item['snippet']['topLevelComment']['snippet']['publishedAt']

            #append to lists
            comments.append(comment)
            likesCount.append(like_count)
            authornames.append(authorname)
            dates.append(date)
```

Figure 4.2.20 Extract comments

Modified the get comments function with the information that we need. Create empty lists to store the information scrapped and call the API client to our Jupyter Notebook.

```
    #this loop to get the comments from next page
    try:
        if 'nextPageToken' in response:
            response = service.commentThreads().list(
                part=part,
                maxResults=maxResults,
                textFormat=textFormat,
                order=order,
                videoId=videoId,
                pageToken=response['nextPageToken']
            ).execute()
        else:
            break
    except: break
```

Figure 4.2.21 Next Page function

As we know, the Google developer only provided 20 comments per page.This means, the coding provided will print 20 comments only. Hence, we need to add some code to call all the comments from next page to be stored in the lists.

```
#convert dataframe to csv file
df.to_csv('Antihero Comments.csv', index=False)
```

Figure 4.2.22 Save to csv

As last step in the data preparation, all the comments need to be saved in the csv file to do the processing, analysis, and modelling easily.

4.3 PRE-PROCESSING

After the data has been collected and prepare, the dataset need to be understand before proceed with pre-processing of the dataset.

```
print('No of attributes: ', len(anti.columns))
print('No of rows: ', len(anti))

No of attributes:  5
No of rows:  68370

anti.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68370 entries, 0 to 68369
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   comment      68340 non-null  object
 1   author_name  68366 non-null  object
 2   like_count   68370 non-null  int64
 3   date         68370 non-null  object
 4   just_date    68370 non-null  object
dtypes: int64(1), object(4)
memory usage: 2.6+ MB
```

Figure 4.3.1 Dataset Overview

Based on the Figure 3.3.1 above, the dataset consist of 5 attributes and 68,370 rows of data. From the dataset info, we can see that some of the attributes has null values.

```
[5]  anti1=anti.drop(['just_date'],axis=1)
     anti1.head()
```

Figure 4.3.2 Unnecessary Column

The first step in the data pre-processing is to drop the unnecessary column in the dataset which is 'just_date' because this attribute is about time and date when the comments are scrapped. Hence, the attrribute doesn't have important information that can be analyse.

```
[6]  #checking null
     anti1.isnull().sum()

     comment        30
     author_name     4
     like_count      0
     date            0
     dtype: int64
```

Figure 4.3.3 Checking Null Values

Next, the null value in the dataset need to checked and as shown in the figure above, there exists null values in the comments and author_name.

```
[7]  anti2 = anti1.dropna()

[8]  #recheck null
     anti2.isnull().sum()

     comment        0
     author_name    0
     like_count     0
     date           0
     dtype: int64
```

Figure 4.3.4 Treat Null Values

The data type of attributes that contain null values is string, the imputation of mean, mode, median can not be made to treat this null because there is no statistical analysis of mean, mode, and median for this attributes, Hence, to overcome this problem, the rows that consists null values will be drop.

```
[9]  #checking duplicate
     anti2[anti2.duplicated(keep=False)]

[10] #remove theduplicateddataandkeepfirst
     anti2.drop_duplicates(keep='first', inplace=True)
```

Figure 4.3.5 Checking Duplicate Data

Next, the duplicated data need to be checked and remove to prevent the redundant in the dataset and to maintain the accuracy of the model.

```
[13] print('New no of attributes: ', len(anti2.columns))
     print('New no of rows: ', len(anti2))

     New no of attributes:  4
     New no of rows:  68331
```

Figure 4.3.6 New Dataset

The new dataset after has been checked and clean consists of 4 attributes and 68331 rows of data.

Data Transformation

Before dataset can proceed with the modelling, it need to be transformed first because the text data is unstructured and contains a lot of noise, such as punctuation, capitalization, and stop words. These need to be removed or normalized to make the data more consistent and easier to work with.

1. Lower Case

```
[28] anti2["comment_lower"] = anti2["comment"].str.lower()

[29] anti2[['comment','comment_lower']]
```

	comment	comment_lower
0	I love you Taylor	i love you taylor
1	What happened to having a few cans or a cider ...	what happened to having a few cans or a cider ...
2	AMAZING SONG TAY 💗	amazing song tay 💗

Figure 4.3.7 Lower Case Transformation

First step in data transformation is to change all capital letter in comment to small letter. Based on the figure 4.3.7 above, the process to change character from capital to small letter has been successfully done.

2. Remove Punctuation

```
[30] PUNCT_TO_REMOVE = string.punctuation
     def remove_punctuation(text):
         """custom function to remove the punctuation"""
         return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

     anti2["comment_wo_punct"] = anti2["comment_lower"].apply(lambda text: remove_punctuation(text))

[32] anti2[['comment_lower','comment_wo_punct']].loc[[8,11]]
```

	comment_lower	comment_wo_punct
8	anti hero? i dont get it. what this world need...	anti hero i dont get it what this world needs ...
11	genius ! taylor you outdid yourself in this one	genius taylor you outdid yourself in this one

Figure 4.3.8 Punctuation Removal

Next, we remove all the punctuation from the comments because the punctuation marks are not informative for natural language processing tasks. They can add noise to the data and make it harderr for models to identify patterns. Hence, we remove the punctuation from the comments in this dataset as example in figure 4.3.8, the symbols '?' in index 8 and '!' in index 11 has been successfully removed.

3. Remove Stopwords

```
[87] import nltk
     nltk.download('stopwords')
     from nltk.corpus import stopwords
     ", ".join(stopwords.words('english'))

     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
     'i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, you
     rselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, th
     eirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, bee
     n, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, wh
     ile, 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, ho
     w, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, v
     ery, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, c
     ouldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, m
     ightn, mightn't, mustn, must...'
```

Figure 4.3.9 List of stopword from nltk library

```
[34] STOPWORDS = set(stopwords.words('english'))
     def remove_stopwords(text):
         """custom function to remove the stopwords"""
         return " ".join([word for word in str(text).split() if word not in STOPWORDS])

     anti2["comment_wo_stop"] = anti2["comment_wo_punct"].apply(lambda text: remove_stopwords(text))

[35] anti2[['comment_wo_punct','comment_wo_stop']].loc[[666,257]]
```

	comment_wo_punct	comment_wo_stop
666	such a brilliant song a masterpiece ❣ ❣ ❣	brilliant song masterpiece ❣ ❣ ❣
257	this is really great i actually love this	really great actually love

Figure 4.3.10 Remove stopword

By calling all the stopwords from nltk library as shown in figure 4.3.9, we will remove stopwords like this from our comments. Based on figure 4.3.10 the words ‘such’ and ‘a’ from index 666 and words ‘this’, ‘is’, and ‘i’ from index 257 has been successfully removed.

4. Remove Frequent Words

```
[36] from collections import Counter
     cnt = Counter()
     for text in anti2["comment_wo_stop"].values:
         for word in text.split():
             cnt[word] += 1

     cnt.most_common(20)
```

```
[('taylor', 11957),
 ('love', 9562),
 ('song', 6059),
 ('kane', 4722),
 ('brown', 4265),
 ('music', 4230),
 ('like', 3623),
 ('swift', 3392),
 ('video', 3079),
 ('im', 2869),
 ('cant', 2297),
 ('never', 2171),
 ('always', 1778),
 ('much', 1740),
 ('tonedeaf', 1728),
 ('te', 1695),
 ('one', 1654),
 ('best', 1610),
 ('album', 1581),
 ('amazing', 1526)]
```

Figure 4.3.11 Top 20 Frequent Words in Dataset

```
[37] #remove some word
     FREQWORDS_to_be_delete = ['taylor','Taylor','kane','brown','swift','song','music','video','cant','shes','people','never','always','much','album']
     def remove_freqwords(text):
       """custom function to remove the frequent words"""
       return " ".join([word for word in str(text).split() if word not in FREQWORDS_to_be_delete])

     anti2["comment_noFreq"] = anti2["comment_wo_stop"].apply(lambda text: remove_freqwords(text))
```

Figure 4.3.12 Remove Frequent Words

We remove the frequent used words from this dataset to easily vusualized group of words in the words cloud for the analysis step.

5. Lemmatization

```
[38] nltk.download('wordnet')
     nltk.download('omw-1.4')
     from nltk.stem import WordNetLemmatizer

     lemmatizer = WordNetLemmatizer()
     def lemma_words(text):
         return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

     anti2["comment_lemma"] = anti2["comment_noFreq"].apply(lambda text: lemma_words(text))
```

Figure 4.3.13 Lemmatization

After that, we used lemmatization to reduced words to their base form. The choice of using lemaatization instead of stemming is because lemmatization using a dictionary-based approach that takes into account the context and the part of speech of the word in order to determine its base form. This can ensure that the resulting words are still meaningful. Meanwhile stemming just cutting off the end of a word to reduce it to its base form

```
[53] anti2[['comment_noFreq','comment_lemma','comment_stemmed']].loc[[86]]
```

	comment_noFreq	comment_lemma	comment_stemmed
86	please please support insanely talented beauti...	please please support insanely talented beauti...	pleas pleas support insan talent beauti angel ...

Figure 4.3.14 Lemmatization vs Stemming

For example in Figure 4.3.14 above, the word 'insanely' has be stemmed to 'insan' has lose the meaning of the word. Hence, considering the task of sentiment analysis on youtube comments, where the meaning of the text is crucial, lemmatization is a better choice than stemming as it helps to maintain the meaning of the text while reducing the words to their base from.

6. Removal of emojis

```
[54] import re

     anti2["comment_noEmoji"] = anti2["comment_lemma"].apply(lambda x: re.sub(r"[^a-z\s\(\-:\)\\\/\];='#]", '', x))
```

```
[55] anti2[['comment_lemma','comment_noEmoji']].loc[[267,401,26994]]
```

	comment_lemma	comment_noEmoji
267	ticket sold 😭	ticket sold
401	love 😘😘😘	love
26994	take comment seriously watch acting🤣🤣🤣🤣🤣🤣	take comment seriously watch acting

Figure 4.3.15 Remove emoji

Lastly, for the transformation of the data, we need to remove emojis from the comments. This is because emojis can add noise to the data and can change the meaning of a sentences. Based on figure 4.3.15 above, the process to remove emoji has been successfully done.

```
#Drop unnecessary columns
anti2.drop(['author_name','like_count',"comment","comment_lower","comment_wo_punct","comment_wo_stop",
            "comment_noFreq","comment_lemma","comment_stemmed"], axis=1, inplace=True)

#Rename column name
anti2.rename(columns = {'comment_noEmoji':'clean_comment'}, inplace = True)

anti2.head()
```

	date	clean_comment
0	2023-01-09 16:35:12+00:00	love
1	2023-01-09 16:34:23+00:00	happened can cider two nothing glam hard drug ...
2	2023-01-09 16:32:48+00:00	amazing tay

Figure 4.3.16 Drop columns

To complete the transformation of comments, drop all the columns that has been created during transformation except the last column rename to 'clean_comment'.

Sentiment Labelling

```
nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
anti2["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in anti2["clean_comment"]]
anti2["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in anti2["clean_comment"]]
anti2["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in anti2["clean_comment"]]
anti2['Compound'] = [sentiments.polarity_scores(i)["compound"] for i in anti2["clean_comment"]]
score = anti2["Compound"].values
sentiment = []
for i in score:
    if i > 0 :
        sentiment.append('Positive')
    elif i < 0 :
        sentiment.append('Negative')
    else:
        sentiment.append('Neutral')
anti2["Sentiment"] = sentiment
```

```
[62] anti2.loc[20:25]
```

	date	clean_comment	Positive	Negative	Neutral	Compound	Sentiment
20	2023-01-09 12:58:15+00:00	really construction lake ray hubbard go floyd ...	0.074	0.000	0.926	0.3400	Positive
21	2023-01-09 12:57:31+00:00	anti heronoob uuuuuuhhhhhh	0.000	0.535	0.465	-0.3182	Negative
22	2023-01-09 12:50:46+00:00	benfords originally bennett	0.000	0.000	1.000	0.0000	Neutral
23	2023-01-09 12:20:17+00:00	add style hard rock guitar song get better	0.282	0.136	0.583	0.3612	Positive
24	2023-01-09 12:12:50+00:00	awkwardly hot pop star help doesnt like disapp...	0.417	0.342	0.241	0.3711	Positive
25	2023-01-09 11:44:45+00:00	many think add deleted scene would help many e...	0.175	0.175	0.649	0.0000	Neutral

Figure 4.3.17 Sentiment Labelling

We need to label all the comments into sentiment categories which are positive, negative, or neutral. Hence, we call the SentimentIntensityAnalyzer from nltk library to proceed with the sentiment labelling process. It will provide compound score between -1 and 1, where smaller than 0 represents a negative comments, bigger than 0 represents a positive comments, and 0 represents a neutral comments

Label Encoding

```
[116] anti3['Sentiment'].value_counts()

      Neutral     33014
      Positive    27737
      Negative     7580
      Name: Sentiment, dtype: int64

[117] from sklearn.preprocessing import LabelEncoder
      #create instance of label encoder
      lab = LabelEncoder()
      anti3['Sentiment'] = lab.fit_transform(anti3['Sentiment'])

[118] anti3['Sentiment'].value_counts()

      1    33014
      2    27737
      0     7580
      Name: Sentiment, dtype: int64
```

Figure 4.3.18 Label encoding

Perform label encoding to convert the categorical variables into numerical values. As shown in figure 4.3.18 above, neutral represent by 1, positive represent by 2, and negative represent 0.

Balancing Dataset

```
[125] from sklearn.utils import resample
      df_neutral = anti3[(anti3['Sentiment']==1)]
      df_positive = anti3[(anti3['Sentiment']==2)]
      df_negative = anti3[(anti3['Sentiment']==0)]

      # upsample minority classes
      df_positive_upsampled = resample(df_positive,
                                       replace=True,
                                       n_samples= 33014,
                                       random_state=42)

      df_negative_upsampled = resample(df_negative,
                                       replace=True,
                                       n_samples= 33014,
                                       random_state=42)


      # Concatenate the upsampled dataframes with the neutral dataframe
      final_data = pd.concat([df_negative_upsampled,df_neutral,df_positive_upsampled])

[126] final_data['Sentiment'].value_counts()

      0    33014
      1    33014
      2    33014
      Name: Sentiment, dtype: int64
```

Figure 4.3.19 Balanced data

Based on the figure 4.3.18, we can see that the sentiment which is a target variable has imbalance data. To overcome this problem and ensure that the model will not be biased towards certain group in target variable, we need balance the data. Figure 4.3.19 above shows the process of balancing the data set. The data has been resample and put under new data frame that consists of balanced data to make all the category in target variable have equal amount of sentiment.

4.4 DATA ANALYSIS

```
anti3['Sentiment'].value_counts()

Neutral     33014
Positive    27737
Negative     7580
Name: Sentiment, dtype: int64

[103] anti3['Sentiment'].value_counts().plot(kind="bar",color=['orange','green','blue'])
      plt.legend

<function matplotlib.pyplot.legend(*args, **kwargs)>
```

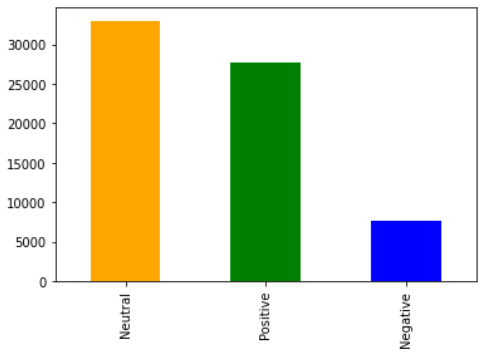


Figure 4.4.1 Total of comment by sentiment

In this study, the sentiment generated are 'Neutral', 'Positive', and 'Negative'. From the plot above, the sentiment with the highest number of comments is 'Neutral'. While the number of negative comments is the least.

```
[104] a=(anti3['Sentiment'].value_counts())/len(anti3)
      a.plot(kind="pie",autopct='%1.1f%%')

<matplotlib.axes._subplots.AxesSubplot at 0x7f8e11c169d0>
```

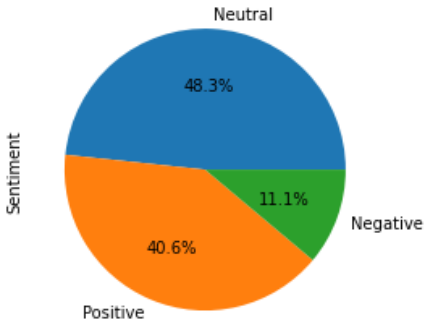


Figure 4.4.2 Pie chart of total comment by sentiment

From the pie chart above, the neutral comments is 48.3% of the total comments written while positive comments is 40.6% of the total comments generated. The negative comments are only 11.1% of the total comments.

The total comments posted is grouped by months. Only four months are involved, October 2022 as Oct, November 2022 as Nov, December 2022 as Dec and January 2023 as Jan. The information is extracted and is printed in a data frame as shown in Figure 4.4.3 below. Figure 4.4.4 shows the graph plotted using the data frame shown in Figure 4.4.3.

```
[420] data=positive.merge(negative,how='inner',on='Month')
      data=data.merge(neutral,how='inner',on='Month')
```

```
data
```

	Month	Positive	Negative	Neutral
0	Oct	23295	5753	28592
1	Nov	3087	961	2893
2	Dec	1461	373	1103
3	Jan	439	92	282

Figure 4.4.3 Data frame of total comments by month and sentiments

```
ax=plt.gca()
data.plot(x="Month",y="Positive",ax=ax,color='orange')
data.plot(x="Month",y="Negative",ax=ax,color='green')
data.plot(x="Month",y="Neutral",ax=ax,color='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f120d09cd00>
```

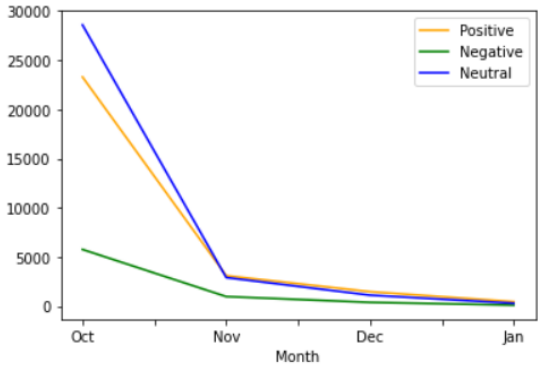


Figure 4.4.4 Graph of types of comments posted from Oct 2022 to Jan 2023

As this video is released in Oct 21, 2022, therefore, as shown in the plot above, comments generated from Oct 2022 to Jan 2023 are analysed. From the plot above, the comments regardless of its sentiment, made in Oct is the highest as it is the month when this video is newly released. The total of comments generated in November drastically dropped. Overall, the neutral and positive comments made are higher than negative comments.

Besides, word clouds have been created for comments of all sentiments. Word cloud is the collection of words appearing in the text and the words in the word cloud are displayed in different sizes. The more often the words appear in the text, the bigger and bolder the words appear in word clouds. Figure 4.4.5, Figure 4.4.6, Figure 4.4.7 shows the word clouds for positive, negative and neutral comments.

```
[75] import matplotlib.pyplot as plt
     fig = plt.figure()
     fig.set_figwidth(14)
     fig.set_figheight(18)
     plt.imshow(pos_com, interpolation='bilinear')
     plt.axis('off')
     plt.show()
```

Figure 4.4.5 Word cloud for positive comment

From the word cloud for positive comments, the word ‘love’ is the biggest compared to the other words displayed in the word cloud. Other than the word ‘love’, words like ‘best’, ‘amazing’, ‘great’, ‘masterpiece’, ‘beautiful’ are among the words depicted in the word cloud.

Figure 4.4.6 Word cloud for negative comment

These words are from the comments which are classified as negative. The words which are displayed in bigger sizes here are ‘im problem’, ‘problem’, ‘disappoint’, ‘tonedeaf person’, ‘bad’, and many more. The words displayed here in the word cloud are not necessarily negative words, some are just the words which appear often in negative comments.

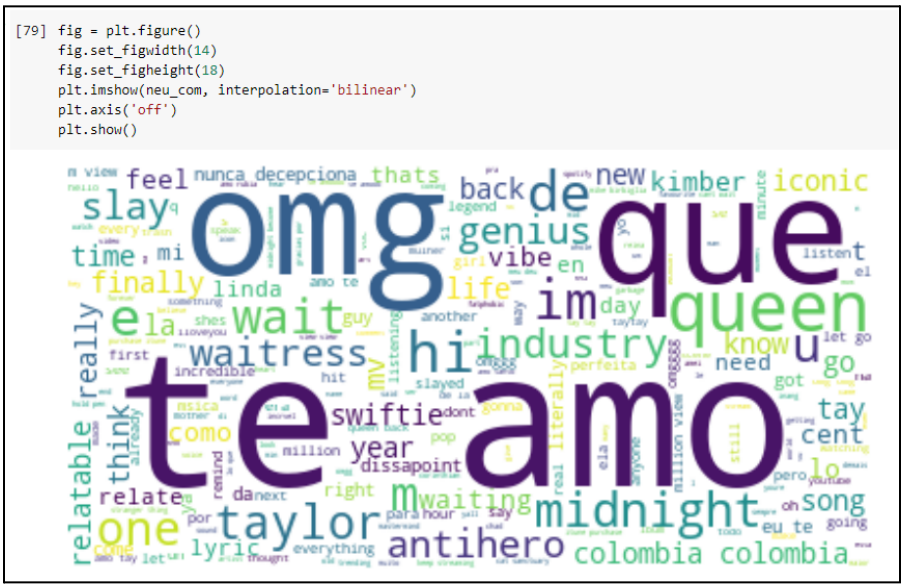


Figure 4.4.7 Word cloud for neutral comment

The word cloud above shows the words which appear in comments which are identified as neutral. From the figure above, ‘te amo’ is the word which appears most often in the neutral comments. Other than the word ‘te amo’, words like ‘queen’, ‘omg’, ‘hi’, ‘midnight’ are among the words depicted in the word cloud.

5.0 MODELLING

In order to perform text classification, this project will use the Logistic Regresion, Linear SVC and 2 Naive Bayes algorithms: GaussianNB and MutinomialNB.

```
[73] from sklearn.feature_extraction.text import CountVectorizer
     cv = CountVectorizer(max_features=1500)
     X = cv.fit_transform(comment).toarray()
     y = final_data["Sentiment"].values
```

Before train these models, the comments of the data is set as input attribute and the sentiment of the data is set as target. The comment attribute has convert to vector by using CountVectorizer() and transform to array by using toarray(). The max_features is set to 1500 due to computer hardware limitations.

```
[75] 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)
```

After that, input and target attributes split 70% to train set and 30% to test set.

```
[76] from sklearn.linear_model import LogisticRegression
     LR = LogisticRegression()
     LR.fit(X_train, y_train)
```

```
[77] from sklearn.svm import LinearSVC
     SVC = LinearSVC()
     SVC.fit(X_train, y_train)
```

```
[78] from sklearn.naive_bayes import GaussianNB
     GNB = GaussianNB()
     GNB.fit(X_train, y_train)
```

```
[79] from sklearn.naive_bayes import MultinomialNB
     MNB = MultinomialNB()
     MNB.fit(X_train, y_train)
```

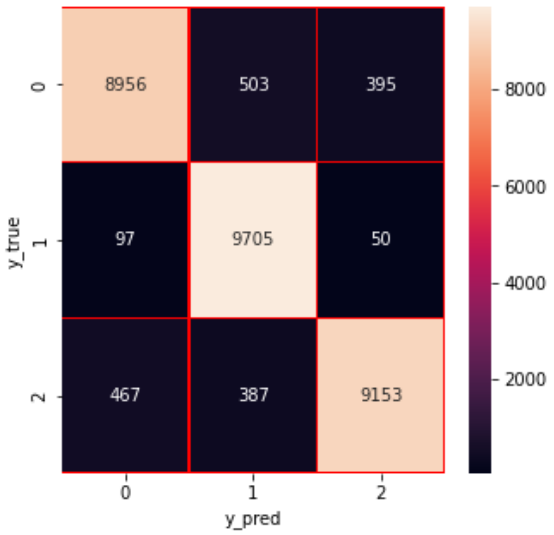
Next, all models are trained with default parameters.

6.0 RESULT DISCUSSION

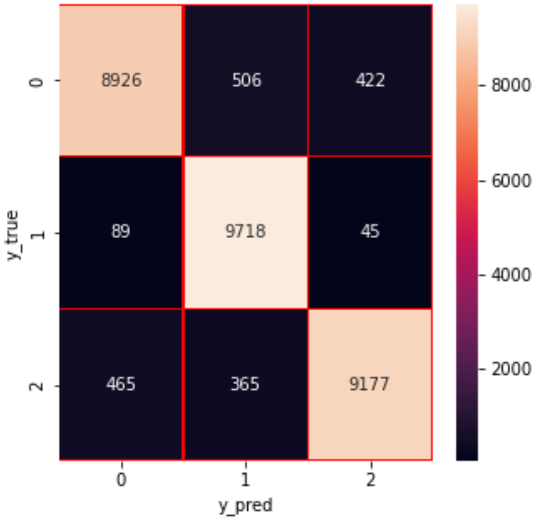
These models are trained to make predictions. To see the result of prediction clearly, the 10 of the comments from the dataset chosen randomly as input of the models to do the predictions, and then compared to the actual sentiment.

	clean_comment	Sentiment	Logistic Regression	Linear SVC	GaussianNB	MultinomialNB
68160	holy shit	0	0	0	0	0
10661	fan since	2	2	2	2	2
35368	taylor kid terrible hope dont actually end li...	0	0	0	0	0
8369	clip look beatifull	1	1	1	1	2
48654		1	1	1	1	1
50014	first heard name antihero mayhem think midnigh...	0	0	0	0	0
180	watched lost found katz lost depiction several...	0	0	0	0	0
54663	everything hit miss	0	0	0	2	0
67568	yass	1	1	1	1	1
20941	im problem yes indeed thats keep avoiding ppl ...	0	0	0	2	0

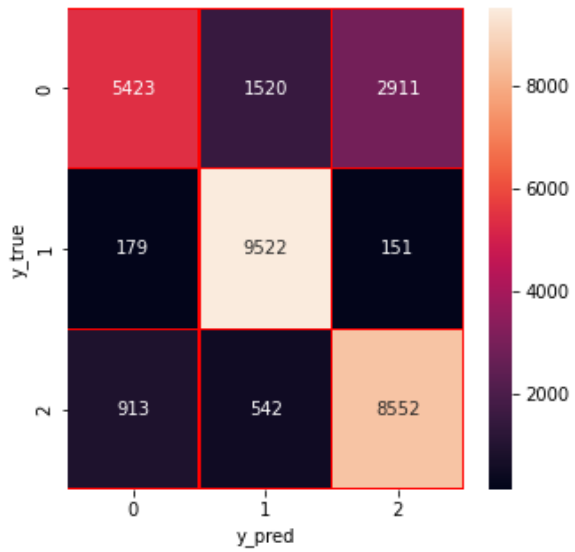
The Sentiment column is the actual sentiment, while the Logistic Regression, Linear SVC, GaussianNB and MultinomialNB columns are the predictions from these models. Out of 10 sentiment comments, Logistic Regression and Linear SVC successfully predicted 10 reviews, GaussianNB correctly predicted 8 comments, and MultinomialNB correctly predicted 9 comments. Let's look at the confusion matrix to see the number of correct predictions for the entire data.



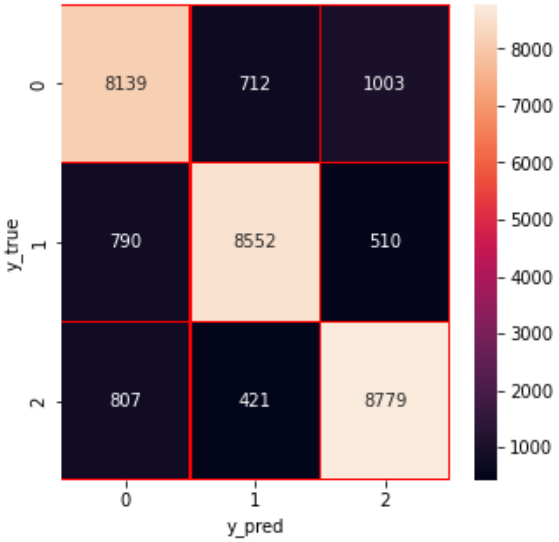
Logistic Regression



Linear SVC



GaussianNB



MultinomialNB

A good confusion matrix is like a diagonal matrix. When y_true = y_pred, it means that the model prediction is accurate. From the confusion matrixes above, the confusion matrix of Logistic Regression and Linear SVC almost like a diagonal matrix. The true positive of Logistic Regression is 8956+ 9705 + 9153 = 27814, and that of Linear SVC is 8926 + 9718 + 9177 = 27821, both of which are very high, indicating that they are very good. However, it can be seen from the confusion matrix of GaussianNB and MultinomialNB that all values other than true positive are high, indicating that they not as good as the two models mentioned before. From these confusion matrices, accuracy, precision, recall and f1-scores can be calculated. These evaluation matrices will also be performed using jupyter.

Logistic Regression

	precision	recall	f1-score	support
0	0.94	0.91	0.92	9854
1	0.92	0.99	0.95	9852
2	0.95	0.91	0.93	10007
accuracy			0.94	29713
macro avg	0.94	0.94	0.94	29713
weighted avg	0.94	0.94	0.94	29713

Linear SVC

	precision	recall	f1-score	support
0	0.94	0.91	0.92	9854
1	0.92	0.99	0.95	9852
2	0.95	0.92	0.93	10007
accuracy			0.94	29713
macro avg	0.94	0.94	0.94	29713
weighted avg	0.94	0.94	0.94	29713

GaussianNB

	precision	recall	f1-score	support
0	0.83	0.55	0.66	9854
1	0.82	0.97	0.89	9852
2	0.74	0.85	0.79	10007
accuracy			0.79	29713
macro avg	0.80	0.79	0.78	29713
weighted avg	0.80	0.79	0.78	29713

MultinomialNB

	precision	recall	f1-score	support
0	0.84	0.83	0.83	9854
1	0.88	0.87	0.88	9852
2	0.85	0.88	0.86	10007
accuracy			0.86	29713
macro avg	0.86	0.86	0.86	29713
weighted avg	0.86	0.86	0.86	29713

The accuracy of Logistic regression and Linear SVC is same, both 0.94. This means that 94% of the comment sentiments were accurately predicted. MultinomialNB also not bad, 86% of our data was predicted accurately. The performance of GaussianNB is a bit low compare to other models, with an accuracy rate of only 79%.

```
[101] from sklearn.metrics import mean_squared_error
      rms_LR = mean_squared_error(y_test, LR_y_pred, squared=False)
      rms_SVC = mean_squared_error(y_test, SVC_y_pred, squared=False)
      rms_GNB = mean_squared_error(y_test, GNB_y_pred, squared=False)
      rms_MNB = mean_squared_error(y_test, MNB_y_pred, squared=False)
```

```
rms_LR
```

```
0.38242872517767723
```

```
[103] rms_SVC
```

```
0.38812320114392784
```

```
[104] rms_GNB
```

```
0.7938102257166745
```

```
[105] rms_MNB
```

```
0.5645613671217954
```

When the RMSE is low, which is ideal, it shows the mistake at the outcome that depends on the input. From the result, RMSE Logistic Regression is lowest, which is 0.3824. Then followed by Linear SVC (0.3881) , MultinomialNB (0.5646) and GaussianNB (0.7938. Therefore, it can be concluded that logistic regression is the best model.

The efficacy of a model has been known through 30% of the test dataset. Now, let's look at using new comments to make these models predict.

```
[575] new_txt=pd.DataFrame({"text": ["love this song!!!", "bad song...", "10/1/2023 viewed"]})
      new_txt["text"] = new_txt["text"].apply(lambda text: remove_punctuation(text))
      new_txt["text"] = new_txt["text"].apply(lambda text: remove_stopwords(text))
      new_txt["text"] = new_txt["text"].apply(lambda text: remove_freqwords(text))
      new_txt["text"] = new_txt["text"].apply(lambda text: lemma_words(text))
      new_txt['text']= new_txt['text'].apply(lambda x:re.sub(r"[^a-z\s\(\-:\)\\\/\];='#]", '', x))
      new_txt
```

	text
0	love song
1	bad song
2	viewed

Here, 3 new comments are defined, which one is a positive comment, one is a negative comment and another one is neutral comment. The expected result should be [2, 0, 1], where 2 is a positive comment, 0 is a negative comment, and 1 is a neutral comment. Let's see the result:

```
[147] test = pd.DataFrame({"Model":["GaussianNB", "MultinomialNB", "Logistic Regression", "Linear SVC"],
                "Actual": ["[2, 0, 1]", "[2, 0, 1]", "[2, 0, 1]", "[2, 0, 1]"],
                "Prediction": [GNB.predict(xx), MNB.predict(xx), LR.predict(xx), SVC.predict(xx)]})
      test
```

	Model	Actual	Prediction
0	GaussianNB	[2, 0, 1]	[2, 2, 1]
1	MultinomialNB	[2, 0, 1]	[2, 0, 1]
2	Logistic Regression	[2, 0, 1]	[2, 0, 1]
3	Linear SVC	[2, 0, 1]	[2, 0, 1]

Logistic Regression, Linear SVC, and MultinomialNB correctly predicted sentiment reviews, while GaussianNB got 1 out of 3 predictions wrong.

7.0 CONCLUSION

Based on the analysis that had been made to achieve the objectives, we can clearly see that neutral comments has the highest sentiment values for Taylor Swift's Anti-Hero music video. Meanwhile, the negative comments have the lowest count. As for model development, Logistic Regression and Linear SVC turn out to be the best models with the highest accuracy results which are 94%. However, if want to choose only one model to be used, we can say that Linear SVC is the best model among all the models that are trained and tested with the data. This can be proved by its confusion matrix which is the diagonal matrix of Linear SVC has the highest values. Even though Linear SVC shares the same model accuracy value with Logistic Regression, the RMSE of Logistic Regression model is slightly higher than Linear SVC model suggesting that Logistic Regression model is the better one among these two models. By considering all the aspects stated above, we can conclude that Logistic Regression is the best model among the models used. Therefore, Logistic Regression is suitable for sentiment analysis.

In conclusion, sentiment analysis of YouTube comments is a challenging task due to the unstructured, informal, and sometimes offensive nature of the comments. However, it is important for YouTube content creators and marketers to understand the sentiment of their audience's comments in order to improve their content and engagement. For Taylor Swift's Anti-Hero song, we can say that the edited music video has been accepted by her audience with positive outlook. Taylor has made a great decision by editing the video to reduce the controversy. However, it's also important to note that sentiment analysis is a difficult task and no matter how good the model is, it will not be able to understand the context or sarcasm in a comment, so human validation should be considered in order to achieve the best results.

8.0 REFERENCES

1. Aniftos, R. (2022, October 27). Taylor Swift ‘Anti-Hero’ Music Video’s ‘Fat’ Scale Scene Removed After Controversy. Retrieved from https://www.billboard.com/music/pop/taylor-swift-anti-hero-video-fat-scale-scene-cut-after-backlash-1235161678/

2. Arun. (2022). Youtube comments sentiment analyser. Retrieved from https://www.kaggle.com/code/arunkumar1809/youtube-comments-sentiment-analyser

3. Chan, S. (2022). YouTube Comments to CSV (✓ Official YouTube API, ✘ scraping) w Python. Retrieved from https://www.youtube.com/watch?v=XTjtPc0uiG8

4. Georgi, M. (2022, October 29). Taylor Swift let fans down by removing ‘fatphobic’ scene from ‘Anti-Hero’ video. Retrieved from https://www.nbcnews.com/think/opinion/taylor-swift-should-not-remove-fatphobic-scene-anti-hero-video-rcna54617

5. Herald, NZ. (2022, October 31). Taylor Swift cuts ‘offensive’ scene from music video after being accused of fatphobia. Retrieved from https://www.nzherald.co.nz/entertainment/taylor-swift-cuts-offensive-scene-from-music-video-after-being-accused-of-fatphobia/BRDH7WIBORBRNDPIQTFOUZBFDQ/

6. Nabi, J. (2018,September 13). Machine Learning - Text Processing. Retrieved from https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

7. Upendra. (2022). YouTube API to extract Comments. Retrieved from https://www.youtube.com/watch?v=N--0rmo0ctA

8. Wikipedia. (n.d.). Anti-Hero (song). Retrieved from https://en.wikipedia.org/wiki/Anti-Hero_(song)

APPENDICES

1. Comment scrapping:

   PDF - https://drive.google.com/file/d/1qUTZz0vNQpabJuyGoEblI71XHFVygUl7/view?usp=share_link

   Code - https://drive.google.com/file/d/1iCuXAH8Vghc9kJhfbwAlEurbVdndnDbb/view?usp=share_link

2. Preprocessing, Analysis, and Modelling:

   PDF - https://drive.google.com/file/d/1v0M-726_OhQPem0xVtHUIHlF_HV6vK41/view?usp=share_link

   Code - https://drive.google.com/file/d/1yZAavZqwejKdrM9AkPeZA1n2riS9oPlH/view?usp=share_link

3. Datasets - https://drive.google.com/drive/folders/1khmixbNKX1EXkFKjdfWxu6x5iFHoCeoH?usp=share_link