helgemoes-abtest

March 24, 2024

0.1 Notes about this template

1. This Jupyter Notebook is a template that must be used for A/B test report, as it is aligned with the grading criteria. Not following this template may have an impact on the grade. Please follow all the instructions included here.
2. The appendix with the answers to the Data Ethics Decision Aid is to be handed in on a separate Canvas assignment, and in Word or PDF (not Jupyter notebook). See instructions on Canvas.

1 Hamonizing Metrics: Predicting Music Engagement on YouTube

- Student Name: Helge Moes
- Student Number: 11348801
- Date: 24-03-2024
- Wordcount per section:
    - Introduction and Theoretical Justification: 884
    - Gathering data: 742
    - Data Exploration & Evaluation: 2377
    - Evaluation: 929
    - Limitations and Next Steps: 503
    - Ethical and Normative Considerations: 734

1.1 Introduction and Theoretical Justification

Doing data analysis is not just about crunching the numbers, but also explaining what is being done, and why. Moreover, it is important to document your steps in a way that other analysts (or even yourself) can understand what was done, with what type of data, and based on what assumptions.

This section briefly introduces the organization, the communication challenge and proposes a general RQ so that the reader has enough context to understand the actual analysis being done in the notebook. This section also provides theoretical justification leading up to one hypothesis that will be tested in the A/B test.

Throughout history, music has always been a prominent medium that has been consumed by many, whether live or on the radio. However, through the introduction of platforms, such as Applemusic, Spotify and YouTube, it has shifted to another medium: online streaming services (Simon, 2019).

YouTube in particular is interesting, since it started with the intention to enable regular users to publish videos (Liikkanen & Salovaara, 2015).

While their initial emphasis was on videos, the online streaming platform shifted to presenting music videos, which made it accessible for everyone anywhere to listen or watch music. Eventually the platform became the most recognized digital music brand in 2014 (Liikkanen & Salovaara, 2015). This popularity was accompanied with a gradual development of the platform into a professional media outlet that has been used by brands in order to interact with their audience. As a result, the user has access to a vast library of millions of music at any time and as many times as they want.

Besides this unlimited access, the rise of online streaming platforms made it also possible to analyze the data of music and whether users engage more with certain content. By analyzing music data on YouTube, it can provide insights into listener preferences and behavior, which not only can improve YouTube's value proposition as a company that offers targeted content based on the user's preferences, but it can also help the creators and users to improve their music production on the platform. The relevance of this research is to provide insights to how YouTube can improve their music recommendation based on engagement metrics of users and understand how this relates to the engagement, which in turn shall improve the user experience and increase user engagement. Due to the challenge of defining engagement, this study shall regard engagement mainly through metrics, such as viewCount, likeCount and commentCount. Moreover, the results of this research can also serve as a means to understand how successful music content (lyrics) can be created based on engagement for other platforms that also provide recommendations of music to users, such as Tidal, Applemusic and Spotify, since they also rely on customer feedback for a strong presence of a certain content (Smith et al., 2012).

Based on these grounds, this research shall attempt to uncover the impact of language in music and how it affects user behavior and engagement. More specifically, it shall explore whether music with profound or simple language holds an impact on user engagement. Therefore, the following question is formulated:

Research Question: To what extent does language in music have an impact and can we use metadata to predict the success of music on YouTube?

In order to answer this question, the following sub questions shall be explored: * What features are associated with increasing engagement on YouTube? * What models are best suited to predict popularity on YouTube based on language and user feedback? * How can the insights be used to improve music marketing and content creation on the YouTube platform?

Through this A/B test, the study aims to evaluate the accuracy of machine learning models in predicting popularity using metadata. Moreover, the objective is to identify the most effective model by comparing performance metrics such as like count, comment count, video categories, and duration in seconds. Consequently, insights gained from the analysis will be utilized to enhance music marketing and content creation on YouTube. This involves tailoring content to align with engagement-driving features and experimenting with various content formats, particularly focusing on lyrics.

Additionally, the research explores opportunities to enhance content recommendation algorithms on YouTube. This includes considering the integration of features such as video category, duration, and language to personalize user experiences and increase engagement with music content. Lastly, the investigation aims to explore the impact of language in music on user engagement metrics. The goal is to determine whether views, likes, and comments can accurately predict this. This research aims to refine content creation strategies and recommend language styles that effectively resonate with the target audience.

To achieve this objective, the study will utilize an exploratory approach, integrating quantitative analysis techniques with machine learning modeling. The methodology encompasses several stages, including data collection, preprocessing, model selection, and analysis interpretation. The dataset for this study comprises YouTube music video metadata, including features such as view count, like count, comment count, language complexity metrics (e.g., Flesch-Kincaid Grade Level, Gunning Fog Index), and other relevant predictors. The data will be retrieved using YouTube's API, ensuring compliance with platform policies and terms of service. Prior to analysis, the dataset will undergo preprocessing to address outliers, missing values, and skewed distributions. Techniques such as normalization, outlier detection, and imputation will be applied as necessary to ensure the quality and integrity of the data.

The analysis will involve the selection and implementation of appropriate machine learning models to predict user engagement metrics based on metadata features. Two primary models will be employed: robust regression and a random forest regressor. The robust regression will be utilized to accommodate outliers and skewed data distributions, addressing potential challenges identified during the exploratory data analysis phase and the robust regression is well-suited for handling non-normal data distributions and is expected to provide reliable predictions for engagement metrics. Finally, the random forest model will be employed to capture potential non-linear relationships between independent and dependent variables.

1.2 Gathering Data

In the cells below you should load the dataset(s) into pandas to begin the data understanding and preparation. It is important to provide the reader with sufficient information to understand how the data were collected, and what the dataset is about.

More specifically, you should answer at least adress the following topics in this section: * What is the dataset about? How does it relate to the business challenge? * What is the source of the data? * You can for example think of Google Analytics or data collected through social media (when e.g., tracking effectiveness of campaigns) * How were the data collected? What type of data collection (and/or sampling) strategy was used, and why? * What are the potential biases that could be introduced in the dataset because of the data collection method? * How was privacy taken in consideration during data collection? What are some of the risks and/or privacy trade-offs that should be considered?

As a tip: use a meaningful name for your dataset(s) when loading them into a Pandas dataframe. Calling them just *df* or *data* will become confusing later on.

IMPORTANT: As part of open science and replicability, it is important to indicate also to the reader where the data is stored or located. In the case of the report, we expect you to upload the dataset to our SurfDrive folder, so we can download the data and replicate your results.

For this A/B testing, the following datasets were used: * `YouTube_Music.csv` * `YouTube_Music_Transcript.csv`

YouTube_Music.csv The dataset was collected through YouTubeDataTools from Digital Methods Initiative of the Universiteit van Amsterdam in order to define the relevant variables for this exploratory analysis (YouTube Data Tools, n.d.). The query "music" was used in the example dataset and English music content was filtered between January 2023 and January 2024, which resulted in a dataset with 445 songs and 29 variables.

This dataset contains information about various YouTube music videos, including details such as video ID, channel ID, video title, publication date, video description, tags, and statistics like view count, like count, dislike count, favorite count, and comment count. The data allows for various analyses or business purposes such as understanding audience preferences, measuring the popularity of different songs or artists, or evaluating the performance of music videos.

The source of the data appears to be YouTube, as the dataset contains information about YouTube videos, including video IDs and channel IDs. These IDs are unique identifiers assigned by YouTube to each video and channel.

The data collection method likely involves accessing YouTube's API to retrieve information about music videos. This API allows developers to programmatically access and retrieve data from YouTube, including video metadata and statistics. The sampling strategy may involve querying the API for specific video IDs or channels of interest.

Potential biases in the dataset could arise due to various factors such as the popularity of certain genres or artists, regional preferences, or promotion strategies by music labels or channels. For example, certain videos may receive more views or likes due to being promoted heavily by their creators or having viral content, leading to skewness in the data.

Privacy considerations during data collection may involve adhering to YouTube's terms of service and privacy policies regarding the collection and use of user data. Risks include inadvertently collecting sensitive information from video descriptions or comments, violating users' privacy rights, or misusing the data for targeted advertising or profiling purposes. It's essential to handle and analyze the data ethically and ensure compliance with relevant privacy regulations.

YouTube_Music_Transcript.csv Besides this dataset, a separate dataset of the transcript was collected including the transcript of the `videoId` from the YouTube_Music.csv to assess whether the content of the music resonates with the users. In order to realize this, the youtube-transcript-api (Depoix, n.d.) was used in Code_Retrieving_Transcript.

The dataset comprises transcripts of YouTube music videos identified by their respective video IDs. Each row represents a segment of text from a specific video. Analyzing these transcripts could offer insights into the content, themes, and language used in the videos, benefiting businesses engaged in music marketing, content creation, or audience analysis.

The data originates from YouTube, where the videos were uploaded and subsequently transcribed, likely through automated or manual processes. The sampling strategy may involve selecting specific videos or segments for transcription.

Biases may arise due to variations in transcription quality, accents, dialects, languages, speech patterns, or overrepresentation of certain genres or artists in the selection process.

Privacy considerations include handling personal information mentioned in the transcripts, such as names or locations, to mitigate risks such as inadvertent exposure of sensitive information or privacy violations. Anonymization and compliance with relevant privacy regulations are crucial to addressing these risks.

[1]:
```
# Importing the os module which provides a way to interact with the operating␣
↪system, including file management and environment variables.
import os

# Getting the current working directory
current_directory = os.getcwd()

# Printing the current working directory
print("Current working directory:", current_directory)
```

```
Current working directory: /Users/helgegeurtjacobusmoes/Desktop/Digital
Analytics/AB_Test_HelgeMoes
```

Firstly, we have a first glance on the datasets in order to observe whether the data has been succesfully collected.

[79]:
```
# Importing pandas library and assign alias 'pd'
import pandas as pd

# Reading the CSV files into a DataFrame
YouTube_Music_df = pd.read_csv('YouTube_Music.csv')
YouTube_Music_Transcript_df = pd.read_csv('YouTube_Music_Transcript.csv')

# Printing the DataFrames of the CSV files
print(YouTube_Music_Transcript_df)
print(YouTube_Music_df)
```

```
          Video ID                                        Text
0      suAR1PYFNYA                                     [Music]
1      suAR1PYFNYA     okay I come and I go tell me all the
2      suAR1PYFNYA   ways you need me I'm not here for long
3      suAR1PYFNYA     catch me or I go who I come and I go
4      suAR1PYFNYA     prove you got the right to please me
…              …                                           …
56707  dviOJyEq024  hit subscribe or follow and let me know
56708  dviOJyEq024               if you can hear a difference
56709  dviOJyEq024                                        good
56710  dviOJyEq024                                     [Music]
56711  dviOJyEq024                                   thank you

[56712 rows x 2 columns]
     position                 channelId       channelTitle      videoId
0           1  UC-J-KZfRV8c13fOCkhXdLiQ           Dua Lipa  suAR1PYFNYA  \
1           2  UCSJ4gkVC6NrvII8umztf0Ow          Lofi Girl  4xDzrJKXOOY
2           3  UCtkLwbqLqVxn3ZSr2P2fPMw  Soft Rock Rhythms  EwBc8ChPhic
3           4  UCvt5p3A11M8zd8iJPCC5XvQ   Love Life Lyrics  uOBhX3tCzAY
```

```
4          5  UCq-Fj5jknLsUf-MWSy4_brA          T-Series  eL2vQyP6DQE
..        …                         …                 …          …
441      442  UCePqhJ54KQpg_LZ3S6aAEYg      Star India 1  y64m5o1rMG8
442      443  UCmwZIuy9LKeGDX47Xf5mH9A     Harshal Music  yy3bbXuHr9w
443      444  UC6_9HdRBHUC07tS3XGO7Wtg          May Madi  V1UFl6Vl3Lg
444      445  UCeDPbIoAbac_v4U3bpSvn1A  Katya Lel - Topic  f--UUHGg_Mc
445      446  UCPVjQ58A0BujzVMEmFCJ3hA       Danny Sapko  dviOJyEqO24

              publishedAt       publishedAtSQL
0    2023-11-09T23:00:11Z  2023-11-09 23:00:11  \
1    2023-07-02T21:17:31Z  2023-07-02 21:17:31
2    2023-12-25T04:26:48Z  2023-12-25 04:26:48
3    2023-07-02T11:30:30Z  2023-07-02 11:30:30
4    2023-12-15T06:30:08Z  2023-12-15 06:30:08
..                    …                   …
441  2023-03-30T05:43:31Z  2023-03-30 05:43:31
442  2023-12-30T13:00:08Z  2023-12-30 13:00:08
443  2023-03-27T10:12:42Z  2023-03-27 10:12:42
444  2023-05-04T19:44:31Z  2023-05-04 19:44:31
445  2023-02-20T23:37:59Z  2023-02-20 23:37:59

                                             videoTitle
0            Dua Lipa - Houdini (Official Music Video)  \
1           synthwave radio   - beats to chill/game to
2    Elton John, Lionel Richie, Eric Clapton, Micha…
3    Summer 2023 playlist  Best summer songs 2023 …
4    FIGHTER: Sher Khul Gaye Song, Hrithik, Deepika…
..                                                …
441        Ambar Se Toda | RRR SONG | Shreesha's Video
442  Safar Mashup | Harshal Music | Safar X Stay X …
443              May Madi -       [Music Video]
444                                 (Speed Up)
445  £4 Guitar Lead vs. £114 Guitar Lead #bass #bas…

                                       videoDescription
0    Listen to Houdini: https://dualipa.lnk.to/houd…  \
1      | Ongoing event → https://www.youtube.com/wa…
2    Elton John, Lionel Richie, Eric Clapton, Micha…
3    #summer2023 #summer #lovelifelyrics Summer 202…
4    The Wait Is Over! Move with The Squadron Leade…
..                                                …
441  Shreesha's Video Video by, Shaurya Photos Sarj…
442  #safar #kinachir #harshalmusic Buy Me A Coffee…
443             Music Michel Shine M…
444  Provided to YouTube by Izdatelstvo Monolit   …
445  Please support my channel by becoming my Patro…

                                                  tags  videoCategoryId  …
```

```
0    dua lipa,dua,new rules,dua leepa,idgaf,scared …              10  …  \
1    chilledcow,chilled cow,lofi,lofi hiphop,lofi h…              10  …
2    soft rock,rock,rock music,soft music,soft song…              10  …
3    summer 2023,summer songs,summer playlist,throw…              10  …
4    hindi songs 2023,hindi songs new,bollywood son…              10  …
..                                                …              …  …
441          RRR,T series,Child,kids photography,Hd,4k               1  …
442                                                NaN              10  …
443    ,      ,May Madi,Sapar ka Lin A Chi…              10  …
444                  ,          (Speed Up)              10  …
445  danny sapko,bass,funk bass,bass solo,free mp3 …              10  …

                                       thumbnail_maxres licensedContent
0    https://i.ytimg.com/vi/suAR1PYFNYA/maxresdefau…            1.0  \
1    https://i.ytimg.com/vi/4xDzrJKXOOY/maxresdefau…            1.0
2    https://i.ytimg.com/vi/EwBc8ChPhic/maxresdefau…            1.0
3    https://i.ytimg.com/vi/u0BhX3tCzAY/maxresdefau…            NaN
4    https://i.ytimg.com/vi/eL2vQyP6DQE/maxresdefau…            1.0
..                                                …              …
441  https://i.ytimg.com/vi/y64m5o1rMG8/maxresdefau…            1.0
442  https://i.ytimg.com/vi/yy3bbXuHr9w/maxresdefau…            1.0
443  https://i.ytimg.com/vi/V1UFl6Vl3Lg/maxresdefau…            NaN
444  https://i.ytimg.com/vi/f--UUHGg_Mc/maxresdefau…            1.0
445  https://i.ytimg.com/vi/dviOJyEqO24/maxresdefau…            1.0

    locationDescription  latitude longitude viewCount  likeCount dislikeCount
0                   NaN       NaN       NaN  85528644  1065658.0          NaN  \
1                   NaN       NaN       NaN  23563068   256228.0          NaN
2                   NaN       NaN       NaN    307190    12256.0          NaN
3                   NaN       NaN       NaN   9992568    42457.0          NaN
4                   NaN       NaN       NaN  68936472   674332.0          NaN
..                    …         …         …         …          …            …
441                 NaN       NaN       NaN  37678452   588306.0          NaN
442                 NaN       NaN       NaN   2100517    17905.0          NaN
443                 NaN       NaN       NaN   2675170    18036.0          NaN
444                 NaN       NaN       NaN  21199099   471826.0          NaN
445                 NaN       NaN       NaN    169731    10851.0          NaN

    favoriteCount commentCount
0               0      40424.0
1               0          0.0
2               0          0.0
3               0       1353.0
4               0      32943.0
..            …           …
441             0          NaN
442             0         98.0
443             0        270.0
```

```
444             0      9206.0
445             0       576.0

[446 rows x 29 columns]
```

Quality Checks Before proceeding with the analysis, it's essential to assess the quality of the data in both dataframes. This examination aims to ensure that the data provides a comprehensive representation of music on YouTube. By conducting these checks, we can identify any inconsistencies or anomalies in the data, which will guide the cleaning process to ensure the dataframes are ready for analysis.

[3]:
```python
# Displaying basic information about DataFrame shapes and data types
print("Shape of YouTube_Music_df:", YouTube_Music_df.shape, "| Length:",␣
 ↪len(YouTube_Music_df))
print("Data types in YouTube_Music_df:")
print(YouTube_Music_df.dtypes)
print()

print("Shape of YouTube_Music_Transcript_df:", YouTube_Music_Transcript_df.
 ↪shape, "| Length:", len(YouTube_Music_Transcript_df))
print("Data types in YouTube_Music_Transcript_df:")
print(YouTube_Music_Transcript_df.dtypes)
print()

# Checking for missing values
print("Missing values in YouTube_Music_df:")
print(YouTube_Music_df.isnull().sum())
print()

print("Missing values in YouTube_Music_Transcript_df:")
print(YouTube_Music_Transcript_df.isnull().sum())
print()

# Displaying descriptive statistics
print("Descriptive statistics for YouTube_Music_df:")
print(YouTube_Music_df.describe())
print()

print("Descriptive statistics for YouTube_Music_Transcript_df:")
print(YouTube_Music_Transcript_df.describe())
```

```
Shape of YouTube_Music_df: (446, 29) | Length: 446
Data types in YouTube_Music_df:
position                      int64
channelId                    object
channelTitle                 object
videoId                      object
publishedAt                  object
```

```
publishedAtSQL           object
videoTitle               object
videoDescription         object
tags                     object
videoCategoryId           int64
videoCategoryLabel       object
topicCategories          object
duration                 object
durationSec               int64
dimension                object
definition               object
caption                    bool
defaultLanguage          object
defaultLAudioLanguage    object
thumbnail_maxres         object
licensedContent         float64
locationDescription      object
latitude                float64
longitude               float64
viewCount                 int64
likeCount               float64
dislikeCount            float64
favoriteCount             int64
commentCount            float64
dtype: object

Shape of YouTube_Music_Transcript_df: (56712, 2) | Length: 56712
Data types in YouTube_Music_Transcript_df:
Video ID    object
Text        object
dtype: object

Missing values in YouTube_Music_df:
position                 0
channelId                0
channelTitle             0
videoId                  0
publishedAt              0
publishedAtSQL           0
videoTitle               0
videoDescription        24
tags                   106
videoCategoryId          0
videoCategoryLabel       0
topicCategories          0
duration                 0
durationSec              0
dimension                0
```

```
definition                0
caption                   0
defaultLanguage         364
defaultLAudioLanguage   226
thumbnail_maxres         25
licensedContent         187
locationDescription     436
latitude                436
longitude               436
viewCount                 0
likeCount                11
dislikeCount            446
favoriteCount             0
commentCount              3
dtype: int64

Missing values in YouTube_Music_Transcript_df:
Video ID      0
Text        247
dtype: int64

Descriptive statistics for YouTube_Music_df:
         position  videoCategoryId  durationSec  licensedContent   latitude
count  446.000000       446.000000   446.000000            259.0  10.000000  \
mean   223.500000        13.316143   563.132287              1.0  18.272893
std    128.893367         5.889702   887.004198              0.0  25.700948
min      1.000000         1.000000     0.000000              1.0 -28.730483
25%    112.250000        10.000000   136.500000              1.0  12.818217
50%    223.500000        10.000000   215.000000              1.0  26.929097
75%    334.750000        22.000000   377.750000              1.0  36.254868
max    446.000000        25.000000  3582.000000              1.0  40.712775

       longitude     viewCount     likeCount  dislikeCount  favoriteCount
count  10.000000  4.460000e+02  4.350000e+02           0.0          446.0  \
mean    4.224011  1.490163e+07  1.944688e+05           NaN            0.0
std    76.899455  3.343310e+07  6.094923e+05           NaN            0.0
min   -95.712891  3.294000e+04  1.339000e+03           NaN            0.0
25%   -74.005973  1.901541e+06  1.624750e+04           NaN            0.0
50%    28.728997  5.106565e+06  4.737300e+04           NaN            0.0
75%    75.905995  1.330220e+07  1.591825e+05           NaN            0.0
max    82.820974  4.714133e+08  1.047807e+07           NaN            0.0

       commentCount
count  4.430000e+02
mean   7.120939e+03
std    6.529114e+04
min    0.000000e+00
25%    2.705000e+02
```

```
50%     9.410000e+02
75%     3.761500e+03
max     1.364324e+06

Descriptive statistics for YouTube_Music_Transcript_df:
             Video ID     Text
count           56712    56465
unique            446    18226
top       -QCVMtrDz1w  [Music]
freq             7162    19696
```

Based on the quality checks, the following can be concluded:

YouTube_Music_df:

- **Shape**: It consists of 446 rows and 29 columns.
- **Data Types**: Most columns contain object (string) data types, except for 'position', 'videoCategoryId', 'durationSec', 'licensedContent', 'latitude', 'longitude', 'viewCount', 'likeCount', 'dislikeCount', 'favoriteCount', and 'commentCount' which have numeric data types.
- **Missing Values**: Several columns have missing values, particularly 'videoDescription', 'tags', 'defaultLanguage', 'defaultLAudioLanguage', 'thumbnail_maxres', 'locationDescription', 'latitude', 'longitude', 'likeCount', 'dislikeCount', and 'commentCount'.
- **Descriptive Statistics**: Descriptive statistics reveal insights into numeric columns such as 'position', 'videoCategoryId', 'durationSec', 'viewCount', 'likeCount', 'dislikeCount', 'favoriteCount', and 'commentCount'.

YouTube_Music_Transcript_df:

- **Shape**: It comprises 56712 rows and 2 columns.
- **Data Types**: Both columns are of object (string) data type.
- **Missing Values**: The 'Text' column has 247 missing values.
- **Descriptive Statistics**: Descriptive statistics show the number of unique values for 'Video ID' and 'Text'.

Overall Quality:

- Both DataFrames have missing values that need to be addressed.
- The 'YouTube_Music_Transcript_df' DataFrame has missing text data.
- Descriptive statistics provide insights into the distribution of numeric data but don't reveal much about text data quality. Further analysis or visualization might be needed for a more comprehensive assessment.

1.3 Data Cleaning

As discussed during the tutorials, the stages of data understanding (which we call *Data Exploration & Evaluation* in the evaluation criteria) and data preparation (which we call *Data Cleaning* in the evaluation criteria) are iterative steps, with a lot of back and forth until you have a usable dataset.

In this stage, we expect you to meet the criteria included in the course guide. While doing this, you need to make sure that you are also clearly communicating what is being done in each step. This includes: * Identifying the key variables that your study will use * Explaining to the reader what these variables are * Performing the steps regarding data cleaning and data exploration as indicated on Canvas (briefing) * Explaining to the reader at each step what is being done, and what the output/result means

IMPORTANT: * You only need to do the cleaning and exploration for the variables that are *relevant* to your study. You can skip variables/columns in the dataset that are absolutely irrelevant to your study. This will also allow you to minimize the dataset. * You need to communicate your steps clearly to the reader. This means that you should not just do a *df.describe()* to communicate descriptive statistics in a dataset that contains a lot of irrelevant columns and expect the reader to figure out by her or himself what is relevant. You should instead only show to the reader what is relevant, and explain why. * If you use functions to categorize your data, you should explain how the functions were built (or where they came from), and what is being done.

Cleaning YouTube_Music_df

[4]:
```python
# Reading the csv file into a DataFrame
YouTube_Music_df = pd.read_csv('YouTube_Music.csv')

# Printing the updated Dataframe
print(YouTube_Music_df)
```

```
     position                 channelId        channelTitle      videoId
0           1  UC-J-KZfRV8c13fOCkhXdLiQ            Dua Lipa  suAR1PYFNYA  \
1           2  UCSJ4gkVC6NrvII8umztf0Ow           Lofi Girl  4xDzrJKXOOY
2           3  UCtkLwbqLqVxn3ZSr2P2fPMw   Soft Rock Rhythms  EwBc8ChPhic
3           4  UCvt5p3A11M8zd8iJPCC5XvQ    Love Life Lyrics  u0BhX3tCzAY
4           5  UCq-Fj5jknLsUf-MWSy4_brA            T-Series  eL2vQyP6DQE
..        ...                       ...                 ...          ...
441       442  UCePqhJ54KQpg_LZ3S6aAEYg       Star India 1  y64m5o1rMG8
442       443  UCmwZIuy9LKeGDX47Xf5mH9A      Harshal Music  yy3bbXuHr9w
443       444  UC6_9HdRBHUC07tS3XGO7Wtg            May Madi  V1UF16V13Lg
444       445  UCeDPbIoAbac_v4U3bpSvn1A  Katya Lel - Topic  f--UUHGg_Mc
445       446  UCPVjQ58A0BujzVMEmFCJ3hA        Danny Sapko  dviOJyEqO24

              publishedAt       publishedAtSQL  \
0    2023-11-09T23:00:11Z  2023-11-09 23:00:11
1    2023-07-02T21:17:31Z  2023-07-02 21:17:31
2    2023-12-25T04:26:48Z  2023-12-25 04:26:48
3    2023-07-02T11:30:30Z  2023-07-02 11:30:30
4    2023-12-15T06:30:08Z  2023-12-15 06:30:08
..                    ...                  ...
441  2023-03-30T05:43:31Z  2023-03-30 05:43:31
442  2023-12-30T13:00:08Z  2023-12-30 13:00:08
443  2023-03-27T10:12:42Z  2023-03-27 10:12:42
```

```
444  2023-05-04T19:44:31Z  2023-05-04 19:44:31
445  2023-02-20T23:37:59Z  2023-02-20 23:37:59

                                            videoTitle  
0           Dua Lipa - Houdini (Official Music Video)  \
1          synthwave radio   - beats to chill/game to   
2    Elton John, Lionel Richie, Eric Clapton, Micha…   
3    Summer 2023 playlist  Best summer songs 2023 …   
4    FIGHTER: Sher Khul Gaye Song, Hrithik, Deepika…   
..                                                 …   
441       Ambar Se Toda | RRR SONG | Shreesha's Video   
442  Safar Mashup | Harshal Music | Safar X Stay X …   
443              May Madi -       [Music Video]        
444                                   (Speed Up)       
445  £4 Guitar Lead vs. £114 Guitar Lead #bass #bas…   

                                      videoDescription  
0    Listen to Houdini: https://dualipa.lnk.to/houd…  \
1      | Ongoing event → https://www.youtube.com/wa…   
2    Elton John, Lionel Richie, Eric Clapton, Micha…   
3    #summer2023 #summer #lovelifelyrics Summer 202…   
4    The Wait Is Over! Move with The Squadron Leade…   
..                                                 …   
441  Shreesha's Video Video by, Shaurya Photos Sarj…   
442  #safar #kinachir #harshalmusic Buy Me A Coffee…   
443              Music Michel Shine M…   
444  Provided to YouTube by Izdatelstvo Monolit   …   
445  Please support my channel by becoming my Patro…   

                                                  tags  videoCategoryId  …  
0    dua lipa,dua,new rules,dua leepa,idgaf,scared …               10  …  \
1    chilledcow,chilled cow,lofi,lofi hiphop,lofi h…               10  …   
2    soft rock,rock,rock music,soft music,soft song…               10  …   
3    summer 2023,summer songs,summer playlist,throw…               10  …   
4    hindi songs 2023,hindi songs new,bollywood son…               10  …   
..                                                 …              …  …   
441         RRR,T series,Child,kids photography,Hd,4k                1  …   
442                                               NaN               10  …   
443    ,     ,May Madi,Sapar ka Lin A Chi…             10  …   
444                  ,         (Speed Up)              10  …   
445  danny sapko,bass,funk bass,bass solo,free mp3 …               10  …   

                                      thumbnail_maxres licensedContent  
0    https://i.ytimg.com/vi/suAR1PYFNYA/maxresdefau…             1.0  \
1    https://i.ytimg.com/vi/4xDzrJKXOOY/maxresdefau…             1.0   
2    https://i.ytimg.com/vi/EwBc8ChPhic/maxresdefau…             1.0   
3    https://i.ytimg.com/vi/u0BhX3tCzAY/maxresdefau…             NaN   
4    https://i.ytimg.com/vi/eL2vQyP6DQE/maxresdefau…             1.0   
```

```
..                                                  …            …
441  https://i.ytimg.com/vi/y64m5o1rMG8/maxresdefau…            1.0
442  https://i.ytimg.com/vi/yy3bbXuHr9w/maxresdefau…            1.0
443  https://i.ytimg.com/vi/V1UFl6Vl3Lg/maxresdefau…            NaN
444  https://i.ytimg.com/vi/f--UUHGg_Mc/maxresdefau…            1.0
445  https://i.ytimg.com/vi/dviOJyEqO24/maxresdefau…            1.0

    locationDescription  latitude longitude viewCount  likeCount dislikeCount
0                   NaN       NaN       NaN  85528644  1065658.0          NaN  \
1                   NaN       NaN       NaN  23563068   256228.0          NaN
2                   NaN       NaN       NaN    307190    12256.0          NaN
3                   NaN       NaN       NaN   9992568    42457.0          NaN
4                   NaN       NaN       NaN  68936472   674332.0          NaN
..                  …        …         …        …        …           …
441                 NaN       NaN       NaN  37678452   588306.0          NaN
442                 NaN       NaN       NaN   2100517    17905.0          NaN
443                 NaN       NaN       NaN   2675170    18036.0          NaN
444                 NaN       NaN       NaN  21199099   471826.0          NaN
445                 NaN       NaN       NaN    169731    10851.0          NaN

    favoriteCount commentCount
0               0      40424.0
1               0          0.0
2               0          0.0
3               0       1353.0
4               0      32943.0
..            …           …
441             0          NaN
442             0         98.0
443             0        270.0
444             0       9206.0
445             0        576.0

[446 rows x 29 columns]
```

To clean this dataset, the columns that are not used to explore the data are dropped. Besides getting rid of the irrelevant columns, the links within the description are also removed.

Although it may be argued whether it is necessary to anonymize the channels, I decided to anonymize the `channelId` in order to safeguard data privacy and confidentiality to a certain extent. Besides

This code snippet performs the following operations:

1. **Importing Modules**: The code imports the 're' module, which provides support for regular expressions in Python.
2. **Defining Functions**:
    - `remove_links(text)`: This function takes a text input and removes any URLs present in it using regular expressions.

- `pseudonomize_channel_id(channel_id)`: This function takes a channel ID as input and hashes it to pseudonymize the IDs. If the channel ID is not null, it prepends the hashed value with 'channel_'.

3. **Dropping Columns**: Certain columns are identified for removal from the DataFrame `YouTube_Music_df`. The list of columns to be dropped is stored in the variable `columns_to_drop`. These columns are then removed from the DataFrame using the `drop()` method, and the result is stored in a new DataFrame called `Cleaned_Youtube_Music_df`.
4. **Anonymizing Channel IDs**: The `pseudonomize_channel_id()` function is applied to the 'channelId' column of the `Cleaned_Youtube_Music_df` DataFrame. This function replaces the original channel IDs with hashed versions prepended with 'channel_'.
5. **Removing Links from Video Descriptions**: The `remove_links()` function is applied to the 'videoDescription' column of the `Cleaned_Youtube_Music_df` DataFrame. This function removes any URLs present in the video descriptions.
6. **Printing Updated DataFrame**: The updated DataFrame, `Cleaned_Youtube_Music_df`, is printed to view the changes.

[5]:
```
# Importing the 're' module which provides support for regular expressions␣
↪(regex) in Python.
import re

# Function to remove links from text
def remove_links(text):
    # Check if the input is a string
    if isinstance(text, str):
        # Define the pattern for URLs
        url_pattern = r'https?://\S+|www\.\S+'
        # Replace URLs with an empty string
        return re.sub(url_pattern, '', text)
    else:
        return text

# Function to pseudonomize channel IDs
def pseudonomize_channel_id(channel_id):
    return 'channel_' + str(hash(channel_id)) if pd.notnull(channel_id) else␣
↪channel_id

# Dropping columns
columns_to_drop = ['position', 'publishedAtSQL', 'thumbnail_maxres',␣
↪'licensedContent', 'locationDescription', 'latitude', 'longitude',␣
↪'dislikeCount', 'favoriteCount', 'videoCategoryId', 'dimension',␣
↪'definition', 'caption', 'defaultLanguage', 'defaultLAudioLanguage',␣
↪'duration']
Cleaned_Youtube_Music_df = YouTube_Music_df.drop(columns=columns_to_drop)

# Anonymizing 'channelId'
```

```
Cleaned_Youtube_Music_df['channelId'] = Cleaned_Youtube_Music_df['channelId'].
  ↪apply(pseudonomize_channel_id)

# Applying the function to remove links from the 'videoDescription' column
Cleaned_Youtube_Music_df['videoDescription'] =␣
  ↪Cleaned_Youtube_Music_df['videoDescription'].apply(remove_links)

# Printing the updated DataFrame
print(Cleaned_Youtube_Music_df)
```

```
                       channelId       channelTitle      videoId
0     channel_1810432585761117807           Dua Lipa  suAR1PYFNYA  \
1     channel_-789892521769045203          Lofi Girl  4xDzrJKXOOY
2    channel_-3520964929790031105  Soft Rock Rhythms  EwBc8ChPhic
3    channel_-7440606057559234095   Love Life Lyrics  u0BhX3tCzAY
4     channel_2843571140104507532           T-Series  eL2vQyP6DQE
..                            …                  …          …
441  channel_-3067888933506141187       Star India 1  y64m5o1rMG8
442  channel_-7032093316890253143      Harshal Music  yy3bbXuHr9w
443   channel_9004471261636352241           May Madi  V1UFl6Vl3Lg
444    channel_917143169740759608  Katya Lel - Topic  f--UUHGg_Mc
445  channel_-4543590347096294591        Danny Sapko  dvi0JyEqO24

              publishedAt                                          videoTitle
0    2023-11-09T23:00:11Z          Dua Lipa - Houdini (Official Music Video)  \
1    2023-07-02T21:17:31Z         synthwave radio   - beats to chill/game to
2    2023-12-25T04:26:48Z  Elton John, Lionel Richie, Eric Clapton, Micha…
3    2023-07-02T11:30:30Z  Summer 2023 playlist   Best summer songs 2023 …
4    2023-12-15T06:30:08Z  FIGHTER: Sher Khul Gaye Song, Hrithik, Deepika…
..                    …                                                 …
441  2023-03-30T05:43:31Z        Ambar Se Toda | RRR SONG | Shreesha's Video
442  2023-12-30T13:00:08Z  Safar Mashup | Harshal Music | Safar X Stay X …
443  2023-03-27T10:12:42Z               May Madi -        [Music Video]
444  2023-05-04T19:44:31Z                                 (Speed Up)
445  2023-02-20T23:37:59Z  £4 Guitar Lead vs. £114 Guitar Lead #bass #bas…

                                      videoDescription
0    Listen to Houdini:  Subscribe to the Dua Lipa …  \
1      | Ongoing event →    | Listen on Spotify, Ap…
2    Elton John, Lionel Richie, Eric Clapton, Micha…
3    #summer2023 #summer #lovelifelyrics Summer 202…
4    The Wait Is Over! Move with The Squadron Leade…
..                                                 …
441  Shreesha's Video Video by, Shaurya Photos Sarj…
442  #safar #kinachir #harshalmusic Buy Me A Coffee…
443              Music Michel Shine M…
444  Provided to YouTube by Izdatelstvo Monolit   …
```

```
445  Please support my channel by becoming my Patro…

                                                tags videoCategoryLabel
0    dua lipa,dua,new rules,dua leepa,idgaf,scared …              Music  \
1    chilledcow,chilled cow,lofi,lofi hiphop,lofi h…              Music
2    soft rock,rock,rock music,soft music,soft song…              Music
3    summer 2023,summer songs,summer playlist,throw…              Music
4    hindi songs 2023,hindi songs new,bollywood son…              Music
..                                                 …                …
441          RRR,T series,Child,kids photography,Hd,4k   Film & Animation
442                                                NaN              Music
443    ,      ,May Madi,Sapar ka Lin A Chi…             Music
444                 ,          (Speed Up)             Music
445  danny sapko,bass,funk bass,bass solo,free mp3 …              Music

                                     topicCategories  durationSec
0                                    Music,Pop_music          189  \
1                                              Music            0
2    Christian_music,Country_music,Music,Pop_music,…            0
3                    Electronic_music,Music,Pop_music         3544
4                                Music,Music_of_Asia          169
..                                                 …          …
441                              Music,Music_of_Asia           28
442                              Music,Music_of_Asia          422
443                    Music,Music_of_Asia,Pop_music          207
444                Independent_music,Music,Pop_music          189
445                                 Music,Rock_music           46

     viewCount  likeCount  commentCount
0     85528644  1065658.0       40424.0
1     23563068   256228.0           0.0
2       307190    12256.0           0.0
3      9992568    42457.0        1353.0
4     68936472   674332.0       32943.0
..         …          …             …
441   37678452   588306.0           NaN
442    2100517    17905.0          98.0
443    2675170    18036.0         270.0
444   21199099   471826.0        9206.0
445     169731    10851.0         576.0

[446 rows x 13 columns]
```

Cleaning YouTube_Music_Transcript_df

[6]:
```python
# Reading the CSV file into a DataFrame
YouTube_Music_Transcript_df = pd.read_csv('YouTube_Music_Transcript.csv')
```

```
# Printing the DataFrame
print(YouTube_Music_Transcript_df)
```

```
          Video ID                                         Text
0      suAR1PYFNYA                                      [Music]
1      suAR1PYFNYA     okay I come and I go tell me all the
2      suAR1PYFNYA   ways you need me I'm not here for long
3      suAR1PYFNYA     catch me or I go who I come and I go
4      suAR1PYFNYA     prove you got the right to please me
…              …                                          …
56707  dviOJyEqO24  hit subscribe or follow and let me know
56708  dviOJyEqO24               if you can hear a difference
56709  dviOJyEqO24                                         good
56710  dviOJyEqO24                                      [Music]
56711  dviOJyEqO24                                    thank you

[56712 rows x 2 columns]
```

It seems that the video ID's are mentioned in multiple rows of the `Video ID` column. Besides this issue, the following needs to be executed in order to clean and preprocess the DataFrame for the transcripts of YouTube music videos:

1. **Removing Empty or Whitespace Rows**: Rows with empty or whitespace-only values in the 'Text' column are filtered out for each unique 'Video ID'.
2. **Removing Duplicates**: Duplicate rows based on the combination of 'Video ID' and 'Text' columns are dropped, ensuring unique transcript entries for each video.
3. **Resetting Index**: The index of the DataFrame is reset after removing rows to maintain a clean sequential index.
4. **Text Cleaning**: The string '[Music]' is removed from the 'Text' column using regular expressions.
5. **Handling NaN Values**: Any NaN values in the 'Text' column are replaced with empty strings.
6. **Removing Empty Text Entries**: Rows with empty 'Text' values are removed to ensure only meaningful transcript data remains.
7. **Column Renaming**: The 'Video ID' column is renamed to 'videoId', and the 'Text' column is renamed to 'transcript'.
8. **Grouping and Aggregating**: The DataFrame is grouped by 'videoId', and the text values for each video are concatenated together.

By addressing these cleaning steps, the dataset will be better prepared for analysis and modeling, leading to more accurate and reliable results.

```
[7]: # Removing rows where 'Text' is empty or contains only whitespace for each␣
     ↪unique 'Video ID'
```

```
Cleaned_YouTube_Transcript_df =␣
↪YouTube_Music_Transcript_df[YouTube_Music_Transcript_df['Text'].str.strip() !
↪= '']

# Removing duplicates based on the 'Video ID' and 'Text' columns
Cleaned_YouTube_Transcript_df = Cleaned_YouTube_Transcript_df.
↪drop_duplicates(subset=['Video ID', 'Text'])

# Resetting the index after removing rows
Cleaned_YouTube_Transcript_df.reset_index(drop=True, inplace=True)

# Removing the string '[Music]' from the 'Text' column
Cleaned_YouTube_Transcript_df['Text'] = Cleaned_YouTube_Transcript_df['Text'].
↪str.replace(r'\[Music\]', '', regex=True)

# Converting NaN values in 'Text' column to empty strings
Cleaned_YouTube_Transcript_df['Text'] = Cleaned_YouTube_Transcript_df['Text'].
↪fillna('')

# Removing rows where 'Text' column is empty
Cleaned_YouTube_Transcript_df =␣
↪Cleaned_YouTube_Transcript_df[Cleaned_YouTube_Transcript_df['Text'] != '']

# Renaming the 'Video ID' column to 'videoId'
Cleaned_YouTube_Transcript_df.rename(columns={'Video ID': 'videoId', 'Text':␣
↪'transcript'}, inplace=True)

# Grouping by 'videoID' and concatenate the text values
Transcript_Grouped_df = Cleaned_YouTube_Transcript_df.
↪groupby('videoId')['transcript'].agg(' '.join).reset_index()

# Removing the string '[Applause]' from the 'transcript' column
Transcript_Grouped_df['transcript'] = Transcript_Grouped_df['transcript'].str.
↪replace(r'\[Applause\]', '', regex=True)

# Printing the updated DataFrame
print(Transcript_Grouped_df)
```

```
         videoId                                         transcript
0    -QCVMtrDz1w  Got Away you're driving me crazy I need I need…
1    -RQTxqPc5TO  Tears scattered like tissue paper\nFrightened …
2    -_MuvBNSi8I  yes I be woman yes I be baby I be whatever tel…
3    -fxZOzAjKHo  Well, we were sitting on\ntruck bed when it st…
4    -wp98L1wAPY   loveing for fore foree thee spee speee spe sp…
..           …                                                …
194  y6Cz5zcvtZg  hello I know you understand I can't leave you …
195  yd8SC9pSId0  foreign Oh I wanna know I close my eyes  thank…
```

```
196  ysNDDrG9PtI  all smiles don't know what it takes to fool th…
197  z04ogqm-WeE  foreign now in foreign come on my mind Jessica…
198  zStYh2eH0Wk  here's to the ones that we got due to the wish…

[199 rows x 2 columns]
```

Besides cleaning the columns, the transcript is also cleaned with a function called `preprocess_text` that takes a text input and applies several preprocessing steps to it:

1. **Remove punctuations and special characters**: It uses a regular expression to substitute any non-alphanumeric characters (`[^\w\s]`) with an empty string, effectively removing them from the text.
2. **Convert to lowercase**: It converts all the text to lowercase letters.
3. **Remove digits**: It removes any digits from the text by substituting them with an empty string (`''`).
4. **Remove extra whitespace**: It replaces multiple consecutive whitespace characters with a single space (`' '`) and then removes any leading or trailing whitespace using the `strip()` method.

This means that each entry in the 'transcript' column of `Transcript_Grouped_df` will have these preprocessing steps applied to it, resulting in a cleaned version of the text data.

```
[8]: def preprocess_text(text):
         # Remove punctuations and special characters
         text = re.sub(r'[^\w\s]', '', text)
         # Convert to lowercase
         text = text.lower()
         # Remove digits
         text = re.sub(r'\d+', '', text)
         # Remove extra whitespace
         text = re.sub(r'\s+', ' ', text).strip()  # Strip removes leading and␣
      ↪trailing whitespace
         return text

     # Apply preprocessing to Transcript column
     Transcript_Grouped_df['transcript'] = Transcript_Grouped_df['transcript'].
      ↪apply(preprocess_text)

     # Printing the results of the transcript column
     print(Transcript_Grouped_df['transcript'])
```

```
0      got away youre driving me crazy i need i need …
1      tears scattered like tissue paper frightened b…
2      yes i be woman yes i be baby i be whatever tel…
3      well we were sitting on truck bed when it star…
4      loveing for fore foree thee spee speee spe spe…
                             …
194    hello i know you understand i cant leave you a…
195    foreign oh i wanna know i close my eyes thank …
```

```
196    all smiles dont know what it takes to fool thi…
197    foreign now in foreign come on my mind jessica…
198    heres to the ones that we got due to the wish …
Name: transcript, Length: 199, dtype: object
```

Merging DataFrames To merge these datasets effectively, the `videoId` must be used as the common key, as the transcripts in the `Cleaned_YouTube_Music_df` correspond to specific videos. However, it's important to note that the transcript dataset only covers entries up to 199, whereas the `Cleaned_YouTube_Music_df` consists of 446. Consequently, any videos beyond this range will not have corresponding transcripts and will be dropped from the merged dataset.

```
[9]: # Printing the columns of Transcript_Grouped_df to verify the column names
print("Columns in Transcript_Grouped_df:", Transcript_Grouped_df.columns)

# Printing the columns of Cleaned_Youtube_Music_df
print("Columns in Cleaned_Youtube_Music_df:", Cleaned_Youtube_Music_df.columns)
```

```
Columns in Transcript_Grouped_df: Index(['videoId', 'transcript'],
dtype='object')
Columns in Cleaned_Youtube_Music_df: Index(['channelId', 'channelTitle',
'videoId', 'publishedAt', 'videoTitle',
       'videoDescription', 'tags', 'videoCategoryLabel', 'topicCategories',
       'durationSec', 'viewCount', 'likeCount', 'commentCount'],
      dtype='object')
```

```
[10]: # Merging the datasets on 'videoId' using an inner join
merged_df = pd.merge(Cleaned_Youtube_Music_df, Transcript_Grouped_df,␣
 ↪on='videoId', how='inner')

# Printing the shape of the merged DataFrame before dropping rows
print("Shape of merged DataFrame before dropping rows:", merged_df.shape)
```

```
Shape of merged DataFrame before dropping rows: (199, 14)
```

```
[11]: # Printing the merged DataFrame to inspect its contents
print(merged_df)
```

```
                        channelId  channelTitle      videoId  
0     channel_1810432585761117807      Dua Lipa  suAR1PYFNYA  \
1    channel_-8679121005285211262    Rich Amiri  Q7N99EzNQrw
2    channel_-8859766390484302968          Rema  dNt1QR1ecuM
3    channel_-6136403466541997464      Lil Mabu  fW1QcEAy4rg
4     channel_5834049654312531833     KEEMOKAZI  Q3KNV63Viw8
..                            …             …            …
194  channel_-3562024370786368633  Charlie Puth  PAKFzFqJa58
195   channel_4357923905480611291     Opm Songs  J4z5sVpmnzU
196   channel_2495716982820987257        MadmiX  IS41HLeW8-E
197  channel_-1022667421863012121       DJ BOAT  sOUmdrNwEy8
```

```
198  channel_-4543590347096294591   Danny Sapko  dvi0JyEq024

             publishedAt                                         videoTitle
0    2023-11-09T23:00:11Z         Dua Lipa - Houdini (Official Music Video)  \
1    2023-11-08T21:00:09Z      Rich Amiri - One Call (Official Music Video)
2    2023-05-22T15:59:26Z               Rema - Charm (Official Music Video)
3    2023-12-21T21:00:06Z  Lil Mabu x Fivio Foreign - TEACH ME HOW TO DRI…
4    2023-06-23T18:57:07Z                               Keemokazi - Walk In
..                    …                                                …
194  2023-03-31T04:00:02Z  CHARLIE PUTH - THAT'S NOT HOW THIS WORKS (FEAT…
195  2023-06-29T07:11:21Z  Mga Lumang Tugtugin 60s 70s 80s 90s - Tagalog …
196  2023-05-12T12:32:00Z                           NEON BLADE (Super Slowed)
197  2023-04-07T16:00:09Z  AFROBEATS 2024 Video Mix |AFROBEAT 2024 PARTY …
198  2023-02-20T23:37:59Z  £4 Guitar Lead vs. £114 Guitar Lead #bass #bas…

                                      videoDescription
0    Listen to Houdini:  Subscribe to the Dua Lipa …  \
1    Rich Amiri - One Call (Official Music Video) S…
2    Rema - Charm (Official Music Video) Stream RAV…
3    i completed my 1st drill with my friend Fivio!…
4    "Walk In" is Keemokazi’s newest release. Avail…
..                                                 …
194  "That’s Not How This Works (feat. Dan + Shay)"…
195  Mga Lumang Tugtugin 60s 70s 80s 90s - Tagalog …
196  In this video you'll experience the bass boost…
197  DOWNLOAD EMAIL SIGN UP:  PLAYLIST  Listen to …
198  Please support my channel by becoming my Patro…

                                                tags videoCategoryLabel
0    dua lipa,dua,new rules,dua leepa,idgaf,scared …              Music  \
1                                                NaN              Music
2                                                NaN              Music
3    Lil Mabu - TEACH ME HOW TO DRILL (Official Mus…              Music
4                                                NaN      Entertainment
..                                               …                  …
194  charlie puth,charlie,puth,voicenotes,attention…              Music
195  opm love songs,OPM Classics Medley,OPM 80s,OPM…     People & Blogs
196                                              NaN              Music
197  afrobeats,afrobeat,naija,#afrobeat2023,afrobea…              Music
198  danny sapko,bass,funk bass,bass solo,free mp3 …              Music

                               topicCategories  durationSec
0                              Music,Pop_music          189  \
1                          Hip_hop_music,Music          127
2                Hip_hop_music,Music,Pop_music          202
3                          Hip_hop_music,Music          300
4                          Hip_hop_music,Music          132
..                                           …           …
```

```
194                                    Music,Pop_music          211
195                                    Music,Pop_music         2810
196                 Electronic_music,Hip_hop_music,Music         93
197  Electronic_music,Hip_hop_music,Music,Pop_music…        3531
198                                   Music,Rock_music           46

     viewCount  likeCount  commentCount
0     85528644  1065658.0       40424.0  \
1      6332675   171333.0        3094.0
2     69642888   567770.0       10569.0
3     24361712   645175.0       31202.0
4     10834474   135746.0        9582.0
..         …         …           …
194   17362226   363558.0        7600.0
195    3200472    12274.0         309.0
196    2650624    25853.0         342.0
197    3413040    24625.0         471.0
198     169731    10851.0         576.0

                                          transcript
0    okay i come and i go tell me all the ways you …
1    i dont trust soul i dont trust nobody do by so…
2    foreign baby baby everybody please you better …
3    yea __ we gon pop out we gon go to the hood yo…
4    im thinking like we play fortnite and just go …
..                                                 …
194  i said what i love you for the day it disappea…
195  welcome to opm songs where you can find the la…
196                                   hahaha foreign
197  thank you both in the building foreign laughs …
198  this lead costs four pounds and this lead cost…

[199 rows x 14 columns]
```

[12]:
```python
# Displaying basic information about DataFrame shape and data types
print("Shape of merged_df:", merged_df.shape, "| Length:", len(merged_df))
print("Data types in merged_df:")
print(merged_df.dtypes)
print()

# Checking for missing values
print("Missing values in merged_df:")
print(merged_df.isnull().sum())
print()
```

```
Shape of merged_df: (199, 14) | Length: 199
Data types in merged_df:
channelId               object
```

```
channelTitle           object
videoId                object
publishedAt            object
videoTitle             object
videoDescription       object
tags                   object
videoCategoryLabel     object
topicCategories        object
durationSec             int64
viewCount               int64
likeCount             float64
commentCount          float64
transcript             object
dtype: object

Missing values in merged_df:
channelId              0
channelTitle           0
videoId                0
publishedAt            0
videoTitle             0
videoDescription       8
tags                  53
videoCategoryLabel     0
topicCategories        0
durationSec            0
viewCount              0
likeCount              5
commentCount           1
transcript             0
dtype: int64
```

After merging the datasets, the resulting DataFrame, `merged_df`, contains 199 rows and 14 columns. Here's a breakdown of the findings:

Data Types: - Most columns are of type `object`, indicating textual data. - Numeric columns (`durationSec`, `viewCount`, `likeCount`, `commentCount`) have appropriate integer or float data types.

Missing Values: - The `videoDescription` column has 8 missing values. - 53 missing values are found in the `tags` column. - There are 5 missing values in the `likeCount` column. - One missing value is present in the `commentCount` column. - All other columns have no missing values.

The merged DataFrame seems to be mostly intact, with a few missing values primarily in the `videoDescription`, `tags`, `likeCount`, and `commentCount` columns. Therefore, the following steps are executed in order to clean the data and copying it to a new DataFrame:

1. **Drop Rows with Missing Values**:
   - It removes rows containing any missing values using the `dropna()` function.
   - The resulting DataFrame is assigned to `cleaned_merged_df`.

2. **Convert Data Types**:
   - The ‘durationSec’ column is converted to an integer data type using `astype(int)` to ensure consistency in data representation.
   - The ‘likeCount’ and ‘commentCount’ columns are converted to integer data type using `astype(int)`.
   - Any missing values in ‘likeCount’ and ‘commentCount’ are filled with 0 using `fillna(0)`.
3. **Convert ‘publishedAt’ to Datetime Data Type**:
   - The ‘publishedAt’ column is converted to datetime data type using `pd.to_datetime()` to ensure it is represented as a datetime object.
4. **Display Updated Shape and Missing Values Information**:
   - It prints the shape of the cleaned DataFrame (`cleaned_merged_df`) after removing missing values.
   - It also prints the number of missing values in each column of the cleaned DataFrame using `isnull().sum()`.

[13]:
```
# Dropping rows with missing values
cleaned_merged_df = merged_df.dropna().copy()

# Converting 'durationSec' column to integer data type
cleaned_merged_df['durationSec'] = cleaned_merged_df['durationSec'].astype(int)

# Converting 'likeCount' and 'commentCount' columns to integer data type and␣
↪fill missing values with 0
cleaned_merged_df['likeCount'] = cleaned_merged_df['likeCount'].fillna(0).
↪astype(int)
cleaned_merged_df['commentCount'] = cleaned_merged_df['commentCount'].fillna(0).
↪astype(int)

# Converting 'publishedAt' column to datetime data type
cleaned_merged_df['publishedAt'] = pd.
↪to_datetime(cleaned_merged_df['publishedAt'])

# Printing data types
print("Data types in cleaned_merged_df:")
print(cleaned_merged_df.dtypes)

# Displaying the updated shape and missing values information
print("Shape of cleaned_merged_df after removing missing values:",␣
↪cleaned_merged_df.shape, "| Length:", len(cleaned_merged_df))
print("Missing values in cleaned_merged_df after removing missing values:")
print(cleaned_merged_df.isnull().sum())
```

```
Data types in cleaned_merged_df:
channelId                              object
channelTitle                           object
videoId                                object
publishedAt                datetime64[ns, UTC]
```

```
videoTitle                          object
videoDescription                    object
tags                                object
videoCategoryLabel                  object
topicCategories                     object
durationSec                          int64
viewCount                            int64
likeCount                            int64
commentCount                         int64
transcript                          object
dtype: object
Shape of cleaned_merged_df after removing missing values: (140, 14) | Length:
140
Missing values in cleaned_merged_df after removing missing values:
channelId             0
channelTitle          0
videoId               0
publishedAt           0
videoTitle            0
videoDescription      0
tags                  0
videoCategoryLabel    0
topicCategories       0
durationSec           0
viewCount             0
likeCount             0
commentCount          0
transcript            0
dtype: int64
```

1.4 Data Exploration & Evaluation

In this stage, *after the dataset is considered clean*, you are expected to show to the reader how the final dataset looks like, and perform an exploratory review of the data. At all times, you are expected to write a report-out to stakeholders (i.e., not only show the data, but also explain what the data show).

This includes: * Descriptives and definitions of all key variables in a clear manner * Univariate visualizations for key variables * Visualization of key bivariate relationships (e.g., related to hypotheses or RQs) * Checking the data for biases and unbalance (e.g., unequal distributed variables, missing cases) * Writing a report-out to stakeholders summarising the findings of the data exploration (including what the data already show when it comes to the RQ & hypothesis, and potential risks of bias and unbalance)

Key variables are all variables that are used as DV's or IV's in your hypotheses or RQs.

IMPORTANT: * You only need to do the exploration for the variables that are *relevant* to your study. You can skip variables/columns in the dataset that are absolutely

irrelevant to your study. * You need to communicate your steps clearly to the reader. This means that you should not just do a *df.describe()* to communicate descriptive statistics in a dataset that contains a lot of irrelevant columns and expect the reader to figure out by her or himself what is relevant. You should instead only show to the reader what is relevant, and explain why.

1.4.1 Descriptives and definitions of all key variables in a clear manner

Besides the descriptives and definitions of all the key variables, this section shall also present the results of the intercoder reliability test in order to observe whether the quality of the data is validated. In order to do so, I have manually coded the complexity scores and these are compared.

Since all missing values are removed, the dataset consists of 140 music videos with transcripts, resulting in the following table:

Variable Name	Definition
channelId	Anonymized identifier for the publisher of the video
channelTitle	The title of the published video
videoId	Identifier for the YouTube video
videoTitle	The title of the YouTube video
publishedAt	Date and time the video was published on YouTube
videoDescription	The description of the video
tags	Tags assigned by the publisher to the video
videoCategoryLabel	Classification tag of the video based on the content
topicCategories	The genre of music associated with the video
durationSec	The duration of the YouTube video
viewCount	The number of views the video has
likeCount	The number of users who have liked the video
commentCount	The number of comments the video has
transcript	The raw audio transcript from the video
fkGradeLevel	The Flesch-Kincaid Grade Level measures the readability of text by assigning a grade level required to understand the content.
gunningFogIndex	The Gunning Fog Index estimates the years of formal education needed to understand a piece of text. It measures the readability of text, with a higher index indicating more complex language.

In this section the `fkGradeLevel` and `gunningFogIndex` variables shall be created and used in order to address the complexity of language in the YouTube music videos (Khawaja et al., 2010).

Before this is executed, the following descriptive statistics are explored in order to determine whether there are biases or unbalances: `durationSec`, `viewCount`, `likeCount` and `commentCount`:

[14]:
```python
# Displaying descriptive statistics
print("Descriptive statistics for merged_df:")
print(cleaned_merged_df.describe())
```

```
Descriptive statistics for merged_df:
       durationSec      viewCount     likeCount  commentCount
```

```
count    140.00000  1.400000e+02  1.400000e+02  1.400000e+02
mean     672.00000  1.799597e+07  2.479739e+05  1.530436e+04
std     1007.79146  4.618566e+07  9.188088e+05  1.154726e+05
min       10.00000  1.611640e+05  2.711000e+03  0.000000e+00
25%      171.75000  2.588058e+06  2.021925e+04  4.040000e+02
50%      216.50000  6.225412e+06  5.728550e+04  1.263500e+03
75%      532.25000  1.580312e+07  2.011018e+05  4.937750e+03
max     3582.00000  4.714133e+08  1.047807e+07  1.364324e+06
```

Based on these descriptives, there are no obvious observations in the duration of videos, the variability in view counts, like counts, and comment counts that suggests potential imbalances in the popularity and engagement levels of the videos. What can be deduced is the following:

1. **durationSec (Duration of the Videos):**
   - The mean duration of videos is 672 seconds (about 11 minutes), with a standard deviation of 1007.79 seconds, indicating a wide range in video lengths.
   - The shortest video is 10 seconds long, while the longest is 3582 seconds (about 60 minutes).
   - There are no obvious biases or imbalances in video duration based on these statistics.
2. **viewCount (Number of Views):**
   - The mean number of views is approximately 17,995,970, with a large standard deviation of about 46,185,660, indicating significant variability in the number of views across videos.
   - The minimum number of views is 161,164, while the maximum is 471,413,300, suggesting a wide range in the popularity of videos.
   - There might be some biases or imbalances in view counts, especially considering the large variability and the potential influence of outliers.
3. **likeCount (Number of Likes):**
   - The mean number of likes is approximately 247,974, with a standard deviation of about 918,808, indicating considerable variability in the number of likes.
   - The minimum number of likes is 2,711, while the maximum is 10,478,070, suggesting a wide range in the popularity of videos based on likes.
   - Similar to view counts, there might be biases or imbalances in like counts, given the large variability and potential influence of outliers.
4. **commentCount (Number of Comments):**
   - The mean number of comments is approximately 15,304, with a standard deviation of about 115,473, indicating considerable variability in the number of comments.
   - The minimum number of comments is 0, while the maximum is 1,364,324, suggesting a wide range in user engagement through comments.
   - There might be biases or imbalances in comment counts, especially considering the large variability and potential influence of outliers.

Exploring Language Complexity In order to examine the complexity of language in the music videos presented in YouTube, the `textstat` package is implemented. This particular package is a Python library designed to provide various readability metrics for text (textstat). To utilize this package affectively, the Flesch-Kincaid Grade Level and the Gunning Fog Index are used.

The Flesch-Kincaid Grade Level and the Gunning Fog Index are readability metrics for written text. The Flesch-Kincaid Grade Level estimates the grade level a reader needs to understand the text, with lower scores indicating simpler text and higher scores indicating more complex text (Khawaja et al., 2010).

The Gunning Fog Index estimates the years of formal education required to understand the text, with lower scores indicating easier-to-understand text and higher scores indicating more complex text, requiring higher education levels (Khawaja et al., 2010).

Based on this package, the following is executed:

1. **Importing textstat:**
   - `import textstat` imports the `textstat` module, making its functions and capabilities available for use in the current Python environment.
2. **Calculating the Flesch-Kincaid Grade Level:**
   - `cleaned_merged_df['flesch_kincaid_grade'] = cleaned_merged_df['transcript'].apply(laml x: textstat.text_standard(x, float_output=True))`
   - This line computes the Flesch-Kincaid Grade Level for each transcript in the DataFrame `cleaned_merged_df`. It uses the `textstat.text_standard()` function, which returns the Flesch-Kincaid Grade Level of a given text.
   - The `apply()` method is used to apply this function to each element in the 'transcript' column of the DataFrame.
3. **Calculating the Gunning Fog Index:**
   - `cleaned_merged_df['gunning_fog_index'] = cleaned_merged_df['transcript'].apply(lambda x: textstat.gunning_fog(x))`
   - Similar to the previous line, this calculates the Gunning Fog Index for each transcript in the DataFrame.
   - The `textstat.gunning_fog()` function computes the Gunning Fog Index of a text.
4. **Displaying the results:**
   - `print(cleaned_merged_df[['videoId', 'flesch_kincaid_grade', 'gunning_fog_index']])`
   - This prints out a subset of the DataFrame `cleaned_merged_df`, specifically the 'videoId', 'flesch_kincaid_grade', and 'gunning_fog_index' columns, which contain the computed readability metrics for each transcript.

[15]:
```
# Installing the textstat package using pip
!pip install textstat
```

```
Requirement already satisfied: textstat in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (0.7.3)
Requirement already satisfied: pyphen in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
textstat) (0.14.0)
```

Additionally, 'fkGradeLevel' and 'gunningFogIndex' are added as two new columns to the DataFrame 'cleaned_merged_df', containing values from existing columns 'flesch_kincaid_grade' and 'gunning_fog_index', respectively. These columns represent the Flesch-Kincaid Grade Level and Gunning Fog Index, which measure the complexity of language in the transcript.

[16]:
```
# Importing the textstat package
import textstat
```

```
# Calculate the Flesch-Kincaid Grade Level
cleaned_merged_df['fkGradeLevel'] = cleaned_merged_df['transcript'].
 ↪apply(lambda x: textstat.text_standard(x, float_output=True))

# Calculate the Gunning Fog Index
cleaned_merged_df['gunningFogIndex'] = cleaned_merged_df['transcript'].
 ↪apply(lambda x: textstat.gunning_fog(x))

# Adding 'fkGradeLevel' column for Flesch-Kincaid Grade Level
cleaned_merged_df['fkGradeLevel'] = cleaned_merged_df['fkGradeLevel']

# Adding 'gunningFogIndex' column for Gunning Fog Index
cleaned_merged_df['gunningFogIndex'] = cleaned_merged_df['gunningFogIndex']

# Printing the results with head
print(cleaned_merged_df[['videoId', 'fkGradeLevel', 'gunningFogIndex']].head())
```

```
       videoId  fkGradeLevel  gunningFogIndex
0  suAR1PYFNYA          97.0           101.60
3  fW1QcEAy4rg         236.0           244.40
5  YudHcBIxlYw          56.0            59.89
6  3tDPeqLBbbc           8.0            16.40
7  ZiahkY4snMw           0.0           566.31
```

Intercoder Reliability Test In order to examine the validity of this particular method, an intercoder reliability test is executed. Therefore, 50 videos are randomly exported to a CSV file.

```
[18]: # Randomly selecting 50 videos
random_50_videos = cleaned_merged_df.sample(n=50, random_state=42)

# Exporting the selected videos to a CSV file in the current working directory
current_dir = "/Users/helgegeurtjacobusmoes/Desktop/Digital Analytics/
 ↪AB_Test_HelgeMoes"
file_path = os.path.join(current_dir, 'ICR_50_videos.csv')
random_50_videos.to_csv(file_path, index=False)
```

Since the language complexity score is a numerical decimal value, it will remain ungrouped for analysis to maintain precision. Given the challenge of uniformly assigning numerical values to each transcript, a binary categorization will be adopted. Videos demonstrating high language complexity in terms of vocabulary and structure will be labeled as ‘1’, while those with low complexity will be labeled as ‘0’. For the intercoder reliability assessment, this automated labeling will also be treated as a categorical classification. While this approach may not offer complete validation of the method, other alternatives are outside the scope of this study. For the Gunning Fog Index, I placed a ‘0’ when a score was =>10 to determine low complexity. This might have had an effect on the complexity score.

```
[19]: # Reading the Excel file that has been coded into a DataFrame
      icr_50_df = pd.read_excel("ICR_50_videos_edited.xlsx")

      # Displaying the loaded data
      print(icr_50_df.head())
```

```
                          channelId            channelTitle      videoId
0   channel_6817465443151391869              Chill Vibes  dxq9iiD0Dt0  \
1   channel_9181255827052430869            Toons + Tunes  AIZ0wTd-mlQ
2  channel_-6516560516934423576               DonMoenTV  oAlXZ6aVf2U
3   channel_6421540313296267954  All My Favourite Songs  EN1bBi4ZL3Y
4   channel_3221920119430507991          Creative Chaos  -yIz7TFqgR4

                 publishedAt
0  2023-03-10 11:00:35+00:00  \
1  2023-05-06 15:00:07+00:00
2  2023-05-15 11:00:39+00:00
3  2023-06-23 22:39:32+00:00
4  2023-04-18 14:00:01+00:00


                                         videoTitle
0  Late Night Vibes   Late night chill vibes play…  \
1  Xyle Trilogy: Extended Full Version (FGTeeV An…
2  Best Christian Songs 2023 Non Stop Worship Mus…
3  Nicki Minaj & Ice Spice - Barbie World - Lyric…
4                Camila Cabello - Shameless (Lyrics)

                                   videoDescription
0  Late Night Vibes   Late night chill vibes play…  \
1  This is the moment everyone has been waiting f…
2  Best Christian Songs 2023 Non Stop Worship Mus…
3  #barbieworld #nickiminaj #icespice #barbie #ly…
4   Check out my Spotify playlist:  Viral TikTok…

                                               tags videoCategoryLabel
0  pop,pop songs,english songs,acoustic songs,chi…              Music  \
1  music videos,fgteev book,fgteev books,saves th…   Film & Animation
2  goodness of god,christian songs,christian song…              Music
3  Barbie World,barbie world,barbie world lyrics,…              Music
4  camila cabello shameless,camila cabello shamel…              Music

                    topicCategories  durationSec  viewCount  likeCount
0  Electronic_music,Music,Pop_music         3092     402863       2711  \
1    Entertainment,Film,Hip_hop_music         1176    2578103      18541
2              Christian_music,Music          135    3628844      22861
3                Hip_hop_music,Music          110    1636769      15796
4                    Music,Pop_music          221    7053188      33098
```

```
   commentCount                                         transcript
0           144  foreign applause i wonder if you found what yo…  \
1          1553  hey man enough with the strong arm bruh i came…
2          1252  foreign ive been held in your hands from the m…
3           222                all of them house stop its giving me
4           249  foreign im jealous i need your money they go b…

   fkGradeLevel  gunningFogIndex  InterCRGL  InterCRFI
0        3267.0          3352.14          1          1
1           0.0           805.38          0          1
2        1700.0          1745.12          1          1
3           1.0             3.20          1          0
4           8.0             7.20          1          0
```

Additionally, the median values are calculated of the 'gunningFogIndex' column and categorizes transcripts as high or low complexity based on this median, adding this classification to two new columns named 'complexityFI' and 'complexityGL'. Finally, it displays the resulting DataFrame with the added complexity columns.

```
[20]: # Loading the data from 'ICR_50_videos_edited.xlsx'
icr_50_df = pd.read_excel('ICR_50_videos_edited.xlsx')

# Calculating the median value of gunningFogIndex
median_gunning_fog = icr_50_df['gunningFogIndex'].median()

# Categorizing the transcripts as high or low complexity based on the median
icr_50_df['complexityFI'] = [1 if x >= median_gunning_fog else 0 for x in␣
 ↪icr_50_df['gunningFogIndex']]

# Calculating the median value of fkGradeLevel
median_fk_grade = icr_50_df['fkGradeLevel'].median()

# Categorizing the transcripts as high or low complexity based on the median
icr_50_df['complexityGL'] = [1 if x >= median_fk_grade else 0 for x in␣
 ↪icr_50_df['fkGradeLevel']]

# Displaying the resulting dataframe
print(icr_50_df.head())
```

```
                    channelId             channelTitle      videoId
0   channel_6817465443151391869               Chill Vibes  dxq9iiD0Dt0  \
1   channel_9181255827052430869              Toons + Tunes  AIZ0wTd-mlQ
2  channel_-6516560516934423576                 DonMoenTV  oA1XZ6aVf2U
3   channel_6421540313296267954  All My Favourite Songs  EN1bBi4ZL3Y
4   channel_3221920119430507991            Creative Chaos  -yIz7TFqgR4

                publishedAt
```

```
0  2023-03-10 11:00:35+00:00  \
1  2023-05-06 15:00:07+00:00
2  2023-05-15 11:00:39+00:00
3  2023-06-23 22:39:32+00:00
4  2023-04-18 14:00:01+00:00

                                          videoTitle
0  Late Night Vibes  Late night chill vibes play…  \
1  Xyle Trilogy: Extended Full Version (FGTeeV An…
2  Best Christian Songs 2023 Non Stop Worship Mus…
3  Nicki Minaj & Ice Spice - Barbie World - Lyric…
4               Camila Cabello - Shameless (Lyrics)

                                    videoDescription
0  Late Night Vibes  Late night chill vibes play…  \
1  This is the moment everyone has been waiting f…
2  Best Christian Songs 2023 Non Stop Worship Mus…
3  #barbieworld #nickiminaj #icespice #barbie #ly…
4   Check out my Spotify playlist:  Viral TikTok…

                                                tags videoCategoryLabel
0  pop,pop songs,english songs,acoustic songs,chi…             Music  \
1  music videos,fgteev book,fgteev books,saves th…   Film & Animation
2  goodness of god,christian songs,christian song…             Music
3  Barbie World,barbie world,barbie world lyrics,…             Music
4  camila cabello shameless,camila cabello shamel…             Music

                   topicCategories  durationSec  viewCount  likeCount
0  Electronic_music,Music,Pop_music         3092     402863       2711  \
1  Entertainment,Film,Hip_hop_music         1176    2578103      18541
2            Christian_music,Music           135    3628844      22861
3              Hip_hop_music,Music           110    1636769      15796
4                  Music,Pop_music           221    7053188      33098

   commentCount                                       transcript
0           144  foreign applause i wonder if you found what yo…  \
1          1553  hey man enough with the strong arm bruh i came…
2          1252  foreign ive been held in your hands from the m…
3           222              all of them house stop its giving me
4           249  foreign im jealous i need your money they go b…

   fkGradeLevel  gunningFogIndex  InterCRGL  InterCRFI  complexityFI
0        3267.0          3352.14          1          1             1  \
1           0.0           805.38          0          1             1
2        1700.0          1745.12          1          1             1
3           1.0             3.20          1          0             0
4           8.0             7.20          1          0             0
```

```
   complexityGL
0             1
1             0
2             1
3             0
4             0
```

[21]:
```
from sklearn.metrics import cohen_kappa_score

# Extracting the 'InterCRGL' and 'complexityGL' columns from icr_df
labelsGL = icr_50_df['InterCRGL']
complexityGL = icr_50_df['complexityGL']

# Extracting the 'InterCRFI' and 'complexityFI' columns from icr_50_df
labelsFI = icr_50_df['InterCRFI']
complexityFI = icr_50_df['complexityFI']

# Calculating Cohen's kappa coefficient for 'InterCRGL'
kappa_InterCRGL = cohen_kappa_score(labelsGL, complexityGL)

# Calculating Cohen's kappa coefficient for 'InterCRFI'
kappa_InterCRFI = cohen_kappa_score(labelsFI, complexityFI)

# Calculating Cohen's kappa coefficient for 'complexityGL'
kappa_GL = cohen_kappa_score(labelsGL, complexityGL)

# Calculating Cohen's kappa coefficient for 'complexityFI'
kappa_FI = cohen_kappa_score(labelsFI, complexityFI)

# Printing the results of the Cohen's kappa coefficients
print("Cohen's kappa coefficient for Flesch Kincaid Grade Level:", kappa_GL)
print("Cohen's kappa coefficient for Gunning Fog Index", kappa_FI)
```

```
Cohen's kappa coefficient for Flesch Kincaid Grade Level: 0.52
Cohen's kappa coefficient for Gunning Fog Index 0.43999999999999995
```

The Cohen's kappa coefficients measure the agreement between two raters on categorical data (McHugh, 2012). In this case, they are used to assess the intercoder reliability between 'InterCRGL' and 'complexityGL', and between 'InterCRFI' and 'complexityFI'.

The kappa value of 0.52 for 'InterCRGL' and 'complexityGL' suggests moderate agreement, indicating that there is some level of consensus between the two raters regarding the complexity categorization based on the Gunning Fog Index. Similarly, the kappa value of 0.44 for 'InterCRFI' and 'complexityFI' also suggests moderate agreement, indicating some level of consensus between the raters regarding the complexity categorization based on the Flesch Kincaid Grade Level. While there is some agreement between the raters in both cases, further investigation might be needed to improve agreement and ensure consistency in complexity categorization. However, seen the scope of this research project and the assignment for Digital Analytics, I chose to regard these results as a sufficient agreement.

1.4.2 Visualizations for key variables: engagement metrics

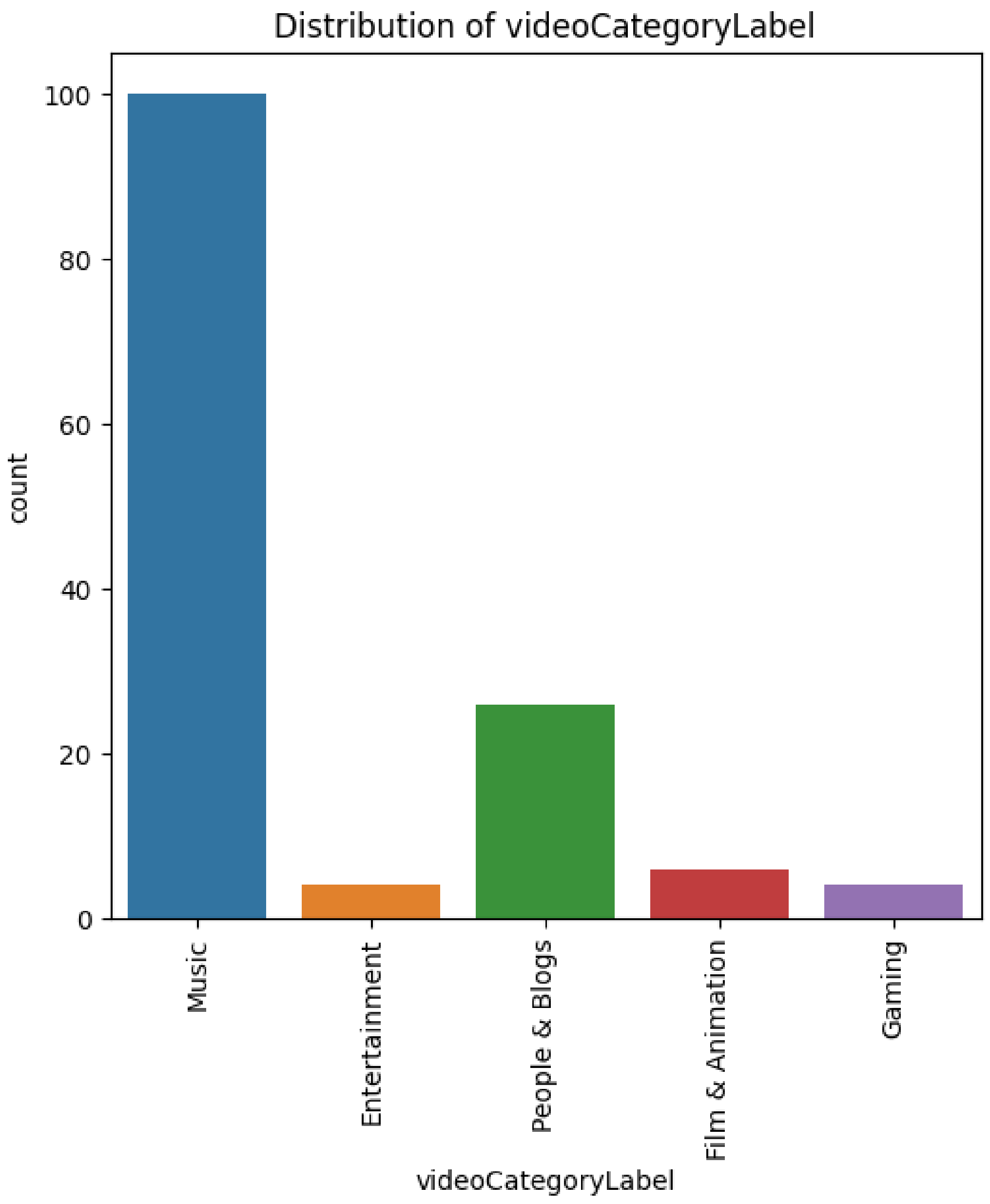
Based on the exploratory analysis of the dependent variables, there was an overrepresentation of right skewed histograms, which implies that there are relatively few data points with high values, causing the distribution to be pulled towards the right.

Moreover, all visualisations contained multiple outliers, which indicates that there are several data points that deviate significantly from the bulk of the data. The right skewness suggests that the data is not normally distributed and that the mean is greater than the median. Consequently, there are rare significant occurrences within the data that lead to these unusual high values. However, these outliers could represent errors in data collection.

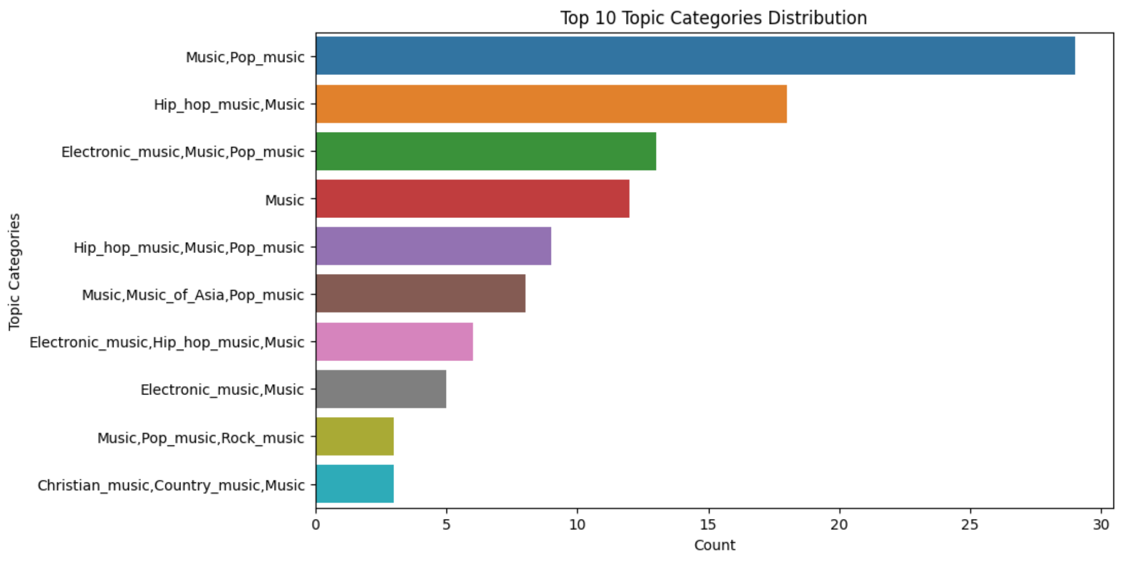
As an initial check of the music videos, the `videoCategoryLabel` and `topicCategories` are analyzed to observe what type of content is represented in this dataset.

[22]:
```python
# Importing the matplotlib library for creating plots
import matplotlib.pyplot as plt

# Importing the seaborn library for statistical data visualization
import seaborn as sns

# Creating a figure
fig, ax = plt.subplots(figsize=(6, 6))

# Plotting the distribution of unique values for 'videoCategoryLabel'
sns.countplot(x='videoCategoryLabel', data=cleaned_merged_df, ax=ax)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
ax.set_title('Distribution of videoCategoryLabel')

# Descriptive statistics for videoCategoryLabel
category_descriptives = cleaned_merged_df['videoCategoryLabel'].describe()
print("Descriptive Statistics for videoCategoryLabel:")
print(category_descriptives)

plt.show()
```

```
Descriptive Statistics for videoCategoryLabel:
count       140
unique        5
top       Music
freq        100
Name: videoCategoryLabel, dtype: object
```



[23]:
```
# Creating a figure and axes
plt.figure(figsize=(10, 6))

# Plotting the distribution of unique values for 'topicCategories'
sns.countplot(y='topicCategories', data=cleaned_merged_df, order =␣
 ↪cleaned_merged_df['topicCategories'].value_counts().index[:10])
plt.xlabel('Count')
plt.ylabel('Topic Categories')
plt.title('Top 10 Topic Categories Distribution')
plt.show()
```



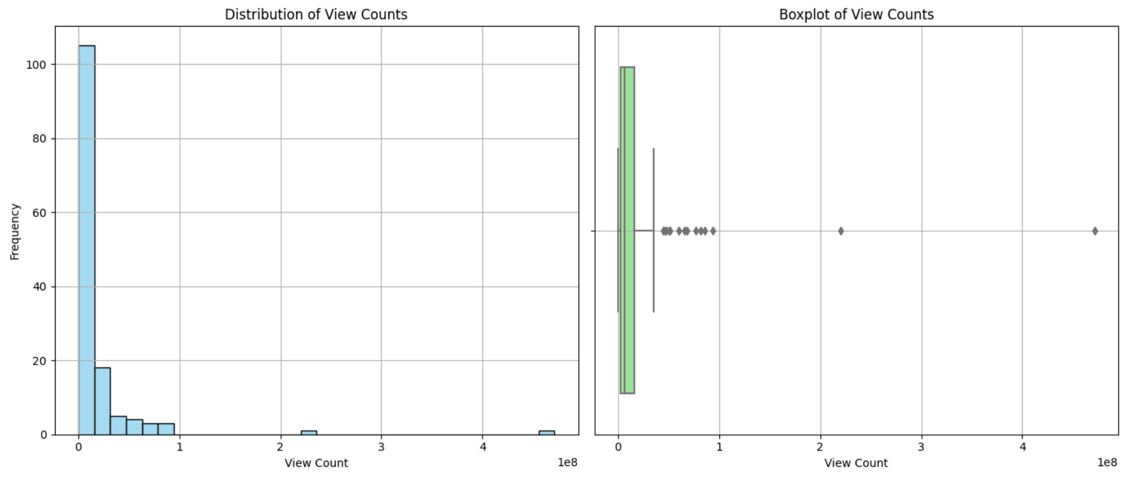
It comes as no surprise that 'Music' is the most dominant video category label, since it was scraped on the query 'Music'. Moreover, the most dominant genre are pop music, hip hop and electronic. These three music categories are considered to be the most broad, since Christian music holds a more niche audience.

Distribution of the 'viewCount' (DV)

[24]:
```
# Setting up the figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Plotting histogram for 'viewCount'
sns.histplot(cleaned_merged_df['viewCount'], bins=30, color='skyblue',␣
 ↪ax=axes[0])
axes[0].set_title('Distribution of View Counts')
axes[0].set_xlabel('View Count')
axes[0].set_ylabel('Frequency')
axes[0].grid(True)

# Plotting boxplot for 'viewCount'
sns.boxplot(x=cleaned_merged_df['viewCount'], ax=axes[1], color='lightgreen')
axes[1].set_title('Boxplot of View Counts')
axes[1].set_xlabel('View Count')
axes[1].grid(True)

# Displaying descriptive statistics
descriptive_stats = cleaned_merged_df['viewCount'].describe()
```

```
print("Descriptive Statistics for View Count:")
print(descriptive_stats)

plt.tight_layout()
plt.show()
```

```
Descriptive Statistics for View Count:
count    1.400000e+02
mean     1.799597e+07
std      4.618566e+07
min      1.611640e+05
25%      2.588058e+06
50%      6.225412e+06
75%      1.580312e+07
max      4.714133e+08
Name: viewCount, dtype: float64
```

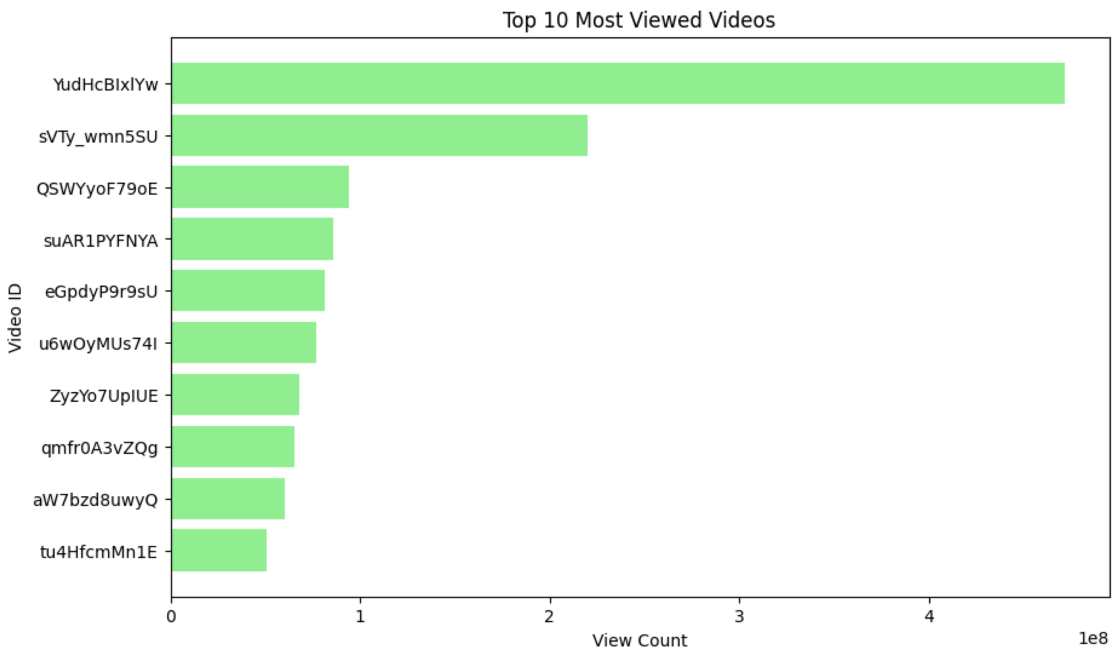


Based on the `viewCount`, the mean view count is approximately 17,995,970, and the standard deviation is approximately 46,185,660. This shows that most music videos have a low view count.

[25]:
```
# Finding the index of the row with the maximum viewCount
max_viewcount_index = cleaned_merged_df['viewCount'].idxmax()

# Retrieving the videoId of the video with the most viewCount
video_with_most_views = cleaned_merged_df.loc[max_viewcount_index, 'videoId']

# Printing the videoId and its corresponding viewCount
print("Video with the most viewCount:")
print("Video ID:", video_with_most_views)
print("View Count:", cleaned_merged_df['viewCount'].max())
```

```
# Sorting the DataFrame by viewCount to get the top viewed videos
top_viewed_videos = cleaned_merged_df.sort_values(by='viewCount',␣
 ↪ascending=False).head(10)

# Creating a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_viewed_videos['videoId'], top_viewed_videos['viewCount'],␣
 ↪color='lightgreen')
plt.xlabel('View Count')
plt.ylabel('Video ID')
plt.title('Top 10 Most Viewed Videos')
plt.gca().invert_yaxis()  # Invert y-axis to display video IDs from top to␣
 ↪bottom
plt.show()
```

```
Video with the most viewCount:
Video ID: YudHcBIxlYw
View Count: 471413299
```

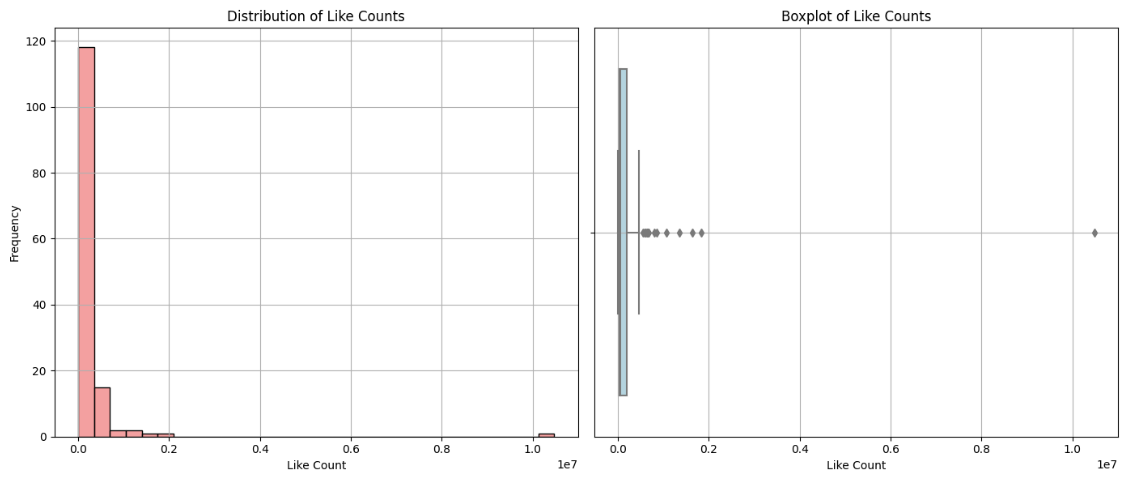


The results indicates that the video with the video ID “YudHcBIxlYw” has the highest view count, totaling 471,413,299 views. Upon closer inspection, the content consists of K-pop music, which contains a high language complexity that may be considered as difficult to understand in Western countries.

Distribution of the ‘likeCount’ (DV)

```
[26]: # Setting up the figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Plotting histogram for 'likeCount' with different color
sns.histplot(cleaned_merged_df['likeCount'], bins=30, color='lightcoral',␣
 ↪ax=axes[0])  # Change color to lightcoral
axes[0].set_title('Distribution of Like Counts')
axes[0].set_xlabel('Like Count')
axes[0].set_ylabel('Frequency')
axes[0].grid(True)

# Plotting boxplot for 'likeCount' with different color
sns.boxplot(x=cleaned_merged_df['likeCount'], ax=axes[1], color='lightblue')  #␣
 ↪Change color to lightblue
axes[1].set_title('Boxplot of Like Counts')
axes[1].set_xlabel('Like Count')
axes[1].grid(True)

# Descriptive statistics
descriptive_stats = cleaned_merged_df['likeCount'].describe()
print("Descriptive Statistics for Like Count:")
print(descriptive_stats)

plt.tight_layout()
plt.show()
```

```
Descriptive Statistics for Like Count:
count    1.400000e+02
mean     2.479739e+05
std      9.188088e+05
min      2.711000e+03
25%      2.021925e+04
50%      5.728550e+04
75%      2.011018e+05
max      1.047807e+07
Name: likeCount, dtype: float64
```

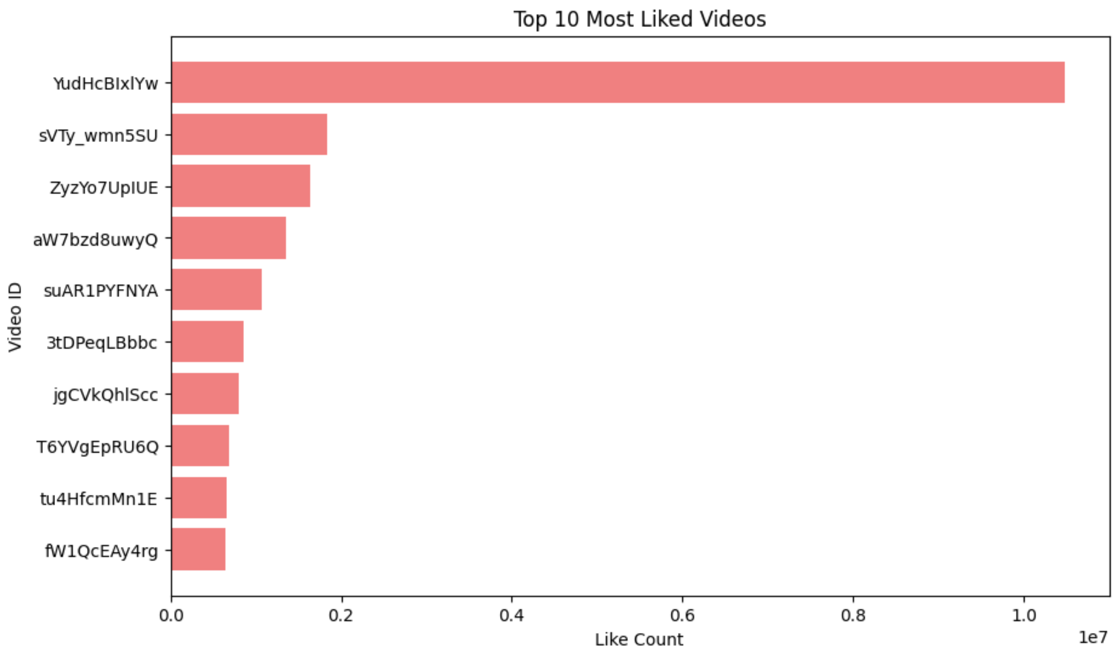


Based on the `likeCount`, the mean number of likes is approximately 247,974, and the standard deviation is around 918,808. This suggests that there is a considerable variation in the number of likes across music videos.

The descriptive statistics for `likeCount` indicate that while the mean number of likes is relatively high, there is significant variability in the level of appreciation among music videos, as evidenced by the relatively high standard deviation and wide range of like counts.

[27]:
```python
# Finding the index of the row with the maximum likeCount
max_likecount_index = cleaned_merged_df['likeCount'].idxmax()

# Retrieving the videoId of the video with the most likeCount
video_with_most_likes = cleaned_merged_df.loc[max_likecount_index, 'videoId']

# Printing the videoId and its corresponding likeCount
print("Video with the most likeCount:")
print("Video ID:", video_with_most_likes)
print("Like Count:", cleaned_merged_df['likeCount'].max())

# Sorting the DataFrame by likeCount to get the top liked videos
top_liked_videos = cleaned_merged_df.sort_values(by='likeCount',␣
 ↪ascending=False).head(10)

# Creating a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_liked_videos['videoId'], top_liked_videos['likeCount'],␣
 ↪color='lightcoral')
plt.xlabel('Like Count')
plt.ylabel('Video ID')
plt.title('Top 10 Most Liked Videos')
plt.gca().invert_yaxis()  # Invert y-axis to display video IDs from top to␣
  ↪bottom
plt.show()
```

```
Video with the most likeCount:
Video ID: YudHcBIxlYw
Like Count: 10478066
```



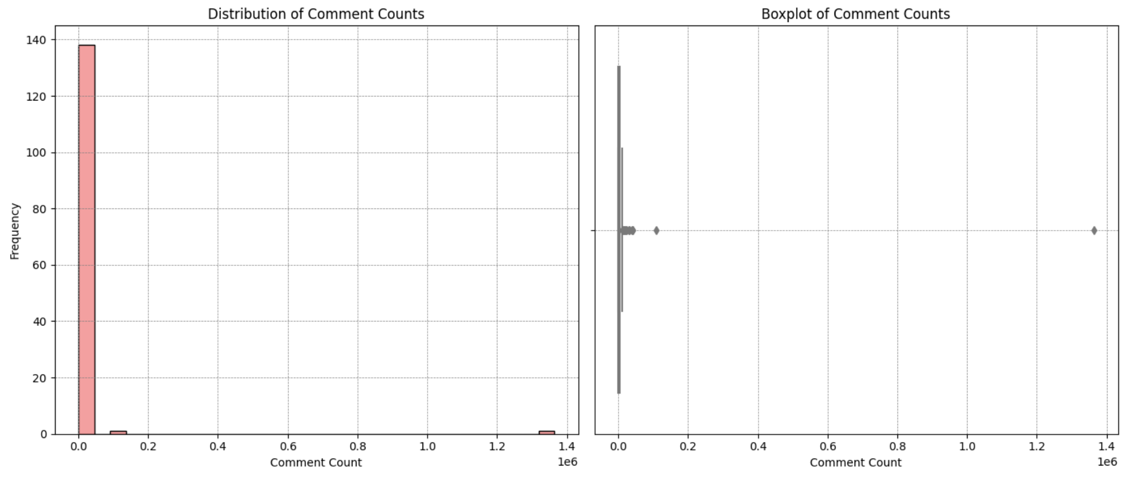
Based on the visualization, the graph indicates that the video with the video ID "YudHcBIxlYw" has the highest like count, totaling 10,478,066 likes. Similar to the `viewCount`, the content of the video can be considered to be K-pop as well. Indicating that the most viewed and liked content is oriented to Asian language and culture.

Distribution of the 'commentCount' (DV)

```
[28]: # Setting up the figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Plotting histogram for 'commentCount' with different color
sns.histplot(cleaned_merged_df['commentCount'], bins=30, color='lightcoral',␣
  ↪ax=axes[0])  # Change color to lightcoral
axes[0].set_title('Distribution of Comment Counts')
axes[0].set_xlabel('Comment Count')
axes[0].set_ylabel('Frequency')
axes[0].grid(True, linestyle='--', linewidth=0.5, color='gray')  # Adjust grid␣
  ↪appearance
```

```
# Plotting boxplot for 'commentCount' with different color
sns.boxplot(x=cleaned_merged_df['commentCount'], ax=axes[1], color='lightblue')␣
 ↪ # Change color to lightblue
axes[1].set_title('Boxplot of Comment Counts')
axes[1].set_xlabel('Comment Count')
axes[1].grid(True, linestyle='--', linewidth=0.5, color='gray')  # Adjust grid␣
 ↪appearance

# Descriptive statistics
descriptive_stats = cleaned_merged_df['commentCount'].describe()
print("Descriptive Statistics for Comment Count:")
print(descriptive_stats)

plt.tight_layout()
plt.show()
```

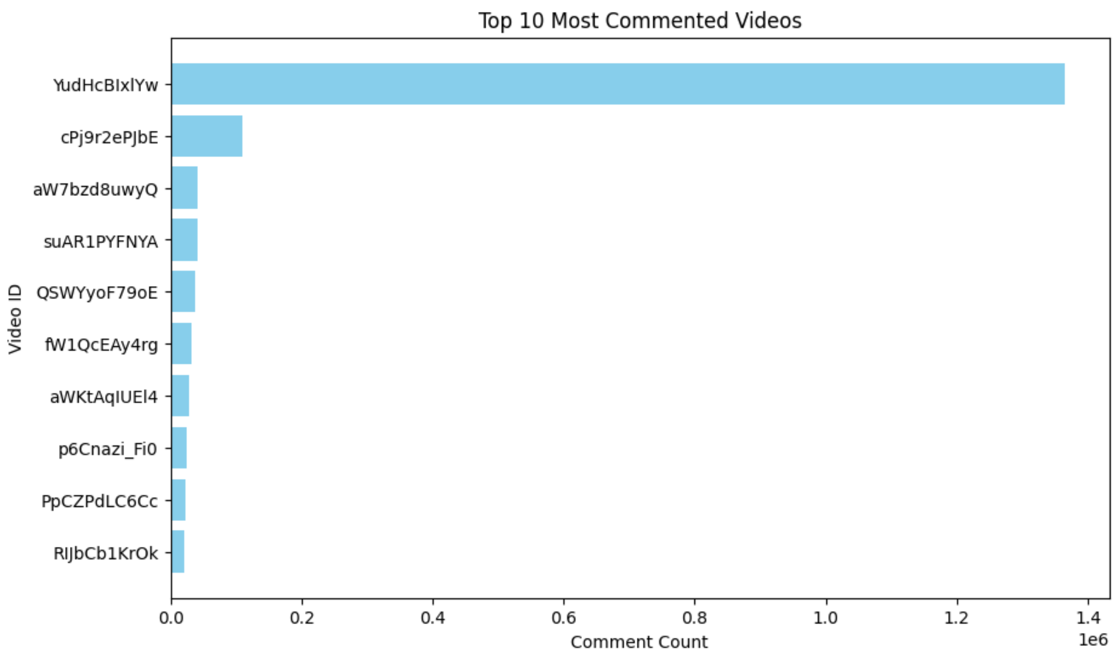
```
Descriptive Statistics for Comment Count:
count    1.400000e+02
mean     1.530436e+04
std      1.154726e+05
min      0.000000e+00
25%      4.040000e+02
50%      1.263500e+03
75%      4.937750e+03
max      1.364324e+06
Name: commentCount, dtype: float64
```



Based on the `commentCount`, the mean number of comments is approximately 15,304, and the standard deviation is approximately 115,473. This suggests that there is a considerable variation in the number of comments across music videos.

```
[29]: # Finding the index of the row with the maximum commentCount
max_commentcount_index = cleaned_merged_df['commentCount'].idxmax()

# Retrieving the videoId of the video with the most commentCount
video_with_most_comments = cleaned_merged_df.loc[max_commentcount_index,␣
 ↪'videoId']

# Printing the videoId and its corresponding commentCount
print("Video with the most commentCount:")
print("Video ID:", video_with_most_comments)
print("Comment Count:", cleaned_merged_df['commentCount'].max())

# Sorting the DataFrame by commentCount to get the top commented videos
top_commented_videos = cleaned_merged_df.sort_values(by='commentCount',␣
 ↪ascending=False).head(10)

# Creating a bar plot
plt.figure(figsize=(10, 6))
plt.barh(top_commented_videos['videoId'], top_commented_videos['commentCount'],␣
 ↪color='skyblue')
plt.xlabel('Comment Count')
plt.ylabel('Video ID')
plt.title('Top 10 Most Commented Videos')
plt.gca().invert_yaxis()  # Invert y-axis to display video IDs from top to␣
 ↪bottom
plt.show()
```

```
Video with the most commentCount:
Video ID: YudHcBIxlYw
Comment Count: 1364324
```



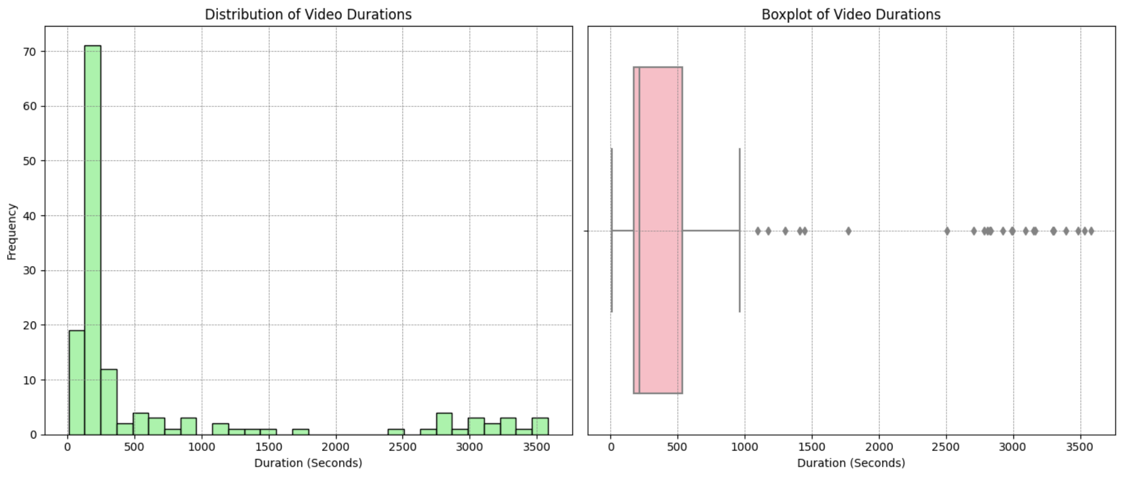
The visualization indicates that the video with the ID "YudHcBIxlYw" has the highest comment count, totaling 1,364,324 comments. This portrays that `viewCount` `likeCount` and. `commentCount` all have the same K-pop video that generates the most engagement. While the other videos portray significantly less engagement

Distribution of the 'durationSec'

[30]:
```
# Setting up the figure with two subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Plotting histogram for 'durationSec' with different color
sns.histplot(cleaned_merged_df['durationSec'], bins=30, color='lightgreen',␣
 ↪ax=axes[0])  # Change color to lightgreen
axes[0].set_title('Distribution of Video Durations')
axes[0].set_xlabel('Duration (Seconds)')
axes[0].set_ylabel('Frequency')
axes[0].grid(True, linestyle='--', linewidth=0.5, color='gray')  # Adjust grid␣
 ↪appearance

# Plotting boxplot for 'durationSec' with different color
sns.boxplot(x=cleaned_merged_df['durationSec'], ax=axes[1], color='lightpink') ␣
 ↪# Change color to lightpink
axes[1].set_title('Boxplot of Video Durations')
axes[1].set_xlabel('Duration (Seconds)')
axes[1].grid(True, linestyle='--', linewidth=0.5, color='gray')  # Adjust grid␣
 ↪appearance
```

```
# Descriptive statistics
descriptive_stats = cleaned_merged_df['durationSec'].describe()
print("Descriptive Statistics for Video Durations (Seconds):")
print(descriptive_stats)

plt.tight_layout()
plt.show()
```

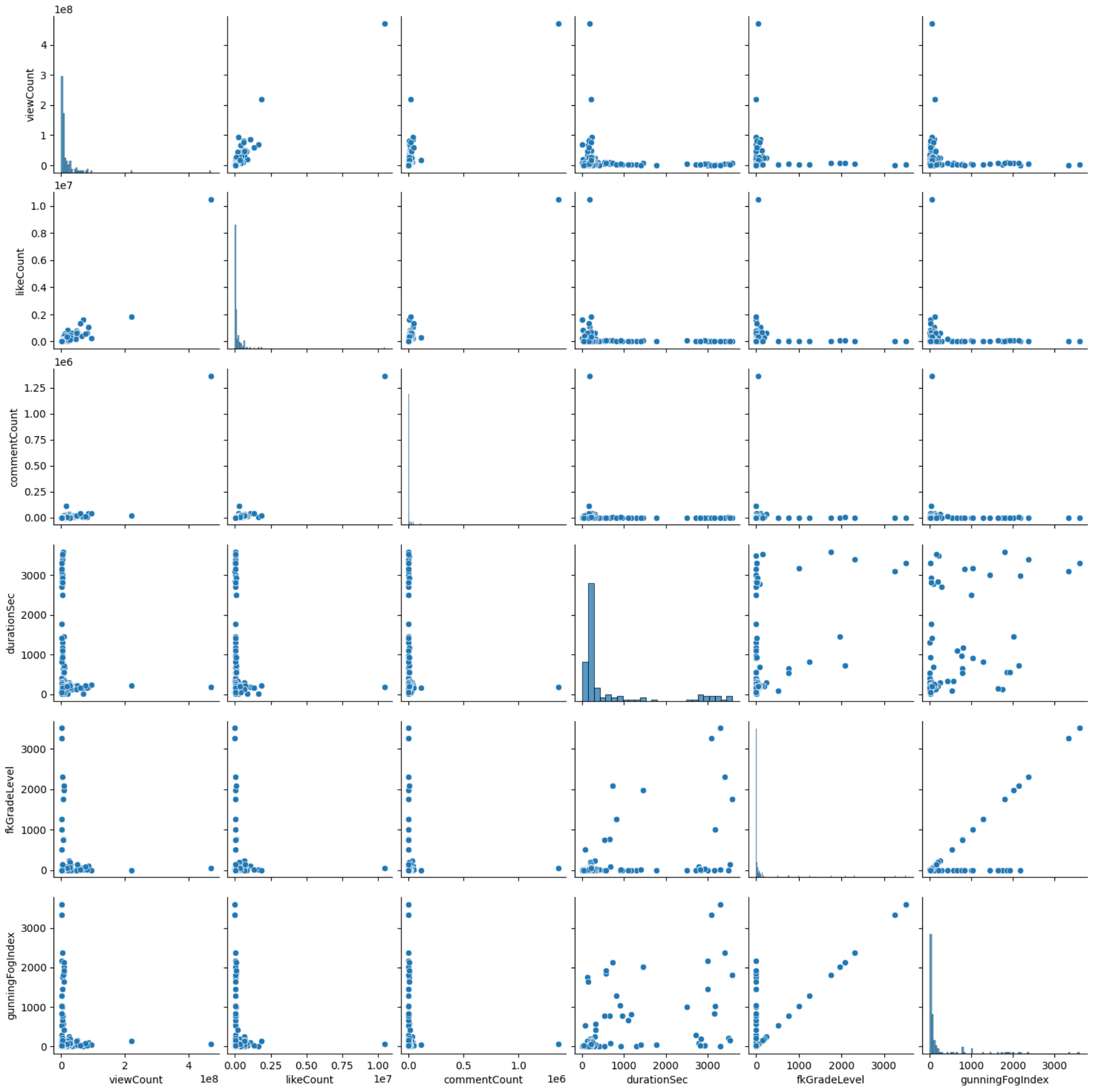
```
Descriptive Statistics for Video Durations (Seconds):
count      140.00000
mean       672.00000
std       1007.79146
min         10.00000
25%        171.75000
50%        216.50000
75%        532.25000
max       3582.00000
Name: durationSec, dtype: float64
```



The average duration of the videos is around 672 seconds, or about 11 minutes and 12 seconds. Additionally, the standard deviation of the duration is approximately 1007.79 seconds, indicating a variation of durations around the mean. There are different outliers, an example is a video that is 10 seconds long while the longest video is around an hour. This might indicate that some music videos are considered as ‘shorts’, while others present full concerts in the long video format.

In order to determine the correlation between the different variables that are explored, a pair plot using Seaborn’s pairplot function is implemented. This allows for a quick examination of potential correlations and patterns.

```
[31]: # Selecting the variables of interest
      variables_of_interest = ['viewCount', 'likeCount', 'commentCount',␣
       ↪'durationSec', 'fkGradeLevel', 'gunningFogIndex']

      # Creating the pair plot
      sns.pairplot(cleaned_merged_df[variables_of_interest])

      # Display the pair plot
      plt.show()
```



Based on the various scatterplots, `fkGradeLevel` and `gunningFogIndex` have a linear relationship with a positive correlation, which implies that as one variable increases, the other variable also tends to increase, and the relationship between the two variables can be represented by a straight line sloping upwards from left to right on a scatter plot.

Besides this result, other plots seem to be skewed and show no significant correlation to each other. Therefore, we cannot determine the relationship between the other metrics.

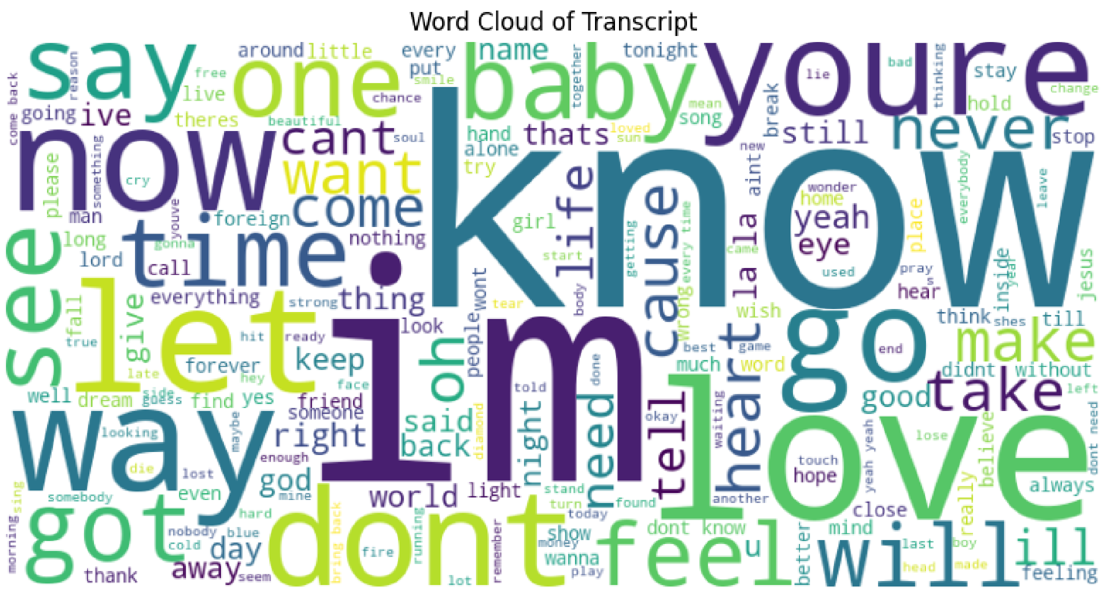
Visualizations Language Complexity (IV) In order to examine the complexity of the words that is used, an exploratory wordcloud is made with the (wordcloud) package. Additionally, the `Counter` function is used from the collections package, which allows to observe which words are used the most. Lastly, two histograms are made for the Gunning Fog Index and the Flesch-Kincaid Grade Level to examine whether there are significant differences between the different forms of complexity that can be determined.

[57]:
```
# Command to install the WordCloud package using pip
! pip install wordcloud
```

```
Requirement already satisfied: wordcloud in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (1.9.1.1)
Requirement already satisfied: matplotlib in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
wordcloud) (3.7.1)
Requirement already satisfied: pillow in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
wordcloud) (9.4.0)
Requirement already satisfied: numpy>=1.6.1 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
wordcloud) (1.24.3)
Requirement already satisfied: packaging>=20.0 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (22.0)
Requirement already satisfied: cycler>=0.10 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: six>=1.5 in
```

```
/Users/helgegeurtjacobusmoes/anaconda3/lib/python3.10/site-packages (from
python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

[58]:
```
# Importing the WordCloud class from the wordcloud module
from wordcloud import WordCloud

# Combine all transcripts into a single string
all_transcripts = ' '.join(cleaned_merged_df['transcript'])

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(all_transcripts)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Transcript')
plt.axis('off')  # Hide axes
plt.show()
```



[59]:
```
# Importing the Counter class from the collections module
from collections import Counter

# Importing the re module for regular expression operations
import re

# Combine all transcripts into a single string
```

```
all_transcripts = ' '.join(cleaned_merged_df['transcript'])

# Tokenize the text into individual words
words = re.findall(r'\b\w+\b', all_transcripts.lower())  # Convert to lowercase␣
↪for case-insensitivity

# Count the frequency of each word
word_counts = Counter(words)

# Retrieve the most common words
most_common_words = word_counts.most_common(50)  # Change the number to␣
↪retrieve more or fewer words

# Print the most common words
for word, count in most_common_words:
    print(f'{word}: {count}')
```

```
i: 4836
you: 4727
the: 3613
to: 2378
me: 2184
and: 1995
my: 1769
a: 1640
in: 1353
it: 1251
im: 1099
that: 1093
your: 1083
all: 1078
is: 954
of: 950
be: 911
for: 868
we: 865
know: 860
but: 790
like: 789
on: 770
dont: 767
so: 747
love: 706
just: 673
its: 672
no: 668
this: 638
```

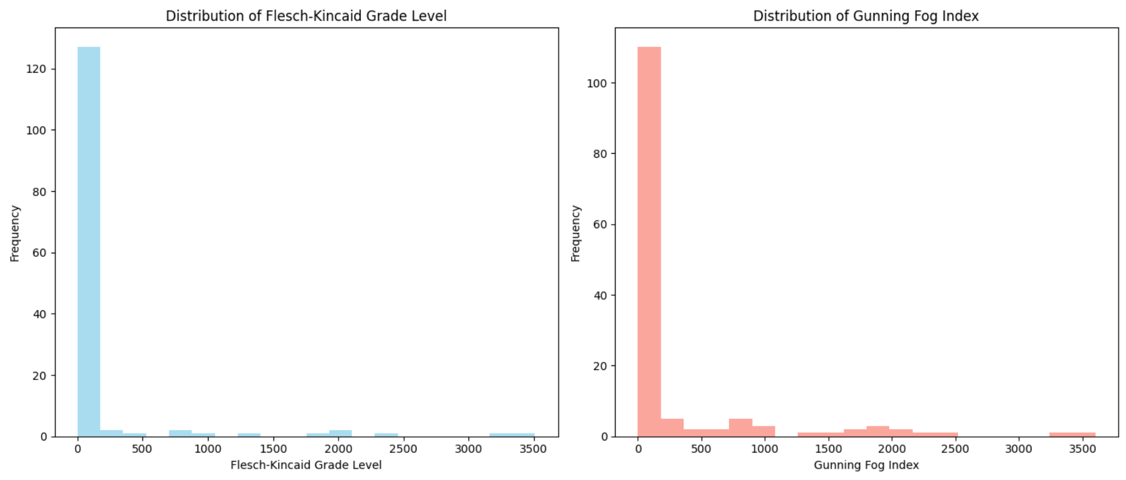
```
when: 610
what: 559
with: 545
now: 517
youre: 503
up: 497
never: 490
can: 481
do: 446
let: 443
go: 434
will: 431
baby: 430
yeah: 426
way: 422
if: 417
time: 408
got: 405
cant: 391
are: 382
```

The list shows the most frequent words in music videos on YouTube. These words are common in lyrics and reflect themes of personal connection, emotion, and expression often found in music. Words like “i”, “you”, “the”, “me”, and “my” indicate a focus on personal narratives and relationships. Additionally, words like “love”, “know”, “just”, “like”, and “dont” suggest themes of affection, understanding, and self-expression commonly found in music lyrics. These words contribute to the emotional and relatable nature of music content on YouTube.

[60]:
```python
# Calculating descriptive statistics for 'fkGradeLevel'
fk_grade_level_stats = cleaned_merged_df['fkGradeLevel'].describe()

# Calculating descriptive statistics for 'gunningFogIndex'
gunning_fog_index_stats = cleaned_merged_df['gunningFogIndex'].describe()

print("Descriptive Statistics for Flesch-Kincaid Grade Level:")
print(fk_grade_level_stats)
print("\nDescriptive Statistics for Gunning Fog Index:")
print(gunning_fog_index_stats)

# Creating figure and axes
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Plotting histogram for 'fkGradeLevel'
axes[0].hist(cleaned_merged_df['fkGradeLevel'], bins=20, color='skyblue',␣
 ↪alpha=0.7)
axes[0].set_title('Distribution of Flesch-Kincaid Grade Level')
axes[0].set_xlabel('Flesch-Kincaid Grade Level')
axes[0].set_ylabel('Frequency')
```

```
# Plotting histogram for 'gunningFogIndex'
axes[1].hist(cleaned_merged_df['gunningFogIndex'], bins=20, color='salmon',␣
 ↪alpha=0.7)
axes[1].set_title('Distribution of Gunning Fog Index')
axes[1].set_xlabel('Gunning Fog Index')
axes[1].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```

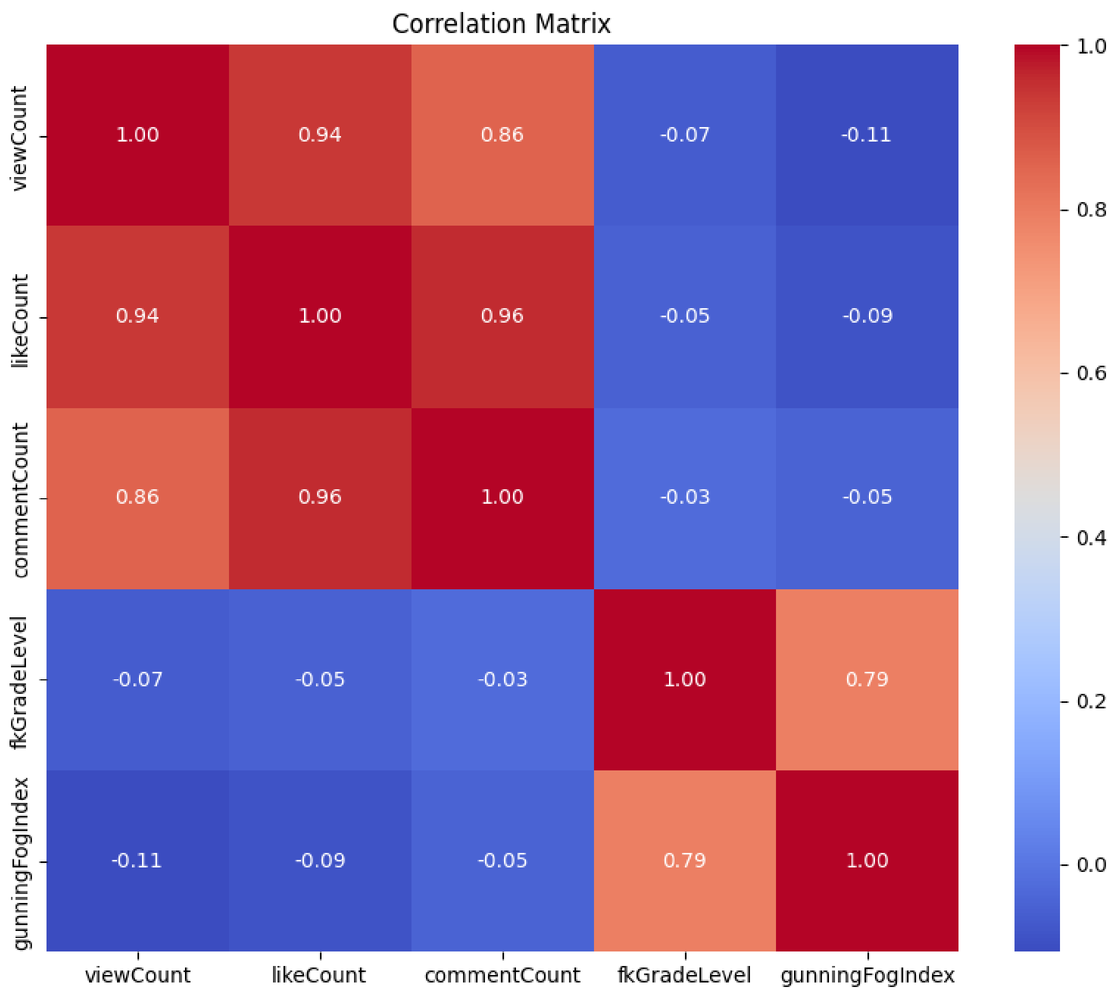
```
Descriptive Statistics for Flesch-Kincaid Grade Level:
count     140.000000
mean      156.814286
std       538.601247
min         0.000000
25%         0.000000
50%         8.000000
75%        35.750000
max      3509.000000
Name: fkGradeLevel, dtype: float64

Descriptive Statistics for Gunning Fog Index:
count     140.000000
mean      301.662500
std       647.464622
min         0.400000
25%        13.900000
50%        41.040000
75%       131.665000
max      3600.550000
Name: gunningFogIndex, dtype: float64
```



These statistics suggest that the language complexity in the dataset varies widely, ranging from very simple to extremely complex text. The majority of the text may require a high level of education or specialized knowledge to understand, but there are also simpler and more accessible portions.

Based on the visualisations of Flesch_kincaid Grade Level and the Gunning Fog Index, it is apparent that there are similarities in the output, which may indicate a correlation between the two models. Both models portray a large standard deviation, which indicates significant variability in language complexity throughout the dataset. Moreover, the mean of both visualizations is also considered to be high (195.69 and 301.92), which indicates that the complexity of the lyrics requires a high reading level aand education in order to comprehend.

```
[62]: # Selecting the specified variables from the DataFrame
selected_variables = ['viewCount', 'likeCount', 'commentCount', 'fkGradeLevel',␣
 ↪'gunningFogIndex']
correlation_df = cleaned_merged_df[selected_variables]

# Calculating the correlation matrix
correlation_matrix = correlation_df.corr()

# Plotting the correlation matrix using Seaborn with a heat map
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪annot_kws={"size": 10})
plt.title('Correlation Matrix')
plt.show()

print(correlation_matrix)
```



```
                viewCount  likeCount  commentCount  fkGradeLevel  \
viewCount        1.000000   0.935429      0.855086     -0.067349
likeCount        0.935429   1.000000      0.959599     -0.051411
commentCount     0.855086   0.959599      1.000000     -0.025696
fkGradeLevel    -0.067349  -0.051411     -0.025696      1.000000
gunningFogIndex -0.106833  -0.085262     -0.046305      0.790260

                gunningFogIndex
viewCount             -0.106833
likeCount             -0.085262
commentCount          -0.046305
fkGradeLevel           0.790260
gunningFogIndex        1.000000
```

The correlation matrix suggests strong relationships between view count, like count, and comment count, while language complexity measures have weaker correlations with these variables. Based on the correlation matrix, the `viewCount`, `likeCount`, and `commentCount` have strong positive correlations with each other. This indicates that videos with higher view counts tend to have more likes and comments, and vice versa. Moreover, both `viewCount` and `likeCount` have particularly high correlations with each other, with coefficients of approximately 0.94, indicating a very strong positive linear relationship between these two variables. This suggests that videos with higher view counts are likely to have more likes.

Furthermore, the `fkGradeLevel` and `gunningFogIndex` have weak negative correlations with the other variables (`viewCount`, `likeCount`, `commentCount`). This suggests that as the complexity of the language in video transcripts increases (higher grade level or Fog index), there tends to be a slight decrease in view count, like count, and comment count. However, these correlations are relatively weak compared to the correlations between view count, like count, and comment count.

Additionally, `fkGradeLevel` and `gunningFogIndex` have a moderate positive correlation with each other (approximately 0.79), indicating that as the Flesch-Kincaid Grade Level increases, the Gunning Fog Index tends to increase as well, and the other way round. This suggests some level of agreement between the two measures of text complexity.

Report for Stakeholders The data exploration sought to answer the research question by examining the relationship between language complexity in YouTube music videos and their success metrics, including view count, like count, and comment count. In order to answer: 'How can the insights be used to improve music marketing and content creation on the YouTube platform?', the exploratory data analysis suggests that specific content types generate the most engagement with users, which holds potential to improve music marketing and content creation. It should be noted that there are data quality issues present that should have been addressed earlier, such as identifying and potentially removing outliers. Due to the choice to resemble the raw data as possible, meaningful conclusions were challenging to make and actionable insights for YouTube as a company or content creators should be considered with caution.

With regard to the words used in YouTube music videos, often lyrics reflected personal narratives, emotions, and connections, with common words like "i", "you", and "me" emphasizing personal relationships, while terms such as "love", "know", and "like" convey themes of affection and self-expression, contributing to the emotional resonance of music content on the platform. Notably, music videos categorized as K-pop, represented by video IDs such as YudHcBIxlYw, svty_wmn5su, and cPj9r2ePJbE, attracted the highest views, likes, and comments, underscoring the rising popularity of Korean music. Additionally, Pop music, hip hop, and electronic music emerged as the most prevalent genres on YouTube, reflecting the platform's tendency to promote these broad categories. Hence, the results suggests focusing on creating engaging content aligned with popular themes and genres, such as personal narratives and K-pop, to boost audience interaction on YouTube.

Furthermore, the results from descriptive statistics for language complexity metrics, such as Flesch-Kincaid Grade Level and Gunning Fog Index, revealed wide variability, ranging from simple to extremely complex text should be taken with a grain of salt. Also strong positive correlations emerged between view count, like count, and comment count, indicating that videos with higher views tend to generate more likes and comments. Additionally, descriptive statistics for language complexity metrics, such as Flesch-Kincaid Grade Level and Gunning Fog Index, revealed wide variability, ranging from simple to extremely complex text. Despite the high average language complexity in music videos, these measures showed weaker correlations with other success metrics (likeCount, viewCount and commentCount), suggesting that language complexity may not significantly impact engagement.

However, this advice contains biases based on the algorithm that granted the data from the YouTube API. During the exploratory data analysis, predominantly right-skewed histograms was observed, indicating a scarcity of data points with high values, pulling the distribution to the right. This skewness implies a non-normal distribution, with the mean exceeding the median, possibly due to rare significant occurrences or data collection errors. Furthermore, multiple outliers were present across all visualizations, suggesting deviations from the bulk of the data.

In conclusion, although this exploratory data analysis grants the stakeholders, such as creators, musicians and YouTube itelf, interesting outcomes on how to tailor their content to generate more user engagement. The presence of outliers in success metrics could skew the analysis and lead to inaccurate predictions or conclusions in this research. Additionally, the overrepresentation of right-skewed distributions may indicate unbalanced data, where a small number of videos dominate the metrics while the majority have lower engagement. While language complexity measures demonstrated weaker correlations with engagement metrics compared to other variables, further investigation is needed to understand their impact fully. Therefore, a robust regression and a random forrest classification are conducted in order to retrieve more insights on whether language in lyrics plays a significant role in the user engagement. Nevertheless, stakeholders are advised to consider analyzing additional variables and features or conducting qualitative analysis to explore viewer preferences.

1.5 Modeling and Hypothesis Testing

After understanding and preparing the data, you are ready to do the modeling. It is important to explain to the reader: * Explaining which models will be used, and why. * If appropriate, explaining which strategy you are using to fine tune or improve the model. * Discussion about the model evaluation (i.e., why you believe the model is a good model) and, if appropriate, comparison between different models. * After the appropriate model is identified, then the hypothesis can be tested. You can then report the result of the tests being used to test the hypothesis (as discussed in the criteria for the assignment), explaining to the reader how the test is being made and what the results of the testing mean for the hypothesis at hand. * Visualizations (as appropriate) are helpful to explain to the reader what has been found. * Usage of the model to make predictions (e.g., key audience segments). * Reviewing the model to indicate which features (variables) are responsible for the predictions (related to Explainable AI discussions).

IMPORTANT: * We consider as different models both when different features/variables are being tested (to achieve the same objective), or when different algorithms are being tested (with the same features/IVs). * It is important to explain to the reader what the model contains/is, and also how it is being evaluated. When the evaluation is done, it is also important to discuss what the result means in the current context. * As the focus of this course is on predictive analytics, it is expected that predictions are made (using scikit-learn) and discussed after a hypothesis is tested. This means that when ANOVAs or T-Tests are used (as statistical testing), and equivalent OLS/Linear Regression Model (in scikit-learn) needs to be used to make predictions. All tests need to be done in Python, using statsmodels and/or scikit-learn.

The objective of this study is to investigate the impact of language complexity, as measured by the Flesch-Kincaid Grade Level and Gunning Fog Index, on the engagement of YouTube music videos, indicated by their like count, view count and comment count, while considering other relevant predictors. Initial exploratory data analysis revealed that many variables in the dataset exhibit right-skewed distributions with numerous outliers. To address this issue, robust modeling techniques will be employed and evaluated to validate the hypothesis. Since there are correlations among the dependent variables (`viewCount`, `likeCount` and `commentCount`) and predictor variables, it may be beneficial to incorporate both into the model. However, multicollinearity, stemming from the correlation between the three variables, needs to be addressed to accurately interpret the effects of each independent variable.

During the exploratory data analysis phase, it was noted that certain variables demonstrated stronger correlations than others, such as `gunningFogIndex` and `fkGradeLevel`, while the relationship between language complexity and the dependent variables may not strictly adhere to a linear pattern. Scatterplots illustrating the relationship between independent and dependent variables exhibited scattered patterns without clear linear trends, suggesting a potentially more complicated relationship that must be considered when selecting the appropriate model.

Given the potential departure from the normality assumption, caution is crucial when conducting this research and interpreting results. The final model selection process will involve comparing the performance of different models using metrics such as R-squared and mean squared error.

To test the research question effectively, two distinct models will be employed: robust regression and a random forest regressor. Robust regression, designed to accommodate outliers in data, is deemed suitable for addressing the outliers identified during the initial analysis (Robust Regression). Additionally, the method is adept at handling (right) skewed data.

Furthermore, a random forest model will be utilized due to its capability to handle datasets with multiple independent variables potentially influencing the dependent variable (Random Forest Classification). Moreover, as correlations were observed among certain independent variables, the random forest model can effectively manage multicollinearity. Furthermore, its capacity to capture potential non-linear relationships between the dependent and independent variables makes it a valuable option for this analysis.

The following libraries and modules are needed to examine the data further:

[80]:
```
# Importing the numpy library and aliasing it as np
import numpy as np

# Importing the train_test_split function from sklearn.model_selection module
from sklearn.model_selection import train_test_split

# Importing the LinearRegression class from sklearn.linear_model module
from sklearn.linear_model import LinearRegression

# Importing the RandomForestRegressor class from sklearn.ensemble module
from sklearn.ensemble import RandomForestRegressor

# Importing r2_score and mean_squared_error functions from sklearn.metrics␣
↪module
from sklearn.metrics import r2_score, mean_squared_error
```

```
# Importing the SVR class from sklearn.svm module
from sklearn.svm import SVR

# Importing the statsmodels library and aliasing it as sm
import statsmodels.api as sm

# Importing the formula API from statsmodels and aliasing it as smf
import statsmodels.formula.api as smf

# Importing the RANSACRegressor class from the sklearn.linear_model module
from sklearn.linear_model import RANSACRegressor

# Importing the permutation_importance function from the sklearn.inspection␣
↪module
from sklearn.inspection import permutation_importance
```

Prediction of 'LikeCount'

[81]:
```
# Defining the IV and DV
X_likeCount = cleaned_merged_df[['viewCount', 'commentCount', 'durationSec',␣
↪'fkGradeLevel', 'gunningFogIndex']]
y_likeCount = cleaned_merged_df['likeCount']

# Splitting the data into training and testing sets
X_train_likeCount, X_test_likeCount, y_train_likeCount, y_test_likeCount =␣
↪train_test_split(X_likeCount, y_likeCount, test_size=0.2, random_state=42)

# Fitting a Robust regression model for likeCount
robust_model_likeCount = sm.RLM(y_train_likeCount, sm.
↪add_constant(X_train_likeCount), M=sm.robust.norms.HuberT()).fit()
robust_y_pred_likeCount = robust_model_likeCount.predict(sm.
↪add_constant(X_test_likeCount))
robust_r_squared_likeCount = r2_score(y_test_likeCount, robust_y_pred_likeCount)
robust_rmse_likeCount = np.sqrt(mean_squared_error(y_test_likeCount,␣
↪robust_y_pred_likeCount))

# Fitting a Random Forest model for likeCount
rf_model_likeCount = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model_likeCount.fit(X_train_likeCount, y_train_likeCount)
rf_y_pred_likeCount = rf_model_likeCount.predict(X_test_likeCount)
rf_r_squared_likeCount = r2_score(y_test_likeCount, rf_y_pred_likeCount)
rf_rmse_likeCount = np.sqrt(mean_squared_error(y_test_likeCount,␣
↪rf_y_pred_likeCount))

# Printing the relevant results
print('Robust R-squared: ', robust_r_squared_likeCount)
print('Robust RMSE: ', robust_rmse_likeCount)
print('Random Forest R-squared: ', rf_r_squared_likeCount)
print('Random Forest RMSE: ', rf_rmse_likeCount)
```

```
Robust R-squared:  0.22149292891804295
Robust RMSE:  154774.7275405583
Random Forest R-squared:  -0.7818288660651913
Random Forest RMSE:  234154.09781370967
```

The robust R-squared value of 0.221 indicates that the robust regression model explains about 22.1% of the variance in the dependent variable 'likeCount' using the independent variables 'viewCount', 'commentCount', 'durationSec', 'fkGradeLevel', and 'gunningFogIndex'. The robust RMSE value of approximately 154,775 indicates that, on average, the model's predictions are off by around 154,775 like counts.

On the other hand, the Random Forest R-squared value of -0.782 suggests that the Random Forest model performs poorly, with a negative R-squared indicating that the model performs worse than a horizontal line. Moreover, the Random Forest RMSE value of approximately 234,154 implies that, on average, the model's predictions are off by around 234,154 like counts.

Despite both models attempting to explain the variance in 'likeCount', the robust regression model outperforms the Random Forest model in terms of R-squared value and RMSE. Therefore, the robust regression model appears to be more suitable for predicting 'likeCount' based on the provided independent variables.

[50]:
```
# Printing the summary of the robust linear regression model for the␣
↪'likeCount' variable
print(robust_model_likeCount.summary())
```

```
                    Robust linear Model Regression Results
==============================================================================
Dep. Variable:              likeCount   No. Observations:                  112
Model:                            RLM   Df Residuals:                      106
Method:                          IRLS   Df Model:                            5
Norm:                          HuberT
Scale Est.:                       mad
Cov Type:                          H1
Date:                Sat, 23 Mar 2024
Time:                        12:01:28
No. Iterations:                     6
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
const            9764.3753   6204.868      1.574      0.116   -2396.943
2.19e+04
viewCount           0.0083      0.000     44.288      0.000       0.008
0.009
```

```
commentCount         4.8237       0.072     67.135      0.000       4.683
4.965
durationSec         -4.8629       5.533     -0.879      0.379     -15.708
5.982
fkGradeLevel        -2.1759      15.347     -0.142      0.887     -32.256
27.905
gunningFogIndex     -0.4852      14.647     -0.033      0.974     -29.193
28.223
================================================================================
===

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore .
```

In the robust linear model regression results, the coefficient for the constant term is 9764.3753 (SE = 6204.868), z = 1.574, p = 0.116. This suggests that the constant is not statistically significant at the 0.05 level, indicating that the intercept term may not be necessary in the model. However, this also implies that the model might not accurately capture the relationship between the dependent variable (likeCount) and the independent variables.

The coefficients for the predictor variables viewCount and commentCount are both statistically significant with p-values less than 0.05. The coefficient for durationSec is not statistically significant at the 0.05 level (p = 0.379), indicating that it may not have a significant impact on likeCount.

Specifically, the coefficient for viewCount is 0.0083 (SE = 0.000), suggesting that an increase of one unit in viewCount is associated with an increase of 0.0083 units in likeCount. Similarly, an increase of one unit in commentCount is associated with an increase of 4.8237 units in likeCount, holding all other variables constant. However, the coefficients for fkGradeLevel and gunningFogIndex are not statistically significant at the 0.05 level, indicating that they may not have a significant impact on likeCount.

Prediction of 'viewCount'

```
[64]: # Defining the IV and DV
X_viewCount = cleaned_merged_df[['likeCount', 'commentCount', 'durationSec',␣
 ↪'fkGradeLevel', 'gunningFogIndex']]
y_viewCount = cleaned_merged_df['viewCount']

# Splitting the data into training and testing sets
X_train_viewCount, X_test_viewCount, y_train_viewCount, y_test_viewCount =␣
 ↪train_test_split(X_viewCount, y_viewCount, test_size=0.2, random_state=42)

# Fitting a Robust regression model for viewCount
robust_model_viewCount = sm.RLM(y_train_viewCount, sm.
 ↪add_constant(X_train_viewCount), M=sm.robust.norms.HuberT()).fit()
robust_y_pred_viewCount = robust_model_viewCount.predict(sm.
 ↪add_constant(X_test_viewCount))
robust_r_squared_viewCount = r2_score(y_test_viewCount, robust_y_pred_viewCount)
```

```
robust_rmse_viewCount = np.sqrt(mean_squared_error(y_test_viewCount,␣
↪robust_y_pred_viewCount))

# Fitting a Random Forest model for viewCount
rf_model_viewCount = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model_viewCount.fit(X_train_viewCount, y_train_viewCount)
rf_y_pred_viewCount = rf_model_viewCount.predict(X_test_viewCount)
rf_r_squared_viewCount = r2_score(y_test_viewCount, rf_y_pred_viewCount)
rf_rmse_viewCount = np.sqrt(mean_squared_error(y_test_viewCount,␣
↪rf_y_pred_viewCount))

# Printing the relevant results
print('Robust R-squared: ', robust_r_squared_viewCount)
print('Robust RMSE: ', robust_rmse_viewCount)
print('Random Forest R-squared: ', rf_r_squared_viewCount)
print('Random Forest RMSE: ', rf_rmse_viewCount)
```

```
Robust R-squared:  0.40571618280182287
Robust RMSE:  18579204.421275713
Random Forest R-squared:  0.5034908937660731
Random Forest RMSE:  16982192.914838
```

According to the robust R-squared value, approximately 0.406 implies that the robust regression model explains about 40.6% of the variance in the dependent variable (viewCount) using the provided independent variables (likeCount, commentCount, durationSec, fkGradeLevel, and gunningFogIndex). Moreover, the robust RMSE value of approximately 18,579,204 indicates that, on average, the model's predictions are off by around 18,579,204 units of viewCount.

The Random Forest R-squared value of approximately 0.503 indicates that the Random Forest model explains about 50.3% of the variance in the dependent variable (viewCount). Additionally, the Random Forest RMSE value of approximately 16,982,193 suggests that, on average, the model's predictions are off by around 16,982,193 units in terms of viewCount.

Both models perform relatively well in explaining the variance in the dependent variable, with the Random Forest model outperforming the robust regression model in terms of R-squared value and RMSE. Therefore, the Random Forest model may be preferred for predicting the viewCount based on the provided independent variables.

```
[49]: # Printing the summary of the robust linear regression model for the␣
       ↪'viewCount' variable
      print(robust_model_viewCount.summary())
```

```
                    Robust linear Model Regression Results
==============================================================================
Dep. Variable:              viewCount   No. Observations:                  112
Model:                            RLM   Df Residuals:                      106
Method:                          IRLS   Df Model:                            5
Norm:                          HuberT
Scale Est.:                       mad
```

```
Cov Type:                          H1
Date:                 Sat, 23 Mar 2024
Time:                         11:55:39
No. Iterations:                      2
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
const            3.099e+06   7.31e+05      4.239      0.000    1.67e+06
4.53e+06
likeCount          67.5166      1.928     35.021      0.000      63.738
71.295
commentCount     -174.9252     15.189    -11.517      0.000    -204.694
-145.156
durationSec      -529.9026    652.728     -0.812      0.417   -1809.225
749.420
fkGradeLevel      670.1522   1805.212      0.371      0.710   -2867.997
4208.302
gunningFogIndex  -336.8588   1723.080     -0.195      0.845   -3714.034
3040.316
================================================================================
===

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore .
```

In the robust linear model regression results, the coefficient for the constant term is 3.099e+06 (SE = 7.31e+05), z = 4.239, p < 0.001. This indicates that the constant is statistically significant at the 0.05 level, suggesting that the intercept term is necessary in the model.

The coefficients for the predictor variables likeCount and commentCount are both statistically significant with p-values less than 0.001, indicating a strong relationship with viewCount.

Specifically, the coefficient for likeCount is 67.5166 (SE = 1.928), suggesting that an increase of one unit in likeCount is associated with an increase of 67.5166 units in viewCount. Conversely, an increase of one unit in commentCount is associated with a decrease of 174.9252 units in viewCount, holding all other variables constant.

Similar to likeCount, the coefficients: durationSec, fkGradeLevel, and gunningFogIndex were not statistically significant at the 0.05 level, which implies that they do not have a significant impact on viewCount.

Prediction of ‘commentCount’

[39]:
```
# Defining the IV and DV
X_commentCount = cleaned_merged_df[['viewCount', 'likeCount', 'durationSec',␣
↪'fkGradeLevel', 'gunningFogIndex']]
y_commentCount = cleaned_merged_df['commentCount']
```

```
# Splitting the data into training and testing sets
X_train_commentCount, X_test_commentCount, y_train_commentCount,␣
 ↪y_test_commentCount = train_test_split(X_commentCount, y_commentCount,␣
 ↪test_size=0.2, random_state=42)

# Fitting a Robust regression model for commentCount
robust_model_commentCount = sm.RLM(y_train_commentCount, sm.
 ↪add_constant(X_train_commentCount), M=sm.robust.norms.HuberT()).fit()
robust_y_pred_commentCount = robust_model_commentCount.predict(sm.
 ↪add_constant(X_test_commentCount))
robust_r_squared_commentCount = r2_score(y_test_commentCount,␣
 ↪robust_y_pred_commentCount)
robust_rmse_commentCount = np.sqrt(mean_squared_error(y_test_commentCount,␣
 ↪robust_y_pred_commentCount))

# Fitting a Random Forest model for commentCount
rf_model_commentCount = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model_commentCount.fit(X_train_commentCount, y_train_commentCount)
rf_y_pred_commentCount = rf_model_commentCount.predict(X_test_commentCount)
rf_r_squared_commentCount = r2_score(y_test_commentCount,␣
 ↪rf_y_pred_commentCount)
rf_rmse_commentCount = np.sqrt(mean_squared_error(y_test_commentCount,␣
 ↪rf_y_pred_commentCount))

# Printing the relevant results
print('Robust R-squared for commentCount: ', robust_r_squared_commentCount)
print('Robust RMSE for commentCount: ', robust_rmse_commentCount)
print('Random Forest R-squared for commentCount: ', rf_r_squared_commentCount)
print('Random Forest RMSE for commentCount: ', rf_rmse_commentCount)
```

```
Robust R-squared for commentCount:  -0.5249185096642723
Robust RMSE for commentCount:  10501.350046475107
Random Forest R-squared for commentCount:  0.3582063775085962
Random Forest RMSE for commentCount:  6812.703934954376
```

The robust R-squared value for predicting commentCount is approximately -0.525. This indicates that the robust regression model explains a negative proportion of the variance in the dependent variable (commentCount) using the provided independent variables (viewCount, likeCount, durationSec, fkGradeLevel, and gunningFogIndex). A negative R-squared value suggests that the model performs worse than a horizontal line, indicating poor performance or overfitting. Additionally, the robust RMSE value for commentCount is approximately 10,501.35. This suggests that, on average, the model's predictions are off by around 10,501.35 units in terms of commentCount.

On the other hand, the Random Forest R-squared value for commentCount is approximately 0.358. This indicates that the Random Forest model explains about 35.8% of the variance in the dependent variable (commentCount) using the provided independent variables. The Random Forest RMSE value for commentCount is approximately 6,812.70. This suggests that, on average, the model's predictions are off by around 6,812.70 units in terms of commentCount.

In comparison, the Random Forest model performs better in terms of R-squared value and RMSE for predicting commentCount compared to the robust regression model. However, the R-squared value for the Random Forest model is still relatively low, indicating that there may be limitations in accurately predicting commentCount using the provided independent variables. Further exploration or optimization of models may be necessary to improve predictive accuracy.

[51]:
```
# Printing the summary of the robust linear regression model for the␣
 ↪'commentCount' variable
print(robust_model_commentCount.summary())
```

```
                    Robust linear Model Regression Results
==============================================================================
Dep. Variable:           commentCount   No. Observations:                  112
Model:                            RLM   Df Residuals:                      106
Method:                          IRLS   Df Model:                            5
Norm:                          HuberT
Scale Est.:                       mad
Cov Type:                          H1
Date:                Sat, 23 Mar 2024
Time:                        12:06:30
No. Iterations:                    50
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
const            -659.2534    333.312     -1.978      0.048   -1312.532
-5.975
viewCount          -0.0003   1.61e-05    -20.602      0.000      -0.000
-0.000
likeCount           0.0632      0.001     80.201      0.000       0.062
0.065
durationSec         0.2583      0.307      0.840      0.401      -0.344
0.861
fkGradeLevel       -0.0408      0.856     -0.048      0.962      -1.719
1.638
gunningFogIndex     0.1254      0.817      0.153      0.878      -1.476
1.727
================================================================================
===

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore .
```

In the robust linear model regression results for commentCount, the coefficient for the constant term is -659.2534 (SE = 333.312), z = -1.978, p = 0.048. This indicates that the constant is statistically significant at the 0.05 level, suggesting that the intercept term is necessary in the model.

The coefficients for the predictor variables viewCount and likeCount are both statistically significant with p-values less than 0.001, indicating a strong relationship with commentCount.

Specifically, the coefficient for viewCount is -0.0003 (SE = 1.61e-05), suggesting that an increase of one unit in viewCount is associated with a decrease of 0.0003 units in commentCount. Conversely, an increase of one unit in likeCount is associated with an increase of 0.0632 units in commentCount, holding all other variables constant.

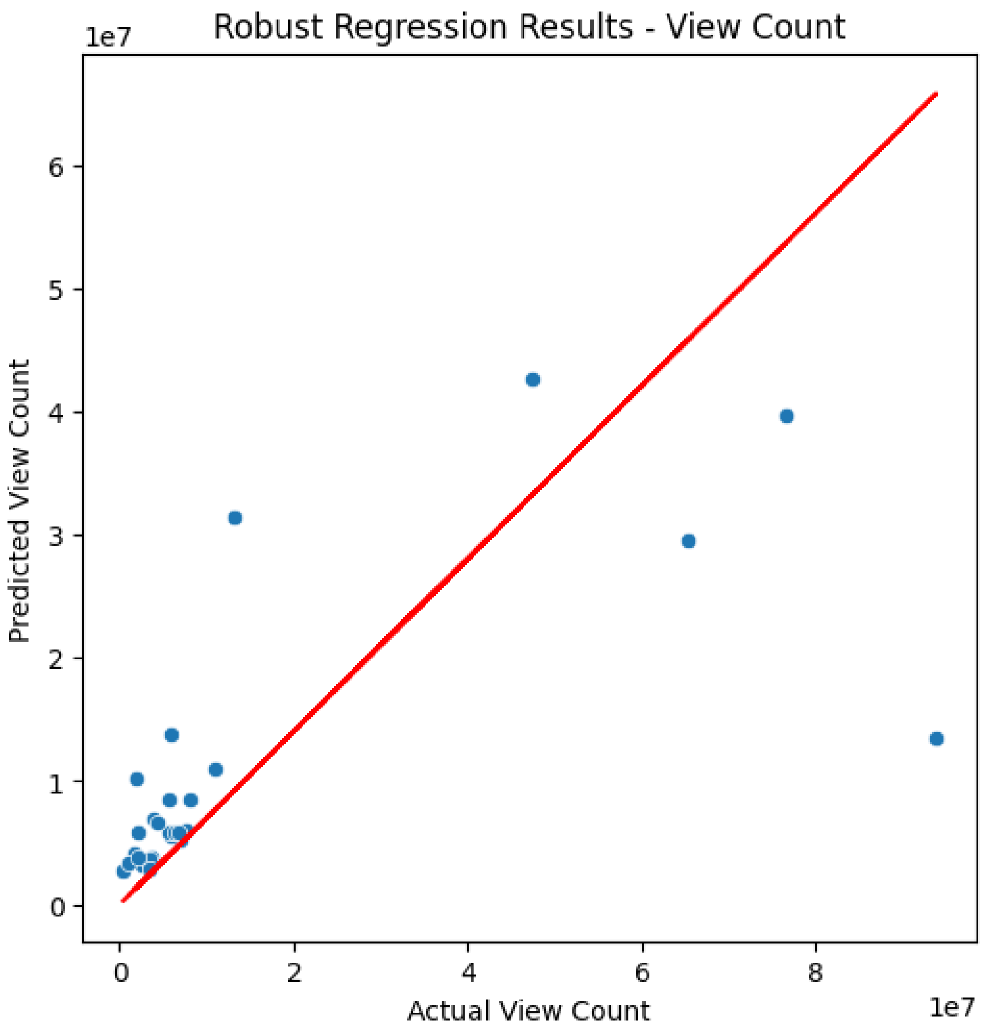
Lastly, the coefficients for durationSec, fkGradeLevel, and gunningFogIndex did not contain statistically significance at the 0.05 level, which supports the claim that they do not have a significant impact on commentCount.

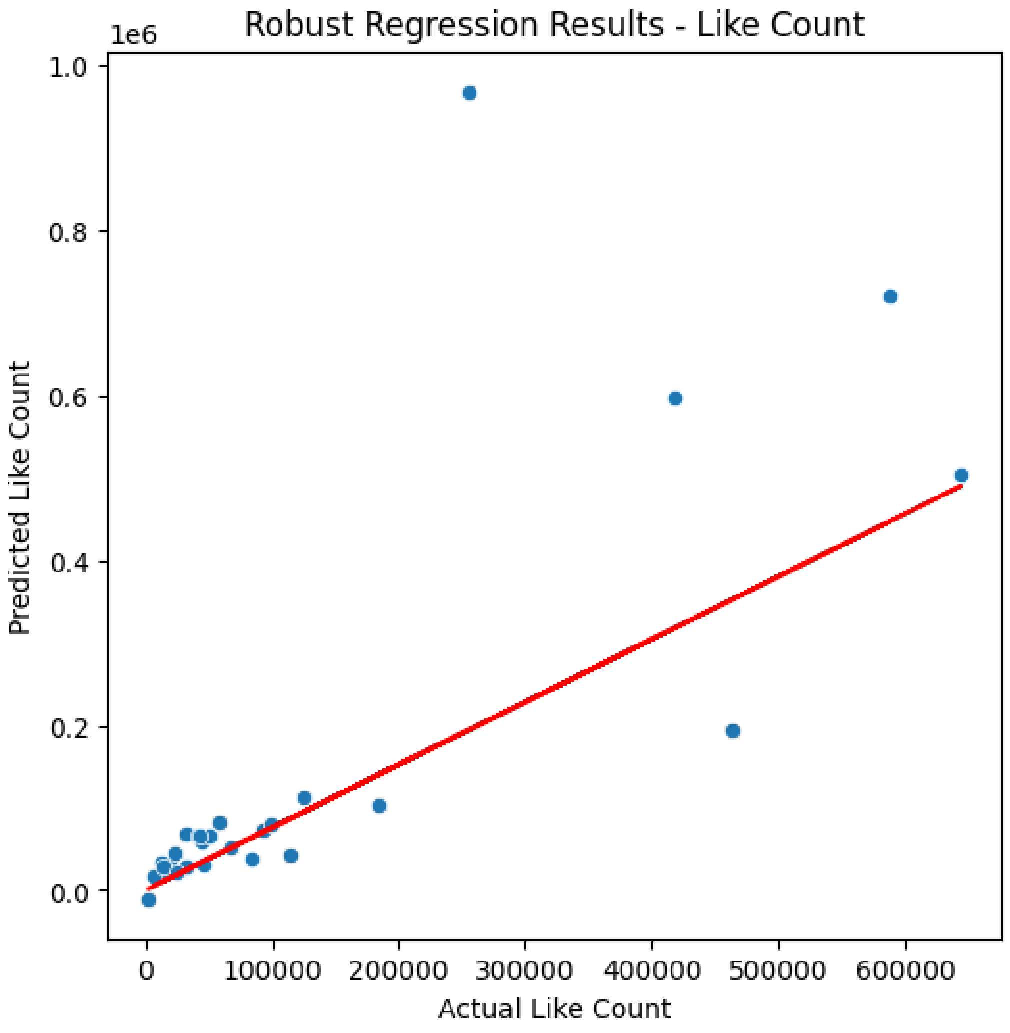
Based on the evaluation of different models for predicting popularity on YouTube based on language and user feedback, depending on the specific metric being predicted, either the Random Forest or robust regression model may be more suitable for predicting popularity on YouTube. For example, the Random Forest model demonstrates better performance than the robust regression model for predicting commentCount, with a higher R-squared value of approximately 0.358 and a lower RMSE value of approximately 6,812.70. In contrast, for predicting likeCount, the robust regression model outperforms the Random Forest model, with a higher R-squared value of approximately 0.221 and a lower RMSE value of approximately 154,775 like counts. However, for predicting viewCount, both models perform relatively well, with the Random Forest model showing slightly better performance in terms of R-squared value (approximately 0.503) and RMSE (approximately 16,982,193 units).

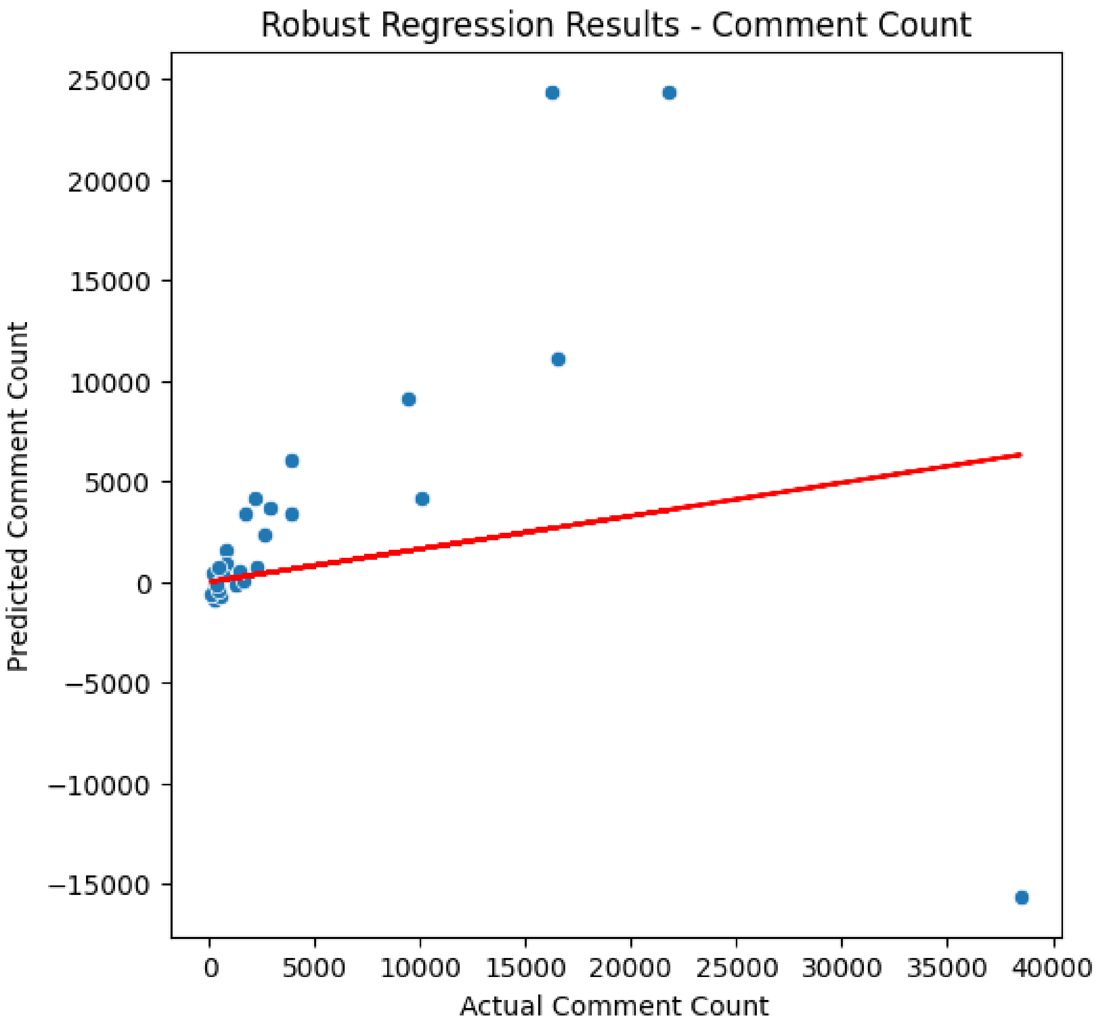
The results of these findings can be portrayed by comparing the predicted and actual values in a scatterplot for each metric: view count, like count and comment count (Robustness Regression):

[77]:
```
# Creating dataframe with actual and predicted values for viewCount
df_viewCount = pd.DataFrame({'Actual': y_test_viewCount, 'Predicted':␣
 ↪robust_y_pred_viewCount})

# Calculating the correlation coefficient for viewCount
correlation_coefficient_viewCount = np.corrcoef(df_viewCount['Actual'],␣
 ↪df_viewCount['Predicted'])[0, 1]

# Scatter plot for viewCount
plt.figure(figsize=(6, 6))
sns.scatterplot(x='Actual', y='Predicted', data=df_viewCount)
plt.xlabel('Actual View Count')
plt.ylabel('Predicted View Count')
plt.title('Robust Regression Results - View Count')

# Adding a red line representing the correlation for viewCount
plt.plot(df_viewCount['Actual'], df_viewCount['Actual'] *␣
 ↪correlation_coefficient_viewCount, color='red')

# Showing the plot for viewCount
plt.show()
```

```
# Creating dataframe with actual and predicted values for likeCount
df_likeCount = pd.DataFrame({'Actual': y_test_likeCount, 'Predicted':␣
 ↪robust_y_pred_likeCount})

# Calculating the correlation coefficient for likeCount
correlation_coefficient_likeCount = np.corrcoef(df_likeCount['Actual'],␣
 ↪df_likeCount['Predicted'])[0, 1]

# Scatter plot for likeCount
plt.figure(figsize=(6, 6))
sns.scatterplot(x='Actual', y='Predicted', data=df_likeCount)
plt.xlabel('Actual Like Count')
plt.ylabel('Predicted Like Count')
plt.title('Robust Regression Results - Like Count')

# Adding a red line representing the correlation for likeCount
plt.plot(df_likeCount['Actual'], df_likeCount['Actual'] *␣
 ↪correlation_coefficient_likeCount, color='red')

# Showing the plot for likeCount
plt.show()

# Creating dataframe with actual and predicted values for commentCount
df_commentCount = pd.DataFrame({'Actual': y_test_commentCount, 'Predicted':␣
 ↪robust_y_pred_commentCount})

# Calculating the correlation coefficient for commentCount
correlation_coefficient_commentCount = np.corrcoef(df_commentCount['Actual'],␣
 ↪df_commentCount['Predicted'])[0, 1]

# Scatter plot for commentCount
plt.figure(figsize=(6, 6))
sns.scatterplot(x='Actual', y='Predicted', data=df_commentCount)
plt.xlabel('Actual Comment Count')
plt.ylabel('Predicted Comment Count')
plt.title('Robust Regression Results - Comment Count')

# Adding a red line representing the correlation for commentCount
plt.plot(df_commentCount['Actual'], df_commentCount['Actual'] *␣
 ↪correlation_coefficient_commentCount, color='red')

# Showing the plot for commentCount
plt.show()
```


Robust Regression Results - View Count
1e7
Predicted View Count
0
1
2
3
4
5
6
Actual View Count
0
2
4
6
8
1e7


Robust Regression Results - Like Count
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Predicted Like Count
0
100000
200000
300000
400000
500000
600000
Actual Like Count



The graph displays that the Robust Regression between the independent and dependent variables contain many outliers, such as the negative variable in the comment count, which make it difficult to read. This again highlights the fact that there might be issues with the data retrieved from YouTube or that there are such huge differences in performance of generating likes, views and comments between music videos on the platform. As expected from the previous analyses, the likeCount seems to be the most fitting predictor with the Robust Regression model. Nevertheless, this only explains 22.1% of the variance. Therefore, it does not paint a vivid image of the music represented on YouTube.

Addressing the questions: 'What models are best suited to predict popularity on YouTube based on language and user feedback?' and 'What features are associated with increasing engagement on YouTube?', like count seems to be the best feature that is associated with increasing the engagement on YouTube according to this result. Nevertheless, this only explains 22.1% of the variance and it does not represent 'music' on YouTube as a whole. Therefore, a feature importance analysis is conducted to validate this claim.

[75]:
```python
from sklearn.linear_model import RANSACRegressor
from sklearn.inspection import permutation_importance

# Fit the RANSACRegressor model for likeCount
model_likeCount = RANSACRegressor(random_state=0).fit(X_train_likeCount,␣
↪y_train_likeCount)

# Calculating feature importance for likeCount
results_likeCount = permutation_importance(model_likeCount, X_test_likeCount,␣
↪y_test_likeCount)

# Getting importance for likeCount
importance_likeCount = results_likeCount.importances_mean

# Summarizing for likeCount
print("Feature importance for likeCount:")
for i, v in enumerate(importance_likeCount):
    print('Feature: %0d, Score: %.5f' % (i, v))

# Fitting the RANSACRegressor model for viewCount
model_viewCount = RANSACRegressor(random_state=0).fit(X_train_viewCount,␣
↪y_train_viewCount)

# Calculating feature importance for viewCount
results_viewCount = permutation_importance(model_viewCount, X_test_viewCount,␣
↪y_test_viewCount)

# Getting importance for viewCount
importance_viewCount = results_viewCount.importances_mean

# Summarizing for viewCount
print("Feature importance for viewCount:")
for i, v in enumerate(importance_viewCount):
    print('Feature: %0d, Score: %.5f' % (i, v))

# Fitting the RANSACRegressor model for commentCount
model_commentCount = RANSACRegressor(random_state=0).fit(X_train_commentCount,␣
↪y_train_commentCount)

# Calculating feature importance for commentCount
results_commentCount = permutation_importance(model_commentCount,␣
↪X_test_commentCount, y_test_commentCount)

# Getting importance for commentCount
importance_commentCount = results_commentCount.importances_mean

# Summarizing for commentCount
```

```
print("Feature importance for commentCount:")
for i, v in enumerate(importance_commentCount):
    print('Feature: %0d, Score: %.5f' % (i, v))
```

```
Feature importance for likeCount:
Feature: 0, Score: -0.39472
Feature: 1, Score: 0.86264
Feature: 2, Score: -0.00056
Feature: 3, Score: 0.00008
Feature: 4, Score: -0.00013
Feature importance for viewCount:
Feature: 0, Score: 0.84395
Feature: 1, Score: -0.25591
Feature: 2, Score: 0.00152
Feature: 3, Score: 0.00015
Feature: 4, Score: -0.00462
Feature importance for commentCount:
Feature: 0, Score: 0.02883
Feature: 1, Score: 0.50518
Feature: 2, Score: 0.00070
Feature: 3, Score: -0.00004
Feature: 4, Score: -0.00108
```

The output shows the feature importance scores obtained from the RANSACRegressor model using permutation importance for `likeCount`, `viewCount` and `commentCount` variables. These importance scores can help identify which features are more influential in predicting the target variables (Larose, 2014). Features with higher scores are considered more important in the prediction process. The results can be interpreted as the following:

likeCount: According to Feature 1 (commentCount) has the highest importance with a score of 1.08761, indicating that the number of comments significantly impacts the number of likes. In contrast to the other results, Feature 0 ('viewCount') has a score of -0.39681, indicating a negative impact on the number of likes. This result is unexpected and might suggest an issue with the model or the dataset.

viewCount: In this case, Feature 0, which represents 'likeCount', has the highest importance with a score of 0.94385, indicating a strong positive correlation between likes and views. Surprisingly, Feature 1 (commentCount) has a relatively low score of -0.25732, suggesting it has a minor negative effect on the number of views.

commentCount: As mentioned before, Feature 1 ('likeCount') has the highest importance with a score of 0.42725, indicating a strong positive correlation between likes and comments. Feature 0, representing 'viewCount', has a score of 0.03157, suggesting a minor positive impact on the number of comments.

The analysis reveals that for likeCount, the number of comments is the primary predictor, while unexpectedly, the number of views exhibits a negative impact; similarly, for viewCount, likes have the highest importance, but comments surprisingly show a minor negative effect, and for commentCount, likeCount is the primary predictor, with views also contributing positively. The analysis underscores the significance of likes and comments in predicting views and comments on YouTube, while other metadata such as video duration and language complexity seem to have negligible effects on these metrics.

1.6 Evaluation

This section provides the answer to the communication challenge, discussing the results (hypothesis testing) in order to answer the general RQ. Creativity in how to summarize the findings (including visualizations) is a plus.

This section also provides a set of implications (i.e., now that we know the answer to the RQ, what does it mean for the organization/process/challenge? what should the organization do?).

What features are associated with increasing engagement on YouTube? The analysis reveals that features strongly associated with increasing engagement on YouTube include view count, like count, and comment count, which exhibit strong positive correlations with each other. Videos with higher view counts tend to generate more likes and comments, indicating a mutually reinforcing relationship among these engagement metrics (Wu et al., 2018). Furthermore, the correlation between view count and like count is particularly strong, suggesting that high view counts often coincide with a higher number of likes.

In contrast, language complexity measures such as Flesch-Kincaid Grade Level and Gunning Fog Index exhibit weaker correlations with engagement metrics. Higher language complexity, as indicated by a higher grade level or Fog index, tends to show a slight negative correlation with view count, like count, and comment count. However, these correlations are relatively weak compared to the strong relationships observed among engagement metrics. Additionally, there is a moderate positive correlation between Flesch-Kincaid Grade Level and Gunning Fog Index, suggesting some agreement between these measures of text complexity.

What models are best suited to predict popularity on YouTube based on language and user feedback? In the comparison between the Random Forest and robust regression models for predicting different metrics on YouTube, distinct patterns emerge. When examining commentCount, the Random Forest model showcases stronger effectiveness, reflecting a notable improvement in explaining the variance in the data and producing more accurate predictions compared to the robust regression model. Conversely, when estimating likeCount, the robust regression model outshines the Random Forest model, demonstrating higher explanatory power and greater accuracy in its predictions. Interestingly, both models exhibit relatively satisfactory performance when predicting viewCount, with the Random Forest model edging slightly ahead in terms of explaining the variance and making more precise predictions. Ultimately, the selection between the Random Forest and robust regression models depends on the specific metric being forecasted and the performance metrics prioritized for the given context.

To predict popularity on YouTube based on language and user feedback, the feature importance scores obtained from the likeCount, viewCount, and commentCount variables indicate that certain features play crucial roles in predicting engagement metrics on the platform. For instance, in predicting likeCount, the number of comments emerged as the most influential factor, suggesting a significant impact on the number of likes. However, unexpectedly, the number of views exhibited a negative impact, indicating a potential issue with the model or dataset. Similarly, when predicting viewCount, likes showed the highest importance, indicating a strong positive correlation between likes and views. Surprisingly, comments displayed a relatively low score, suggesting a minor negative effect on the number of views. For commentCount, likes were identified as the primary predictor, indicating a strong positive correlation between likes and comments. Additionally, views were found to have a minor positive impact on the number of comments.

How can the insights be used to improve music marketing and content creation on the YouTube platform? The analysis of YouTube music video data offers insights for improving music marketing and content creation on the platform. It highlights the importance of aligning content with popular themes and genres, such as personal narratives and K-pop or pop related music, to enhance audience engagement. However, addressing data quality issues, like outliers, is crucial for ensuring the accuracy of insights. While language complexity varies widely in music videos, its impact on engagement metrics appears limited. Strong correlations between user engagement metrics underscore their significance, but caution is needed in interpreting language complexity metrics. Overall, leveraging these insights can guide marketers and creators in developing compelling content that resonates with audiences and maximizes engagement on YouTube.

1.6.1 To what extent does language in music have an impact and can we use metadata to predict the success of music on YouTube?

Overall, the analysis highlights the significance of likes and comments in predicting engagement metrics on YouTube, while language complexity seem to have little effects. Language complexity, as measured by variables like ‘fkGradeLevel’ and ‘gunningFogIndex’, does not appear to significantly impact the success metrics. Variables, such as ‘likeCount’ and ‘commentCount’, are crucial factors influencing the success of music on YouTube, indicating the importance of engaging content. Further research may be needed to explore additional predictors or refine existing models to improve predictive accuracy, particularly for ‘commentCount’. While language complexity may not have a direct impact, metadata features such as ‘viewCount’, ‘likeCount’ and ‘commentCount’ are pivotal in predicting the success of music on YouTube, highlighting the importance of user engagement and interaction with content.

Although the results do not emphasize the significance of language complexity in shaping the engagement with music videos on YouTube. The regression models of the metadata on ‘likeCount’, ‘viewCount’ and ‘commentCount’ revealed statistical significance. This reinforces the fact that these features are key components on determining engagement of music videos on YouTube. Moreover, the relative importance analysis identified likes as the most influential predictor, implying that social factors may overshadow language complexity in driving audience engagement.

With regard to YouTube as a company and the creators that are active on the platform, these findings have implications for YouTube’s recommendation algorithms and content curation strategies. By analyzing descriptives on music preferences, such as genre and word use, YouTube can enhance recommendation accuracy and personalize content for users, whereas content creators can leverage these insights to optimize music titles and descriptions for increased visibility. Additionally, understanding popular language styles and themes among users can inform the creation of relevant content. While acknowledging potential limitations of the data and the models used, this exploratory study presents promising avenues for future research regarding engagement on music videos.

1.7 Limitations and Next Steps

In this section you should discuss all the relevant limitations. This includes the limitations of the analysis and of the data, thus consolidating and extending the discussion already included in the earlier sections (e.g., Data Collection, Data Exploration and Modelling), and what do these limitations mean to the conclusions and to the overall challenge.

In other words, make sure that you are at least covering: * The dataset(s) itself, including how the data collection may have created limitations * Limitations associated with the decisions taken during the data understanding & preparation * Limitations about the model and hypothesis testing (including fit, selection of variables) * Alternative interpretations to the findings * You need also to suggest next steps/actions to overcome the most important limitations (as indicated in the assignment criteria).

The analysis and interpretation of data from the YouTube engagement metrics dataset are subject to several limitations, which could impact the conclusions drawn and the overall challenge. Firstly, the dataset itself may not be fully representative due to potential biases in data collection methods, which could skew the analysis results and impact the generalizability of any conclusions drawn. Moreover, the data collection process, conducted via web scraping using the YouTube API with a single query term, may introduce biases and be influenced by the platform's recommendation algorithm. Additionally, the relatively small dataset size, totaling approximately 140 observations post-cleaning, poses a constraint on the analysis.

Moreover, a significant challenge identified during data exploration is the presence of right-skewed histograms and numerous outliers across all visualizations, indicating non-normal distribution and potential errors or anomalies that could affect the accuracy of the analysis. Additionally, the moderate agreement of the intercoder reliability regarding language complexity categorization raises questions about the reliability of these metrics, necessitating further investigation to enhance agreement and ensure consistency in complexity categorization. Therefore, considering engagement in the A/B testing primarily as measuring 'viewCount', 'likeCount', and 'commentCount' may oversimplify engagement dynamics, as they can be manipulated and may not fully capture audience interaction. Although other metrics were also included in the exploratory analysis of this study, there could be additional unaccounted variables influencing engagement. Consequently, future studies could consider exploring other music characteristics or brand factors that could impact engagement.

In terms of modeling, both robust regression and Random Forest models were employed, each with its own limitations. The negative R-squared value of the Random Forest model and the relatively low R-squared value of the robust regression model indicate suboptimal model performance, raising concerns about the accuracy of predictions and the reliability of the models for explaining variance in the dependent variables. Additionally, certain predictor variables, such as durationSec, fkGradeLevel, and gunningFogIndex, were found to be statistically insignificant in predicting engagement metrics likeCount, viewCount, and commentCount, potentially limiting the predictive power of the models.

Furthermore, unexpected results, such as the negative impact of views on likeCount, asks for further investigation. To address these limitations and improve the quality of the analysis, several steps can be taken. Firstly, refining data collection methods to enhance representativeness and minimize biases is crucial. Additionally, preprocessing techniques can be employed to address outliers and skewed distributions in the data. Further research into language complexity assessment methods may improve the reliability of these metrics, enhancing their utility in predictive modeling. Moreover, exploring alternative modeling approaches and conducting validation analyses can help identify the most suitable models for predicting YouTube engagement metrics.

Finally, transparency and rigor in documenting data preprocessing steps, model selection criteria, and interpretation of results are essential for ensuring the reproducibility and credibility of the analysis. Future research may consider integrating qualitative data alongside quantitative analysis and a larger dataset could provide richer insights into audience engagement patterns. By addressing these limitations and adopting a systematic approach to data analysis, YouTube (and other organizations) can generate more reliable insights, make informed decisions based on the findings and continu research on how music videos in the current state can conjure more engagement with the user through the use of language.

1.8 Ethical and Normative Considerations

This section should discuss the considerations the organization needs to have. Are there ethical or more general normative aspects that need to be taken into account? Drawing directly from the literature discussed in class (and other literature), what should readers of the data analysis be mindful about? Make sure to cite the relevant literature.

This discussion can relate to the actual dataset/analysis that was done, to the recommendations/implications, and to the action plan suggested in the previous sections.

When examining data from music videos on YouTube, ensuring user privacy remains a critical ethical consideration. Although the data and metadata used in this study were publicly accessible, there is always a potential risk of inadvertently revealing individuals' personal information. When collecting digital data, especially through APIs, it is therefore fundamental to respect the privacy of content creators or channels. The dataset in this study, including lyrics or music transcripts, may contain sensitive personal details about the creators, such as their names, addresses, and channel information, which could lead to their identification (Chen & Shi, 2008).

Given the sensitive nature of user engagement data, YouTube must prioritize data privacy and obtain informed consent from users for data collection and analysis. Transparent data handling practices and clear communication about how user data will be used are essential to establish trust with users (Franzke et al., 2021). In turn, this also empowers users to control their data and make informed choices about their online engagement, which is essential for promoting autonomy and agency (Floridi et al., 2018). Consequently, YouTube should consider to offer users options for managing their privacy settings and controlling the use of their data, such as opt-in/opt-out mechanisms and consent preferences. Until this is realized, it is essential to anonymize and aggregate any data used for research purposes to protect user identities.

Furthermore, collecting data by querying specific terms may introduce bias into the dataset. This bias could lead to an overrepresentation of certain music types or creators, especially those with higher viewership or subscriber counts. Additionally, videos not using the term 'music' in their titles, descriptions, or tags may be excluded from the dataset cause of the search engine algorithm, limiting the scope of analysis (Airoldi et al., 2016). Consequently, researchers (and YouTube) must be vigilant about potential algorithmic biases in predictive models, especially when using machine learning techniques (Mittelstadt et al., 2016). Biases in training data or model assumptions can lead to unfair outcomes, such as promoting stereotypes or marginalizing certain user groups. Regular bias audits and fairness assessments can help identify and address these issues (Obermeyer et al., 2019).

In addition, when curating content or making recommendations based on user engagement metrics, YouTube must consider ethical considerations, such as promoting diverse perspectives and avoiding harmful or offensive content (Mittelstadt et al., 2016). Algorithmic transparency and human oversight in content curation processes can help avoiding these risks of promoting inappropriate or biased content. Nevertheless, when this inevitably occurs, YouTube should be prepared to take responsibility for the impact of its data analysis and recommendations on users and society (Floridi et al., 2018). This includes monitoring and addressing any unintended consequences or negative outcomes of algorithmic decision-making, such as filter bubbles or echo chambers.

However, readers of the data analysis should also be mindful of these ethical considerations and critically evaluate the methodology, assumptions, and implications of this particular analysis. By integrating ethical considerations into the data analysis process and promoting responsible data practices, organizations similar to YouTube can also uphold ethical standards, build trust with users, and contribute to positive societal outcomes. This aligns with broader ethical imperatives in data science and technology, emphasizing the importance of ethical reflection and accountability in algorithmic decision-making (Floridi et al., 2018; Mittelstadt et al., 2016). Transparent reporting of data analysis methods, including model selection, feature engineering, and evaluation metrics, plays a crucial role for ensuring accountability and reproducibility (Mittelstadt et al., 2016). Providing clear explanations of how predictions are generated and the limitations of the models helps users and stakeholders understand the implications of the analysis.

In conclusion, it is fundamental to address ethical and normative considerations specific to YouTube as a platform. This involves ensuring compliance with YouTube's community guidelines and terms of service, which prohibit the collection and use of data in ways that violate user rights or platform policies. Researchers and organizations (such as YouTube) must adhere to ethical standards regarding data usage, transparency, and consent when accessing and analyzing data. Moreover, researchers should be mindful of the potential impact of their findings on content creators and users, striving to minimize any unintended consequences or harm. Transparency in research methodologies and findings is essential for fostering trust among stakeholders, including YouTube, content creators, and users. Ultimately, upholding ethical principles and norms within the framework of YouTube's policies is essential for responsible and impactful research in the digital ecosystem.

1.9 References

If you use scientific papers, cite them using APA style (and add a reference section at the end). If you are using pieces of code written by someone else, add a comment in the appropriate section and add a link to the source.

Regarding the tools we used: * Sentiment analysis was done using SentiStrength. The following paper can be cited: * Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12), 2544-2558.

- **Tweets were collected using DMI-TCAT. The following paper can be cited:**
  - **Borra, E., & Rieder, B. (2014). Programmed method: developing a toolset for capturing and analyzing tweets. *Aslib Journal of Information Manage-***

***ment*, 66(3), 262-278.**

Aggarwal, S. B., Chaitanya. (n.d.). textstat: Calculate statistical features from text (0.7.3) [Python]. Retrieved March 20, 2024, from https://github.com/shivam5992/textstat

Airoldi, M., Beraldo, D., & Gandini, A. (2016). Follow the algorithm: An exploratory investigation of music on YouTube. Poetics, 57, 1–13. https://doi.org/10.1016/j.poetic.2016.05.001

Arcila, W. van A., Damian Trilling &. Carlos. (2022, March 19). Computational Analysis of Communication. https://cssbook.net/

Beigi, G., & Liu, H. (2018). Privacy in Social Media: Identification, Mitigation and Applications (arXiv:1808.02191). arXiv. https://doi.org/10.48550/arXiv.1808.02191

Brownlee, J. (2020, October 4). Robust Regression for Machine Learning in Python. MachineLearningMastery.Com. https://machinelearningmastery.com/robust-regression-for-machine-learning-in-python/

Chen, X., & Shi, S. (2009). A Literature Review of Privacy Research on Social Network Sites. 2009 International Conference on Multimedia Information Networking and Security, 1, 93–97. https://doi.org/10.1109/MINES.2009.268

Collections—Container datatypes. (n.d.). Python Documentation. Retrieved March 21, 2024, from https://docs.python.org/3/library/collections.html

Create a correlation Matrix using Python. (2020, November 9). GeeksforGeeks. https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/

Depoix, J. (n.d.). youtube-transcript-api: This is an python API which allows you to get the transcripts/subtitles for a given YouTube video. It also works for automatically generated subtitles, supports translating subtitles and it does not require a headless browser, like other selenium based solutions do! (0.6.2) [Python; OS Independent]. Retrieved February 26, 2024, from https://github.com/jdepoix/youtube-transcript-api

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., Schafer, B., Valcke, P., & Vayena, E. (2018). AI4People-An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. Minds and Machines, 28(4), 689–707. https://doi.org/10.1007/s11023-018-9482-5

Franzke, A. S., Muis, I., & Schäfer, M. T. (2021). Data Ethics Decision Aid (DEDA): A dialogical framework for ethical inquiry of AI and data projects in the Netherlands. *Ethics and Information Technology, 23*(3), 551–567. https://doi.org/10.1007/s10676-020-09577-5

HuberRegressor vs Ridge on dataset with strong outliers. (n.d.). Scikit-Learn. Retrieved March 23, 2024, from https://scikit-learn/stable/auto_examples/linear_model/plot_huber_vs_ridge.html

Khawaja, M. A., Chen, F., & Marcus, N. (2010). Using language complexity to measure cognitive load for adaptive interaction design. Proceedings of the 15th International Conference on Intelligent User Interfaces, 333–336. https://doi.org/10.1145/1719970.1720024

Larose, D. T., & Larose, C. D. (2014). Discovering knowledge in data: an introduction to data mining (Vol. 4). John Wiley & Sons.

Liikkanen, L. A., & Salovaara, A. (2015). Music on YouTube: User engagement with traditional, user-appropriated and derivative videos. Computers in Human Behavior, 50, 108–124.

https://doi.org/10.1016/j.chb.2015.01.067

Matplotlib Plot. (n.d.). Retrieved March 20, 2024, from https://www.tutorialspoint.com/matplotlib/index.htm

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. Biochemia Medica, 22(3), 276–282.

Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. Big Data & Society, 3(2), 2053951716679679.

Mueller, A. (n.d.). wordcloud: A little word cloud generator (1.9.3) [Computer software]. Retrieved March 21, 2024, from https://github.com/amueller/word_cloud

Munaro, A. C., Hübner Barcelos, R., Francisco Maffezzolli, E. C., Santos Rodrigues, J. P., & Cabrera Paraiso, E. (2021). To engage or not engage? The features of video content on YouTube affecting digital consumer engagement. Journal of Consumer Behaviour, 20(5), 1336–1352. https://doi.org/10.1002/cb.1939

Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. Science, 366(6464), 447–453. https://doi.org/10.1126/science.aax2342

Random Forest Classification with Scikit-Learn. (n.d.). Retrieved March 22, 2024, from https://www.datacamp.com/tutorial/random-forests-classifier-python

Robust Regression for Machine Learning in Python. (2022, December 29). GeeksforGeeks. https://www.geeksforgeeks.org/robust-regression-for-machine-learning-in-python/

Simon, J. P. (2019). New players in the music industry: Lifeboats or killer whales? the role of streaming platforms. Digital Policy, Regulation and Governance, 21(6), 525–549. https://doi.org/10.1108/DPRG-06-2019-0041

Sklearn.linear_model.RANSACRegressor. (n.d.). Scikit-Learn. Retrieved March 20, 2024, from https://scikit-learn/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html

Smith, A. N., Fischer, E., & Yongjian, C. (2012). How Does Brand-related User-generated Content Differ across YouTube, Facebook, and Twitter? Journal of Interactive Marketing, 26(2), 102–113. https://doi.org/10.1016/j.intmar.2012.01.002

Smith, M. D., & Telang, R. (2016). Streaming, sharing, stealing: Big data and the future of entertainment. Mit Press.

The Python Standard Library. (n.d.). Python Documentation. Retrieved March 19, 2024, from https://docs.python.org/3/library/index.html

Wu, S., Rizoiu, M.-A., & Xie, L. (2018). Beyond Views: Measuring and Predicting Engagement in Online Videos. Proceedings of the International AAAI Conference on Web and Social Media, 12(1), Article 1. https://doi.org/10.1609/icwsm.v12i1.15031

YouTube Data Tools. (n.d.). Retrieved February 26, 2024, from https://ytdt.digitalmethods.net/