# Improving the User Experience of YouTube by Increasing Available Video Information

Kunal Dilip Indore Larson Ost Adwait Patil Smit Shah

December 11th, 2022

## 1 Summary

YouTube is the leading video platform with over 2 billion monthly active users and over 200,000 hours of content uploaded each day [1]. Visitors to the platform are able to browse videos through a basic search bar, keywords, and subscriptions, however, this is still largely subject to the platform's search algorithm. Consequentially, we feel that the important components of a video's features are neglected during the search, namely sentiment and controversy. For example, in November of 2021 YouTube removed video dislikes that prevent the user from seeing the proportion of positive and negative engagement. Also, videos that seed controversy can often become popular; however, the user does not have a way to filter out this type of content. In this project, we will introduce new video information to empower the user to have a healthier video browsing experience by further refining their search.

There is an abundant amount of videos constantly uploaded to YouTube. Therefore for this project, we are going to be focusing solely on Trending YouTube videos [2]. Trending videos would benefit the most from receiving additional information because they have the most engagement. More specifically, our dataset consists of over 40,000 trending YouTube videos in the United States from several months in 2017. Basics such as the video ID, video title, and channel title are included. There is time-relevant information such as the date it was published and the date it was trending. Also, there are some additional metrics such as video keywords, likes, dislikes, and views.

The goal of this project is to increase the number of video parameters available for search. We will be introducing video sentiment, dislikes, controversy, and video categories. Sentiment will be added by a third-party sentiment analysis platform trained to recognize the positivity and negativity in a sentence. Video sentiment will be based on the title. As previously mentioned, YouTube has removed dislikes in 2021, however, our dataset is from 2017 and contains dislikes information. We will use this to create a regression model to predict video dislikes. As an extension, we will take the proportion of dislikes to likes to create a simple measure for controversy. We will also create a classification model to predict the video's category to ensure no videos go unlabeled. Exploratory data analysis and visualization will be performed to find any interesting insights into the refined dataset. Finally, we will present a demonstration of what the new search features will look like through a searchable user interface.

# 2 Methods

## 2.1 Preprocessing

Sentiment analysis is a technique that detects the underlying sentiment in a piece of text. It uses machine learning techniques to classify text as having either a positive, negative, or neutral sentiment. Sentiment analysis is essential for processing large amounts of data without manual intervention. We are using this analysis to get the sentiment of YouTube video titles because that is one of the first things any user sees when deciding whether the video is interesting enough to watch. Since the data we have is about trending videos, we would like to see the sentiments of these titles and analyze them further to get some interesting insights.

We have a structured dataset that includes the video title, channel title, publish time, tags, views, likes, dislikes, description, and comment count. The data also includes a category_id field, which varies between regions. We can retrieve the categories for a specific video, in the associated JSON file. The category from the JSON file was mapped to the category id column in our dataset. After doing sentiment analysis on the dataset's "title" column, the "Title Sentiment" and "Title Sentiment values" columns are formed. We use quality measurement processes to eliminate duplication and missing values and drop columns that are redundant.

To generate sentiments for the title of each video in our dataset we are using the Hugging Face library, a little about hugging face: it has built and trained huge data sets related to sentiment analysis for Twitter, Reddit, and various data sets and built models which we call the pre-trained models. We are using 'federicopascual/finetuning-sentiment-model-3000-samples' model to generate sentiments [3]. We will import the hugging face model from the transformer library and use the model. As an example, if we feed it the sentences "I love this movie", Hugging face will return "Label: LABEL_1, Score: 0.9558". If we give it the sentence "This movie sucks!", it will return "Label: LABEL_0, Score: 0.9413". We just convert the sentence to 1 if it is positive i.e. if the label is label_1, and 0 for the sentence with negative sentiment.

Furthermore, we have utilized two variables of type date i.e "trending_date" and "publish_time" to calculate the time taken by videos to trend after they are published.

Now before feeding the data to the model, we have to clean the data i.e. tidy the title. We will remove things that will not contribute to calculating sentiments such as links, emojis, stop-words (except negative words such as don't, wouldn't, etc because those contribute to the sentiment), punctuation, and finally lemmatize the words. The code used to perform the tidying in the appendix labeled "Python code used to generate title sentiment and tidy the text data".

## 2.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the process of generating summary statistics for numerical data in a dataset and constructing various graphical representations to better comprehend the data. Before we begin exploring the dataset, we should understand the attributes present in the dataset and the data type of those attributes.

### 2.2.1 Glimpse of the dataset and visualizations

From the Appendix (7.3 table 1 and figure 10), we can see an overview of the dataset along with all the attributes and their data types.

Below are the numerical variables present in the dataset:

- Category Id
- Views
- Likes
- Dislikes
- Comment Count
- Title Sentiment Values

- We computed the correlation between all variables and found that likes, dislikes, and comment counts are all positively correlated with one another (see Appendix: 7.3 figure 16).
- We visualized the distributions of the positively associated numerical variables to learn more about them. (see Appendix: 7.3 figure 11-12) Views, likes, dislikes, and comment counts all have a normal distribution. It is also worth noting that the number of likes outnumbers the number of dislikes. The average trending video has 1,000,000 views, 10,000 likes, 1,000 dislikes, and 2,000 comments.
- Furthermore, we can see that the category with the most videos is "Entertainment", followed by "Music", "How to & Style", and so on. This is supported by the plot in figure 2 (Sum of views of top 20 channels), which shows that the majority of the channels fall into the "Music" or "Entertainment" categories.

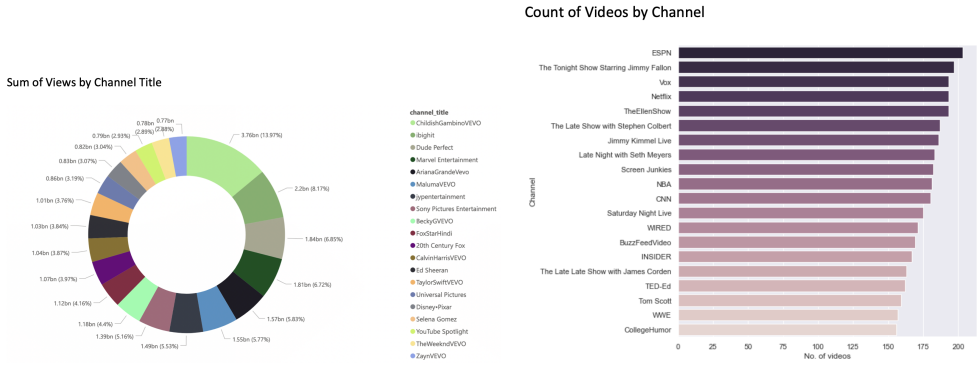


Figure 1: Top 20 channels by the count of videos and views

- Figure 2 (Count of videos per channel) shows that the top 20 channels with the most videos posted include "The Tonight Show Starring Jimmy Fallon", "TheEllenshow", and several channels in the category "Show". Despite the fact that visualization depicting "count of videos by category" (see Appendix: 7.3 figure 12) indicates that the category "Show" has the fewest videos.

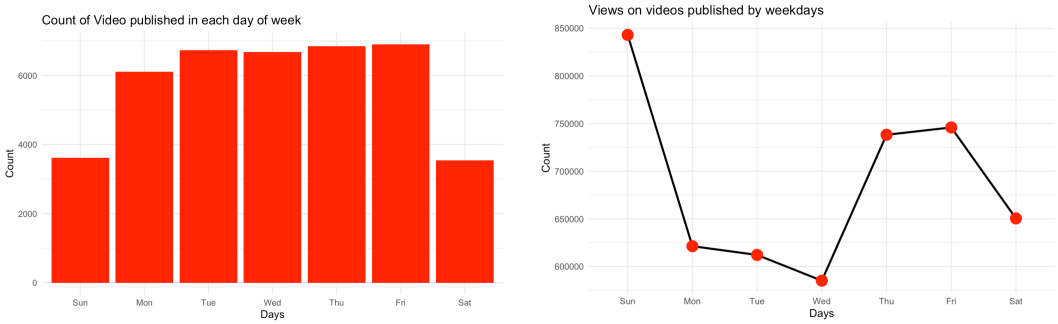


Figure 2: Count of videos published in each month and days of week

- We visualized the number of videos posted on each day of the week, here we see that the least number of videos are posted on Sunday. Though the number of videos posted on Sunday is the lowest, there is a chance that the most views will be gained on Sunday.

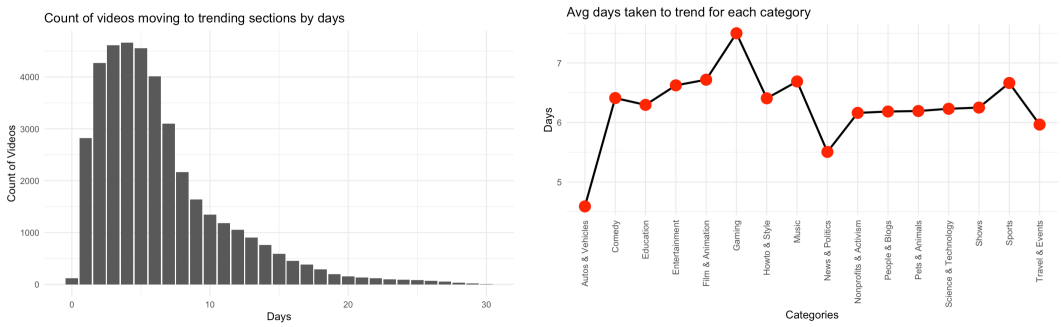


Figure 3: Visualizing videos moving to trending sections by days

- We computed the number of days a video takes to trend using the published date and trending date variables and plotted visualization to gain insights into the days videos take to trend after they are published. Most videos begin to trend within 2 to 10 days of their first publishing.
- We also plotted the time to trend for each category and discovered that videos in the "Autos & vehicles" category require less than 5 days to trend. However, videos in the category "Gaming" take longer than 7 days to trend.
- As mentioned in preprocessing, to get more robust insights from the data we are using the sentiment column introduced in the dataset following sentiment analysis of video titles to understand how sentiment plays a critical role. It can be seen that there are around 22,000 videos with positive sentiment and approximately 18,000 videos with negative sentiment of titles. (see Appendix: 7.3 figure 14)
- It was also observed that videos with good sentiments earn more views and likes, whilst videos with negative sentiments receive more dislikes and comments. It can be deduced that individuals tend to comment more on videos with bad titles. (see Appendix: 7.3 figure 15)
- We displayed the percentage of videos with negative sentiment in the title for each category in the visualization below. It may be extrapolated that when the category is "Nonprofit & Activism," the likelihood of a negative sentiment value is 1.36 times greater than in all other categories.

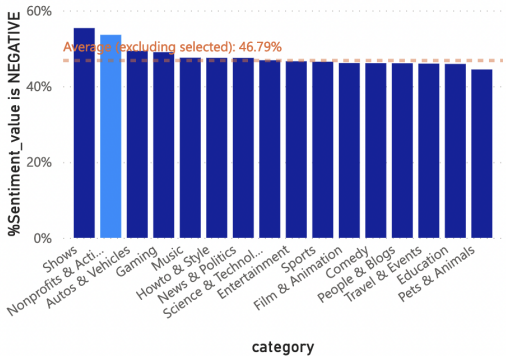


Figure 4: Percentage of negative sentiment value by categories

#### 2.2.2 Building Category Classification Model

The original dataset lists over 40 possible video categories. Many categories have very few or no videos within them, therefore, we will only be considering videos in the top 10 most popular categories so we have an adequate amount of data for analysis. The video categories are Film & Animation, Music, Sports, People & Blogs, Comedy, Entertainment, News & Politics, Howto & Style, and Science & Technology

Now we move to the next step: using the categories as a predictive variable and building a model that predicts the video category. This will help us in categorizing unlabeled videos, which can prove to be a very useful feature for available video criteria for the user when browsing YouTube. Once we have the title sentiments, we will remove the title from the dataset along with some other variable such as, trending time, tags, and description as these do not contribute that much in deciding the category of the data. We do some preprocessing such as converting the publish_time to either morning, afternoon, evening or night. Interestingly, we found that most trending videos were published in the afternoon when we would expect either evening or night to have more trending videos. Apart from this we also onehot encoded the text data to feed it to the machine learning model.

We will apply Random Forest algorithm on this data set with a train test split of 75% and 25% respectively, we achieve a good accuracy with all these transformations and this algorithm which we will see in the result section.

### 2.3 Predicting likes andd dislikes

As discussed in the summary, the primary objective of this project is to increase the number of parameters available for searching a video. A very important parameter of the search bar is controversy which is a ratio of likes and dislikes. As dislikes is not publicly available on Youtube now, we developed a regression model to predict dislikes.

The model is a RandomForest Regressor with n_estimators as 100. This value of estimators was derived using cross-validation. As the data set contained categorical features like comments_disabled,ratings_disable, and video_removed the first task was to create a one-hot encoded column for these features. Once the features were ready the next part was creating a train test split, we used a 70 to 30 split for train and test. The next step was feature selection, we did this using step-wise selection with root mean squared error (rmse) as the criteria for selecting a subset. The selected variables were title sentiments, comments disabled, ratings disabled and video removed or error. This shows that we can calculate dislikes even if a channel keeps its dislikes hidden.

The likes model was developed for getting the values of likes in cases where they are kept hidden. After step-wise feature selection we observed that title sentiments, views , comment count and dislikes gave the best rmse. This also assures that incase where likes are not available we can use the dislikes model to get dislikes and then use those values to predict the likes.

## 3 Results

### 3.1 Categories Prediction

We aim to categorize the unlabelled videos so that the search experience of the user can improve significantly, as they will be able to see videos of categories they like, which they weren't able to see initially as these videos were not labeled. We use the Random Forest algorithm and we have chosen this after using various algorithms and found this to work the best for this data set. We achieve an accuracy of 94.5 %. This algorithm requires a parameter called n_estimator which is a number of decision trees used by the model and after experimenting, the best results were achieved when n_estimators is equal to 100. Now to further see how well our model is

performing we take a look at precision, recall, and f1-score for the categories. We observe that we have achieved great results for each one of them. Let's take a look at the classification report.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.92 | 0.95 | 301 |
| 1 | 0.99 | 0.97 | 0.98 | 278 |
| 2 | 0.99 | 0.97 | 0.98 | 279 |
| 3 | 0.97 | 0.94 | 0.96 | 300 |
| 4 | 0.99 | 0.92 | 0.95 | 558 |
| 5 | 0.91 | 0.99 | 0.95 | 1280 |
| 6 | 0.97 | 0.96 | 0.96 | 374 |
| 7 | 0.95 | 0.94 | 0.95 | 532 |
| 8 | 0.99 | 0.92 | 0.96 | 288 |
| accuracy | | | 0.95 | 4190 |

Figure 5: Classification Report

## 3.2 Model Performance

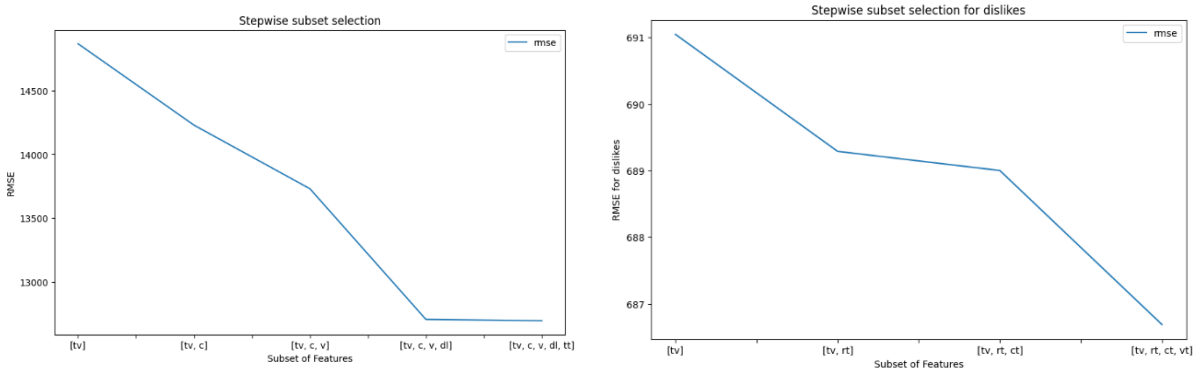


Figure 6: Performance of the likes and dislikes models on each subset (Abbreviations for feature names available in Appendix)

The two models were created with the aim to get the approximate values of likes and dislikes for any video posted on Youtube. The dislike model gave a rmse of 686 using the best features, as the dislikes are in the range of $10^3$ we concluded that this model was a good fit for predicting dislikes. The likes model on other hand gave a rmse of 12,500 using four features. We decided not to include the fifth feature as rmse didn't improve after adding it. Given that the likes were in the range of $10^5$ we deemed this model to be a good fit for predicting likes.

## 3.3 Demonstration of the new video search

With all the newly created parameters for a video, we are able to demonstrate a concept for what the new video search's will be like. We have created an R Shiny application that includes the following parameters for searches: video channel, category, sentiment, controversy, and views. The following is an image of the new interface:

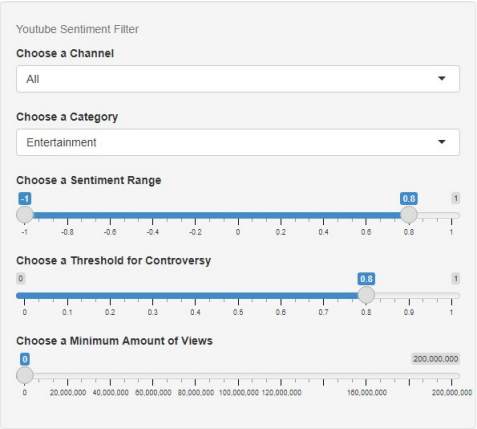


Figure 7: Image of the search panel.

The benefit of this section will be demonstrated by searching within the "News and Politics" category of videos. This category is typically very negative and overstates the bad news for the purposes of getting more engagement. We would like for our program to be able to filter out the negative aspects of the bad news and include only the positive.

When setting the accepted sentiment in the range 0.9 - 1.0, setting the acceptable controversy down to 0.1, and filtering for the "News and Politics" section, some of the videos in the top search results are now "VeteransDay: Thank You for Everything", "funfetti is extremely fun", and "Lin-Manuel Miranda's next act: Helping rebuild Puerto Rico". The positivity is further exemplified by the following plot that displays the count of the fifteen most common words:

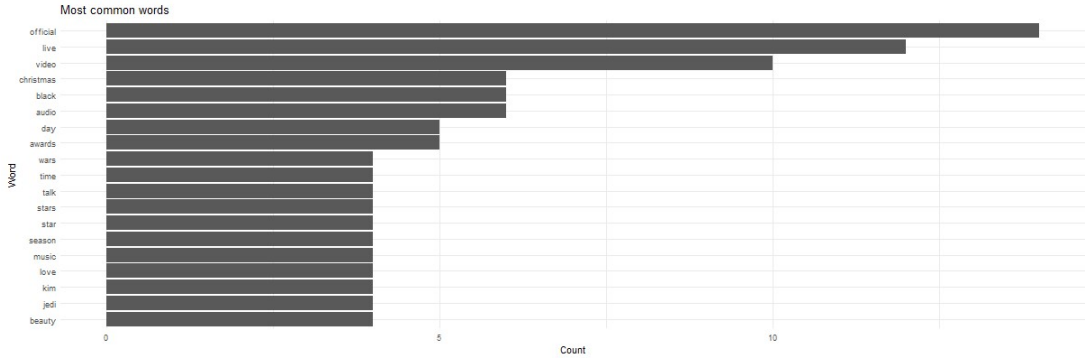


Figure 8: Count of the 15 most common words in the News and Politics section for current filter

Words such as "Christmas", "awards", and "love" appear as some of the most common results ("Wars" is only common due to the prevalence of Star Wars content). We find that this demonstrates the program accurately filtered out the negative content.

Furthermore, we filtered for the sentiment to be in the range of -1 to -0.9. We also allowed for the controversy to have no maximum. Some of the top titles in this search are "Which Countries Are About To Collapse?", "Iraq-Iran earthquake: Deadly tremor hits border region", and "Train Swipes Parked Vehicle". The negativity is once again shown using the following word count plot:

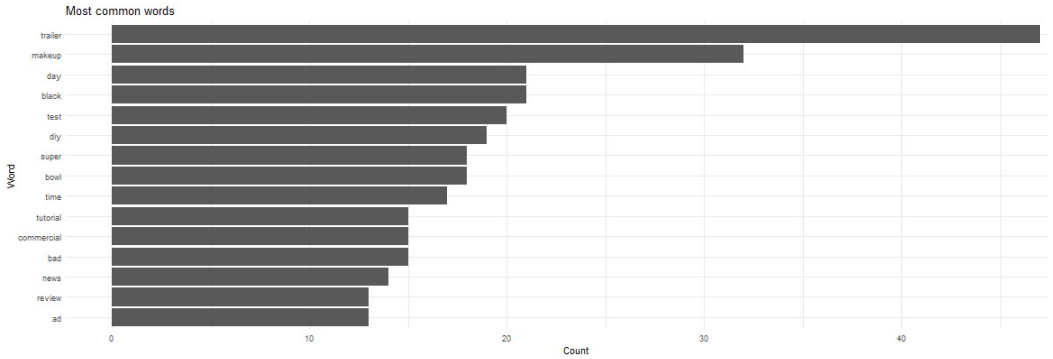


Figure 9: Count of the 15 most common words in the News and Politics section for current filter

Words such as "Bad", "ad", and "commercial" now appear. Also, the positive words are now removed. We find that this new search algorithm properly adds the additional search parameters created throughout this project.

# 4 Discussion

Our project has created the methodology to add new parameters to a YouTube video search and therefore enhances the user's ability to browse content. This will allow the user an option to have a healthier browsing experience by displaying all necessary information about a video before engaging with it, including its category, controversy, and sentiment. Previously, people browsing YouTube were subject to seeing whatever the underlying algorithm suggested with the inability to decide what atmosphere of videos they would like to be exposed to. This could lead to the user feeling overwhelmed by videos related to saddening news, negative click-bait, or controversial videos. These may be eye-catching to many others and receive a lot of engagement, but there is no metric for how "healthy" it is to be consistently exposed to it.

We have been able to demonstrate through our R Shiny application a glimpse into what it is like to search with these extra parameters. By filtering out content with negative sentiment and high controversy, we have shown we can protect the user from being bombarded by negative videos. This will benefit anyone who feels overwhelmed at the prospect of being exposed to negative videos online by allowing them to make better-informed decisions about the content they watch.

For future work, this methodology could be applied to a larger set of content that does not only involve trending YouTube videos. Also, the underlying sentiment analysis model was created by a third party to understand the sentiment of sentences. YouTube videos are unique and the words in the title may not represent the sentiment of the video. For example, music videos usually have titles that would not make sense as a standalone sentence. The underlying model could be improved to account for outliers like these.

# 5 Statement of contributions

Kunal Dilip Indore authored the "Exploratory Data Analysis" section and created the visualizations.

Larson Ost authored the "Summary", "Demonstration of the new video search", and "Discussion" sections. Also, he wrote the R Shiny application code.

Adwait Patil authored the "Predictive Model" and 'Model Performance' subsections. Also, he worked on creating the 'Time to Trend' variable and the two prediction models.

Smit Shah authored the "Processing", "Building Category Classification Model", and "Categories Prediction". Also, he wrote the Python code to generate title sentiment, tidy the text data, and build a model for category classification

# 6 References

# References


[1] "21 Essential YouTube Statistics You Need to Know in 2022." The Social Shepherd, https://thesocialshepherd.com/blog/youtube-statistics

[2] J, Mitchell. "Trending YouTube Video Statistics." Kaggle, 3 June 2019, https://www.kaggle.com/datasets/datasnaek/youtube-new?select=CAvideos.csv.

[3] "Federicopascual/Finetuning-Sentiment-Model-3000-Samples-Testcopy - Hugging Face, https://huggingface.co/federicopascual/finetuning-sentiment-model-3000-samples-testcopy.

# 7 Appendix

## 7.1 R code used to create the R Shiny Applet

```
#Author: Larson Ost
# Shiny application DS 5110 Project
# Import the libraries
library(shiny)
library(dplyr)
library(markdown)
library(tokenizers)
library(tidyverse)
library(tidytext)
library(ggplot2)

#Set irectory and get data
setwd("~/Northeastern/DS 5110/Project")
data <- read.csv("US_full.csv")

#basic reformatting, adding controversy column
my_data <- data[data$title != "", ]
my_data <-
  mutate(my_data, Controversy = dislikes / (likes + dislikes))
my_data <-
  rename(my_data, Sentiment = Title_Sentiment_values, Channel = channel_title)
my_data <- na.omit(my_data)
my_data <-
  distinct(my_data, title, .keep_all = TRUE)

#find most popular channels
most_common <- my_data %>%
  count(Channel, sort = TRUE) %>%
  top_n(20)

#define the UI
ui <- fluidPage(navbarPage(
  "Navbar!",
  tabPanel("Table",
           sidebarLayout(
             sidebarPanel(
               helpText("Youtube Sentiment Filter"),

               #search most popular channels
               selectInput(
                 "var2",
                 label = "Choose a Channel",
                 choices = c("All", unique(most_common$Channel)),
                 selected = "All"
               ),
               #Search category
               selectInput(
                 "var",
                 label = "Choose a Category",
                 choices = c("All", unique(data$category)),
```

```
              selected = "All"
            ),
            #search sentiment
            sliderInput(
              "range",
              label = "Choose a Sentiment Range",
              min = -1,
              max = 1,
              value = c(-1, 1),
              step = 0.1
            ),
            #search controversy
            sliderInput(
              "rangeC",
              label = "Choose a Threshold for Controversy",
              min = 0,
              max = 1,
              value = 1,
              step = 0.1
            ),
            #search views
            sliderInput(
              "rangeV",
              label = "Choose a Minimum Amount of Views",
              min = 0,
              max = 200000000,
              value = 0,
              step = 1000000
            )
          ),
          #display the table
          mainPanel(tableOutput("DataTable"))
        )),
  #display the word table
  tabPanel("Summary",
           mainPanel(plotOutput("WordTable")))
))

#Define the server
server <- function(input, output) {
  #Table generator
  output$DataTable <- renderTable({
    #apply filters
    dt <-
      my_data[my_data$Sentiment >= input$range[1] &
                my_data$Sentiment <= input$range[2], ]
    dt <- dt[dt$Controversy <= input$rangeC[1], ]
    dt <- dt[dt$views >= input$rangeV[1], ]
    if (input$var != "All") {
      dt <- dt[dt$category == input$var, ]
    }
    if (input$var2 != "All") {
      dt <- dt[dt$Channel == input$var2, ]
    }
```

```
  #return partial table
  dt[, c("Sentiment",
         "Controversy",
         "title",
         "Channel",
         "category",
         "views")]#,input$var)]
}, include.rownames = FALSE)

#table generator for the plot
plot2_obj <- reactive({
  #apply the filters
  dt <-
    my_data[my_data$Sentiment >= input$range[1] &
              my_data$Sentiment <= input$range[2], ]
  dt <- dt[dt$Controversy <= input$rangeC[1], ]
  dt <- dt[dt$views >= input$rangeV[1], ]
  if (input$var != "All") {
    dt <- dt[dt$category == input$var, ]
  }
  if (input$var2 != "All") {
    dt <- dt[dt$Channel == input$var2, ]
  }
  dt <-
    dt[, c("Sentiment",
           "Controversy",
           "title",
           "Channel",
           "category",
           "views")]

  #tidy format the tokens
  titles <- tibble(text = dt$title, cat = dt$category)
  titles_tidy <- unnest_tokens(titles, output = "word", input = text)

  #clean the tokenized tweets for demonstration purposes
  titles_tidy <- anti_join(titles_tidy, stop_words, by = "word")
  titles_tidy <- titles_tidy[titles_tidy$word != "39", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "amp", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "quot", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "2022", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "ft", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "2018", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "2017", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "2", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "3", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "5", ]
  titles_tidy <- titles_tidy[titles_tidy$word != "1", ]


  #return the top 20 words in the plot
  return(
    titles_tidy %>%
      count(word, sort = TRUE) %>%
```

```
        top_n(15) %>%
        ggplot(aes(x = reorder(word, n), y = n)) +
        geom_col() +
        coord_flip() +
        labs(x = "Word", y = "Count",
             title = "Most common words") +
        theme_minimal()
    )
  })
  #plot the resulting plot
  output$WordTable <- renderPlot({
    plot2_obj()
  })

}

#run the app
shinyApp(ui, server)
```

## 7.2 Python code used to generate title sentiment, tidy the text data, and build a model for category classification

Created function that return sentiment for the title

```
#Smit Shah
#Generate title sentiment
from transformers import pipeline
sentiment_model = pipeline(model="federicopascual/finetuning-sentiment-model-3000-samples")

def title_sentiment(text):
result = sentiment_model(text)
if result[0]['label'] == 'LABEL_1':
    return 1
else:
    return 0
```

Tidying the text data and generating sentiment

```
#Smit Shah
#Tidy the text data
import re
def remove_emojis(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F" # emoticons
        u"\U0001F300-\U0001F5FF" # symbols & pictographs
        u"\U0001F680-\U0001F6FF" # transport & map symbols
        u"\U0001F1E0-\U0001F1FF" # flags (iOS)
        u"\U00002500-\U00002BEF" # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f" # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)

def processing_comments(text):

    text = re.sub(r"\S*https?:\S*", "", text) ## Removes Links

    text = text.lower() ## Lowers texts

    text = text.translate(str.maketrans("","", string.punctuation)) ## Removes Punctuation
```

```
    text = remove_emojis(text) ## Removes Emojis

    ## Removes weird format apostrophe
    text = text.replace(chr(8217), "")
    text = text.replace(chr(8216), "")

    tokens = word_tokenize(text) # Tokenizing Text

    ## Since stop words removes the negative words like not, dont, etc, and having that in our
        text is very
    ## important so we take care of such words, as these words are very crucial for sentiment
        analysis
    text_tokens = []
    for token in tokens:
        if token == 'not' or token == 'dont' or token == "shouldnt" or token == "wouldnt" or token
            == "wont" or token == "arent":
            text_tokens.append(token)
        elif token not in stopword:
            text_tokens.append(token)

    ## We apply lemmatization on words and return the final text
    lemm = WordNetLemmatizer()
    final_tokens = []
    for token in text_tokens:
        if len(token) > 1:
            ftoken = lemm.lemmatize(token)
            final_tokens.append(ftoken)

    return ' '.join(final_tokens)
```

Category Classification Model building

```
#Smit Shah
#Model building with RandomForest Algorithm
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn import metrics

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Classification Report
print(classification_report(y_test, y_pred))
```

## 7.3 Visualizations

| | |
|---|---|
| Dataset size | 66.9 MB |
| Number of observations | 40949 |
| Number of variables | 19 |
| Categorical variables | 12 |
| Numerical variables | 6 |
| Missing cells | 1280 |

Table 1: Overview of Data

```
cols(
  video_id = col_character(),
  trending_date = col_character(),
  title = col_character(),
  channel_title = col_character(),
  category_id = col_double(),
  category = col_character(),
  publish_time = col_datetime(format = ""),
  tags = col_character(),
  views = col_double(),
  likes = col_double(),
  dislikes = col_double(),
  comment_count = col_double(),
  thumbnail_link = col_character(),
  comments_disabled = col_logical(),
  ratings_disabled = col_logical(),
  video_error_or_removed = col_logical(),
  description = col_character(),
  Title_Sentiment = col_character(),
  Title_Sentiment_values = col_double()
)
```

Figure 10: Column name and data type.

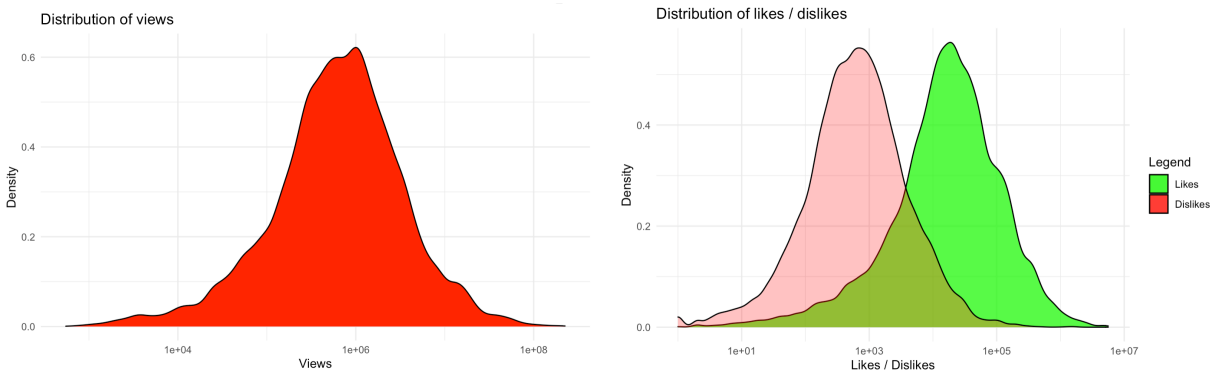


Figure 11: Distribution of views, likes, and dislikes

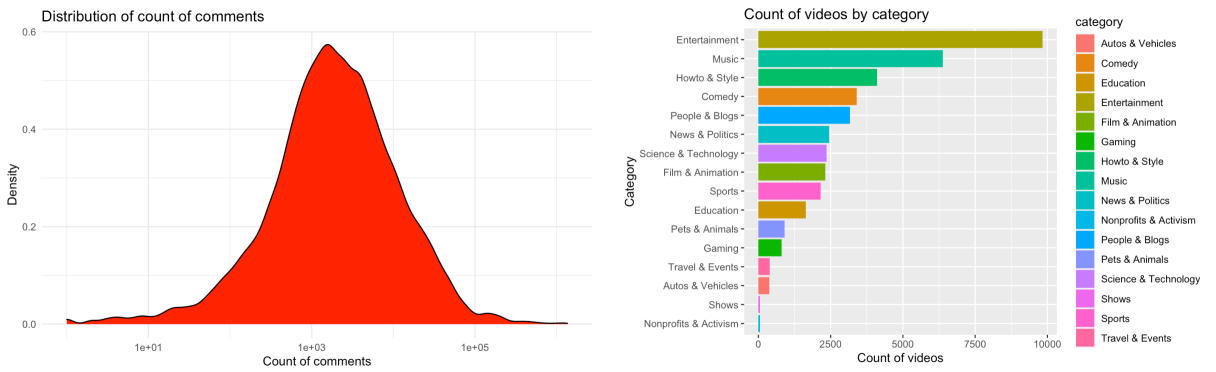


Figure 12: Distribution of comment count and Count of videos by categories


Figure 13: Word Cloud for title

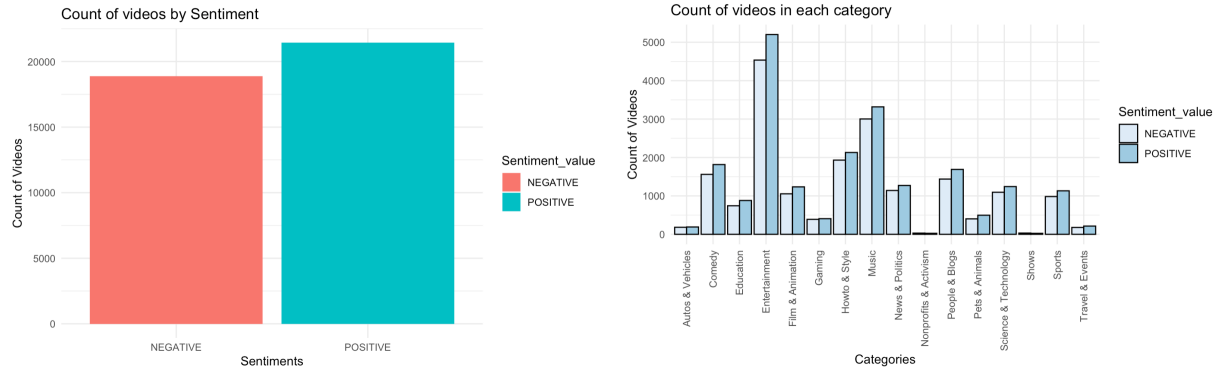


Figure 14: Count of Videos by sentiment in each category

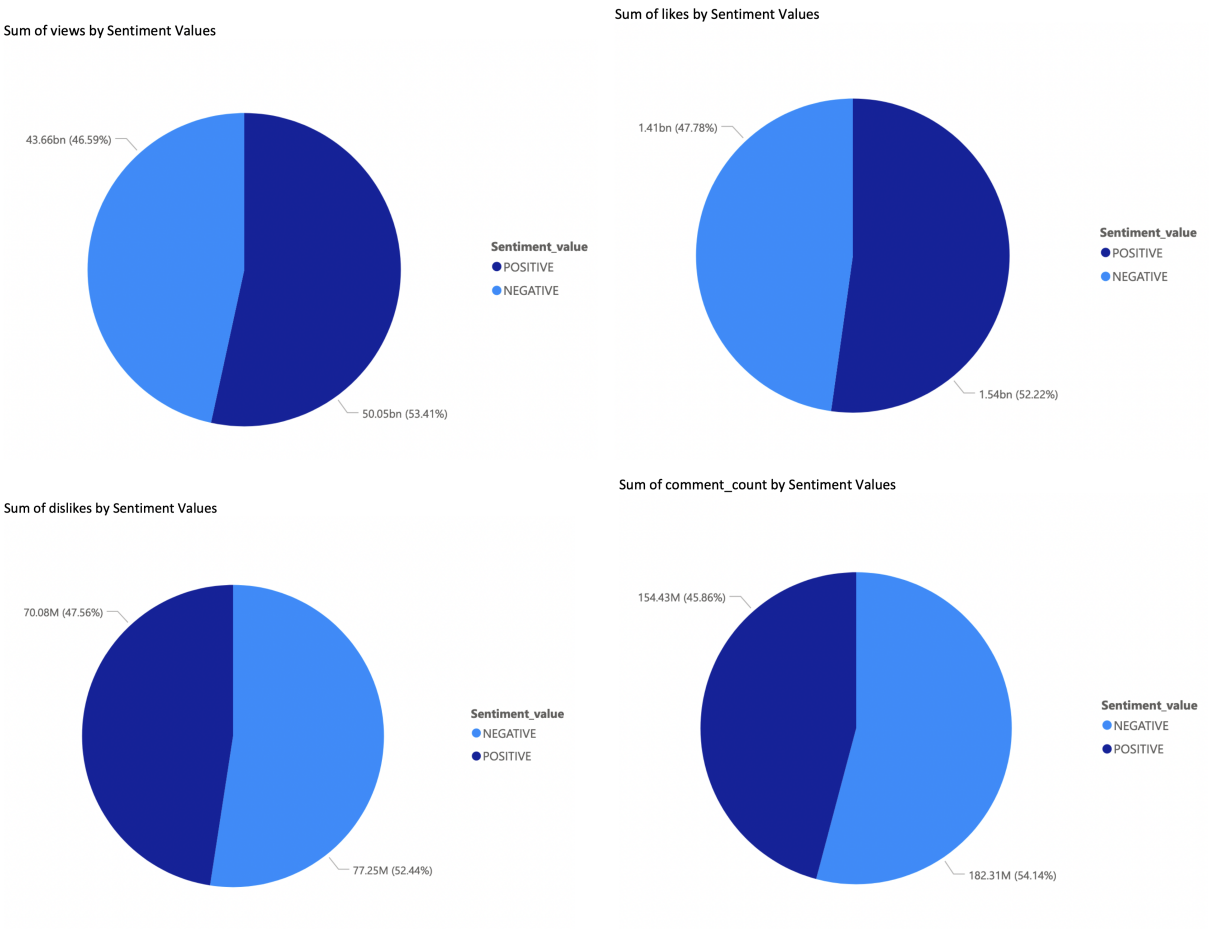


Figure 15: Percentage of views, likes, dislikes, comment count by sentiments

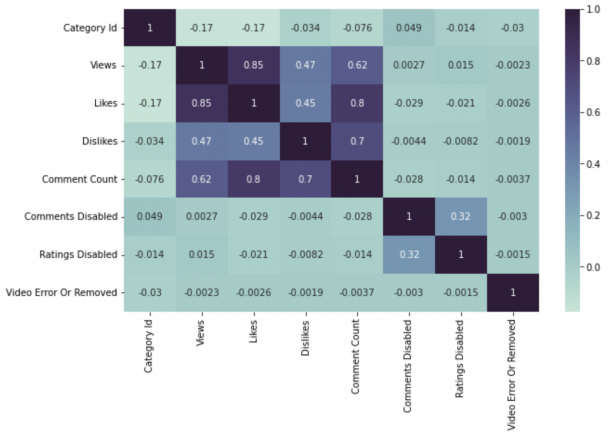


Figure 16: Correlation between numerical variables.

## 7.4 Predictive model

| | Abbrevations |
|---|---|
| views | v |
| comment_count | c |
| Title_Sentiment_values | tv |
| Time_to_trend | tt |
| dislikes | dl |
| comments_disabled__True | ct |
| ratings_disabled__True | rt |
| video_error_or_removed__True | vt |

Figure 17: Abbreviations for the features used in Figure 7