PROJECT REPORT
YOUTUBE VIDEO STATISTICS

December 13th, 2022

By: Aniruddha Sudhir Gujar
Under: Dr. Riaz Sikora

Table of Content

1. Introduction

YouTube is the world's second biggest search engine, according to the most recent traffic data, with 22.77 billion total visits in the past month (28 days). YouTube, being one of the most prominent Art & Entertainment content-providing platforms, crosses cultures and nations, with trending videos seen by a diverse spectrum of consumers worldwide. Thus, understanding the features of popular YouTube videos aids in the design and assessment of personalized services ranging from precise advertising to recommender systems.

YouTube trending videos span from anticipated blockbuster trailers to amusing talk show interviews to home recordings of a small guy yodeling at Walmart. According to YouTube, Trending strives to promote videos that are appealing to a broad audience and demonstrate what's occurring on YouTube and, to a lesser degree, what's happening throughout the world. Except for India, this list of hot videos is the same for all users in each nation. Because Trending isn't individualized and is the same for all viewers, it's an excellent list for YouTube producers to be on because it increases the odds of their video getting seen. YouTube may be an effective tool for product promotion through youtubers that have a large reach and interaction with their audience. We can come up with new techniques to increase more engagement of a new youtuber by analyzing previous historical data of the top Youtubers, applying advanced statistical algorithms, and visualizing the obtained output, and such data can be used efficiently for marketing relevant product to the relevant content viewer.

Google owns YouTube, an American online video sharing and social networking platform. According to the source, the global YouTube user base increased 9.2 percent in 2017 over the previous year. So, for the present generation, it has been a terrific platform to generate money because YouTube has a pay per view policy. YouTube may be an effective tool for product promotion through youtubers that have a large reach and interaction with their audience.

2. Data Description

This dataset was obtained from Kaggle: https://www.kaggle.com/datasnaek/youtube-new

This dataset contains statistics on daily popular YouTube videos for several months (and counting). Data is available for the US, GB, DE, CA, and FR areas (the United States, Great Britain, Germany, Canada, and France, respectively), with up to 200 trending videos displayed each day. We are analyzing data from two countries: the United States and India. The dataset includes both quantitative and qualitative information. Also, it has separate .json file which needs to be mapped with the column category_ID to get the category of the video.

DATA	DATA TYPE	DESCRIPTION
Video Id	String	It is a unique Id given to each video
Title	String	Tittle of the video
Channel title	String	Name of channel on YouTube
Category Id	Integer	It's the Unique Id given to the category on YouTube
Publish time	Date & time	It is the date and time of video published on YouTube
Tags	Strings	Hashtags given to video for easy search
Views	Integer	Number of views on videos
Likes	Integer	Number of likes on videos
Dislikes	Integer	Number of dislikes on videos
Comment count	Integer	Total number of comments on video
Thumb line link	String	URL for a video
Comment disables	Boolean	Whether Youtuber has disabled the comment section
Rating disables	Boolean	Whether Youtuber has disabled the rating for video
Video error/removed	Boolean	Any error occurred during publishing a video or video removed due to any error
Description	String	The description given by the youtuber for a video
Trending date	Date	The date when the video went on trending list

Table 1: Data and it's data type

3. Data Cleaning

The practice of correcting or deleting inaccurate, damaged, improperly formatted, duplicate, or missing data from a dataset is known as data cleaning. There are several ways for data to be duplicated or incorrectly categorized when merging different data sources.

The data has been cleaned as per the requirement of different outputs we desired. The data has been cleaned as follows for following methods:

- **For like prediction and Linear Regression:**
  For like prediction, in data first the duplicate rows are removed from the data frame, and then the category_id column has been mapped with the actual category and finally a new column has been made called country_code. We only had category Id and were unable to categorize the data in particular YouTube category. The dataset had .Json file which had category names which when mapped with dataset gave us clear idea about the video category. Also, we combine all the dataset from various countries. So, to know the location of from where the video was uploaded, we created new column as country_code.

```
In [4]: def initialize_country_dataframe(dataframe,json_fname,country_code):
            df=dataframe.copy()
            df.drop_duplicates(inplace=True)

            with open(json_fname,'r') as f:
                json_data=json.loads(f.read())

            mapping_dict=dict([(int(dictionary['id']),dictionary['snippet']['title']) for dictionary in json_data['items']]

            df['category']=df['category_id'].replace(mapping_dict)
            del df['category_id']

            df['country_code']=country_code

            return df
        dataframes=[]
        for ind,code in enumerate(country_codes):
            try:
                df=pd.read_csv(csv_files[ind])
            except:
                df=pd.read_csv(csv_files[ind],engine='python')

            df=initialize_country_dataframe(df,json_files[ind],code)
            print(code,df.shape)
            dataframes.append(df)
        dataframe=pd.concat(dataframes)
        print(dataframe.shape)


        # Remove videos that have unknown video_id
        drop_index=dataframe[dataframe.video_id.isin(['#NAME?','#VALUE!'])].index
        dataframe.drop(drop_index, axis=0, inplace=True)
```

Figure 1:Cleaning of data for Linear Regression models

- **For Clustering Similar meanings of words:**

  For this, in data we found of missing values, then replaced the empty spaces with empty strings, found out duplicate values so dropped them. Also removes all the words like https, special characters, replaced all the commas and colon with empty strings

```
In [5]: df.columns[df.isna().any()] #missing values
Out[5]: Index(['description'], dtype='object')

In [6]: df.fillna(" ",inplace=True) #replacing them with empty strings

In [7]: print("Number of duplicate rows ",list(df.duplicated()).count(True)) #duplicate
        Number of duplicate rows  4263

In [8]: df=df.drop_duplicates() #remove
        print(df.shape)
        (33089, 16)
```

```
In [11]: def text_preprocess(data):
             data=re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\), ]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',"", data)
             data=re.sub(r'\\n+'," ",data)
             data=data.replace("|"," ")
             data=re.sub(r'[?|!|\'|"|#|,|-|@|(|)]', "", data)
             data=data.replace(": "," ").replace("-"," ").replace(" :"," ")
             data=re.sub(r' +',' ',data).lower()
             return data
```

figure 2: Cleaning of data for K-mean clustering

4. Research Questions

Defining your aim entails developing a hypothesis and determining how to test it. I began by asking, "What business problem am I attempting to solve?" And here were the questions I came with which can help in further analysis.

- Besides the correlation between likes, views, dislikes, and comment counts, is there any other independent variable which can help to predict likes accurately?
- Are there any patterns of similar words in the title of video?
- Can we predict the category of the video using the title of the video?
- Does trending in greater number of countries get videos a greater number of views and like also does it help to stay in trend for greater number of days?
- What are the frequencies of the number of videos published by months from all different years?
- Do the trending videos from the data set are published in specific time slot of 24 hours day more than the other times?
- How long usually it takes for the videos to become trending?

5. Methodology

5.1. Prediction of Likes using Linear Regression

Below figure (**Figure 4**) demonstrates the flow of how the Linear Regression model was deployed for the prediction of likes for videos.

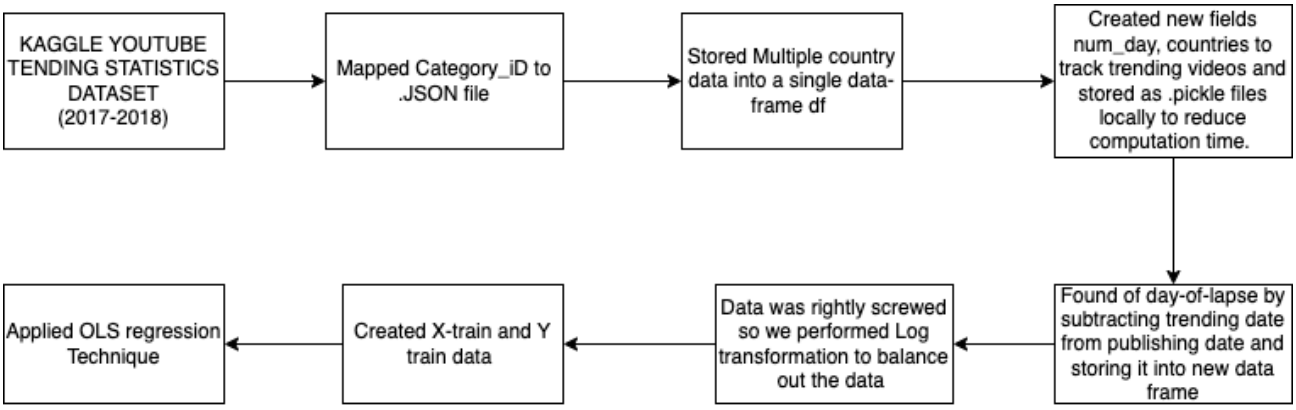

Figure 3: Process for prediction of likes using Linear regression

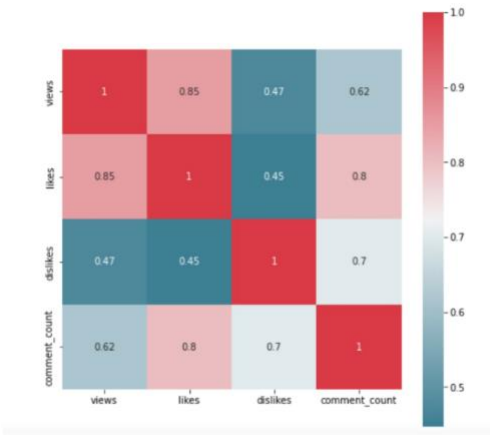

Fig 4 : correlation between quantitative variables

```
In [43]: train_rows=X_train.shape[0]
         data=pd.concat([X_train,X_test])

         data=pd.get_dummies(data)

         X_train=data[:train_rows].copy()
         X_test=data[train_rows:].copy()

         del data
         gc.collect()

         X_train.shape,X_test.shape
Out[43]: ((269055, 71), (75134, 71))

In [44]: # Baseline linear regression model
         from sklearn.linear_model import LinearRegression

         lr=LinearRegression()
         lr.fit(X_train,np.log(y_train+1))
         lr.score(X_train,np.log(y_train+1))
Out[44]: 0.8690920374345559
```

Fig 5 : Baseline linear regression model

We had 4 quantitative variables In our dataset which were views, likes , dislikes and comment counts. So we tried to find a correlation between them using heat map and there was a high correlation So we implemented linear regression for predicting likes and we got the accuracy on 86%. But we very curious about are there other factors which can affect likes and how can we use them.

So, First, we are loading Kaggle YouTube Trending Statistics dataset into a data frame (df). For like prediction, in data first the duplicate rows are removed from the data frame, and then the

category_id column has been mapped with the actual category and finally a new column has been made called country_code (**Figure 5**) to differentiate the videos based on countries which is very much useful to predict whether the country of upload influences likes received for the videos.

```
In [4]: def initialize_country_dataframe(dataframe,json_fname,country_code):
            df=dataframe.copy()
            df.drop_duplicates(inplace=True)

            with open(json_fname,'r') as f:
                json_data=json.loads(f.read())

            mapping_dict=dict([(int(dictionary['id']),dictionary['snippet']['title']) for dictionary in json_data['items']])

            df['category']=df['category_id'].replace(mapping_dict)
            del df['category_id']

            df['country_code']=country_code

            return df
        dataframes=[]
        for ind,code in enumerate(country_codes):
            try:
                df=pd.read_csv(csv_files[ind])
            except:
                df=pd.read_csv(csv_files[ind],engine='python')

            df=initialize_country_dataframe(df,json_files[ind],code)
            print(code,df.shape)
            dataframes.append(df)
        dataframe=pd.concat(dataframes)
        print(dataframe.shape)


        # Remove videos that have unknown video_id
        drop_index=dataframe[dataframe.video_id.isin(['#NAME?','#VALUE!'])].index
        dataframe.drop(drop_index, axis=0, inplace=True)
```

Figure 6: Creating new column Country_Code in the DataFrame

Lot of steps went into predicting the likes for a video, we had to check multiple sources and every source and way possible to calculate whether it influences the likes received for a video.
The most popular videos on YouTube appear at the top of the search results thanks to the YouTube algorithm. The more views a YouTuber receives, the more money they may generate from ad income. A person's video receives more hits if you like it. A playlist on your channel is created when you like a YouTube video. Additionally, it adds that video to everyone your follow's newsfeed. But I was very curious about the other factors which can affect the likes received and how we can use them.

So, we had a date when the video came to trending and timestamp it was in trending in our dataset, so we created a dictionary with number of days and number of countries in which the video was trending.

```
In [5]: # Create feature num_days that indicates the number of days the videos are in trend
        video_ids=dataframe.video_id.unique().tolist()
        num_days=[]
        id_days={}
        for vid in tqdm(video_ids):
            days=len(dataframe[dataframe.video_id==vid].trending_date.unique())
            id_days[vid]=days
            num_days.append(days)
        with open("id_days.pickle", "wb") as file:
            pickle.dump(id_days, file, pickle.HIGHEST_PROTOCOL)

        # Create feature num_countries that indicates the number of countries in the videos trended
        video_ids=dataframe.video_id.unique().tolist()
        num_countries=[]
        id_countries={}
        for vid in tqdm(video_ids):
            days=len(dataframe[dataframe.video_id==vid].country_code.unique())
            id_countries[vid]=days
            num_countries.append(days)

        with open("id_countries.pickle", "wb") as file:
            pickle.dump(id_countries, file, pickle.HIGHEST_PROTOCOL)

          0%|          | 0/178699 [00:00<?, ?it/s]

          0%|          | 0/178699 [00:00<?, ?it/s]
```

Figure 7: Creating new dictionaries ID_days & ID_countries

This procedure takes almost 3hrs to compute and I have used **tqdm** library to show the statue in form of status bar. But to save time I have stored the dictionary in the form of ".pickle" files which can be directly loaded to reduce processing time.

```
In [6]: # Reading pre-calculated dictionaries
        with open('/Users/aniruddha/Documents/data/id_days.pickle','rb') as f:
            id_days=pickle.load(f)

        with open('/Users/aniruddha/Documents/data/id_countries.pickle','rb') as f:
            id_countries=pickle.load(f)

        num_days=id_days.values()
        num_countries=id_countries.values()

        # Adding feature num_days into the dataframe
        def n_days_replace(vid):
            return id_days[vid]

        dataframe['num_days']=dataframe.video_id.apply(func=n_days_replace)

        # Adding feature num_countries into the dataframe
        def n_countries_replace(vid):
            return id_countries[vid]

        dataframe['num_countries']=dataframe.video_id.apply(func=n_countries_replace)
```

Figure 8: Creating new columns days, Countries

Now these 2 dictionaries id_days and id_countries are used as new data for analysis to check the days of trending in each country.

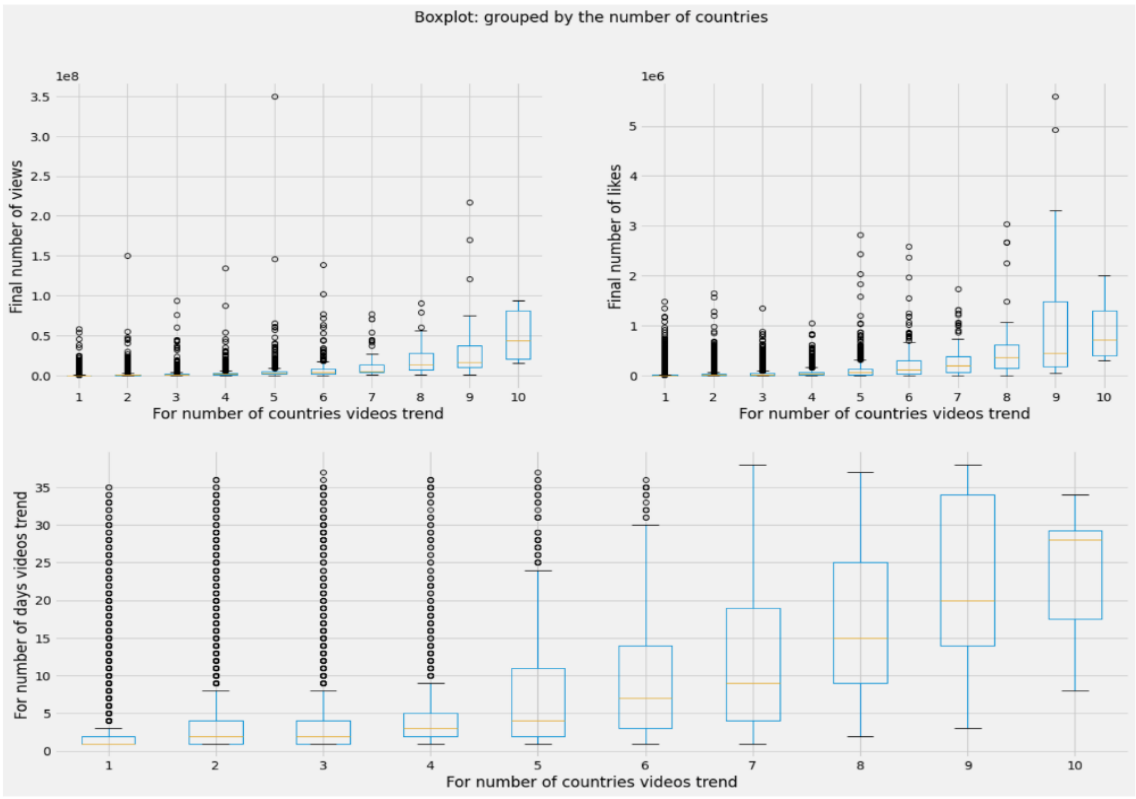

Figure 9: Does the trend in a greater number of countries get videos more number views and likes? Also does it help to stay in trend for a greater number of days?

Here the countries are represented with numbers **1–** USA, **2–** UK, **3–** China, **4–** Japan, **5–** Russia, **6–** India, **7**-Canada, **8-** Australia, **9–** Germany, **10–** Middle East

In the first two box plots, we see being in trend in a greater number of countries help the videos to gain more views and likes, which is obvious. But the third boxplot gives evidence that the videos that trend in a greater number of countries tend to be in trend for longer period of days than the videos that are in trend in lesser number of countries. The box plot shows that the median or 2nd quartile, that represents the median number of days videos are in trend, increases from left plot to right plot.

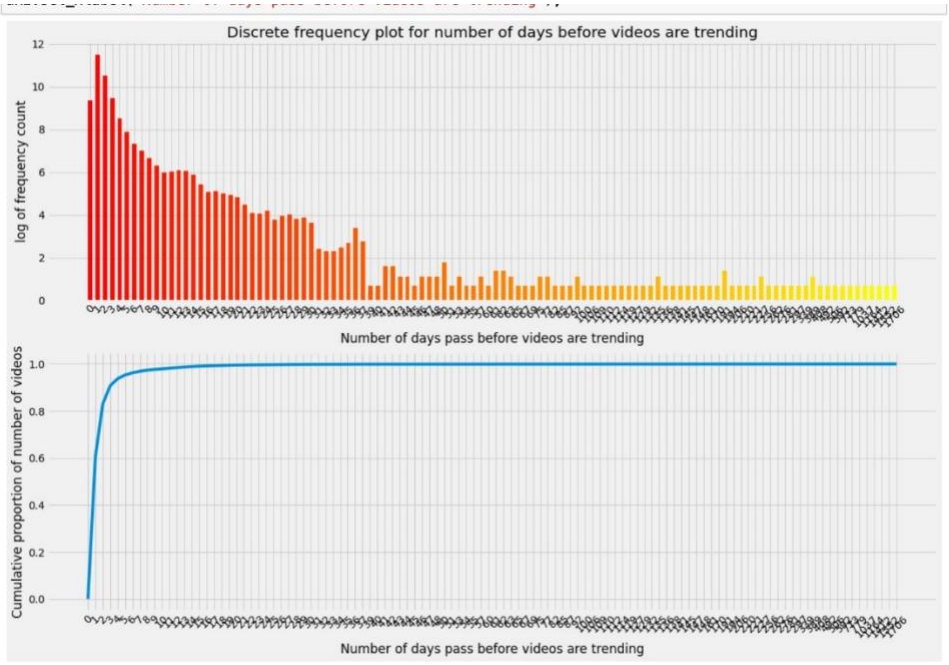

Figure 10: How long usually it takes for the videos to become trending?

We concluded that the time context is an important factor in the trending videos. In the above frequency plot, most trending videos take less than a week before coming into trend. As days pass after publishing the videos, we see less and lesser videos becoming trending.

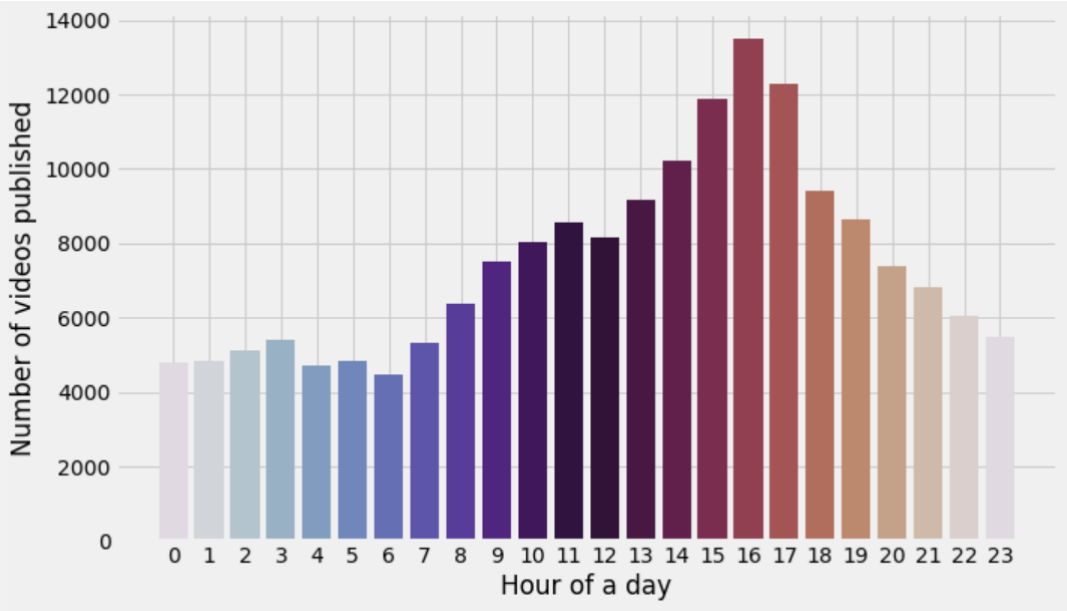

Figure 11: Videos Do the trending videos from the data sets are published in a specific time slot of 24 hrs. day more than the other times?

From the plot above we can say that more videos are published during afternoon time slot than the other times of the day.

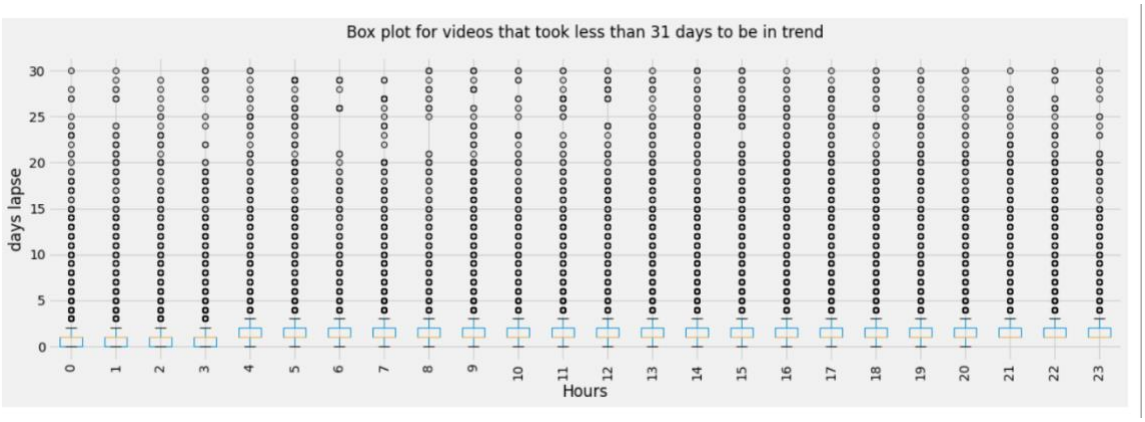

Figure 12: Videos Resolving the misconception of the hour's videos are published.

We observed in plot that after-noon hours are the most popular timeframe for publishing trending videos. But, in the box-plot diagram above, we see no significant difference in the videos becoming trending (quickly) based on the hours in which they are published.

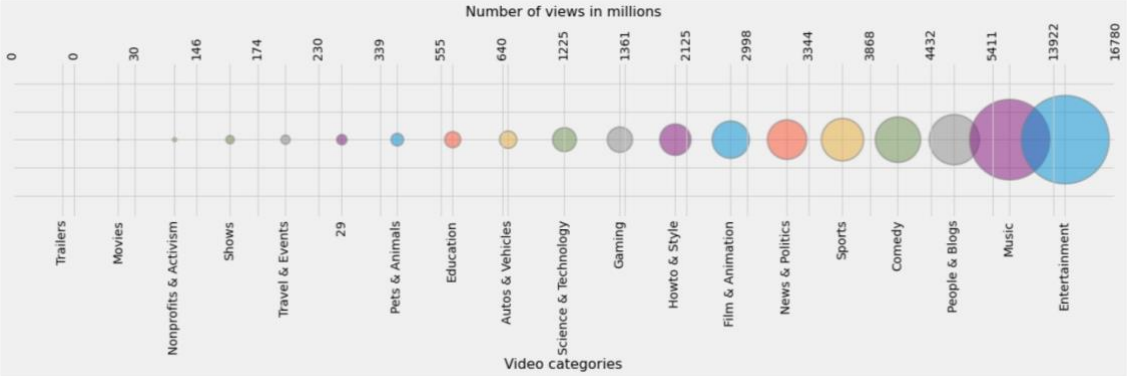

Figure 13: Videos Number of views by Video Category

Renamed the 29th category to 'Other'

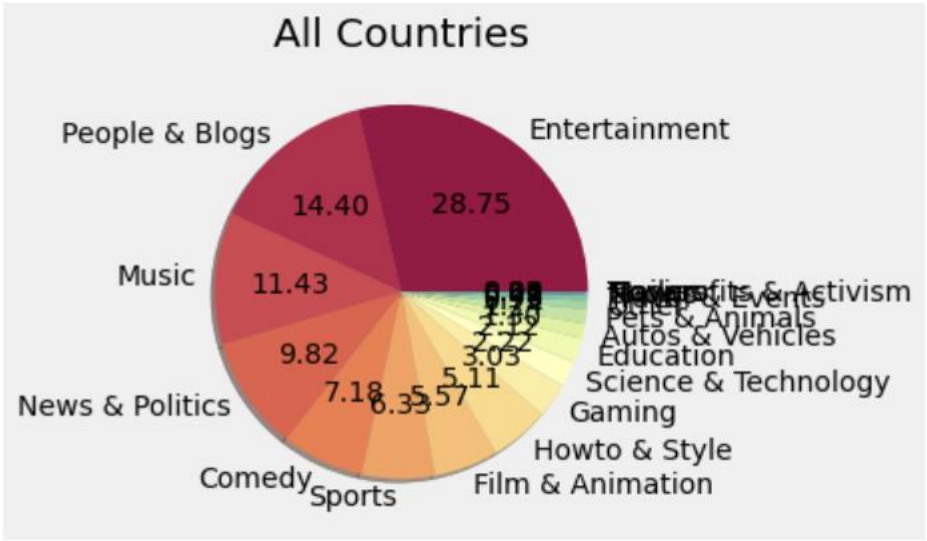

Figure 14: Proportion of videos categories trending in the countries over the entire period of given time.

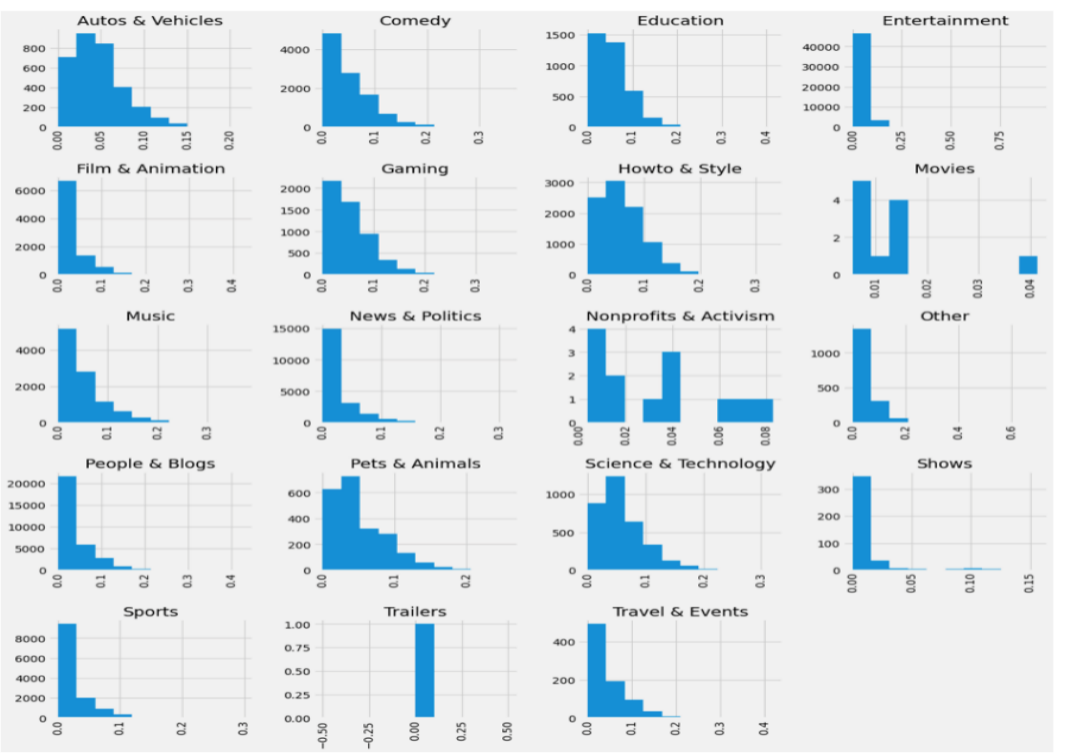

Figure 15: Distribution and boxplot for the proportion of likes per views for each category

Here the main objective was to find the odd of likes i.e., Ration of likes to view. In the histogram plots above, we can see that even though these are trending videos a very smaller number of viewers hit these videos likes. So, we calculated the odds of likes by views by applying the formula (likes/views) and displayed it in the form of histogram where you van see the there is a very less percentage of likes in comparison of the views on the videos.

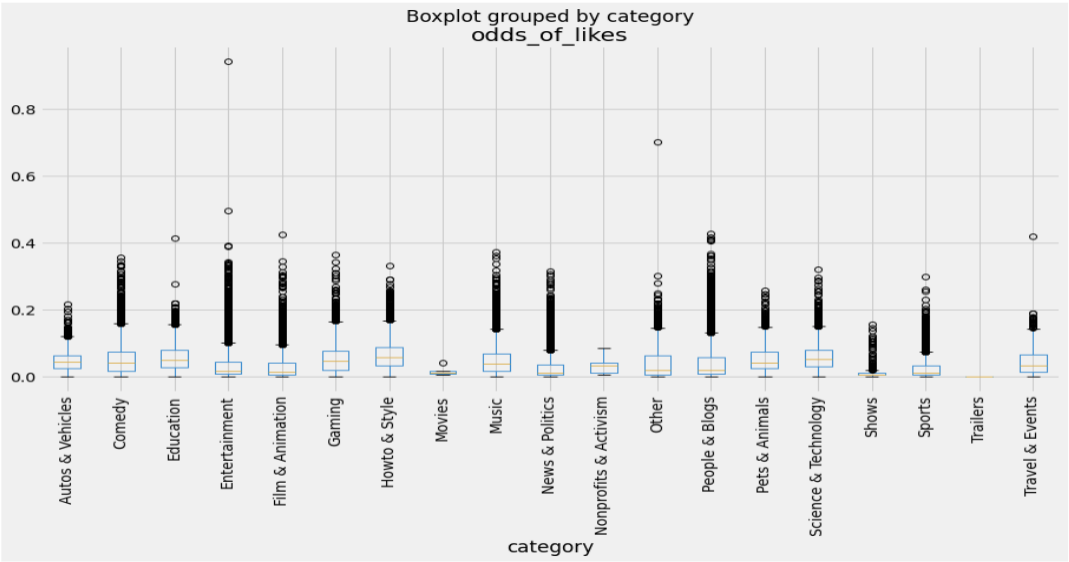

Figure 16: Videos Number of views by Video Category

In the boxplot, we see for all video categories 3rd quartile falls below 20% hit likes to viewers ratio.

- Percentage of videos by the countries that have comments section disabled

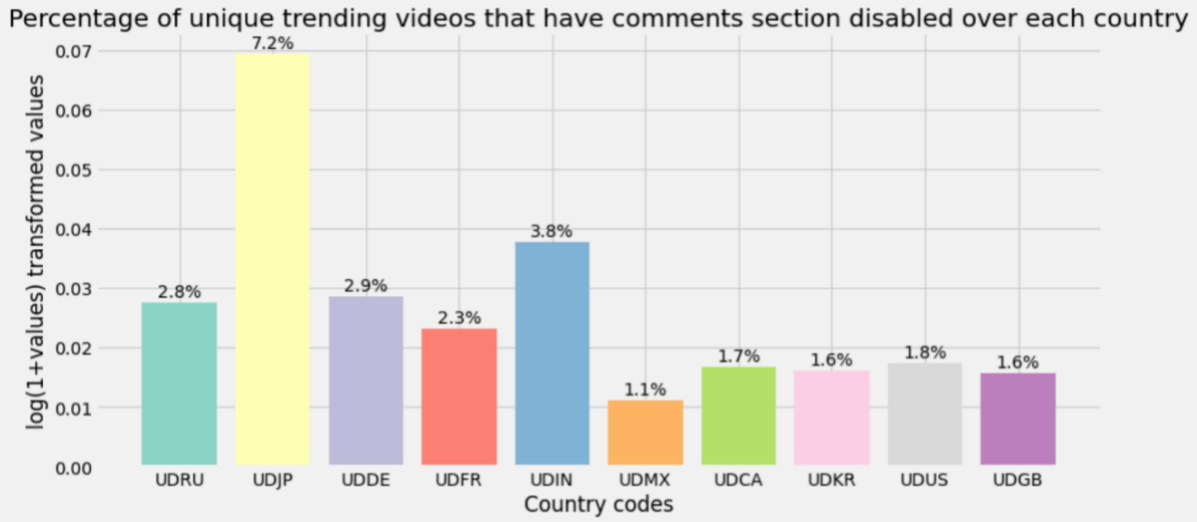

Figure 17: Percentage of videos trending with comment section disabled

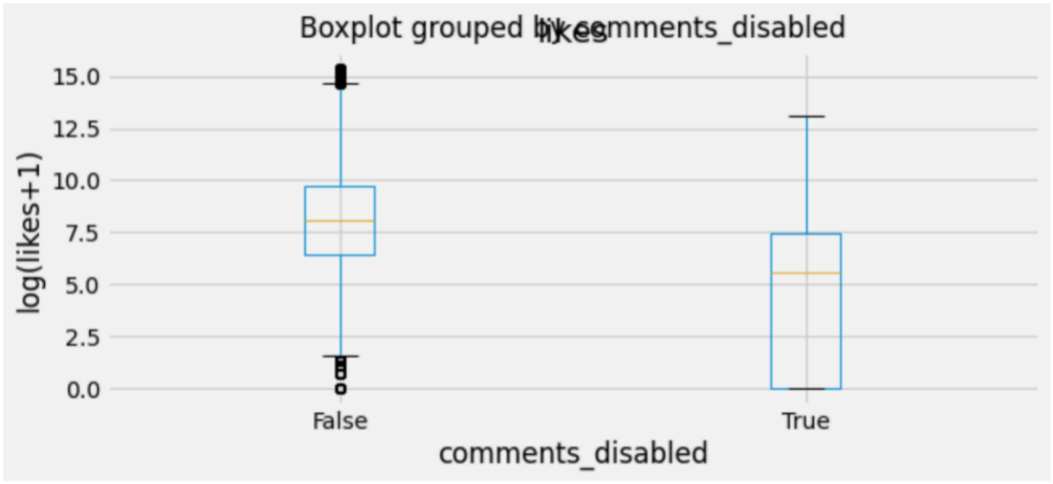

Figure 18: BoxPlot Comments Disabled

It shows that comment disable doesn't affect the probability of video coming into tending page.

When our original continuous data do not follow the bell curve, we can log transform this data to make it as "normal" as possible so that the statistical analysis results from this data become more valid as the graph shows the data is rightly screwed.

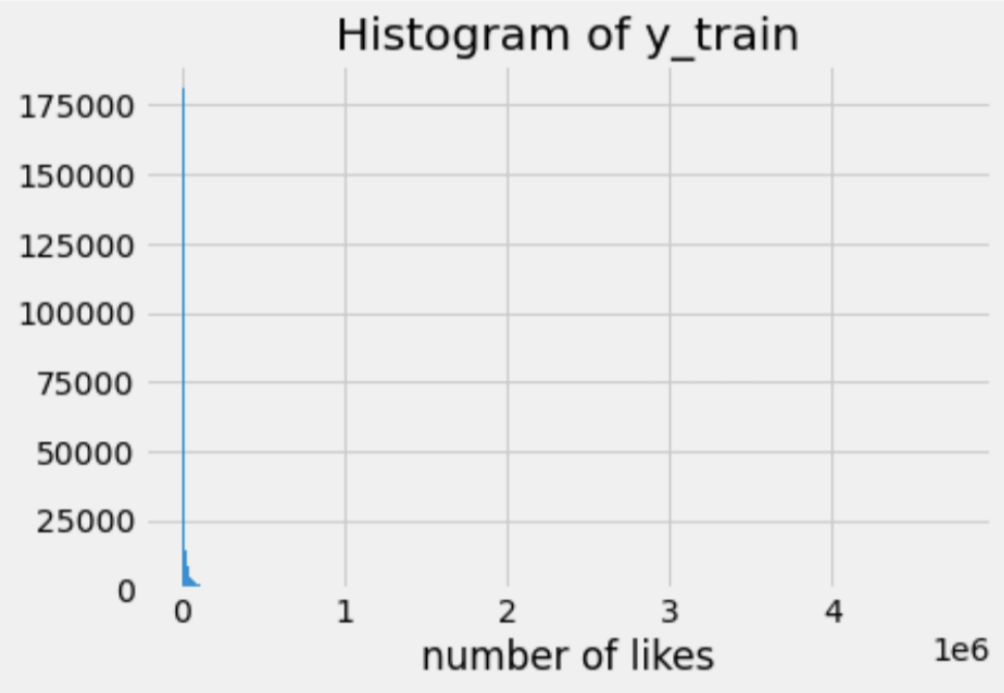

Figure 19: Log Transforming

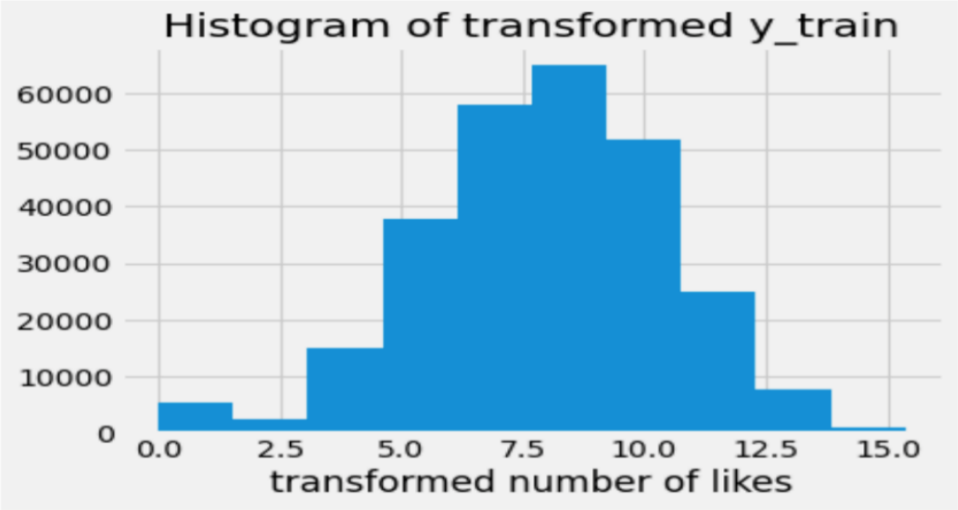

Figure 20: Log Transforming for balancing the data

5.2 Using Decision tree to classify the video into categories using the video titles:

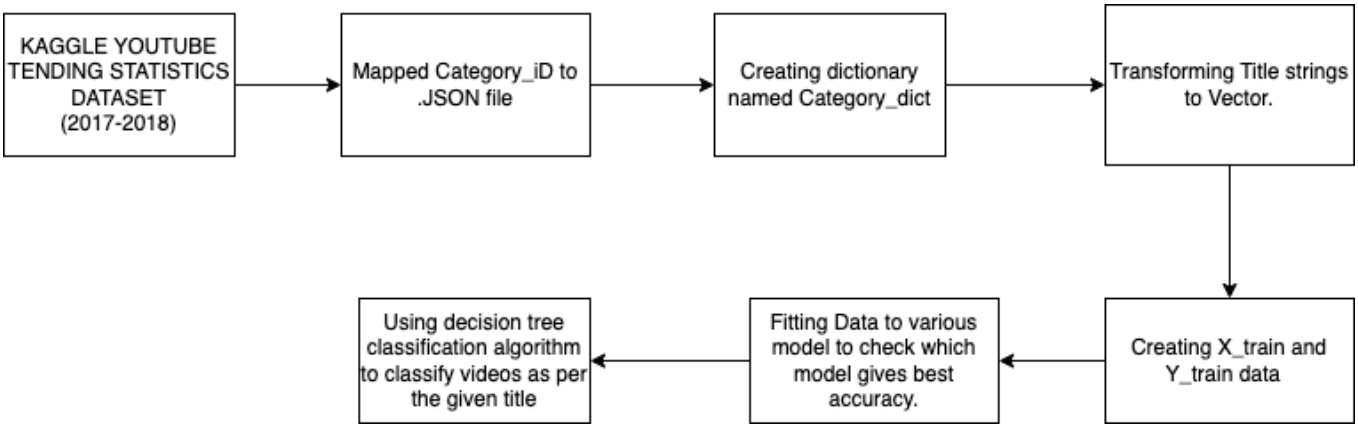

Figure 21: Implementation of Decision tree for classification

Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. It is a supervise learning method. Here I have used 5 techniques K neighbors Classifier, Decision Tree Classifier, Random Forest Classifier, SVC, Naive Bayes and have then compared their accuracy and then used it for finding the category using the video title. The techniques used here are K neighbors Classifier, Decision Tree Classifier, Random Forest Classifier, SVC, Naive Bayes. The algorithm which gives good accuracy has been used to predict the category of video using the hypothetical video title.

```
category_dict = [{'id': item['id'], 'title': item['snippet']['title']} for item in category_json['items']]
category_dict
```

```
[{'id': '1', 'title': 'Film & Animation'},
 {'id': '2', 'title': 'Autos & Vehicles'},
 {'id': '10', 'title': 'Music'},
 {'id': '15', 'title': 'Pets & Animals'},
 {'id': '17', 'title': 'Sports'},
 {'id': '18', 'title': 'Short Movies'},
 {'id': '19', 'title': 'Travel & Events'},
 {'id': '20', 'title': 'Gaming'},
 {'id': '21', 'title': 'Videoblogging'},
 {'id': '22', 'title': 'People & Blogs'},
 {'id': '23', 'title': 'Comedy'},
 {'id': '24', 'title': 'Entertainment'},
 {'id': '25', 'title': 'News & Politics'},
 {'id': '26', 'title': 'Howto & Style'},
 {'id': '27', 'title': 'Education'},
 {'id': '28', 'title': 'Science & Technology'},
 {'id': '29', 'title': 'Nonprofits & Activism'},
 {'id': '30', 'title': 'Movies'},
 {'id': '31', 'title': 'Anime/Animation'},
 {'id': '32', 'title': 'Action/Adventure'},
 {'id': '33', 'title': 'Classics'},
 {'id': '34', 'title': 'Comedy'},
 {'id': '35', 'title': 'Documentary'},
 {'id': '36', 'title': 'Drama'},
 {'id': '37', 'title': 'Family'},
 {'id': '38', 'title': 'Foreign'},
 {'id': '39', 'title': 'Horror'},
 {'id': '40', 'title': 'Sci-Fi/Fantasy'},
 {'id': '41', 'title': 'Thriller'},
 {'id': '42', 'title': 'Shorts'},
 {'id': '43', 'title': 'Shows'},
 {'id': '44', 'title': 'Trailers'}]
```

Screenshot

Figure 22: Mapping the category ID with category JSON file

We have mapped a JSON file with the category id, the JSON file contained the category names.

```
In [16]: X = counts
         Y = output
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = .1)

In [17]: NBtest = MultinomialNB().fit(X_train,Y_train)
         nb_predictions = NBtest.predict(X_test)
         acc_nb = NBtest.score(X_test, Y_test)
         print('The Naive Bayes Algorithm has an accuracy of', acc_nb)

         The Naive Bayes Algorithm has an accuracy of 0.9037851037851038

In [18]: RFCtest = RandomForestClassifier().fit(X_train,Y_train)
         rfc_predictions = RFCtest.predict(X_test)
         acc_rfc = RFCtest.score(X_test, Y_test)
         print('The Random Forest Algorithm has an accuracy of', acc_rfc)

         The Random Forest Algorithm has an accuracy of 0.9904761904761905

In [19]: SVCtest = SVC().fit(X_train,Y_train)
         svc_predictions = SVCtest.predict(X_test)
         acc_svc = SVCtest.score(X_test, Y_test)
         print('The Support Vector Algorithm has an accuracy of', acc_svc)

         The Support Vector Algorithm has an accuracy of 0.9885225885225886

In [20]: KNCtest = KNeighborsClassifier().fit(X_train,Y_train)
         knc_predictions = KNCtest.predict(X_test)
         acc_knc = KNCtest.score(X_test, Y_test)
         print('The K Neighbors Algorithm has an accuracy of', acc_knc)

         The K Neighbors Algorithm has an accuracy of 0.9496947496947497

In [21]: DTCtest = DecisionTreeClassifier().fit(X_train,Y_train)
         dtc_predictions = DTCtest.predict(X_test)
         acc_dtc = DTCtest.score(X_test, Y_test)
         print('The Decision Tree Algorithm has an accuracy of', acc_dtc)

         The Decision Tree Algorithm has an accuracy of 0.9868131868131869
```

Figure 23: Fitting the data in various algorithms to find which algorithm gives best accuracy

Here we are getting a good **accuracy** for both random forest and Decision tree which is **99.84%** and **98.68%** respectively. But we are going ahead with Decision Tree as Decision Trees are more intuitive than Random Forests and thus are easier to explain to a nontechnical person.

In [46]:
```
Titles_counts = vector.transform(Titles)
PredictDTC = DTC_Model.predict(Titles_counts)

CategoryNamesListDTC = []
for Category_ID in PredictDTC:
    MatchingCategoriesDTC = [x for x in category_dict if x["id"] == str(Category_ID)]
    if MatchingCategoriesDTC:
        CategoryNamesListDTC.append(MatchingCategoriesDTC[0]["title"])

TitleDataFrameDTC = []
for i in range(0, len(Titles)):
    TitleToCategoriesDTC = {'Title': Titles[i],  'Category': CategoryNamesListDTC[i]}
    TitleDataFrameDTC.append(TitleToCategoriesDTC)

PredictDFdtc = pd.DataFrame(PredictDTC)
TitleDFdtc = pd.DataFrame(TitleDataFrameDTC)
PreFinalDFdtc = pd.concat([PredictDFdtc, TitleDFdtc], axis=1)
PreFinalDFdtc.columns = (['Categ_ID', 'Predicted Category', 'Hypothetical Video Title'])
FinalDFdtc = PreFinalDFdtc.drop(['Categ_ID'],axis=1)
colsDTC = FinalDFdtc.columns.tolist()
colsDTC = colsDTC[-1:] + colsDTC[:-1]
FinalDFdtc= FinalDFdtc[colsDTC]
FinalDFdtc
```

Out[46]:

	Hypothetical Video Title	Predicted Category
0	Pets & Animals	Hilarious cat plays with toy
1	Entertainment	Best fashion looks for Spring 2018
2	Sports	Olympics opening ceremony highlights
3	Travel & Events	Warriors basketball game versus the cavs
4	Comedy	CNN world news on donald trump
5	Entertainment	Police Chase in Hollywood
6	Music	Ed Sheeran - Perfect (Official Music Video)
7	Entertainment	how to do eyeshadow

Figure 24: Final prediction of category using title

Here I converted the titles into vectors and using decision tree tried to predict the category. But the algorithm is not 100% accurate there is an **error rate of 0.2%** as we can see that "CNN WORLD NEWS ON DONALD TRUMP "has been categories into comedy but others are perfectly categories. Sometimes the video on YouTube is classified into different categories than the content of the title. Classifying videos using the video title can be more significant.

5.3 Clustering of similar Works Using K-Means Algorithm:

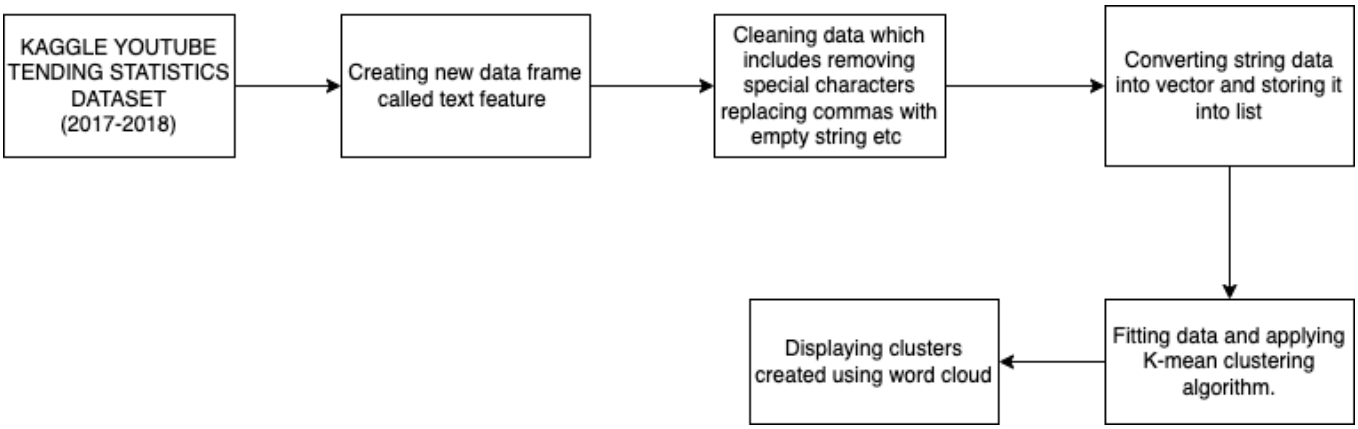

Figure 25: Linear Regression Model using entities based on exploration analysis.

Clustering is a Machine Learning technique that involves the grouping of data points. For Indian dataset it was hard to categories data as it there are many languages and religious videos on Indian YouTube. We decided to use unsupervised learning K-mean clustering using variables video title, channel, title, descriptive, tags. Also, we have used K as 20 as we plotted loss and number of cluster where min value of loss landed between 15 to 20 as shown in the below

In [192]:
```
def k_means(data,cluster_range):
    models=[]
    loss=[]
    for k in cluster_range:
        kmeans=KMeans(n_clusters=k,init='k-means++',n_jobs=-1).fit(data)
        models.append(kmeans)
        loss.append(kmeans.inertia_)
    plt.plot(cluster_range,loss)
    plt.xlabel('k-clusters')
    plt.ylabel('loss')
    plt.show()
    return models
```

In [197]:
```
def cluster_analysis(train_data,k):
  #For each cluster
  for i in range(0,k):
    #Extract cleaned text column
    data=train_data[train_data['labels']==i]
    list_of_words=[];
#     print("data: ",data)
    for sent in data['title']:
        #print("Title: ",sent)
        for word in sent.split():
            list_of_words.append(word)
    final_text=" ".join(list_of_words)
    #print("Cluster : ",i+1)
    #print("Number of reviews",len(data))
    #print("      Word Cloud ")
    wordcloud = WordCloud(collocations=True).generate(final_text)
    plt.figure()
    title="\nCluster : "+str(i+1)+"\n Number of Reviews: "+str(len(data))
    plt.title(title)
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

Figure 26: K-mean algorithm for clustering similar word meaning

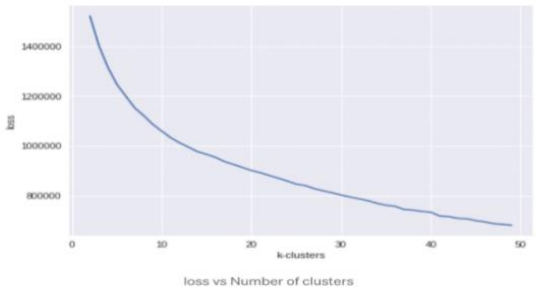

Figure 27: Elbow method to plot loss vs Number of cluster

In this code I have first cleaned and converted the word titles into vectors and then tried to cluster similar meaning words together and display them into word cloud. By using k-mean clustering we found similar word patterns in the titles of trending videos which can be further used to get the new videos on the search results.

Figure 28: Final clusters represented using wordcloud

5.4 Sentiment Analysis Using NLP:

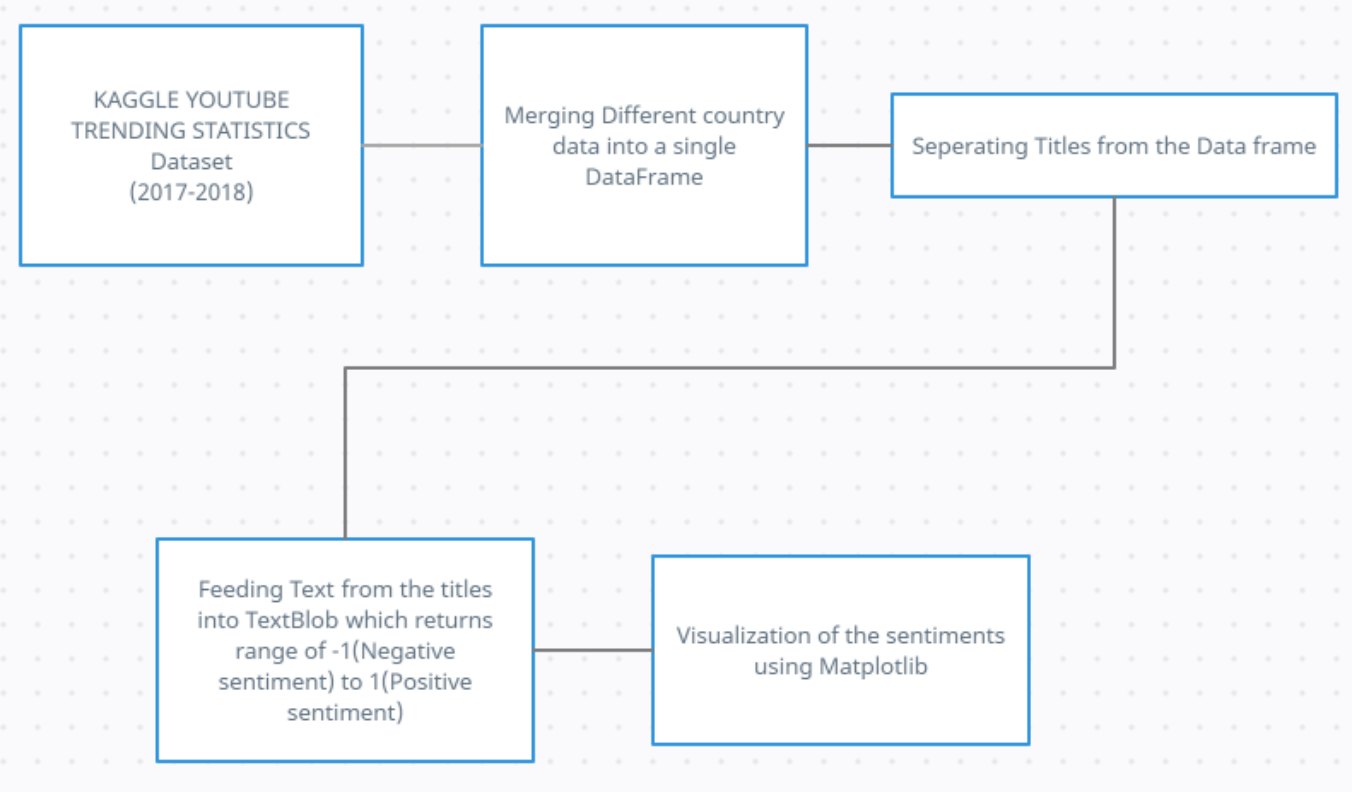

Figure 29: Natural Language Processing

Sentiment Analysis can help us decipher the mood and emotions of the public and gather insightful information regarding the context. Sentiment Analysis is a process of analyzing data and classifying it based on the need of the research. Textblob is a python library for Natural Language Processing (NLP). Textblob actively used Natural Language Toolkit (NLTK) to achieve its tasks. NLTK is a library which gives easy access to a lot of lexical resources and allows users to work with categorization, classification, and many other tasks. Textblob is a simple library which supports complex analysis and operations on textual data.

Taking Data from all the country data into separate Data Frame and combining them into a single source so that the data can be used to find the sentiment

```
df_usa=pd.read_csv("./USvideos.csv")
df_ca=pd.read_csv("./CAvideos.csv")
df_de=pd.read_csv("./DEvideos.csv")
df_fr=pd.read_csv("./FRvideos.csv")
df_gb=pd.read_csv("./GBvideos.csv")
```

Figure 30: Different countries data being loaded into DataFrames

```
bloblist_title = list()

df_usa_title_str=df_usa['title']
for row in df_usa_title_str:
    blob = TextBlob(row)
    bloblist_title.append((row,blob.sentiment.polarity, blob.sentiment.subjectivity))
    df_usa_polarity_title = pd.DataFrame(bloblist_title, columns = ['sentence','sentiment','polarity'])

def f_title(df_usa_polarity_title):
    if df_usa_polarity_title['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_title['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_title['Sentiment_Type'] = df_usa_polarity_title.apply(f_title, axis=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_title)
```

Figure 31: Sentimental Analysis on titles

In the above figure we can see that the sentiment is being predicted based on title with the titles being passed as input into the textblob library which return value as -1,0,1 which is

-1 - Negative Sentiment
0 - Neutral Sentiment
1 – Positive Sentiment

```
bloblist_tags = list()

df_usa_tags_str=df_usa['tags']
for row in df_usa_tags_str:
    blob = TextBlob(row)
    bloblist_tags.append((row,blob.sentiment.polarity, blob.sentiment.subjectivity))
    df_usa_polarity_tags = pd.DataFrame(bloblist_tags, columns = ['sentence','sentiment','polarity'])

def f_tags(df_usa_polarity_tags):
    if df_usa_polarity_tags['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_tags['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_tags['Sentiment_Type'] = df_usa_polarity_tags.apply(f_tags, axis=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_tags)
```

Figure 32: sentimental analysis on tag column

In the above figure we can see that the sentiment is being predicted based on Tags with the tags being passed as input into the textblob library which return value as -1,0,1 which is

-1 - Negative Sentiment
0 - Neutral Sentiment
1 – Positive Sentiment

```
!pip install -U textblob
from textblob import TextBlob

bloblist_desc = list()

df_usa_descr_str=df_usa['description'].astype(str)
for row in df_usa_descr_str:
    blob = TextBlob(row)
    bloblist_desc.append((row,blob.sentiment.polarity, blob.sentiment.subjectivity))
    df_usa_polarity_desc = pd.DataFrame(bloblist_desc, columns = ['sentence','sentiment','polarity'])

def f(df_usa_polarity_desc):
    if df_usa_polarity_desc['sentiment'] > 0:
        val = "Positive"
    elif df_usa_polarity_desc['sentiment'] == 0:
        val = "Neutral"
    else:
        val = "Negative"
    return val

df_usa_polarity_desc['Sentiment_Type'] = df_usa_polarity_desc.apply(f, axis=1)

plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment_Type", data=df_usa_polarity_desc)
```

Figure 33: Sentimental Analysis on description

In the above figure we can see that the sentiment is being predicted based on description with the description being passed as input into the textblob library which return value as -1,0,1 which is

-1 - Negative Sentiment

0 - Neutral Sentiment

1 – Positive Sentiment

6. Results and Discussion

6.1 Prediction of Likes Using Linear Regression:

```
In [33]: train.num_countries.replace(dict(zip(list(range(1,11)),'a b c d e f g h i j'.split())),
         train.comments_disabled.replace({False:'false',True:'true'},inplace=True)
         train.ratings_disabled.replace({False:'false',True:'true'},inplace=True)
         train.video_error_or_removed.replace({False:'false',True:'true'},inplace=True)

         test.num_countries.replace(dict(zip(list(range(1,11)),'a b c d e f g h i j'.split())),i
         test.comments_disabled.replace({False:'false',True:'true'},inplace=True)
         test.ratings_disabled.replace({False:'false',True:'true'},inplace=True)
         test.video_error_or_removed.replace({False:'false',True:'true'},inplace=True)

In [36]: # Create train and test data frames for prediction
         X_train=train[['comments_disabled','ratings_disabled','video_error_or_removed','categor
         X_test=test[['comments_disabled','ratings_disabled','video_error_or_removed','category'

In [39]: train_rows=X_train.shape[0]
         data=pd.concat([X_train,X_test])

         data=pd.get_dummies(data)

         X_train=data[:train_rows].copy()
         X_test=data[train_rows:].copy()

         del data
         gc.collect()

         X_train.shape,X_test.shape
         x = sm.add_constant(x)

In [42]: import statsmodels.api as sm
         result = sm.OLS(y_train+1, X_train+1).fit()
         print(result.summary())
```

Figure 34: Linear Regression Model using entities based on exploration analysis.

```
                                 OLS Regression Results
==============================================================================
=
Dep. Variable:                      y   R-squared (uncentered):                   0.92
7
Model:                            OLS   Adj. R-squared (uncentered):              0.92
7
Method:                 Least Squares   F-statistic:                          4.726e+0
5
Date:                Tue, 29 Nov 2022   Prob (F-statistic):                        0.0
0
Time:                        15:33:32   Log-Likelihood:                         -35165
.
No. Observations:               39781   AIC:                                  7.038e+0
4
Df Residuals:                   39757   BIC:                                  7.058e+0
4
Df Model:                          24
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.5490      0.004    133.339      0.000       0.541       0.557
x2            -0.1485      0.004    -38.249      0.000      -0.156      -0.141
x3             0.6401      0.004    173.808      0.000       0.633       0.647
const       2.426e-13   2.78e-13      0.872      0.383   -3.03e-13    7.88e-13
x4          9.011e-14   1.02e-13      0.884      0.377    -1.1e-13     2.9e-13
x5          -1.21e-13   1.38e-13     -0.876      0.381   -3.92e-13     1.5e-13
x6             0.0020      0.001      3.717      0.000       0.001       0.003
x7             0.0668      0.012      5.717      0.000       0.044       0.090
==============================================================================
Omnibus:                     5151.631   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            34821.082
Skew:                          -0.432   Prob(JB):                         0.00
Kurtosis:                       7.501   Cond. No.                     1.01e+16
==============================================================================
```

Figure 35: OLS regression results

In Statistics, Mean Squared Error (MSE) is defined as Mean or Average of the square of the difference between actual and estimated values. This is used as an evaluation measure for regression models and the lower value indicates a better fit. The reason for constant is it forces the residuals to have that crucial zero mean. R-squared is the measurement of how much of the independent variable is explained by changes in our dependent variables. With this model we achieved the accuracy of **76.40%**.

6.2 Using Decision tree to classify the video into categories using the video titles

We went with decision trees as it yielded the best accuracy out of all the other algorithms

In [46]:
```
Titles_counts = vector.transform(Titles)
PredictDTC = DTC_Model.predict(Titles_counts)

CategoryNamesListDTC = []
for Category_ID in PredictDTC:
    MatchingCategoriesDTC = [x for x in category_dict if x["id"] == str(Category_ID)]
    if MatchingCategoriesDTC:
        CategoryNamesListDTC.append(MatchingCategoriesDTC[0]["title"])

TitleDataFrameDTC = []
for i in range(0, len(Titles)):
    TitleToCategoriesDTC = {'Title': Titles[i],  'Category': CategoryNamesListDTC[i]}
    TitleDataFrameDTC.append(TitleToCategoriesDTC)

PredictDFdtc = pd.DataFrame(PredictDTC)
TitleDFdtc = pd.DataFrame(TitleDataFrameDTC)
PreFinalDFdtc = pd.concat([PredictDFdtc, TitleDFdtc], axis=1)
PreFinalDFdtc.columns = (['Categ_ID', 'Predicted Category', 'Hypothetical Video Title'])
FinalDFdtc = PreFinalDFdtc.drop(['Categ_ID'],axis=1)
colsDTC = FinalDFdtc.columns.tolist()
colsDTC = colsDTC[-1:] + colsDTC[:-1]
FinalDFdtc= FinalDFdtc[colsDTC]
FinalDFdtc
```

Out[46]:

	Hypothetical Video Title	Predicted Category
0	Pets & Animals	Hilarious cat plays with toy
1	Entertainment	Best fashion looks for Spring 2018
2	Sports	Olympics opening ceremony highlights
3	Travel & Events	Warriors basketball game versus the cavs
4	Comedy	CNN world news on donald trump
5	Entertainment	Police Chase in Hollywood
6	Music	Ed Sheeran - Perfect (Official Music Video)
7	Entertainment	how to do eyeshadow

Figure 36: Classification of videos using titles computed using decision tree

Here I converted the titles into vectors and using decision tree tried to predict the category. But the algorithm is not 100% accurate there is an **error rate of 0.2%** as we can see that "CNN WORLD NEWS ON DONALD TRUMP "has been categories into comedy but others are perfectly categories. Sometimes the video on YouTube is classified into different categories than the content of the title. Classifying videos using the video title can be more significant.

6.3 Clustering Similar Words Using K Means Algorithm:

Figure 37: Results of K-mean clustering using wordcloud

Using the unsupervised learning method, we attempted to combine words in the tiles that have similar meanings. Unsupervised learning is the process of training a Machine Learning model with unsupervised learning and allowing the model to act on that data without supervision. So we converted the titles into vector format first, then fitted the data and generated a word cloud with 20 clusters. As we can see in the title, words with similar meanings or words related to each other have been grouped together. This words indicate that they have good search results, so if a new youtuber wants to get more views or appear in search results, he or she can incorporate this language into their titles.

6.4 Sentiment Analysis Using NLP:

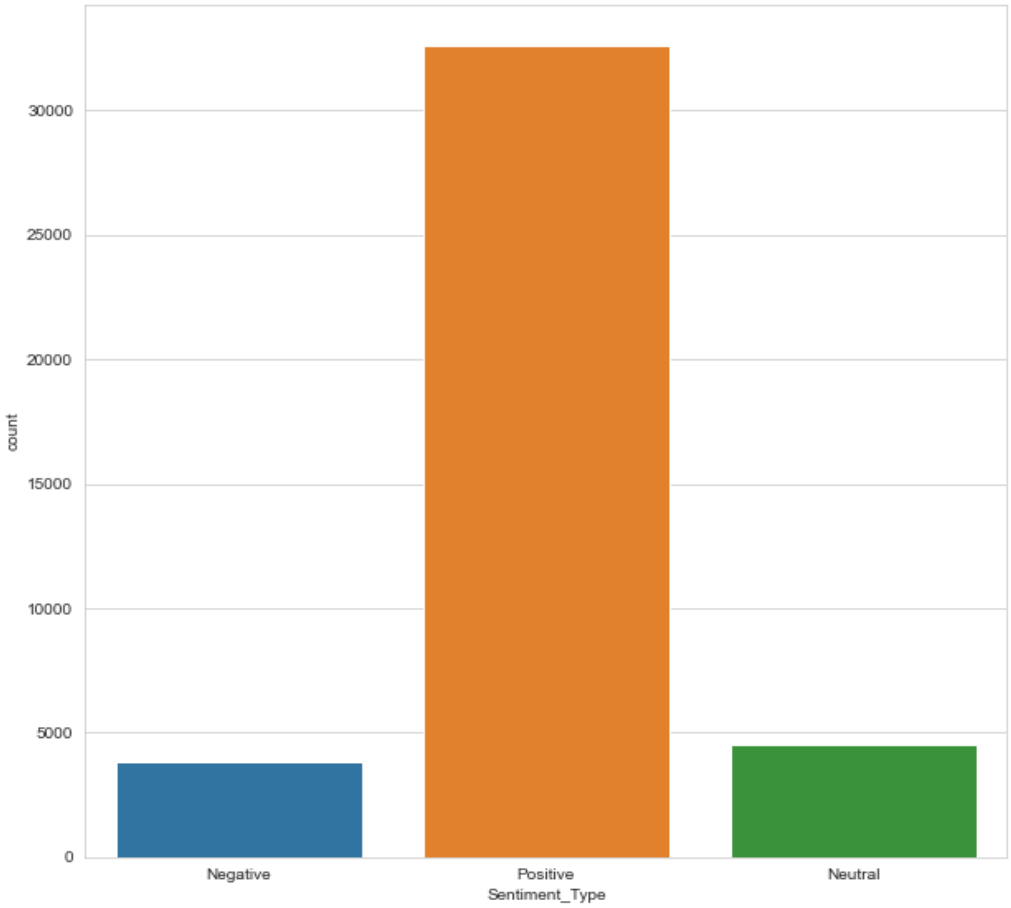

Figure 38: Sentiment Analysis results on description

Sentiment Analysis based on description, proves that most of the description on videos had more positive sentiments and equal Negative and Neutral sentiments

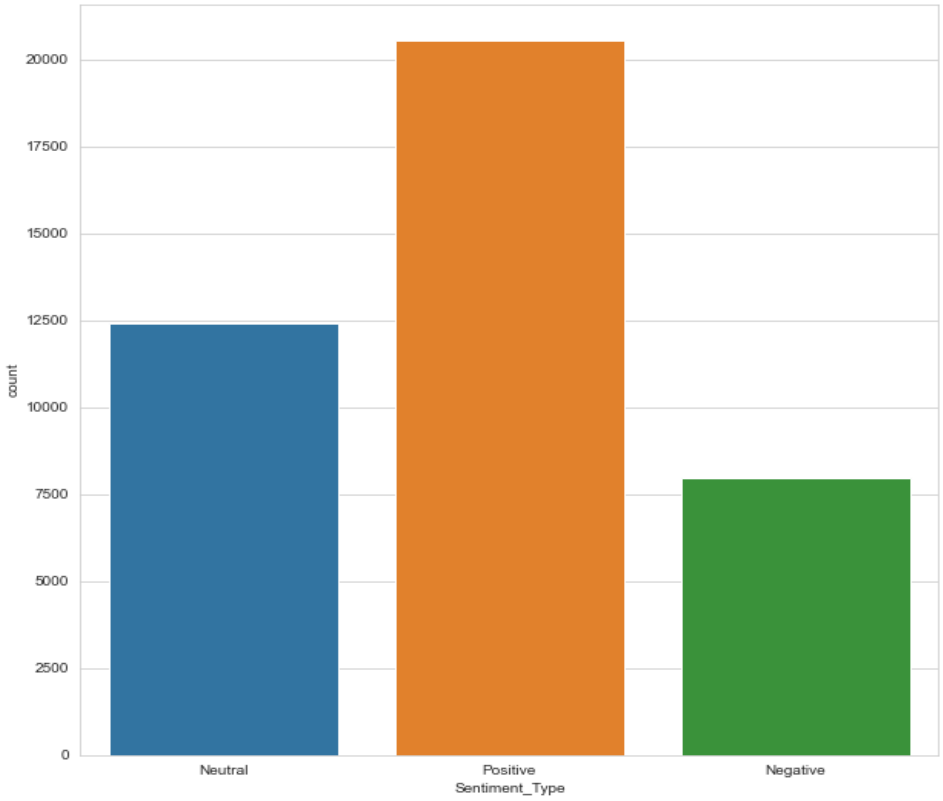

Figure 39: sentimental analysis results on tags

Sentiment Analysis based on tags, proves that most of the tags on videos had mostly positive sentiments with Neutral sentiments and less Negative sentiments

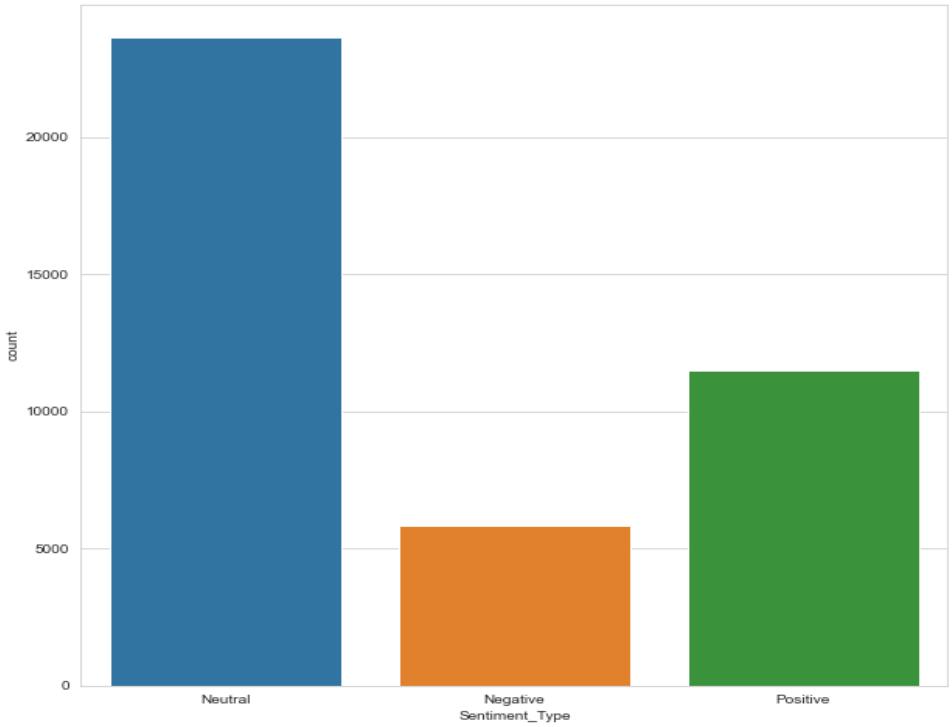

Figure 40: sentimental analysis results on titles

Sentiment Analysis based on title, proves that most of the tags on videos had mostly Neutral sentiments with Positive sentiments and less Negative sentiments.

So, from these above sentiments based on tags, description, and title we can firmly say that most of the videos which end up in the trending category tend to have a positive sentiment associated with it, so the YouTube Algorithm is designed in such a way that videos with positive title, description, tags have more chance of falling into the trending category than normal videos.

7. Conclusion

• The analysis states that for getting the video into trending page its just not important to increase the likes on the video, it depends on various factors like category, country from where its uploaded, comment count, comment disable, day laps, number of countries, time etc.
• Sometimes the video on YouTube is classified into different category than the content of the title. Classifying videos using the video title can be more significant.
• Sometimes the video on YouTube is classified into different category than the content of the title. Classifying videos using the video title can be more significant.
• By using k-mean clustering we found of similar word pattern in the titles of trending videos which can be further used to get the new videos on the search results.

8. Acknowledgements and References

- https://www.brandwatch.com/blog/youtube-stats/
- https://www.kaggle.com/datasnaek/youtube-new
- https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need- to-know-a36d136ef68
- https://datascienceparichay.com/article/save-python-dictionary-to-a-pickle-file/
- https://www.datacamp.com/tutorial/wordcloud-python
- http://scikit-learn.org/stable/modules/tree.html
- https://medium.com/@deepak.nov14/exploratory-data-analysis-on-trending- youtube-videos-india-83dcd59193a5
- https://en.wikipedia.org/wiki/Ordinary_least_squares#:~:text=In%20statistics%2C%20ordinary%20least%20squares,in%20a%20linear%20regression%20model.&text=Under%20these%20conditions%2C%20the%20method,the%20errors%20have%20finite%20variances