**B.Tech Minor Project Report**

**COT-415**

**On**

**Real Time Scrapping and Engineering over YouTube**

**By**

**ABHINAV MUDGAL (11520008)**

**PRIYAL DHIMAN (11520002)**

**MUKESH BHAKAR (11510638)**

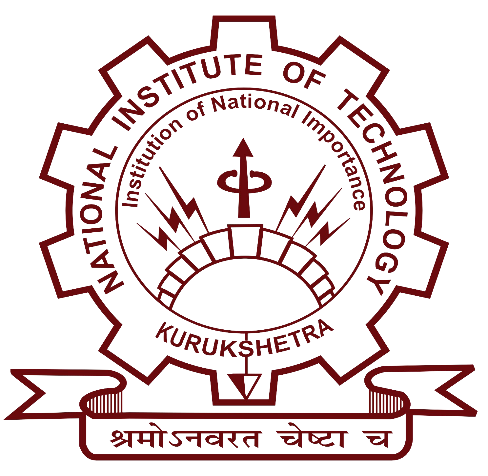
**BHAGOTI CHOUDHARY (11520005)**

**Group No.: 13**

**Under the Supervision of**

**Dr. R.M. Sharma**

**Associate Professor**

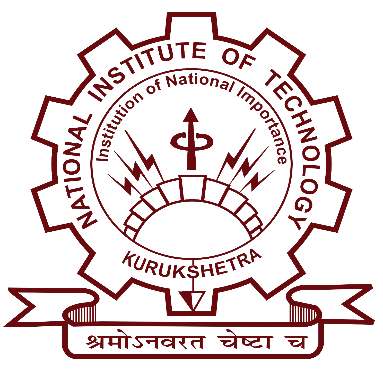
****

**DEPARTMENT OF COMPUTER ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGY**

**KURUKSHETRA-136119, HARYANA (INDIA)**

**July-Dec, 2018**



**CERTIFICATE**

I hereby certify that the work which is being presented in this B.Tech. Minor Project (COT-415) report entitled **“Real Time Data Scrapping and Engineering over YouTube”,** in partial fulfilment of the requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record of my own work carried out during a period from July 2018 to December 2018 under the supervision of **Dr. R.M. Sharma Associate Professor**, Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

*Signature of Candidate*

**ABHINAV MUDGAL (11520008)**

**PRIYAL DHIMAN (11520002)**

**MUKESH BHAKAR (11510638)**

**BHAGOTI CHOUDHARY (11520005)**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

*Date:*  *Signature of Supervisor*

**Dr. R.M. Sharma**

**Associate Professor**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Section No.** | **TITLE** | **Page No.** |
|  | ABSTRACT |  |
| I | INTRODUCTION | 1 |
| II | MOTIVATION | 2 |
| III | LITERATURE SURVEY | 3 |
|  | III.1 THESIS ON YOUTUBE VIDEO POPULARITY | 3 |
|  | III.2 RESEARCH PAPER ON TRENDS ON SOCIAL MEDIA | 3 |
| IV | APPROACH | 4 |
| V | EXPERIMENTAL SETUP | **7** |
| VI | DATA FLOW DIAGRAM | 10 |
|  | V.1 Level 0 DFD | 10 |
|  | V.2 Level 1 DFD | 10 |
|  | V.3 Level 2 DFD | 11 |
| VII | RESULTS/OBSERVATIONS | 13 |
| VIII | CONCLUSION | 19 |
|  | REFERENCES | 20 |
|  |  |  |
|  | APPENDIX: |  |
| A | COMPLETE CONTIBUTARY SOURCE CODE | 21 |
|  |  |  |

**ABSTRACT**

YouTube is one of the largest video-sharing websites with humongous number of videos on it.  This website allows its users to upload, like, view, comment over videos. A deep and clean analysis of such a huge data can help a new Youtuber to perform well and understand the trend over YouTube. Also, it will provide a great platform to train and test various machine learning models for predictive analysis over different features of a video like number of likes or number of views on a video and depict the future trends. For that, we will be building a tool that will perform following operations:

**Web Scrapping and Data collection** through YouTube data API (version 3) in real time.

**Data Cleaning and Refining** for proper structuring of raw data using python libraries like pandas and numpy.

**Feature Engineering**- Filtering out humongous amount of data to minimal relevant features/information to train the model upon.

**Data Analysis and Visualization** using matplotlib library to generate various charts and graphs depicting various trends and correlations among different attributes related to a video.

**Predictive Analysis** using different machine learning models, grid search and cross validation techniques to identify the best fit model in the real time scenario.

**I. INTRODUCTION**

YouTube is a platform where youtubers can upload their videos, share their ideas with the world and can also earn a platinum through it. YouTube has grown tremendously in India especially after the high clashes of data rates in the country. This helped in the growth of YouTube community that we see today. YouTube is responsible for making a lot of people, films, short films, animations successful through its popularity.

One important aspect of popularity of a video is its position in the current trending list of videos by YouTube. A trending video tends to become much more popular compare to a non-trending one. Now the decision for a video to come on to the trending list is solely of YouTube based on some factors that we can see through their documentation provided for the selection of trending video. This selection is highly dependent on the region of upload of the video. So, the questions may arise like –

Does a video with a high number of likes tends to appear on the trending list?

How does views of a trending video change?

What are the factors that trend a video in a particular region? Is it number of likes? Or views? Or channel subscribers?

Correlation between the attributes of a video?

The growth in the popularity?

1. Hence, a trend analysis over YouTube videos of different countries can help us to identify the kind of material that the generation is liking and going to watch. It can help a new your tuber to come along and produce a well-suited and relevant material for the society. It can help understand the trends of different countries, trends in different categories and people’s inclinations towards them. Also, the kind of growth to be seen in the likes of a video on YouTube after a certain time is sometimes exceptional or indescribable to usual terms. This again is due to the sudden increased popularity of the video.

2. A machine model can be trained onto the data taken out from the YouTube data API. This data will comprise of some old videos (trending as well as non-trending). Now, the trained model can be tested and procure good results. But what about the fact that the videos are constantly growing and changing in real time. A model predicting a particular may or may not vary according to the video. Hence multiple models can be trained and we can determine the model with the minimum variance to be highly suitable for the background.

**II. MOTIVATION**

When it comes to internet’s user created web portal for sharing videos, YouTube holds the topmost place, with more than 75,000 new videos up-loaded and more than 100 million views daily. Popularity of such sites has raised interest of individuals as well as business organizations to gather ’likes’ for their uploaded content from as large number of viewers/ subscribers as possible. From the commercial point of view, business organizations are interested in getting likes for their content for reasons ranging from financial benefits to market share among others. For example, the number of likes of a movie trailer might result in more footfalls at the theatre for the movie.

Since 60% of all online videos are watched through YouTube, it has become the best source for data scraping and data analysis. It provides a great platform to train and test various machine learning models for predictive analysis over different features of a video like view count, number of shares, number of comments etc. and have a qualitative comparison among them.

This project contributes to the topic of trend prediction by comparing the like count of a YouTube video with the number of views on the video. This ability of the most impactful and essential data retrieval and analysis of a YouTube video can be converted into knowledge of how to generate content so that it becomes more popular and gathers more attention.

**III. LITERATURE SURVEY**

**III.1 Thesis on YouTube video Popularity**

Initially we get the reference about this project from a thesis which is entitled by “YouTube Video Popularity: Predicting Video View Count From User-Controlled Features”, presented by Mr. Jordy Snijders at the Tilburg University, Tilburg in July 2015.This thesis discusses about the many social networking sites which allows user to share their content on sites which can be seen, enjoyed and appreciated by other users. In current internet era we have more than hundred sites which provides this facility in which YouTube, Facebook, twitter etc. are very famous and now-a-days people are earning from these sites.

This thesis specially focuses on popularity on YouTube which is mostly used by the users where users can share their videos and have total control over video content, description and title among other video properties. Other user can like, dislike, watch and comment on videos and can subscribe the favorite channel. But now a days in India after launching JIO uploading YouTube videos on daily basis is increased drastically so by analyzation of the real time video plays very important role. This thesis used supervised machine learning for analyzation of the YouTube videos. In supervised machine learning we used supervised regression algorithms for example:-Random forest, linear regression etc.

**III.2 Research Paper on “Trends on social Media”**

This research paper is published by Sitaram Asur and Bernardo A. Huberman and Gabor Szabo and Chunyan Wang at Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media in 2011. From this paper, we get to know that how the content of the social media go into trend and which factors are responsible to affect the same.so we pick YouTube as our website to analyze the trend. These are main factor which affect the trend of video:-

1. Type of content of video (e.g. Comedy, games, knowledge etc.)

2. Title of the video (e.g. Long title, short title)

3. Age of the channel (e.g. Old channel will get more videos)

4. Views count, like count, channel subscribers.

5. Day and Time of uploading a video (e.g. If video uploaded on Friday evening has more chances to go in trend more than other days of the week) So, the trend analysis is done using python libraries (pandas, numpy, matplotlib). We drew plots between related features of the datasets and by plotting these videos we analyze the trend.

**IV. APPROACH**

The Google API – YouTube data API v3 provides us with the data of videos along with all its attributes. Now, this data can become really helpful for us to –

1. Train and test different machine learning models for predictions like of like counts that can help us to identify the performance of different models onto the dataset. Also, how the proposed model behaves over a new trending video in real time changing scenario.

2.  Visualize different attributes of a trending video in order to analyze the variations and reasons for the recent trends.

**Data Scrapping**

YouTube data API intend to provide potential attributes corresponding to any video. The API can provide the details of a video in the JSON file with appropriate tags for corresponding parts which can be vivisected to extract relevant attributes and store in desired manner.

Hence, a python script was created to gather data from YouTube data API v3.Initial attempts at storing the data directly as JSON file resulted in failure due to frequent loss of Internet connectivity. This issue was resolved by gathering data in a SQLite database with frequent commits at regular intervals. In case of Internet failure, the script was restarted from the date till when the data was collected successfully. This resulted in a data set of attributes of more than 50000 YouTube videos.

**Data Warehousing**

The scrapped data is stored in a database using sqlite3. This data stored in the warehouse can further be manipulated, altered, used for different utilities.

**Feature Selection**

The collection of features taken from various YouTube videos, were selectively chosen to train our model. The YouTube API provided the following features:

*Google Plus Shares* - The total number of times the video has been shared on Google Plus.

*LinkedIn Shares* - The total number of times the video has been shared on Linked In.

*Latitude* - The latitude coordinates of the video’s originin degrees.

*Description* **-**The description of the video.

*Publishing Date* - The date and time the video was published. From the publishing date- a newfeature, lifetime of video was calculated.

*Title* -The video’s title.

*Category* - The video category associated with the video.

*Longitude* - The longitude coordinates of the video’sorigin in degrees.

*Total Shares* - The number of times the video has been shared on all social media platforms.

*View Count* - The number of times a video has been viewed.

*Comment Count* - The number of comments the video has received.

*Dislike Count* - The number of users who had indicated that they disliked the video.

*Duration* -The length of the video in seconds.

*Licensed Content* - Whether the video contained licensed content.

*Viability of Statistics* -Whether the video’s statisticscould be viewed by anyone.

*Favorite Count* - The number of users who have added the video to their Favorites.

*Definition* - Whether the video was available in high definition or standard definition.

If all of the above-mentioned features are considered then the results are not found to be favorable. So, we have used four relevant features (view count, comment count, duration and dislike count) to ensure that a proper data cleaning and feature selection is performed. This is being done by fitting different regression models on this data set and predicting the like counts of videos scrapping directly through YouTube.

**Model Training**

1. The features stored in the database of videos are extracted setting up a connection with sqlite3.
2. Only View count, comment count, dislike count and duration of videos are taken for training the primary model for like counts.
3. The features are scaled to the proper range and different models are fitted with them for training.
4. The scoring and predictions are observed and declared.

**Visualization**

Visualization can help a newbie to improve quality of his videos. Visualization is done by using various libraries of python. Python libraries help in manipulating data in a way it is required. So libraries like seaborn, numpy, matplotlib, glob are used to graphically represent data and use it to further analyze the pattern. Visualization is performed by plotting histograms, bar graphs and pie charts amongst various features. Graphical representation of the data is done to know the factors that can make a video trending. These factors can be found by answering some relevant and related questions:

1. Which channels have the largest number of trending videos?
2. Are views and likes correlated?
3. How many likes are generally required for a video to be trending?
4. How many views are generally required for a video to be trending?
5. How many comments are there on a trending video?
6. Does title of the video affect its trend? Does the presence of capital letter in title make a change?

Analysis is being done in order to come up with a result that can directly or indirectly help the youtuber to make his video in the trending list and hence get human attention.

**V. EXPERIMENTAL SETUP**

**Software and Hardware Requirements**

**** Python 3.0

 Anaconda

 DB Browser for SQLite

 Atom (Text Editor)

Dataset: YouTube. dB file containing data of more than 50000 videos across the world taken through YouTube data API v3 scrapper.

Data of trending videos categorized by region.

Region: [Canada, Great Britain, Germany, United States, France]

The working of some significant components is shown below using the pseudocode

Pseudo code:

1.   Import the required libraries

2.   Initialize variables

**PSEUDO CODE:**

SCRAPPING\_TOOL ():

1.   Define the time duration (Published After and Published Before) within which the data must be scrapped.

2.   Build a YouTube data API object with own authorized API key and version control information.

3.   Create and setup a connection with sqlite3 database.

4.   Search on the list of items between published after and published before.

5.   Execute on snippet part of video type and run over all the results.

6.   For every result of items tag taken from api, store the required attributes into variables.

7.   Skip and show integrity errors for old videos.

8.   Commit the results to the database and close it.

EXTRACT\_DATABASE (database name):

1.   Connect the YouTube database stored.

2.   Fetch all the features row wise using the database cursor.

3.   For every row select only like counts, view counts, comment counts, dislike counts and duration.

4.   Append like counts in numpy array Y and other features into numpy array X.

5.   Close the sqlite3 connection.

6.   Return X and Y.

Arguments:

Database name- Name of the database where the YouTube videos and channel data are stored.

REAL\_VIDEO\_FETCH (videoid):

1.   Send request to the googleapis.com/youtube/v3/videos and wait for response to the specified api\_key.

2.   Load the JSON response.

3.   Extract the numerical features from the tags and store it.

4.   Extract the actual likes of the video.

5.   Return numerical features and likes of the video in real time.

Arguments:

Videoid- A video’s id whose predictions are to be made.

INIT\_PREDICTOR (videoids):

1.   Call EXTRACT\_DATABASE method and get X and Y.

2.   Initialize the MINMAX SCALER.

3.   Iterate over the list of video ids.

4.   For every video id, Call REAL\_VIDEO\_FETCH function.

5.   Perform feature scaling using Min-Max Scaler.

6.   Fit and train various regression models onto the scaled feature set.

7.   Call TRAIN\_TEST\_CHECK on every model.

8.   Return Predicted values.

Arguments:

Videoids: List of video ids of all the real time videos on which the model needs to do predictions.

TRAIN\_TEST\_CHECK (X, Y):

1.   Split the data X and Y into training and testing part.

2.   Fit different models with different random states.

3.   Apply and store scoring results on every model.

4.   Display/Store the scoring results.

Arguments:

X – Youtube videos selected feature set

Y- set of likes of all the videos.

VISUAL\_YOUTUBE ():

1. Import libraries needed for visualization (pandas, matplotlib, seaborn, numpy, glob).
2. Fetch all the data (Data of different countries is present in different .csv files).
3. Using glob and pandas compile this data for future use.
4. Clean the data for better results. Replace null values present in the data.
5. Use pyplot from matplotlib for visualization of data.
6. Draw histogram, pie charts, bar graphs to make comparisons and realize patterns.
7. Use distplot() and barplot() function from seaborn to plot histograms and bar graphs and pie() from matplotlib to plot pie chart.
8. Analyze graphical plots and observe patterns in the outcome.

**VI. DATA FLOW DIAGRAM**

**VI.1 Level 0 DFD**

In this we provide the YouTube data, which is gathered from YouTube using YouTube API, to the predictive model and like prediction and trend analysis is done by the model.

Input YouTube data Like Prediction &

Trend Analysis

**VI.2 Level 1 DFD**

In this, data is taken from YouTube and provided to data preparing process which clean the raw data with help of python libraries (pandas, numpy). Now cleaned data is used by two processes: -

1. The cleaned data is provided to predictive process which gives predicted likes.

2. The cleaned data is provided to data visualization process which provides the trend analysis of the real time YouTube video.

Input YouTube dataraw data cleaned

data

cleaned data

Like prediction

Trend analysis

**VI.3 Level 2 DFD**

**(**a) The data is scrapped by data scrapping process and data is stored in database using SQLite. In feature selection process most, suitable features of data sets are selected and feature scaling is done by feature process using min-max scaling. The scaled data sets split into two-part training model data and testing model data and using training data the predictive model was fitted. Now by this fitted model the likes of real time video were predicted.

Database warehouse

scrapped data fetched data

input youtube scrapped relevant

data data set feature set

scaled feature set

like prediction fitted model X\_train,Y\_train

Database warehouse

(b) This DFD is of visualization part. The raw data is provided to data cleaning process which gives cleaned data to the plotting and visualization process. The graphs and plots are provided to data set variable correlation process and get the variable relations. The process trend analyzer takes the variable relations and analyses the trend.

Input Youtube data raw data cleaned data

set

graphs & plots

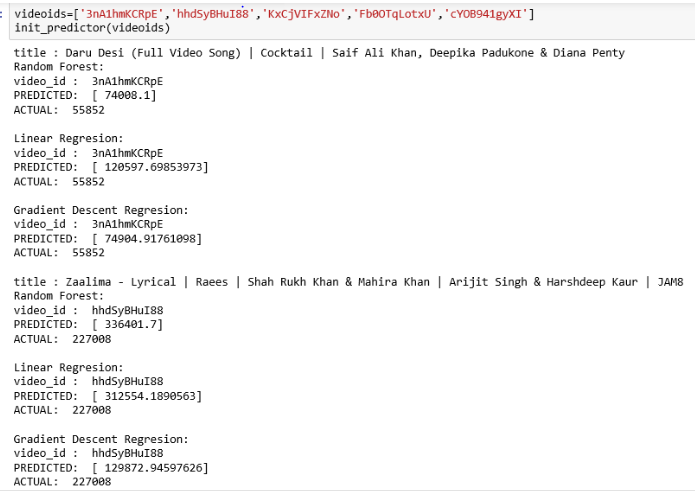
variable relation

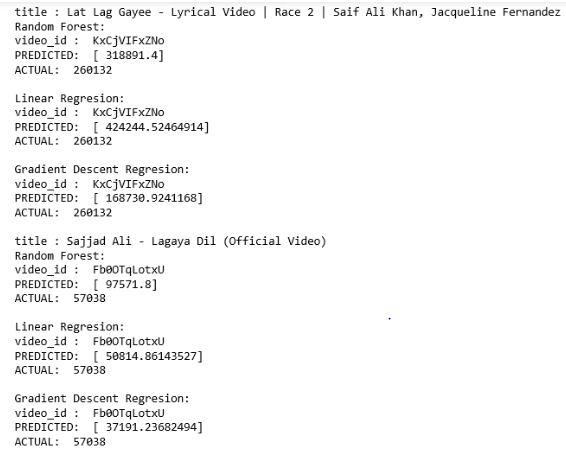
trend analysis

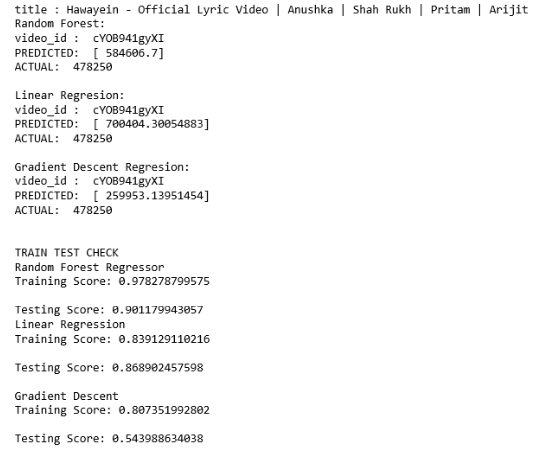
**VII. RESULTS AND OBSERVATIONS**

**PREDICTION**

A list of videoids of multiple videos taken from the respective videos URLs is passed to the init\_predictor function which further does all the other task for us. It extracts the database, fetch the data of videos given in the list in real time using real\_fetch function. The extracted YouTube videos data is used to fit multiple regressor models. Their scores are calculated after train test split and the predictions are displayed as shown below:







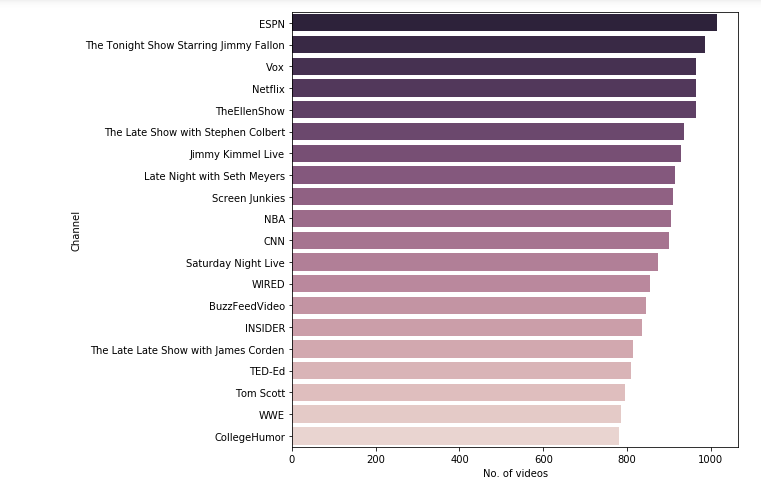
**VISUALIZATION**

Analysis is done on the below mentioned questions:

1. Which channels have the largest number of trending videos?
2. Are views and likes correlated?
3. How many likes are generally required for a video to be trending?
4. How many views are generally required for a video to be trending?
5. How many comments are there on a trending video?
6. Does title of the video affect its trend? Does the presence of capital letter in title make a change?

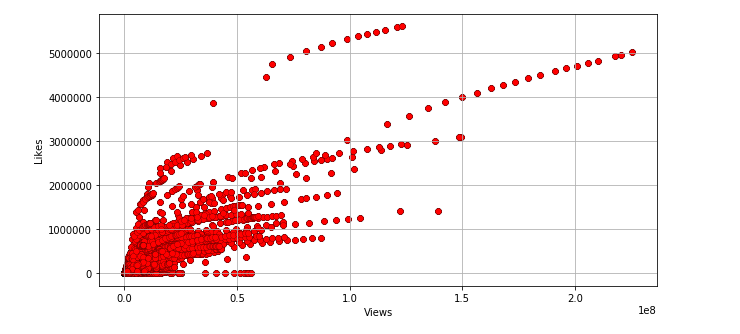
Observations are as follows:

1. **Which channels have the largest number of trending videos?**



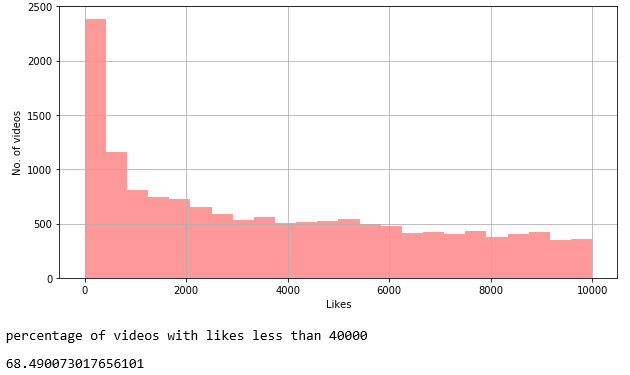
**Observation**: This bar graph enables us to know the channel whose videos usually get into trending list. So the newbie can know the type of content he/she should prefer to upload to make its videos trending.

**2. Are views and likes correlated?**

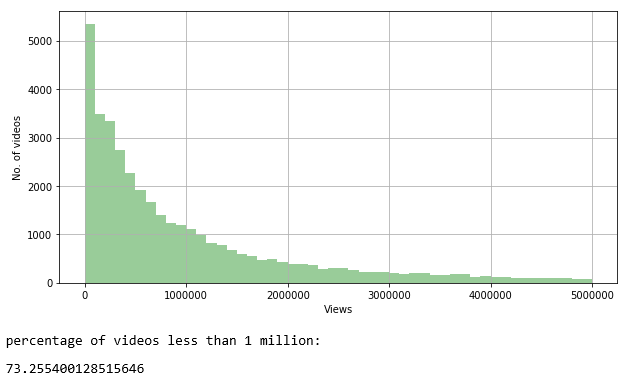
****

**Observation:** This graph shows that there is a strong co-relation between the likes and views. Likes and views are proportional i.e., as the number of views increases, the number of likes also increases.

**3. How many likes are generally required for a video to be trending?**

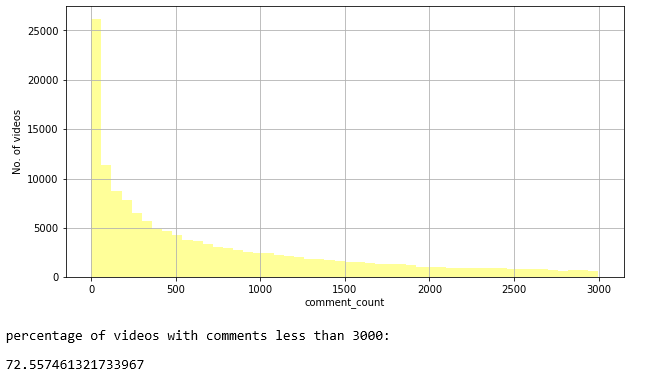
****

**Observation:** It can easily be observed that most of the trending videos get likes around 40000 and few among them gets likes greater than 40000.This 75% approximately shows the percentage of videos which got likes less than 40000.

**4. How many views are generally required for a video to be trending? **

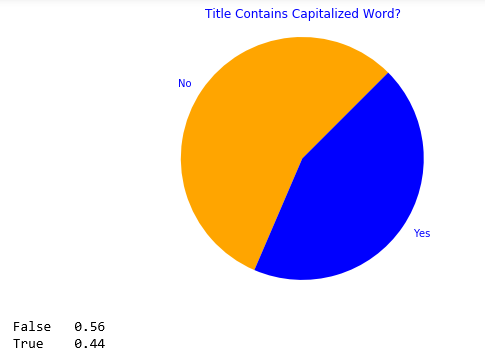
**Observation:** We can observe that majority of videos have 1 million views or less. So for a video to be trending, views should lie in the above range.

**5. How many comments are there on a trending video?**

****

**Observation:** Trending videos have approximately a maximum of 3000 comments. Majority of trending videos have comments in the range of 500-100.

**6. Does title of the video affect its trend? Does the presence of capital letter in title make a change?**

****

**Observation:** From this pie chart we can observe that 44% of the trending videos contain at least a capital word. But, from this nothing can be inferred about the trend.

**VIII. CONCLUSION**

Data and Predictive Analysis over YouTube videos helps in analyzing the trend that is currently being followed and liked by everyone. Data scraping helps in analyzing the category of videos that are liked the most and type of content people generally prefer. This humongous amount of data is further used to have a qualitative analysis of machine learning models. This is being done by using YouTube API coupled with various libraries of Python that ultimately enables us to extract data, clean it for efficient processing and then use that data for deep analysis. Now a days, YouTube has become a platform for video makers to earn a fortune and hence such trend analysis and predictive models can help youtubers to identify the kind of material they should produce that attracts the potential viewers and help them evolve in the community.

**REFERENCES**

[1] D. C. S. Jordy Snijders “YouTube Video Popularity.”(This is a Master’s thesis submitted

by C.S. Jordy at Tilburg University, Tilburg in July 2015)

[2] S. Asur, B. A. Huberman, G. Szabo, and C. Wang, “Trends in social media: persistence and

decay”, accepted by the 5th International AAAI Conference on Weblogs and Social Media, 2011.

URL: <https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2815/3205>

[3] Coding Ninjas- Machine learning course:

URL:<https://codingninjas.in/app/classroom/batch/276/wall>

[4] Open source, scikit-learn library from <http://scikit-learn.org/stable/>

[5] YouTube data API documentation: <https://developers.google.com/youtube/2.0/reference>

**APPENDIX**

**A. COMPLETE CONTIBUTARY SOURCE CODE**

**A.1 Data Scrapper Code**

#!/usr/bin/python

import strict\_rfc3339

import json

import sqlite3

import calendar

import datetime

from time import sleep

from oauth2client.tools import argparser

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import re

Connection = sqlite3.connect('youtube.db')

print "---------Database opened-------";

Connection.execute('''CREATE TABLE VIDEOS

(videoId INT PRIMARY KEY NOT NULL,

channelId INT,

channelTitle TEXT,

title TEXT,

description TEXT,

categoryId INT,

publishedAt INT,

currentTime INT,

life INT,

definition TEXT,

caption INT,

duration INT,

durationCategory TEXT,

licensedContent TEXT,

dimension INT,

allowed TEXT,

allowedCount INT,

recordingDate INT,

latitude INT,

longitude INT,

publicStatsViewable INT,

privacyStatus INT,

license TEXT,

embeddable INT,

commentCount INT,

viewCount INT,

favoriteCount INT,

dislikeCount INT,

likeCount INT);''')

print "---------TABLE CREATION DONE-------";

Connection.commit()

Connection.close()

time1 = datetime.datetime(2018, 1, 1, 0, 0, 0) # from January 1, 2018

time1 = calendar.timegm(time1.timetuple())

time2 = datetime.datetime(2018, 7, 1, 0, 0, 0) # to July 1, 2018

time2 = calendar.timegm(time2.timetuple())

time = time1

while True:

try:

youtube = build(

"youtube",

"v3",

developerKey="AIzaSyCJG4Qpa7p57i5kRPQ3ZytiP\_bud1rvvmI"

)

conn = sqlite3.connect('youtube.db')

print "Opened database successfully"

database = conn.cursor()

while True:

index = 1

pageToken = ""

videoIds = []

publishedAfter = strict\_rfc3339.timestamp\_to\_rfc3339\_utcoffset(time)

publishedBefore = strict\_rfc3339.timestamp\_to\_rfc3339\_utcoffset(time + (60\*60))

print("> %s - %s" % (publishedAfter, publishedBefore))

while True:

scrapper = youtube.search().list(

part="snippet",

type="video",

order="viewCount",

publishedAfter=publishedAfter,

publishedBefore=publishedBefore,

maxResults=50,

pageToken=pageToken,

safeSearch="none",

).execute()

for r in scrapper.get("items", []):

channelId = r["snippet"]["channelId"]

videoId = r["id"]["videoId"]

# don't process videos found in earlier batches

if videoId in videoIds:

continue

videoIds.append(videoId)

videos = youtube.videos().list(

id=videoId,

part="snippet, contentDetails, recordingDetails, statistics, status", #id, snippet, contentDetails, localizations, player, statistics, status

).execute()

for r in videos.get("items", []):

# skip live broadcasts

try:

liveBroadcastContent = r["snippet"]["liveBroadcastContent"]

if liveBroadcastContent != "none":

continue

except KeyError:

continue

# basics

videoId = videoId

channelId = channelId

channelTitle = r["snippet"]["channelTitle"]

# status

publicStatsViewable = int(r["status"]["publicStatsViewable"])

privacyStatus = r["status"]["privacyStatus"]

license = r["status"]["license"]

embeddable = int(r["status"]["embeddable"])

# snippet

title = r["snippet"]["title"]

description = r["snippet"]["description"]

categoryId = int(r["snippet"]["categoryId"])

publishedAt = r["snippet"]["publishedAt"]

publishedAt= int(strict\_rfc3339.rfc3339\_to\_timestamp(publishedAt))

currentTime = datetime.datetime.utcnow()

# current time as rtf3339

currentTime =

datetime.datetime.timetuple(currentTime)

# current time as timetuple

currentTime = calendar.timegm(currentTime)

# current time as epoch timestamp

life = currentTime - publishedAt

# statistics

try:

commentCount=int(r["statistics"]["commentCount"])

except Exception:

commentCount = None

try:

viewCount=int(r["statistics"]["viewCount"])

except Exception:

viewCount = None

try:

favoriteCount = int(r["statistics"]["favoriteCount"])

except Exception:

favoriteCount = None

try:

dislikeCount = int(r["statistics"]["dislikeCount"])

except Exception:

dislikeCount = None

try:

likeCount = int(r["statistics"]["likeCount"])

except Exception:

likeCOunt = None

# recordingDetails

try:

recordingDate = r["recordingDetails"]["recordingDate"]

recordingDate = int(strict\_rfc3339.rfc3339\_to\_timestamp(recordingDate))

except Exception:

recordingDate = None

try:

latitude = r["recordingDetails"]["location"]["latitude"]

except Exception:

latitude = None

try:

longitude= r["recordingDetails"]["location"]["longitude"]

except Exception:

longitude = None

# contentDetails

definition = r["contentDetails"]["definition"]

caption = 0 if r["contentDetails"]["caption"].lower() == 'false' else 1

duration = r["contentDetails"]["duration"]

duration\_w = re.search(r"(\d+)w", duration, re.I)

duration\_w = int(duration\_w.group(1))

if duration\_w else 0

duration\_d = re.search(r"(\d+)d", duration, re.I)

duration\_d = int(duration\_d.group(1))

if duration\_d else 0

duration\_h = re.search(r"(\d+)h", duration, re.I)

duration\_h = int(duration\_h.group(1))

if duration\_h else 0

duration\_m = re.search(r"(\d+)m", duration, re.I)

duration\_m = int(duration\_m.group(1))

if duration\_m else 0

duration\_s = re.search(r"(\d+)s", duration, re.I)

duration\_s = int(duration\_s.group(1))

if duration\_s else 0

duration = 0

duration += duration\_w \* 7 \* 24 \* 60 \* 60

duration += duration\_d \* 24 \* 60 \* 60

duration += duration\_h \* 60 \* 60

duration += duration\_m \* 60

duration += duration\_s \* 1

durationCategory = "short"

durationCategory = "medium" if duration\_m >= 4 else "short"

durationCategory = "long" if duration\_m >= 20

else "medium"

licensedContent = 0

if r["contentDetails"]["licensedContent"] == False else 1

dimension = r["contentDetails"]["dimension"]

try:

allowed = ','.join(r["contentDetails"]["regionRestriction"]["allowed"])

except Exception:

allowed = None

try:

allowedCount = len(r["contentDetails"]["regionRestriction"]["allowed"])

except Exception:

allowedCount = 0

sql\_insert\_vars = (videoId,channelId,channelTitle,

title,description,categoryId,publishedAt,

currentTime,life,definition,caption,duration,

durationCategory,licensedContent,dimension,

allowed,allowedCount,recordingDate,

latitude,longitude,publicStatsViewable,

privacyStatus,license,embeddable,

commentCount,viewCount,favoriteCount,

dislikeCount,likeCount,

)

try:

database.execute("INSERT INTO VIDEOS VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)", (sql\_insert\_vars))

#print "Record inserted successfully";

except sqlite3.IntegrityError:

print('sqlite3.IntegrityError: videoId=%s' % videoId)

print("%s - %s --- #%d --- %s" % (publishedAfter, publishedBefore, index, videoId.encode('utf-8'))) # --- %s , title.encode('utf-8')

index += 1

conn.commit()

#print "SUCCESSFULLY COMMITED"

try:

pageToken = scrapper["nextPageToken"]

except KeyError:

pageToken = None

break

time += (60\*60)

if time >= time2:

break

conn.close()

break

except HttpError, e:

print("An HTTP error has %d occurred:\n%s" % (e.resp.status, e.content))

conn.close()

#print("Waiting for 60 seconds...")

sleep(60)

pass

**A.2 Predictor Code**

import math

import sqlite3

import numpy as np

import json

import urllib

import datetime

import calendar

import re

import sys

from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import OneHotEncoder, StandardScaler,MinMaxScaler

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import SGDRegressor

import warnings

warnings.filterwarnings('ignore')

#Extract Database Function

def extract\_database(database\_name):

db\_connection = sqlite3.connect(database\_name)

db\_crsr = db\_connection.cursor()

X = []

Y = [] # LikeCount

try:

db\_crsr.execute("Select likeCount,viewCount,commentCount,dislikeCount,duration from Videos")

rows=db\_crsr.fetchall()

for row in rows:

Y.append(row[0] if row[0] else 0) #likeCount

# numerical features

viewCount=row[1] if row[1] else 0

commentCount=row[2] if row[2] else 0

dislikeCount=row[3] if row[3] else 0

duration=row[4] if row[4] else 0

X.append([

viewCount,

commentCount,

dislikeCount,

duration,

])

except (sqlite3.OperationalError,e):

print ('sqlite3.OperationalError:',e)

db\_connection.close()

return X,Y

#Real\_Video\_Fetch function

def real\_video\_fetch(video\_id):

api\_key="AIzaSyAkNjqcFjNT86o-m3uloLS-EzhR1aCtlQE"

numerical\_features\_real=[]

url = "https://www.googleapis.com/youtube/v3/videos?id=" + video\_id + "&key=" + api\_key + "&part=status,statistics,contentDetails,snippet"

response = urllib.request.urlopen(url).read()

data = json.loads(response)

all\_data = data['items']

#print (all\_data)

#Snippet

channelId = all\_data[0]['snippet']['channelId']

channelTitle = all\_data[0]['snippet']['channelTitle']

title = all\_data[0]['snippet']['title']

print ('title :',title)

description = all\_data[0]['snippet']['description']

category\_id = all\_data[0]['snippet']['categoryId']

publishedAt = all\_data[0]['snippet']['publishedAt']

#publishedAt = int(strict\_rfc3339.rfc3339\_to\_timestamp(publishedAt))

currentTime = datetime.datetime.utcnow() # current time as rtf3339

currentTime = datetime.datetime.timetuple(currentTime) # current time as timetuple

currentTime = calendar.timegm(currentTime) # current time as epoch timestamp

#life = currentTime - publishedAt

#Content Details

defintion = all\_data[0]['contentDetails']['definition']

caption = all\_data[0]['contentDetails']['caption']

licensedContent = all\_data[0]['contentDetails']['licensedContent']

dimension = all\_data[0]['contentDetails']['dimension']

duration = all\_data[0]['contentDetails']['duration']

duration\_w = re.search(r"(\d+)w", duration, re.I)

duration\_w = int(duration\_w.group(1)) if duration\_w else 0

duration\_d = re.search(r"(\d+)d", duration, re.I)

duration\_d = int(duration\_d.group(1)) if duration\_d else 0

duration\_h = re.search(r"(\d+)h", duration, re.I)

duration\_h = int(duration\_h.group(1)) if duration\_h else 0

duration\_m = re.search(r"(\d+)m", duration, re.I)

duration\_m = int(duration\_m.group(1)) if duration\_m else 0

duration\_s = re.search(r"(\d+)s", duration, re.I)

duration\_s = int(duration\_s.group(1)) if duration\_s else 0

duration = 0

duration += duration\_w \* 7 \* 24 \* 60 \* 60

duration += duration\_d \* 24 \* 60 \* 60

duration += duration\_h \* 60 \* 60

duration += duration\_m \* 60

duration += duration\_s \* 1

durationCategory = "short"

durationCategory = "medium" if duration\_m >= 4 else "short"

durationCategory = "long" if duration\_m >= 20 else "medium"

try:

allowed = ','.join(all\_data[0]["contentDetails"]["regionRestriction"]["allowed"])

except Exception:

allowed = None

try:

allowedCount = len(all\_data[0]["contentDetails"]["regionRestriction"]["allowed"])

except Exception:

allowedCount = 0

# recordingDetails

try:

recordingDate = all\_data[0]["recordingDetails"]["recordingDate"]

recordingDate = int(strict\_rfc3339.rfc3339\_to\_timestamp(recordingDate))

except Exception:

recordingDate = None

try:

latitude = all\_data[0]["recordingDetails"]["location"]["latitude"]

except Exception:

latitude = None

try:

longitude = all\_data[0]["recordingDetails"]["location"]["longitude"]

except Exception:

longitude = None

# status

publicStatsViewable = int(all\_data[0]['status']['publicStatsViewable'])

privacyStatus = all\_data[0]['status']['privacyStatus']

license = all\_data[0]['status']['license']

embeddable = int(all\_data[0]['status']['embeddable'])

#Statistics

commentCount = int(all\_data[0]['statistics']['commentCount'])

viewCount = int(all\_data[0]['statistics']['viewCount'])

favoriteCount = int(all\_data[0]['statistics']['favoriteCount'])

likeCount = int(all\_data[0]['statistics']['likeCount'])

dislikeCount = int(all\_data[0]['statistics']['dislikeCount'])

numerical\_features\_real.append([

viewCount,

commentCount,

dislikeCount,

duration,

])

return (numerical\_features\_real,likeCount)

#INIT\_Predictor

def init\_predictor(videoids):

X,Y=extract\_database('youtube.db')

scaler = MinMaxScaler(feature\_range=(-2,2))

for videoid in videoids:

x\_test\_real,likeCount=real\_video\_fetch(videoid)

#Feature Scaling

numerical\_features\_total=np.append(X,x\_test\_real,axis=0)

X\_total\_scaled=scaler.fit\_transform(numerical\_features\_total) #numerical\_features

x\_test\_real\_scaled=X\_total\_scaled[X\_total\_scaled.shape[0]-1]

x\_test\_real\_scaled=x\_test\_real\_scaled.reshape(1,-1)

X\_scaled=X\_total\_scaled[:X\_total\_scaled.shape[0]-1] #removing real

#diff regressors

print ('Random Forest: ')

rfg=RandomForestRegressor()

rfg.fit(X\_scaled,Y)

print ('video\_id : ', videoid)

print('PREDICTED: ',rfg.predict(x\_test\_real\_scaled))

print('ACTUAL: ',likeCount,'\n')

lr=LinearRegression()

lr.fit(X\_scaled,Y)

print ('Linear Regresion: ')

print ('video\_id : ', videoid)

print('PREDICTED: ',lr.predict(x\_test\_real\_scaled))

print('ACTUAL: ',likeCount,'\n')

sgd=SGDRegressor()

sgd.fit(X\_scaled,Y)

print ('Gradient Descent Regresion: ')

print ('video\_id : ', videoid)

print('PREDICTED: ',sgd.predict(x\_test\_real\_scaled))

print('ACTUAL: ',likeCount,'\n')

if videoid==videoids[len(videoids)-1]:

train\_test\_check(X\_scaled,Y)

#Train\_Test\_Check

def train\_test\_check(x,y):

xtrain,xtest,ytrain,ytest = train\_test\_split(x,y,train\_size=.9,random\_state=2)

xtrain=np.array(xtrain)

ytrain=np.array(ytrain)

ytest=np.array(ytest)

rfg=RandomForestRegressor(random\_state=2)

lr=LinearRegression()

sgd=SGDRegressor (max\_iter=1500)

rfg.fit(xtrain,ytrain)

print('\nTRAIN TEST CHECK\nRandom Forest Regressor\nTraining Score:',rfg.score(xtrain,ytrain))

print('\nTesting Score:',rfg.score(xtest,ytest))

lr.fit(xtrain,ytrain)

print('Linear Regression\nTraining Score:',lr.score(xtrain,ytrain))

print('\nTesting Score:',lr.score(xtest,ytest),'\n')

sgd.fit(xtrain,ytrain)

print('Gradient Descent\nTraining Score:',sgd.score(xtrain,ytrain))

print('\nTesting Score:',sgd.score(xtest,ytest),'\n')

#Calling the predictor

videoids=['3nA1hmKCRpE','hhdSyBHuI88','KxCjVIFxZNo','Fb0OTqLotxU','cYOB941gyXI']

init\_predictor(videoids)

**A.3 Visualizer Code**

import glob

import seaborn as sns

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

data\_files = [i for i in glob.glob('\*.{}'.format('csv'))]

print(data\_files)

data\_frame = list()

for file in data\_files:

df = pd.read\_csv(csv, index\_col='video\_id')

df['country'] = file[0:3]

data\_frame.append(df)

final\_data = pd.concat(data\_frame)

final\_data.head(3)

final\_data[final\_data["description"].apply(lambda x: pd.isnull(x))].head(3)

final\_data["description"] = final\_data["description"].fillna(value="")

Distribution of views among videos:

figure, ax = plt.subplots(figsize=(15,5))

figure = sns.distplot(final\_data["views"], color="green", kde=False, ax=ax)

figure = ax.set(xlabel="Views", ylabel="No. of videos", xticks=np.arange(0, 2.0e8, 1e7))

figure = plt.xticks(rotation=90)

plt.grid(axis="both")

plt.show(figure)

Most of the trending videos have 5 million or less views.So let's plot the histogram where views are 25 million or less.

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(df[df["views"] < 5e6]["views"], kde=False, color="green", ax=ax)

figure = ax.set(xlabel="Views", ylabel="No. of videos")

plt.grid(axis='both')

plt.show(figure)

final\_data[final\_data['views'] < 1e6]['views'].count() / final\_data['views'].count() \* 100

Distribution of likes among different videos:

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(df["likes"], color="red",rug=False,kde=False,ax=ax)

figure = ax.set(xlabel="Likes", ylabel="No. of videos")

figure = plt.xticks(rotation=90)

plt.grid(axis='both')

plt.show(figure)

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(df[df["likes"] <= 1e4]["likes"], color="red",kde=False,

ax=ax)

figure = ax.set(xlabel="Likes", ylabel="No. of videos")

plt.grid(axis='both')

plt.show(figure)

print("percentage of videos with likes less than 40000")

final\_data[final\_data['likes'] < 4e4]['likes'].count() / final\_data['likes'].count() \* 100

It can easily be observed that most of the trending videos get likes around 40000 and few amomg them gets likes greater than 40000.This 75% approximately shows the percentage of videos which got likes less than 40000.

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(final\_data["comment\_count"], color="yellow",kde=False,ax=ax)

figure = ax.set(xlabel="comment\_count", ylabel="No. of videos")

figure = plt.xticks(rotation=90)

plt.grid(axis='both')

plt.show(figure)

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(final\_data[final\_data["comment\_count"] < 3000]["comment\_count"], color="yellow",kde=False,

ax=ax)

figure = ax.set(xlabel="comment\_count", ylabel="No. of videos")

plt.grid(axis='both')

plt.show(figure)

print("Since most of the trending videos have around 25000/8 = around 3000 comments...since since each division in the graph has six histogram bins.")

print("percentage of videos with likes less than 3000:")

final\_data[final\_data['comment\_count'] < 3000]['comment\_count'].count() / final\_data['comment\_count'].count() \* 100

final\_data["title\_length"] = final\_data["title"].apply(lambda x: len(x))

figure, ax = plt.subplots(figsize=(10,5))

figure = sns.distplot(final\_data["title\_length"],kde=False,

color="blue", ax=ax)

figure = ax.set(xlabel="Title Length", ylabel="No. of videos", xticks=range(0, 110, 10))

plt.grid(axis='both')

plt.show(figure)

figure, ax = plt.subplots(figsize=(10,5))

figure = plt.scatter(x=final\_data['views'], y=final\_data['likes'], color="red", edgecolors="black", linewidths=0.2)

figure = ax.set(xlabel="Views", ylabel="Likes")

plt.grid(axis='both')

plt.show()

cdf = final\_data.groupby("channel\_title").size().reset\_index(name="video\_count") \

.sort\_values("video\_count", ascending=False).head(20)

figure, ax = plt.subplots(figsize=(8,8))

figure = sns.barplot(x="video\_count", y="channel\_title", data=cdf,

palette=sns.cubehelix\_palette(n\_colors=20, reverse=True), ax=ax)

figure = ax.set(xlabel="No. of videos", ylabel="Channel")

plt.show()

def contains\_capitalized\_word(s):

for w in s.split():

if w.isupper():

return True

return False

final\_data["capital\_word"] = final\_data["title"].apply(contains\_capitalized\_word)

value\_counts = final\_data["capital\_word"].value\_counts().to\_dict()

figure, ax = plt.subplots(figsize=(10,5))

figure= ax.pie([value\_counts[False], value\_counts[True]], labels=['No', 'Yes'], colors=['orange', 'blue'], textprops={'color': 'blue'}, startangle=45)

figure = ax.axis('equal')

figure = ax.set\_title('Title Contains Capitalized Word?',color='blue')

plt.show()

final\_data["capital\_word"].value\_counts(normalize=True)