RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

MITA CAPSTONE PROJECT (Prof. Sergei Schreider)

“YouTube Trending Video Data Analysis & Visualization”

Project report submitted by team:

Abhishek Jaiswal (197007548)
Mohit Shukla (195009793)
Rohan Gandhi (196001680)

YouTube Trending Video and Sentimental Analysis

December 1, 2020

1.0.1 Group Members

1. Mohit Milind Shukla
2. Abhishek Jaiswal
3. Rohan Gandhi

1 Analysis of YouTube trending videos

1.1 Introduction

YouTube is the most popular and most used video platform in the world today. YouTube has a list of **trending videos** that is updated constantly. Here we will use **Python** with some packages like **Pandas** and **Matplotlib** to analyze a dataset that was collected over 20 days. For each of those days, the dataset contains data about the trending videos of that day. It contains data about **more than 1000 trending videos**. We will analyze this data to get insights into YouTube trending videos, to see what is common between these videos. Those insights might also be used by people who want to increase popularity of their videos on YouTube.

The dataset that we will use is obtained from YouTube API. It contains data about trending videos for many countries. Here we will analyze USA trending videos.

1.2 Goals of the project

We want to answer questions like: * What makes a YouTube video go trending? * Which YouTube channels have the largest number of trending videos? * What is the gross viewer feedback of a particular video (obtained by running Sentiment Analysis on comments left by viewers).

1.3 Target Audience

Our target audience are the content creators on YouTube.

1.4 Importing some packages.

First, we import some Python packages that will help us analyzing the data, especially pandas for data analysis and matplotlib for visualization.

[1]:

```
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind

from mpl_toolkits.mplot3d import Axes3D

import warnings
from collections import Counter
import datetime
import json
from pandas.io.json import json_normalize
import datetime
from datetime import datetime
import mpld3
mpld3.enable_notebook()
```

2 Script to collect live data from YouTube API

import requests, sys, time, os, argparse import mysql.connector #List of simple to collect features snippet_features = [“title”, “publishedAt”, “channelId”, “channelTitle”, “categoryId”]

#Any characters to exclude, generally these are things that become problematic in CSV files unsafe_characters = [‘’, ‘"’]

#Used to identify columns, currently hardcoded order header = [“video_id”] + snippet_features + [“trending_date”, “tags”, “view_count”, “likes”, “dislikes”, “comment_count”, “thumbnail_link”, “comments_disabled”, “ratings_disabled”, “description”] def setup(api_path, code_path): with open(api_path, ‘r’) as file: api_key = file.readline().strip()

with open(code_path) as file: country_codes = [x.rstrip()

for x in file] return api_key, country_codes

def prepare_feature(feature): # Removes any character from the unsafe characters list and surrounds the whole item in quotes for ch in unsafe_characters: feature = str(feature).replace(ch, "“) return f”’{feature}"’ def api_request(page_token, country_code): # Builds the URL and requests the JSON from it request_url = f“https://www.googleapis.com/youtube/v3/videos?part=id,statistics,snippet{page_token}chart= request = requests.get(request_url) if request.status_code == 429: print(“Temp-Banned due to excess requests, please wait and continue later”) sys.exit() if request.status_code == 400: print(“Error”) sys.exit() return request.json()

def setup_db(conn): cur = conn.cursor() table_schema = ‘CREATE TABLE IF NOT EXISTS %s_videos (video_id VARCHAR(20) NOT NULL PRIMARY KEY, title TEXT, publishedAt DATETIME,channelId VARCHAR(50), channelTitle TEXT,categoryId INT,trending_date DATETIME,tags TEXT,view_count INT,likes

INT,dislikes INT,comment_count INT, thumbnail_link TEXT,comments_disabled TEXT,ratings_disabled TEXT,description TEXT) DEFAULT CHARSET=utf8;' for country_code in country_codes: #query_str = "CREATE TABLE IF NOT EXISTS %s_videos (%s);" % (country_code,",".join(x + ' TEXT' for x in header)) cur.execute(table_schema % (country_code)) conn.commit() import pdb if **name** == "**main**":

parser = argparse.ArgumentParser() parser.add_argument('--key_path', help='Path to the file containing the api key, by default wil parser.add_argument('--country_code_path', help='Path to the file containing the list of countr parser.add_argument('--db_name', help='Name of the database file where to store the data', defa parser.add_argument('--output_dir', help='Path to save the outputted files in', default='output args = parser.parse_args()

output_dir = args.output_dir db_name = args.db_name api_key, country_codes = setup(args.key_path, args.country_code_path) if not os.path.exists(output_dir):
    os.makedirs(output_dir)
conn = mysql.connector.connect(user='root',host='127.0.0.1',database='youtube_data') setup_db(conn) get_data() conn.close()

The above script collects data from Official Youtube API and saves the data obtained on Amazon web server. We connect to the Youtube API by providing a unique API key.

alt text

Here we can see data collected for different regions like US, Europe, Japan, Mexico etc From the timestamps we can observe that the script runs once every 5 minutes and if new data is present it is stored or else no operation is performed.

alt_text

[2]:

```
# Hiding warnings for cleaner display
warnings .filtcrwarnings( 'ignorc')

# Configuring some options
%matplotlib inline
%config                              = 'retina'
# If you want interactive plots, uncomment the next line
# %matplotlib notebook
```

2.1 Reading the dataset

Then we read the dataset file which is in csv format

[3]:

```
df = pd.read_csv('us_videos_new.csv') non_df =
pd.read_csv('non_trending_us_videos.csv')
```

2.2 Getting a feel of the dataset

Let's get a feel of our dataset by displaying its first few rows For
Trending Videos.

Here we can get an overview of different columns in our dataset like trending_date, view_count, Likes, dislikes etc

[4]:
```
df.head()
```

[4]:
```
          video_id                                                            title \
0  iN3ttHug-BU I've been Banned from Fortnite (I'm Sorry)
1  qjsU5876iB0 HIGHLIGHTS | Canelo vs. Sergey Kovalev 2 93qq-6Sydsk      Patriots vs.
   Ravens Week 9 Highlights | NFL 2020
3 no6hSNBB32w Jason Mitchell Speaks On Misconduct Allegation... 4
xiXusdahIPw Boat Stuck At Niagara Falls For More Than 100 ...

         published_at                  channel_id                    channel_title \
0 03-11-2020 17:32 UCvxfEIG3PHpgM0TMJJ_SH-w                                Jarvis
1 03-11-2020 06:42 UCurvRE5fGcdUgCYWgh-BDsg                             DAZN USA
2 04-11-2020 04:30 UCDVYQ4Zhbm3S2dlz7P1GBDg                                  NFL
3   04-11-2020 13:03 UChi08h4577eFsNXGd3sxYhw Breakfast Club Power 105.1 FM
4   04-11-2020 00:26 UCeY0bbntWzzVIaj2z3QigXg      NBC News

     category_id           trending_date \
0              20 05-11-2020 01:00
1              17 05-11-2020 01:00
2              17 05-11-2020 01:00
3              24 05-11-2020 01:00
4              25 05-11-2020 01:00

                                                      tags view_count          likes \
0   faze kay little brother|jarvis|fortnite kid|fo...   2787171 236563
1   canelo|canelo kovalev|dazn|boxing|sergey koval...      4588294          45219
2   sp:ty=high|sp:dt=2020-11-04T01:20:00Z|sp:st=fo...      2645930          34320
3   the breakfast club|breakfast club|power1051|ce...        378778 6600
4   Nightly News|World|NBC Nightly News with Leste... 549085 5302

      dislikes comment_count                                        thumbnail_link \
0     34681     106245     https://i.ytimg.com/vi/iN3ttHug-BU/default.jpg      1      4756      9584
https://i.ytimg.com/vi/qjsU5876iB0/default.jpg    2    1848    10807    https://i.ytimg.com/vi/93qq-
6Sydsk/default.jpg 3 665 6555 https://i.ytimg.com/vi/no6hSNBB32w/default.jpg
 4 502      803 https://i.ytimg.com/vi/xiXusdahIPw/default.jpg comments_disabled rating_disabled \
0                   False                 False
1                   False                 False
2                   False                 False
3                   False                 False
4                   False                 False

                                              description
```

```
0                                                    I'm sorry

1 Big fights. Any device. One price. DAZN is the... 2 The New
England Patriots take on the Baltimore...
3   Jason Mitchell drops in to talk the sexual mis...
4   Heavy rains and wind managed to move a massive...
```

For Non-Trending Videos

This dataframe contains the non trending videos

[5]:
```
non_df.head()
```

[5]:
```
          video_id                                                         title \
 0  KQHHF-IQFE8 Carmelo Anthony Drops Season-HIGH 25 Points Fu...
 1  -7heK6LRfLU Exclusive Audio: Jay Leno Dines With Ukraine P... 2   #NAME? B/R
    Countdown: LeBron James All-Time Triple Do... 3 1b30OxMjrWo Nice Garry! Lyon
    leads clinic ahead of Adelaid...
 4 fKAixUpvJD8                Media lauds McGahn decision as silver bullet


          published_at                          channel_id \
0   26-11-2020 03:12 UCqQo7ewe87aYAe7ub5UqXMw
1   22-11-2020 08:35 UCMtFAi84ehTSYSE9XoHefig
2   22-11-2020 17:38 UC9-OpMMVoNP5o10_Iyq7Ndw
3   27-11-2020 07:45 UCkBY0aHJP9BwjZLDYxAQrKg
4   27-11-2020 03:53 UCXIJgqnII2ZOINSWNOGFThA


                                 channel_title category_id            trending_date \
0                           House of Highlights             17 01-12-2020 19:53
1 The Late Show with Stephen Colbert                         24 01-12-2020 19:53
2                               Bleacher Report              17 01-12-2020 19:53
3                                cricket.com.au              17 01-12-2020 19:53
4                                     Fox News               25 01-12-2020 19:53
                                                                tags view_count likes \
0 carmelo anthony|carmelo anthony full game high...                   82520 1660
1 The Late Show|Late Show|Stephen Colbert|Steven...                 235641 2538
2 bleacher report|br|nba|lebron james|lebron jam...                   8344        208
3                                                          [none]        13400       482
4 politics|personality|politics|trump_impeachmen...                 48854       816

   dislikes comment_count                                       thumbnail_link \
0               15583 https://i.ytimg.com/vi/KQHHF-IQFE8/default.jpg
1               139 264 https://i.ytimg.com/vi/-7heK6LRfLU/default.jpg 2 5 28 https://i.ytimg.com/vi/-
                ZnV8WDBAwo/default.jpg
3               1725 https://i.ytimg.com/vi/1b30OxMjrWo/default.jpg
4               46272 https://i.ytimg.com/vi/fKAixUpvJD8/default.jpg

   comments_disabled rating_disabled \
0                    False  False
1                    False  False
2                    False  False
3                    False  False
4                    False  False

                                           description 0
 Portland Trail Blazers vs Chicago Bulls - Full...
1    The Late Show has acquired the audio from Jay ...
2    The King will be remembered as one of the all-...
3    Australia spinner Nathan Lyon led a clinic wit... 4 Reaction and analysis from Claremont Institute...
```

Now, let’s see some information about our dataset

2.3 Dataframe Statistics

For Trending Videos

```
[6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1122 entries, 0 to 1121 Data
columns (total 16 columns):
video_id                 1122 non-null object
title                    1119 non-null object
published_at             1122 non-null object
channel_id               1122 non-null object
channel_title            1122 non-null object
category_id              1122 non-null int64
trending_date            1122 non-null object
tags                     1122 non-null object
view_count               1122 non-null int64
likes                    1122 non-null int64
dislikes                 1122 non-null int64
comment_count            1122 non-null int64
thumbnail_link           1122 non-null object
comments_disabled        1122 non-null bool
rating_disabled          1122 non-null bool
description              1113 non-null object
dtypes: bool(2), int64(5), object(9) memory usage:
125.0+ KB
```

We can see that there are 1122 entries in the dataset. We can see also that all columns in the dataset are complete (i.e. they have 1122 non-null entries) except description & title columns which have some null values; it only has 9 & 3 null values respectively.

For Non-Trending Videos

```
[7]: non_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 473 entries, 0 to 472 Data
 columns (total 16 columns):
video_id                 473 non-null object
title                    471 non-null object
published_at             473 non-null object
channel_id               473 non-null object
channel_title            469 non-null object
category_id              473 non-null int64
trending_date            473 non-null object
tags                     473 non-null object
view_count               473 non-null int64
likes                    473 non-null int64
dislikes                 473 non-null int64
comment_count            473 non-null int64
thumbnail_link           473 non-null object
comments_disabled        473 non-null bool
rating_disabled          473 non-null bool
description              461 non-null object
dtypes: bool(2), int64(5), object(9) memory usage:
 52.8+ KB
```

We can see that there are 473 entries in the **non-trending** dataset. We can see also that all columns in the dataset are complete (i.e. they have 473 non-null entries) except description, title & Channel Title columns which have some null values; it only has 2, 12 & 4 null values respectively.

We set some configuration options just for improving visualization graphs; nothing crucial

```
[8]: PLOT_COLORS = ["#268bd2", "#0052CC", "#FF5722", "#b58900", "#003f5c"]
 pd.options.display.float_format = '{:.2f}'.format sns.set(style="ticks") plt.rc('figure', figsize=(8,
 5), dpi=100) plt.rc('axes', labelpad=20, facecolor="#ffffff", linewidth=0.4, grid=True,␣
   ↪labelsize=14) plt.rc('patch', linewidth=0) plt.rc('xtick.major',
 width=0.2) plt.rc('ytick.major', width=0.2) plt.rc('grid',
 color='#9E9E9E', linewidth=0.4) plt.rc('font', family='Arial',
 weight='400', size=10) plt.rc('text', color='#282828') plt.rc('savefig',
 pad_inches=0.3, dpi=300)
```

2.4 Description of numerical columns

Now, let's see some statistical information about the numerical columns of our dataset

```
[9]: df.describe()
```

```
[9]:               category_id view_count              likes dislikes comment_count
 count          1122.00         1122.00      1122.00   1122.00          1122.00
 mean                19.62 1249435.64       65064.04   1854.96          5846.89
 std                 7.20 1880014.14 115281.14          5494.31         12652.46
```

min	1.00	0.00	0.00	0.00	0.00
25%	17.00	417417.25	13155.25	319.25	1442.00
50%	23.00	729990.00	30999.50	636.50	2775.00
75%	24.00	1288713.25	65994.00	1473.25	5449.50
max	29.00	22635062.00	1650388.00	91081.00	266924.00

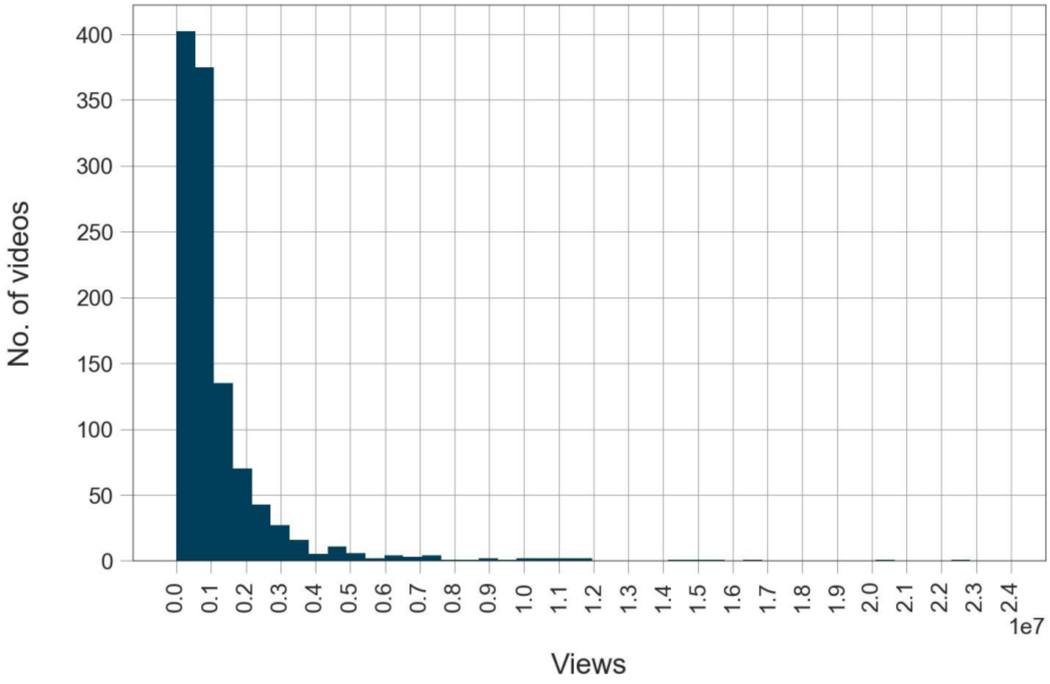
We note from the table above that - The average number of views of a video is 1,249,435. The median value for the number of views is 729,990, which means that half the videos have views that are less than that number, and the other half have views larger than that number - The average number of likes of a trending video is 65,064, while the average number of dislikes is 1,854. The - Average comment count is 5846 while the median is 2,775

How useful are the observations above? Do they really represent the data? Let's examine more.

2.4.1 Views histogram

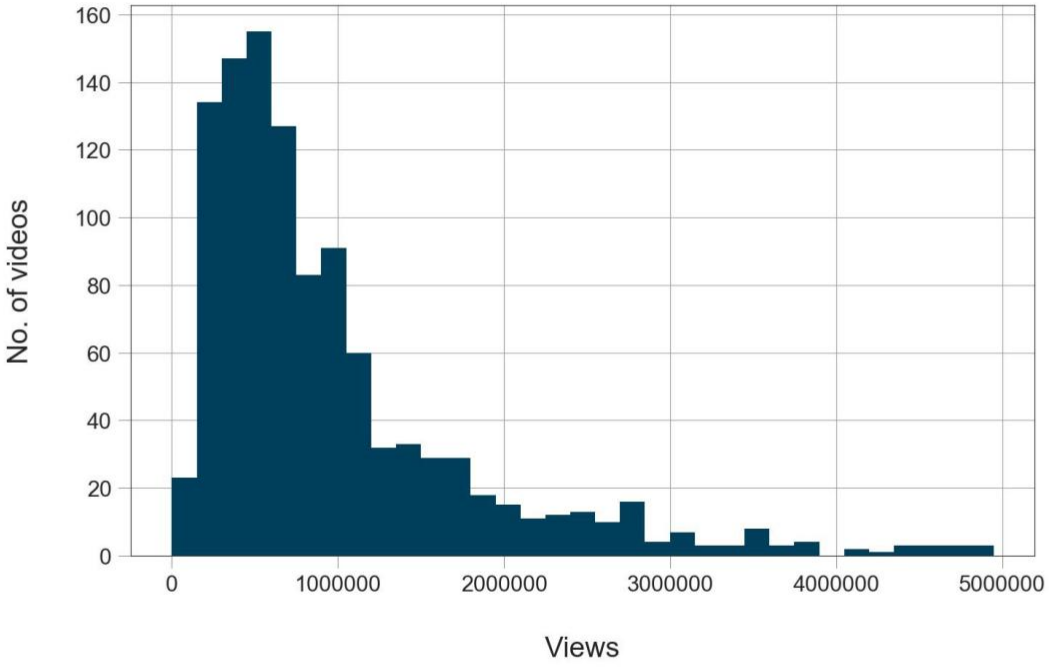
let's plot a histogram for the views column to take a look at its distribution

```
[10]: fig, ax = plt.subplots()
_ = sns.distplot(df["view_count"], kde=False, color=PLOT_COLORS[4], hist_kws={'alpha': 1},
                 bins=np.linspace(0, 2.5e7, 47), ax=ax)
_ = ax.set(xlabel="Views", ylabel="No. of videos", xticks=np.arange(0, 2.5e7,␣
↪1e6))
_ = ax.set_xlim(right=2.5e7)
_ = plt.xticks(rotation=90)
```



We note that the vast majority of trending videos have 5 million views or less. We get the 5 million number by calculating

[11]:
```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["view_count"] < 5e6]["view_count"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Views", ylabel="No. of videos")
```



Now we see that the majority of trending videos have 2 million views or less. Let’s see the exact percentage of videos less than 1 million views

[12]:
```
df[df['view_count'] < 2e6]['view_count'].count() / df['view_count'].count() *␣
↪100
```

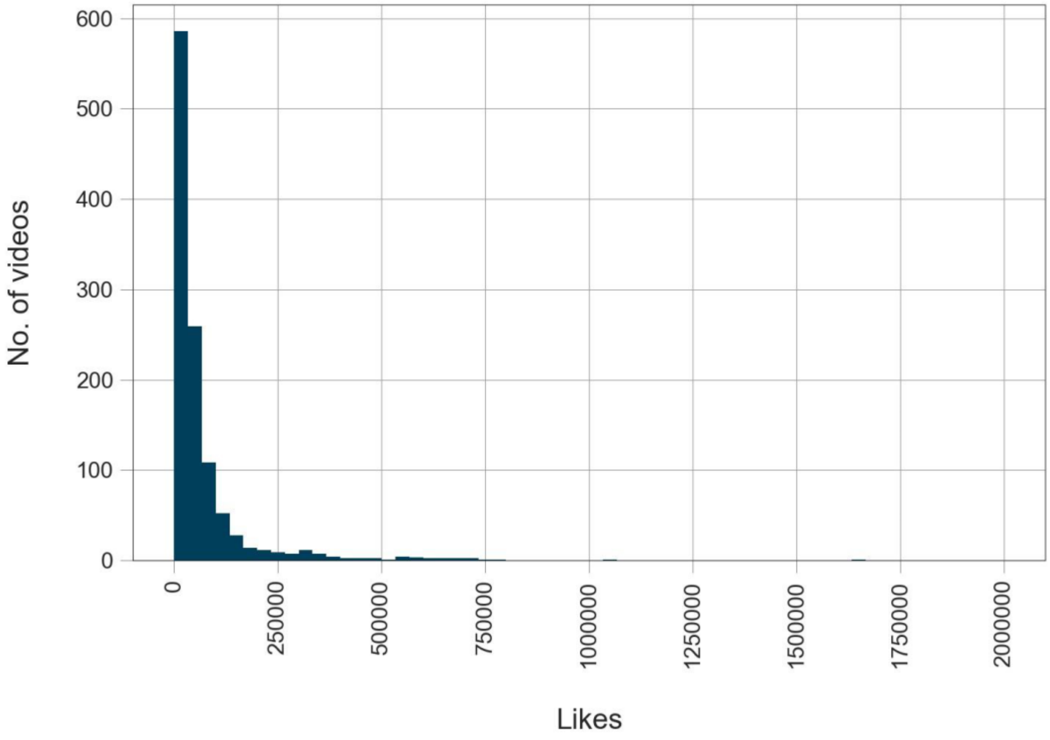
[12]: 86.36363636363636

So, it is around 86%. Similarly, we can see that the percentage of videos with less than 1.5 million views is around 78%, and that the percentage of videos with less than 5 million views is around 96%.

2.4.2 Likes histogram

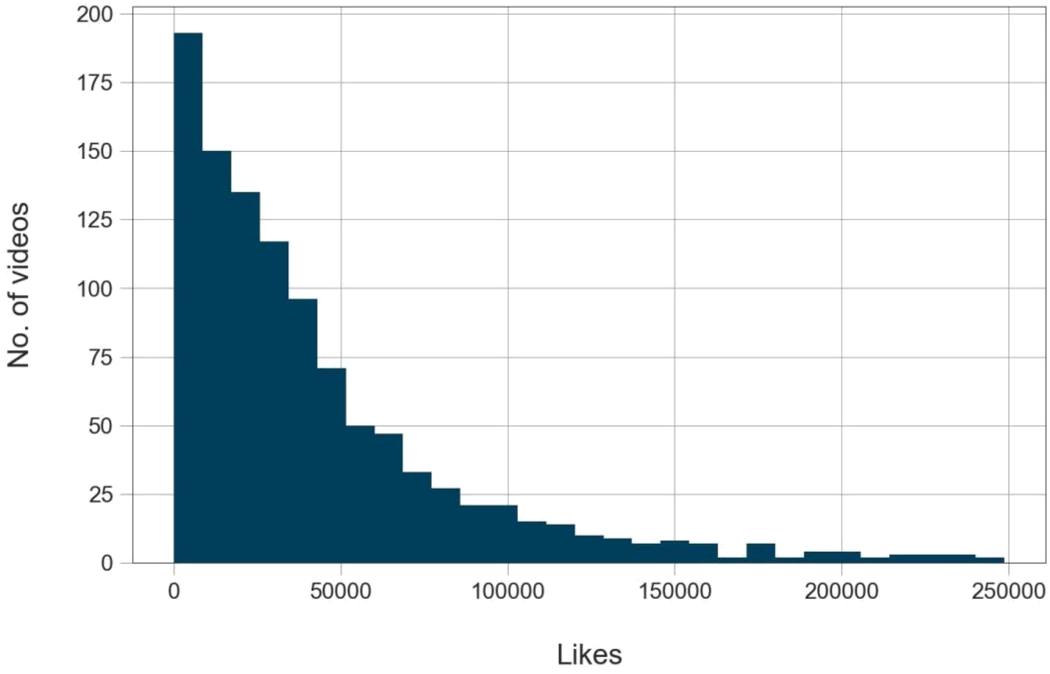
After views likes column

[13]:
```
plt.rc('figure.subplot ', wspace=0.9)
fig, ax = plt.subplots()
_ = sns.distplot(df["likes"], kde=False,
                 color=PLOT_COLORS[4], hist_kws={'alpha': 1},
                 bins=np.linspace(0, 2c6, 61), ax=ax)
  = ax.set(xlabel="Likes", ylabel="No. of videos")
  = plt.xticks(rotation=90)
```



We note that the vast majority of trending videos have between 0 and 250,000 likes. Let us plot the histogram just for videos with 250,000 likes or less to get a closer look at the distribution of the data

[14]:
```
fig, ax = plt.subplots()
_ = sns.distplot(df[df["likes"] <= 25e4]["likes"], kde=False, color=PLOT_COLORS[4],
                 hist_kws={'alpha': 1}, ax=ax)
_ = ax.set(xlabel="Likes", ylabel="No. of videos")
```



Now we can see that the majority of videos have 100,000 likes or less.
Let’s see the exact percentage of videos with less than 50,000 likes

```
[15]: df[df['likes'] < 5e4]['likes'].count() / df['likes'].count() * 100
```

```
[15]: 67.29055258467022
```

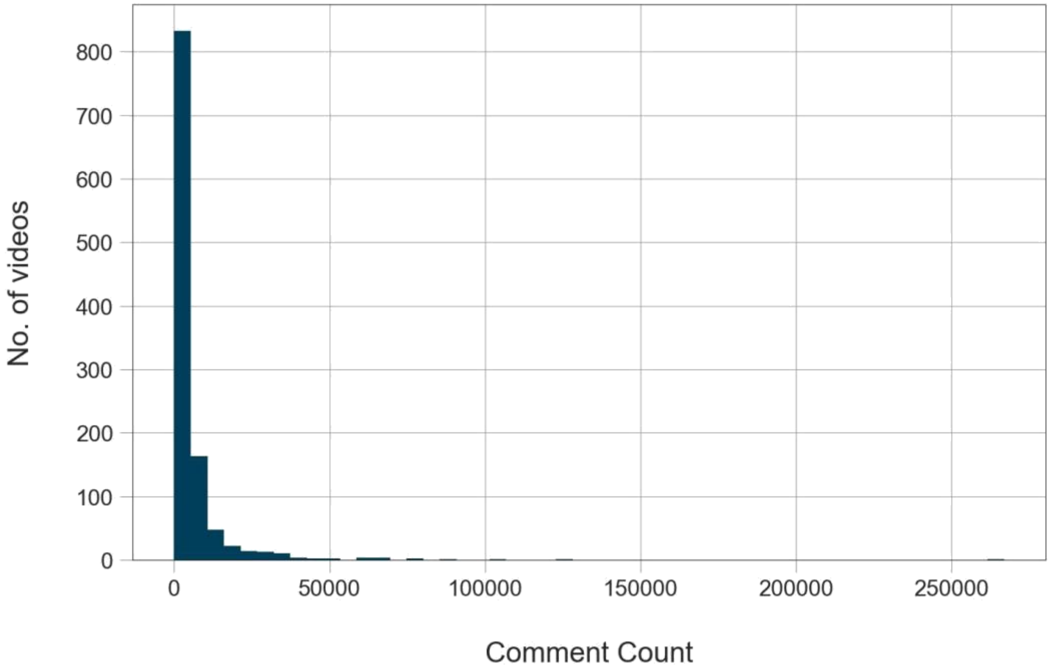
Similarly, we can see that the percentage of videos with less than 100,000 likes is around 84%

```
[16]: df[df['likes'] < 1e5]['likes'].count() / df['likes'].count() * 100
```

```
[16]: 84.93761140819964
```

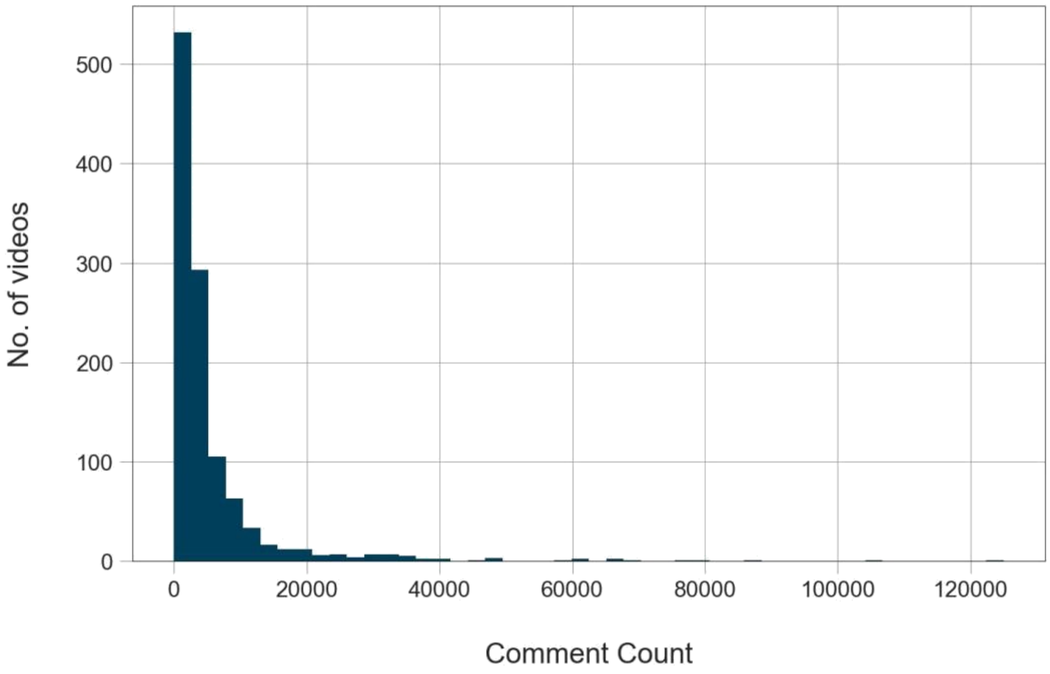
2.4.3 Comment count histogram

```
[17]: fig, ax = plt.subplots()
      _ = sns.distplot(df["comment_count"], kde=False, rug=False, color=PLOT_COLORS[4],
                       hist_kws={'alpha': 1}, ax=ax)
      _ = ax.set(xlabel="Comment Count", ylabel="No. of videos")
```



Let’s get a closer look by eliminating entries with comment count larger than 125,000 comments

```
[18]: fig, ax = plt.subplots()
_ = sns.distplot(df[df["comment_count"] < 200000]["comment_count"], kde=False, ␣
 ↪rug=False,
                        color=PLOT_COLORS[4], hist_kws={'alpha': 1}, bins=np.linspace(0, 12.5e4,
                        49), ax=ax)
_ = ax.set(xlabel="Comment Count", ylabel="No. of videos")
```



We see that most trending videos have around

As with views and likes, let’s see the exact percentage of videos with less than 4000 comments

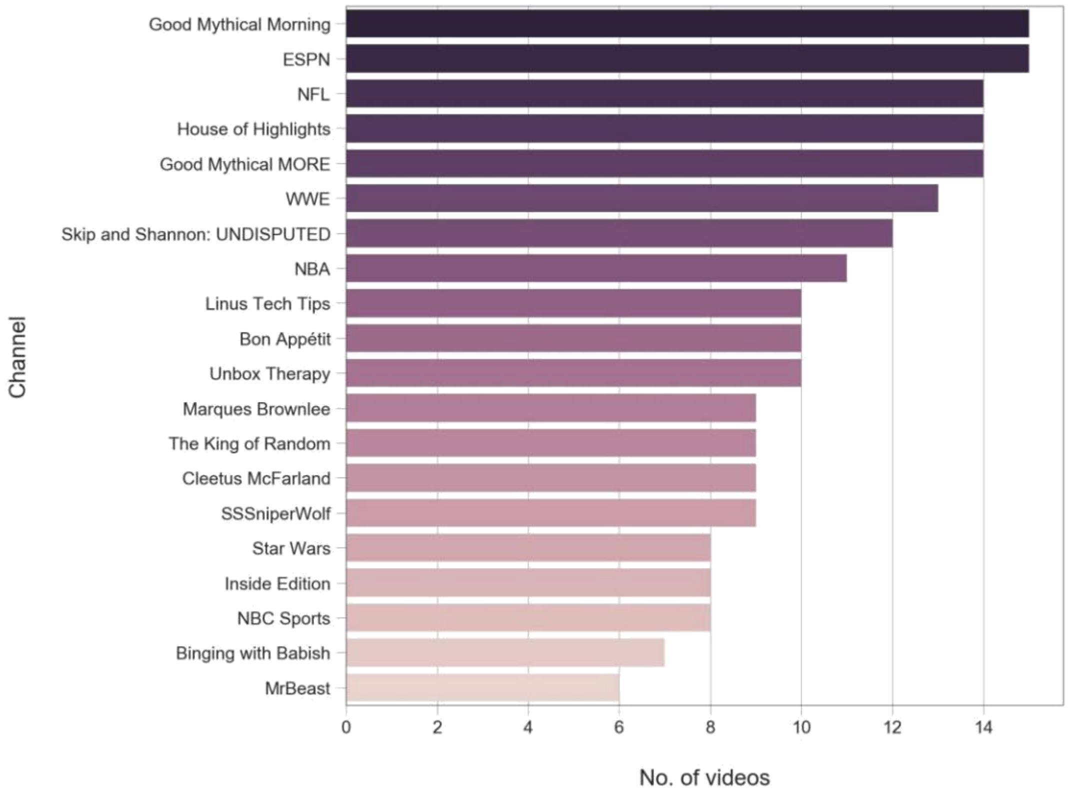
```
[19]: df[df['comment_count'] < 4000]['comment_count'].count() / df['comment_count']. ↪count() * 100
```

```
[19]: 65.59714795008912
```

In a similar way, we can see that the percentage of videos with less than 25,000 comments is around 96%.

2.5 Which channels have the largest number of trending videos?

```
[20]: cdf = df.groupby("channel_title").size().reset_index(name="video_count") \
          .sort_values("video_count", ascending=False).head(20)

      fig, ax = plt.subplots(figsize=(8,8))
      _ = sns.barplot(x="video_count", y="channel_title", data=cdf,
                      palette=sns.cubehelix_palette(n_colors=20, reverse=True), ax=ax)
      _ = ax.set(xlabel="No. of videos", ylabel="Channel")
```



2.6 Which video category has the largest number of trending videos?

First, we will add a column that contains category names based on the values in category_id column. We will use a category JSON file provided with the dataset which contains information about each category.

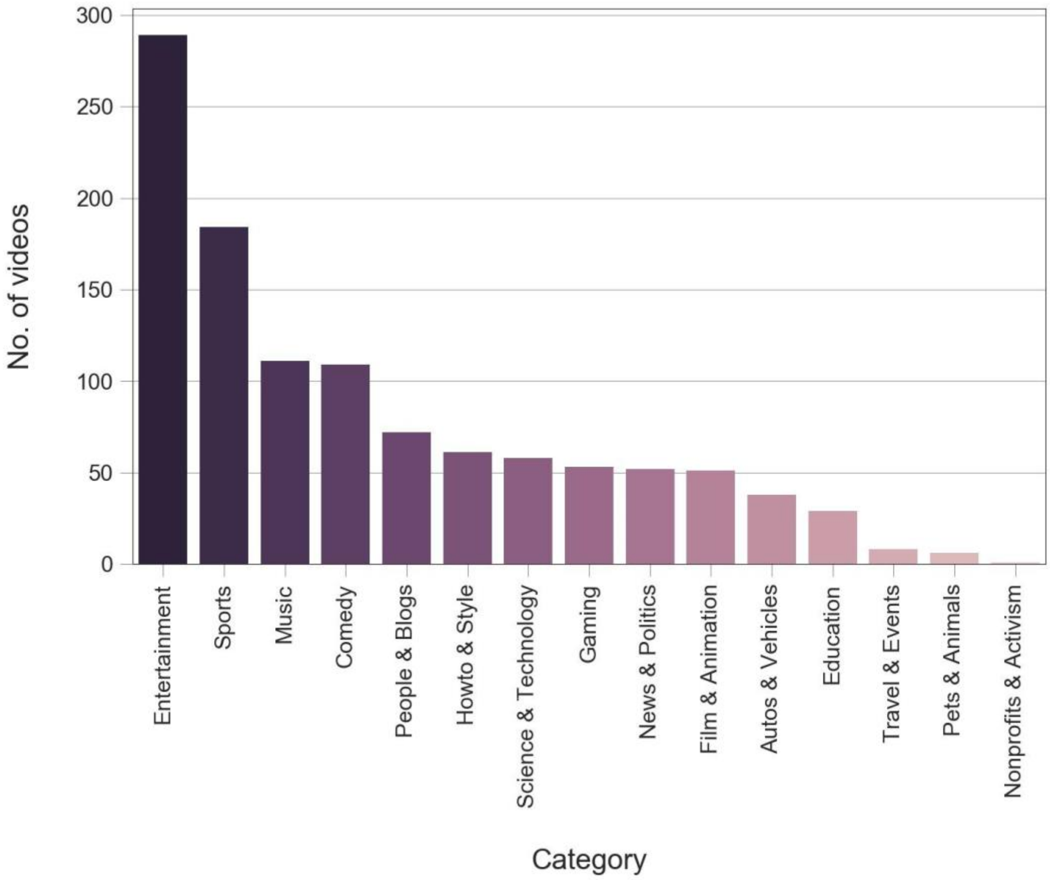
[21]:
```
with open('                              ') as f:
    catcgorics = json.load(f)["itcms"]
cat_dict = {}
for cat in categories:
    cat dict[int(cat["id"])] = cat["snippet "]["title"]
df['category name '] = df['category id'].map(cat dict)
```

Now we can see which category had the largest number of trending videos

[22]:
```
cdf = df["category_name"].value_counts().to_frame().reset_index() cdf.rename(columns={"index":
"category_name", "category_name": "No_of_videos"},␣
↪inplace=True) fig, ax =
plt.subplots()
_ = sns.barplot(x="category_name", y="No_of_videos", data=cdf,
                palette=sns.cubehelix_palette(n_colors=16, reverse=True), ax=ax)
```

```
_ = ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
  = ax.set(xlabel="Category ", ylabel="No. of videos")
```



We see that the Entertainment category contains the largest number of trending videos among other categories: around 270 videos, followed by Sports category with around 170 videos, followed by Music category with around 110 videos, and so on.

2.7 Trending videos and their publishing time

An example value of the publish_time column in our dataset is 2017-11-13T17:13:01.000Z. And according to information on this page: https://www.w3.org/TR/NOTE-datetime, this means that the date of publishing the video is 2017-11-13 and the time is 17:13:01 in Coordinated Universal Time (UTC) time zone.

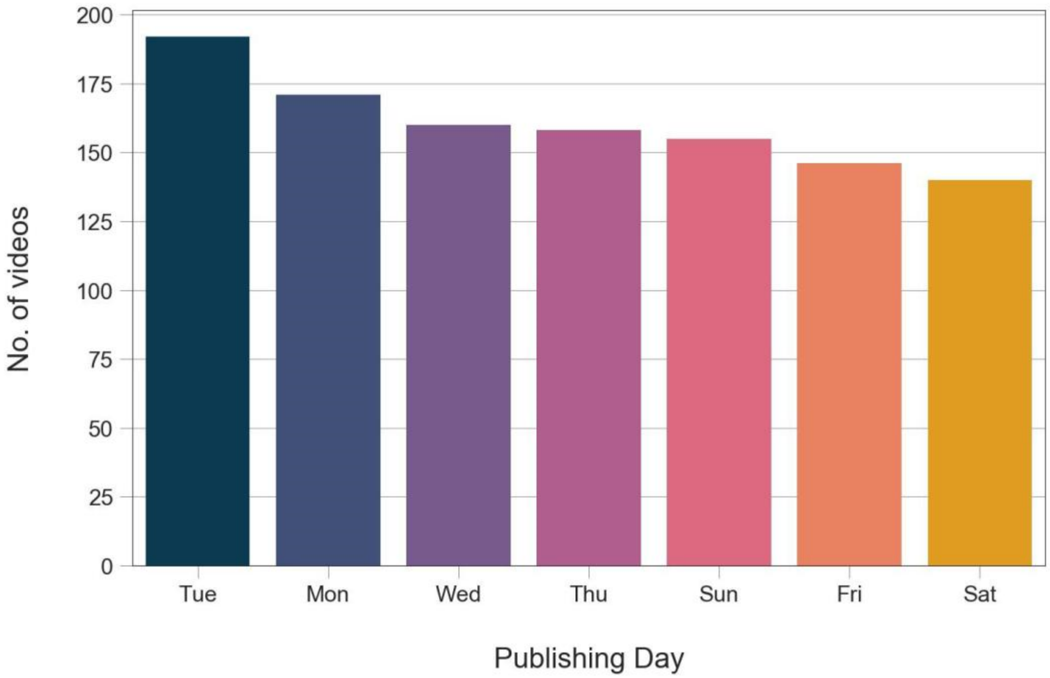
Let's add two columns to represent the date and hour of publishing each video, then delete the original publish_time column because we will not need it anymore

```
[23]: df["publishing_day"] = df["published_at"].apply( lambda x: datetime.strptime(x[:10], "%d-
          %m-%Y").date().strftime('%a'))
 df["publishing_hour"] = df["published_at"].apply(lambda x: x[11:13])
```

```
#df.drop(labels='published_at', axis=1, inplace=True)
```

videos

```
[24]: cdf = df["publishing_day"].value_counts()\
            .to frame().reset index() .rename(columns ={"index": "publishing_day",
      ↪"publishing day      "  :"No of videos      "})
      fig, ax = plt.subplots()
       = sns.barplot(x ="publishing day",   ="No of videos ", data=cdf,
                        palette=sns.color_palette([ '#003f5c ', '#374c80', '#7a5195 ',
                                                   '#bc5090', '#cf5675 ', '#ff764a',
      ↪'#ffa600   ']  n_colors =7)        ax=ax)
      _ = ax.set(xlabel="Publishing  Day", ylabel="No. of videos")
```

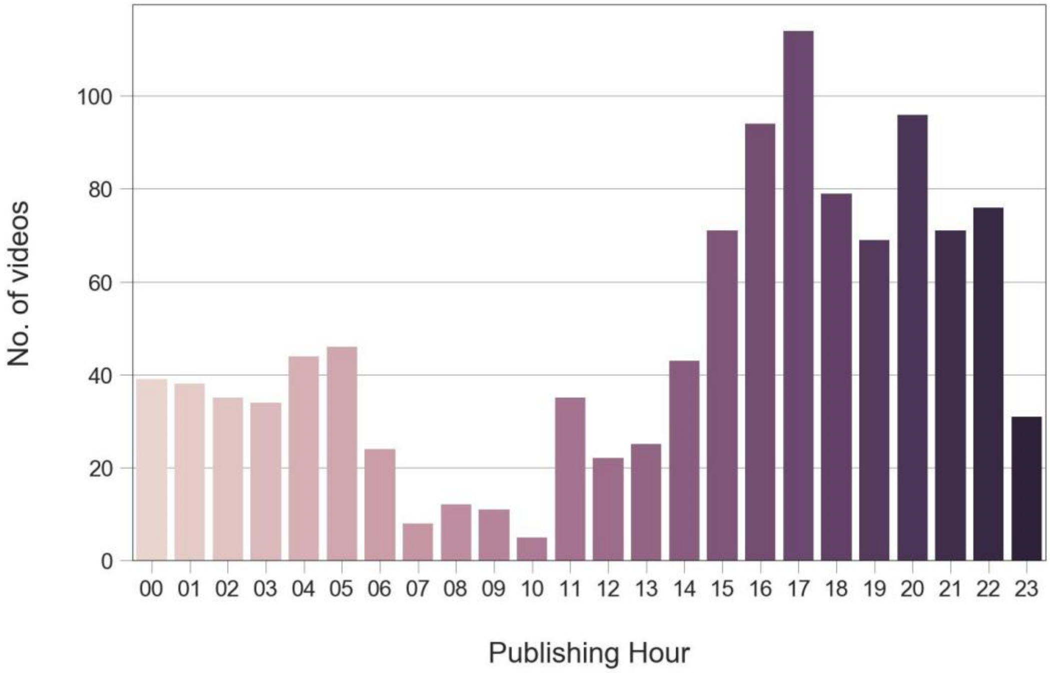


We can see that the number of trending videos published on Sunday and Saturday are noticeably less than the number of trending videos published on other days of the week.

Now let's use publishing_hour column to see which publishing hours had the largest number of trending videos

```
[25]: cdf = df["publishing_hour"].value_counts().to_frame().reset_index()\
                  .rename(columns={"index": "publishing_hour", "publishing_hour":␣
      ↪"No_of_videos"}) fig, ax =

       plt.subplots()
```

```
_ = sns.barplot(x="publishing_hour", y="No_of_videos", data=cdf,
                palette=sns.cubehelix_palette(n_colors=24), ax=ax)
_ = ax.set(xlabel="Publishing Hour", ylabel="No. of videos")
```



We can see that the period between 2PM and 7PM, peaking between 4PM and 5PM, had the largest number of trending videos. We notice also that the period between 12AM and 1PM has the smallest number of trending videos.

2.8 Data cleaning

The description(last column) ,title & channel title(non-trending data) columns have some null values. These are some of the rows whose description values are null. We can see that null values are denoted by NaN

[26]:
```
df[df["description"].apply(lambda x: pd.isna(x))].sample(3)
```

[26]:
```
             video_id                                               title \
59          4GE-7W3Kyyw IMPOSSIBLE Dinner in Iran!!! Home-Cooked Ghorm...
1004 Aa_u84fikZk Resumen y Goles | Necaxa vs Querétaro | Cuarto... 970 DIfokZSC6tM
   EXTREME FOREST HIDE & SEEK CHALLENGE!

              published_at                  channel_id                  channel_title \
59 03-11-2020 12:25 UCcAd5Np7fO8SeejB1FVKcYw Best Ever Food Review Show 1004 28-11-
2020 05:05 UCq8BPLXtFeiSFOvmJrknWGg LIGA BBVA MX
   970 26-11-2020 22:00 UCwIWAbIeu0xI0ReKWOcw3eg                        Unspeakable
```

```
      category_id          trending_date \
59                  19 05-11-2020 01:00
1004                17 29-11-2020 03:04
970                 22 28-11-2020 03:04


                                                              tags view_count likes \
59            iran|iran food|iran street food|tehran|tehran ...          647004 25823
1004 soccer|fútbol|sports|deportes|Apertura 2020 - ...                703600     3477
970 unspeakable|vlog|vlogs|unspeakablegaming|prank...                 1077882 30496

      dislikes comment_count                                          thumbnail_link \
59 413 3431 https://i.ytimg.com/vi/4GE-7W3Kyyw/default.jpg 1004 333 433
https://i.ytimg.com/vi/Aa_u84fikZk/default.jpg
970 678 9977 https://i.ytimg.com/vi/DIfokZSC6tM/default.jpg comments_disabled rating_disabled
      description category_name \
59                    False            False          NaN Travel & Events
1004                  False            False          NaN             Sports
970                   False            False          NaN People & Blogs
     publishing_day publishing_hour
59                  Sun             12
1004                Thu             05
970                 Tue             22
```

[27]:
```
df[df["title"].apply(lambda x: pd.isna(x))].head(3)
```

[27]:
```
                 video_id title            published_at                    channel_id \
140 SumDHcnCRuU          NaN 30-10-2020 11:15 UC2C_jShtL725hvbm1arSV9w
 177 euy4UaYJXzY         NaN 29-10-2020 12:24 UCHu2KNu6TtJ0p4hpSW7Yv7Q
220 QWGGtKgalDo          NaN 06-11-2020 14:10 UCFctpiB_Hnlk3ejWfHqSm6Q


                                    channel_title category_id          trending_date \
140                                    CGP Grey              27 05-11-2020 01:00
177                                        Jazza              24 05-11-2020 01:00
220 The Official Pokémon YouTube channel                       20 07-11-2020 12:30
                                                                  tags view_count likes \
140 cgpgrey|education|space|earth|venus|mars|mercu...              1379986 98654
177 josiah|brooks|jazza|jazzastudios|animation|new...               898741 34985
220 Pokemon|Pokémon|Pokémon Sword Shield|Pokémon S...               524061 27250

     dislikes comment_count                                        thumbnail_link \
  140       1024                 5997 https://i.ytimg.com/vi/SumDHcnCRuU/default.jpg
  177        738                  4479 https://i.ytimg.com/vi/euy4UaYJXzY/default.jpg
  220       2774                 4826 https://i.ytimg.com/vi/QWGGtKgalDo/default.jpg

     comments_disabled rating_disabled \
  140                False             False
```

```
177                    False                    False
220                    False                    False
                                                          description category_name \
140 Thank you, my patrons, for making this video p...     Education 177 Get my APP,
Courses, eBooks, Brushes and mor... Entertainment
220                                                         NaN             Gaming

     publishing_day publishing_hour
140              Wed                   11
177              Tue                   12
220              Wed                   14
```

So to do some sort of data cleaning, and to get rid of those null values, we put an empty string in place of each null value in the description, title & Channel Title column

[28]:
```
df["description"] = df["description"].fillna(value="") df["title"] =
 df["title"].fillna(value="") non_df["description"] =
 df["description"].fillna(value="") non_df["title"] = df["title"].fillna(value="")
 non_df["channel_title"] = non_df["channel_title"].fillna(value="")
```

Remove 1st 200 data points since they were already on the trending section when the above script ran for the 1st time and hence we cannot determine the exact time they were added to the section

[29]:
```
df = df[200:]
```

Here we remove the data points which have view count zero for trending videos

[30]:
```
df = df[df.view_count  != 0]
```

2.8.1 Labeling and Merging dataframes

Now we label the two dataframes **df** and **non_df** which contains the **trending** and **Non-Trending** video data respectively as one and zero and then we merge the dataframes

[31]:
```
df["label"] = 1
non df[ "label"] = 0
```

Now we merge both the dataframes

[32]:
```
```
df = pd.concat([df,non_df])

2.8.2 Invalid Data

In the dataset collected there are 8 data points where the date when a video was published comes after the date when the video was promoted to trending section. This is not logically possible and hence we exclude such data points.

Let us convert the **trending_date** and **published_at** strings to datetime objects to better understand and process the data

[33]:
```
df["published_at"] = pd.to_datetime(df["published_at"],format = '%d-%m-%Y %H:
  ↪%M',exact=True,infer_datetime_format=False)
```

```
df["trending_date"] = pd.to_datetime(df["trending_date"],format = '%d-%m-%Y %H:
  ↪%M',exact =True                                   =False)
df.dtypes
```

[33]:
```
category_id                int64
category_name             object
channel_id                object
channel_title             object
comment_count              int64
comments_disabled           bool
description               object
dislikes                   int64
label                      int64
likes                      int64
published_at      datetime64[ns]
publishing_day            object
publishing_hour           object
rating_disabled             bool
tags                      object
thumbnail_link            object
title                     object
trending_date     datetime64[ns]
video_id                  object
view_count dtype:          int64
object
```

[34]:
```
df_sample = df[df.trending_date < df.published_at]
df_sample
```

[34]:
```
          category_id          category_name                    channel_id \
 280                1        Film & Animation UC_IRYSp4auq7hKLvziWVH6w
294 28 Science & Technology UCXuqSBlHAE6Xw-yeJA0Tunw 295 22 People &
Blogs UCbAwSkqJ1W_Eg7wr3cp5BUA 1098 24 Entertainment
UCSAUGyc_xA8uYzaIVG6MESQ
1105               24   Entertainment UCITqR49EAUY8i1vZtXTwe-A
1106               17   Sports UCDVYQ4Zhbm3S2dlz7P1GBDg
1120                1      Film & Animation UCwTkM6CvIsYFaFiMKIKCqHw
190                26                    NaN UCSC_8gNeqj7hVDSPRzTc9_A
416                25                    NaN UC0M-_02RJqMlGTKUjF1WhJg

          channel_title comment_count comments_disabled \
280                Pixar           3072             False
294 Linus Tech Tips                1429             False
295        Safiya Nygaard          6301             False
```

```
1098           nigahiga           3078           False
1105       Dolan Twins            6015           False
1106                NFL           4937           False
1120    James Bond 007             322           False
190           CNNArabic              0           False
416                      145              False

                                                    description dislikes label \
      280 Next summer, Joe Gardner will discover his bri...          640      1

294     Get yourself a dbrand skin at https://dbrand.c...  549     1

295     So a few months ago when Cristine was in Los A...     323     1

      1098 Leave your dear ryan's in the comments for the...          113      1

 1105  Here is our Van Tour! We built a fully custom ... 487     1

1106    The San Francisco 49ers take on the Baltimore ...378     1

      1120 Bond is back. The first trailer for NO TIME TO...           38      1

   190 SUBSCRIBE TODAY SO I CAN BEAT PEWDIEPIE TO 1,0...             1      0

     416 Fabinho, Mohamed Salah and Sadio Mane all foun...          224      0

       likes              published_at publishing_day publishing_hour \

     280 44179 2020-11-07 13:53:00              Thu              13

294      16784 2020-11-07 19:41:00     Thu     19

295      54149 2020-11-07 21:00:00     Thu     21

    1098 34443 2020-12-01 21:16:00              Sun              21

1105     97188 2020-12-01 21:03:00     Sun     21

1106     10275 2020-12-01 21:14:00     Sun     21

     1120 2134 2020-12-02 01:20:00              Mon              01

190            7 2020-12-01 20:15:00            NaN             NaN

     416 11801 2020-12-03 01:32:00              NaN             NaN

      rating_disabled                                           tags \

280             False          Pixar|Disney|Disney Pixar|Pixar Movie|Animation
294            False Apple|Airpods pro|review|Airpods|Pro|Apple Air...
295            False mixing custom nail polish colors|making custom...
1098                   False ryan|higa|higatv|nigahiga|epic mime fight|dear...
1105           False Dolan|Twins|Van tour|livable van|custom van|li...
1106           False NFL|Football|offense|defense|American Football...
```

```
1120                    False James Bond|Daniel Craig|No Time To Die|Bond25|...
190                 False CNN|CNNArabic|cnnarabia|        ...
416             False                                             [none]

                                             thumbnail_link \
280 https://i.ytimg.com/vi/4TojlZYqPUo/default.jpg
294     https://i.ytimg.com/vi/XziVC8YUE5M/default.jpg
295     https://i.ytimg.com/vi/UoSSCUMk-7I/default.jpg
1098 https://i.ytimg.com/vi/CRTQUacD1GA/default.jpg
1105 https://i.ytimg.com/vi/zGwrBIscb24/default.jpg
1106 https://i.ytimg.com/vi/j-ryInG6ErA/default.jpg
1120 https://i.ytimg.com/vi/QMrGxC60vzk/default.jpg
190 https://i.ytimg.com/vi/VK8OngAUX74/default.jpg
416 https://i.ytimg.com/vi/JrSzQSxyNrg/default.jpg
                                                        title           trending_date \
  280                              Soul | Official Teaser Trailer 2020-11-07 13:17:00
294     Sometimes Apple just does it better - AirPods ... 2020-11-07 18:43:00
295     Making Custom Nail Polish Colors feat. Simply ... 2020-11-07 18:43:00
  1098                             Epic Mime Fight! (Dear Ryan) 2020-12-01 20:23:00
  1105        VAN TOUR | Custom Built For Twins To Live In 2020-12-01 20:59:00
  1106        49ers vs. Ravens Week 13 Highlights | NFL 2020 2020-12-01 20:59:00
  1120                                   NO TIME TO DIE Teaser 2020-12-01 22:53:00
190 My 1,200HP Built LLY DURAMAX IS BACK And I Bou... 2020-12-01 19:53:00
416 Liverpool v. Manchester City | PREMIER LEAGUE ... 2020-12-02 21:36:00 video_id view_count
280 4TojlZYqPUo         358392
294 XziVC8YUE5M         155925
295 UoSSCUMk-7I         380800
1098 CRTQUacD1GA        257095
1105 zGwrBIscb24        674102
1106 j-ryInG6ErA        479371
1120 QMrGxC60vzk         19521
190 VK8OngAUX74            248
416 JrSzQSxyNrg          84126
```

Now we remove these data points

[35]:
```
df = df[df.trending_date > df.published_at]
```

Let us now see the time difference between time when video went on trending and the time when the video was published

[36]:
```
df["time_diff"] = (df["trending_date "] - df["published_at "])
diff_hr = df["time_diff"] / np.timedelta64(1,'h')
df = df.assign(diff_hr = diff_hr )
```

[37]:
```
df["diff_hr "].head(5)
```

[37]:
```
200     22.67
```

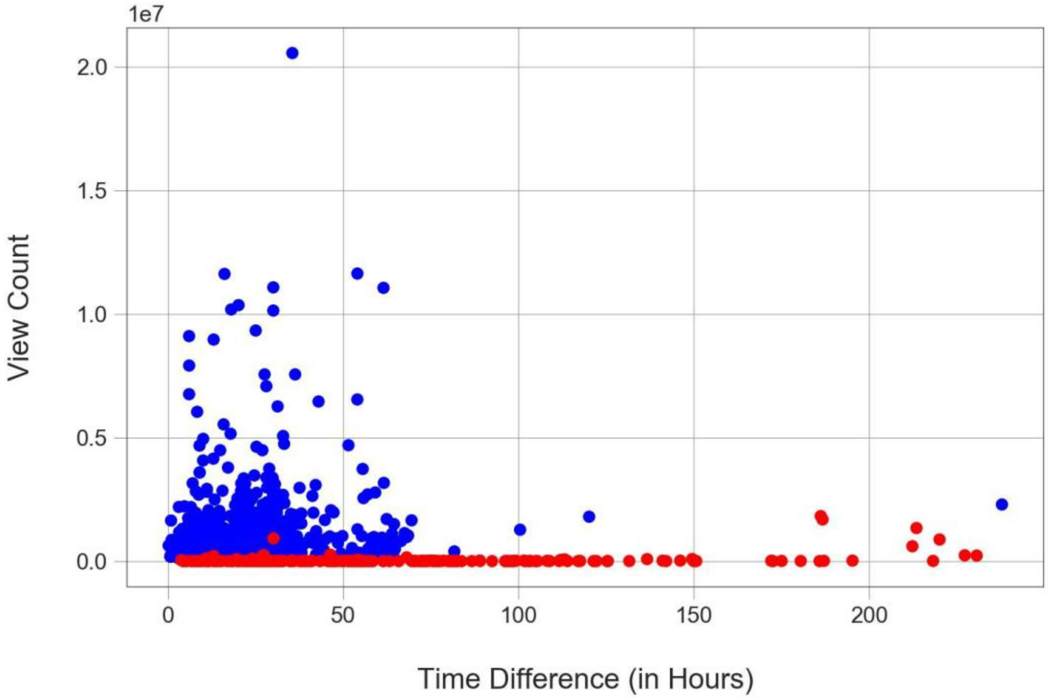
```
201      14.50
202      14.18
203      42.13
204      20.50
Name: diff_hr, dtype: float64
```

2.8.3 Binary Classification

Since our data has only two distinct classes, we will use a binary classifier on our dataset. Let us plot the Time difference vs View count plot of the merged dataframes to gain insights on how the data is distributed

[38]:
```
df[(df.label == 1) & (df.diff_hr == 0)]
df = df[df.diff_hr < 400]
```

[39]:
```
_ = matplotlib.pyplot.scatter(df["diff_hr"],df["view_count"],c=np.
  asarray(df["label"]),cmap= matplotlib colors ,
  ListedColormap(['red','blue'])) ence (in Hours)")
_ = matplotlib.pyplot.ylabel( "View Count")
```



2.8.4 Scaling the data

We need to scale the data because is has values which are high in magnitude, which may interfere with our results when we will use a classifier to classify the videos into trending and non trending.

[40]:

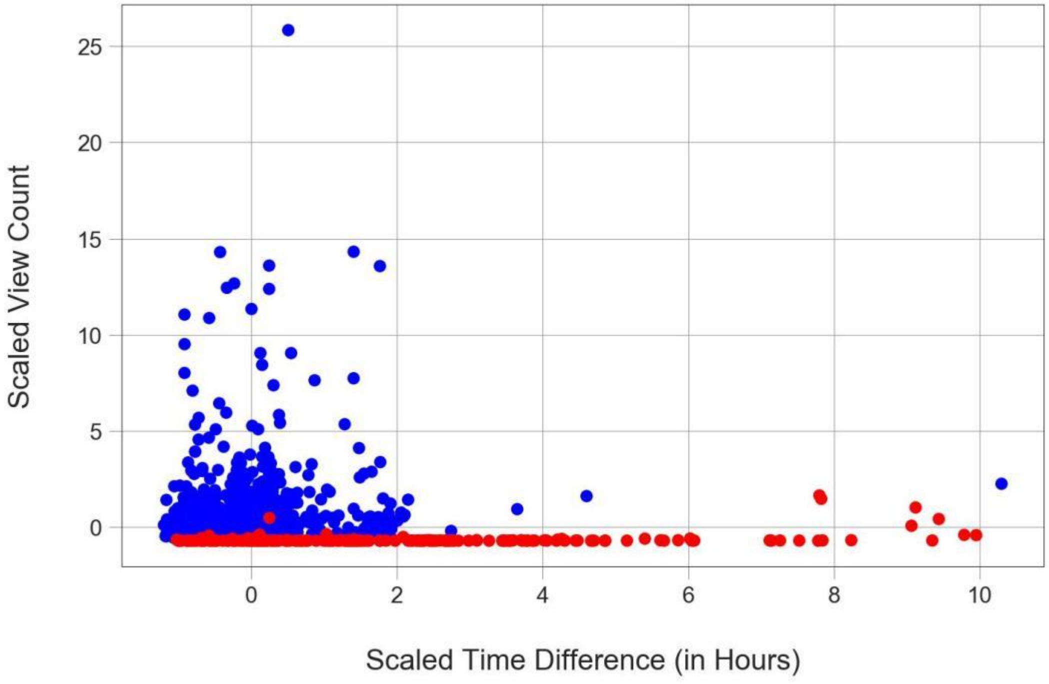
```
from sklearn.preprocessing import RobustScaler

y = df["label"]
print(X)
trans = RobustScaler().fit(X)
X_fit = trans.transform(X)
#print(X[:,0])        X = df[['view_count','likes','dislikes','diff_hr']].values
```

```
[[1.34054400e+06 4.64000000e+04 2.31900000e+03 2.26666667e+01] [8.78750000e+05
  4.00640000e+04 5.40000000e+02 1.45000000e+01]
 [3.11470000e+05 4.69400000e+03 1.33000000e+02 1.41833333e+01] ...
 [4.38000000e+02 5.00000000e+00 3.00000000e+00 1.04833333e+02]
      [2.19000000e+02 0.00000000e+00 0.00000000e+00 3.48666667e+01]
  [6.06000000e+02 1.00000000e+00 1.00000000e+00 8.01833333e+01]]
```

2.8.5 Scaled Time Difference V/s View Count

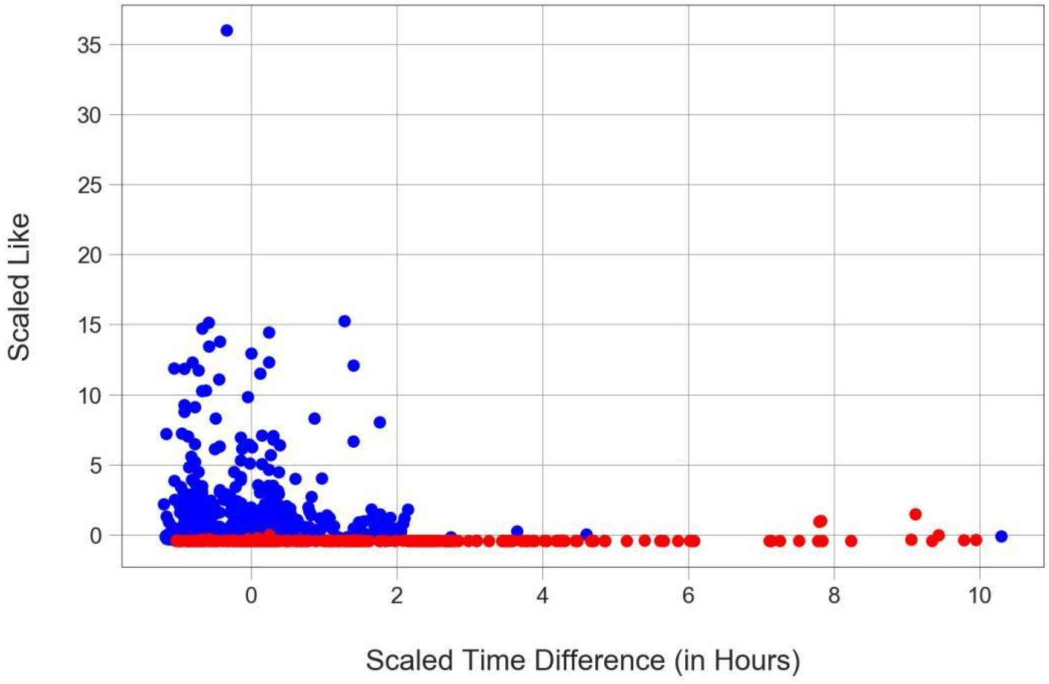
[41]:

```
  = matplotlib.pyplot.scatter(X fit[:, 3],X fit[:, 0],c=np.
↪                     =matplotlib.colors.ListedColormap(['red','blue']))
_ = matplotlib.pyplot.xlabel( "Scaled Time Difference (in Hours)")
_ = matplotlib.pyplot.ylabel( "Scaled View Count")
```



2.8.6 Scaled Time Difference V/s Like

```
[42]: _ = matplotlib.pyplot.scatter(X_fit[:,3],X_fit[:,1],c=np.
                        ↪asarray(y),cmap=matplotlib.colors.ListedColormap(['red','blue']))
      _ = matplotlib.pyplot.xlabel("Scaled Time Difference (in Hours)")
      _ = matplotlib.pyplot.ylabel("Scaled Like")
```



2.9 Ratio of View count and Time Difference (in Hours)

```
[43]: df temp = df["view count "]
```

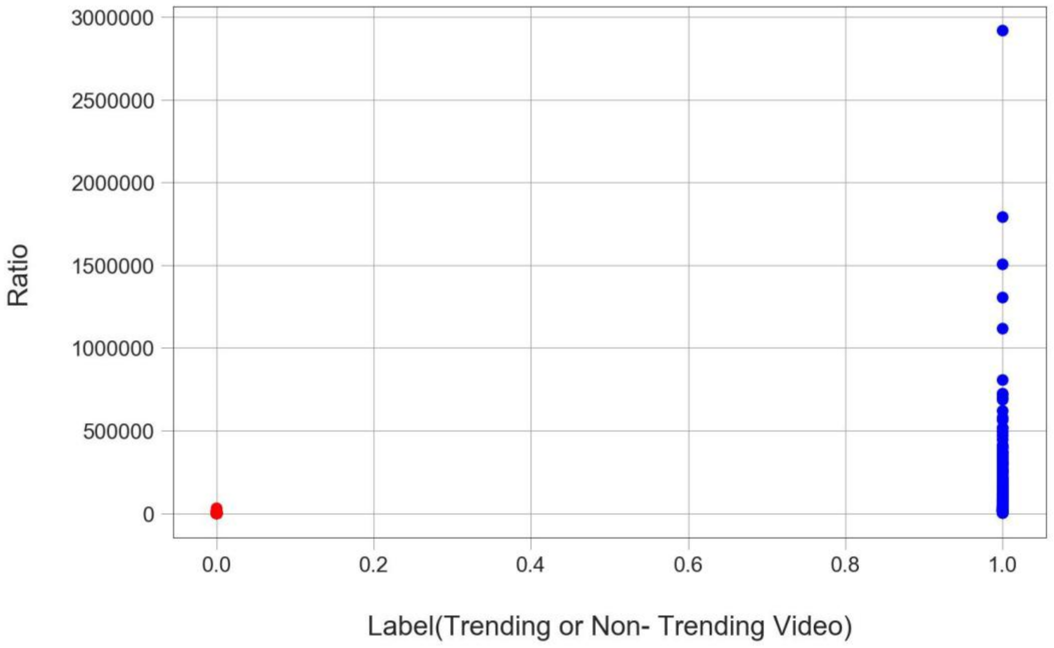
```
[44]: df temp .head(5)
```

```
[44]: 200      1340544
      201       878750
      202       311470
      203      3080069
      204      1435870
      Name: view_count, dtype: int64
```

```
[45]: df_temp = df_temp.astype("float64")
```

```
[46]: df["ratio"] = df["view_count"] / df["diff_hr"]
```

```
[47]: _ = matplotlib.pyplot.scatter(y,df["ratio"],c=np.asarray(y),cmap=matplotlib.
       ↪colors.ListedColormap(['red','blue']))

      _ = matplotlib.pyplot.ylabel("Ratio")
      _ = matplotlib.pyplot.xlabel("Label(Trending or Non- Trending Video)")
```



As we can see that the ratio has a higher magnitude for trending videos as they have high view count and low time difference between trending time and publish time.

```
[48]: # Before imputing the data
```

```
[49]: df[df.label == 0 ]["ratio"].describe()
```

```
[49]: count      215.00
      mean       695.04
      std       2853.77
      min          0.35
      25%         13.79
      50%         31.80
      75%        101.71
      max      30491.56
      Name: ratio, dtype: float64
```

```
[50]: df[df.label == 1 ]["ratio"].describe()
```

```
[50]: count       913.00
      mean      72516.31
      std      160203.48
```

```
min            2635.00
25%           18206.66
50%           33645.83
75%           73102.90
max         2917583.08
Name: ratio, dtype: float64
```

Now let us split the dataset into training and testset.

[51]:
```
from sklearn.model_selection import train_test_split from sklearn.metrics import
classification_report , accuracy_score , ␣ ↪confusion_matrix
```

[52]:
```
from imblearn.over_sampling import SMOTE
#resampling need to be done on training dataset only
X_train_res, y_train_res = SMOTE().fit_sample(X_fit, y)
X_train, X_test, y_train, y_test = train_test_split( X_train_res, y_train_res, ␣ ↪test_size=0.3,
  shuffle=True, stratify=y_train_res)
```

2.9.1 SGD Classifier

[53]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
#model
model = SGDClassifier()
#parameters
#params = {'loss ':  "deviance" , "exponential"],
#            'learning_rate ':[0.001, 0.0001, 0.00001]}
params = {'loss':  "hinge", "log", "perceptron"],
          'alpha':[0.001 , 0.0001 , 0.00001 ]}

#carrying out grid search
clf =                       params)
clf.fit(X train, y train)
#the selected parameters by grid search
print(clf.best_estimator_)


clf = clf.best_estimator_
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
```

```
SGDClassifier(alpha=1e-05, average=False, class_weight=None, early_stopping=False, epsilon=0.1,
              eta0=0.0, fit_intercept=True, l1_ratio=0.15, learning_rate='optimal', loss='log',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2', power_t=0.5,
              random_state=None, shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0,
              warm_start=False)
```

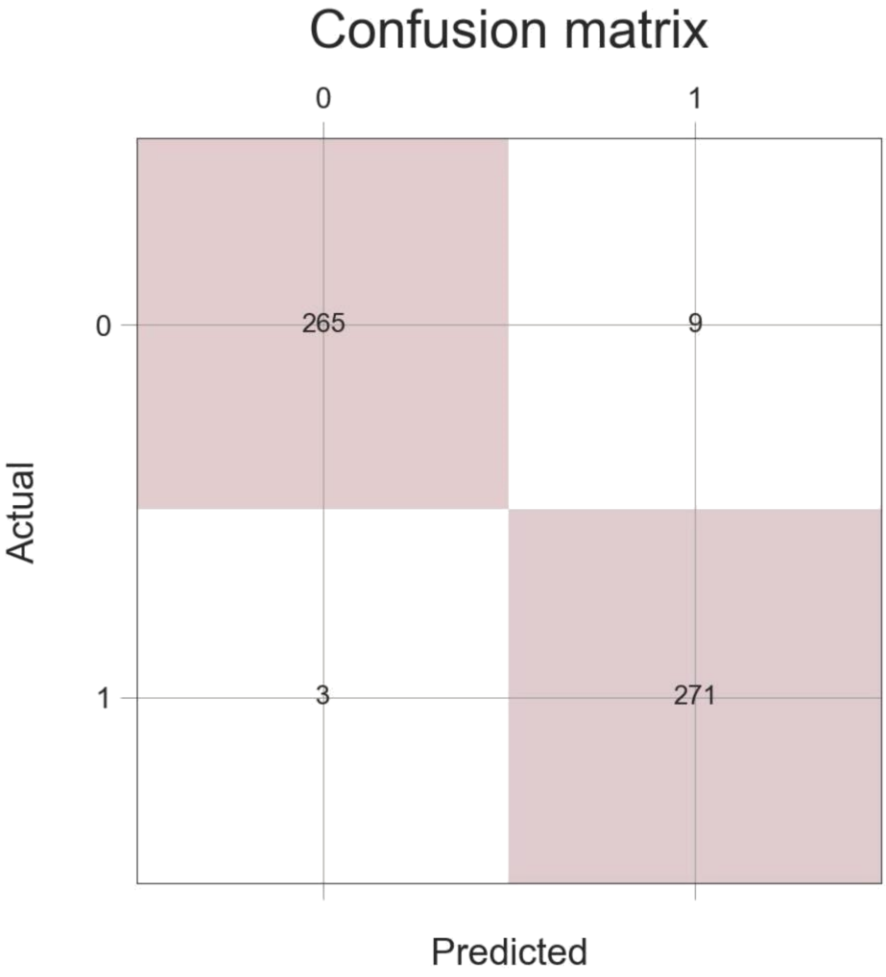
```
[54]: print(                                    pred))
      print(                            pred))
```

```
                   precision         recall f1-score      support
                0        0.99         0.97       0.98          274
                1        0.97         0.99       0.98          274

         accuracy                                0.98          548
        macro avg        0.98         0.98       0.98          548
     weighted avg        0.98         0.98       0.98          548
0.9781021897810219
```

```
     Model Accuracy is 97.81%
```

```
[56]: plt.figure(figsize=(16, 8)) plt.matshow(conf_mat,
      cmap=plt.cm.Reds, alpha=0.2) for i in range(2):
          for j in range(2):
                                  plt.text(x=j, y=i, s=conf_mat[i, j], ha="center", va="center")
      plt.title("Confusion matrix", y=1.1, fontdict={"fontsize": 20}) plt.xlabel("Predicted",
      fontdict={"fontsize": 14}) plt.ylabel("Actual", fontdict={"fontsize": 14});
```

```
<Figure size 1600x800 with 0 Axes>
```



3 Aspect Based Sentiment Analysis of YouTube comments

ABSA is really interesting since it gives a deep view of the variance of sentiments within a large corpus of text.

Most sentiment systems run sentiment analysis on the entire text, which sort of ‘averages out’ the sentiment. Running aspect based sentiment analysis on things like product reviews or YouTube comments can help the creators understand the specific parts that the users liked.

[57]:
```
%matplotlib inline
from textblob import TextBlob, Word
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color codes =True)
```

[58]:

```
import csv
with open(r'data.csv ', encoding="utf8") as f:
  reader = csv.reader(f)
  result = list(reader)

result = result[ 0]
rcsult[: 10]
```

[58]:

```
['Is any resource that shows clearer the conversion and the format of the input reviews?',
  'I subscribed because I like the content, but i think these videos are more for people who have an
intermediate level understanding of python. I can follow the conceptual bits, but as soon as you start
programming it with python, you explain your steps but only on a very high level.\nCould you
recommend a place where I can learn programming ML by building on the fundamental
concepts?\n\nbtw nice rap ;p',
  'What text editor are you using, Siraj?',
, "lol, as an german i never used waldeinsamkeit :P i'v only used einsamkeit, without wald",
  "Just curious, how long did it take y'all to train the neural net that siraj wrote. I ran mine on a gtx 1080
Ti and it took 26 seconds per epoch.",
  "Hi Siraj, \n\nI think your videos are great but I spotted a small mistake in this one. In the last part you
call validation set like this:\n'validation_set(textX, textY)' \n\nUnfortunately this doesn't work. I did
some research and learned that you may need to call it like this:\n\n'validation_set=(textX,testY)'.
\n\nNot sure if this a versioning problem. \n\nThe other thing is... you go so fast but you don't actually
show how to run this or how to actually use it. You quickly move on the AWS but there's no
explanation of how I'm supposed to use. How do I supply new text to it? How can I get a prediction
out? \n\nI'd really appreciate some help with this because I'm trying to apply this to real world problems
but I can't get an example running at the moment. :( \n\nThank you for all of the information you have
provided though. \n\nKiran",
  'Hi. Love your videos and humor. Curiously, once training is complete, how do you feed a movie
review into the model or access the model? (i.e. Where is model saved at the completion of training?)',
  'Awesome videso. with a lot of replaying, and background research im almost following... However,
whats in the imdb database? what are the labels? single words? what o they spell out? \n\nand the
storage format of the descriptions... is it a matrix, with each location having an index to a word? where
are the words? \n\nive read somewhere something about only frequency of words being stored? there
are a lot of unknowns for an AI pleb such as myself...',
                    'Hey Siraj, I am getting an error "list index out of range" for the last
statement. I tried your code as well just in case I made a typo but I am still getting the same error.
\nThanks for all the videos.']
```

[59]:

```
from string import punctuation
import re
import nltk
nltk.download('punkt')

def clean sentence(sentence):
    sentence = re.sub(r"(?:\@|https?\://)\S+|\n+", "", sentence.lower())
    # Fix spelling errors in comments!
    sent = TextBlob(sentence)
    sent.correct()
    clean = ""
    for sentence in sent.sentences:
        words = sentence.words
        # Remove punctuations
        words = [''.join(c for c in s if c not in punctuation) for s in words]
        words = [s for s in words if s]
        clean += " ".join(words)
        clean += ". "
    return clean

result = [clean sentence(x) for x in result]
result[:10]
```

```
[nltk_data] Downloading package punkt to C:\Users\Mohit [nltk_data]
    \AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
```

[59]:

```
['is any resource that shows clearer the conversion and the format of the input reviews. ',
 'i subscribed because i like the content but i think these videos are more for people who have an intermediate level understanding of python. i can follow the conceptual bits but as soon as you start programming it with python you explain your steps but only on a very high levelcould you recommend a place where i can learn programming ml by building on the fundamental concepts btw nice rap p. ',
 'what text editor are you using siraj. ',
                  'python version361 anaconda440 code for this version of this tutorial. ',
 'lol as an german i never used waldeinsamkeit p i v only used einsamkeit without wald. ',
 'just curious how long did it take yall to train the neural net that siraj wrote. i ran mine on a gtx 1080 ti and it took 26 seconds per epoch. ',
 'hi siraj i think your videos are great but i spotted a small mistake in this one. in the last part you call validation set like this validationset textx texty unfortunately this does nt work. i did some research and learned that you may need to call it like this validationset textx testy. not sure if this a versioning problem. the other thing is you go so fast but you do nt actually show how to run this or how to actually use it. you quickly move on the aws but there s no explanation of how i m supposed to use. how do i supply new text to it. how can i get a prediction out. i d really appreciate some help with this because i m trying to apply this to real world problems but i ca nt get an example running at the moment. thank you for all of the information you have provided though. kiran. ',
  'hi. love your videos and humor. curiously once training is complete how do you feed a movie review
into the model or access the model. ie. where is model saved at the completion of training. ',
  'awesome videso. with a lot of replaying and background research im almost following however whats
in the imdb database. what are the labels. single words. what o they spell out. and the storage format of
the descriptions is it a matrix with each location having an index to a word. where are the words. ive
read somewhere something about only frequency of words being stored. there are a lot of unknowns for
an ai pleb such as myself. ',
  'hey siraj i am getting an error list index out of range for the last statement. i tried your code as well
just in case i made a typo but i am still getting the same error. thanks for all the videos. ']
```

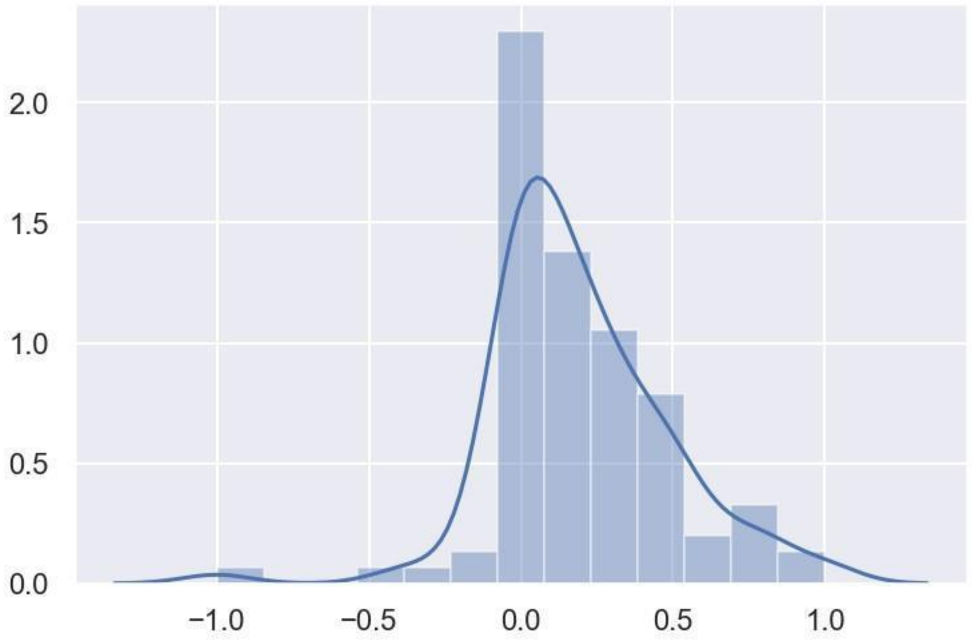
[60]:
```
sentiment scores = list()
i = 0
for sentence in result:
    line = TextBlob(sentence)
    sentiment scores.append(line.sentiment.polarity)
    if(i <= 10):
        print(sentence + ": POLARITY=" + str(line.sentiment.polarity))
        i += 1
```

```
is any resource that shows clearer the conversion and the format of the input reviews. : POLARITY=0.0
i subscribed because i like the content but i think these videos are more for people who have an
intermediate level understanding of python. i can follow the conceptual bits but as soon as you start
programming it with python you explain your steps but only on a very high levelcould you recommend
a place where i can learn programming ml by building on the fundamental concepts btw nice rap p. :
POLARITY=0.32699999999999996 what text editor are you using siraj. : POLARITY=0.0 python
version361 anaconda440 code for this version of this tutorial. : POLARITY=0.0 lol as an german i
never used waldeinsamkeit p i v only used einsamkeit without wald. :
POLARITY=0.26666666666666666 just curious how long did it take yall to train the neural net that
siraj wrote. i ran mine on a gtx 1080 ti and it took 26 seconds per epoch. : POLARITY=-
0.05000000000000001 hi siraj i think your videos are great but i spotted a small mistake in this one. in
the last part you call validation set like this validationset textx texty unfortunately this does nt work. i
did some research and learned that you may need to call it like this validationset textx testy. not sure if
this a versioning problem. the other thing is you go so fast but you do nt actually show how to run this
or how to actually use it. you quickly move on the aws but there s no explanation of how i m supposed
to use. how do i supply new text to it. how can i get a prediction out. i d really appreciate some help
with this because i m trying to apply this to real world problems but i ca nt get an example running at
the moment. thank you for all of the information you have provided though. kiran. :
POLARITY=0.057284382284382276 hi. love your videos and humor. curiously once training is
complete how do you feed a movie review into the model or access the model. ie. where is model saved
at the completion of training. : POLARITY=0.16666666666666666 awesome videso. with a lot of
replaying and background research im almost following however whats in the imdb database. what are
the labels. single words. what o they spell out. and the storage format of the descriptions is it a matrix
with each location having an index to a word. where are the words. ive read somewhere something
about only frequency of words being stored. there are a lot of unknowns for an ai pleb such as myself. :
POLARITY=0.18571428571428572 hey siraj i am getting an error list index out of range for the last
statement. i tried your code as well just in case i made a
```

typo but i am still getting the same error. thanks for all the videos. : POLARITY=0.06666666666666667 what is the current state of art in sentiment analysis. : POLARITY=0.0

[61]:
```
sns.distplot(sentiment_scores)
```

[61]: <matplotlib.axes._subplots.AxesSubplot at 0x274023f79c8>



We can see that a majority of the comments are marked as neutral (though slightly on the positive side). This basically implies that TextBlob SA ‘averages’ out over a sentence.

So for a sentence containing : “Love your videos and humor.” the polarity is 0.167 plainly because it is also followed by “curiously once training is complete how do you feed a movie review into the model or access the model ie where is model saved at the completion of training”.

[63]:

```
comments.noun_phrases
```

[62]: comments = TextBlob(' '.join(result))

```
[nltk_data] Downloading package brown to C:\Users\Mohit [nltk_data]
\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\brown.zip.
```

[63]:
```
WordList(['shows clearer', 'input reviews', 'intermediate level understanding',
 'conceptual bits', 'high levelcould', 'fundamental concepts btw', 'nice rap p.', 'text editor', 'python
 version361 anaconda440 code', 'german i', 'waldeinsamkeit p i v', 'neural net', 'hi siraj i', 'small mistake',
'validationset textx texty', 'nt work', 'validationset textx testy', 'i m', 'new text', 'i d', 'i m', 'real world
problems', 'i ca nt', 'movie review', 'awesome videso', 'background research im', 'imdb database', 'storage
format', 'ai pleb', 'hey siraj i', 'error list index', 'case i', 'current state', 'sentiment analysis', 'size thats', 'i
download code', 'nnet architecture', 'wow i', 'catchy informative', 'just awesome course i', 'absolute
popularity', 'integers source', 'name validationset', 'neural net', 'btw thanks', 'great videos', 'image
recognition',
 '                ', '
', '
', 'i m', 'siraj btw i m', 'accuracy score', 'imdb dataset', 'great channel', 'glad i', 'great videos', 'new
text', 'hehe testy', 'great man', 'google thanks', 'cool stuff', 'buddhist vipassana tradition', 'i ve',
'dissolution nyana', 'bhanga nyana', 'dark night', 'christian traditionare',
'meditation siraj', 'hi siraj', 'great videos', 'sentiment analysisdo',
'software materialsdata', 'i dont', 'code trains', 'inthe code', 'new phrases',
'new data inputs', 'tflearn tutorial', 'step i', 'deep learning videos',
'conceptual side', 'quick cuts', 'import pieces', 'inexplicable reason', 'jurassic park scene', 'great teaching
methods', 'modelload modeltfl newsentence', 'testdata padsequences newsentence maxlen100 value0',
'prob modelpredict testdata print prob', 'nice video', 'hey siraj', 'new delhi india', 'computer science', 'i
plan', 'college project', 'sentiment analysis i', 'naive bayes', 'deep learning tensorflow', 'main area',
'specific reviews', 'great help', 'final year project', 'sentiment analysis', 'pro series', 'machine learning',
'videos i', 'previous video', 'textblob polarity thing', 'right predictions anyways i', 'default textblob', 'such
thing', 'dependency thing', 'i m', 'i m', 'new dimension', 'machine learning thanks', 'i ca nt figure', 'hi
siraj', 'size vectors', 'lstm need use', 'input neural net word', 'example text', 'neural net', 'index
representation', 'output dimention', 'i m', 'input vector', 'i m', 'trainx vector', 'research problems',
'sentiment analysis', 'possible output values', 'negative thanks', 'accurate hahaha xd', 'hi siraj',
'great video', 'i wonder', 'raw word', 'super awesome video', 'windows plubs',
'i ve', 'side effect tflearn', 'fat beats', 'good work', 'output results',
'sorry i m', 'deep learning field', 'deep learning plzalbum name siraj', 'perception perceptrontrack', 'baby
activation functionstrack', 'level mlptrack', 'recurrent nn dlove', 'great videos man', 'lil wayne s
deposition video', 'great videos siraj', 'life video', 'great video mate', 'okay depoc', 'softmax function',
'code challenge submission', 'hey siraj look', 'submission thanks', 'challenge code', 'install tf', 'itinstall
tensorflow', 'git repo pip install git', 'dev version', 'dependency curses', 'jupyter notebook', 'normal
python script', 'mayby i', 'error time', 'file conda commands conda', 'n tftest python35activate tftestpip
install tensorflowpip install gitconda install h5py scipyit', 'runs anywayhope', 'time series data', 'hi siraj',
'twitter sentiments analysis project', 'deep learning', 'text file format', 'great video', 'mins versions',
'sentiment analysis', 'accuracies phere', 'great video', 'neural net i', 'learnt sentiment analysis', 'long time',
'hi siraj', 'great work', 'different things', 'hey siraj', 'lol man', 'natural language processing class',
'waldeinsamkeit d i m german', 'great stuff', 'import single files', 'machines i', 'layer dip', 'random rap
moments', 'def jam', 'wonderful library', 'official highlevel wrapper', 'air time', 'awesome vid', 'hello
siraj', 'jie xun s code i m', 'i m', 'different resultsall', 'rap tho', 'complex ideas', 'tensorflow example',
'great video', 'nt fancy lua', 'pytorch solves', 'dynamic computational graphs',
'waldeinsamkeitsmeditation zen', 'direct link', 'i m rappin siraj', 'i m', 'sayi ll', 'special way', 'learning
stuffnice', 'deep learning'])
```

4 Pruning

Quite a lot of these noun phrases are repeated or have the same subset of words. We now run modified versions of redundancy pruning and compactness pruning. Compactness pruning:

We check for compact phrases and see if the words in the phrases make sense. For e.g the phrase “i m” fails the compact pruning test and is pruned. A simple way to carry out compact pruning is by checking the words in a phrase and seeing if a dictionary meaning exists. If the number of words in the phrase without dictionary meanings cross a certain threshold, we prune the phrase.

[65]:
```
import nltk
nltk.download('wordnet')

cleaned = list()
for phrase in comments.noun_phrases:
    count = 0
    for word in phrase.split():
        # Count the number of small words and words without an English␣
↪definition
        if len(word) <= 2 or (not Word(word).definitions):
            count += 1
    # Only if the 'nonsensical' or short words DO NOT make up more than 40%␣
↪(arbitrary) of the phrase add
    # it to the cleaned list, effectively pruning the ones not added.
    if count < len(phrase.split())*0.4:
        cleaned.append(phrase)

print("After compactness pruning:\nFeature Size:" + str(len(cleaned)))
#len(cleaned)
```

```
[nltk_data] Downloading package wordnet to C:\Users\Mohit [nltk_data]
    \AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\wordnet.zip.
```

```
After compactness pruning:
Feature Size:125
```

5 Redundancy pruning:

I am using a naive decision of choosing the largest common noun phrase as a non-redundant feature. A better way would be to find ‘important’ terms in common noun phrases and choose those. One approach to that could be something called TF-IDF.

[66]:
```
for phrase in cleaned:
    match = list()
    temp = list()
    word_match = list()
    for word in phrase.split():
        # Find common words among all phrases
        word_match = [p for p in cleaned if re.search(word, p) and p not in
 ↪word_match]
        # If the size of matched phrases set is smaller than 30% of the cleaned
 ↪phrases,
        # then consider the phrase as non-redundant.
        if len(word_match) <= len(cleaned)*0.3 :
            temp.append(word)
            match += word_match

    phrase = ' '.join(temp)
#     print("Match for " + phrase + ": " + str(match))

    if len(match) >= len(cleaned)*0.1 :
        # Redundant feature set, since it contains more than 10% of the number
 ↪of phrases.
        # Prune all matched features.
        for feature in match:
            if feature in cleaned:
                cleaned.remove(feature)

        # Add largest length phrase as feature
        cleaned.append(max(match, key=len))


print("After redundancy pruning:\nFeature Size:" + str(len(cleaned)))
#print("Cleaned features:")
#cleaned
```

```
After redundancy pruning:
```

Feature Size:78

[67]:

```
import nltk
nltk.download( 'stopwords ')

from nltk.corpus import stopwords
feature_count = dict()
for phrase in cleaned:
    count = 0
    for word in phrase.split():
        if word not in stopwords .words( 'english '):
            count += comments.words.count(word)

    #print(phrase + ": " + str(count))
                            = count
```

[nltk_data] Downloading package stopwords to C:\Users\Mohit [nltk_data]
    \AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\stopwords.zip.

[68]:

```
counts = list(fcature_count .valucs())
features = list(feature count .keys())
threshold = len(comments.noun_phrases)/100

print("Threshold: " + str(threshold ))

frequent_features = list()

for fcature, count in fcature count .itcms():
    if count >= threshold:
        frequent features .append(feature)

#print( 'Frequent Features:')
```

Threshold:2.25

[ ]:

```
#sns.set()

#f, ax =                              50))
                              , x=counts, color="c", ax=ax)
                       threshold], [0, len(features)], linewidth=4, color="r")
```

[70]:

```
absa list = dict()
# For each frequent feature
for f in
    # For each comment
    absa list[f] = list()
    for comment in result:
        blob =
        # For each sentence of the comment
        for sentence in blob.sentences:
            # Search for frequent feature 'f'
            q = '|'.join(f.split())
            if re.search( r'\w*(' + str(q) + ')\w*', str(sentence)):
                absa_list[f].append(sentence)
```

6 Aspect based sentiment scoring

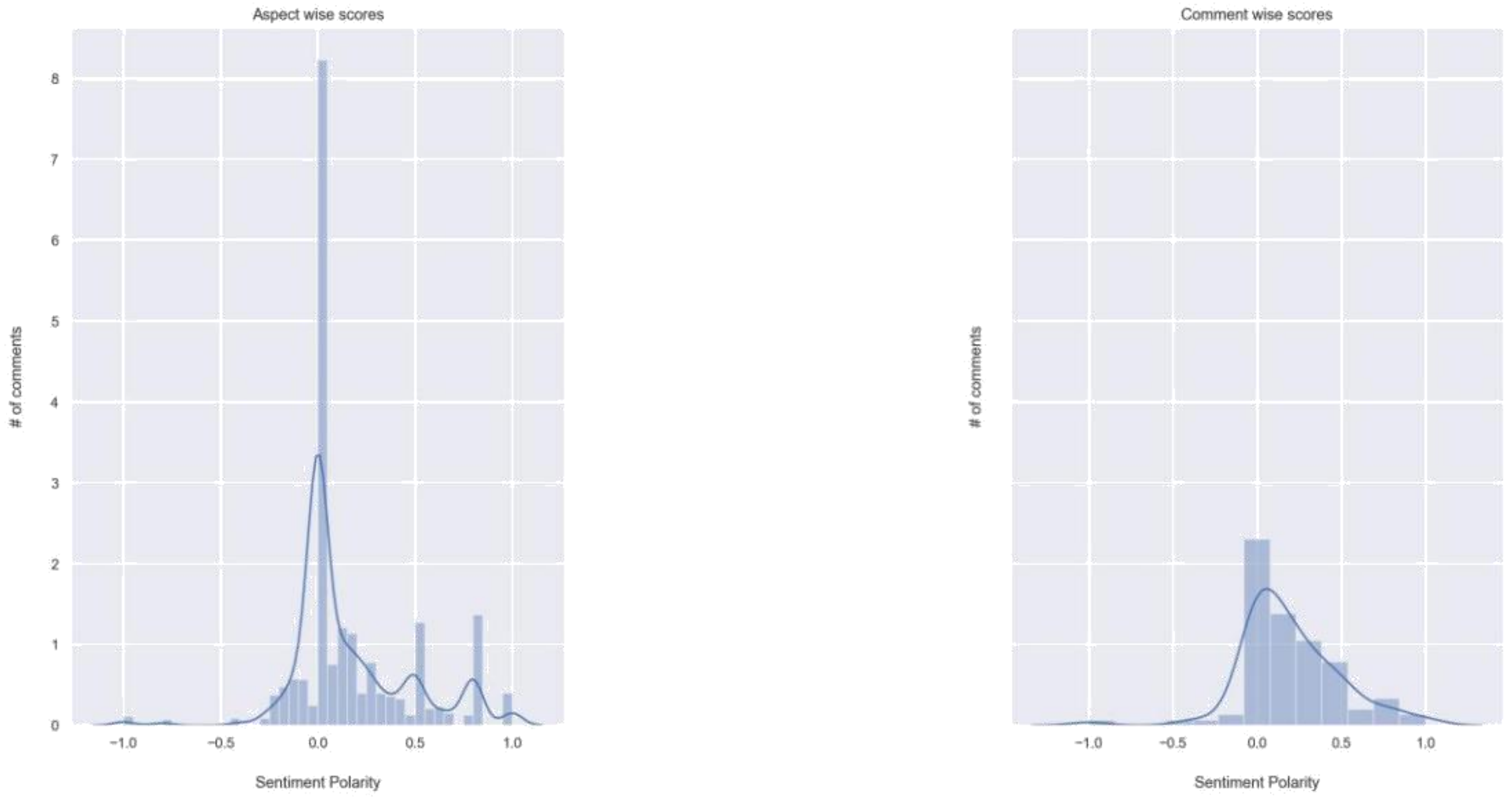
Now that we have aspect specific sentences, all we have to do is run sentiment analysis on each sentence using TextBlob’s sentiment analyzer.

[71]:

```
scores = list()
absa scores = dict()
for k, v in absa list.items():
    absa scores[k] = list()
    for sent in v:
        score = sent.sentiment.polarity
        scores.append(score)
        absa_scores[k].append(score)
```

[72]:

```
fig, (ax1, ax2) = plt.subplots(ncols =2, sharey=True, figsize=(20, 10))
plot1 = sns.distplot(scores, ax=ax1)

ax1.set_title('Aspect wise scores')
ax1.set_xlabel( 'Sentiment Polarity')
ax1.set ylabel( '# of comments')

ax2.set_title('Comment wise scores')
ax2.set_xlabel( 'Sentiment Polarity')
ax2.set_ylabel( '# of comments')

plot2 = sns.                              ax=ax2)
```



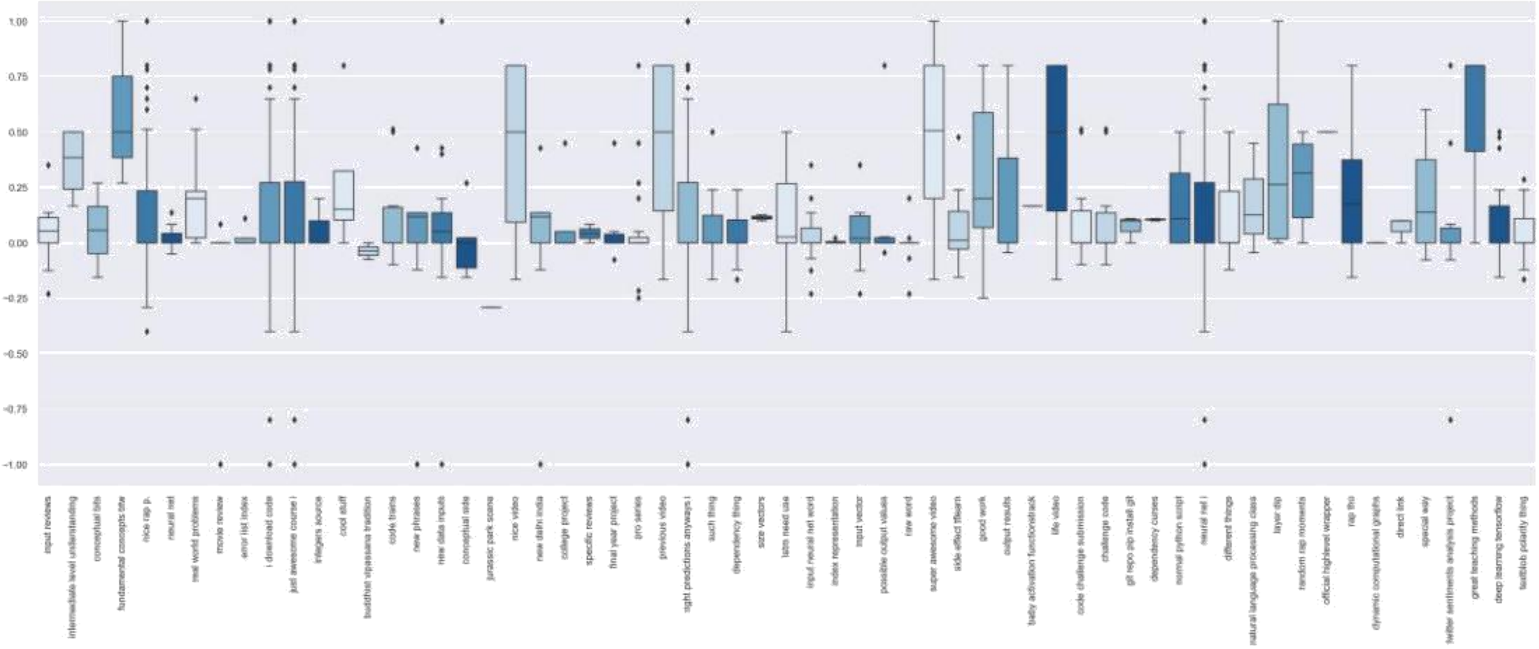
7 Graph Analysis

Notice the high amount of variance in the aspect based scores on the left. Even though a majority of the scores are neutral, there is lot of variance in the number of comments with positive sentiments. The total number of scores have also increased since one sentence of a comment may contain multiple frequent features.

[73]:
```
vals = dict()
vals["aspects "] = list()
vals["scores "] = list()
for k, v in absa scores.items():
    for score in v:
        vals["aspects "].append(k)
        vals["scores "].append(score)
```

[74]:
```
fig, ax1 = plt.subplots(figsize =(30, 10))

color = sns.color_palette( "Blues ", 6)
plt.xticks(rotation=90)
sns.set_context( "paper", font_scale=3)
sns.boxplot(x="aspects ",  ="scores ", data=vals, palette=color, ax=ax1)
```

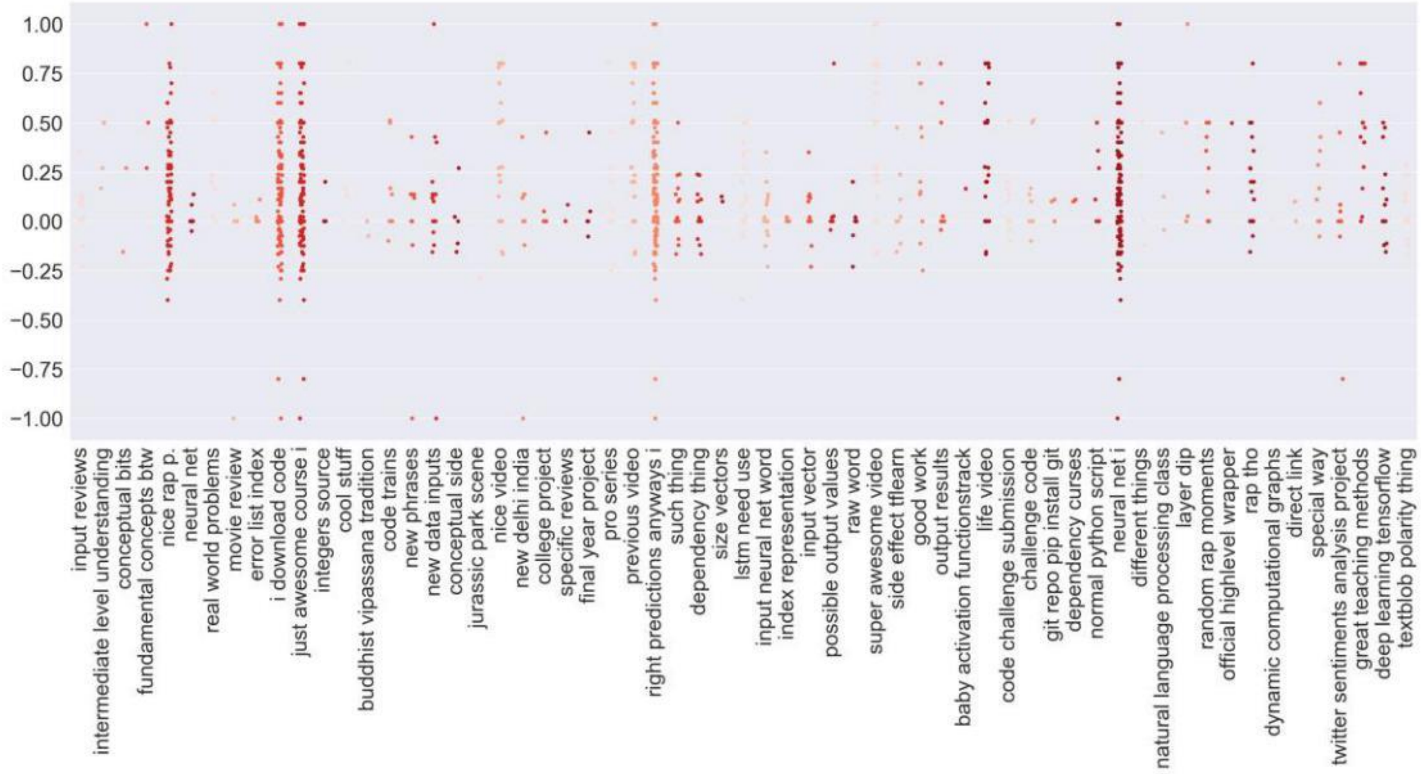
[74]: <matplotlib.axes._subplots.AxesSubplot at 0x27407ba6508>



[75]:

```
color = sns.color palette( "Reds", 6)
fig, ax1 = plt.subplots(figsize =(30, 10))
plt.xticks(rotation=90)
sns.sct context( "papcr", font scalc=2)
sns.stripplot(x="aspects ",  ="scores ",data=vals, palette=color)
```

[75]: <matplotlib.axes._subplots.AxesSubplot at 0x27408289a08>



7.1 Conclusions

Here are the some of the results we extracted from the analysis:

- We analyzed a dataset that contains information about YouTube trending videos for 20 days. The dataset was collected in 2020. It contains **1122** video entry.
- We ran Aspect Based Sentiment Analysis (ABSA) on a YouTube video and found that ABSA actually gives a more in-depth understanding of people's reviews.
- The Largest number of trending videos come under the **Entertainment** Category and **Good Mythical Morning** Channel have the largest number of trending video
- We used a Stochastic Gradient Decent Classifier to train a model which classifies the trending and non trending video based on various parameters.
- The model displayed approximately **97%** accuracy on the test set