YouTube Videos Analysis

Presentation by Group 4:
Justin Tapia
Kevin MacDonald
Snehal Desavle
Zara Khan

Outline

- Topic
- Questions We Hope to Answer
- Technologies Used
- Data Source
- Data Exploration Phase
- Data Analysis Phase
- Machine Learning
- Dashboard
- Things We Would Have Done Differently

Topic

Which YouTube video and channel metrics play the biggest role in creating a video that will gain the largest amount of views?

Why

YouTube's dense and diverse arrays of users and content providers pose ample room for analyzing and characterizing the popularity of its videos

Problem Statement

Predict a YouTube video’s success by analyzing the metrics of channels and videos by the topic category of the video.

Research Questions

What features and traits do successful videos have in common across categories?

How do meta-level features drive user engagement and popularity of a video?

Language	Dashboard	Data Storage
Python	tableau	PostgreSQL
SQL	plotly	AWS Amazon Web Services

Data Source

YouTube Data
API v3

Data Exploration Phase

Categories

- Cooking
- Fitness
- History
- Science
- News
- Music
- Comedy
- Travel

Channels

- Grabbed the top 10 channels from each category.
- Grabbed random channels from each category
- Grabbed top 20 English language channels

Videos

From Each Channel, grab the 50 most recent uploads, and recent comments from each video

Data Source (Code)

- Get list of channels for analysis:
  - [Wikipedia - List of most-subscribed YouTube Channels]
  - ["Best YouTube Channels by Category" - compiled by Clifford Chi]
  - "Random Sampling" of channels compiled by Zara Khan
- Feed channel list into YouTube Data API v3

```
# for loop to get channel details
for channel in video_list:
    response = youtube.channels().list(
        part=['snippet', 'statistics', 'topicDetails', 'contentDetails'],
        id=channel
    ).execute()
    # append response to dataframe
    df = df.append(response['items'], ignore_index=True)
```

- Get 50 most recent videos from each channel

```
response = youtube.channels().list(
    part=['contentDetails'],
    id=channel_id
).execute()

playlist_id = response['items'][0]['contentDetails']['relatedPlaylists']['uploads']

response = youtube.playlistItems().list(
    part=['contentDetails'],
    playlistId=playlist_id,
    maxResults=50
).execute()
```

- Get recent comments from each video

```
def get_comments(video_id):
    try:
        results = youtube.commentThreads().list(
            part="snippet",
            videoId=video_id,
            textFormat="plainText",
            maxResults=20
        ).execute()

        comments = []
        for item in results["items"]:
            comment = item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]
            comments.append(comment)
        return comments
    except:
        return None
```

- Get sentiment analysis for comments

```
for video in comment_df['video_id']:
    comments = get_comments(video)
    if comments is not None:
        comment_df.loc[comment_df['video_id'] == video,
        'sentiment'] = sentiment_analyzer_scores(comments)
```

Ran Channel IDs for Channel Metadata

```python
def channel_information_grabber(video_list, title):
    # create empty dataframe
    df = pd.DataFrame()

    # for loop to get channel details
    for channel in video_list:
        response = youtube.channels().list(
            part=['snippet', 'statistics', 'topicDetails', 'contentDetails'],
            id=channel
        ).execute()
        # append response to dataframe
        df = df.append(response['items'], ignore_index=True)
```

Dropped columns: title, description, thumbnails, default language, topicIds, related Playlists, and category_title

```
id                          object
title                       object
description                 object
customUrl                   object
publishedAt                 object
thumbnails.default.url      object
defaultLanguage             object
viewCount                    int64
subscriberCount              int64
videoCount                   int64
topicIds                    object
topicCategories             object
relatedPlaylists.uploads    object
category_title              object
dtype: object
```

Ran Channel IDs for Top 50 Videos Metadata

```python
# get the 50 videos from each channel in all_channels
video_list = []
for channel in all_channels:
    video_list.append(get_50_videos(channel))

# flatten the list
video_list = [item for sublist in video_list for item in sublist]

# convert to csv
df = pd.DataFrame(video_list)
df.to_csv('video_list.csv', index=False)

video_list
```

```
video_id                             object
channel_id                           object
video_title                          object
video_title_clean                    object
published               datetime64[ns, UTC]
video_views                           int64
video_madeforkids                      bool
video_likes                           int64
video_comment_count                   int64
video_length                         object
video_description                    object
video_tags                           object
dtype: object
```

Dropped columns: video_title, video_madeforkids, video_description, video_tags

Data Source (Tables)

After using the YouTube API, the resulting data was contained in two tables:

Our joined dataframe contains 8603 rows and 13 columns.

- Channels Table
  - Channel ID
  - Channel Name
  - Channel Category
  - View Count
  - Subscriber Count
  - Video Count

Id	Custom Url	Video_category	View Count	Subscriber Cou	Video Count
UCVaXclURQZlal	@news19wltx	Society	150105437	249000	24026
UCQqaNnVhS1v	@channel4come	Humour	132644274	167000	726
UC6YN4FNhAKN	@queencitynew:	Society	83768935	125000	11538
UCjlgDApB1OrU	@cookingwithsh	Food	44025876	422000	453
UCf7J0vxbg6SsIj	@britishcomedy	Film	40935727	37800	951

- Videos Table
  - Channel ID
  - Video ID
  - Video Title
  - Published Date
  - Video Length
  - Comment Count
  - Like Count
  - View Count

Channel_id	Video_id	Video_title_clea	Published	Video_length	Video_commen	Video_likes	Video_views
UC0VOyT2OCBK	Sujm6756pZU	Ariana Grande m	2020-10-30 04:0	2:40	14048	300403	13537474
UCbCmjCuTUZo	hqFfJBOrvHw	Humpty Dumpty	2022-09-27 07:0	2:42	0	33185	8768272
UCbCmjCuTUZo	LA2q3QwhG54	Belly Button Dar	2022-07-05 16:0	2:42	0	105966	31790993
UC3gNmTGu-TT	686jgVnwh5M	Morbius 2022 Br	2022-10-31 15:4	2:43	12	111	5855
UC3gNmTGu-TT	RMrcKVzwqyU	Resident Evil Aft	2022-10-27 15:4	2:43	6	105	6452

Data Analysis Phase

Channels Dataframe

	id	customUrl	video_category	viewCount	subscriberCount	videoCount
0	UCVaXclURQZlakiTMzuwHvRw	@news19wltx	Society	150105437	249000	24026
1	UCQqaNnVhS1w_iTeFalJsXog	@channel4comedy	Humour	132644274	167000	726
2	UC6YN4FNhAKN3MDO5DbJSnOA	@queencitynews	Society	83768935	125000	11538
3	UCjlgDApB1OrU_3-1dLMHOZg	@cookingwithshotgunred	Food	44025876	422000	453
4	UCf7J0vxbg6SsljY9587PEiQ	@britishcomedyguide	Film	40935727	37800	951
...	...	...	...	...	...	...
105	UCXsQlHGuoWqukC9vz-uonrg	@collinabroadcast	Tourism	163749185	1460000	91
106	UCdPambxHRj0kdFPNoJFM98A	@georgebenson	Sport	151941207	1020000	1266
107	UC_ptyMRLOsS1Uj0a34a_xCA	@chonnyday	Food	126569583	689000	427
108	UCJsSEDFFnMFvW9JWU6XUn0Q	@seekerstories	Lifestyle	88299661	542000	257
109	UCd5xLBi_QU6w7RGm5TTznyQ	@sundayfundayz	Tourism	75488060	643000	400

Videos Dataframe

	channel_id	video_id	video_title_clean	published	video_length	video_comment_count	video_likes	video_views
	UCbCmjCuTUZos6Inko4u57UQ	lmH5uqwaFq8	Airplane Song CoComelon Nursery Rhymes Kids Songs	2022-11-01 07:00:15+00:00	2:59	0	19653	3425275
	UCbCmjCuTUZos6Inko4u57UQ	0SY0Yn0yF9o	Wheels On The Bus More Nursery Rhymes Kids Son...	2022-10-29 07:00:00+00:00	29:52:00	0	15076	2882582
	UCbCmjCuTUZos6Inko4u57UQ	sNyF7BvVfxs	Bingos Bath Song CoComelon Nursery Rhymes Kids...	2022-10-25 07:00:16+00:00	2:49	0	47763	8673081
	UCbCmjCuTUZos6Inko4u57UQ	K4kqqCzF-BA	Play Outside at the Beach Song More Nursery Rh...	2022-10-22 07:00:12+00:00	1:01:21	0	57936	11744611
	UCbCmjCuTUZos6Inko4u57UQ	gfZmvllWVwY	Halloween Song Dance Dance Party CoComelon Nur...	2022-10-18 07:00:19+00:00	2:39	0	46041	8775011
	...	...	...	...	...	...	...	...
	UCC7jlYxfWti7WAW8r7ef1RQ	xRi8SGcRDOY	In Paris use the Navigo pass like locals and s...	2022-03-28 15:00:24+00:00	9	0	4	311
	UCC7jlYxfWti7WAW8r7ef1RQ	C-34plsWZPk	How to save money in Paris by using the Paris ...	2022-03-27 15:00:04+00:00	4:49	79	166	4642
	UCC7jlYxfWti7WAW8r7ef1RQ	L2GdB1gB1ZM	Magical Malta should 100 be on your bucket lis...	2022-03-26 15:00:09+00:00	16	7	133	5091
	UCC7jlYxfWti7WAW8r7ef1RQ	89lewFGQQ6E	Is France on your bucket list shorts france tr...	2022-03-25 18:00:03+00:00	16	0	1	31
	UCC7jlYxfWti7WAW8r7ef1RQ	QS32cM-BQ6M	Subscribe for more travel tips and inspiration...	2022-03-24 15:00:11+00:00	11	0	2	76

Joining Databases Together

postgres/postgres@AWS

Query Editor

```sql
CREATE TABLE channel_data(
    channel_id varchar NOT NULL,
    custom_url varchar,
    topic_category varchar,
    channel_view_count bigint,
    subscriber_count bigint,
    channel_video_count bigint,
    PRIMARY KEY (channel_id)
);

CREATE TABLE video_data(
    channel_id varchar NOT NULL,
    video_id varchar NOT NULL,
    video_title_clean varchar,
    published_at timestamp,
    video_length varchar,
    comment_count bigint,
    like_count bigint,
    view_count bigint,
FOREIGN KEY (channel_id) REFERENCES channel_data (channel_id)
);
```

```sql
--- join tables together
SELECT c.channel_id,
    c.custom_url,
    c.topic_category,
    c.channel_view_count,
    c.subscriber_count,
    c.channel_video_count,
    v.video_id,
    v.published_at,
    v.video_length,
    v.like_count,
    v.comment_count,
    v.view_count
INTO joined_data
FROM channel_data AS c
INNER JOIN video_data AS v
ON c.channel_id=v.channel_id;

SELECT * FROM joined_data;
```

Joined Data Table

	channel_id	custom_url	topic_category	channel_view_count	subscriber_count	channel_video_count	video_id	video_length	like_count	comment_count	view_count
0	UCbCmjCuTUZos6Inko4u57UQ	@cocomelon	Music	142468175305	146000000	811	ImH5uqwaFq8	2:59	19653	0	3425275
1	UCbCmjCuTUZos6Inko4u57UQ	@cocomelon	Music	142468175305	146000000	811	0SY0Yn0yF9o	29:52:00	15076	0	2882582
2	UCbCmjCuTUZos6Inko4u57UQ	@cocomelon	Music	142468175305	146000000	811	sNyF7BvVfxs	2:49	47763	0	8673081
3	UCbCmjCuTUZos6Inko4u57UQ	@cocomelon	Music	142468175305	146000000	811	K4kqqCzF-BA	1:01:21	57936	0	11744611
4	UCbCmjCuTUZos6Inko4u57UQ	@cocomelon	Music	142468175305	146000000	811	gfZmvllWVwY	2:39	46041	0	8775011

Binned Data Sample View

channel_views_binned	subscribers_binned	video_count_binned	like_count_binned	comment_binned	video_views_binned
1 bil- 100 billion views	5-15 mil subs	5,000-10,000 videos	1,000-5,000 likes	100-500 comments	50,000-500,000
1 mil-5 mil views	10,000-500,000 subs	100-500 videos	less than 50 likes	less than 100 comments	less than 1000 views
1 bil- 100 billion views	5-15 mil subs	500-1500 videos	1,000-5,000 likes	less than 100 comments	50,000-500,000
5 mil- 100 mil	500,000- 1 mil subs	100-500 videos	50-1000 likes	100-500 comments	10,000-50,000 views
1 bil- 100 billion views	5-15 mil subs	greater than 10,000 videos	1,000-5,000 likes	1,000-10,000 comments	50,000-500,000

Combined Topic Categories

```
['Music',
 'Video_game_culture',
 'Lifestyle',
 'Hobby',
 'Hip_hop_music',
 'Entertainment',
 'Action-adventure_game',
 'Rock_music',
 'Pop_music',
 'Film',
 'Knowledge',
 'Food',
 'Physical_fitness',
 'Society',
 'Strategy_video_game',
 'Technology',
 nan,
 'Television_program',
 'Politics',
 'Electronic_music',
 'Sport',
 'Tourism',
 'Classical_music',
 'Christian_music',
 'American_football',
 'Military',
 'Humour']
```

26 to 16 topic categories

```
['Music',
 'Video_game_culture',
 'Lifestyle',
 'Hobby',
 'Entertainment',
 'Film',
 'Knowledge',
 'Food',
 'Physical_fitness',
 'Society',
 'Technology',
 'Television_program',
 'Politics',
 'Sport',
 'Tourism',
 'Humour']
```

Adding Day of Week Published Column

```
#turn published at to day of week published
file["datetime"]=pd.to_datetime(file['published_at'])
file['day_of_week_published']=file['datetime'].dt.day_name()
```

published_at	v
2022-07-22 19:00:16	
2021-06-26 20:00:02	
2019-11-19 20:00:11	
2022-10-28 17:00:27	
2019-09-21 13:00:02	

day_of_week_published
Friday
Saturday
Tuesday
Friday
Saturday

Dropping Video Length

```
channel_id                     object
custom_url                     object
topic_category                 object
channel_view_count              int64
subscriber_count                int64
channel_video_count             int64
video_id                       object
published_at           datetime64[ns]
video_length                   object
like_count                      int64
comment_count                   int64
view_count                      int64
day_of_week_published          object
dtype: object
```

Video_length object refuses to convert to timestamp or duration datatype

Machine Learning

Our Features

topic_category	channel_view_count	subscriber_count	channel_video_count	like_count	comment_count	day_of_week_published	viral
Lifestyle	346822518	1610000	1919	2057	86	Wednesday	0
Music	17894740372	35700000	97	353367	10950	Monday	1
Lifestyle	513233182	5580000	268	25982	660	Thursday	0
Hobby	83311249514	102000000	1030	66433	0	Thursday	1
Lifestyle	462242	2620	275	20	2	Sunday	0

Target Variable

How did we get the target column?

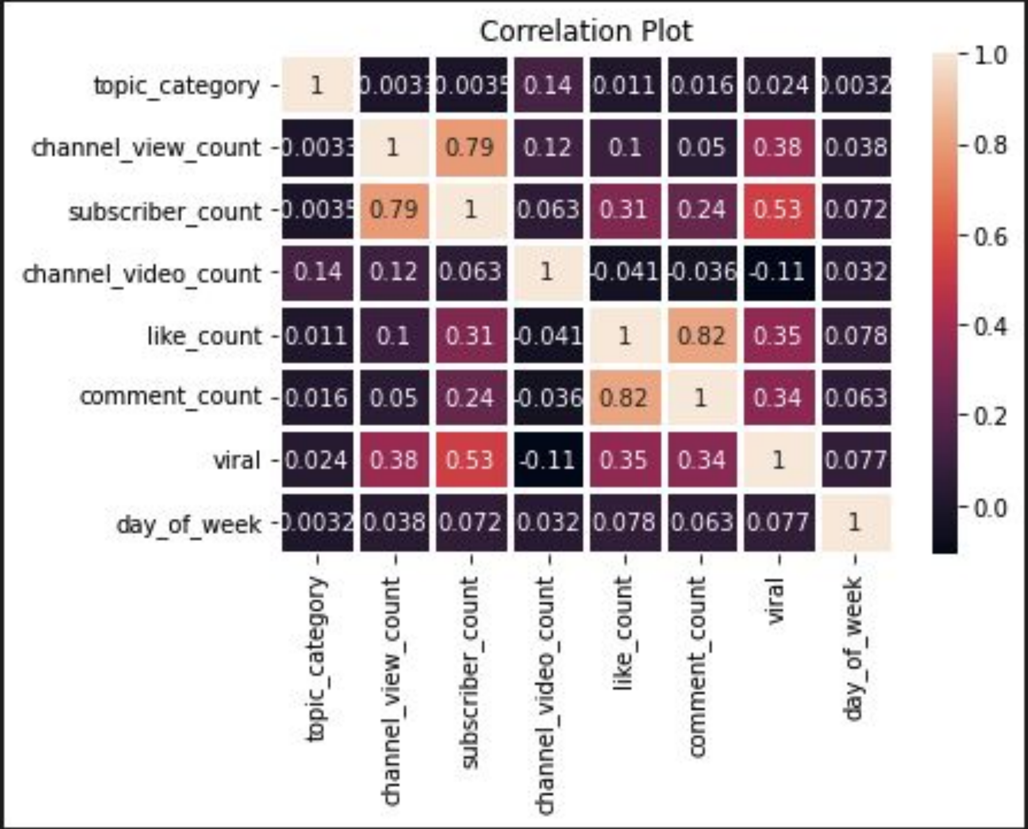
```python
#make new column that has binary classification. if view count is greather than 1,000,000 then add 1 if less than add 0
def viral(row):
    if row['view_count'] > 1000000:
        return 1
    else:
        return 0
```

✓ 0.1s

```python
#add the viral column
df['viral']=df.apply(lambda row: viral(row), axis=1)
df.sample(5)
```

✓ 0.3s

Correlation Exploration



Encoding and Scaling

```
#encode categorical data
le = LabelEncoder()
df2 = df.copy()
df2['topic_category'] = le.fit_transform(df2['topic_category'])
df2['day'] = le.fit_transform(df2['day_of_week_published'])
df2.sample(5)
```

✓ 0.9s

	topic_category	channel_view_count	subscriber_count	channel_video_count	like_count	comment_count	day_of_week_published	viral	day
6704	6	767928	10000	461	22	7	Friday	0	0
2819	15	106661887	895000	82	17590	1398	Saturday	0	2
4364	0	10215915230	24900000	1406	27894	1097	Monday	0	1
4150	7	19286274611	110400000	250	513038	18690	Friday	1	0
6861	2	145679	1870	115	13	5	Monday	0	1

```python
# Day of Week dictionary
weekday_num = {
    "Sunday": 1,
    "Monday": 2,
    "Tuesday": 3,
    "Wednesday": 4,
    "Thursday": 5,
    "Friday": 6,
    "Saturday": 7
}
```

✓ 0.1s

Custom encoding for the days of the week

```python
# weekdays names encoded using the dictionary values
df2["day_of_week"] = df2["day_of_week_published"].apply(lambda x: weekday_num[x])
df2=df2.drop(columns=['day','day_of_week_published'])
df2.head()
```

✓ 0.1s

	topic_category	channel_view_count	subscriber_count	channel_video_count	like_count	comment_count	viral	day_of_week
0	7	142468175305	146000000	811	19653	0	1	3
1	7	142468175305	146000000	811	15076	0	1	7
2	7	142468175305	146000000	811	47763	0	1	3
3	7	142468175305	146000000	811	57936	0	1	7
4	7	142468175305	146000000	811	46041	0	1	3

Scaling Feature Data

```
stds=StandardScaler()
df_scaled=stds.fit_transform(X.to_numpy())
df_scaled=pd.DataFrame(df_scaled,columns=['topic_category','channel_view_count','subscriber_count','channel_video_count','like_count','comment_count','day_of_week'])

df_scaled.head()
```

✓ 0.3s

	topic_category	channel_view_count	subscriber_count	channel_video_count	like_count	comment_count	day_of_week
0	0.21331	7.794011	4.370754	-0.204552	-0.148934	-0.188519	-0.613740
1	0.21331	7.794011	4.370754	-0.204552	-0.159197	-0.188519	1.578497
2	0.21331	7.794011	4.370754	-0.204552	-0.085904	-0.188519	-0.613740
3	0.21331	7.794011	4.370754	-0.204552	-0.063094	-0.188519	1.578497
4	0.21331	7.794011	4.370754	-0.204552	-0.089765	-0.188519	-0.613740

Logistic Regression

We chose this model because we wanted to use it to train on our labeled datasets and aim to predict which category the new observations (testing data) will belong to.

Our results before resampling the data.

The precision score was very low.

Accuracy Score: 0.9488609948860995

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1694
1	0.96	0.79	0.87	457
accuracy			0.95	2151
macro avg	0.95	0.89	0.92	2151
weighted avg	0.95	0.95	0.95	2151

Random Forest Classifier

This model is better at handling our numerous variables and big data set.

```
Accuracy Score:  0.9700468121494579
```

```
[(0.4956314070603417, 'like_count'),
 (0.18758883148965227, 'comment_count'),
 (0.11188914380045174, 'subscriber_count'),
 (0.10249570056339609, 'channel_view_count'),
 (0.06270051475775136, 'channel_video_count'),
 (0.0234488363920342, 'topic_category'),
 (0.016245565936372698, 'day_of_week')]
```

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1694
1	0.89	0.97	0.93	457
accuracy			0.97	2151
macro avg	0.94	0.97	0.95	2151
weighted avg	0.97	0.97	0.97	2151

Easy Ensemble (AdaBoost) Classifier

This ML Model was used to see if we could get a better accuracy score than the Random Forest Model.

```
Accuracy Score:  0.9558946623299119
```

```
# Display the confusion matrix
confusion_matrix(y_test, y_pred)
✓ 0.1s

array([[1615,   79],
       [  19,  438]])
```

Classification Report- Easy Ensemble (AdaBoost) Classifier

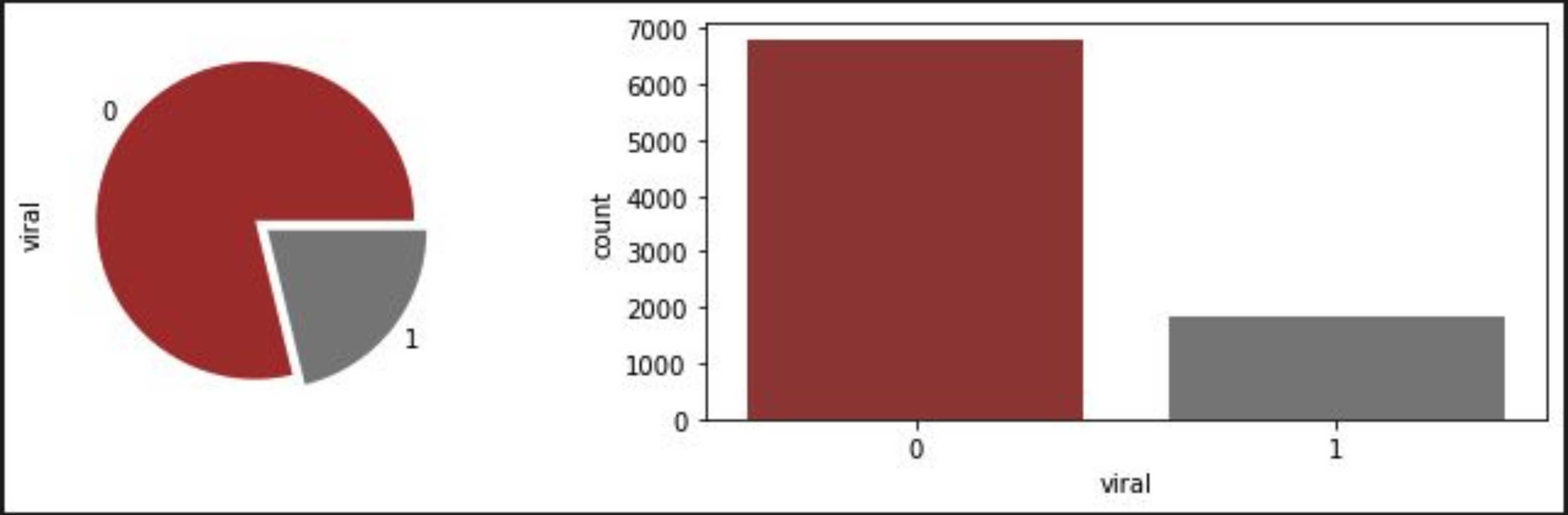
	pre	rec	spe	f1	geo	iba	sup
0	0.99	0.95	0.96	0.97	0.96	0.91	1694
1	0.85	0.96	0.95	0.90	0.96	0.91	457
avg / total	0.96	0.95	0.96	0.96	0.96	0.91	2151

Wait… Our Data is Imbalanced

```
#percentage viral/not viral
print("Viral: ", df.viral.value_counts()[1]/len(df)*100,"%")
print("Not Viral: ", df.viral.value_counts()[0]/len(df)*100,"%")
```

✓ 0.8s

```
Viral:  21.225154016040914 %
Not Viral:  78.77484598395908 %
```



Resampling using SMOTEENN

```python
#split data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=1,
                                                    stratify=y)

X_train.shape
```

✓ 0.2s

```
(6452, 7)
```

```python
y.value_counts()
```

✓ 0.1s

```
0    6777
1    1826
Name: viral, dtype: int64
```

```python
from imblearn.combine import SMOTEENN
from collections import Counter

smote_enn = SMOTEENN(random_state=0)
X_resampled, y_resampled = smote_enn.fit_resample(X, y)
Counter(y_resampled)
```

✓ 0.9s

```
Counter({0: 6451, 1: 6366})
```

Logistic Regression

```
#create logistic regression model
classifier = LogisticRegression(solver='lbfgs',
                                max_iter=200,
                                random_state=1)

#fit train the model
classifier.fit(X_resampled, y_resampled)
```

✓ 2.5s

```
LogisticRegression(max_iter=200, random_state=1)
```

Was 94% before resampling!

```
#accuracy score
print(accuracy_score(y_test, y_pred))
```

✓ 0.1s

```
0.9539748953974896
```

```
# Display the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

✓ 0.8s

```
array([[1631,   63],
       [  36,  421]])
```

Logistic Regression Classification Report

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1694
1	0.96	0.79	0.87	457
accuracy			0.95	2151
macro avg	0.95	0.89	0.92	2151
weighted avg	0.95	0.95	0.95	2151

Before resampling

	precision	recall	f1-score	support
0	0.98	0.96	0.97	1694
1	0.87	0.92	0.89	457
accuracy			0.95	2151
macro avg	0.92	0.94	0.93	2151
weighted avg	0.96	0.95	0.95	2151

After resampling

Random Forest Model

```
# Create a random forest classifier
from imblearn.ensemble import BalancedRandomForestClassifier

model = BalancedRandomForestClassifier(n_estimators=100, random_state=1)

# Fitting the model
model.fit(X_resampled, y_resampled)
```

✓ 2.4s

```
BalancedRandomForestClassifier(random_state=1)
```

Was 97.00% before resampling!

```
# Calculated the balanced accuracy score
y_pred=model.predict(X_test)
balanced_accuracy_score(y_test,y_pred)
```

✓ 0.2s

```
0.9774257968011697
```

```
# List the features sorted in descending order by feature importance
sorted(zip(model.feature_importances_, X.columns), reverse=True)
```

✓ 0.1s

```
[(0.5087001150346812, 'like_count'),
 (0.19153776410912765, 'comment_count'),
 (0.11684245155319044, 'subscriber_count'),
 (0.0911579778379475, 'channel_view_count'),
 (0.06393489936816082, 'channel_video_count'),
 (0.020181369871470057, 'topic_category'),
 (0.007645422225422378, 'day_of_week')]
```

Random Forest Model-Classification Report

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1694
1	0.89	0.97	0.93	457
accuracy			0.97	2151
macro avg	0.94	0.97	0.95	2151
weighted avg	0.97	0.97	0.97	2151

Before resampling

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1694
1	0.93	0.97	0.95	457
accuracy			0.98	2151
macro avg	0.96	0.98	0.97	2151
weighted avg	0.98	0.98	0.98	2151

After resampling

Easy Ensemble (AdaBoost) Model

```python
# Train the EasyEnsembleClassifier
from imblearn.ensemble import EasyEnsembleClassifier

model_eec=EasyEnsembleClassifier(n_estimators=100, random_state=1)
model_eec.fit(X_resampled, y_resampled)
```

✓ 1m 21.9s

```
EasyEnsembleClassifier(n_estimators=100, random_state=1)
```

Was 95.5% before resampling!

```python
# Calculated the balanced accuracy score
y_pred=model_eec.predict(X_test)

balanced_accuracy_score(y_test, y_pred)
```

✓ 3.3s

```
0.968535492754709
```

```python
# Display the confusion matrix
confusion_matrix(y_test, y_pred)
```

✓ 0.1s

```
array([[1643,   51],
       [  15,  442]])
```

Classification Reports- AdaBoost Model

Before resampling

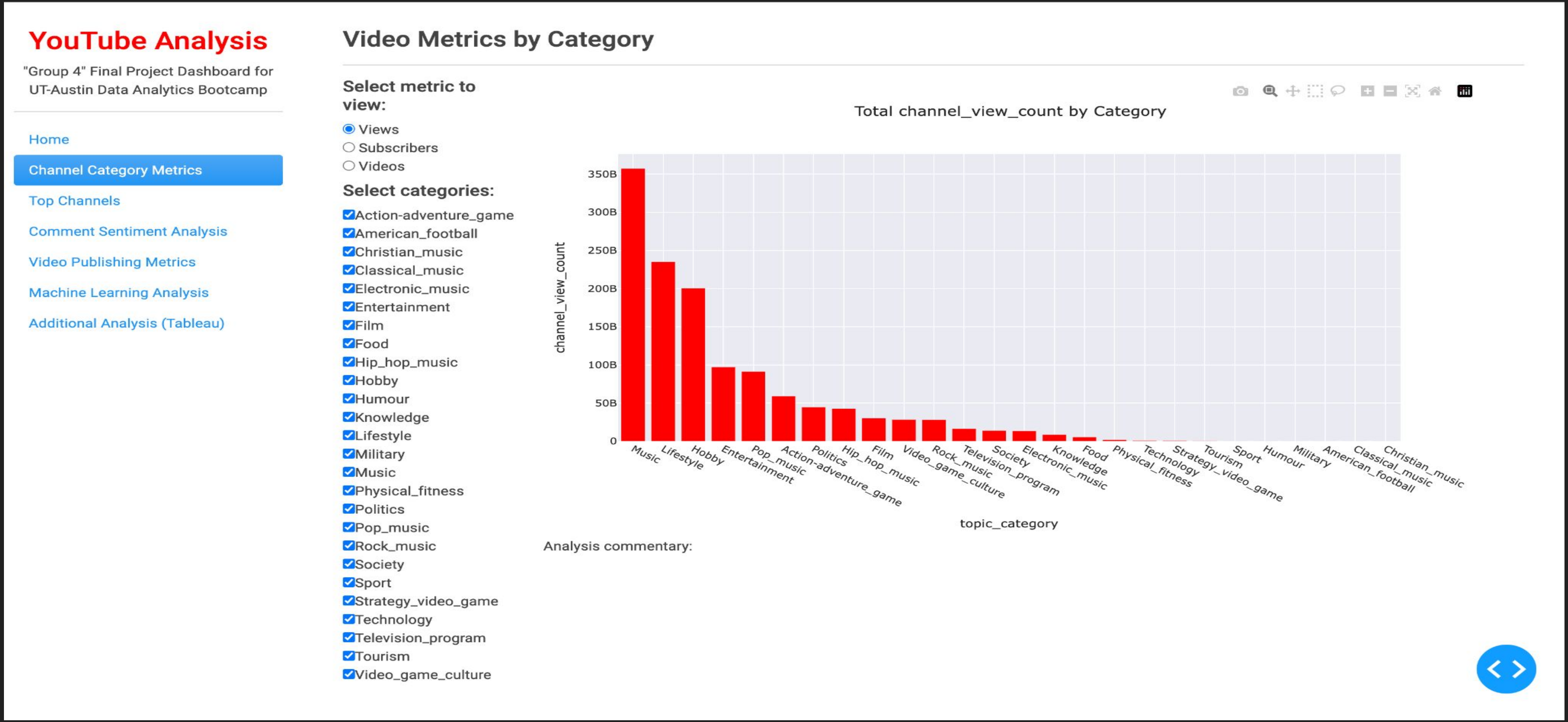
	pre	rec	spe	f1	geo	iba	sup
0	0.99	0.95	0.96	0.97	0.96	0.91	1694
1	0.85	0.96	0.95	0.90	0.96	0.91	457
avg / total	0.96	0.95	0.96	0.96	0.96	0.91	2151

After resampling

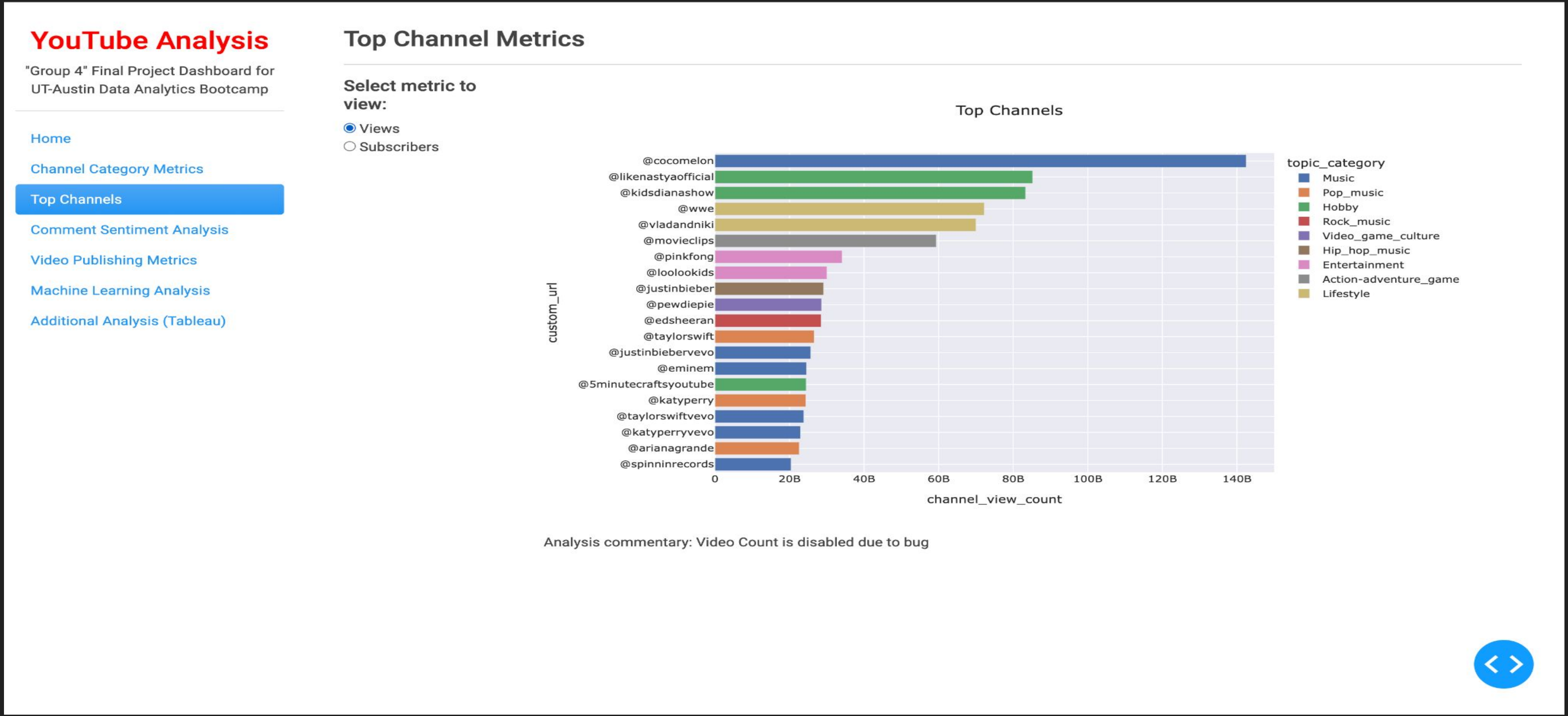
	pre	rec	spe	f1	geo	iba	sup
0	0.99	0.97	0.97	0.98	0.97	0.94	1694
1	0.90	0.97	0.97	0.93	0.97	0.94	457
avg / total	0.97	0.97	0.97	0.97	0.97	0.94	2151

Dashboard

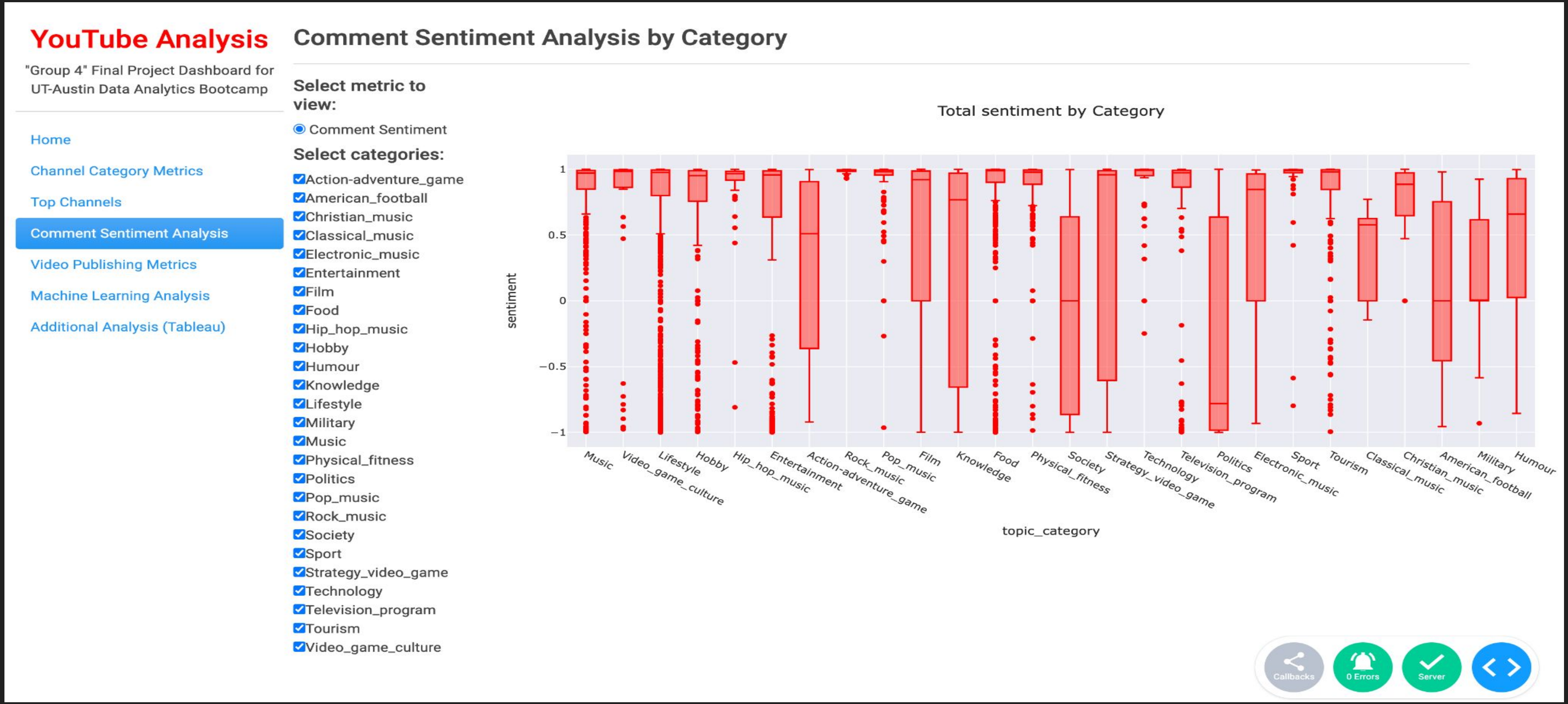
Images of Dashboard



Images of Dashboard



Images of Dashboard



Images of Dashboard

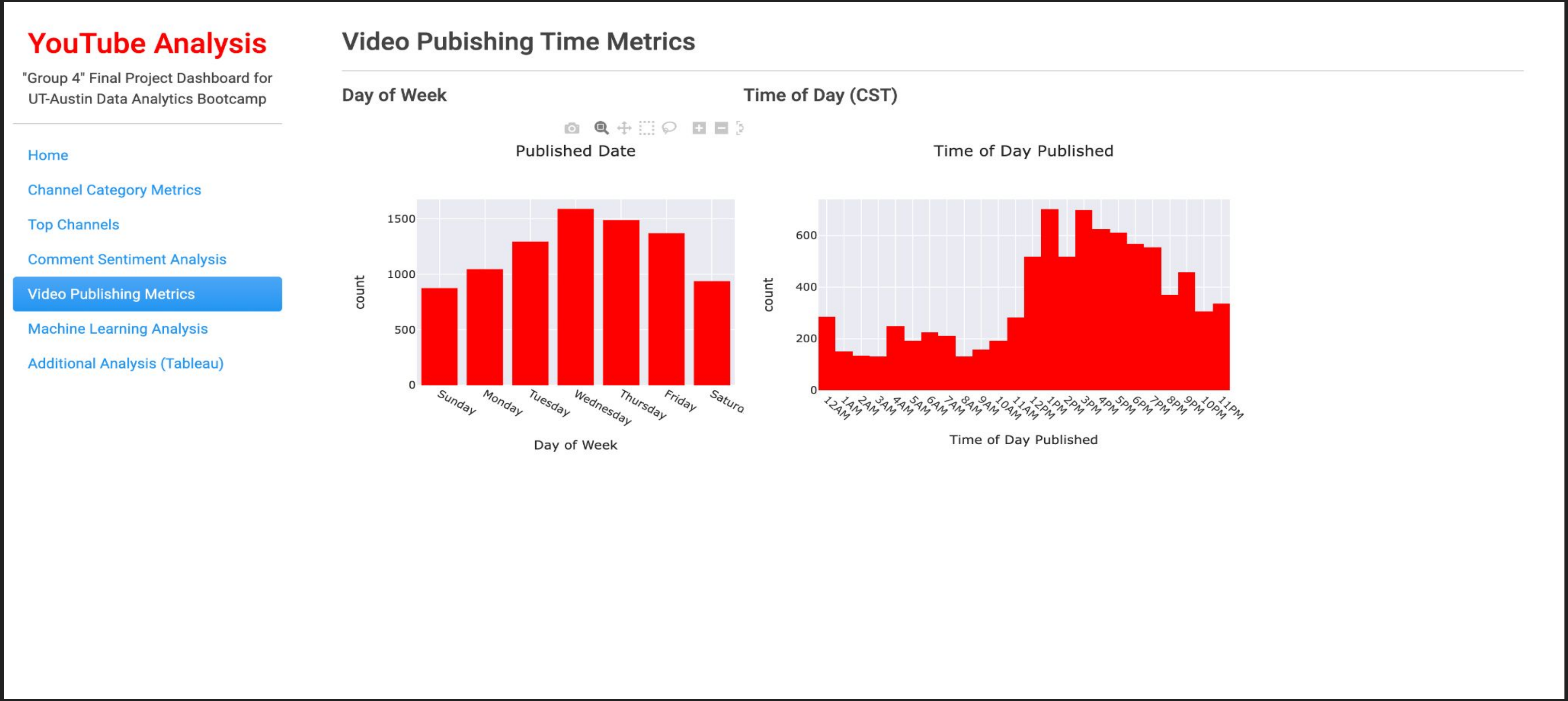
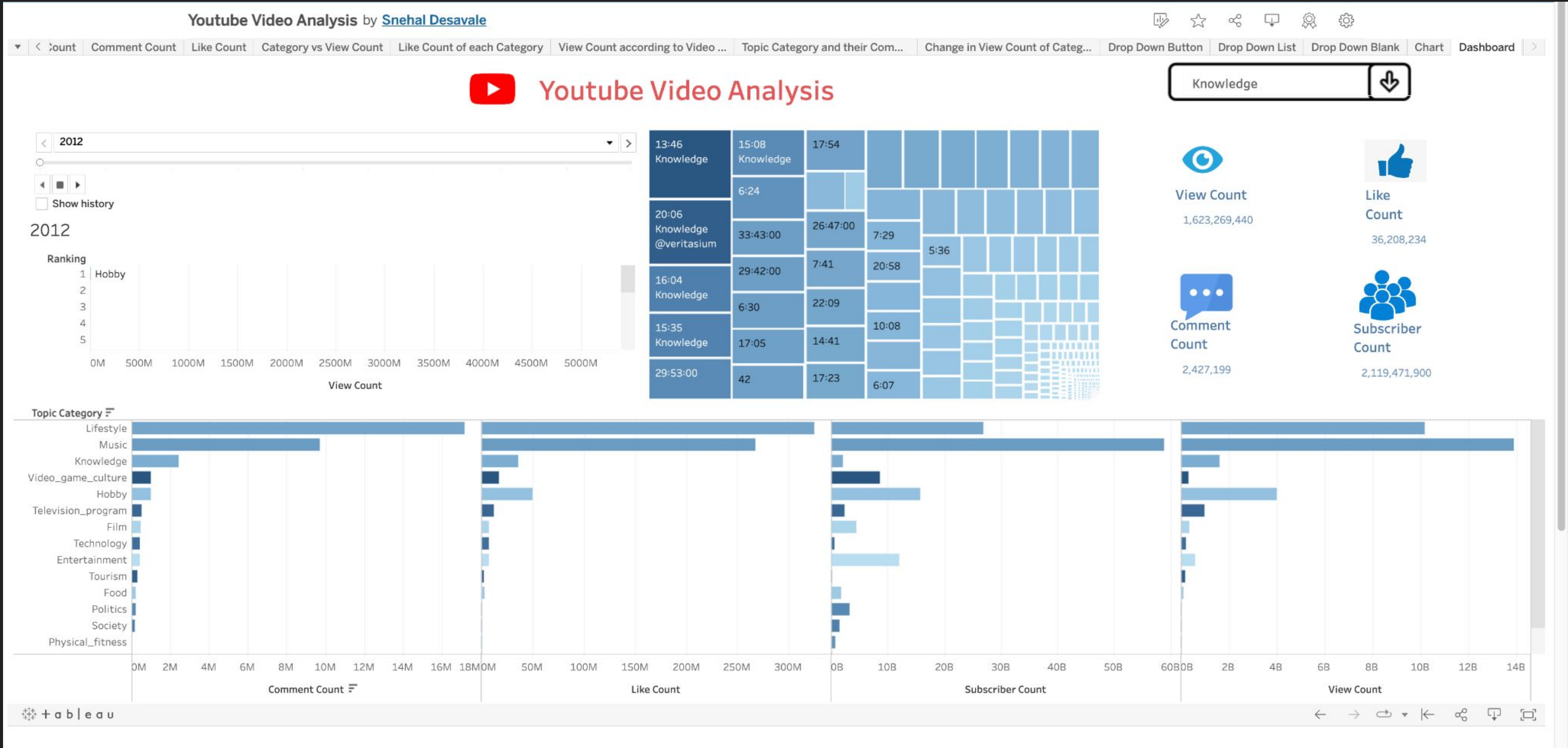


Tableau Dashboard



Things We Would Have Done Differently

- More interactive elements on the Dashboard
- Included video length as a feature in machine learning component
- Included the hour video was published as a feature in our machine learning component
- We really wanted to use API to gather data, but wished we had more time exploring what the YouTube API could do before sticking with it

Any Questions?