# YouTube Like Prediction for Trending Videos

Chandler Ebrahimi

# The Problem

People who post videos on YouTube often, such as professional YouTuber’s would need to know what kind of video, with what results they should strive to achieve to get a viral video.

# Who might care?

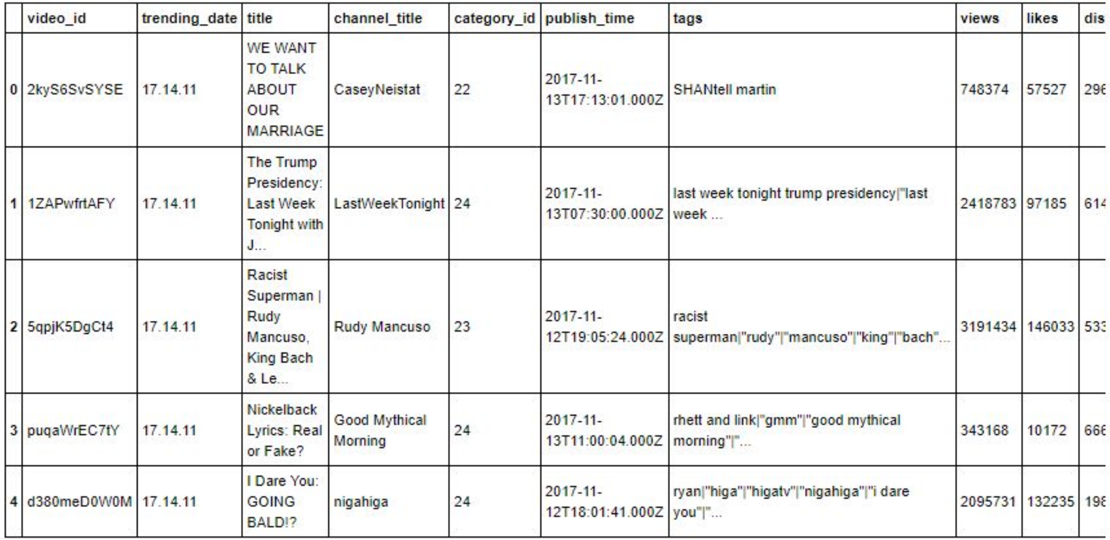
Existing YouTubers

People who want to get into the industry

Sponsors and advertisements seeking out videos to place ads on.

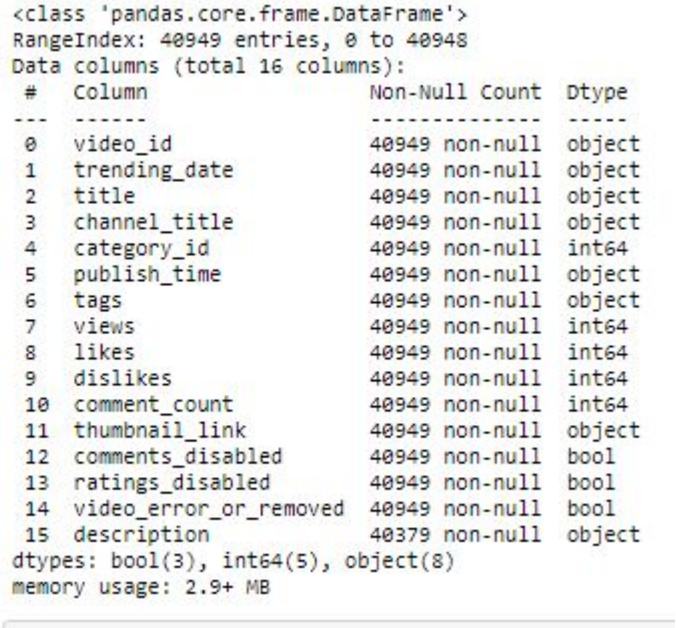
# Data Acquisition

In the dataset there were 40,000 entries of viral youtube videos that trended mainly in 2017 and 2018. All formatted in a csv file.



| | video_id | trending_date | title | channel_title | category_id | publish_time | tags | views | likes | dis |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2kyS6SvSYSE | 17.14.11 | WE WANT TO TALK ABOUT OUR MARRIAGE | CaseyNeistat | 22 | 2017-11-13T17:13:01.000Z | SHANtell martin | 748374 | 57527 | 296 |
| 1 | 1ZAPwfrtAFY | 17.14.11 | The Trump Presidency: Last Week Tonight with J... | LastWeekTonight | 24 | 2017-11-13T07:30:00.000Z | last week tonight trump presidency\|"last week ... | 2418783 | 97185 | 614 |
| 2 | 5qpjK5DgCt4 | 17.14.11 | Racist Superman \| Rudy Mancuso, King Bach & Le... | Rudy Mancuso | 23 | 2017-11-12T19:05:24.000Z | racist superman\|"rudy"\|"mancuso"\|"king"\|"bach"... | 3191434 | 146033 | 533 |
| 3 | puqaWrEC7tY | 17.14.11 | Nickelback Lyrics: Real or Fake? | Good Mythical Morning | 24 | 2017-11-13T11:00:04.000Z | rhett and link\|"gmm"\|"good mythical morning"\|"... | 343168 | 10172 | 666 |
| 4 | d380meD0W0M | 17.14.11 | I Dare You: GOING BALD!? | nigahiga | 24 | 2017-11-12T18:01:41.000Z | ryan\|"higa"\|"higatv"\|"nigahiga"\|"i dare you"\|"... | 2095731 | 132235 | 198 |

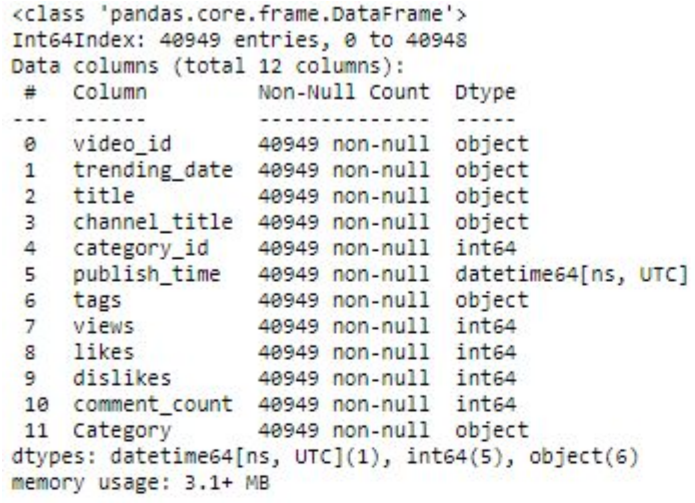
The listed category columns names that was needed to be cleaned.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   video_id                40949 non-null  object
 1   trending_date           40949 non-null  object
 2   title                   40949 non-null  object
 3   channel_title           40949 non-null  object
 4   category_id             40949 non-null  int64 
 5   publish_time            40949 non-null  object
 6   tags                    40949 non-null  object
 7   views                   40949 non-null  int64 
 8   likes                   40949 non-null  int64 
 9   dislikes                40949 non-null  int64 
 10  comment_count           40949 non-null  int64 
 11  thumbnail_link          40949 non-null  object
 12  comments_disabled       40949 non-null  bool  
 13  ratings_disabled        40949 non-null  bool  
 14  video_error_or_removed  40949 non-null  bool  
 15  description             40379 non-null  object
dtypes: bool(3), int64(5), object(8)
memory usage: 2.9+ MB
```

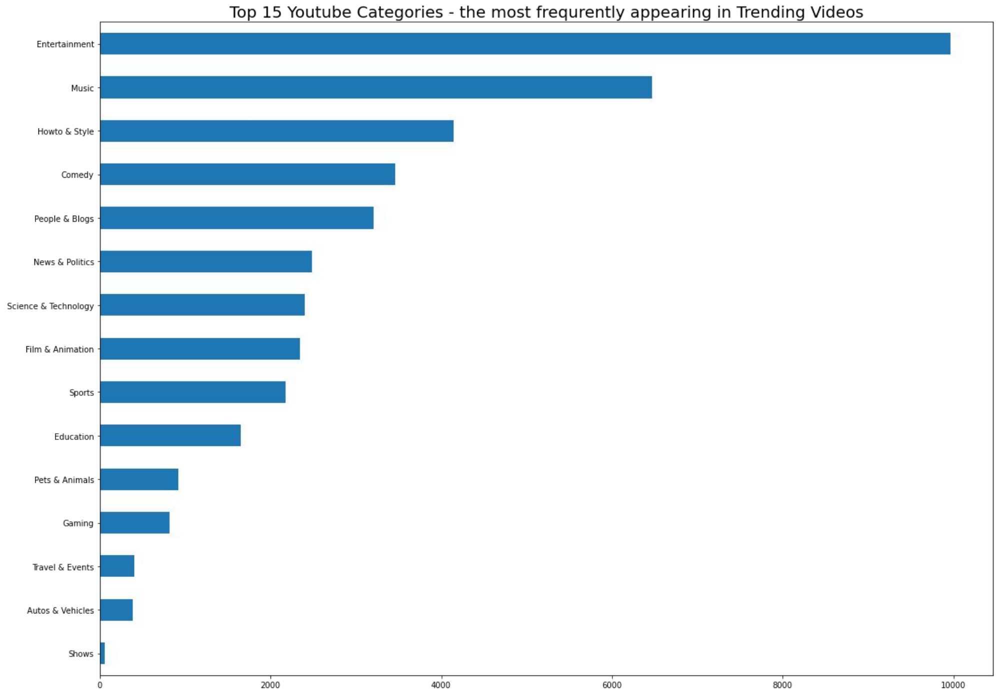
Luckly, there wasn’t any null values that needed to be addressed in any of the columns

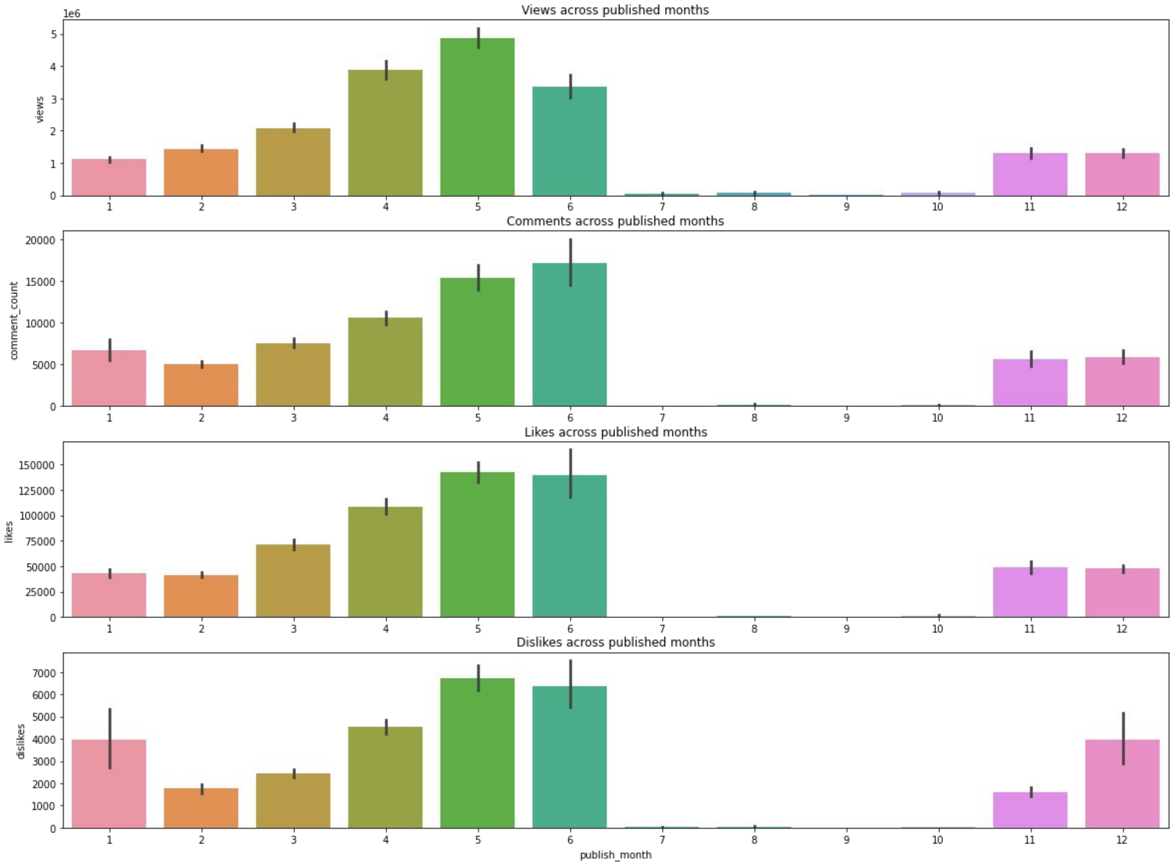
After converting date to its proper object type and dropping irrelevant columns we ended with 11 columns to do exploratory data analysis



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40949 entries, 0 to 40948
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   video_id       40949 non-null  object
 1   trending_date  40949 non-null  object
 2   title          40949 non-null  object
 3   channel_title  40949 non-null  object
 4   category_id    40949 non-null  int64
 5   publish_time   40949 non-null  datetime64[ns, UTC]
 6   tags           40949 non-null  object
 7   views          40949 non-null  int64
 8   likes          40949 non-null  int64
 9   dislikes       40949 non-null  int64
 10  comment_count  40949 non-null  int64
 11  Category       40949 non-null  object
dtypes: datetime64[ns, UTC](1), int64(5), object(6)
memory usage: 3.1+ MB
```

# EDA (Exploratory Data Analysis)



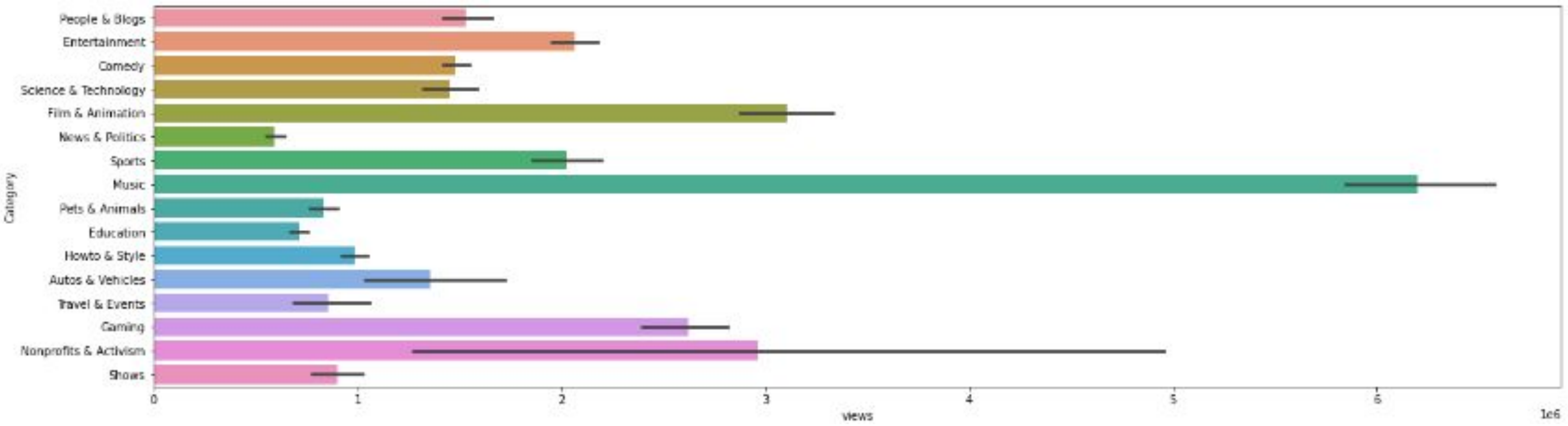


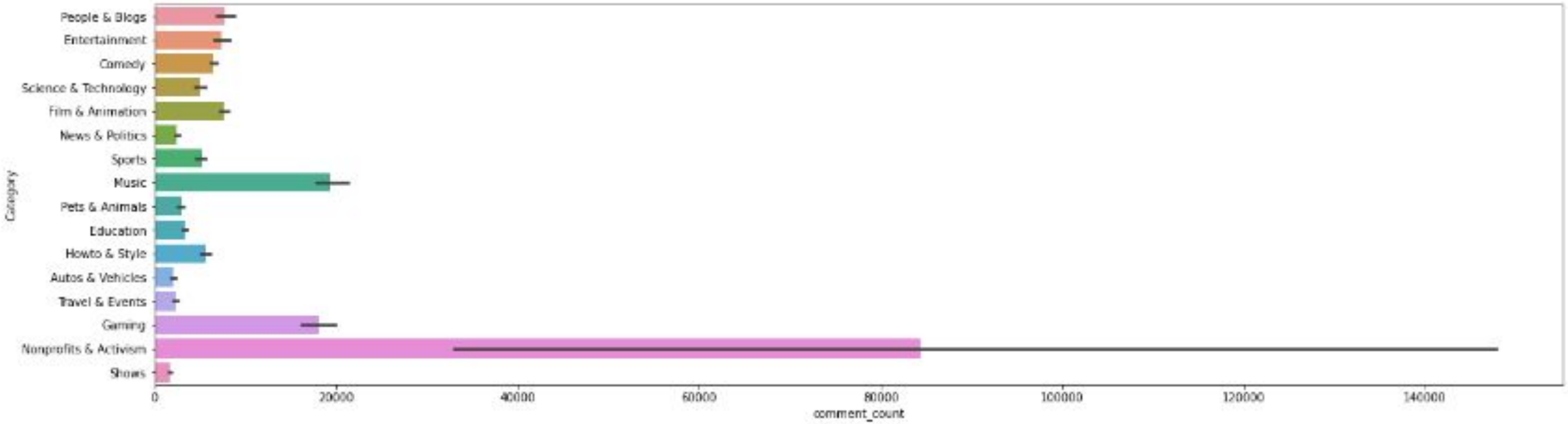
Views across published months

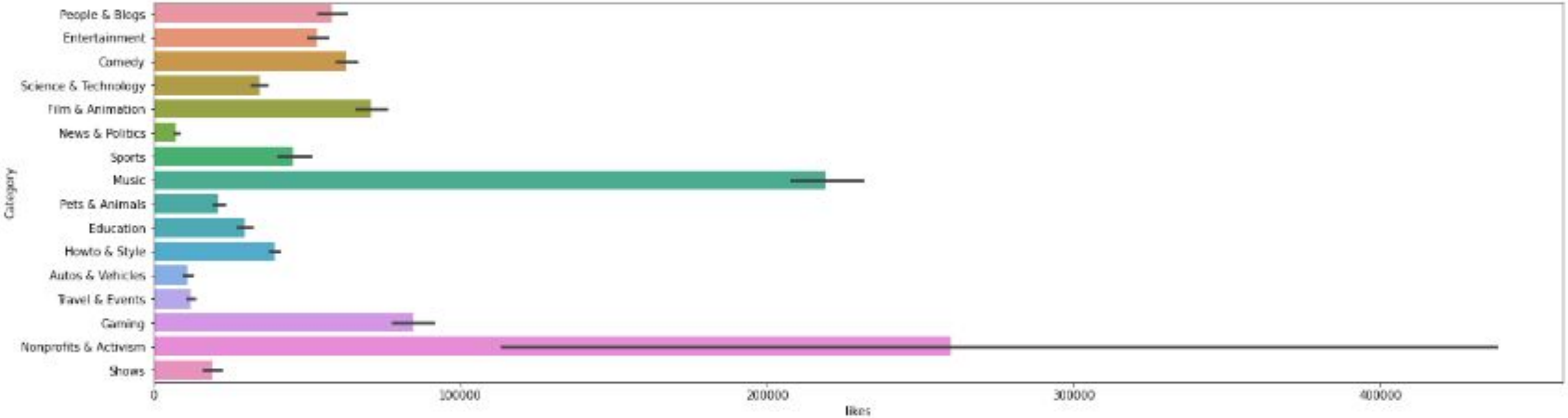
Comments across published months

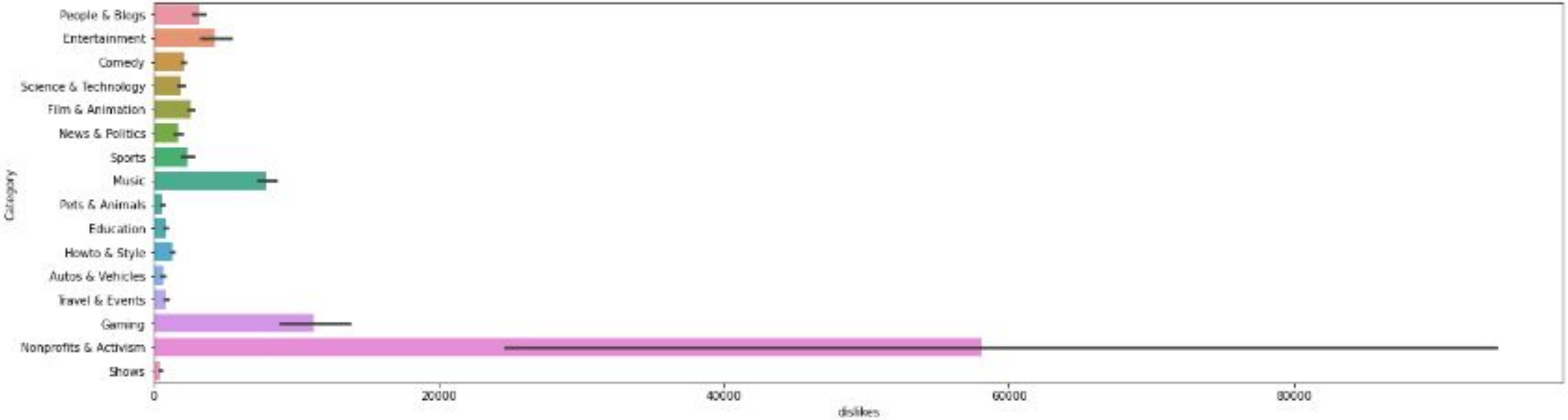
Likes across published months

Dislikes across published months


People & Blogs
Entertainment
Comedy
Science & Technology
Film & Animation
News & Politics
Sports
Music
Pets & Animals
Education
Howto & Style
Autos & Vehicles
Travel & Events
Gaming
Nonprofits & Activism
Shows
Category
0
1
2
3
4
5
6
1e6
views


People & Blogs
Entertainment
Comedy
Science & Technology
Film & Animation
News & Politics
Sports
Music
Pets & Animals
Education
Howto & Style
Autos & Vehicles
Travel & Events
Gaming
Nonprofits & Activism
Shows
Category
0
20000
40000
60000
80000
100000
120000
140000
comment_count

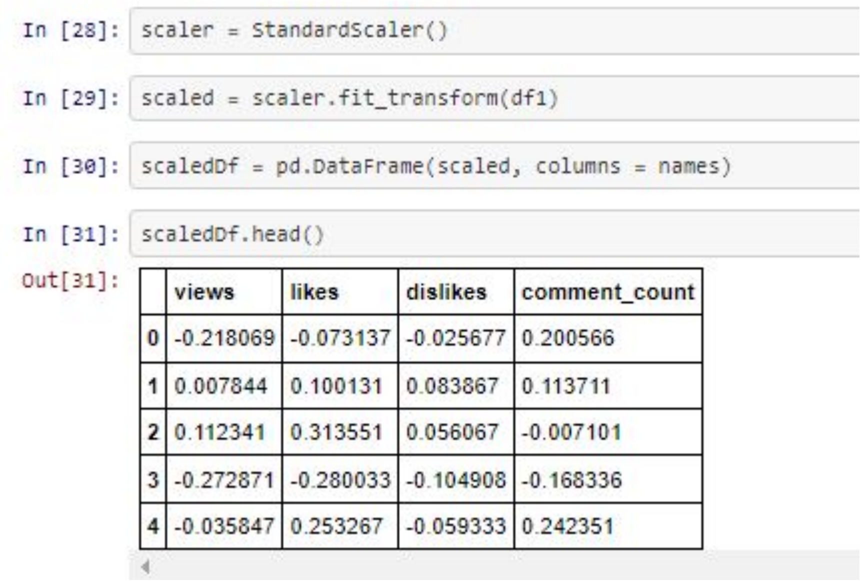

People & Blogs
Entertainment
Comedy
Science & Technology
Film & Animation
News & Politics
Sports
Music
Pets & Animals
Education
Howto & Style
Autos & Vehicles
Travel & Events
Gaming
Nonprofits & Activism
Shows
Category
0
100000
200000
300000
400000
likes


People & Blogs
Entertainment
Comedy
Science & Technology
Film & Animation
News & Politics
Sports
Music
Pets & Animals
Education
Howto & Style
Autos & Vehicles
Travel & Events
Gaming
Nonprofits & Activism
Shows
Category
0
20000
40000
60000
80000
dislikes

# Pre-processing

I took the data and preformed steps:

- Scaled it using Standard Scaler
- Split into train/test with 70/30 split



```
In [28]: scaler = StandardScaler()

In [29]: scaled = scaler.fit_transform(df1)

In [30]: scaledDf = pd.DataFrame(scaled, columns = names)

In [31]: scaledDf.head()

Out[31]:
```

| | views | likes | dislikes | comment_count |
|---|---|---|---|---|
| 0 | -0.218069 | -0.073137 | -0.025677 | 0.200566 |
| 1 | 0.007844 | 0.100131 | 0.083867 | 0.113711 |
| 2 | 0.112341 | 0.313551 | 0.056067 | -0.007101 |
| 3 | -0.272871 | -0.280033 | -0.104908 | -0.168336 |
| 4 | -0.035847 | 0.253267 | -0.059333 | 0.242351 |

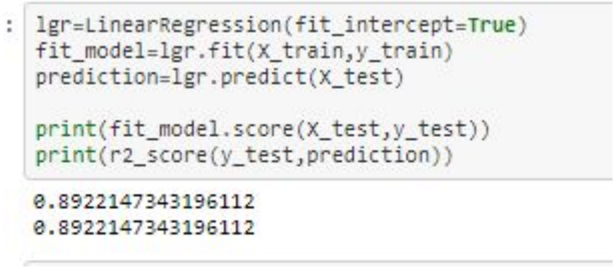
# Modeling

I used the 5 following models

1. Linear Regression
2. Random Forest Classifier
3. Decision Tree Classifier
4. KNeighborsClassifier
5. SVC

# Results

After getting the results back, one model came out more successful in testing than the others. The Linear Regression model came back with an accuracy of 89.22%, while the other models were less significant. The Random Forest Classifier had 46.4%, the Decision Tree Classifier had 47.2%, the K-Neighbors Classifier had a puny 1. 5%, and SVC had even worse, .4%.



```
lgr=LinearRegression(fit_intercept=True)
fit_model=lgr.fit(X_train,y_train)
prediction=lgr.predict(X_test)

print(fit_model.score(X_test,y_test))
print(r2_score(y_test,prediction))
```

```
0.8922147343196112
0.8922147343196112
```