Kate Radford

11537868

CptS 315

Course Project Report

Introduction

Every form of social media today has its own form of an explore or trending page and I've always wondered the criteria needed to qualify. The algorithms behind the selection are so complex and hush that there is room to wonder how fair it all is. How can something that was uploaded so short a time ago already be 'trending?'. And why is the content always from the same sources? I have always viewed trending social media content as a little rigged, so I set out to get some answers.

Some broad questions I began with were

- What attributes make a YouTube video popular?
- Which attributes in combination can predict if a video is trending?
- Is it possible to build a classifier? If so, which algorithm should I use?

In the beginning I only had my questions and my Kaggle dataset found in the course-project-ideas.txt. To gather more direction, I headed to google, searching strings like "data mining trending YouTube videos". I found a couple articles from others who tinkered with the same dataset, which gave me some inspiration for types of analysis I could explore.

Looking at the datapoints available for each video in the dataset, I picked out the useful ones: title, category, tags, views, likes, dislikes, and date/ time posted. I then figured out what data I could get out of these attributes.

For the classifier, I zeroed in on binary classification since the video is either trending or not trending. I remembered the perceptron algorithm from class and could see turning the numerical attributes into feature sets. I also pondered the idea of a decision tree, either making my own or finally exploring the sklearn python library. I decided to start with the decision tree to be an adventurous learner, and if I had time I would return to the perceptron (I didn't).

One important challenge I faced was the fact that my dataset only included trending YouTube videos. This was fine for my simpler analysis of the trending video attributes, but to build a classifier, I also needed non-trending video data. I googled this to confirm and came up with the term "survivorship-bias", which I recognized from my ethics class and knew I did not want.

I was successful in identifying attribute values that correlate to trending videos like time uploaded, title words and tags. With this knowledge I was able to optimize my decision tree classifier to an 83% accuracy.

Data Mining Task

Input Data

It consists of a series of YouTube video statistics in a csv file that contains the data records in the form [video_id, title, publishedAt, channelId, channelTitle, categoryId, trending_date, tags, view_count, likes, dislikes, comment_count, thumbnail_link, comments_disabled, ratings_disabled, description]. There was also a supporting JSON file containing the category ids and names.

For my classifier, the data records were in the form [video_id, title, channel_title, category_id, publish_time, tags, views, likes, dislikes, label]. The label set as 1 for trending or 0 for not-trending. I split the data into 70% training set and 30% testing set.

Task Details

My data mining process was to start simple and work my way up. I drew out a plan for trending video data analysis

Most common:

- Time posted
- Video category
- Tag words
- Video title words

Numerical Statistics:

- Likes
- Dislikes
- Views

Relationships:

- Trending vs Publish date
- Likes vs Dislikes

I knew the results of these would be a good baseline for answering my questions and useful when creating the classifier later on.

The task of picking a binary classification algorithm was overwhelming. I had no idea how to pick one so I scanned through a bunch of high level papers on their own analysis of trending YouTube video data. After that, I had a list of algorithms to choose from: Multinomial Naive Bayes, Random Forest, Support Vector, K Neighbors, and Decision Tree. The first algorithm I thought of on my own was the perceptron algorithm because I could see feature sets working nicely in this case. However, I decided to be adventurous and go with decision tree.

Questions

My broader questions are detailed in the Introduction section. Some more specific ones were transformed into my plan for trending video data analysis, such as: what are the most common tag words for trending videos? Can I then use that set of words to optimize my classification algorithm? That then lead to, are there any relationships between attributes? How do I find them? How can I utilize non-numerical attributes or change them into numerical form for feature sets? Talking classifiers, I felt I was working from a near base-zero knowledge point- how can I use sklearn's built in classifiers? How do they work?

Challenges

To combat the survivorship-bias problem, I searched the internet for a regular YouTube video dataset, but failed to find one, probably because it'd be ridiculously large and unspecific. After too much time spent attempting to utilize the YouTube API, I ended up manually creating the youtubevideos.csv file used for the decision tree classifier (which is why it's smaller than preferable).

Technical Approach

The programming section consists of four types of scripts: most common attributes, numerical stats, relationships, and the classifier.

The first one gathers the most common occurrences under each attribute (time posted, category, tag words, title words). First, I prepped the data by gathering the column in question and reformatting the strings into an easier form to use. For the title words, I knew I needed to filter out any 'stop words' first. I was excited to discover the nltk python library, but for some reason it wouldn't work in my pycharm. Luckily, I found the nltk stop list on the internet and was able to create my own in .txt form. After prepping, I gathered the top counts for each attribute and plotted them to a bar graph using matplotlib for a nice visual representation.

The second type focused on the numerical attributes. After gathering the column and converting to a numpy array, I was able to perform statistical analysis' like min, max and average via the statistics library. I wanted to visualize this data as well, but had trouble plotting a 1d array and decided to choose my battles.

The third type analyzed the relationships between the attributes. I picked two sets of comparable attributes- likes/dislikes and publish/trending date. For the relationship between likes/dislikes, I created a scatter plot for visualization and then found the average ratio. For publish vs trending date, I used the datetime library to calculate the time difference between the two dates and plotted that with counts on a bar graph. After those, I wasn't sure how else to analyze the relationships, so I took to google. From there, I discovered the concept of correlation matrix and was able to visualize it with the seaborn library.

The last three types of scripts were not only useful for answering my main questions of "what attributes make a YouTube video trending?" but were gathered in hopes of supporting my classifier. The task of implementing the decision tree was very daunting, even after reading through the sklearn documentation. Thankfully, I found a nice tutorial from datacamp.com that walked me through it. I initially created the tree with a feature set of only numerical attributes: category id, views, likes, dislikes. This only yielded a classifier accuracy of 50%. This is where I got to use my previously collected data from the trending videos to convert time, title words and tags into a numerical value I could use for the feature set:

Time: 2020-05-03T19:00:01.000Z -> 19

Title: String -> # words from top 30 trending video title words

Tags: String -> # words from top 30 trending video tag words

After including these new attributes as feature values, the accuracy rose to 83%.

```
for t in titles:
    gatherTitleWords = t.split(" ")
    l = [word for word in gatherTitleWords if word in trendingTitleWords]
    anotherList.append(len(l))
return anotherList
```

Figure 1. Creating Title Feature

Evaluation Methodology

The input data for this project came from the Trending YouTube Video Statistics in Kaggle. The dataset includes data for each country, but I chose to only analyze the US for simplicity. The entire dataset contains 9139 trending YouTube videos extracted daily.

The output of my data mining task that could be evaluated was the decision tree classifier. By the end of my project, the accuracy calculated was 83%. I compared this to the "good" accuracy of 67% specified in the datacamp.com tutorial, leaving me satisfied with my model.

Results and Discussion

The results of the most common categories for trending videos are shown below.

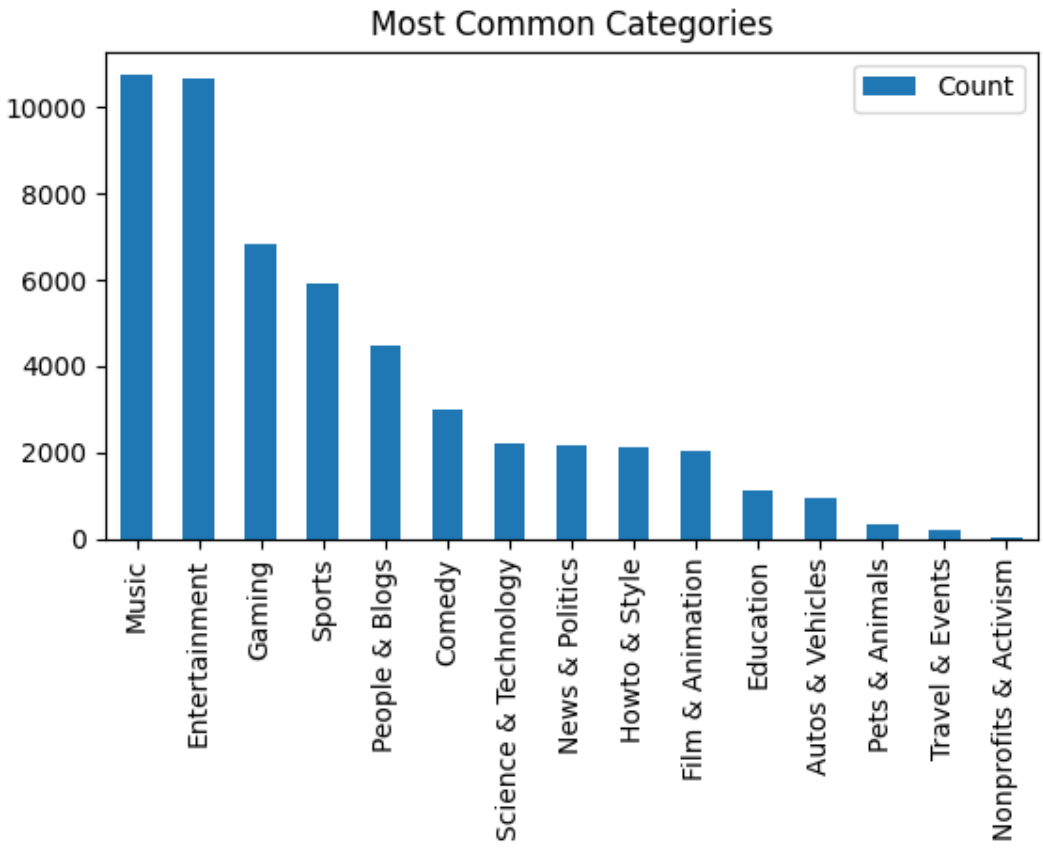

Figure 2. trending video categories

The results of the most common tags for trending videos are shown below.

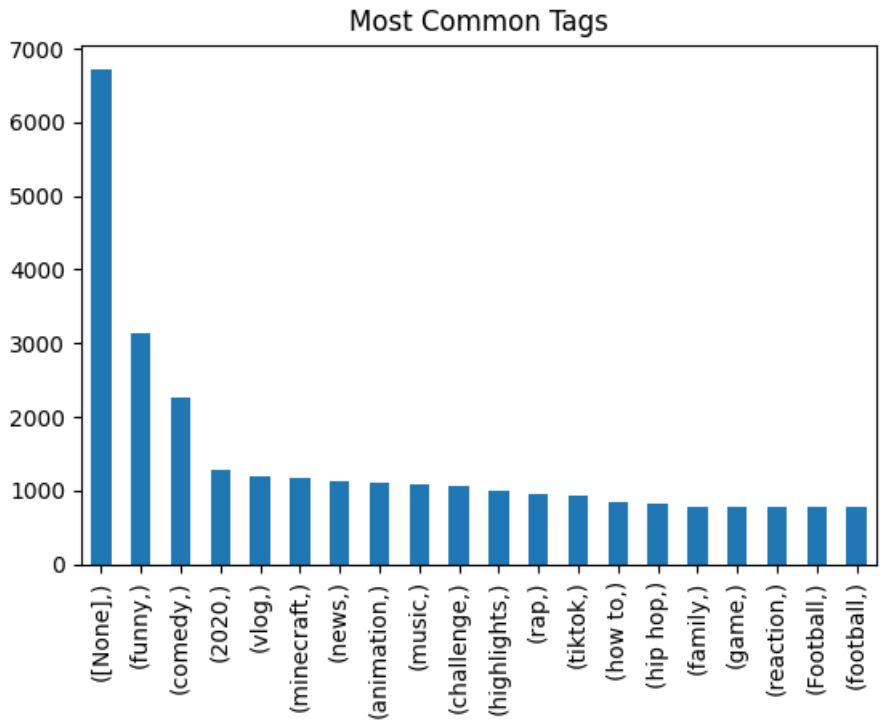

Figure 3. trending video tags

The results of the most common tags for trending videos are shown below. (17:00 is 5:00pm)

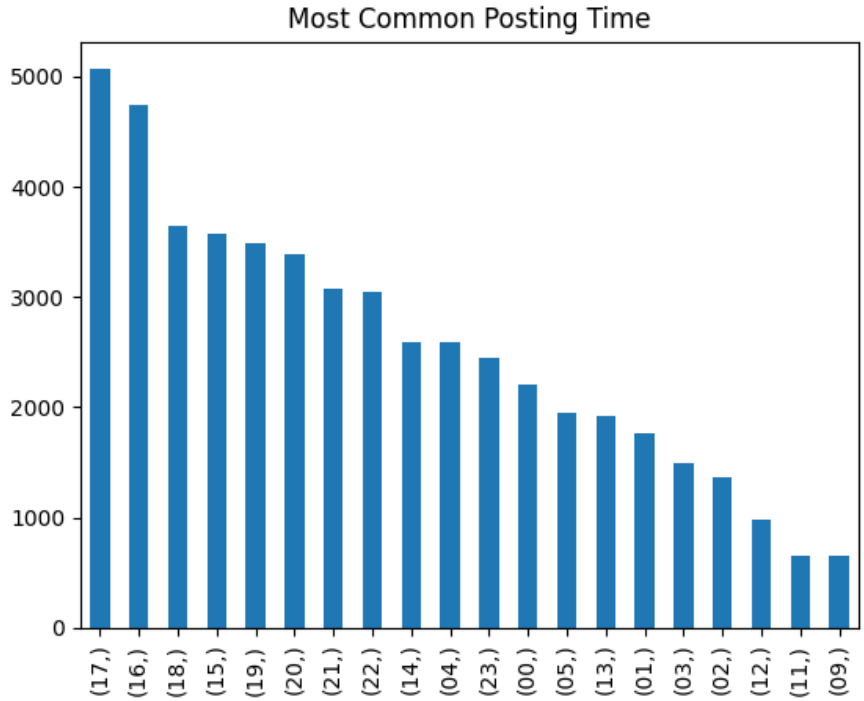

Figure 4. trending video posting time

The results of the most common title words for trending videos are shown below.

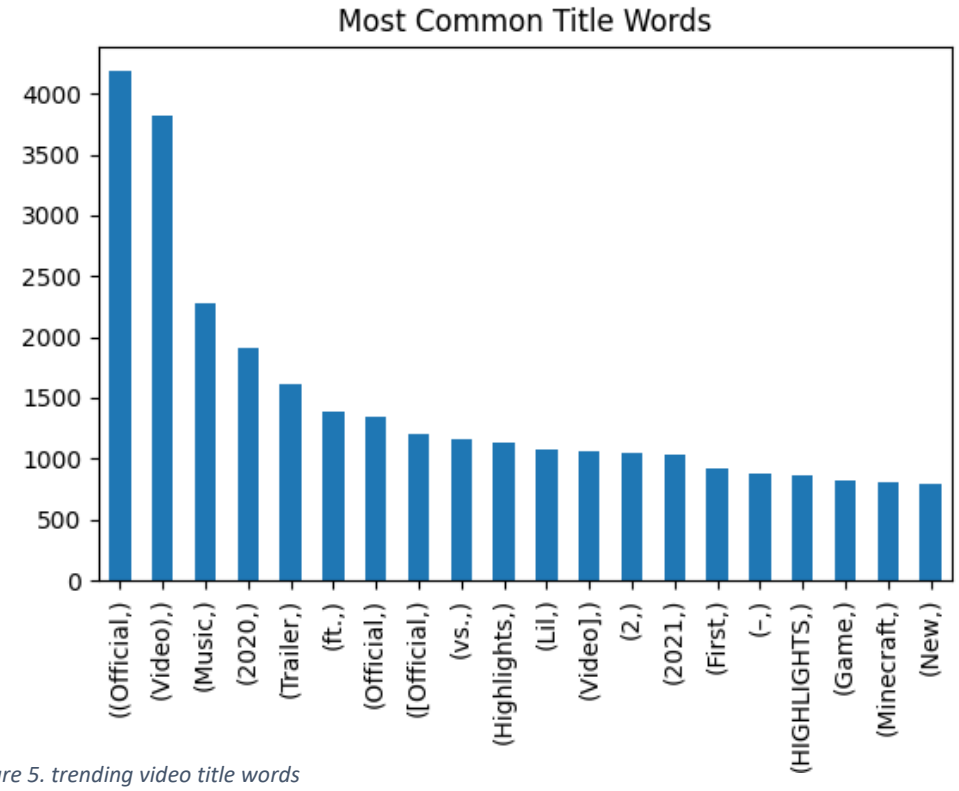

Figure 5. trending video title words

The results of the most common year uploaded for trending videos are shown below. You can see all trending videos have been posted in the last two years.

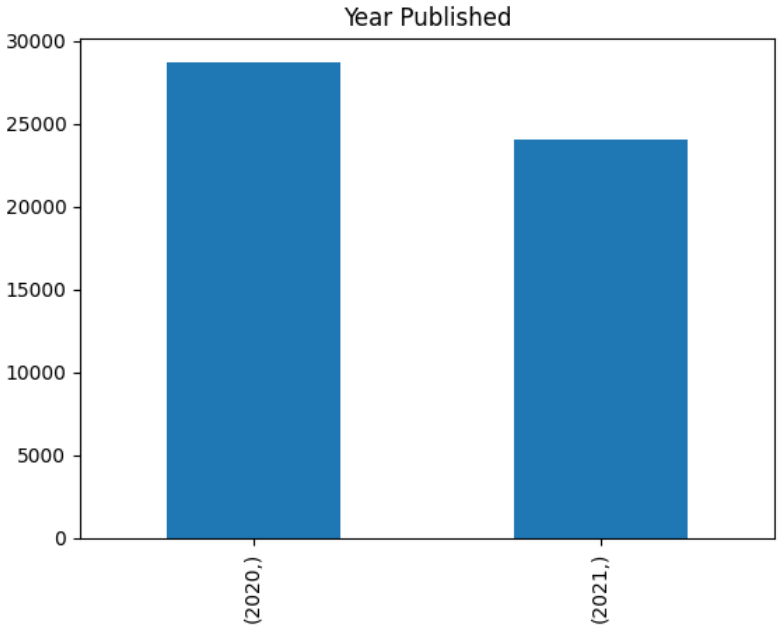

Figure 6. trending video year posted

Statistical analysis of trending video likes.

```
Analysis of likes:
Min:  0
Max:  15735533
Mean:  150471.36701331666
Average:  150471.36701331666
```

Figure 7. trending video likes

Statistical analysis of trending video dislikes.

```
Analysis of dislikes:
Min:  0
Max:  879354
Mean:  3398.888825746813
Average:  3398.888825746813
```

Figure 8. trending video dislikes

Statistical analysis of trending video views.

```
Analysis of views:
Min:  0
Max:  232649205
Mean:  2710626.930347976
Average:  2710626.930347976
```

Figure 9. trending video views

Relationship between likes and dislikes.

```
Average Likes/Dislikes:
78.49998818534769
```

Figure 10. Avg likes/dislikes

Relationship between likes and dislikes (cont.)

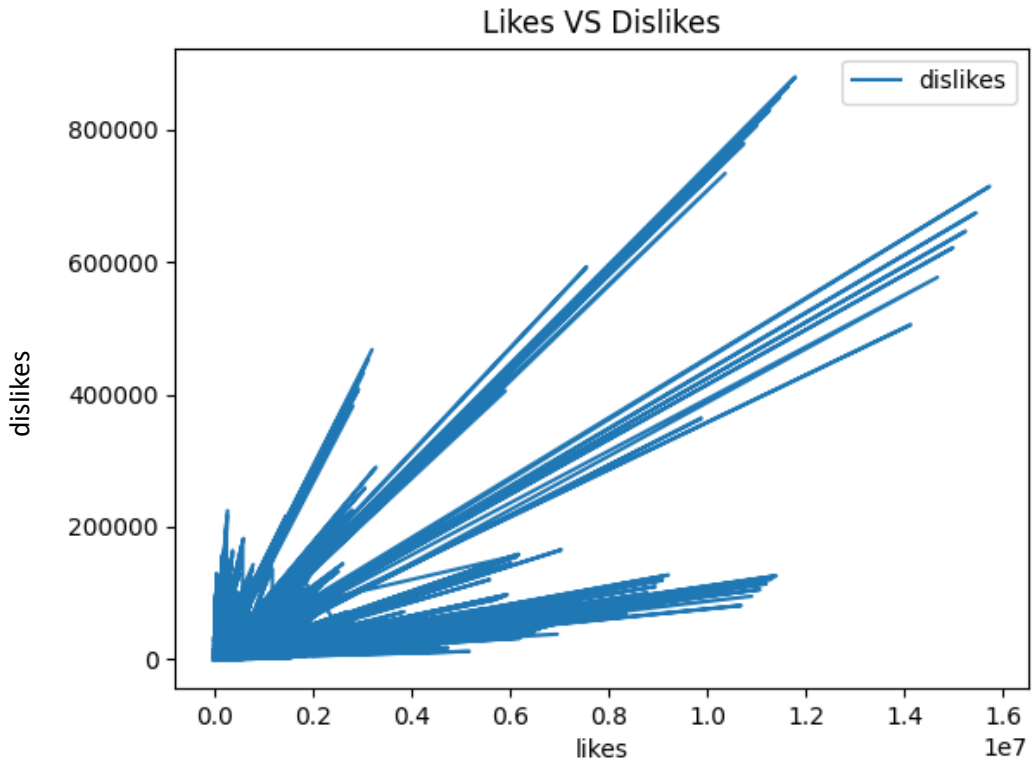

Figure 11. trending video likes vs dislikes

Relationship between published and trending date.

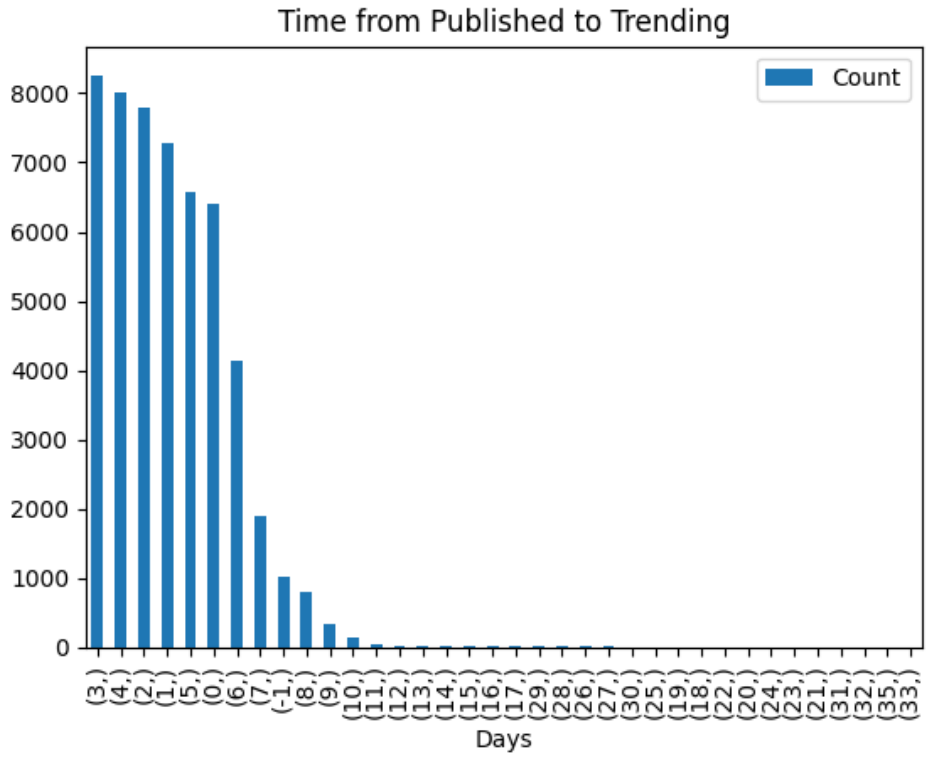

Figure 12. trending video published vs trending date

Correlation matrix between attributes.

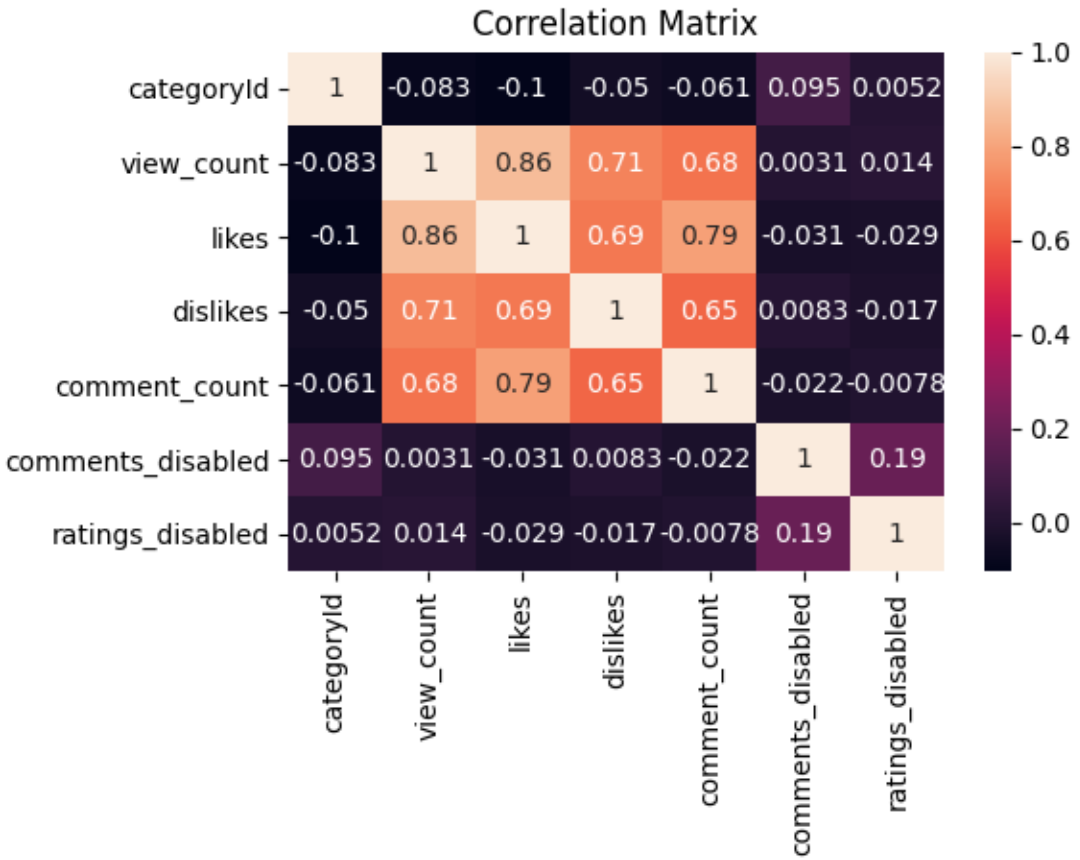

Figure 13. correlation matrix heat map

Decision Tree Visualization w/ features = [‘category_id’, ‘views’, ‘likes’, ‘dislikes’]. Accuracy = 50%.

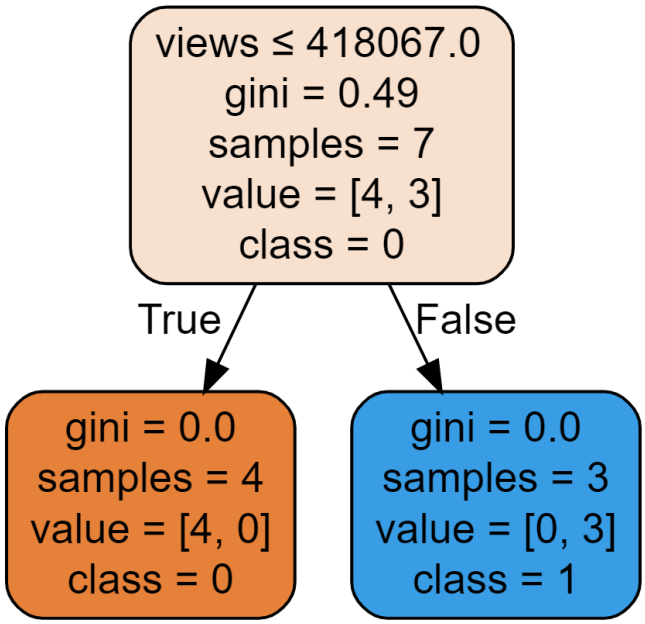

Figure 14. original decision tree

Decision Tree Visualization w/ features = [‘category_id’, ‘views’, ‘likes’, ‘dislikes’, ‘time’, ‘tags’, ‘title’].

Accuracy = 80%

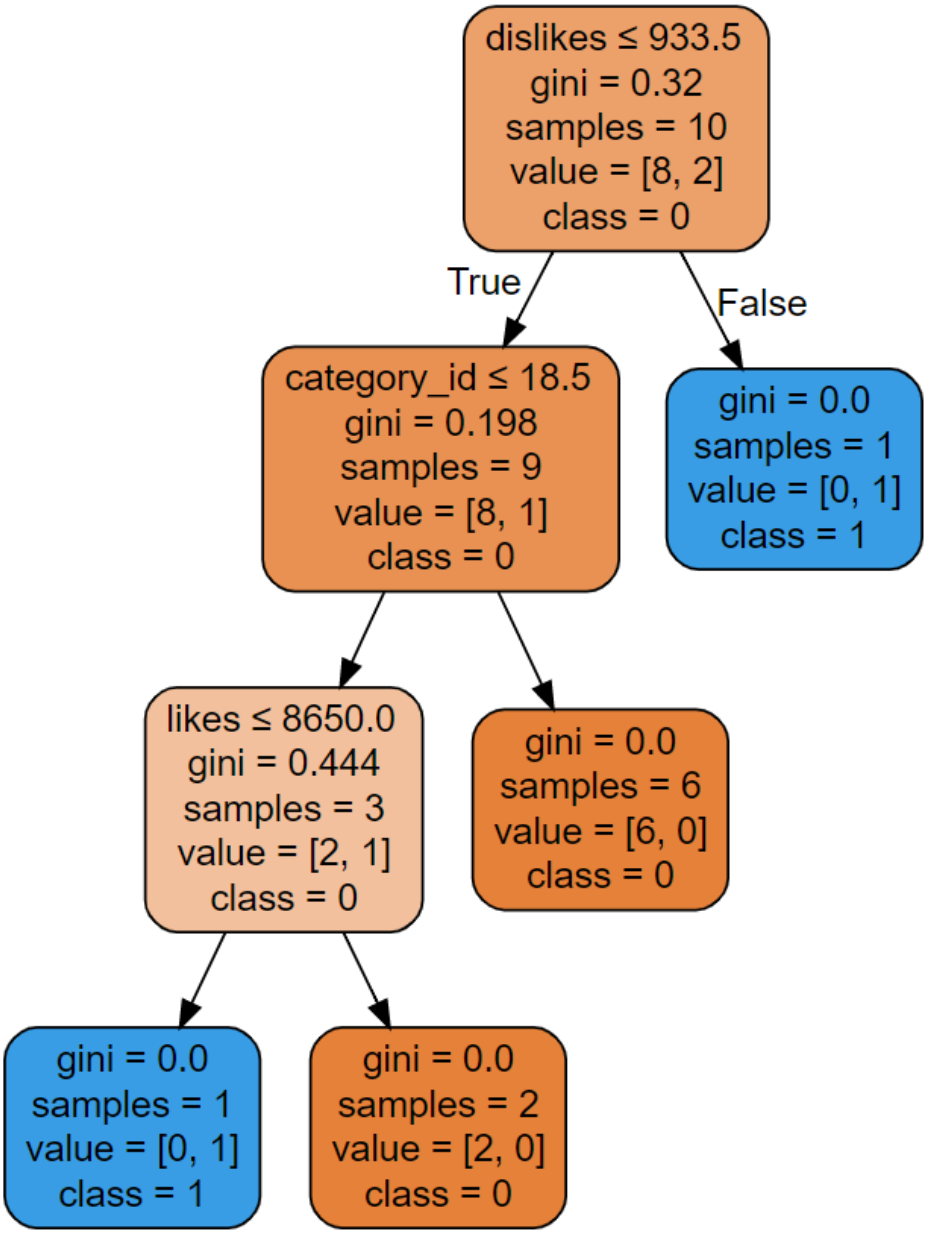

Figure 15. decision tree after new feature creation

What Didn’t Work

Creating visualization plots for the statistical analysis of views, likes and dislikes. My data was in the form of a 1d array which is impossible to plot as is. I recognized I’d have to do more data manipulation in order to get a nice visualization which I ran out of time to do.

Lessons Learned

This was such an open-ended assignment I wasn't sure when to stop. If I had more time, I probably would have kept going and tried out my original idea of using a perceptron classifier. For better or for worse, time determined my project length. There are bits of the project that I wish I had more time to improve, like the statistical analysis plots, or gathering more datapoints for the classifier. I lot of time would have been saved if I could have figured out the YouTube API as I manually gathered the data from non-trending YouTube videos.

I have only had one other open-ended course project in my undergrad career and so it was difficult to formulate my own personal project that would hopefully be smart enough to yield a good grade. As an already anxious person, this project was nerve wrecking especially at the beginning when I hadn't found my direction yet.

I learned how to take an idea, gather lots of broad research on it, then pick a path to continue down.

Acknowledgements

Galarnyk, Michael. "Visualizing Decision Trees with Python (Scikit-Learn, Graphviz, Matplotlib)." Medium, Towards Data Science, 3 Feb. 2021, towardsdatascience.com/visualizing-decision-trees-with-python-scikit-learn-graphviz-matplotlib-1c50b4aa68dc.

Navlani, Avinash. "Decision Tree Classification in Python." DataCamp Community, 28 Dec. 2018, www.datacamp.com/community/tutorials/decision-tree-classification-python.

Qiu, Sky. "Why Study Statistics behind YouTube Trending Videos." Medium, Towards Data Science, 15 Feb. 2020, towardsdatascience.com/why-study-statistics-behind-youtube-trending-videos-231b72c81256.