CIS 531 Semester Project: YouTube Sentiment Analysis

Written by: Nathan Sellers, Charles Swisher
Kansas State University Department of Computer Science
2061 Rathbone Hall
Manhattan, Kansas 66506

Introduction

For this particular project, we set out to learn more about YouTube video data, and the relationships between certain aspects of the data. To do this, our main task would be to build a system that would predict the sentiment of a comments section in a video, with a subtask of creating a system that would additionally predict the number of views that a video would have. YouTube content makers are constantly looking for better ways of producing videos that get positive reactions from viewers, and our project can definitely help with that. There is a lot of data that is associated with a particular YouTube video, so the usage of data science and analytics is very fitting in order to solve this problem.

Related Work

For this project, we took inspiration from a YouTube Data Analysis project prompt given to us in our data science class, however we built off of what was given to us within that prompt. The dataset was limited for what we were wanting to do, so we decided to use a different Kaggle dataset that was produced by Kaggle user Mitchell J https://www.kaggle.com/datasnaek/youtube. The user was able to collect this data from the trending videos section of YouTube using a trending YouTube video data scraper (https://github.com/DataSnaek/Trending-YouTube-Scraper). This particular data set had comments associated with particular videos, which was a big portion of our project. We tried using the dataset that was provided to us in the prompt, but we could not find a worthwhile method to extract comments from YouTube.

Several of the labs and homeworks for the class were used. One of the last assignments that covered sentiment analysis was very helpful to us and we used Spark's pretrained sentiment analysis pipeline. Additionally, several of the assignments that used Skicit learn were very useful to us as well. We used sklearn to build our model.

Lastly, we used an article and guide on random forest for inspiration as well (https://towardsdatascience.com/random-forest-in-python-24d0893d51c0). In the article, they work with weather and build a model to predict temperature. The way that they set up their random forest model was used and we used the way they generated certain figures as well.

Data Set

As stated previously in this write-up, the dataset we chose to use came from Kaggle

user Mitchell J: https://www.kaggle.com/datasnaek/youtube. The dataset was created using Google's YouTube api two years ago. It collected data about the trending videos on the site at the time. It was created for data analytics and machine learning analysis purposes.
This dataset came with two tables that pertained to YouTube video data. The first table was a csv file that contained video data. Useful columns that we used from this table included video_id, views, likes, and dislikes. Other attributes in this table consisted of title, tags, channel_title, category_id, and thumbnail_link. The second table contained data pertaining to comments associated with videos from the first table. The two columns we used from this table were video_id, which helped us link the two tables together, and comment_text. From there, we put the data from these two tables into corresponding PySpark data frames, and cleaned the data, specifically the text columns, to prepare for sentiment analysis. To clean the data, we removed punctuation from text fields, and removed stop words so that they would not be used in the sentiment analysis calculations.
For the stop words, we converted our PySpark dataframes into Pandas dataframes, then applied the stop words list provided by Natural Language Toolkit (NLTK). After all of this was done, our data was ready for sentiment analysis and model training.

Methodology

As stated in the previous section, our data preprocessing consisted of stripping punctuation from text columns (title and comment_text), and removing stop words by converting our PySpark dataframes to Pandas dataframes, then applying the NLTK stop words list, and finally converting back to a PySpark dataframe. From there, our data was ready for sentiment analysis.
For this, we used one of SparkNLP's pretrained pipelines, specifically the analze_sentiment pretrained pipeline. This would allow us to receive whether a given text field was positive or negative. For the comment_text column, we grouped the data by video_id, and made two additional columns to count the number of positive comments, as well as the number of negative comments. Since this data was grouped by video_id, we were then able to join this new dataframe with the video dataframe. After we had all of that data, we created a positive to negative ratio on the new comment column, as well as a like to dislike ratio on the video itself using the like and dislike columns. The positive comment ratio is what we then used in our model.
One challenge we found within this preprocessing step was trying to receive additional data from YouTube. The specific data we wanted was the transcript for each video we were using in our dataset. After lots of experimentation with several different data scraping tools, we could not find a single reliable scraping tool to get us the transcript for each video. The most promising one we found would stop working after we processed roughly one hundred videos. The entire dataset consists of 8000 videos, so this was not a viable method. Instead we decided to start with a smaller set of about 550 videos. After the cleaning the

data as discussed previously, about 400 videos are left.

Evaluation

The random forest model is evaluated on how accurately it can use the five features (views, like count, dislike count, like ratio, comment total, and date) to predict the positive comment ratio. Our project uses the Skicit learn library to perform its evaluation. The data is split into training and testing sets, with the testing set being 25% (100 videos) of the entire set.

The scikit learn random forest regressor is set to 1000 estimators and the model is trained on the training data with 'fit'. The test data is then run through the model with 'predict'. The mean absolute error from the process is calculated by finding the absolute value between the difference of the predictions and the test label data and then using the nump library to calculate the mean. The mean absolute error for the model is 0.07.

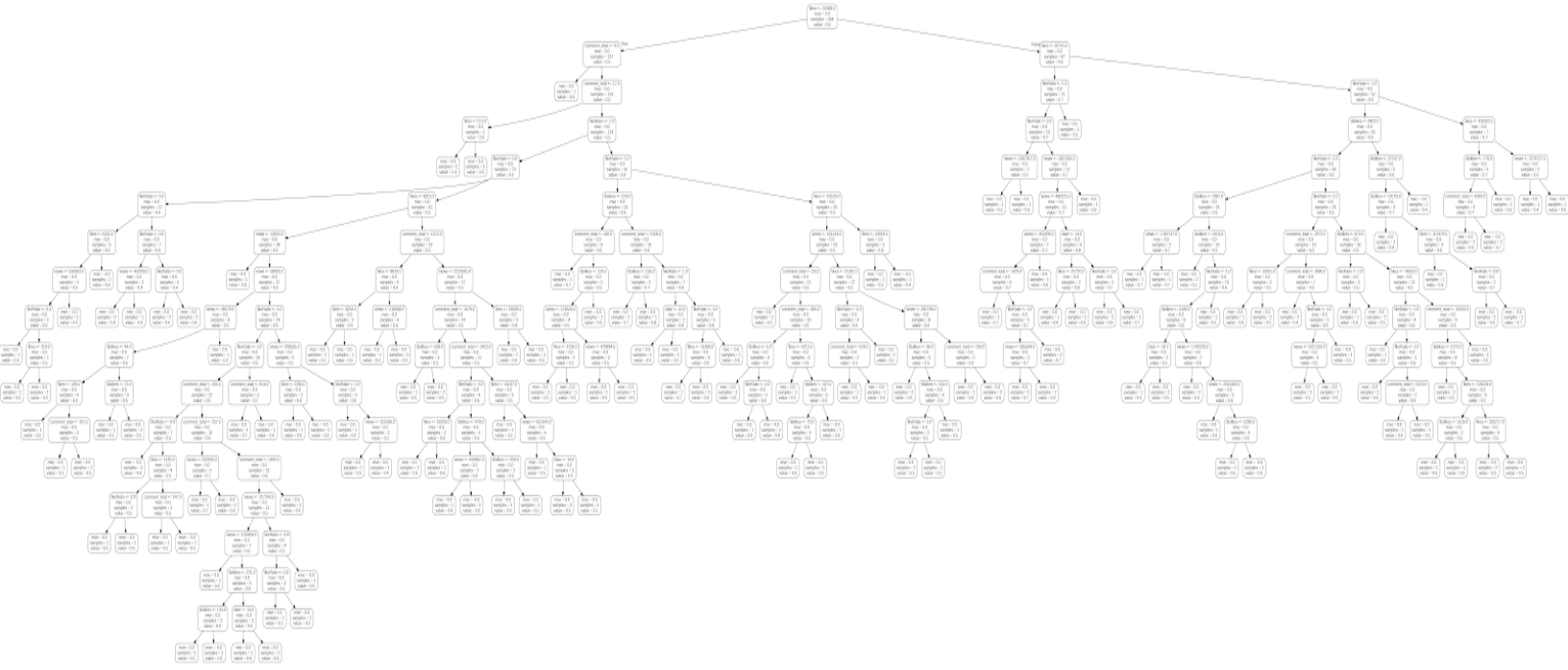
Next, the mean absolute percentage error is calculated by dividing the errors by the test labels and multiplying by 100. The accuracy of the model can then be found by subtracting the mean of the mean absolute percentage error (calculated using the numpy library again) from 100. The accuracy of the model is 88.27%.

Results

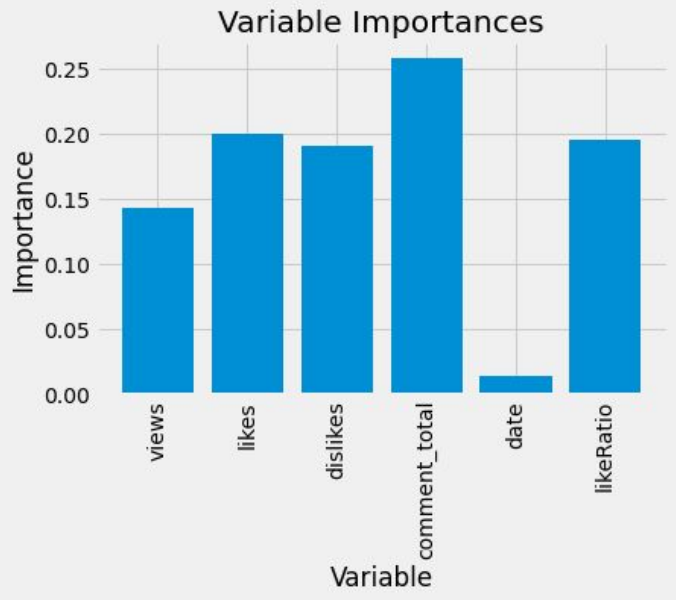
One of the trees from one of the forests of the random forest model is below. The far right side has the fewest layers with 13. The far left side has the most layers with 18 layers. The different features are fairly evenly dispersed throughout the tree indicating that there is no one feature that heavily relied upon. The date feature is used extremely seldomly in the nodes of the tree.

Looking further into the features, the variable importance of each was calculated with the 'feature_importances_' of the random forest model. The results are depicted in the 'Variable Importances' figure. The features, with the exception of date, are all of very similar importance. Comment total, likes, dislikes, likeRatio, views, and date have importance values of 0.26, 0.2, 0.19, 0.19, 0.14, and 0.01 respectively.

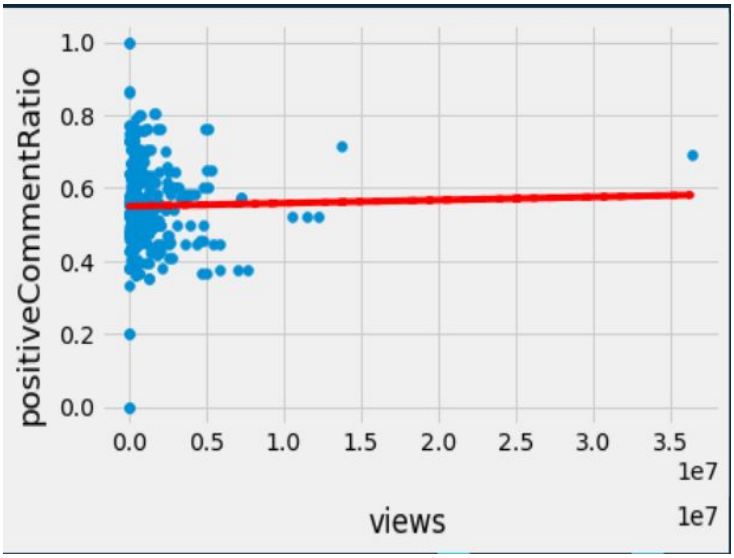
Comment count was the most important feature of the model but not by much. Since all of the features have similar importance,



with the exception of date, we know that the positive comment ratio is not strongly tied to any one of these features. Since the positive comment ratio is meant to function as a measure of how much an audience enjoyed and related with a video, a content creator could use this information to know that having a video with many views or few views, or having a video with many likes or few does not indicate how much those that did watch the video enjoyed it.
While comment count was most important, since its importance is similar to all the other

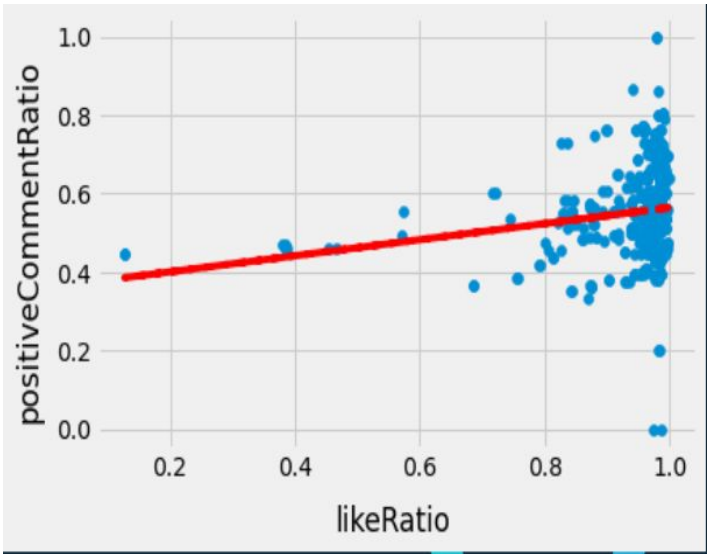


features but date, creating a new model using just comment total or even the two or three highest performing features would not make sense. A new random forest was created without date, since it is essentially useless. Of course, since the date feature already had such little impact on the model, the removal of the feature only increases the accuracy of the model by 0.35% to 88.62%.
The videos come from a three day span, which is a relatively small amount of time. Something that might be interesting to try to see if the date feature can be more influential in the model would be to use a larger range of dates, for instance a month or a year. Similarly, adding features that tracks what day of the week or what month a video is released might also be of interest to see if videos have a different reception based on the day of the week or the season.
Going into the project, we thought that the positive comment ratio and the views that a video had might be related and could then play a significant factor in determining the reception that a video received. The figure below shows the amount of views a video has against the positive comment ratio.



As shown in the figure, there appears to be no strong relation between the positive comment ratio and the view count. Most of the videos have a positive comment ratio between 0.4 and 0.8 and the majority of the videos have less than 250,000 views. There are many videos with similar view counts and the positive comment ratio of those videos are not similar.
Another prediction that we made about the data is that the positive comment ratio and

the like ratio on a video would be correlated as well. The figure above shows the like ratio against the positive comment ratio. Just like the previous prediction, the figure shows no strong correlation between the like ratio and positive comment ratio. Something we found surprising about the data is that is how the like ratio for most of the videos was above 0.9. It seems that the majority of people rate a video with a like if they rate it at all and that rating a video with a dislike is much more rare.



While there are a few outliers, even videos with low positive comment sentiment ratios, around 0.4, have a very high like ratio. This means that even on videos where the viewers response to the video contained more negative language, it did not necessarily mean that the like ratio would be reduced.

Another random model was built as well, this time to predict the view count a video has. The features included are likes, dislikes, like ratio, comment total, positive comment count, negative comment count, positive comment ratio, and date. The random forest model has the same parameters as the previous model for predicting the positive comment ratio.

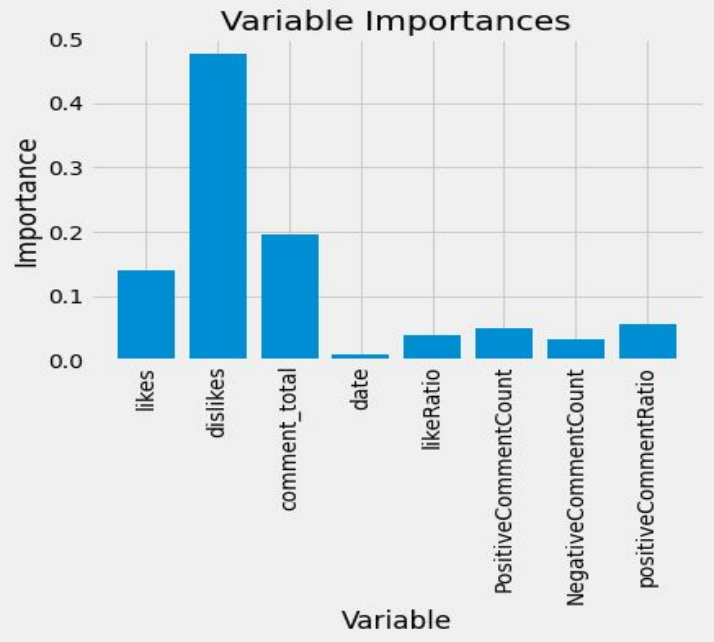
One difference that the model has is that the videos used all had less than 500,000 views. When the model was built using all videos in the set, the accuracy was around 40%. This is likely due to some of the outliers in the set. Some videos have millions of views with one video having over 3,500,000 views while most of the videos have less than 500,000 views. The view count was selected to eliminate the outliers that were skewing the data while also trying to maintain as many videos in the set as possible. The mean absolute error was 44,495 views and the accuracy of the model was 65.43%. By eliminating some of the outliers, the accuracy of the model increased by about 25%.

The variable importances were calculated for this model as well. The three most important features are dislikes, comment total, and likes with importances of 0.48, 0.20, and 0.14.

Dislikes is clearly the most important feature when it comes to predicting the view count of videos. We think there are two likely reasons that this is the case, both involving the YouTube algorithm since when browsing for videos all that is visible to a user is the length of the video, the title, and the thumbnail. First, the algorithm could be deciding that since the video has many dislikes, it should not be served to users. The other thought is that the algorithm specifically serves videos with large amounts of dislikes because of their controversial nature, hoping that users will watch the video and want to share it with

their friends. Either way, we found the high variable importance of dislikes to be surprising.

While the dislike importance may be surprising, the fact that all three of the most important features had to do with audience engagement and participation was not. We



interpreted the high importance of the comment count feature to mean that the model had correctly identified that if users are taking the time to write a comment and are engaging with the video that others would likely want to watch the video as well. Similarly, user ratings of a video are shown to play an important part in the view count of video. Since the like ratio did not have a high importance, it seems that the view count of a video is not tied to how much a video is liked versus disliked, but instead if users decide to rate the video at all.

Attempting to build another model using only the most important features resulted in the accuracy of the model falling several percentage points.

Conclusion

The project uses many of the data science concepts that we have learned about in class. We read in and clean our data using mainly pyspark but sometimes pandas to create dataframes that we query and process to clean. The majority of our time, just as it was said in class, was spent cleaning the data and trying to get it into a usable format. We were really hoping to include the transcripts of the videos we analysed in some way, like looking at the sentiment of the sentences in the video. Unfortunately, the tool we were using to read in the transcripts broke and so we had to pivot the project to not include that data.

Lastly we built two random forest models, one that very accurately predicts the viewer reaction to a video using the positive comment ratio and one that predicts the view count of a video. Interestingly, all of the features used to predict the comment positive ratio are used fairly evenly while the dislikes feature importance was nearly 50%.

Future work could include using a larger data set to look at a longer stretch of time, for instance months or years. We could also add more features to the data set like video length or data about the transcript, like the sentiment of its sentences or the topics discussed. Lastly, and perhaps most differently, we could compare performance stats of different channels such as view count and audience engagement statistics to attempt to understand what makes a successful channel. This work could then lead to information on how to run a profitable YouTube channel or how

YouTube could help their content creators and, in turn, themselves.

References

https://classes.ischool.syr.edu/ist718/content/unit09/lab-sentiment_analysis/#data-science-pipeline-for-estimating-sentiments

https://www.kaggle.com/datasnaek/youtube-new?select=USvideos.csv
https://pypi.org/project/youtube-transcript-api/

https://sites.temple.edu/tudsc/2019/04/03/computational-text-analysis-of-youtube-video-transcripts/

https://github.com/philbot9/youtube-comment-scraper

https://github.com/jessecordeiro/youtube-trending-videos-scraper

https://github.com/DataSnaek/Trending-YouTube-Scraper

https://tools.digitalmethods.net/netvizz/youtube/index.php

https://towardsdatascience.com/sentiment-analysis-with-pyspark-bc8e83f80c35

https://towardsdatascience.com/random-forest-in-python-24d0893d51c0