Final Project Report: Predicting YouTube Trending Duration

Presented by William Ward
Brown University
December 1st, 2019
https://github.com/wward97/data1030project

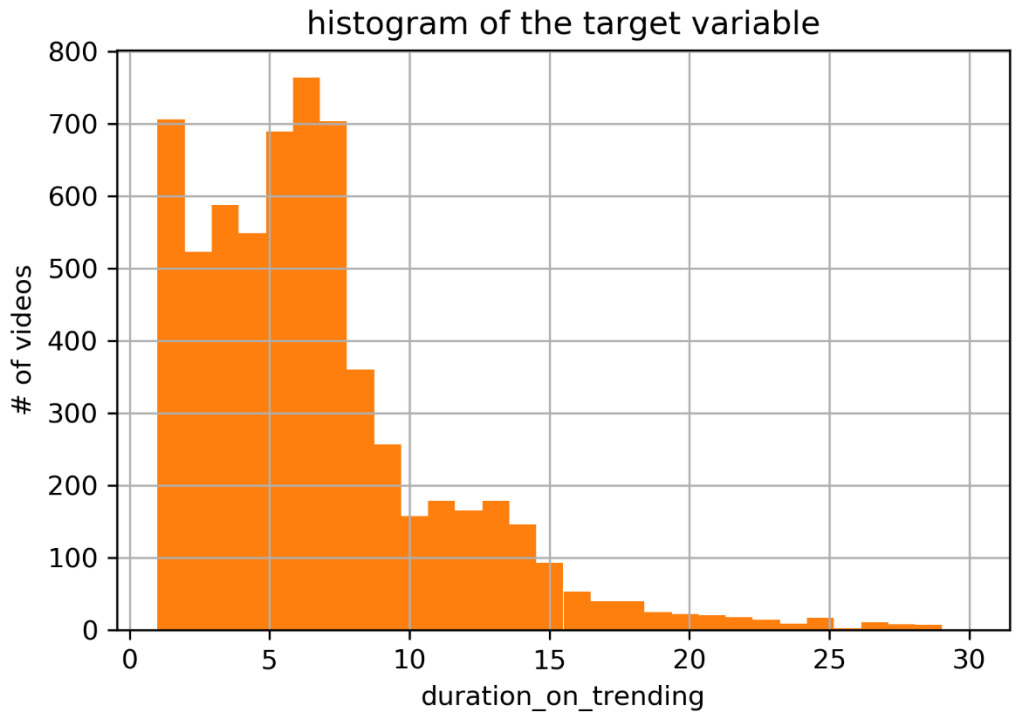
Introduction:

My project focuses on predicting how a long a video will be trending on YouTube given its initial readings when it first goes trending. The motivation behind this is as YouTube becomes more and more of an influential platform, performers and content creators will start really wanted to be able to predict how well a video does. This research is mainly directed towards already influential creators however, since it relies on the video reaching trending to begin with.

Once this happens my model hopes to be able to predict how long it will be there based on the likes, dislikes, views, comment count, and category the video falls into. My dataset comes from Kaggle. It included a csv of 40 thousand YouTube videos over about a 2-year period. Many of the videos were repeats. There were only 6153 unique video ids of the 40000 total videos that appeared on trending. This means on average a video appeared for around 6 days.

- The features included are:
    - trending date
    - title
    - channel
    - tags
    - categories
    - likes
    - dislikes
    - views
    - comment count
    - description

Below is a histogram of my target variable.

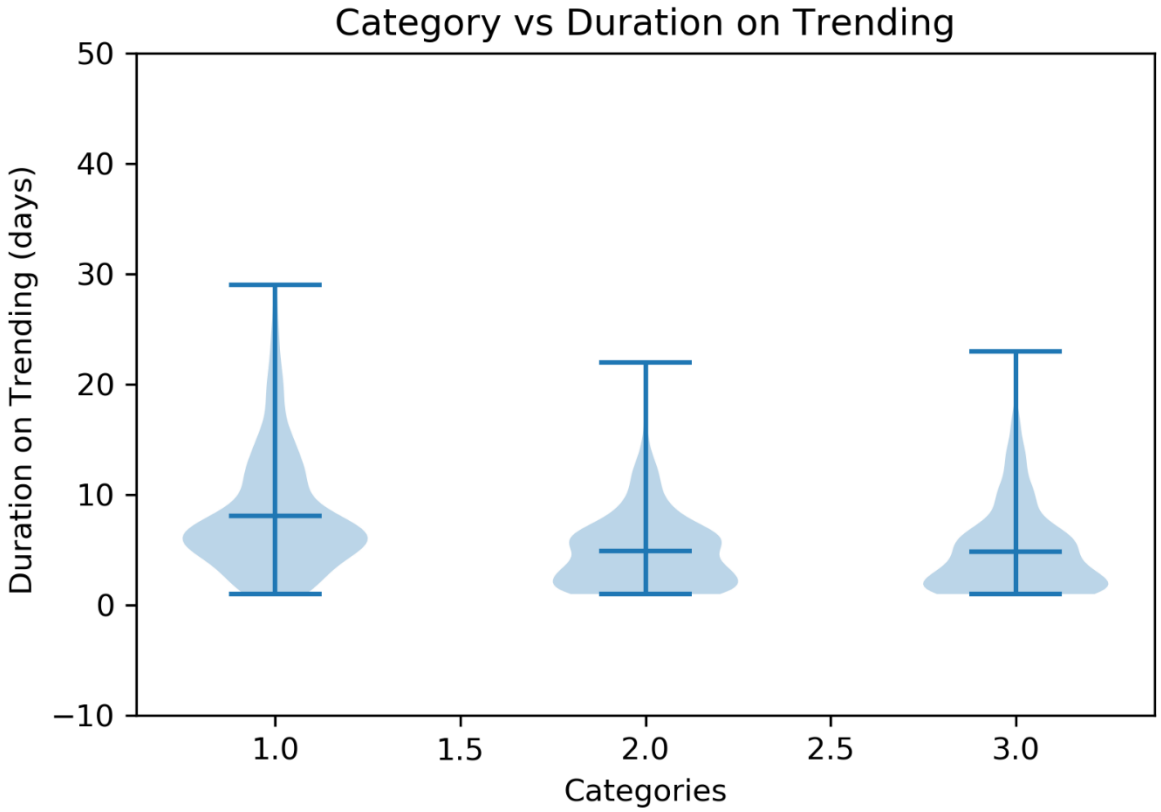


EDA:

When I first started the project I was working with a different target variable. My EDA then however was still similar to my new EDA (target variable is duration on trending). I looked at the distribution of my data set. Found the average duration on trending was 6.4 days with a standard deviation of 4.6. This alone shows that there is quite a big spread. The standard deviation is more than 2/3^{rds} of the mean!

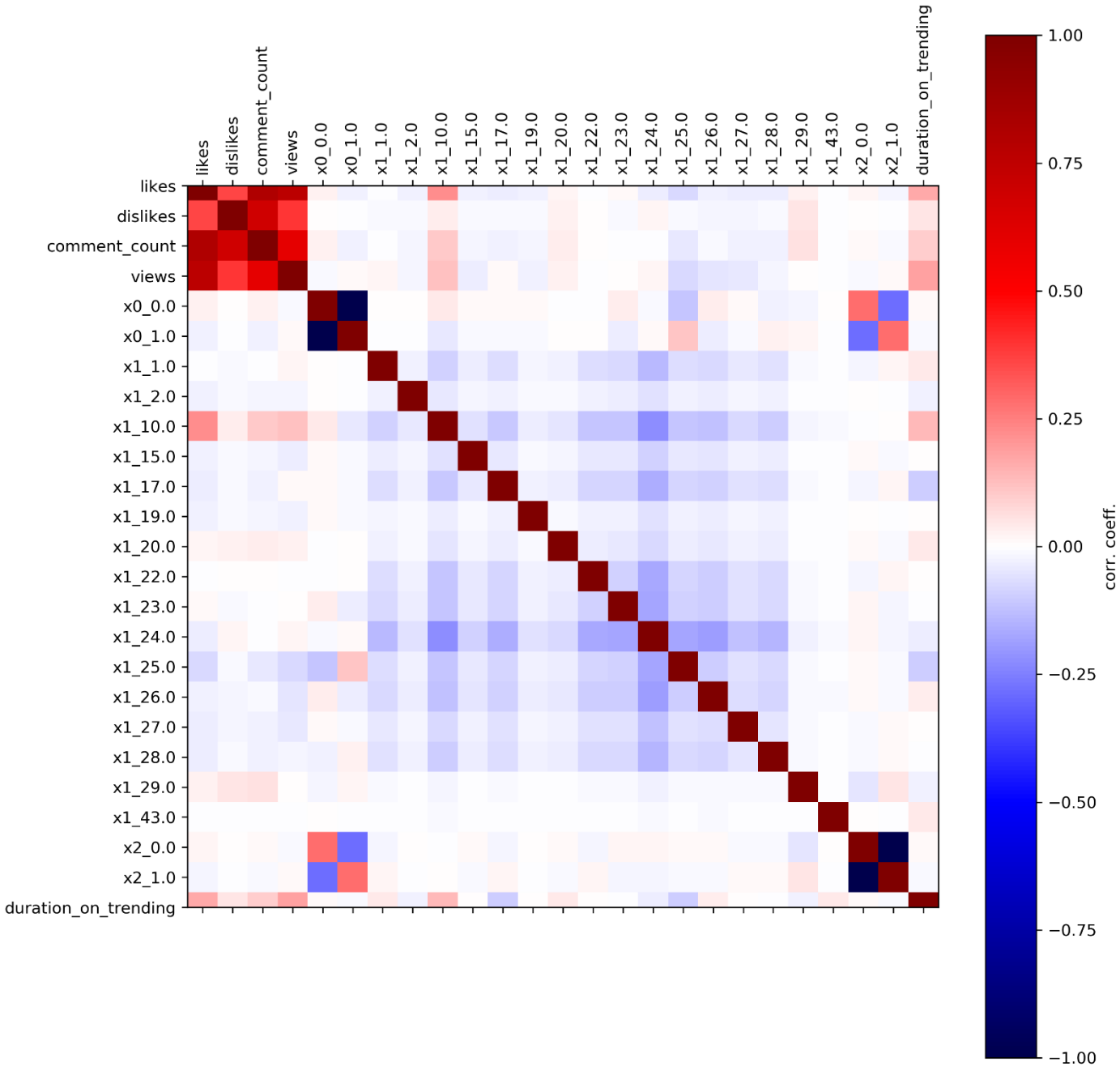
I then tried to get a sense of important features by using f regression on my data. I ended up finding that views, likes and the category ‘music’ were the most correlated features. This is off to a promising start, since the first two make a lot of sense. YouTube wants to promote successful videos, and nothing measures success more clearly than the number of likes and the number of views. Something to keep in mind is that I feel these numbers are closely correlated. I decided to

show these three relationships with a violin chart:



As we can see above the relationship between the first feature (Views) and the duration on trending looks very clear. Feature 2 and 3, likes and music respectively, have a less obvious relationship with duration on trending. It seems that high likes *can* mean a decent duration on trending, but it is not guaranteed. We can now take a broader look at the correlations between all

our features and our target variable by looking at this scatter matrix:



As we can see likes, dislikes, comment count and views are all highly correlated amongst themselves and are decently correlated with duration on trending. As the f regression showed views and likes are the most correlated of those. The very dark correlations are usually features that are forced binary. In particular x2_0 and x2_1 and x0_0 and x0_1 are whether comments are allowed and whether likes and dislikes are allowed. It makes sense then that between the 2 sets

they are also correlated, if you blocked comments you are more likely to have blocked likes and dislikes.

Method:

The data processing was not horrible. My biggest challenge was deciding how to create my target variable. I originally was thinking of making my target variable consecutive days on trending, this way, if a video dropped off trending then went back on I could use that as a new data point. However, I then realized this would be very hard to achieve and would not add many more data points than if I just went for total cumulative time on trending. As a result, this is what I went for.

To create this I used pandas' .unique() on the video id column. I went with pandas' method because it maintained order and that was very important. I then created a for loop through my list of ids' and searched for any row that contained that video id. I then took the length of the remaining dataframe and was left with the number of times each id appeared in the df (dataframe). I then removed every instance of a video but very first. I also preprocessed the columns I would use as my features. I used standard scaler on: ['likes','dislikes', 'comment_count','views'] and One Hot Encoder on ['comments_disabled' , 'category_id','ratings_disabled'].

I decided to train a couple Machine Learning models. In total I tried 5 different models. I did SVR, Random Forest, Logistic Regression Lasso, Logistic Regression Ridge and Linear Regression.

For each of the 5 I used K fold. This was to account for uncertainty in splitting the test and train sets and ensure that there is no accidental bias. Since my problem was a regression problem I used R2 as an evaluation metric. This felt better than MSE because MSE alone did not shed much light on how good the model was. Using R2 I was able to decide which models improved the baseline and which did not. For non-deterministic models I locked the random seed and took the standard deviation of the results every time as an error bar.

For each model I used a grid search to determine the best combination of parameters. I will list each model and list the parameters I tested and the parameter range I provided I will then provide the best parameters in the results:

- SVR
  - C, between 10^-5 and 10^4, 10 samples
  - Gamma, between 10^-5 and 10^4, 10 samples
- Random Forest Regressor
  - Min sample split between 2 and 20, 4 samples
  - Max depth between 1 and 30, 5 samples
- Lasso
  - C, between 10^-5 and 10^4 , 10 samples
- Ridge

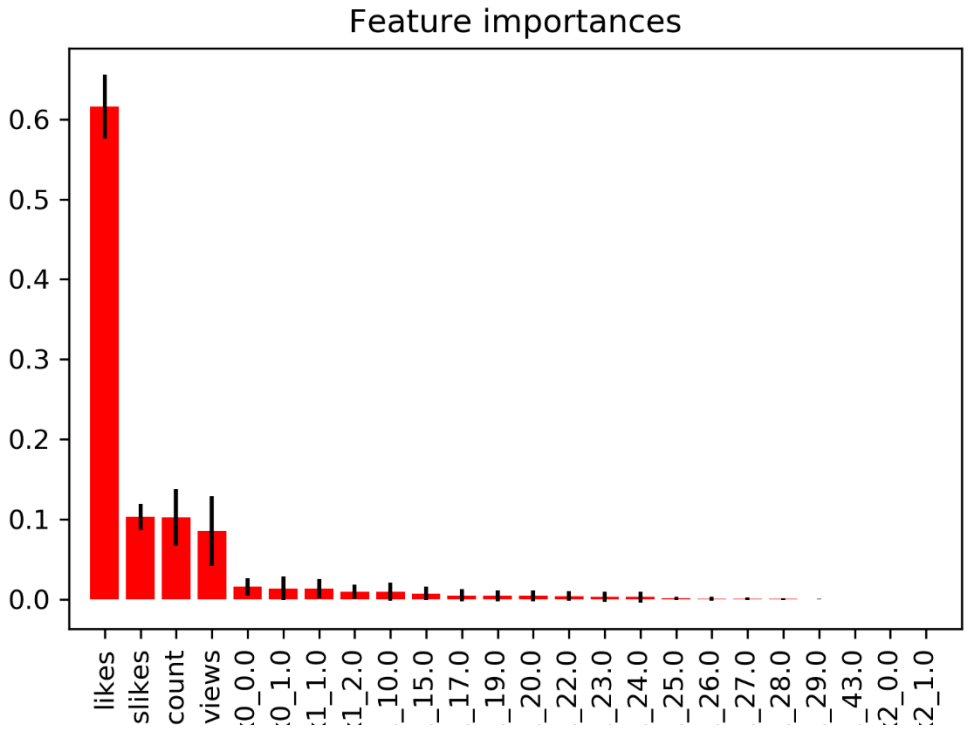
        - C, between 10^-5 and 10^4, 10 samples
    - Linear regression

Results:

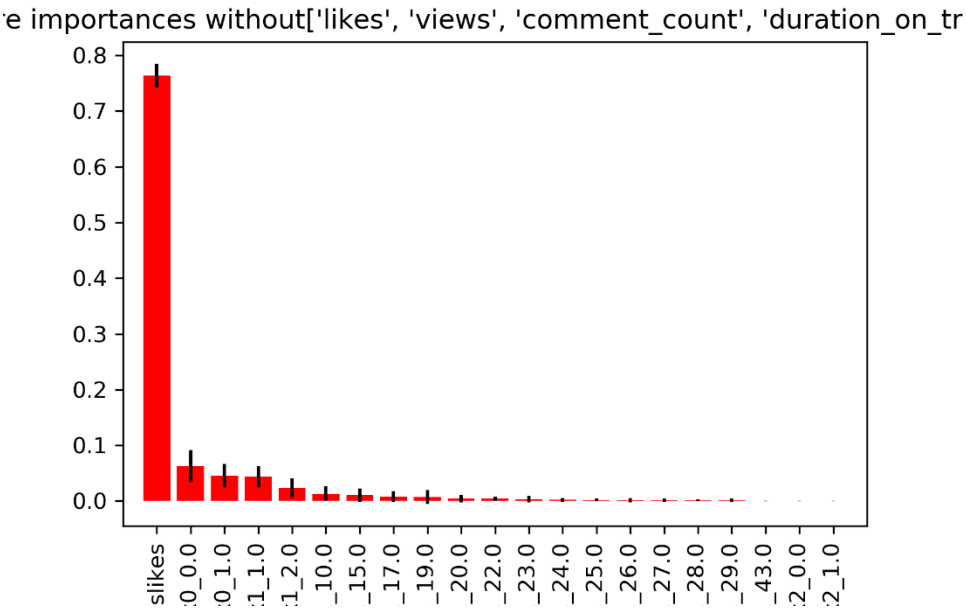
After testing all the models, the best performer by a decent margin was Random Forest Regression. However, this is not saying much. The highest R2 I was able to reach was 0.13 +- 0.01. This is not amazing. This was achieved with a depth of 6 and min sample split of 18. This is about 10 standard deviations above the baseline. The next best performer was SVM clocking in at about 0.06, which is barely better than the baseline. This had an error of about 0.01 and this was about 6 standard deviations above the baseline this was achieved with C of 10^-1 and Gamma = 56. An interesting note was that every best was at this C and Gamma. Next, we have Linear regression with 0.059 +- 0.015.

Now comes both logistic regressions with lasso at -0.16 and a variance of 0.46 which is very high, and Ridge at -0.009 +- 0.006. Clearly this was just barely worse than average. Bests were at C = 10^-2 and C = 10^-4 respectively.

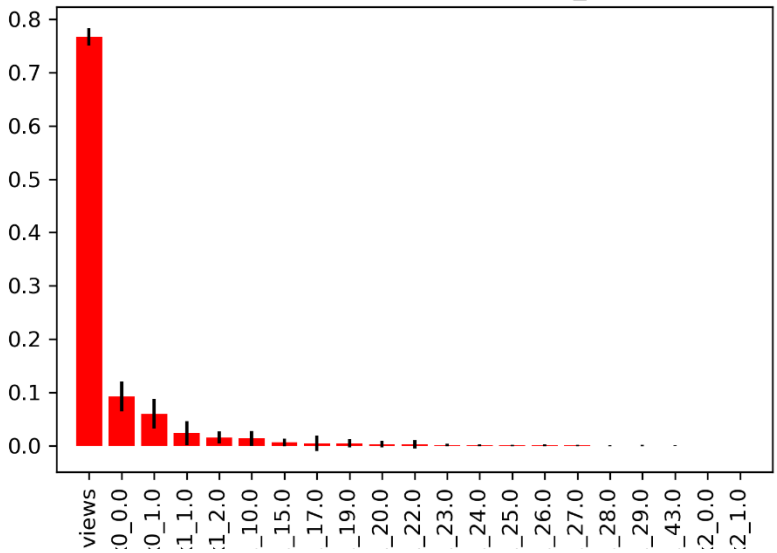
After these lackluster performances I decided to focus solely on Random Forest since it preformed the best of the lot. I wanted to get an idea of how impactful the different features were. To do this I used Random Forests' .feature_importances_ function. This uses the "gini importance", this is : "the total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble." (https://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined). When I originally did this, I got 'likes' completely dominating the other features:



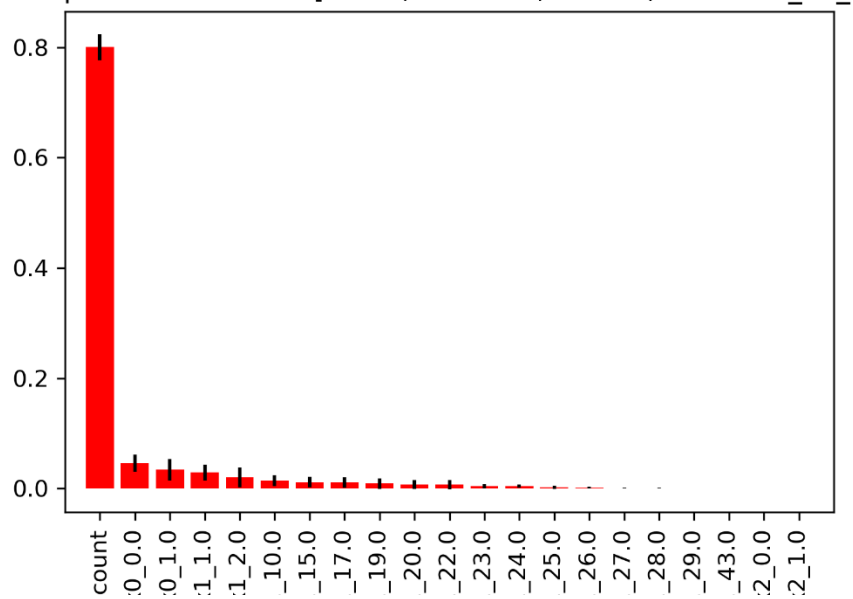
I was worried that because likes, dislikes, comment count, and views were all super correlated that it would skew the results. So, I decided to test the feature importance of each in a mix where the other three weren't there. These were the results:

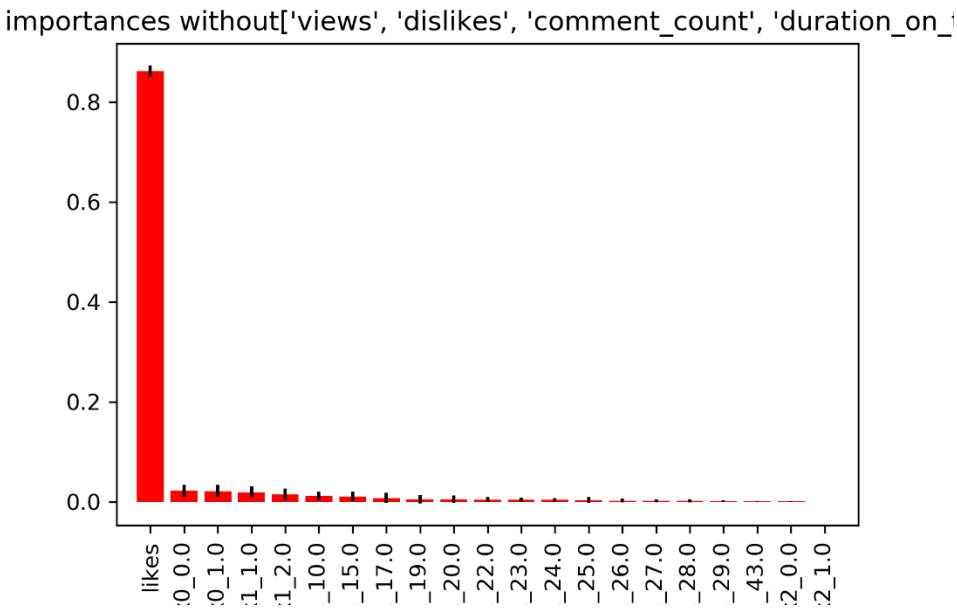


e importances without['likes', 'dislikes', 'comment_count', 'duration_on_t



eature importances without['likes', 'dislikes', 'views', 'duration_on_trendi





As we can see the most important was still likes, however the other three are still way more important than the category of content you create.

From this we can deduce that, with the features I currently have it is hard to come up with an easy formula to get on trending. It seems like the category of videos you create are not as impactful as either: having a strong following that can provide you with views and likes, or creating a video that a lot of people like to watch and engage with via comments and likes.

Outlook:

I think there are a lot of really cool things you can do with this kind of data. For one if I were able to separate and clean out tags and use those as more features it might really reveal some interesting things. Tags would have given me a lot more data points and especially if I were able to clean tags (currently just strings) to make it so I could identify groups of tags as similar and turn them into categories, then I feel I probably could have made a better model.

I think that you can probably also find a way to crosscheck current events (or current events for the time) and tags for videos on trending and find that videos who cover current events are more likely to stay on trending. It would be interesting to see if the duration on trending is heavily tied to the duration of the current event.

Other interesting things would be to instead of figuring out how long a video is on trending, how a video could get on trending in the first place. This might have more large reaching affects for the community since being on trending is excellent exposure for smaller channels. This would require scrapping YouTube however and keeping track of which of the videos you scrape end up on trending.

References:

https://www.kaggle.com/datasnaek/youtube-new#USvideos.csv

https://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined