Author: Brandon Yuong

Virality Predictor

○ What features did you consider?

The linear equation for Virality provided by internal analysis hinted that I would probably be looking at a regression problem. There might have been several ways to approach the data, but to start, I pivoted the data from user interactions to produce columns related to the variables (“LIKE”, “COMMENT”, etc.) for the Virality linear equation. For the most part, the user interactions data looked like a time series of single events. This would not work for our regression, which prompted me to pivot for new columns.

I noticed that there was no column for the target, Virality. Therefore, I generated that column using the provided linear equation and the new reformed features for the independent variables. This column would later be used as the target for the models. Since the two default data sets were both related, I also joined them by “contentId”.

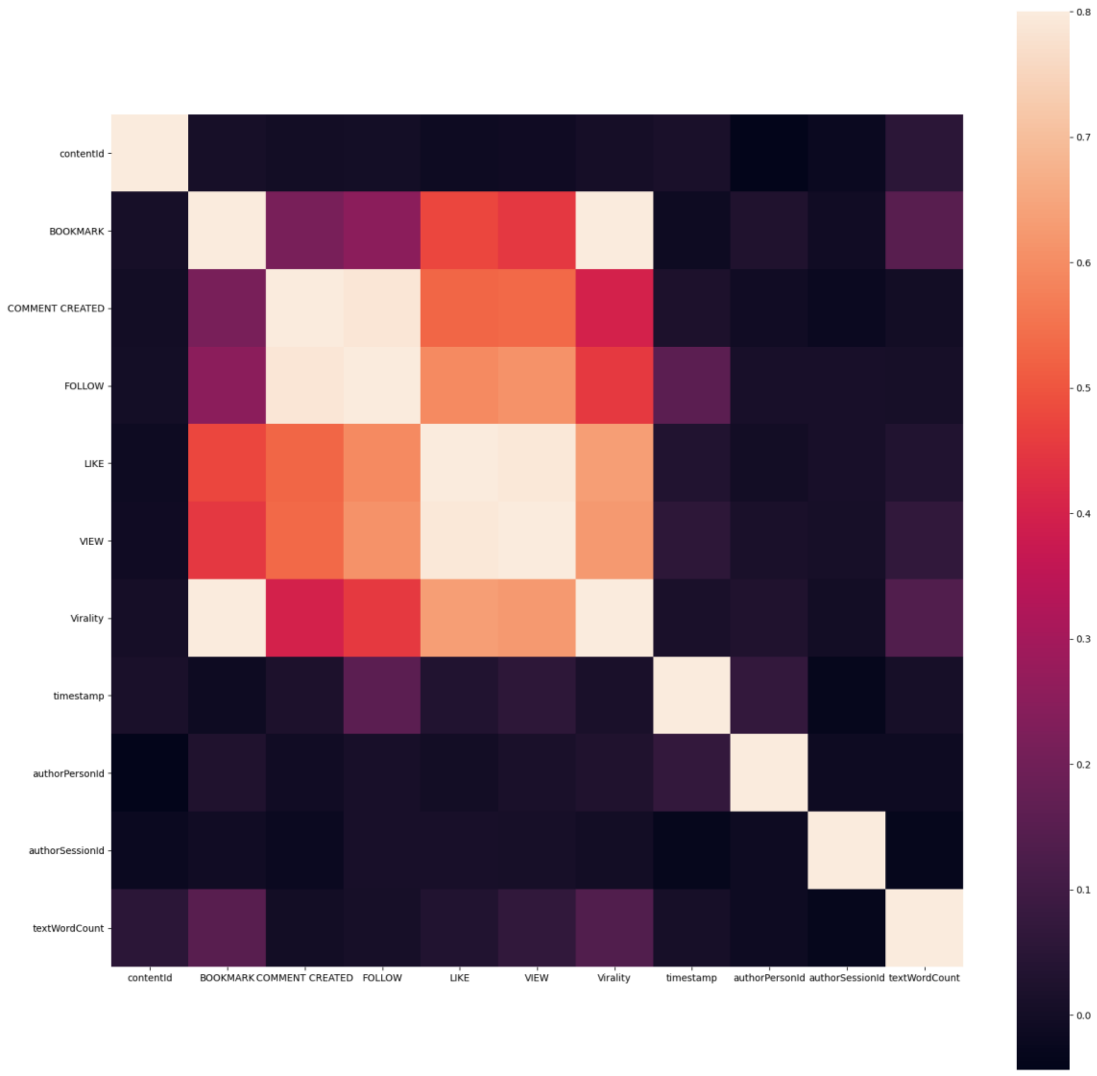
There were missing and null data cells, so I decided to exclude these features because they do not paint a complete picture. There is a risk that they would foul the model as well.

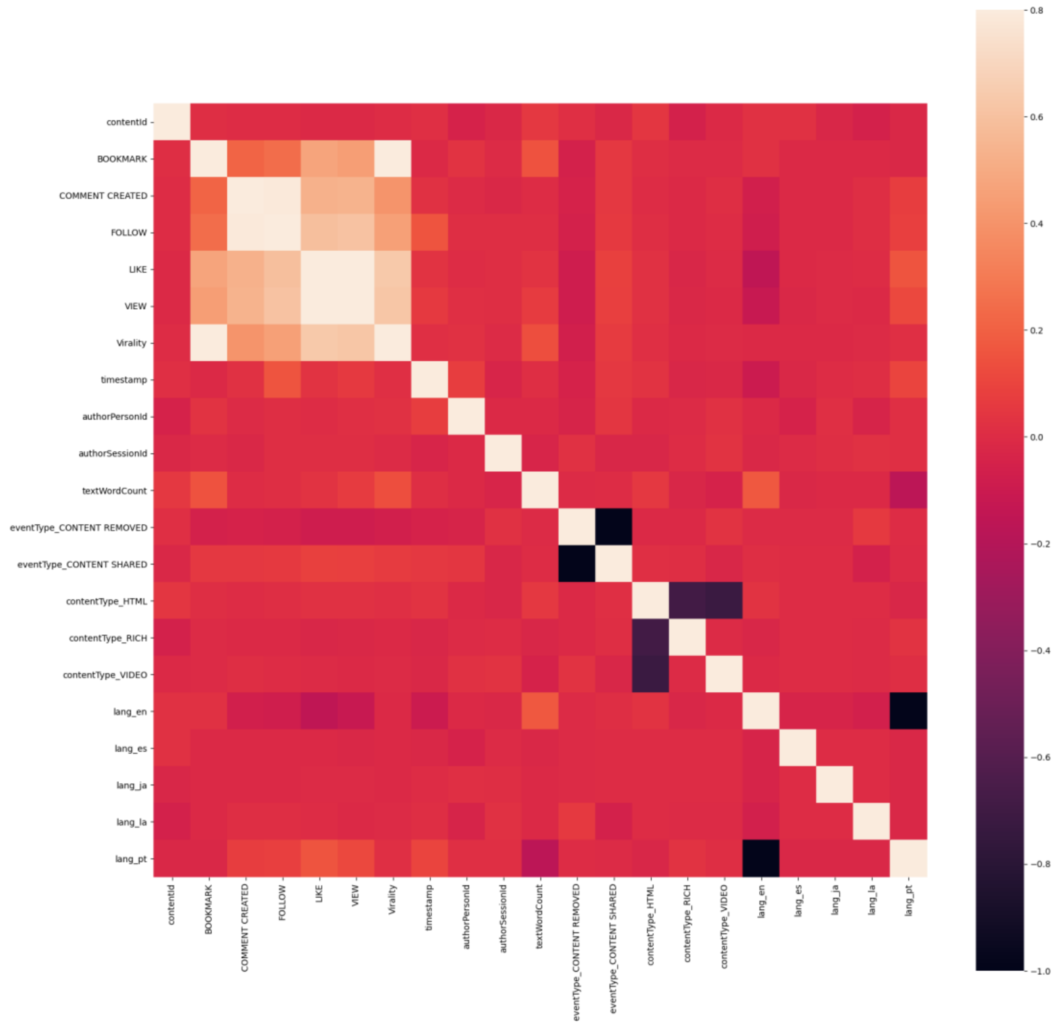
Another big issue is the ‘text’ feature, which contains large blocks of text. This is potentially a separate and significant natural language processing project. We are simply not going to be able to build a model that will efficiently use this feature, given the time frame. The size and variance of it would mostly break any simple models. I opted to transform this feature into a word count instead, so we have numbers for computing.

For similar reasons, the ‘title’ and ‘url’ features do not fit into our regression approach and are not good predictors for viral content. We would need to use NLP to make these features useful to the model, since there is no simple way to transform them into numbers that provide signals. Let us look at this another way. Would a new article with a similar title and URL to a viral article become a viral article? Maybe, but intuitively, probably not. People are not excited by topics that have already trended in the past.

There were a few categorial features. I one-hot encoded these so the models would have numerical data to work with.

I then produced a few correlation heat maps:


contentId
BOOKMARK
COMMENT CREATED
FOLLOW
LIKE
VIEW
Virality
timestamp
authorPersonId
authorSessionId
textWordCount
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0



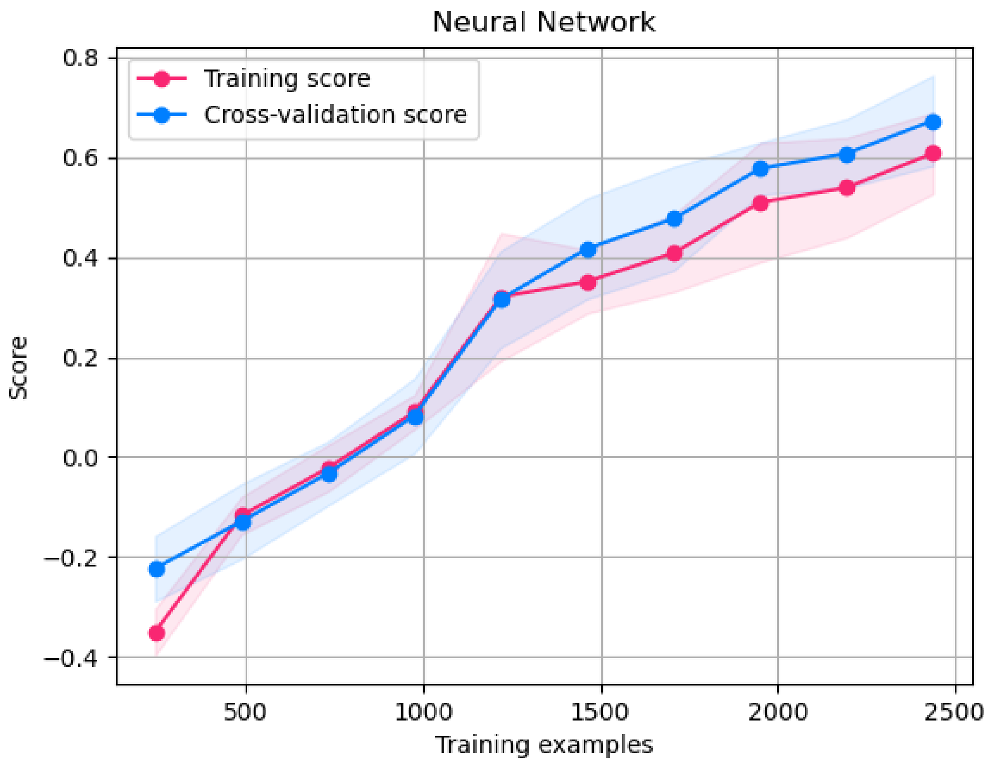
These gave me the notion that some features, such as “contentId” and “authorSessionId” were poorly correlated (value ~ zero) across the board, so I excluded from the analysis. “contentId” is for bookkeeping, so it would not be a good predictor anyway. “authorSessionId” would only be useful if there were certain authors that were significantly prolific with creating viral content. For this data set, it appears that is not the case. The heat maps reinforced the fact that the Virality equation variables would be the most important to consider since they are the most correlated features.

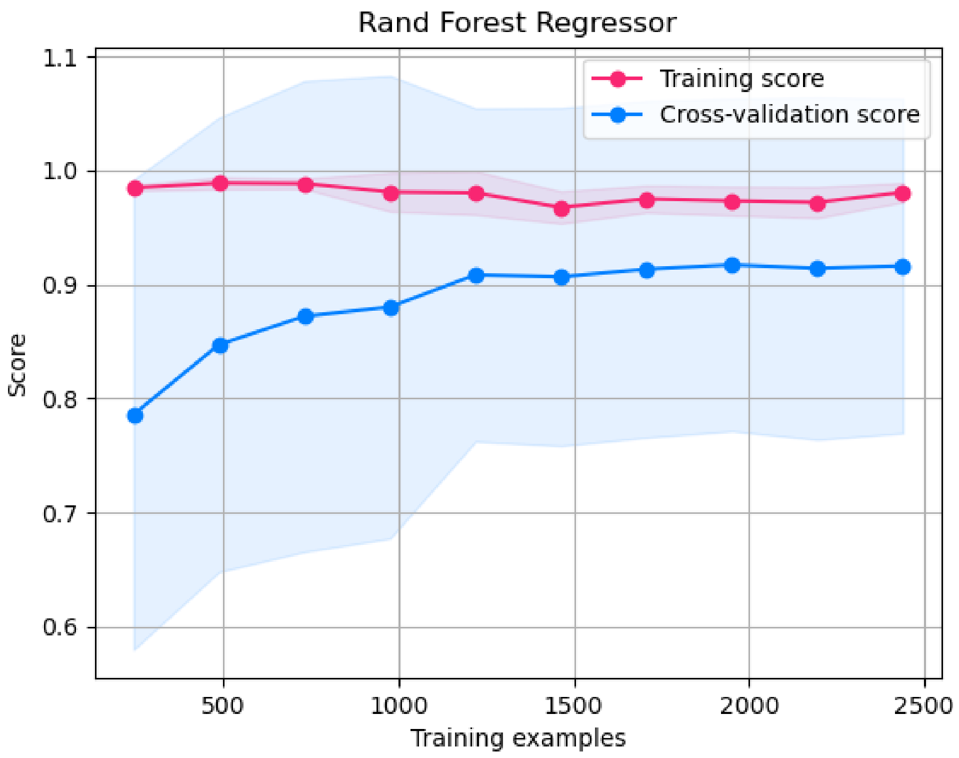
Finally, I made sure to scale the data to make sure that the model would be computed as quickly and efficiently as possible

○ What model did you use and why?

Due to time constraints, I choose regression models that could generate decent results immediately: a neural network regression and random forest regression models. These algorithms tend to be good for initial exploration of new data. Between the two, I would settle for random forest for now, because it is faster to train and performs better. Later on, neural networks have more potential to improve with more data point.

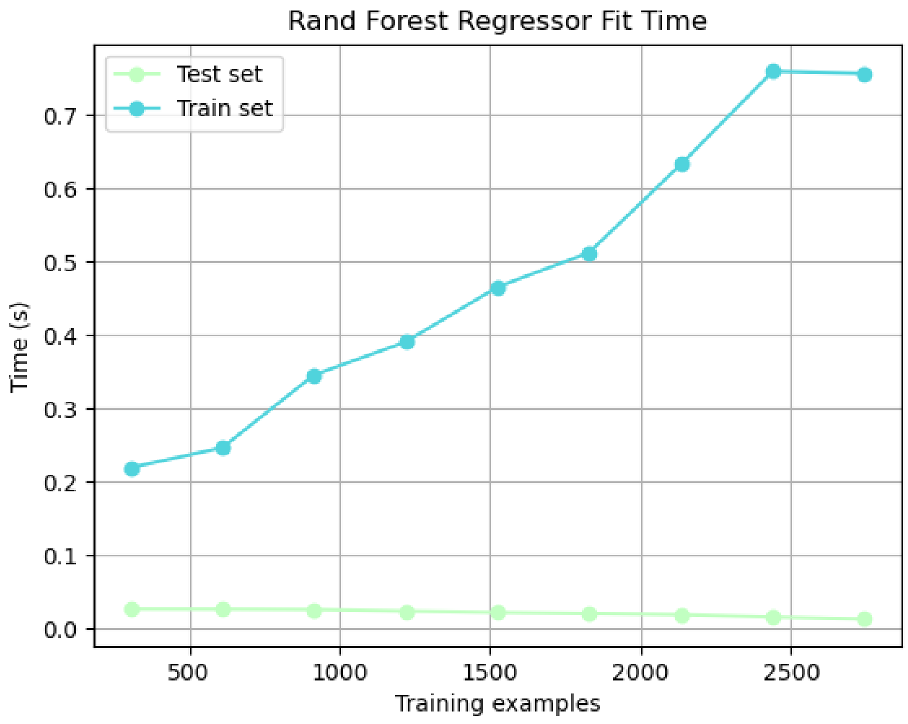
- What was your evaluation metric for this?

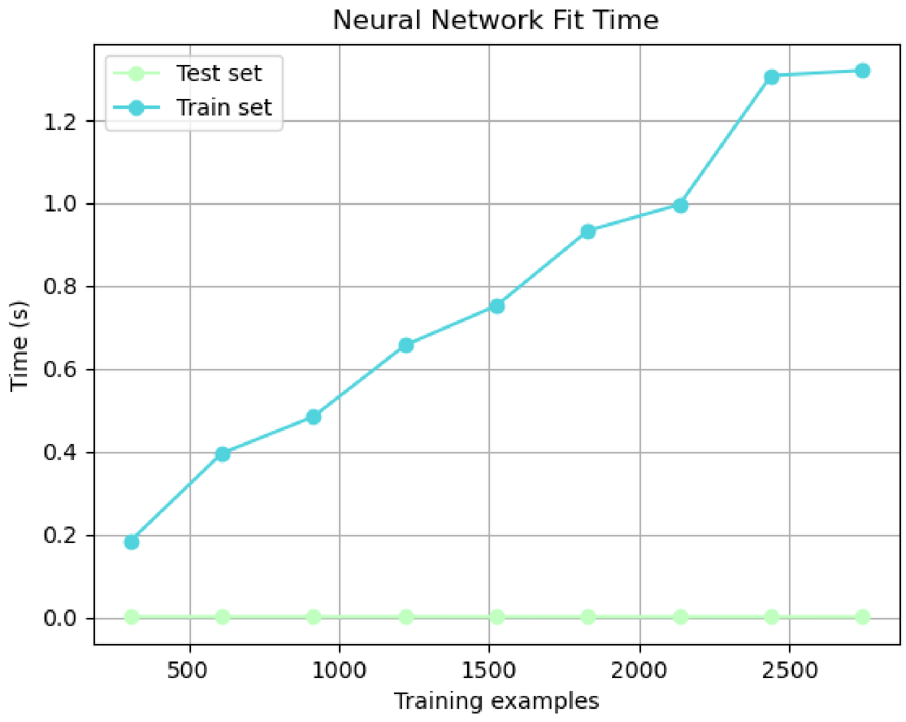




For neural networks, the upward trend of performance indicates that more data points would greatly improve the performance. It has a high-bias and low-variance in performance between training data and validation data. Its generalization accuracy sits around 68% on average.

For random forest, more data points would not improve the model since the performance gains have already flattened with the existing data. It is showing lower-bias and higher-variance performance compared to neural networks. Generalization accuracy sits around 91-92%.





Fit time graphs show that random forest is faster to train.

○ What features would you like to add to the model in the future if you had more time?

In the future, it would be very beneficial to apply time series learning algorithms and NLP algorithms to this type of data set. Time is a very key component in content becoming viral, and the data already contains elements of time in it. NLP would help to generate more signals for predicting virality.

○ What other things would you want to try before deploying this model in production.

I would like to spend more time tuning the hyperparameters of each model, especially neural networks because the upward trend of performance indicates that there potential to be mined. I could even develop a grid search procedure for this. Exploring other models is also not a bad idea.

I like to look at the data pipeline as well and look for ways to decrease and eliminate the problem of missing/NaN/null data. I would like to restore some of the features that were eliminated due to this problem so that there is more opportunity for data exploration. I would also like to look for new features that could be captured that would be useful for predicting virality.

It would also be important to make sure that the model can be updated regularly with new data generated every day. That means making sure that the infrastructure and software is in place to keep the data pipelines robust and tidy. It also means creating scripts/operations to update the model.