Stats 101C Final Kaggle Report

Team: Coffee^2 (Tyler Wu, Chelsea Miao, Charles Liu)

14 December 2020 – Department of Statistics, UCLA

1. Introduction

YouTube is a video-sharing platform that is widely known for its diverse videos uploaded by its users. The site contains various factors including the abilities to upload, view, rate, share, create playlists, comment on videos, and subscribe to other users. From these factors, we realized that views stand out as an important variable because they are a direct measure of engagement with a video and also help to determine how much revenue a content creator will make. Another factor to consider is how quickly a content creator is able to accumulate these views within a short time period. This indicates their overall success and possible future success with videos.

In this project, we aim to predict the percentage change in views on a video between the second and sixth hour since its publishing. In order to predict this metric, we have several video features at our disposal, including thumbnail image features, video title features, channel features, and other features. We utilized the classical statistical learning methods from James et al., *Introduction to Statistical Learning* to help us find the best Root Mean Square Error for this particular problem.

2. Methodology

(A) Understanding the Data

Initially, we mainly investigated predictors we thought might be useful for our model, but narrowed the list down through iteration with variable selection methods. From looking at the distribution for some predictors like views_2_hours (see Figure 1) and Num_Subscribers_Base_x, we found that there were a few outliers with extremely high number of views or base subscribers.

The presence of these outliers informed our decision to later use XGBoost, as it is better at mitigating the effects of potentially overfitting to individual points.

(B) Preprocessing

Cleaning + Feature Engineering:

We first checked that our data was complete, and indeed there were no missing values. We also split the `PublishedDate` feature into a numeric date and time form: `DaysSince1` and `MinsSince1`, and binned hours into 4-hour bins. In addition, for the channel metric data, we added the missing High level as its own column, since it is possible that our model may select only one of the levels as an important variable (as it did with `avg_growth_high`). Of the features created, the most useful ones that were included in our model were `avg_growth_high`, `bin_hour.0`, `DaysSince`, `MinsSince1`, and `punc_num_bar_bi`.

Variable Selection

For finding the predictors that would best fit our model, we explored the methods of Recursive Feature Elimination (*method not covered in class*), Random Forest importance, and LASSO cross-validation.

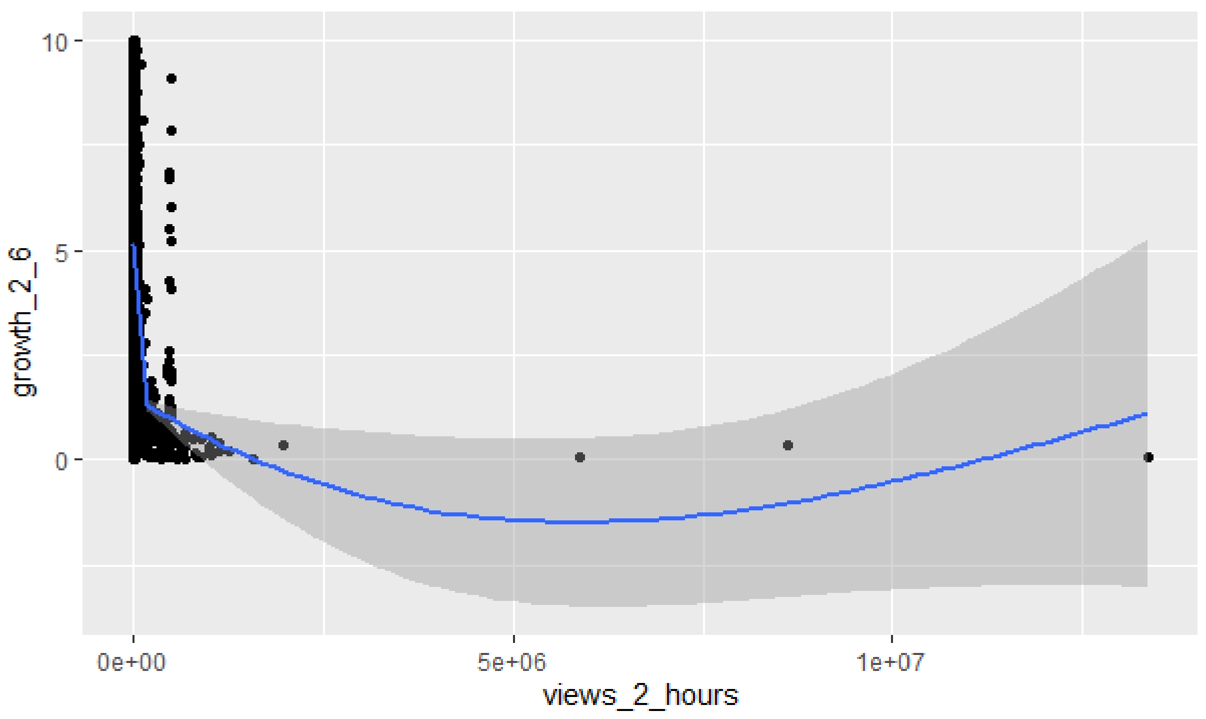

Figure 1: Data Exploration on views_2_hours

Our final model included the following variables: "avg_growth_high, avg_growth_low, avg_growth_low_mid, avg_growth_mid_high, bin_hour.0, cnn_10, cnn_12, cnn_17, cnn_19, cnn_25, cnn_68, cnn_86, cnn_88, cnn_89, count_vids_high, count_vids_low, count_vids_low_mid, DaysSince1, Duration, hog_454, mean_blue, mean_green, mean_pixel_val, mean_red, MinsSince1, num_chars, num_non_stopwords, Num_Subscribers_Base_high, Num_Subscribers_Base_low_mid, Num_Subscribers_Base_mid_high, num_uppercase_chars, Num_Views_Base_high, Num_Views_Base_mid_high, num_words, p_non_stopwords, punc_num_at, punc_num_bar, punc_num_bar_bi, punc_num_com, views_2_hours" (40 variables chosen)

(C) Outside Methods

1. Recursive Feature Elimination -

    i. Recursive feature elimination is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. By recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model. Features are ranked by the model's coef_ or feature_importances_ attributes.
    ii. The advantage is that the algorithm chooses the number of predictors instead of us having to choose it.
    iii. It is a preprocessing method.

2. XGBoost

    i. XGBoost is an implementation of the Gradient Boosted Decision Trees algorithm.
    ii. The parameters allowed us to tune against overfitting.
    iii. It was much faster to run than regular Gradient Boost. Aside from the parameters, XGBoost also differs in that its trees are built on an approximate greedy algorithm, where instead of using a Weighted Gini Index to inform the next tree, it uses a Gain measure, based on how much "similarity" is gained in the observations when moving down nodes of the tree. This algorithm is much faster to run than the usual Gradient Boost because of its inclusion of features like parallel processing, more efficient use of cache, and sparse matrices.

(D) Statistical Model

The final model we selected was XGBoost, or Extreme Gradient Boosting (*method not covered in class*). To tune our model, we wrote a cross validation function, (see *Appendix*), which looped through a randomly generated parameter list with `xgb.cv()` to record the optimal parameter values. The most significant parameters values were $\gamma > 0$ and $\lambda > 1$, indicating that our model was tuned to prevent overfitting (through stricter pruning). The Variable Importance plot (Figure 2) and the SHAP plot (Figure 3) were important in determining the predictors to use for our model.

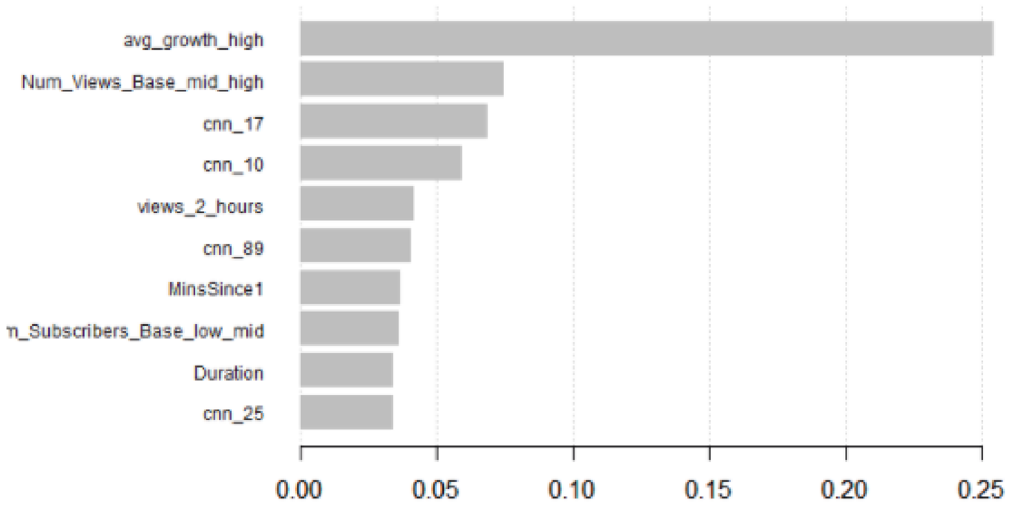

Figure 2: Variable Importance Plot

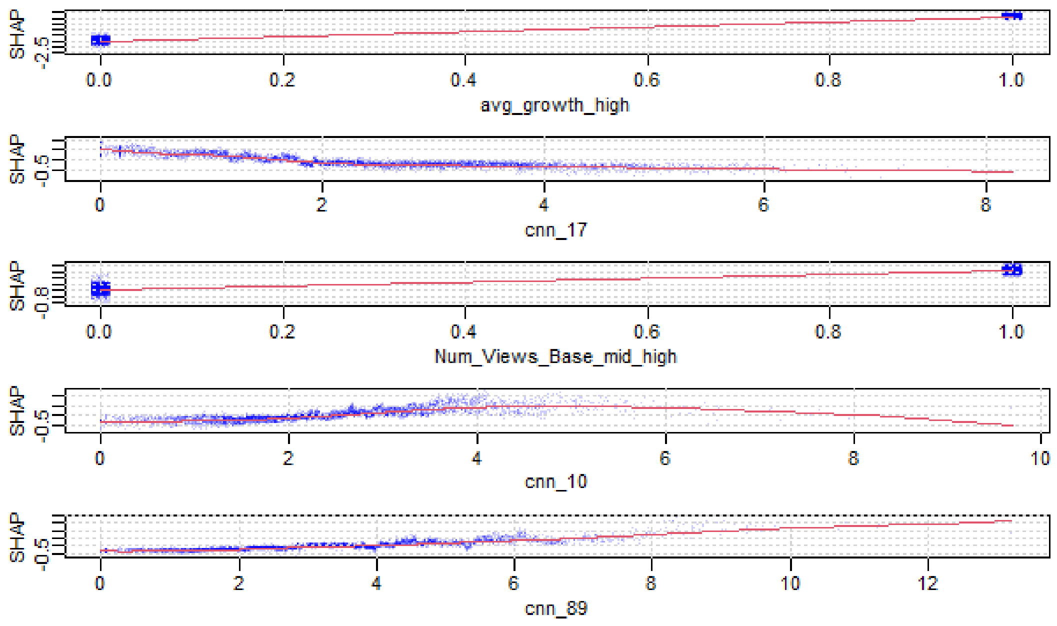

Figure 3: SHAP Plot for Top 5 Predictors

3. Results

The best iteration of our model scored 1.34459 on the public leaderboard, but 1.39679 on the private leaderboard. We believe the disparity is not too large that we need to wonder if we overfit our training data, and also our validation dataset returned an RMSE in a similar range.

4. Conclusions

We believe the model worked well for predictive purposes as it has a low and relatively consistent RMSE score. In terms of interpretability, aside from being able to view the most important variables through SHAP plots (similar to partial dependence plots), it is a little difficult to interpret the xgboost model.

5. Optional: References

1. XGBoost in R
2. XGBoost Parameters
3. XGB Plot Function
4. SHAP (SHapley Additive exPlanations)
5. Recursive Feature Elimination for Feature Selection
6. Recursive Feature Elimination

6. Contributions

Tyler: Data exploration (graphs), feature engineering, variable selection, a few Random Forest models, and tuning and validation of the final XGBoost model.

Chelsea: KNN tries, Boosting tries, RFE explorations, Random Forest importance explorations

Charles: Variable selection (Ridge Regression, LASSO, Elastic-Net, and Random Forest), tuning (Random Forest), and cross-validation (Ridge Regression, LASSO, and Elastic-Net)

7. Appendix

See next page