Predicting YouTube View Growth with Random Forest

Eustina Kim, Tiffany Feng

Introduction

How fast does the view count of a YouTube video grow? For a content creator, knowing the answer to such a question, especially for the first few hours a video goes live, can be extremely important in gauging audience reach and channel performance. Having a good sense of how well one's channel is doing allows a creator to know how to plan and manage their YouTube channels. In this project, we aimed to predict the percentage change in views on a video between the second and sixth hour after it's uploaded using predictors that fall into the following four categories: thumbnail image features, video title features, channel features, other features.

Methodology

1. Pre-processing

Upon initial exploratory analysis of our dataset, we discovered that the `PublishedDate` variable contained quite a bit of information that we felt would be more useful when disaggregated. We decided to split `PublishedDate` into four variables: **month**, **day**, **hour**, and **which_day**. The **month**, **day**, and **hour** variables indicate which month, day, and hour the video was uploaded. Variable **which_day** is a categorical variable indicating the day of the week the video was uploaded, with 0 indicating Sunday, 1 indiciating Monday, and so on. Although year was also included in the PublishedDate variable, we decided not to use the year as a predictor as all of the observations were from 2020 and including the variable would not provide additional insight. We also decided not to include minutes from `PublishedDate` as a predictor since we believe that the **hour** variable captures enough information about the time of day the video is published.

To perform variable selection, we used an algorithm called **Random Forest-Recursive Feature Elimination** (RF-RFE). RF-RFE selects variables by initially fitting a random forest model using all of the predictors available in the training dataset. It then calculates model performance and ranks predictors in the order of importance to the model. After this step, it removes the least important predictor and refits the model with the remaining predictors. This process repeats and continues to remove unimportant predictors. It continues until the algorithm reaches the desired ending subset size specified by the user (Kuhn 18). The rfe() function in the caret package carries out the algorithm and returns a subset of variables that produces the best model performance along with the associated variable names.

RFE has several benefits over other variable selection methods we have learned in class, such as backward selection. RFE is similar to backward selection, but it is not limited to linear regression and performs variable selection. According to our research, RFE is also useful "to reduce the effect of correlations in the last steps of the variable selection procedure and tends to select the variables which are the most able to predict the outcome" (Gregorutti et al. 16). As a result, we decided not to remove highly correlated predictors. RFE is also advantageous in that it does not put the burden of selecting the number of variables to include on the user, since it produces a recommendation for a subset of predictors that gives the lowest RMSE. For our project, we set the size of the subset for RF-RFE to range from 25 to 40. This decision was based on our experience with our previous models, which consistently had the lowest RMSE values when the number of included predictors fell in this range.

The algorithm chose 37 variables as the best subset because the random forest model with those variables had the lowest cross validation RMSE of 1.483. The graph below (figure 1) shows the change in RMSE against change in the number of variables in the model and it supports the fact that the RMSE is the lowest at 37.

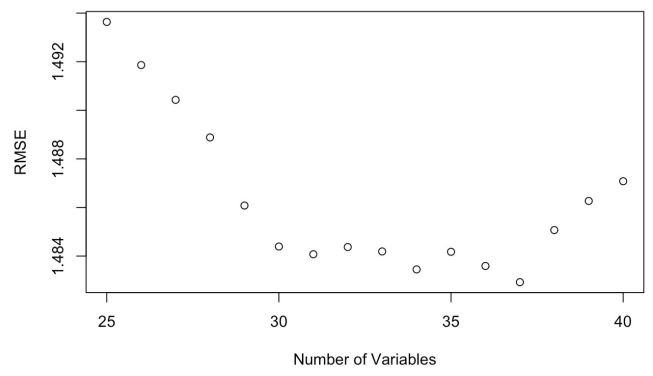


Figure 1

The top five most important variables are "cnn_10", "cnn_89", "cnn_86", "cnn_17", and "Num_Views_Base_mid_high". Four of these five variables are related to the video thumbnail. The full list of all 37 variables used in the final model is available in the Appendix under section (3). The variables are sorted according to their respective categories in the Appendix under section (4), with the most included variables relating to thumbnail image features.

2. Statistical Model

Out of all the methods learned during class, we decided to use random forest to fit the training data for the following reasons: 1. Random forest works well with large, complex datasets. 2. It is non-parametric so we do not have to make assumptions about the distribution of the data. 3. It requires less data pre-processing because they are robust to outliers.

Our final model is a random forest model with 37 predictors, mtry = 37 and ntree = 500 with 10-fold cross validation. `Mtry` is the number of variables available for splitting at each tree node and `ntree` is the number of trees to grow. With these definitions in mind, the final model is technically a bagging model because all predictors are considered for splitting a node. We chose an mtry of 37 because it gave the lowest cross-validation RMSE when we fit random forest models from mtry ranging from 1 to 37, as indicated by the graph below (figure 2).

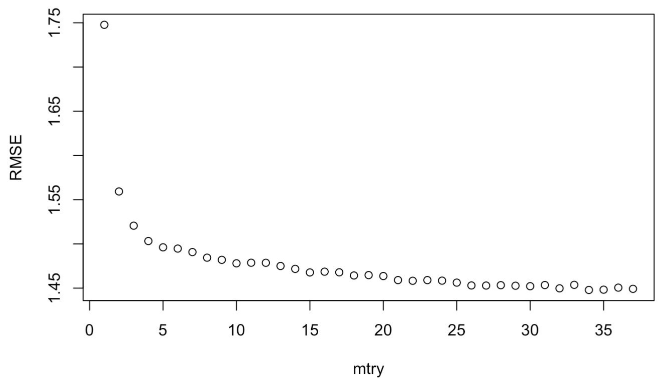


Figure 2

The model's cross-validation RMSE was 1.445015. This is a significantly better performance compared to a different tree based model we experimented with, which was gradient boosting. The boosting model using the same predictors with n.trees = 10000, interaction.depth = 4, and lambda = 0.008 had the cross-validation RMSE of 1.499343. The reason for this difference in performance may be because there are more parameters to tune for boosting than random forest, which we were unable to refine and test in our limited time frame. Also, the goal of boosting is to decrease bias so it may lead to overfitting (Lecture 7-2).

Results/Conclusions

The public RMSE score of the final model on Kaggle is 1.37229 and the private score is 1.39725. From the public to private score, there is a 1.81886% increase in the RMSE. The model performed consistently on both public and private data as indicated by the minimal increase in the RMSE. The score may have increased because our model had relatively high bias leading to underfitting, since the goal of bagging is to reduce variability (Lecture 7-2). There are some improvements we can make to minimize the RMSE even more. We can use more creative methods to transform the data because we only manipulated the `PublishedDate` variable. However, this would require greater domain expertise on the YouTube metrics in the dataset.

Despite the increase in the private RMSE, We believe that our model performed relatively well because we disaggregated the `PublishedDate` variable, tuned the `mtry` parameter, and used cross validation. Through this project experience, we learned how to employ machine learning methods to understand data we are unfamiliar with and predict accurate outcomes.

Statement of Contribution

Eustina Kim: Performed data transformation of data variables, researching about rfe, came up with the best performing model, wrote the report and slides.

Tiffany Feng: Data exploration, testing, tuning, and submission of boosting & bagging models, wrote the report and slides, researched modelling method strengths and weaknesses, edited presentation video.

Appendix

(1) Sources

Kuhn, Max. "The Caret Package." *Github Sites*, 27 Mar. 2019, topepo.github.io/caret/index.html.

Gregorutti, B., Michel, B. & Saint-Pierre, P. Correlation and variable importance in random forests. Stat Comput 27, 659–678 (2017). https://doi.org/10.1007/s11222-016-9646-1.

Vazquez, Alan. "Lecture 7-2: Bagging and Boosting". UCLA. Stats 101C Introduction to Statistical Models and Data Mining.

(2) R code

```
#### Load Libraries ####
library(tidyverse)
library(caret)
library(randomForest)
library(lubridate)

#### Load data,get rid of ID column ####
train <- read.csv("training.csv")
test <- read.csv("test.csv")
train <- train %>% select(-id)
ID <- test$id
test <- test %>% select(-id)

#### Data Transformation ####
# split PublishedDate into date and time
train <- train %>% separate(PublishedDate,c("PublishedDate","PublishedTime"),sep = " ")
test <- test %>% separate(PublishedDate,c("PublishedDate","PublishedTime"),sep = " ")
#change into proper date format
train$PublishedDate <- as.Date(train$PublishedDate,format = "%m/%d/%y")
```

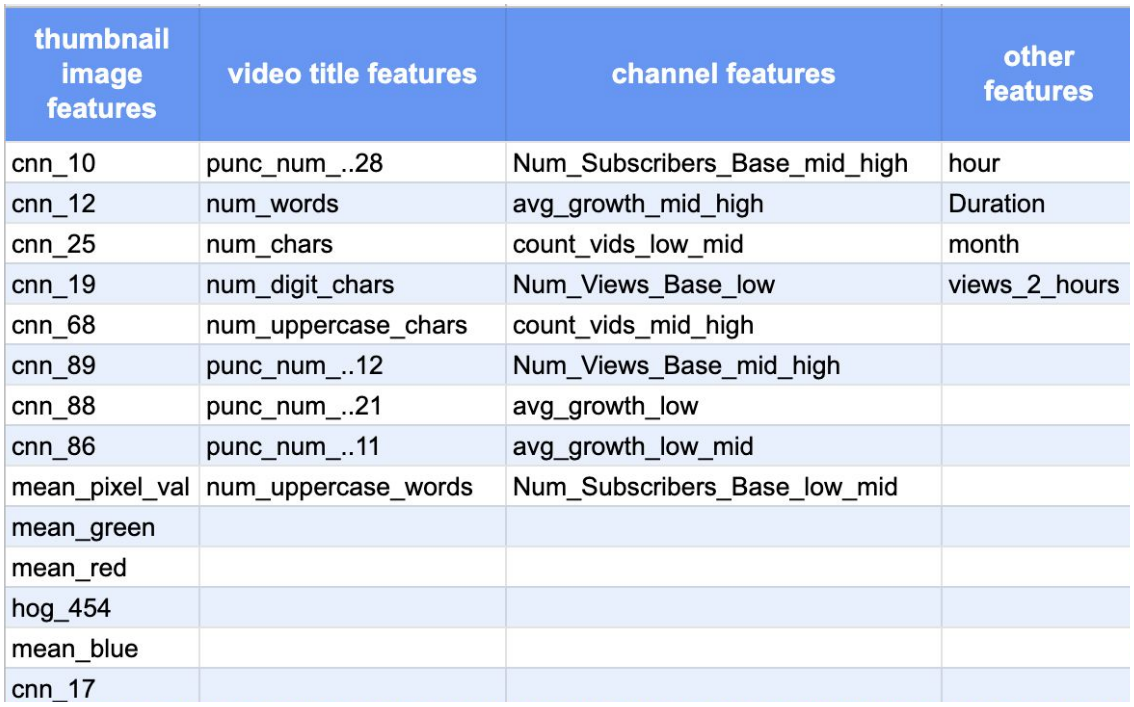
```
test$PublishedDate <- as.Date(test$PublishedDate,format = "%m/%d/%y")
#make variable: which day of the week
train$which_day <- wday(train$PublishedDate)
test$which_day <- wday(test$PublishedDate)
#make variables: month and day
train$month <- as.numeric(format(train$PublishedDate, format = "%m"))
train$day <- as.numeric(format(train$PublishedDate, format = "%d"))
test$month <- as.numeric(format(test$PublishedDate, format = "%m"))
test$day <- as.numeric(format(test$PublishedDate, format = "%d"))
#make variable: hour
train$hour <- as.numeric(format(as.POSIXct(train$PublishedTime,format="%H:%M"),"%H"))
test$hour <- as.numeric(format(as.POSIXct(test$PublishedTime,format="%H:%M"),"%H"))
#get rid of PublishedDate, PublishedTime
train <- train %>% select(-c(PublishedDate,PublishedTime))
test <- test %>% select(-c(PublishedDate,PublishedTime))

#### Random Forest Recursive Feature Elimination ####
set.seed(10)
subsets <- c(25:40)
ctrl <- rfeControl(functions = rfFuncs,
                   method = "cv",
                   number = 10,
                   verbose = FALSE)
x <- train[,-which(colnames(train)=="growth_2_6")]
y <- as.vector(train[,which(colnames(train)=="growth_2_6")])

rrfe <- rfe(x, y,
            sizes = subsets,
            rfeControl = ctrl)

#plot number of variables against change in RMSE
plot(rrfe$results$Variables[-17],rrfe$results$RMSE[-17],xlab="Number of Variables",ylab="RMSE")

#save important variables as a csv so that I don't have to run rfe again later
print(rrfe)
imp_vars <- rrfe$optVariables
write.csv(data.frame(imp_vars), "imp37.csv", row.names = FALSE)

#load the important variables and subset the train and test data
imp_vars <- read.csv("imp37.csv")
imp_vars <- imp_vars$imp_vars

trainImp <- train[,c(imp_vars,"growth_2_6")]
```

```
testImp <- test[,c(imp_vars)]

#### Training the Model  ####
#tune mtrys
control <- trainControl(method = 'cv', number = 10)
set.seed(1)
trys <- seq(1,37,by=2)
rmse <- rep(NA,length(trys))
for(i in 1:length(trys)){
  rf <- randomForest(growth_2_6~.,mtry=trys[i],
                     data = trainImp,
                     importance = T,
                     trControl = control,
                     ntree = 500)
  rmse[i] <- sqrt(mean(rf$mse[length(rf$mse)]))
}
#mtry=37 gives smallest RMSE of 1.446656
print(rmse)
trys[which.min(rmse)]
min(rmse)

#plot mtry against change in RMSE
plot(1:37,rmse,xlab="mtry",ylab="RMSE")

#### Final Model ####
#fit random forest with mtry=37
control <- trainControl(method = 'cv', number = 10)
set.seed(1)
final_mod <- randomForest(growth_2_6~.,mtry=37, data = trainImp,
                          trControl = control, importance = T, ntree = 500)
#cross validation rmse is 1.445236
print(final_mod)

#### Making Predictions ####

#make predictions on test data
predictions <- predict(final_mod,testImp)
final <- data.frame("id" = ID, "growth_2_6" = predictions)

#save results
write.csv(final,
        "submission.csv",  row.names = FALSE)
```

(3) Full list of predictors used in the final model in the order of importance:

"cnn_10","cnn_89","cnn_86","cnn_17","Num_Views_Base_mid_high","avg_growth_low","avg_growth_low_mid","Num_Subscribers_Base_low_mid","views_2_hours","cnn_12","cnn_25","punc_num_..28","hour","num_words","Duration","cnn_19","cnn_68","Num_Subscribers_Base_mid_high","avg_growth_mid_high","num_chars","cnn_88","num_digit_chars","count_vids_low_mid","num_uppercase_chars","month","punc_num_..12","Num_Views_Base_low","punc_num_..21","doc2vec_10","mean_pixel_val","punc_num_..11","mean_green","mean_red","hog_454","num_uppercase_words","mean_blue","count_vids_mid_high"

(4) Full list of predictors used in final model categorized:



thumbnail image features	video title features	channel features	other features
cnn_10	punc_num_..28	Num_Subscribers_Base_mid_high	hour
cnn_12	num_words	avg_growth_mid_high	Duration
cnn_25	num_chars	count_vids_low_mid	month
cnn_19	num_digit_chars	Num_Views_Base_low	views_2_hours
cnn_68	num_uppercase_chars	count_vids_mid_high	
cnn_89	punc_num_..12	Num_Views_Base_mid_high	
cnn_88	punc_num_..21	avg_growth_low	
cnn_86	punc_num_..11	avg_growth_low_mid	
mean_pixel_val	num_uppercase_words	Num_Subscribers_Base_low_mid	
mean_green			
mean_red			
hog_454			
mean_blue			
cnn_17			