Prediction of View Growth on Youtube

Group hairLESS
Group Members: Chelsea Chen, Wei Kong, Xiaoshu Luo

Introduction

YouTube is a video-sharing platform where users can upload and view videos, and the number of views is an important measure of engagement and popularity of the content uploaded, affecting uploaders' earnings from making the videos. Here, instead of using the number of views directly, an alternative variable measuring videos' early growth pattern is used as another indicator of the eventual success of videos. The objective of the project is the prediction of growth_2_6, which is the percentage change in views on a video between the 2nd and 6th hour since publishing, and the predictors are measures related to thumbnail image features, video title features, channel features, and other features.

Methodology

Data Preprocessing

Part 1: Initial data cleaning

The trivial column of ID is deleted at the beginning.

The predictor PublishedDate shows the published time of videos. Since it is a string variable and generates too many levels when used as categorical variables, data transformation is conducted through extracting significant information from it. Using the "stringr" package, we extract the years, months, dates, and time. For the purpose of obtaining additional information as well as making these data ready for the package to use directly, we also figure out the weekdays they were published and include weekday as a new variable in our data, as intuitively people may spend more time watching videos on weekends than on working days. Years are not included, as all of the observations were published in 2020, showing no variability.

Part 2: Selecting predictors

After cleaning the data, we aim to select the most significant variables for the algorithms. Firstly, we remove all the variables that remain zero throughout the dataset, as they are clearly not giving any useful information. Next, we filter out the numerical variables that are highly correlated to other ones to reduce multicollinearity that results in higher variance, and these highly correlated features may also mask the effect of interaction terms in tree models. Then, we use the "regsubset" function to select the most significant predictors, and we choose Mallows' Cp statistic as the standard as it gives the best results in the following training-validation process (Figure 1 shows the

result by different subset selection standards). Finally, since a great number of variables are left, another round of variable selection is performed based on their correlation with the response variable to decrease the number of variables used in the final model.

Part 3: Adding time variables

In the previous step, all the time-correlated variables are not included, since they are unlikely to pose a linear effect on the response variable. More specifically, it is not reasonable to expect a linear increase in growth_2_6 with the increase of time, no matter on the scale of hours, days, or month. However, as shown by Figure 2 and 3 in the Appendix, these variables do pose impacts on the response variable, so we decide to include them in our final model, and it turns out to be useful. At this moment, we are left with 34 predictors.

Statistical Model

The final model that we choose for submission is a bagging tree algorithm constructed on the 34 predictors from the previous process.

For model selection, we explore and tune a wide range of algorithms, including Ridge, Lasso, PCR, PLS, Randomforest and Bagging. To fit the model, we use 70% training data to train the models and leave the remaining 30% as validation set to evaluate the accuracy and adequacy of each model and avoid overfitting before submitting results to Kaggle. For linear models, we apply 10-fold cross-validation to prevent overfitting. For tree-based algorithms, we use the out-of-bag method instead as it produces a similar effect at a higher efficiency by only giving each tree a limited portion of the whole training data. The tuning parameter for Lasso and Ridge is the penalty parameter lambda, while for PCR and PLS it is the number of components included. For random forest, it is the number of predictors taken into consideration during each split. There is no tuning parameter for the bagging algorithm.

We train and tune the linear models with the training data and all the predictors, as some of these methods include auto variable selection. After comparing the performance of all the modes, it turns out that none of the ridge regression, lasso regression, pls regression, and principal component regression methods are able to get training RMSE's lower than 1.6 and most of the test RMSE's are around 1.66. Algorithms using trees generally give lower errors than the others, so we focus on random forest and tree bagging for regression. Indeed, random forest regression outperforms lasso, ridge, pcr, and plsr in both training and testing RMSE, which is reasonable since some of our predictors are not linearly correlated to our dependent variable, while tree models are able to take their effects into consideration. We also notice that while the tuning parameter "mtry" is set to the largest possible value in most times, indicating a positive relation between performance and number of predictors

during each split. The bagging algorithm gives the best result and is thus selected as our final model.

Results

The bagging tree for regression gives a training RMSE of 0.6 and testing RMSE of 1.47 in our local train-validation process. For the Kaggle competition on prediction of new testing data, the model reaches an RMSE of 1.39065 on the public leaderboard, and 1.40951 on the private leaderboard.

Conclusions

Based on the result, our bagging tree model has reached a decent accuray that outperforms all four benchmarks provided on the public leaderboard, and it also generalizes well to the private testing data.

Most importantly, the data preprocessing has effectively prevented high variance caused by over-complexity. While the removal of highly-correlated variables resolves the issue of multicollinearity, the application of subset selection eliminates insignificant predictors. Finally, the remaining variables have been through another round of selection to further simplify the model.

Besides, the technique of out-of-bag sampling in the training process has successfully prevented overfitting by restricting each tree's access to the full training set, contributing to the generality of the result. Also, parameter optimization implies a better performance with an increased number of predictors during each split, guiding us to use the bagging algorithm as a special version of randomforest. Moreover, the addition of transformed time-related predictors, which are non-linearly correlated with the response variable, further improves the accuracy.

However, there is still room for improvement in our method, especially in the predictor selection process. We assume predictors that are really lowly correlated to the response variable are not significant and thus eliminate them from the fitting process, but it is very likely that they are still significant non-linearly, and exclusion of them might lead to lower prediction accuracy. If more time is allowed, it might be better to separately analyze each of the variables.

Appendix

Figure 1: Variable Selection

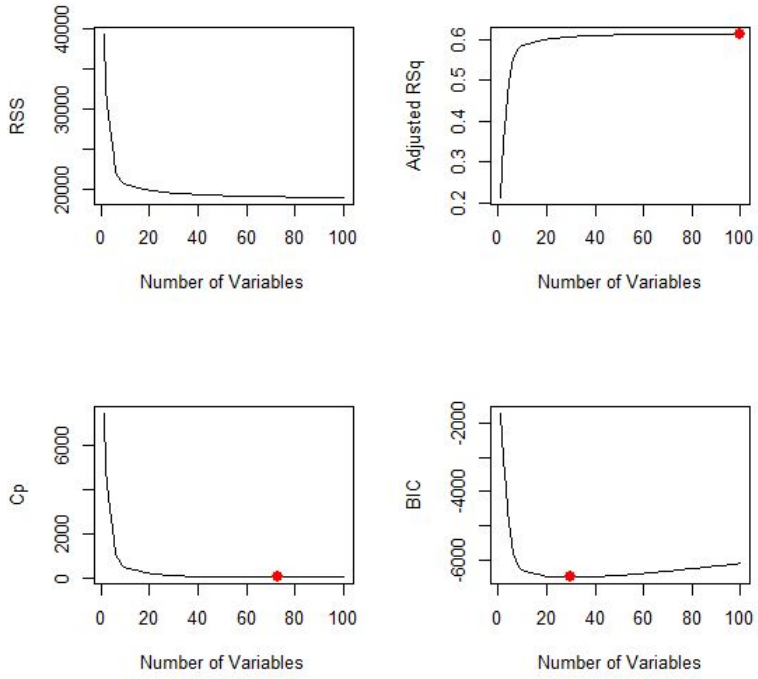


Figure 2: Boxplot of Video Growth v.s Month

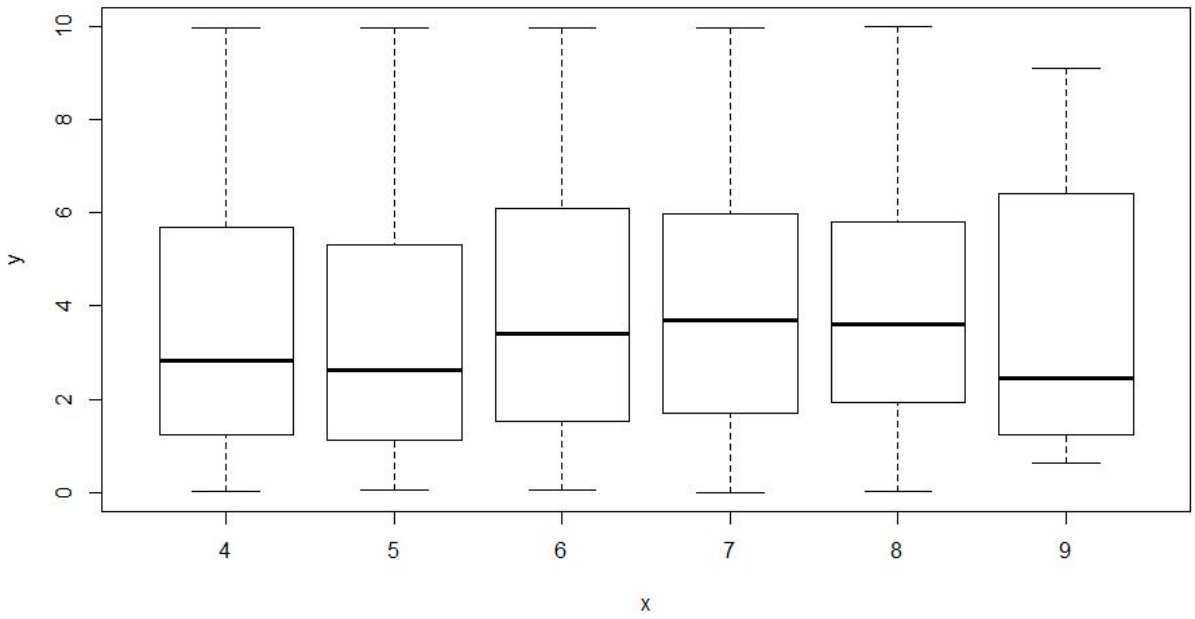
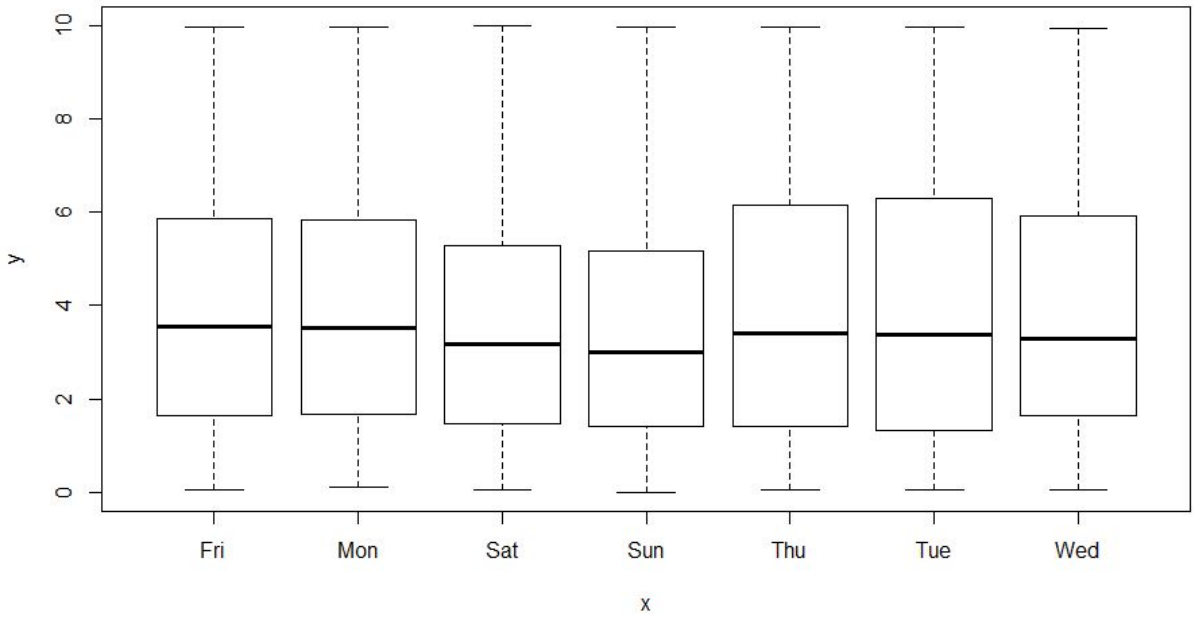


Figure 3: Boxplot of Video Growth v.s Weekdays



Code:

```
#include necessary libraries
library(leaps)
library(glmnet)
library(ISLR)
library(mclust)
library(caret)
library(randomForest)
library(stringr)

#Fix publish time - In this part, month, date, time, and weekday of each submission is calculated
#First fix the training set
temp <- read.csv("training.csv")

#variables are in form of "4/17/2020 10:38"
#exclude years
year <- str_extract_all(temp$PublishedDate, "(?<=\\/)\\d*(?=\\s)") %>% unlist() %>% as.numeric()
summary(year)
```

```
#date only from April to Sept
month <- str_extract_all(temp$PublishedDate, "^\\d*(?=\\/)") %>% unlist() %>%
as.numeric()
summary(month)
date <- str_extract_all(temp$PublishedDate, "(?<=\\/)\\d*(?=\\/)") %>% unlist() %>%
as.numeric()
summary(day)
day <- date
for(i in 1:nrow(temp)){
  if(month[i] == 5)
    day[i] <- day[i]+30
  if(month[i] == 6)
    day[i] <- day[i]+30+31
  if(month[i] == 7)
    day[i] <- day[i]+30+31+30
  if(month[i] == 8)
    day[i] <- day[i]+30+31+30+31
  if(month[i] == 9)
    day[i] <- day[i]+30+31+30+31+31
}
temp$day <- day
temp$month <- month
temp$date <- date

#time, hour and minute into decimals, for example, 10:30 to 10.5
hour <- str_extract_all(temp$PublishedDate, "(?<=\\s)\\d*(?=\\:)") %>% unlist() %>%
as.numeric()
minute <- str_extract_all(temp$PublishedDate, "(?<=\\:)\\d*$") %>% unlist() %>%
as.numeric()
temp$pTime <- hour + minute/60

#find weekday of publication for every observation
weekday <- rep("NA", nrow(temp))
#Apr.1 is wed
for(i in 1:nrow(temp)){
  if(day[i]%%7 == 1)
    weekday[i] <- "Wed"
  if(day[i]%%7 == 2)
    weekday[i] <- "Thu"
```

```
  if(day[i]%%7 == 3)
    weekday[i] <- "Fri"
  if(day[i]%%7 == 4)
    weekday[i] <- "Sat"
  if(day[i]%%7 == 5)
    weekday[i] <- "Sun"
  if(day[i]%%7 == 6)
    weekday[i] <- "Mon"
  if(day[i]%%7 == 0)
    weekday[i] <- "Tue"
}
temp$weekday <- weekday
train <- temp

#Then fix test data in the same way
temp <- read.csv("test.csv")

#variables are in form of "4/17/2020 10:38"
#exclude years
year <- str_extract_all(temp$PublishedDate, "(?<=\\/)\\d*(?=\\s)") %>% unlist() %>%
as.numeric()
summary(year)

#date only from April to Sept
month <- str_extract_all(temp$PublishedDate, "^\\d*(?=\\/)") %>% unlist() %>%
as.numeric()
summary(month)
date <- str_extract_all(temp$PublishedDate, "(?<=\\/)\\d*(?=\\/)") %>% unlist() %>%
as.numeric()
summary(day)
day <- date
for(i in 1:nrow(temp)){
  if(month[i] == 5)
    day[i] <- day[i]+30
  if(month[i] == 6)
    day[i] <- day[i]+30+31
  if(month[i] == 7)
    day[i] <- day[i]+30+31+30
  if(month[i] == 8)
    day[i] <- day[i]+30+31+30+31
```

```
  if(month[i] == 9)
    day[i] <- day[i]+30+31+30+31+31
}
temp$day <- day
temp$month <- month
temp$date <- date

#time, hour and minute into decimals
hour <- str_extract_all(temp$PublishedDate, "(?<=\\s)\\d*(?=\\:)") %>% unlist() %>%
as.numeric()
minute <- str_extract_all(temp$PublishedDate, "(?<=\\:)\\d*$") %>% unlist() %>%
as.numeric()
temp$pTime <- hour + minute/60

#find weekday of publication for every observation
weekday <- rep("NA", nrow(temp))
#apr.1 is wed
for(i in 1:nrow(temp)){
  if(day[i]%%7 == 1)
    weekday[i] <- "Wed"
  if(day[i]%%7 == 2)
    weekday[i] <- "Thu"
  if(day[i]%%7 == 3)
    weekday[i] <- "Fri"
  if(day[i]%%7 == 4)
    weekday[i] <- "Sat"
  if(day[i]%%7 == 5)
    weekday[i] <- "Sun"
  if(day[i]%%7 == 6)
    weekday[i] <- "Mon"
  if(day[i]%%7 == 0)
    weekday[i] <- "Tue"
}
temp$weekday <- weekday
test <- temp

#Preprocessing
timestrain <- train[, 261:265]   #save the time related predictors for later
X <- train[, 3:260]
Y <- train$growth_2_6
```

```
corr <- cor(X, X$growth_2_6)

#find and get rid of zero correlated predictors
zero_pred <- c()
for (i in 1:dim(X)[2]){
  if(is.na(corr[i])){
    zero_pred < -c(zero_pred, i)
  }
}
X <- X[ ,-zero_pred]

#get rid of highly correlated predictors
high_cor_index <- c()
for (i in 1:dim(X)[2]){
  for (j in 1:dim(X)[2]){
    t <- cor(X[, i], X[, j])
    if(t >= 0.8 && i != j){
      High_cor_index <- c(high_cor_index, j)
    }
  }
}
High_cor_index <- unique(high_cor_index)
X <- X[, -high_cor_index]

#Use regular subsets to find potential useful predictors for this problem
rsub <- regsubsets(growth_2_6~., data = X, nbest = 1, nvmax = 100,
                   intercept = TRUE, method = "backward", really.big = T)
sumBS <- summary(rsub)
XX <- (X[, (sumBS$which)[72, ]])

set.seed(904971914)
low_sig_pred <- c()
corr2 <- cor(XX, XX$growth_2_6)

#Get rid of low predictability predictors
for(i in 1:ncol(XX)){
  if(abs(corr2[i]) < 0.09){
    low_sig_pred <- c(low_sig_pred, i)
  }
}
```

```
XX <- cbind(XX[, -low_sig_pred], timestrain)

#Training
oob_train_control <- trainControl(method = "oob", savePredictions = TRUE)
recommended.mtry <- floor(34) #Since using bagging, mtry equal to number of all
predictors
tunegrid <- expand.grid(mtry = recommended.mtry)
#Train the model  using the whole set
forestfit.whole <- train(growth_2_6~.,
                         data = XX, method = 'rf', importance = FALSE,
                         trControl = oob_train_control, tuneGrid = tunegrid)

#Prediction
pred_test <- predict(forestfit.whole, newdata = test)
result2 < -data.frame(id = test[, 1], growth_2_6 = pred_test)
write.csv(result2, "try4.csv", row.names = F)
```

Statement of contributions

Every group member tried different ways of preprocessing the data and produced a variety of models.

C.Chen preprocessed data, came up with the idea to transform published date into the form of month, time and weekdays, trained and tuned models including lasso regression, ridge regression, pls regression, principal component regression, and random forest regression, and wrote up parts of the final report.

W.Kong preprocessed data, trained various linear models including Lasso regression, Ridge regression, PLS regression and PCR regression, generated graphs for better visualization of data, and finalized code for submission, and wrote up parts of the final report.

X.Luo explored data preprocessing by removing insignificant variables, tuned the parameters for ridge, lasso, PCR, PLS and randomforest regression, came up with the final bagging tree model, and wrote up parts of the final report.