Stats 101C Final Report

Eleanor Jiang Hong Xiong Mengyu Zhang

December 15, 2020

Contents

1 Introduction

In this project, our goal is to predict the percentage change in views on a YouTube video between the second and sixth hour since its publishing. The final model was made based on a training data with size 7242*260. The training data set contains 258 predictors which are video features generally fitting into four categories: thumbnail image, video title, channel and others. These are all variables related to a YouTube video's percentage change in view on the first sight, which is the variable 'growth_2_6' in the data set. As result, after selecting potential predictors, we successfully made a random forest model which can be used to make a highly accurate prediction.

2 Methodology

2.1 Preprocessing

2.1.1 Data Cleaning and Transformation

There is no missing value in the data. We then remove variables with zero variance, since they not only are considered to have less predictive power but also might cause computational problems that cause the model to crash. Especially if we are performing a standardization to the data, those zero-variance variables will be transformed nearly zero, which can occur either a numerical problem or a precision issue.

Since the categorical variables, except those we just created in the feature engineering session, are all binary variables, we can apply Pearson correlation along with other numerical variables to check the multi-collinearity. We use the 'findCorrelation' function from the 'caret' package, which would remove the higher mean absolute correlation in dropping one of the highly correlated pair. We consider 0.9 to be our cut-off point of exhibiting high correlation.

It is hard to make a interpretation of the number 0s and 1s if they are considered as numerical data. As a result, after data cleaning, we transform the following binary variables into categorical type with two levels.

Num_Subscribers_Base_low, Num_Subscribers_Base_low_mid, Num_Subscribers_Base_mid_high, Num_Views_Base_low, Num_Views_Base_low_mid, Num_Views_Base_mid_high, avg_growth_low, avg_growth_low_mi, avg_growth_mid_high, count_vids_low_mid, count_vids_mid_high

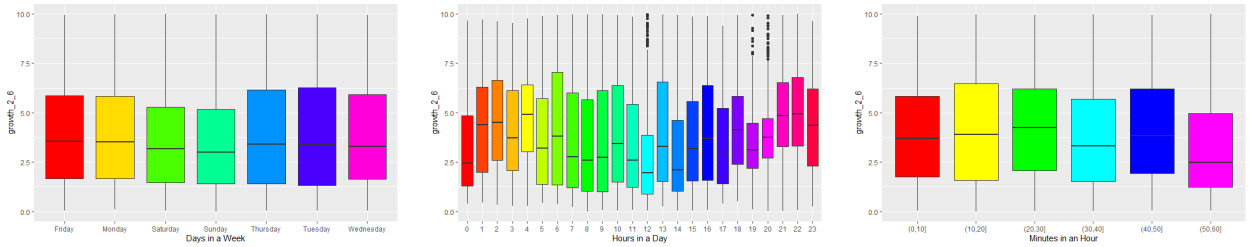
2.1.2 Feature Engineering: 'PublishDate'

After reviewing all the variables that may potentially influence the dependent variable, we find that the *PublishDate* is likely to contribute new information to our model prediction. However, since the data is in a character type which contains both the posted date and detailed time, which is hard to be considered as a categorical variable and make no contribution to the interpretation as well. Then we split the variable into three parts: published hour in a day, published minutes in an hour, and the published date in a week. However, we fail to include the minutes variable in the competition because of time issues.

To make a better approaching of the relation between the published time and the growth rate, following plots are created.

In the first box-plot, which shows that the growth rate on the weekends (Saturday and Sunday) is a little lower than the weekdays. One possible reason is that people have more time during the weekends, so that most videos are viewed in two hours after posted, which lead to a decreasing in the growth rate in two to six hours. So that it is reasonable to consider the days in a week as a new potential categorical predictor.

Then in the second plot, it is clearly that there is one peak during the evening time period. The phenomenon is acceptable since the people usually have more time to relax during "prime time" period. This also shows that



the published hour in a day should be considered as a predictor.

Lastly, when dealing with the specific minutes that a video is posted. Since it is meaningless to take the published minute as a numerical variable, also 60 levels are too much if we just left the 60 minutes in an hour, then we breaks an hour into six 10-minutes intervals. In this way, this variable can be considered as a categorical type and easy to interpret. However, our divide in minutes do have conceptual limitations which we will discuss in the last section.

2.2 Statistical Model

2.2.1 Feature Selection

After data pre-processing, we still have 154 predictors remaining in our dataset. In order to reduce the over-fitting and computational cost of our model, we conduct a feature selection by random forest. Then, we start from selecting 20 predictors with highest importance to see its RMSE, then increase the number of predictors gradually until there is no significant improvement upon RMSE. Finally, we find that 30 predictors with highest importance provide the best RMSE for the test data.

Since Random Forest model will train upon different subsets of data and make average of their results, we apply the whole data set as training data to increase the predicting power and do not worry much about over-fitting. Also to increase model accuracy while decreasing the probability of over-fitting, we increase the default value of 500 trees to 1500 trees, which is a decent balance of this trade-off. Increased training data boosts the predicting power of our Random Forest Model and increased trees further improves the accuracy while not causing over-fitting.

2.2.2 Model Description and Validity

After data pre-processing, we still have 154 predictors remaining in our dataset. In order to reduce the over-fitting and computational cost of our model, we conduct a feature selection by random forest. Then, we start from selecting 20 predictors with highest importance to see its RMSE, then increase the number of predictors gradually until there is no significant improvement upon RMSE. Finally, we find that 30 predictors with highest importance provide the best RMSE for the test data.

After applying Lasso regression, Ridge regression, Principal component regression, Partial least square regression, Gradient Boosting Machine, and Random Forest to our test data partition, we noticed that Random Forest model achieves a significantly lower RMSE compared to all other models. Therefore, we finally choose Random Forest as our final model for this competition. We also use 'Out-of-Bags' as our control method. Comparing to other methods such as cross validation, OOB produced more performant RF model because it uses the complete sample to train and estimate the errors. Moroever, OOB is advantageous in saving computational cost; only one RF has to constructed[1].

Therefore, we run a Random Forest model upon the selected 30 predictors. Since 'ntree' is the number of trees to grow, we set that to a large number of 1500 to ensure as many row inputs can get predicted at least a few times. During our trials, we found out within 30 predictors, the RMSE resulted from different 'mtry' is small in a 0.01 scale with a best performance around 29-30 mtries. As a result, our model explains 68.75% of the variance in the dataset.

3 Result

Here are the results of our final model:

Team Name	: Siege Lion
Members	: Eleanor Jiang(zoe1trick), Hong Xiong(xionghong), Mengyu Zhang(onezmy)
Final Kaggle Score	: 1.41093 on Public Board, higher than Model 4
Model Selection	: Random forest method with 30 predictors
Score Explanation	: We think the decent performance of our model is credited to the choice of random forest method, variable selection, and increased training dataset.

4 Conclusions and Further Discussion

Since we are using *RStudio* to help apply the method we learned to analyze all the data set, the software can offer us a accurate result to show the MSE and other statistical values, but it cannot directly help us to improve the models we have. Also, although we have explored and discussed a lot about the selection of the variables as well as the random forest method, there is definitely a large room for improvement according to our Kaggle ranking and scores. our variable selection bases solely on the importance index provided by Random Forest Model and choice of number of predictors is based on trials instead of more technical and formal methods to analyze.

In a further research, firstly, there might be a better way to divide the time periods in 'PublishedDate' variable when doing feature engineering. For example, to predict growth rate in 2 to 6 hours, individual period of 10 minutes seems to make no sense from intuition. We also fail to order the factor levels in hours, which boosted our score to 1.39 after re-level. Secondly, we can apply more variable selection methods and make a majority vote among those methods to select variables, like Best Subset Selection and Lasso Regression. In this way, we might be able to identify predictors with higher importance in general and avoid the biases from any single model. Then, it is uncertain to make the conclusion that all the rest variables that we did not use in the model are statistically meaningless to appear in the prediction model, and the optimal number of predictors can be better estimated through a RMSE plot with number of predictors as our x-axis from Random Forest. Also, we might apply other methods like Lasso and Principal Component regression to estimate number of predictors needed for an optimal performance.

What is more, we might choose other regression models with decent performances and do a model ensemble to take average of their respective results. In this way, the bias from single model could be reduced and such a vote can help us take advantage of different predicting powers from multiple models.

On the other hand, Our Kaggle score is 1.41093 on the public board and 1.4168 on the private board, which means the model makes a valid model with good approaching of the overall prediction and finally passed the test of given 'Model 4'. The small discrepancy upon two scores shows that we successfully avoid a minor over-fitting effect from applying the whole detest as training set and increased tree numbers. Overall, since the prompt was focused on reducing the mean squared error, our model does decent for this purpose and got a RMSE in an acceptable range with the test of cross-validation. By applying the model to test data, we finally successfully make a prediction of the growth rate within a tolerance range. When doing a further approaching, the interaction terms of categorical data should be taken into consideration, also some hypothesis testing method can be applied to figure out the contribution of some specific predictors. In this way, we believe that a better more with more statistically meaningful predictor can be made after further discussion.

5 Appendix

5.1 R Code

```
library(dplyr)
library(stringr)
library(randomForest)
library(e1071)
library(caret)
library(car)
library(lubridate)

#Load Data
training <- read.csv("training.csv")
test <- read.csv("test.csv")

#Feature Engineering
PublishDate <- c(training$PublishedDate, test$PublishedDate)
date <- PublishDate %>% str\_extract("(.*)(?=\\/)") %>% as.Date(format="%m/%d")
wkds <- weekdays(date)
wkds <- as.factor(wkds)
hrs <- PublishDate %>%
  str\_extract("(?<=\\s)(.*)(?=:)") %>%
  as.factor()
mins <- PublishDate %>%
  str\_extract("(?<=:)(.*)")
mins <- ifelse(mins=="00", "60", mins)
mins <- as.integer(mins)
mins <- cut(mins, breaks=c(0, 10, 20, 30, 40, 50, 60))
##provide graphs
training$wkds <- wkds[1:7242]
training$hrs <- hrs[1:7242]
training$mins <- mins[1:7242]
par(mfrow=c(1, 2))
ggplot(training, aes(x=hrs, y=growth\_2\_6)) +
  geom\_boxplot(fill=rainbow(24)) +
  labs(x="Hours In A Day", y="growth\_2\_6")
ggplot(training, aes(x=wkds, y=growth\_2\_6)) +
  geom\_boxplot(fill=rainbow(7)) +
  labs(x="Hours In A Day", y="growth\_2\_6")
ggplot(training, aes(x=mins, y=growth\_2\_6)) +
  geom\_boxplot(fill=rainbow(6)) +
  labs(x="Hours In A Day", y="growth\_2\_6")

#subset data
subdata <- training[, 3:259]

#remove 0 vars
subdata <- subdata[, -nearZeroVar(subdata)]

#remove high correlated r.v's
subdata <- subdata[, -findCorrelation(cor(subdata), cutoff=.9)]

#Transform categorical
for (i in 141:151) {
  subdata[, i] <- as.factor(subdata[, i])
}
subdata$growth\_2\_6 <- training$growth\_2\_6
```

```
subdata$hrs <- training$hrs
subdata$wkds <- training$wkds
#subdata$mins <- training$mins

#Feature Importance
set.seed(123)
recommended.mtry <- floor(sqrt(ncol(subdata)))
rf.imp <- randomForest(growth\_2\_6~., subdata, mtry=154, ntree=1500)
imp <- importance(rf.imp, scale=F, type=1)
imp.ordered <- imp[order(abs(imp), decreasing=T),]
selected <- c(names(imp.ordered[1:30]), "growth\_2\_6")
subdata2 <- subdata[, which(colnames(subdata)%in%selected)]

#fit model
rf <- randomForest(growth\_2\_6~., subdata2, importance=F, mtry=30, ntree=1500,
                   trControl=oob\_train\_control)

#Convert test data
test$wkds <- wkds[7243:10347]
test$hrs <- hrs[7243:10347]
test$mins <- mins[7243:10347]
for (i in 248:259) {
  test[, i] <- as.factor(test[, i])
}
pred <- predict(rf, newdata=test)
submit <- data.frame(id=test$id, growth\_2\_6=pred)
write.csv(submit, file="submit.csv", row.names = F)
```

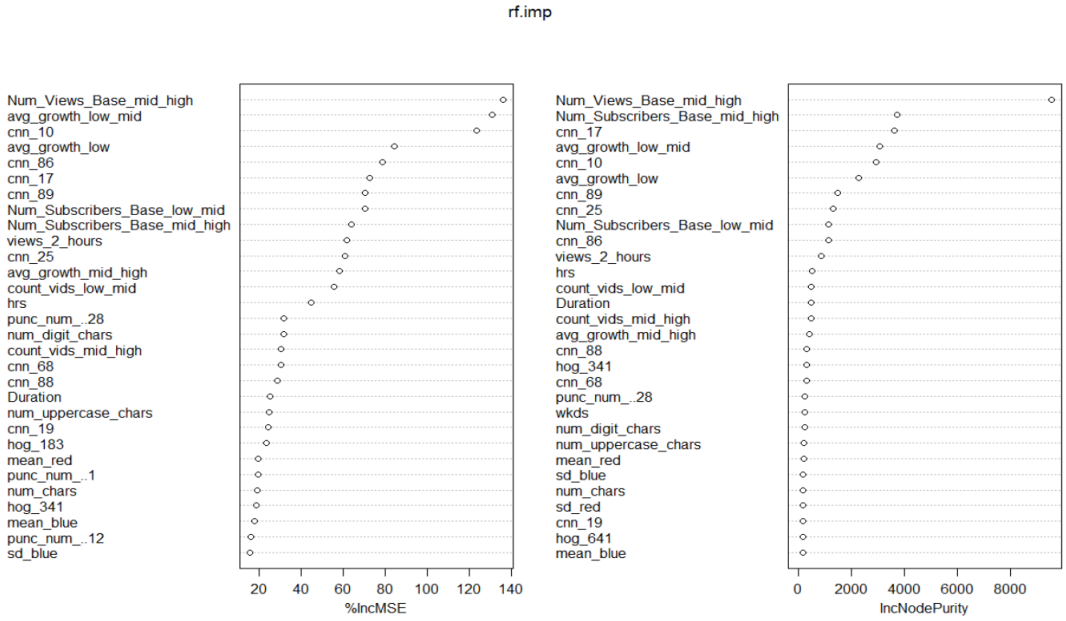
5.2 Statement of Contribution

Eleanor Jiang: Feature engineering, tuning the hyper parameters of RF model, and help organizing the code

Hong Xiong: Feature selection, proposing random forest model, and help write the report.

Mengyu Zhang:Trying use logistic regression to help with variable selection, participate in data transformation part. Doing report writing work.

5.3 Other Useful Graphs



References

[1] Y-h. Taguchi, On the overestimation of random forest's out-of-bag error
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6078316/

[2] Free Code Camp, Key Machine Learning Concepts Explained
https://www.freecodecamp.org/news/key-machine-learning-concepts-explained-dataset-splitting\-and-random-forest/

[3] Erin LeDell, useR! Machine Learning Tutorial
https://chrisalbon.com/machine_learning/trees_and_forests/feature_selection_using_random_forest/