Statistics 101C Final Report:

Predicting Percent Changes in Youtube Video Views using Thumbnail Image, Video Title, Channel and Other Features

Team: Sunflower Cello Hotdog

Anna Piskun —

Derek Wang —

Tomi Rajninger —

1 Introduction

As stated on Kaggle, YouTube is a vastly popular video-sharing platform owned and operated by Google since 2006. Users are given the freedom to share a variety of content on this site as well as interact with other users. These content creators often perceive video views as a measure of their channel's current and future success. Using predictors that broadly fit into the four categories of Thumbnail Image Features, Video Title Features, Channel Features and Other Features, for our final project, we built a model to predict the percentage change in views on a video between the second and sixth hour of its publishing.

2 Methodology

2.1 Data Preprocessing

After downloading the dataset from Kaggle, we first used the is.na() function to identify if there were any missing observations to remove from the data. After verifying that there were none, we ran the str() function to explore the structure of the dataset and its predictors. From this output, we found that all variables in the dataset were numeric, except for *PublishedDate*, which represents the data and time each video was published. We proceeded to remove the *PublishedDate* variable from our dataset, because we didn't think it would affect our outcome variable, *growth_2_6*. Next, in addition to removing *PublishedDate*, we also removed the *id* variable, which simply denotes the index of each observation in the dataset. We removed *id* in order to avoid overfitting, because we knew it would be unhelpful in predicting *growth_2_6*.

Lastly, after removing the variables *id* and *PublishedDate*, we used the lapply() function to produce summary outputs of the remaining predictors, all of which were numeric. From the summary() outputs, we noticed that some variables appeared to contain only values of zero. Thus, because variables of all zeros could not meaningfully affect our outcome variable, we wrote a for() loop to extract the names of these 12 variables, and then we removed them from our training and test sets. So, overall, our data preprocessing for this project involved removing 14 variables from the original dataset.

2.2 Statistical Model

After preprocessing our data, we constructed our final model by first using bagging to identify the most important variables and, therefore, to reduce our chances of overfitting the data. Then, we constructed our final model by fitting this subset of important variables in a bagging model with 500 trees.

We performed feature selection by running the importance() function on a bagging model fit on all of the predictors in the preprocessed dataset, with mtry equal to the number of predictors (245) and number of trees set to 500. After investigating different thresholds for %IncMSE, where %IncMSE represents how much our model accuracy decreases if we leave out that variable, we found that a threshold of 8% gave the lowest RMSE. As a result, we reduced

the number of predictors from 245 to 32, and we then constructed a bagged model on the 32 important variables, with a mtry of 32 and with 500 trees.

Because we decided to use bagging as our final model, we chose not to center or scale our data — this is because the randomForest algorithm is based on decision trees instead of Euclidean distance. In other words, because trees are essentially a collection of partition rules with decisions based on a threshold, the results from bagging shouldn't be affected by scaling.

We further motivate our final model as our best model by comparing the validation RMSE of this model to the validation RMSEs of a few of our other models. A table of our top four models comparing the validation RMSEs is shown below in Figure 1. As evidenced by this table, our final model had the lowest RMSE of our top models.

Figure 1. Motivating performance of our model based on validation RMSE.

Model Description	RMSE*
Random Forest model (mtry = 83/3, ntree = 1000) with 83 important variables selected by LASSO	1.605992
Bagging model (mtry = 83, ntree = 500) with 83 important variables selected by LASSO	1.580627
Bagging model on 40 predictors selected by bagging all predictors and selecting the ones that exceed a %IncMSE of 6%	1.524178
Final Model: Bagging model on 32 predictors selected by bagging all predictors and selecting the ones that exceed a %IncMSE of 8%	**1.520741**

* The values of RMSE in the table above represent those obtained from the validation set we created using 50% of the provided training set. They do *not* reflect any RMSEs obtained on Kaggle.

3 Results

As described above, our final submission to Kaggle used bagging first for feature selection and then again to fit our final model. From the original 245 predictors in the cleaned training set, our final model used only 32 predictors. And, according to the public leaderboard on Kaggle, our final model's value for the RMSE metric was **1.40675**. Figure 2 below displays how the value of RMSE is computed.

$$RMSE = \sqrt{\frac{1}{n} \sum i = 1^{n} (gi - \hat{g}_i)^2}$$

Figure 2. Calculation of root mean squared error.

4 General Conclusions

We think our model works well because bagging, or bootstrap aggregating, aims to construct deep decision trees that have low bias but consequently high variance**.** However, the benefit of bootstrap sampling is that it combats this high variance to help avoid overfitting. Bootstrap sampling is a way of reducing multicollinearity within trees by showing them different training sets. In other words, the predictions from a single decision tree are very sensitive to noise from the training set. But, by averaging (or aggregating) many of these single trees, we can construct a model that is less sensitive to this noise. This works best when the trees are not correlated with each other because this correlation would bias how the trees decide the results. Thus, a strength of our final bagging model is that this decrease in variance serves to improve the predictive power of the model and results in a decreased (improved) RMSE, which is the metric we used to evaluate the performance of our model.

In addition to noting this strength of our model, we also note ways in which it can be improved. First, as described earlier in this report, we used bagging in order to perform feature selection. However, while bagging *can* be used for feature selection by indicating which predictors are the most important (based on %IncMSE), we acknowledge that we could have also used dimension reduction techniques to obtain a different set of predictors that may have improved the performance of our model. For example, as discussed in *An Introduction to Statistical Learning*, we could have used Principal Component Analysis (PCA). Specifically, PCA addresses multicollinearity and efficiently combines variables by constructing new, weighted groups of correlated predictors, thereby reducing dimensionality. In addition to implementing dimension reduction, we could also potentially improve our model by performing more rigorous data preprocessing. For example, we neglected to explore whether we should transform any of the original predictors, and some variable transformations may have helped improve the performance of our model. Additionally, we could've implemented some more advanced preprocessing methods including oversampling and undersampling.

All in all, our final model had an RMSE of 1.40675 on the public leaderboard and an RMSE of 1.41826 on the private leaderboard. One explanation for this 0.01151 increase in RMSE from the public leaderboard to the private leaderboard has to do with our list of important variables from our feature selection with the bagging method. Specifically, as evidenced in the plot in the Appendix on page 16, the variables *Num_Views_Base_mid_high*, *avg_growth_low_mid*, and *cnn_10* have %IncMSEs that are significantly larger than the %IncMSE for the rest of the predictors. As a result, these few variables likely appeared near the top of most of the trees, which may have caused many of the trees to vote in a similar way. As such, some of the trees may have become correlated, in a sense, which could have degraded the performance of bagging and may explain why our model did not perform as well on the private dataset. In other words, the few predictors deemed extremely important on the public dataset may not have been as important on the private dataset.

5 References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. New York: Springer.

Kübler, D. (2019, December 21). Understanding the Effect of Bagging on Variance and Bias visually. Retrieved December 13, 2020, from https://towardsdatascience.com/understanding-the-effect-of-bagging-on-variance-and-bias-visually-6131e6ff1385

Single estimator versus bagging: Bias-variance decomposition. (n.d.). Retrieved December 15, 2020, from https://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html

Vazquez, A. (2020). *Bagging, Random Forest and Boosting*. Lecture presented in Statistics 101C at UCLA.

Random forest. (2020, December 09). Retrieved December 14, 2020, from https://en.wikipedia.org/wiki/Random_forest

6 Statement of Contributions

Anna Piskun: For the first couple of days of the Kaggle competition, I was tasked with fitting multiple Bagging and Boosting models. I also fit multiple Random Forest and SVM models, to help rule out potential model possibilities. Likewise, I helped transition into focusing more on Bagging as our final method trying LASSO/Ridge Regression as potential variable selection methods to improve upon our Bagging model. For the report/presentation, I wrote the *Introduction*, and motivated why Bagging was the best model in the *General Conclusions* section. I also cleaned up all of the code and wrote the *Appendix* section of our report. As per last time, we remained in constant communication as a team helping each other out with code/editing each other's sections in the report.

Derek Wang: I wrote the for() loops to eliminate the columns of zeros for data preprocessing. During the beginning of the competition, I focused on fitting and submitting PCR and PLS models. My focus later shifted to working on improving our bagging model. I found that feature selection via the importance() function marginally improved our RMSE. For the report & presentation, I described the model in the *Statistical Model* section and motivated why our public RMSE differed from the private one, and why our model still performed fairly well in the

General Conclusions section. Additionally, I created the plot describing how the three most important predictors had much greater %IncMSE than the rest of the predictors.

Tomi Rajninger: Throughout the first few days of the final competition, I submitted multiple LASSO and ridge regression models to Kaggle. Then, once our group discovered that the bagging method was yielding the lowest values of RMSE, I transitioned to working to improve our bagging model and submitted a few more sets of predictions to Kaggle. During this time, I also cleaned up the code for our data preprocessing and helped my teammates debug their code as they encountered errors. Lastly, with regards to the report and presentation, I wrote and presented the *Data Preprocessing*, *Results* and part of the *General Conclusions* sections, and I also offered edits and feedback on the portions of the report/presentation completed by my teammates.

6 Appendix — R Code

 # Step 1: Load, Clean, and Preprocess Data

```
# load packages
library(dplyr)
library(randomForest)
library(glmnet)

# download training and test data
setwd("~/Desktop/stats101c-lec4-final-competition (1)")
training <- read.csv("training.csv")
test <- read.csv("test.csv")

# description of training dataset
# each row: youtube video
# each column: features of youtube video (thumbnail image, video title, channel, other)
# 7242 videos/rows and 260 columns (258 predictors because id and growth_2_6 are not predictors)
# response variable: growth_2_6 (num) = percent change in views b/w 2nd and 6th hour after publishing

# description of test dataset
# 3105 videos/rows and 259 columns
# excludes growth_2_6 variable because this is what we're trying to predict
```

Data Preprocessing: Remove *id*, *PublishedDate*, all columns of 0s

```
# Data Cleaning

# 1.  Check for any NA values in training and test sets
any(is.na(training))
any(is.na(test))
# both output FALSE, so don't need to remove any missing observations

#  2. Explore structure of dataset
str(training) # all columns except column 2, PublishedDate, are numeric
# we choose to remove id and PublishedDate from training and test sets frame as they are extra predictors (helps us to avoid overfitting)
training_clean <- select(training, -c(1, 2))
names(training_clean)
test_clean <- select(test, -c(1, 2))
names(test_clean)

# use lapply() to look at summaries for each of the variables in the training set
lapply(training_clean, summary)

#  3. Create for loop to identify the indices of empty columns (columns with only 0s)

cols_of_zero <- NA
for(i in 1:ncol(training_clean)){  # for each column in cleaned training set
if(sum(training_clean[,i]) == 0){  # if all values in the column are 0
cols_of_zero[i] <- i  # then specify their index in cols_of_zero object
}
}
cols_of_zero
good <- complete.cases(cols_of_zero)
columns_to_remove <- cols_of_zero[good]

# total of 12 columns of 0's to remove:
# 164 166 176 180 184 226 228 232 234 235 239 240

# 4. Remove the 12 columns of 0's

training_clean <- training_clean[, -columns_to_remove]
test_clean <- test_clean[, -columns_to_remove]
```

```
ncol(training) # original training set had 260 columns
ncol(training_clean) # training_clean has 246 columns = removed id, PublishedDate, 12 cols of 0s

ncol(test) # original test set had 259 columns (excludes growth_2_6 column)
ncol(test_clean) # test_clean has 246 columns = removed id, PublishedDate,  12 cols of 0s
```

 # Step 2: Split Data into Training and Validation Sets

```
# split training_clean data into training and validation sets (50/50 split)
# 7242/2 = 3621
# training and validation sets should each have 3621 observations

set.seed(123)
i <- seq_len(nrow(training_clean))
i_train <- sample(i, 3621, replace = FALSE)

youtube_train <- training_clean[i_train, ] # training set has 3621 observations
youtube_val <- training_clean[-i_train, ] #validation set has 3621 observations
```

 # Step 3: Model 1 - Random Forest + 83 LASSO Variables

```
# 1. Fit LASSO to extract important predictor variables

set.seed(123)
training_x <- model.matrix(growth_2_6 ~ ., data =
training_clean)[, -1] # remove intercept column
training_y <- training_clean$growth_2_6

# grid of possible values for lambda
i_exp <- seq(10, -2, length = 50) # from very large lambda to very small lambda
grid <- 10^i_exp
```

```
lasso_mod_val <- cv.glmnet(training_x, training_y, family =
"gaussian", alpha = 1, lambda = grid, standardize = TRUE, nfolds
= 10)

coef(lasso_mod_val) # the variables that have values in second column = important
```

LASSO important predictors (83 total)
Duration + hog_1 + hog_11 + hog_78 + hog_106 + hog_116 + hog_166 + hog_182 + hog_195 + hog_215 + hog_241 + hog_279 + hog_295 + hog_304 + hog_316 + hog_351 + hog_359 + hog_402 + hog_454 + hog_476 + hog_492 + hog_640 + hog_641 + hog_649 + hog_665 + hog_668 + hog_675 + hog_702 + hog_705 + hog_716 + hog_738 + hog_755 + hog_791 + hog_797 + hog_815 + hog_832 + hog_849 + hog_855 + cnn_9 + cnn_10 + cnn_12 + cnn_17 + cnn_25 + cnn_36 + cnn_39 + cnn_88 + cnn_89 + max_green + mean_blue + doc2vec_0 + doc2vec_2 + doc2vec_3 + doc2vec_4 + doc2vec_6 + doc2vec_7 + doc2vec_10 + doc2vec_11 + doc2vec_12 + doc2vec_13 + doc2vec_15 + doc2vec_17 + punc_num_..1 + punc_num_..3 + punc_num_..8 + punc_num_..11 + punc_num_..14 + punc_num_..15 + punc_num_..21 + punc_num_..28 + num_words + num_chars + num_stopwords + num_uppercase_chars + num_uppercase_words + num_digit_chars + Num_Subscribers_Base_low_mid + Num_Views_Base_low_mid + Num_Views_Base_mid_high + avg_growth_low + avg_growth_low_mid + avg_growth_mid_high + count_vids_low + count_vids_low_mid

2. Fit Random Forest Model (mtry = 83/3) to Training Data (N = 1000)

Fit on youtube_train to compare to youtube_val in order to calculate the RMSE

```
rf_lasso <- randomForest(growth_2_6~Duration + hog_1 + hog_11 +
hog_78 + hog_106 + hog_116 + hog_166 + hog_182 + hog_195 +
hog_215 + hog_241 + hog_279 + hog_295 + hog_304 + hog_316 +
hog_351 + hog_359 + hog_402 + hog_454 + hog_476 + hog_492 +
hog_640 + hog_641 + hog_649 + hog_665 + hog_668 + hog_675 +
hog_702 + hog_705 + hog_716 + hog_738 + hog_755 + hog_791 +
hog_797 + hog_815 + hog_832 + hog_849 + hog_855 + cnn_9 + cnn_10
+ cnn_12 + cnn_17 + cnn_25 + cnn_36 + cnn_39 + cnn_88 + cnn_89 +
max_green + mean_blue + doc2vec_0 + doc2vec_2 + doc2vec_3 +
doc2vec_4 + doc2vec_6 + doc2vec_7 + doc2vec_10 + doc2vec_11 +
doc2vec_12 + doc2vec_13 + doc2vec_15 + doc2vec_17 + punc_num_..1
+ punc_num_..3 + punc_num_..8 + punc_num_..11 + punc_num_..14
+ punc_num_..15 + punc_num_..21 + punc_num_..28 + num_words +
num_chars + num_stopwords + num_uppercase_chars +
num_uppercase_words + num_digit_chars +
Num_Subscribers_Base_low_mid + Num_Views_Base_low_mid +
```

```
Num_Views_Base_mid_high + avg_growth_low + avg_growth_low_mid +
avg_growth_mid_high + count_vids_low + count_vids_low_mid, data
= youtube_train, mtry = 83/3, ntree = 700, importance = T)

# Run predictions on validation set

rf_pred <- predict(rf_lasso, youtube_val)

# Calculate RMSE
sqrt(mean((youtube_val$growth_2_6 - rf_pred)^2))

# validation RMSE = 1.605992

# 3. Fit Random Forest Model w/ 83 LASSO Variables on full Training Set

rf_lasso.final <- randomForest(growth_2_6~Duration + hog_1 +
hog_11 + hog_78 + hog_106 + hog_116 + hog_166 + hog_182 +
hog_195 + hog_215 + hog_241 + hog_279 + hog_295 + hog_304 +
hog_316 + hog_351 + hog_359 + hog_402 + hog_454 + hog_476 +
hog_492 + hog_640 + hog_641 + hog_649 + hog_665 + hog_668 +
hog_675 + hog_702 + hog_705 + hog_716 + hog_738 + hog_755 +
hog_791 + hog_797 + hog_815 + hog_832 + hog_849 + hog_855 +
cnn_9 + cnn_10 + cnn_12 + cnn_17 + cnn_25 + cnn_36 + cnn_39 +
cnn_88 + cnn_89 + max_green + mean_blue + doc2vec_0 + doc2vec_2
+ doc2vec_3 + doc2vec_4 + doc2vec_6 + doc2vec_7 + doc2vec_10 +
doc2vec_11 + doc2vec_12 + doc2vec_13 + doc2vec_15 + doc2vec_17 +
punc_num_..1 + punc_num_..3 + punc_num_..8 +  punc_num_..11 +
punc_num_..14 +  punc_num_..15 +  punc_num_..21 +  punc_num_..28
+ num_words + num_chars + num_stopwords + num_uppercase_chars +
num_uppercase_words + num_digit_chars +
Num_Subscribers_Base_low_mid + Num_Views_Base_low_mid +
Num_Views_Base_mid_high + avg_growth_low + avg_growth_low_mid +
avg_growth_mid_high + count_vids_low + count_vids_low_mid, data
= training_clean, mtry = 83/3, ntree = 1000, importance = T)

# predictions for growth_2_6
rf_pred_test <- predict(rf_lasso.final, test[, -247])

# output .csv submission file with just id and our predictions of growth_2_6
submission5_ap <- data.frame("id" = test$id, "growth_2_6" =
rf_pred_test)
```

```
write.csv(submission5_ap, file = "submission5_ap.csv", row.names
= FALSE)
```

 # Step 4: Model 2 - Bagging + 83 LASSO Variables

1. Using the same 83 important LASSO variables as above, fit bagging model on training set and predict with validation set (N = 500)

```
bag_mod_val <- randomForest(growth_2_6 ~ Duration + hog_1 +
hog_11 + hog_78 + hog_106 + hog_116 + hog_166 + hog_182 +
hog_195 + hog_215 + hog_241 + hog_279 + hog_295 + hog_304 +
hog_316 + hog_351 + hog_359 + hog_402 + hog_454 + hog_476 +
hog_492 + hog_640 + hog_641 + hog_649 + hog_665 + hog_668 +
hog_675 + hog_702 + hog_705 + hog_716 + hog_738 + hog_755 +
hog_791 + hog_797 + hog_815 + hog_832 + hog_849 + hog_855 +
cnn_9 + cnn_10 + cnn_12 + cnn_17 + cnn_25 + cnn_36 + cnn_39 +
cnn_88 + cnn_89 + max_green + mean_blue + doc2vec_0 + doc2vec_2
+ doc2vec_3 + doc2vec_4 + doc2vec_6 + doc2vec_7 + doc2vec_10 +
doc2vec_11 + doc2vec_12 + doc2vec_13 + doc2vec_15 + doc2vec_17 +
punc_num_..1 + punc_num_..3 + punc_num_..8 +  punc_num_..11 +
punc_num_..14 +  punc_num_..15 +  punc_num_..21 +  punc_num_..28
+ num_words + num_chars + num_stopwords + num_uppercase_chars +
num_uppercase_words + num_digit_chars +
Num_Subscribers_Base_low_mid + Num_Views_Base_low_mid +
Num_Views_Base_mid_high + avg_growth_low + avg_growth_low_mid +
avg_growth_mid_high + count_vids_low + count_vids_low_mid, data
= youtube_train, mtry = 83, ntree = 500, importance = TRUE)

# make predictions on validation set
bag_preds_val <- predict(bag_mod, youtube_val)

# compute validation RMSE
val_RMSE <- sqrt(mean((bag_preds_val -
youtube_val$growth_2_6)^2))
val_RMSE

# validation RMSE = 1.580627
```

2. Fit the same bagging model with full training_clean set to make predictions on final test set

```
bag_mod_test <- randomForest(growth_2_6 ~ Duration + hog_1 +
hog_11 + hog_78 + hog_106 + hog_116 + hog_166 + hog_182 +
hog_195 + hog_215 + hog_241 + hog_279 + hog_295 + hog_304 +
hog_316 + hog_351 + hog_359 + hog_402 + hog_454 + hog_476 +
hog_492 + hog_640 + hog_641 + hog_649 + hog_665 + hog_668 +
hog_675 + hog_702 + hog_705 + hog_716 + hog_738 + hog_755 +
hog_791 + hog_797 + hog_815 + hog_832 + hog_849 + hog_855 +
cnn_9 + cnn_10 + cnn_12 + cnn_17 + cnn_25 + cnn_36 + cnn_39 +
cnn_88 + cnn_89 + max_green + mean_blue + doc2vec_0 + doc2vec_2
+ doc2vec_3 + doc2vec_4 + doc2vec_6 + doc2vec_7 + doc2vec_10 +
doc2vec_11 + doc2vec_12 + doc2vec_13 + doc2vec_15 + doc2vec_17 +
punc_num_..1 + punc_num_..3 + punc_num_..8 +  punc_num_..11 +
punc_num_..14 +  punc_num_..15 +  punc_num_..21 +  punc_num_..28
+ num_words + num_chars + num_stopwords + num_uppercase_chars +
num_uppercase_words + num_digit_chars +
Num_Subscribers_Base_low_mid + Num_Views_Base_low_mid +
Num_Views_Base_mid_high + avg_growth_low + avg_growth_low_mid +
avg_growth_mid_high + count_vids_low + count_vids_low_mid, data
= training_clean, mtry = 83, ntree = 500, importance = TRUE)

# make predictions on test set
bag_preds_test <- predict(bag_mod_test, test_clean)

# create csv file for kaggle submission
t_sub_3 <- data.frame("id" = test$id, "growth_2_6" =
bag_preds_test)
write.csv(t_sub_3, file = "t_sub_3.csv", row.names = FALSE)
```

 # Step 5: Model 3 - Bagging + 48 Variables Bagging (%IncMSE of 6%)

```
# 1. Determine variable importance using bagging

#Fit bagging model with 500 trees on full cleaned training set. Then, identify the most important
predictors from this model

set.seed(123)
bag_mod_vars <- randomForest(growth_2_6 ~ ., data =
training_clean, mtry = 245, ntree = 500, importance = TRUE) #mtry =
245 because training_clean has 246 columns (incl. growth_2_6)
```

```
# make our object (class: randomForest) a data frame so we can manipulate it
important_variables <- as.data.frame(importance(bag_mod_vars))

# order the importance from greatest to least
sorted_important_variables <-
important_variables[order(important_variables[, 1], decreasing =
TRUE),]

#choose variables above %IncMSE of 6%
above_6_percent <-
sorted_important_variables[sorted_important_variables[, 1] > 6,
]
above_6_percent_names <- above_6_percent[, 0]
above_6_percent_names # input these predictors into the model in the next code chunk

# note: there are 40 important predictors that we want to include in our model

# 2. Fit model using bagging and N = 500, predict on validation data

bag_mod_val.1 <- randomForest(growth_2_6 ~
Num_Views_Base_mid_high + avg_growth_low_mid + cnn_10 +
avg_growth_low + cnn_86 + Num_Subscribers_Base_low_mid + cnn_89
+ Num_Subscribers_Base_mid_high + views_2_hours + cnn_25 +
cnn_12    + avg_growth_mid_high + cnn_17 + count_vids_low_mid +
punc_num_..21 + count_vids_mid_high + num_digit_chars + cnn_88+
cnn_68 + num_uppercase_chars + Duration + cnn_19 + hog_183 +
punc_num_..28 + mean_green + mean_pixel_val + punc_num_..12 +
hog_341 + count_vids_low + num_chars + hog_454 + sd_blue +
punc_num_..1 + num_words + mean_blue + mean_red + doc2vec_10 +
sd_green + hog_452 + hog_40, data = youtube_train, mtry = 40,
ntree = 500, importance = TRUE)

bag_preds_val.1 <- predict(bag_mod_val.1, youtube_val)

# compute validation RMSE
val_RMSE.1 <- sqrt(mean((bag_preds_val.1 -
youtube_val$growth_2_6)^2))
val_RMSE.1

# val_RMSE.1 = 1.524178
```

 # Step 6: Model 4 - Bagging + 32 Variables Bagging (%IncMSE of 8%)

```
# 1. Extract important variables using a threshold of  %IncMSE of 8%
# Use importance_variables data frame from earlier

important_variables <- as.data.frame(importance(bag_mod_vars))

# order the importance from greatest to least
sorted_important_variables <-
important_variables[order(important_variables[, 1], decreasing =
TRUE),]

above_8_percent <-
sorted_important_variables[sorted_important_variables[, 1] > 8,]
above_8_percent_names <- above_8_percent[, 0]
above_8_percent_names # input these predictors into the model in the next code chunk

# note: there are 32 important predictors

# 2. Fit final bagging model on 32 predictors (8% %IncMSE threshold) on 500 trees.

set.seed(321)
bag_mod_val <- randomForest(growth_2_6 ~ Num_Views_Base_mid_high
+ avg_growth_low_mid + cnn_10 + avg_growth_low + cnn_86 +
Num_Subscribers_Base_low_mid+ cnn_89 +
Num_Subscribers_Base_mid_high + views_2_hours + cnn_25 + cnn_12
+ avg_growth_mid_high + cnn_17 + count_vids_low_mid +
punc_num_..21 + count_vids_mid_high + num_digit_chars + cnn_88 +
cnn_68 + num_uppercase_chars + Duration + cnn_19 + hog_183 +
punc_num_..28 + mean_green    + mean_pixel_val + punc_num_..12 +
hog_341 + count_vids_low + num_chars + hog_454 + sd_blue, data =
youtube_train, mtry = 32, ntree = 500, importance = TRUE)

bag_preds_val <- predict(bag_mod_val, youtube_val)

# compute validation RMSE
val_RMSE <- sqrt(mean((bag_preds_val -
youtube_val$growth_2_6)^2))
val_RMSE
```

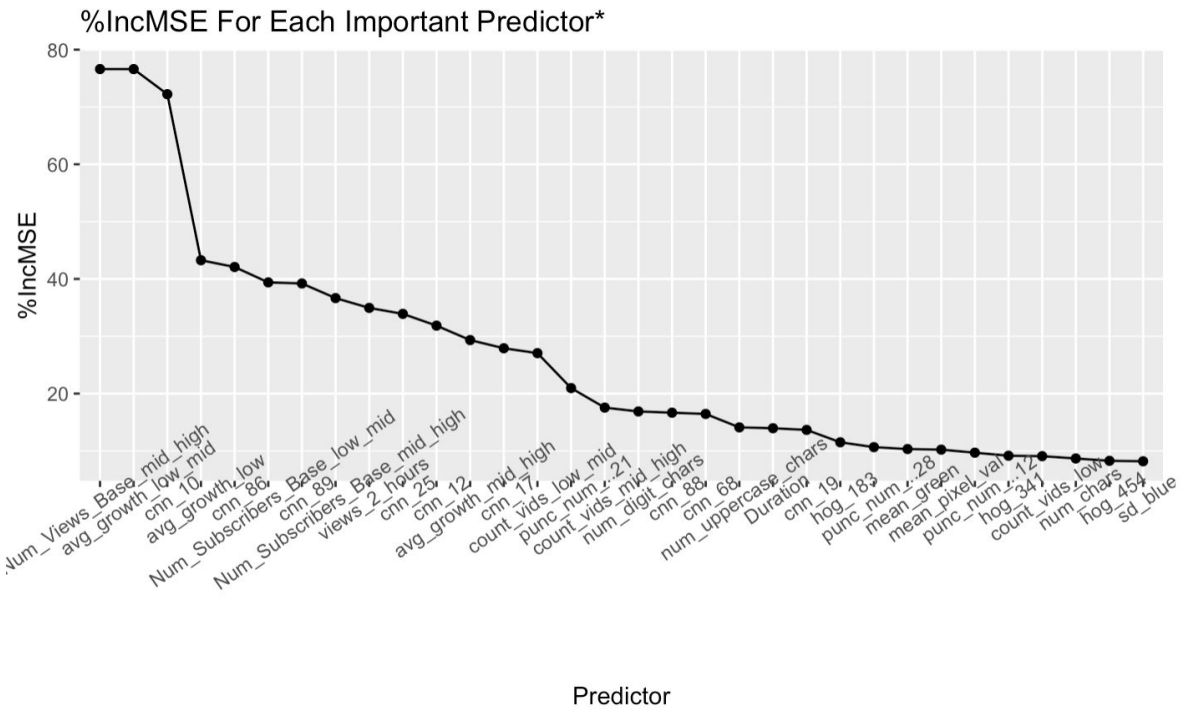
#val_RMSE = 1.520741

Step 7: FINAL MODEL

Fit final bagging model with 32 important predictor variables (%IncMSE 8%), and N = 500 on full, cleaned training dataset.

```
set.seed(321)
bag_mod_test <- randomForest(growth_2_6 ~
Num_Views_Base_mid_high + avg_growth_low_mid + cnn_10 +
avg_growth_low + cnn_86 + Num_Subscribers_Base_low_mid+ cnn_89 +
Num_Subscribers_Base_mid_high + views_2_hours + cnn_25 + cnn_12
+ avg_growth_mid_high + cnn_17 + count_vids_low_mid +
punc_num_..21 + count_vids_mid_high + num_digit_chars + cnn_88 +
cnn_68 + num_uppercase_chars + Duration + cnn_19 + hog_183 +
punc_num_..28 + mean_green + mean_pixel_val + punc_num_..12 +
hog_341 + count_vids_low + num_chars + hog_454 + sd_blue, data =
training_clean, mtry = 32, ntree = 500, importance = TRUE)

#Predict on test dataset
bag_preds_test <- predict(bag_mod_test, test_clean)

# create csv file for kaggle submission
a_sub_10 <- data.frame("id" = test$id, "growth_2_6" =
bag_preds_test)
write.csv(a_sub_10, file = "a_sub_10.csv", row.names = FALSE)


# code for the plot showing the different %IncMSE for each predictor
barplotdata_names <- as.data.frame(above_8_percent[,0])
percent_IncMSE <- above_8_percent[,1]
barplotdata<- cbind(barplotdata_names, percent_IncMSE)
barplotdata

plot(barplotdata)

library(data.table)
a <- setDT(barplotdata, keep.rownames = TRUE)[]

dose <- unlist(a[,1])
Percent_IncMSE <- unlist(a[,2])
```

```
#Turn your 'treatment' column into a character vector
dose <- as.character(dose)
#Then turn it back into a factor with the levels in the correct order
dose <- factor(dose, levels=unique(dose))

# library(ggplot2)
# Basic line plot with points
ggplot(data = a, aes(x=dose, y=len, xlab("yes"),group=1)) +
  geom_line()+
  geom_point() + theme(axis.text.x = element_text(angle = 35))+
  labs(title = "%IncMSE For Each Important Predictor*", x =
"Predictor", y = "%IncMSE")
```



* at a 8% IncMSE threshold