Stats 101C Final Kaggle Report

Team: Coffeeˆ2 (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.

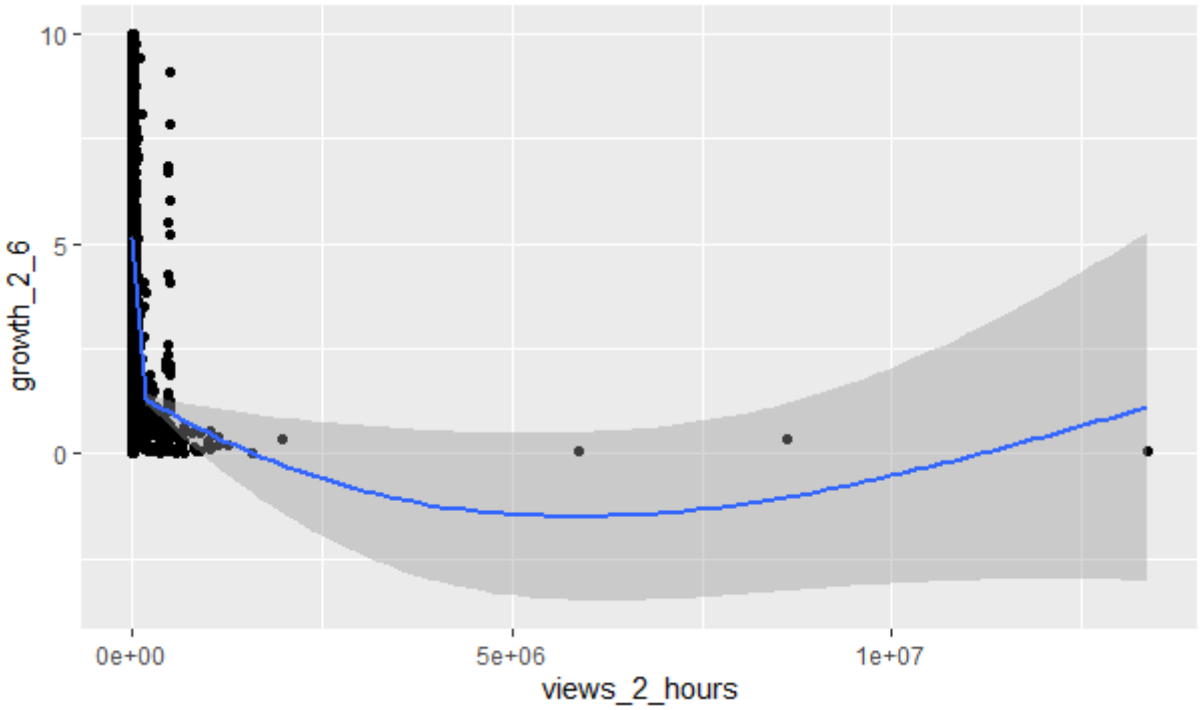


Figure 1: Data Exploration on views_2_hours

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.

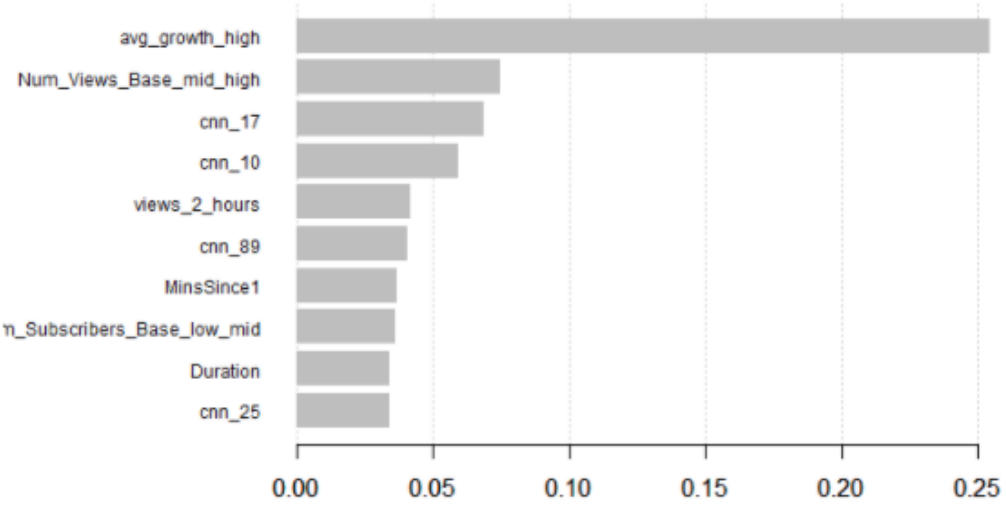


Figure 2: Variable Importance Plot

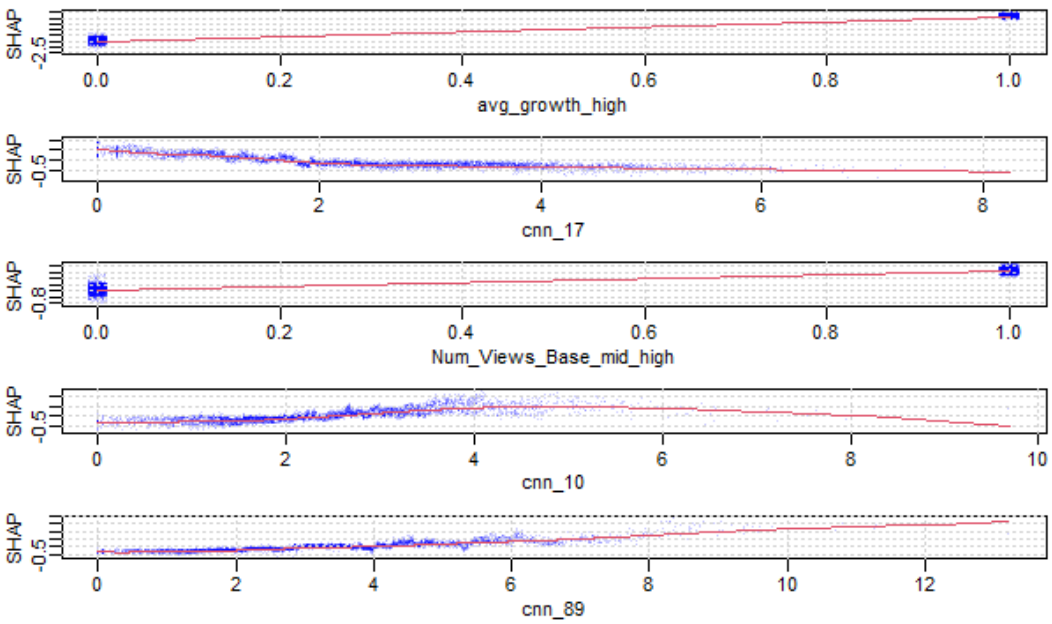


Figure 3: SHAP Plot for Top 5 Predictors

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

AllTheCode

load packages

```
# feature exploration engineering
library(tidyverse) # ggplot, dplyr
library(lubridate) # month()
library(gridExtra) # grid.arrange()
library(caret) # dummyVars()
# subset selection
library(glmnet) # lasso
library(randomForest) # for checking importance()
library(coefplot) # extract.coef()
# model
library(xgboost)
```

load data

```
train <- read_csv('training.csv')
test <- read_csv('test.csv')
test_id = test$id
```

preprocessing

creating high levels for Channel Features

```
attach(train)
train$Num_Subscribers_Base_high <- as.integer(Num_Subscribers_Base_low == 0 & Num_Subscribers_Base_low_m
train$Num_Views_Base_high <- as.integer(Num_Views_Base_low == 0 & Num_Views_Base_low_mid == 0 & Num_Viev
train$avg_growth_high <- as.integer(avg_growth_low == 0 & avg_growth_low_mid == 0 & avg_growth_mid_high
train$count_vids_high <- as.integer(count_vids_low == 0 & count_vids_low_mid == 0 & count_vids_mid_high
detach(train)
attach(test)
test$Num_Subscribers_Base_high <- as.integer(Num_Subscribers_Base_low == 0 & Num_Subscribers_Base_low_mi
test$Num_Views_Base_high <- as.integer(Num_Views_Base_low == 0 & Num_Views_Base_low_mid == 0 & Num_Views
test$count_vids_high <- as.integer(count_vids_low == 0 & count_vids_low_mid == 0 & count_vids_mid_high =
test$avg_growth_high <- as.integer(avg_growth_low == 0 & avg_growth_low_mid == 0 & avg_growth_mid_high =
detach(test)
```

converting datetime to numeric (and time-based features)

```
# split PublishedDate into date and time
train <- train %>% separate(PublishedDate,c("PublishedDate","PublishedTime"),sep = " ")
test <- test %>% separate(PublishedDate,c("PublishedDate","PublishedTime"),sep = " ")
train$PublishedDate <- strptime(train$PublishedDate, "%m/%d/%Y")
```

```
train$PublishedTime <- strptime(train$PublishedTime, "%H:%M")
test$PublishedDate <- strptime(test$PublishedDate, "%m/%d/%Y")
test$PublishedTime <- strptime(test$PublishedTime, "%H:%M")

# convert time to mins since midnight (numeric)
MinsSince1 <- as.integer(train$PublishedTime - min(train$PublishedTime))/60 # store as minutes
train <- cbind(MinsSince1,train)
MinsSince1 <- as.integer(test$PublishedTime - min(test$PublishedTime))/60 # store as minutes
test <- cbind(MinsSince1,test)
# and date to days since first day (numeric)
DaysSince1 <- as.integer(train$PublishedDate - min(train$PublishedDate))/86400 # store as days
train <- cbind(DaysSince1,train)
DaysSince1 <- as.integer(test$PublishedDate - min(test$PublishedDate))/86400 # store as days
test <- cbind(DaysSince1,test)

# creat hour bins (4 hours per bin)
train$bin_hour <- as.factor(train$MinsSince1 %/% 240)
test$bin_hour <- as.factor(test$MinsSince1 %/% 240)
# one-hot encode hour bins
dmy <- dummyVars(" ~ bin_hour", data = train)
trsf <- data.frame(predict(dmy, newdata = train))
train <- cbind(train,trsf)
dmy <- dummyVars(" ~ bin_hour", data = test)
trsf <- data.frame(predict(dmy, newdata = test))
test <- cbind(test,trsf)

# create month
train$month <-  as.factor(month(as.Date(train$PublishedDate)))
test$month <-  as.factor(month(as.Date(test$PublishedDate)))
# one-hot encode month
dmy <- dummyVars(" ~ month", data = train)
trsf <- data.frame(predict(dmy, newdata = train))
train <- cbind(train,trsf)
dmy <- dummyVars(" ~ month", data = test)
trsf <- data.frame(predict(dmy, newdata = test))
test <- cbind(test,trsf)
train$month <-  as.integer(train$month)
test$month <-  as.integer(test$month)

# day of week
train$day_of_week <- weekdays(as.Date(train$PublishedDate))
test$day_of_week <- weekdays(as.Date(test$PublishedDate))
# one hot encode DoW
dmy <- dummyVars(" ~ day_of_week", data = train)
trsf <- data.frame(predict(dmy, newdata = train))
train <- cbind(train,trsf)
dmy <- dummyVars(" ~ day_of_week", data = test)
trsf <- data.frame(predict(dmy, newdata = test))
test <- cbind(test,trsf)
```

creating title features

```
# avg word length
train$avg_word_length <- train$num_chars/train$num_words
test$avg_word_length <- test$num_chars/test$num_words
# number of non-stopwords
train$num_non_stopwords <- train$num_words - train$num_stopwords
test$num_non_stopwords <- test$num_words - test$num_stopwords
# proportion of non-stopwords
train$p_non_stopwords <- train$num_non_stopwords/train$num_words
test$p_non_stopwords <- test$num_non_stopwords/test$num_words
# proportion of digit chars
train$p_digit_chars <- train$num_digit_chars/train$num_chars
test$p_digit_chars <- test$num_digit_chars/test$num_chars
# whether a title has a word with multiple capital letters
train$all_cap <- as.integer(train$num_uppercase_chars > train$num_uppercase_words) # (more uppercase ch
test$all_cap <- as.integer(test$num_uppercase_chars > test$num_uppercase_words)
```

converting some punctuation to binary

the only punctuation predictors in our final model were: punc_num_at punc_num_bar_bi and punc_num_com

```
# had some difficulties with special characters, so renamed them
train <- train %>% rename(`punc_num_fs` = "punc_num_/",
                          `punc_num_bs` = "punc_num_\\",
                          `punc_num_col` = "punc_num_:",
                          `punc_num_bar` = "punc_num_|",
                          `punc_num_exc` = "punc_num_!",
                          `punc_num_num` = "punc_num_#",
                          `punc_num_pls` = "punc_num_+",
                          `punc_num_par_l` = "punc_num_(",
                          `punc_num_par_r` = "punc_num_)",
                          `punc_num_at` = "punc_num_@",
                          `punc_num_eql` = "punc_num_=",
                          `punc_num_q` = "punc_num_?",
                          `punc_num_dol` = "punc_num_$",
                          `punc_num_pct` = "punc_num_%",
                          `punc_num_com` = "punc_num_,",
                          `punc_num_semi` = "punc_num_;",
                          `punc_num_dash` = "punc_num_-",
                          `punc_num_dot` = "punc_num_.",
                          `punc_num_brk_l` = "punc_num_[",
                          `punc_num_crl_l` = "punc_num_{",
                          `punc_num_brk_r` = "punc_num_]",
                          `punc_num_crl_r` = "punc_num_}",
                          `punc_num_leq` = "punc_num_<",
                          `punc_num_geq` = "punc_num_>",
                          `punc_num_crt` = "punc_num_^",
                          `punc_num_and` = "punc_num_&",
                          `punc_num_dquo` = 'punc_num_"',
                          `punc_num_squo` = "punc_num_'",
                          `punc_num_bktk` = "punc_num_`",
                          `punc_num_ast` = "punc_num_*",
```

```
                          `punc_num_tld` = "punc_num_~"
                          )
# names(train)[235]
test <- test %>% rename(`punc_num_fs` = "punc_num_/",
                        `punc_num_bs` = "punc_num_\\",
                        `punc_num_col` = "punc_num_:",
                        `punc_num_bar` = "punc_num_|",
                        `punc_num_exc` = "punc_num_!",
                        `punc_num_num` = "punc_num_#",
                        `punc_num_pls` = "punc_num_+",
                        `punc_num_par_l` = "punc_num_(",
                        `punc_num_par_r` = "punc_num_)",
                        `punc_num_at` = "punc_num_@",
                        `punc_num_eql` = "punc_num_=",
                        `punc_num_q` = "punc_num_?",
                        `punc_num_dol` = "punc_num_$",
                        `punc_num_pct` = "punc_num_%",
                        `punc_num_com` = "punc_num_,",
                        `punc_num_semi` = "punc_num_;",
                        `punc_num_dash` = "punc_num_-",
                        `punc_num_dot` = "punc_num_.",
                        `punc_num_brk_l` = "punc_num_[",
                        `punc_num_crl_l` = "punc_num_{",
                        `punc_num_brk_r` = "punc_num_]",
                        `punc_num_crl_r` = "punc_num_}",
                        `punc_num_leq` = "punc_num_<",
                        `punc_num_geq` = "punc_num_>",
                        `punc_num_crt` = "punc_num_^",
                        `punc_num_and` = "punc_num_&",
                        `punc_num_dquo` = 'punc_num_"',
                        `punc_num_squo` = "punc_num_'",
                        `punc_num_bktk` = "punc_num_`",
                        `punc_num_ast` = "punc_num_*",
                        `punc_num_tld` = "punc_num_~"
                        )
# we found these values by checking whether there was a significant difference
# in growth_2_6 for the most common punctuations (so cutoff somewhat subjective)
train$punc_num_bar_bi <- as.integer(train$punc_num_bar == 1) # eq 1 is opt
test$punc_num_bar_bi <- as.integer(test$punc_num_bar == 1) # eq 1 is opt
```

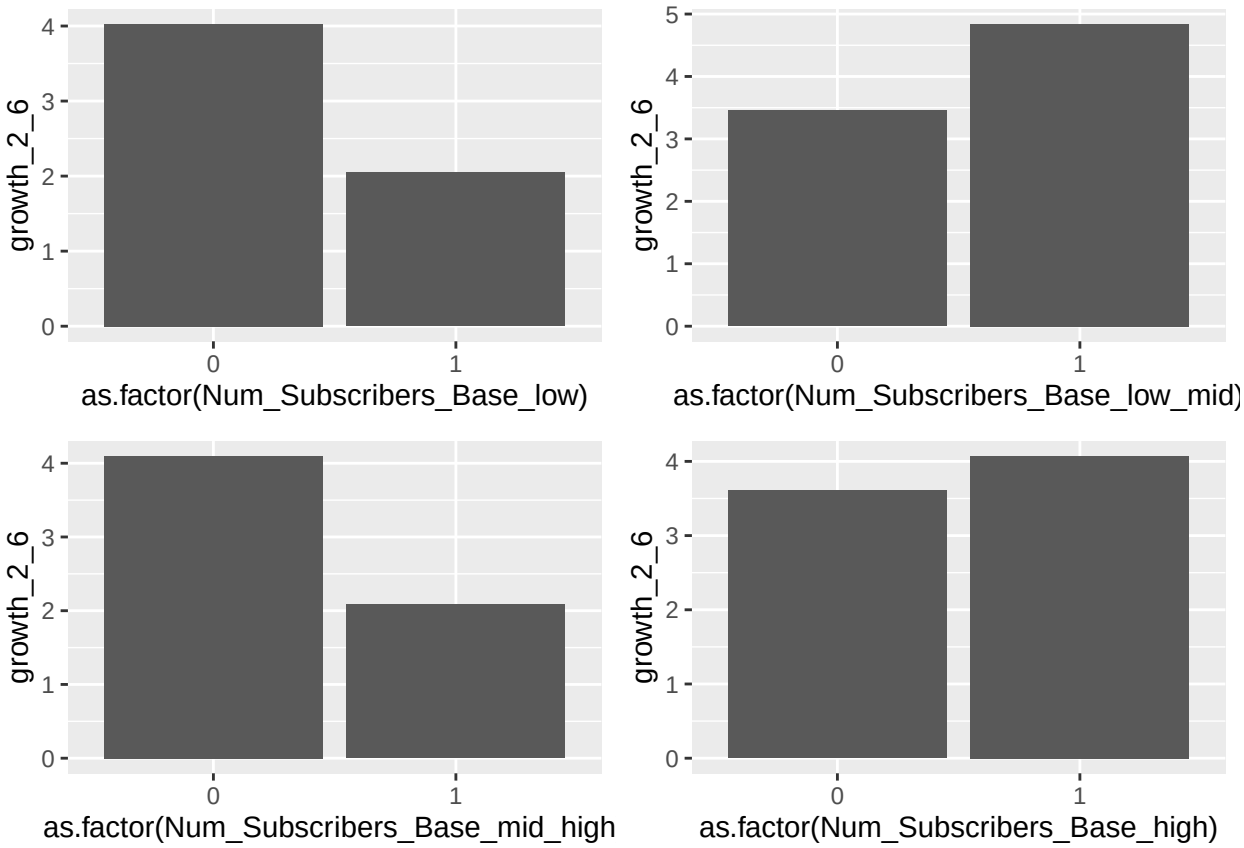
other features

```
# convert duration to binary
train$Duration2 <- as.integer(train$Duration > 6000)
test$Duration2 <- as.integer(test$Duration > 6000)
```

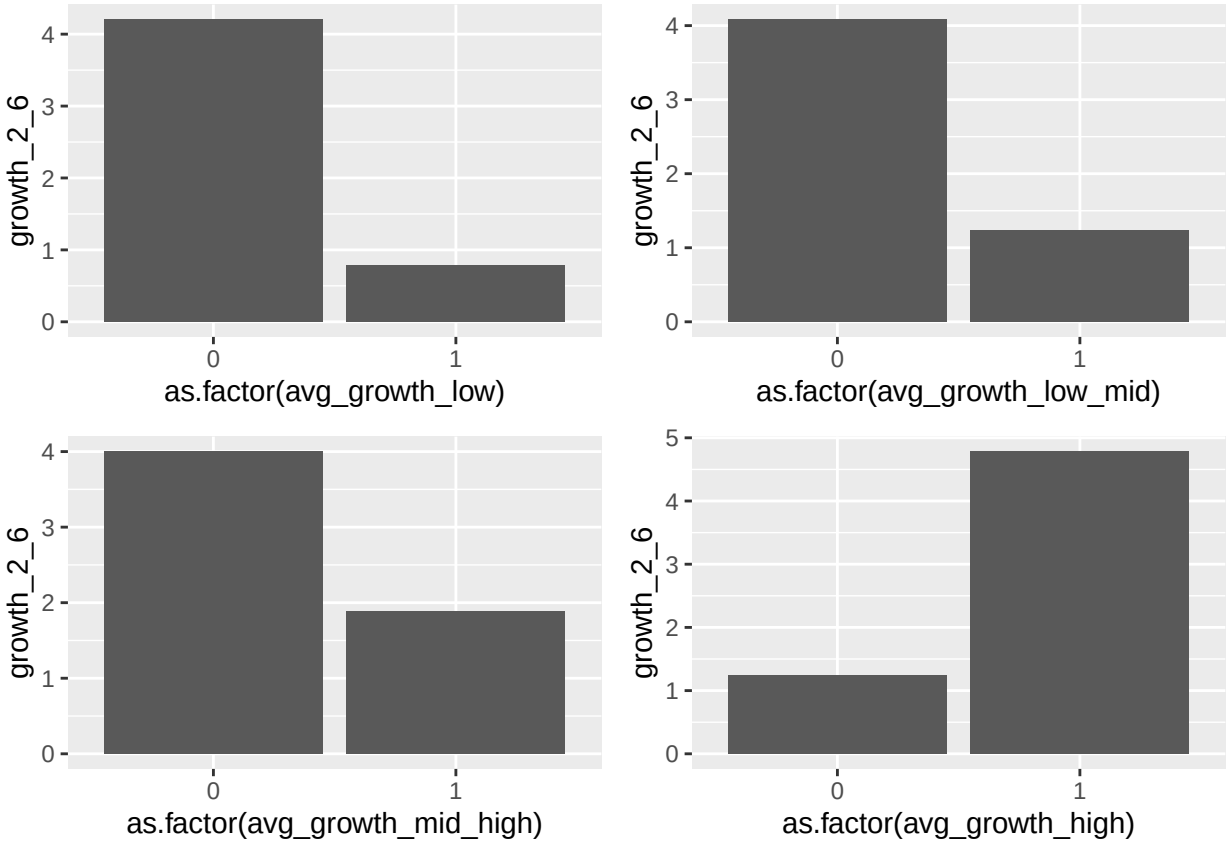
exploring features

```
# subscribers
p1 <- ggplot(train,aes(x=as.factor(Num_Subscribers_Base_low),y=growth_2_6)) + stat_summary(fun = "mean"
p2 <- ggplot(train,aes(x=as.factor(Num_Subscribers_Base_low_mid),y=growth_2_6)) + stat_summary(fun = "m
p3 <- ggplot(train,aes(x=as.factor(Num_Subscribers_Base_mid_high),y=growth_2_6)) + stat_summary(fun = "
```

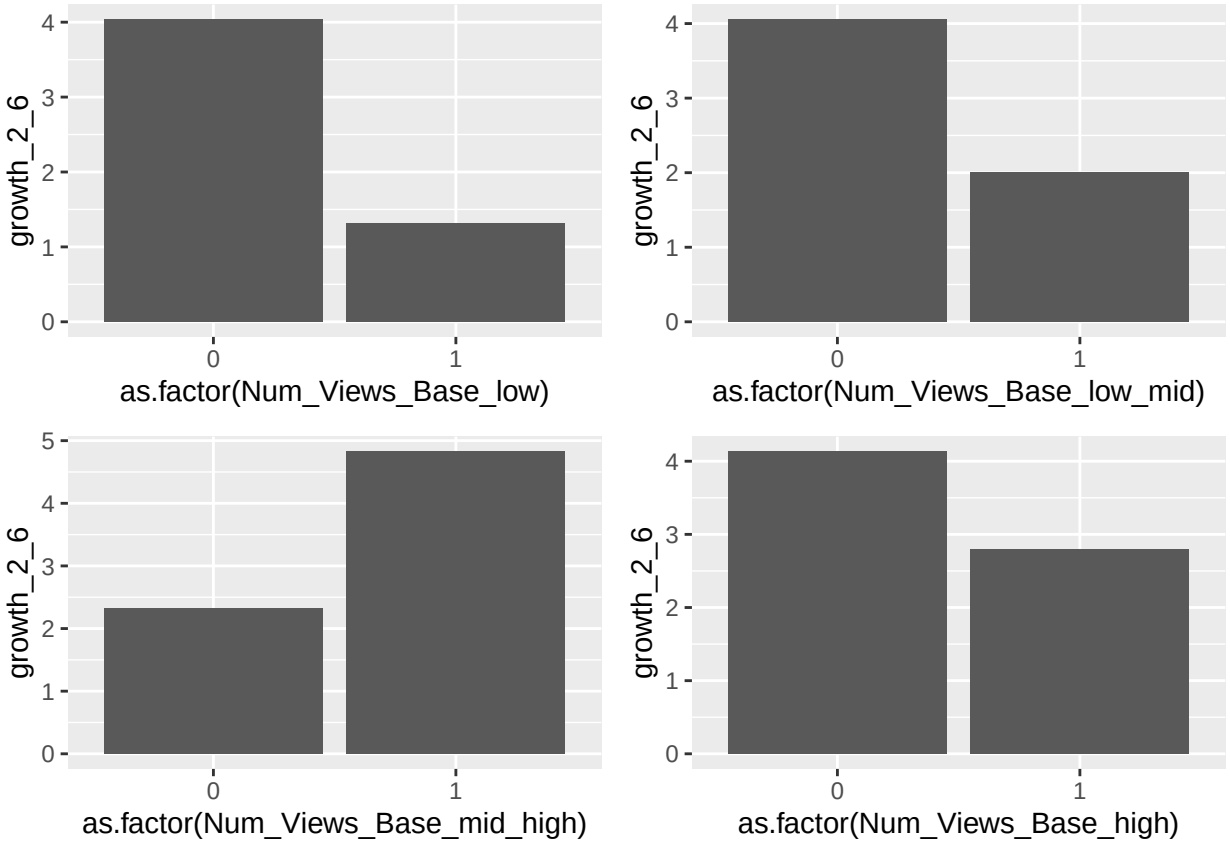
```
p4 <- ggplot(train,aes(x=as.factor(Num_Subscribers_Base_high),y=growth_2_6)) + stat_summary(fun = "mean"
grid.arrange(p1,p2,p3,p4,nrow=2)
```




```
# average growth
p1 <- ggplot(train,aes(x=as.factor(avg_growth_low),y=growth_2_6)) + stat_summary(fun = "mean", geom = "
p2 <- ggplot(train,aes(x=as.factor(avg_growth_low_mid),y=growth_2_6)) + stat_summary(fun = "mean", geom
p3 <- ggplot(train,aes(x=as.factor(avg_growth_mid_high),y=growth_2_6)) + stat_summary(fun = "mean", geo
p4 <- ggplot(train,aes(x=as.factor(avg_growth_high),y=growth_2_6)) + stat_summary(fun = "mean", geom =
grid.arrange(p1,p2,p3,p4,nrow=2) # growth highest for high avg_growth
```

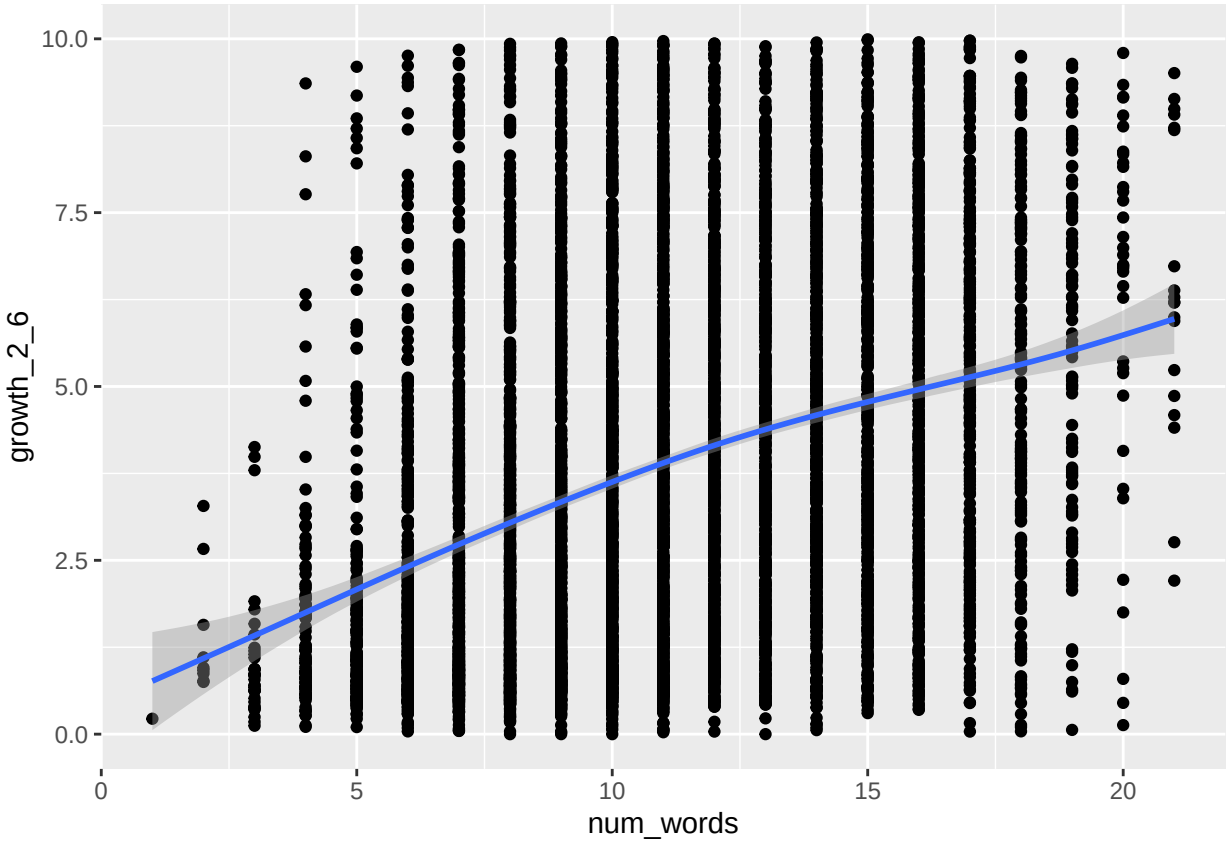



```
# number of views on channel
p1 <- ggplot(train,aes(x=as.factor(Num_Views_Base_low),y=growth_2_6)) + stat_summary(fun = "mean", geom
p2 <- ggplot(train,aes(x=as.factor(Num_Views_Base_low_mid),y=growth_2_6)) + stat_summary(fun = "mean", g
p3 <- ggplot(train,aes(x=as.factor(Num_Views_Base_mid_high),y=growth_2_6)) + stat_summary(fun = "mean",
p4 <- ggplot(train,aes(x=as.factor(Num_Views_Base_high),y=growth_2_6)) + stat_summary(fun = "mean", geo
grid.arrange(p1,p2,p3,p4,nrow=2) # growth highest for Num_Views_Base_mid_high (too high is not good)
```



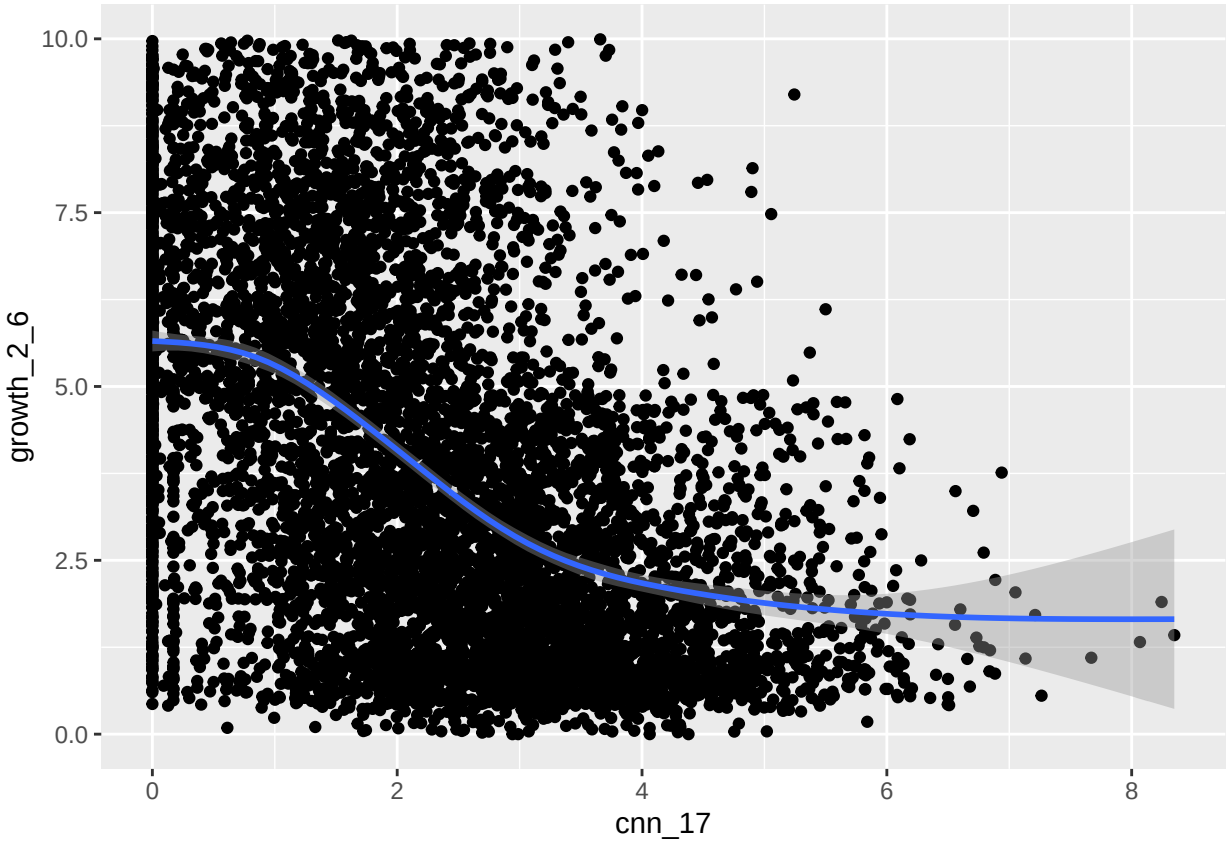

```
# some other significant features
ggplot(train,aes(x=num_words,y=growth_2_6)) + geom_point() + geom_smooth() # num words
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



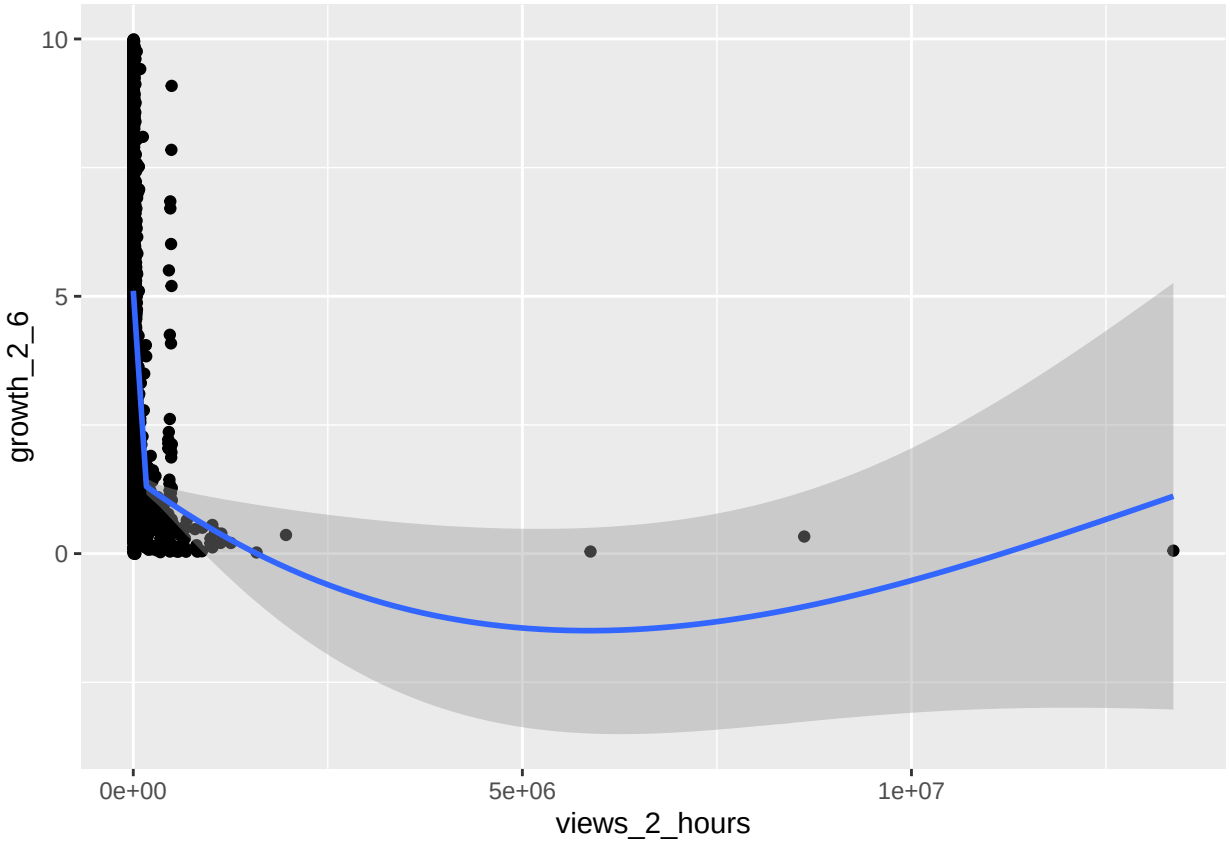

```
ggplot(train,aes(x=cnn_17,y=growth_2_6)) + geom_point() + geom_smooth() # cnn_17
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(train,aes(x=views_2_hours,y=growth_2_6)) + geom_point() + geom_smooth() # views_2_hours
```

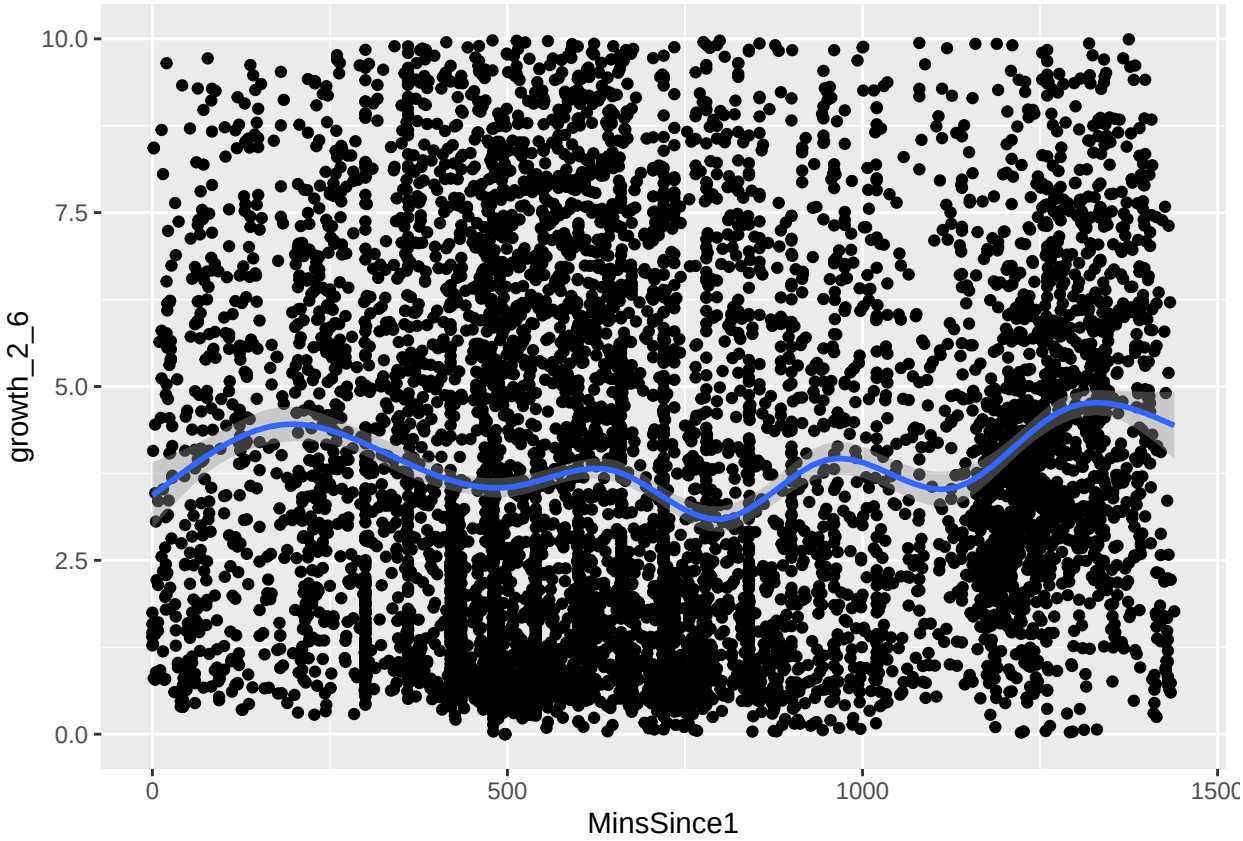
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



exploring our created features

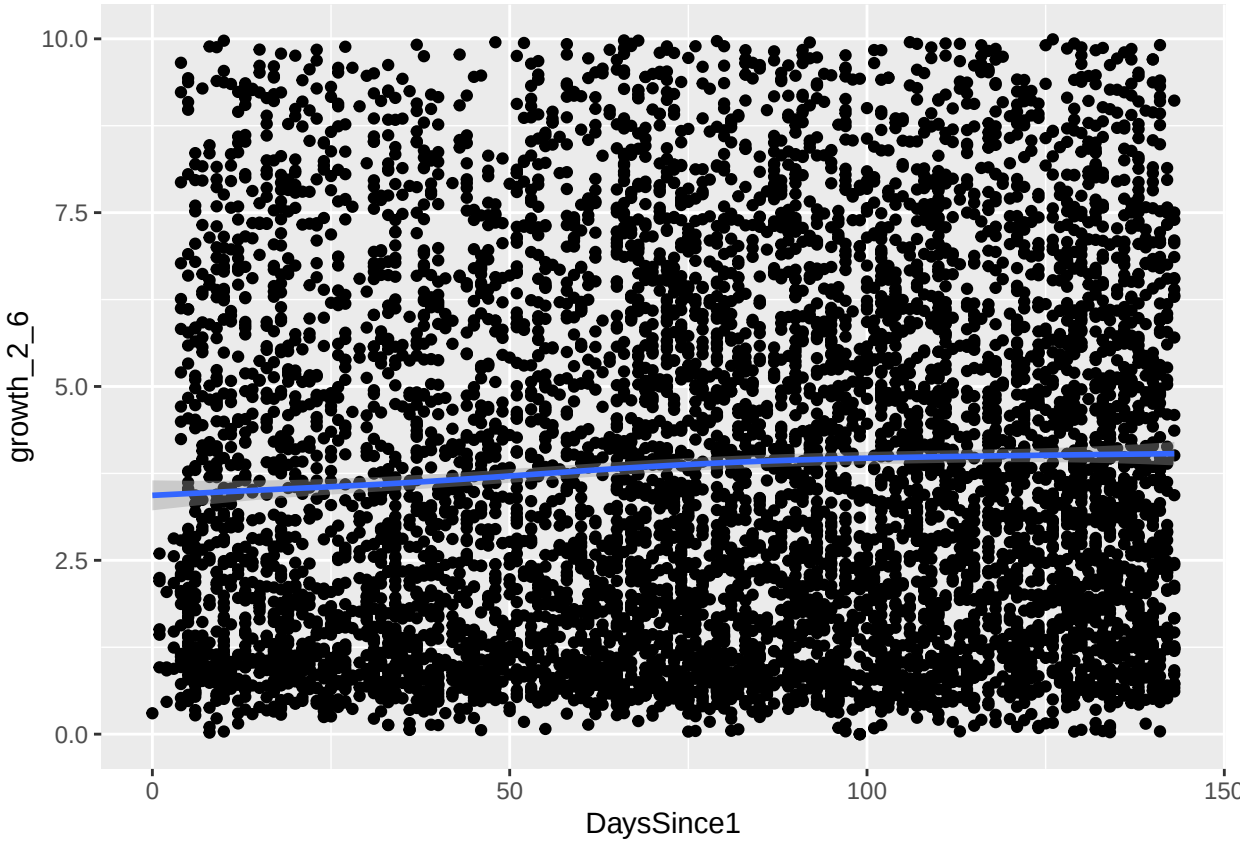
```
# numeric date and time, and binary duration
ggplot(train,aes(x=MinsSince1,y=growth_2_6)) + geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

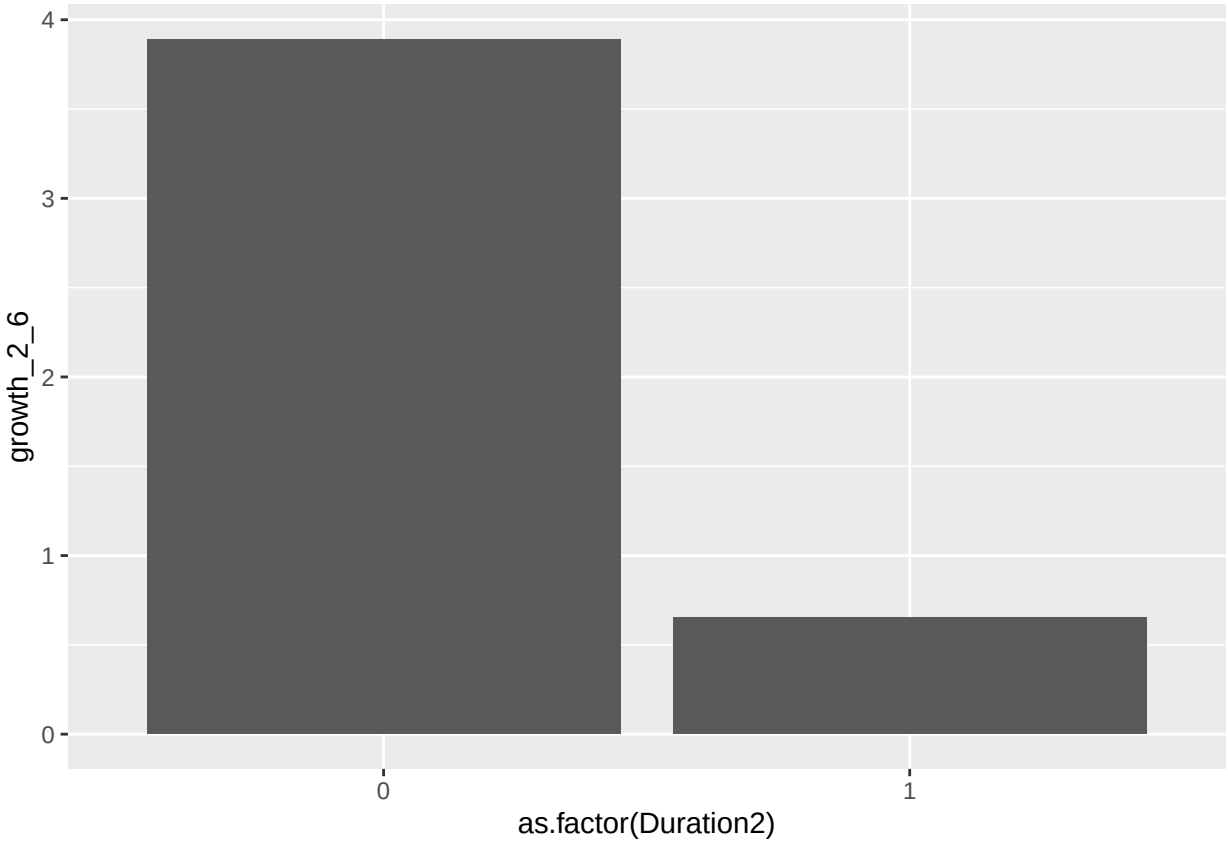



```
ggplot(train,aes(x=DaysSince1,y=growth_2_6)) + geom_point() + geom_smooth()
```

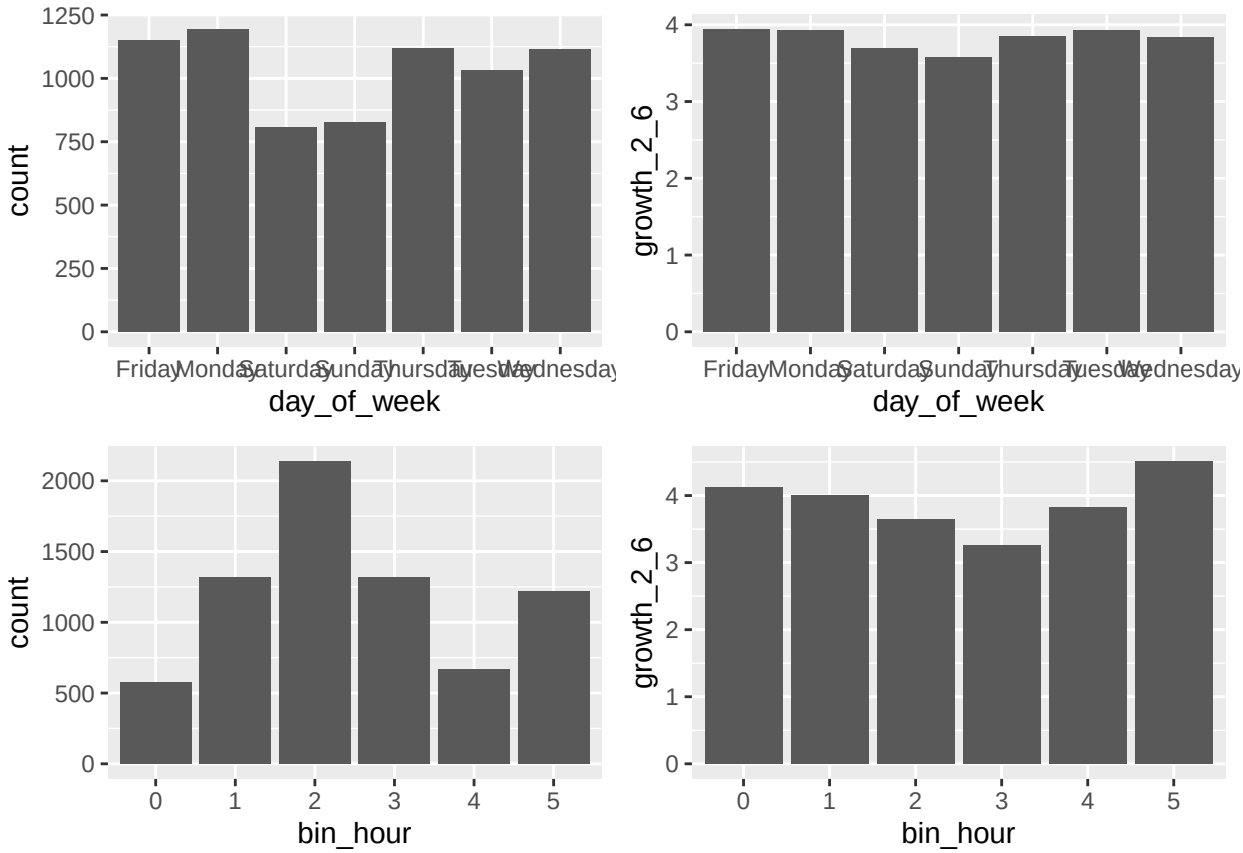
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
ggplot(train,aes(x=as.factor(Duration2),y=growth_2_6)) + stat_summary(fun = "mean", geom = "col") #
```




```
# day of week
p1 <- ggplot(train) + geom_bar(aes(x=day_of_week)) # lower uploads on saturday and sunday, Mondays, Fri
p2 <- ggplot(train,aes(x=day_of_week,y=growth_2_6)) + stat_summary(fun = "mean", geom = "col")# geom_ba
# hour of day
p3 <- ggplot(train) + geom_bar(aes(x=bin_hour)) # upload time dist
p4 <- ggplot(train,aes(x=bin_hour,y=growth_2_6)) + stat_summary(fun = "mean", geom = "col")#+ geom_bar(
grid.arrange(p1,p2,p3,p4,nrow=2)
```



variable selection

first remove all non-numeric features

```
train <- train %>% select(-c(PublishedDate,PublishedTime,day_of_week,bin_hour))
# and move growth to last col
train <- train %>% relocate(growth_2_6, .after = last_col())
```

RFE subselect

```
# the code takes hours to run
# so we have commented it out

# set.seed(22)
# subsets <- c(30:40)
# control <- rfeControl(functions=rfFuncs,
#                       method="cv",
#                       number=10,
#                       verbose=T)
# x <- train[,-which(colnames(train)=="growth_2_6")]
# y <- as.matrix(train[,which(colnames(train)=="growth_2_6")])
# rfe_sub <- rfe(x,y,
#                sizes=subsets,
#                rfeControl = control)
```

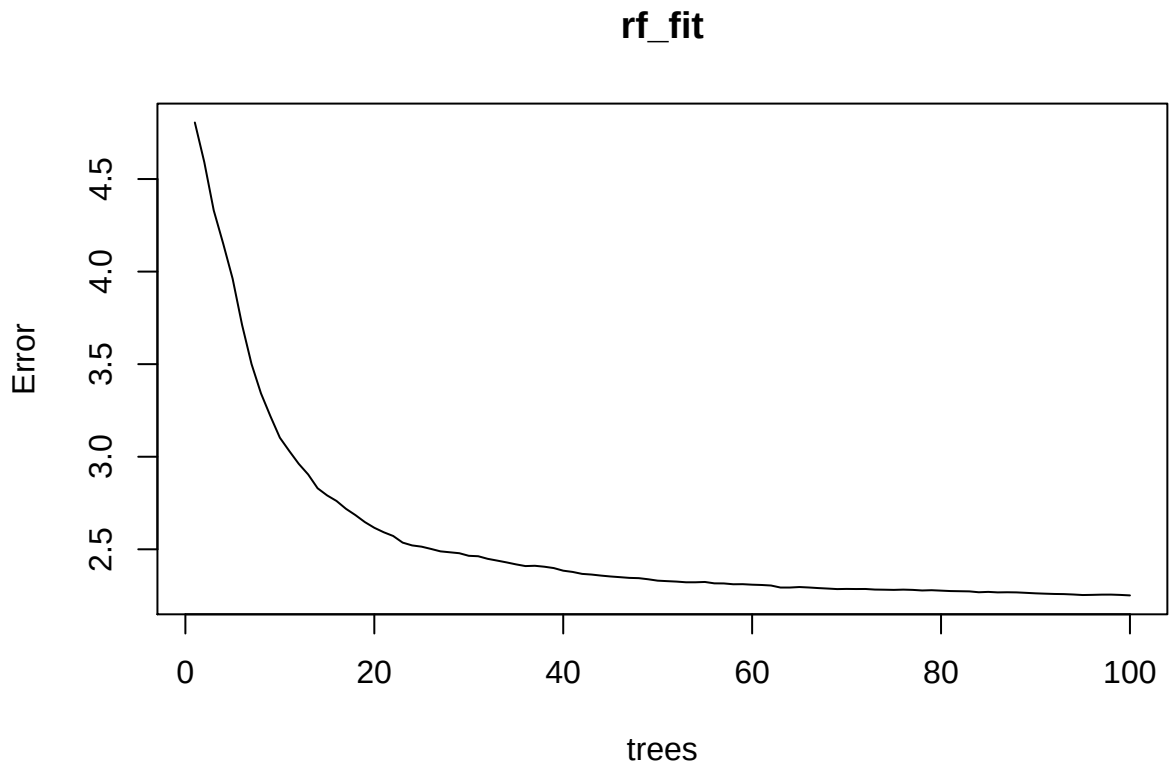
```
# # final variables
# paste(rfe_sub$optVariables,collapse = " ")
```

RF importance

```
# variables sorted by importance for Random Forest
rf_fit = randomForest(growth_2_6~., data = train,
                          ntree = 100,
                          mtry= floor(ncol(train)/3),
                          trControl = oob_control,
                          importance=TRUE)
incMSE_Imp <- importance(rf_fit)[,"%IncMSE"] %>% sort(decreasing = T)
paste(names(incMSE_Imp[incMSE_Imp >= 1]),collapse=" ")
```

```
## [1] "avg_growth_high cnn_10 cnn_86 cnn_89 Num_Subscribers_Base_low_mid cnn_17 views_2_hours cnn_25 Nu
```

```
plot(rf_fit)
```



lasso subselect

```
X_train = model.matrix(growth_2_6~., train)
y_train = train$growth_2_6

lasso_fit <- cv.glmnet(X_train,y_train,data = train, family = "gaussian", alpha = 1, #lambda = 0.00208,
                       standardize = TRUE)
lasso_coeffs <- extract.coef(lasso_fit) %>% arrange(desc(abs(Value)))
paste(lasso_coeffs$Coefficient[1:93],collapse = " ")
```

```
## [1] "(Intercept) avg_growth_high Num_Subscribers_Base_low_mid hog_116 punc_num_bktk hog_454 hog_641 c
```

final set of predictors chosen from above 3 methods

```
subset <- unlist(strsplit("avg_growth_high avg_growth_low avg_growth_low_mid avg_growth_mid_high bin_hou
train <- train[c(subset,"growth_2_6")]
```

creating model

Split train and test

```
set.seed(22)
train_index <- createDataPartition(train$growth_2_6, p = 0.7,
          list = FALSE)
train <- train[train_index,]
valid <- train[-train_index,]
dim(train)
```

```
## [1] 5070   41
```

```
dim(valid)
```

```
## [1] 1514   41
```

Format data for XGBoost

```
# xgboost requires data as matrix, and response separated from predictors
trainm <- train %>% dplyr::select(-c(growth_2_6)) %>% as.matrix()
train_label <- as.matrix(train[,"growth_2_6"])
train_matrix <- xgb.DMatrix(data = as.matrix(trainm),label = train_label)

# do same for valid
validm <- valid %>% dplyr::select(-growth_2_6) %>% as.matrix()
valid_label <- as.matrix(valid[,"growth_2_6"])
valid_matrix <- xgb.DMatrix(data = as.matrix(validm),label = valid_label)
```

Parameter tuning through cv

```
watchlist <- list(train = train_matrix, test = valid_matrix)
set.seed(22)
best_param = list()
best_RMSE = Inf
best_RMSE_index = 0

for (iter in 1:100){
  xgb_params <- list("objective" = "reg:squarederror",
                     "eval_metric" = "rmse",
                     "eta" = runif(1,0.01,0.3), # learning rate: sample 1 between 0.01 and 0.3
                     "lambda" = runif(1,1,2), # (default 1) larger lambda -> less overfit
                     "gamma"=runif(1,0,0.2), # (default 0) similar to lambda
                     "subsample"=runif(1,0.6,0.9), # fraction of observations to sample per tree
                     "colsample_bytree"=runif(1,0.5,0.8), # simliar to mtry in RF (but proportion)
                     "min_child_weight"=sample(1:40, 1) # higher values -> less overfit
```

```
                    )
  seed.number = sample.int(10000, 1)[[1]] # also set random seed
  set.seed(seed.number)
  cv.nround = 100 # need to cv on enough rounds too
  cv.nfold = 5
  xgb_cv <- xgb.cv(params = xgb_params,
                   data = train_matrix,
                   nrounds = cv.nround,
                   nfold = cv.nfold,
                   early_stopping_rounds = 8, # stop if rmse does not decrease in 8 rounds
                   maximize = F, # minimizing RMSE
                   verbose=F)
  min_RMSE = min(xgb_cv$evaluation_log[, test_rmse_mean]) # find minumum RMSE of n rounds
  min_RMSE_index = which.min(xgb_cv$evaluation_log[, test_rmse_mean]) # index of min RMSE

  if (min_RMSE < best_RMSE) {
      best_RMSE = min_RMSE
      best_RMSE_index = min_RMSE_index
      best_seednumber = seed.number
      best_param = xgb_params
  }
  if(iter %% 5 == 0){
    print(paste("Completed iteration ", iter))
    print(paste("Stopped at ", best_RMSE_index))
    print(paste("Current best RMSE ", best_RMSE))
  }
}
```

```
## [1] "Completed iteration  5"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4617234"
## [1] "Completed iteration  10"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4617234"
## [1] "Completed iteration  15"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  20"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  25"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  30"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  35"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  40"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  45"
## [1] "Stopped at  100"
```

```
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  50"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4525334"
## [1] "Completed iteration  55"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  60"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  65"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  70"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  75"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  80"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  85"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  90"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  95"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
## [1] "Completed iteration  100"
## [1] "Stopped at  100"
## [1] "Current best RMSE  1.4511512"
```

```
best_seednumber
```

```
## [1] 557
```

```
best_RMSE_index # number of rounds with best RMSE
```

```
## [1] 100
```

```
best_param # best parameter values for that round
```

```
## $objective
## [1] "reg:squarederror"
##
## $eval_metric
## [1] "rmse"
##
## $eta
## [1] 0.07636546
##
## $lambda
## [1] 1.920363
```

```
##
## $gamma
## [1] 0.04588421
##
## $subsample
## [1] 0.763911
##
## $colsample_bytree
## [1] 0.6528408
##
## $min_child_weight
## [1] 16
```

Create model with tuned parameters

```
set.seed(best_seednumber)
xgb_fit <- xgb.train(data=train_matrix,
                params=best_param,
                nrounds=500,
                watchlist = watchlist)
```

```
## [1]  train-rmse:3.980966 test-rmse:3.940549
## [2]  train-rmse:3.730891 test-rmse:3.690175
## [3]  train-rmse:3.502533 test-rmse:3.461638
## [4]  train-rmse:3.293606 test-rmse:3.252409
## [5]  train-rmse:3.103177 test-rmse:3.062833
## [6]  train-rmse:2.928995 test-rmse:2.889175
## [7]  train-rmse:2.769187 test-rmse:2.727262
## [8]  train-rmse:2.627423 test-rmse:2.586733
## [9]  train-rmse:2.499185 test-rmse:2.457024
## [10] train-rmse:2.382979 test-rmse:2.340647
## [11] train-rmse:2.276610 test-rmse:2.234018
## [12] train-rmse:2.177235 test-rmse:2.135351
## [13] train-rmse:2.088336 test-rmse:2.046504
## [14] train-rmse:2.007513 test-rmse:1.964225
## [15] train-rmse:1.937313 test-rmse:1.892909
## [16] train-rmse:1.873425 test-rmse:1.828277
## [17] train-rmse:1.817077 test-rmse:1.770338
## [18] train-rmse:1.763414 test-rmse:1.717383
## [19] train-rmse:1.715310 test-rmse:1.670209
## [20] train-rmse:1.672044 test-rmse:1.624511
## [21] train-rmse:1.635366 test-rmse:1.586885
## [22] train-rmse:1.601085 test-rmse:1.552493
## [23] train-rmse:1.571411 test-rmse:1.523751
## [24] train-rmse:1.543327 test-rmse:1.495741
## [25] train-rmse:1.518790 test-rmse:1.470045
## [26] train-rmse:1.495421 test-rmse:1.446694
## [27] train-rmse:1.475356 test-rmse:1.425706
## [28] train-rmse:1.455188 test-rmse:1.406342
## [29] train-rmse:1.437881 test-rmse:1.388505
## [30] train-rmse:1.422261 test-rmse:1.372046
## [31] train-rmse:1.408757 test-rmse:1.359000
## [32] train-rmse:1.397552 test-rmse:1.347399
## [33] train-rmse:1.388516 test-rmse:1.338358
```

```
## [34] train-rmse:1.379882 test-rmse:1.329643
## [35] train-rmse:1.367493 test-rmse:1.316407
## [36] train-rmse:1.359725 test-rmse:1.308389
## [37] train-rmse:1.353133 test-rmse:1.301380
## [38] train-rmse:1.342213 test-rmse:1.290771
## [39] train-rmse:1.333939 test-rmse:1.282504
## [40] train-rmse:1.322911 test-rmse:1.270202
## [41] train-rmse:1.317015 test-rmse:1.263745
## [42] train-rmse:1.311355 test-rmse:1.258621
## [43] train-rmse:1.304130 test-rmse:1.251296
## [44] train-rmse:1.296515 test-rmse:1.244013
## [45] train-rmse:1.288016 test-rmse:1.235485
## [46] train-rmse:1.284544 test-rmse:1.231519
## [47] train-rmse:1.279014 test-rmse:1.226407
## [48] train-rmse:1.273328 test-rmse:1.220318
## [49] train-rmse:1.268399 test-rmse:1.214850
## [50] train-rmse:1.258813 test-rmse:1.204619
## [51] train-rmse:1.254298 test-rmse:1.199854
## [52] train-rmse:1.249325 test-rmse:1.194856
## [53] train-rmse:1.245793 test-rmse:1.191412
## [54] train-rmse:1.241816 test-rmse:1.187454
## [55] train-rmse:1.239455 test-rmse:1.185362
## [56] train-rmse:1.234452 test-rmse:1.181023
## [57] train-rmse:1.232270 test-rmse:1.178320
## [58] train-rmse:1.231153 test-rmse:1.177559
## [59] train-rmse:1.229061 test-rmse:1.175881
## [60] train-rmse:1.225523 test-rmse:1.172139
## [61] train-rmse:1.221330 test-rmse:1.168194
## [62] train-rmse:1.219580 test-rmse:1.166551
## [63] train-rmse:1.214482 test-rmse:1.162596
## [64] train-rmse:1.210475 test-rmse:1.158882
## [65] train-rmse:1.207346 test-rmse:1.156511
## [66] train-rmse:1.205369 test-rmse:1.153988
## [67] train-rmse:1.201089 test-rmse:1.150432
## [68] train-rmse:1.197651 test-rmse:1.146894
## [69] train-rmse:1.194386 test-rmse:1.143690
## [70] train-rmse:1.192083 test-rmse:1.141408
## [71] train-rmse:1.189401 test-rmse:1.139332
## [72] train-rmse:1.188164 test-rmse:1.138204
## [73] train-rmse:1.185176 test-rmse:1.136246
## [74] train-rmse:1.184499 test-rmse:1.135748
## [75] train-rmse:1.182506 test-rmse:1.133921
## [76] train-rmse:1.180092 test-rmse:1.131723
## [77] train-rmse:1.178299 test-rmse:1.129266
## [78] train-rmse:1.175901 test-rmse:1.126932
## [79] train-rmse:1.174902 test-rmse:1.126238
## [80] train-rmse:1.173281 test-rmse:1.124637
## [81] train-rmse:1.169641 test-rmse:1.121270
## [82] train-rmse:1.168007 test-rmse:1.120182
## [83] train-rmse:1.163172 test-rmse:1.115692
## [84] train-rmse:1.158854 test-rmse:1.111909
## [85] train-rmse:1.155611 test-rmse:1.109203
## [86] train-rmse:1.152746 test-rmse:1.106630
## [87] train-rmse:1.149361 test-rmse:1.102944
```

```
## [88] train-rmse:1.146411 test-rmse:1.100112
## [89] train-rmse:1.144814 test-rmse:1.098647
## [90] train-rmse:1.139972 test-rmse:1.093068
## [91] train-rmse:1.138213 test-rmse:1.091491
## [92] train-rmse:1.133734 test-rmse:1.087076
## [93] train-rmse:1.130183 test-rmse:1.084084
## [94] train-rmse:1.125750 test-rmse:1.078923
## [95] train-rmse:1.122755 test-rmse:1.076698
## [96] train-rmse:1.119065 test-rmse:1.072559
## [97] train-rmse:1.116044 test-rmse:1.069399
## [98] train-rmse:1.112321 test-rmse:1.066132
## [99] train-rmse:1.109467 test-rmse:1.063829
## [100]    train-rmse:1.107287 test-rmse:1.061993
## [101]    train-rmse:1.105556 test-rmse:1.060501
## [102]    train-rmse:1.102753 test-rmse:1.056482
## [103]    train-rmse:1.100359 test-rmse:1.054304
## [104]    train-rmse:1.098308 test-rmse:1.052449
## [105]    train-rmse:1.096340 test-rmse:1.050071
## [106]    train-rmse:1.094346 test-rmse:1.048479
## [107]    train-rmse:1.089995 test-rmse:1.044490
## [108]    train-rmse:1.088851 test-rmse:1.043283
## [109]    train-rmse:1.085375 test-rmse:1.039856
## [110]    train-rmse:1.081202 test-rmse:1.036354
## [111]    train-rmse:1.078739 test-rmse:1.033474
## [112]    train-rmse:1.074086 test-rmse:1.029593
## [113]    train-rmse:1.070534 test-rmse:1.025988
## [114]    train-rmse:1.067094 test-rmse:1.023349
## [115]    train-rmse:1.065543 test-rmse:1.021735
## [116]    train-rmse:1.061059 test-rmse:1.017380
## [117]    train-rmse:1.058037 test-rmse:1.014292
## [118]    train-rmse:1.056143 test-rmse:1.012086
## [119]    train-rmse:1.054986 test-rmse:1.011330
## [120]    train-rmse:1.052443 test-rmse:1.009112
## [121]    train-rmse:1.050371 test-rmse:1.006881
## [122]    train-rmse:1.048889 test-rmse:1.005309
## [123]    train-rmse:1.046473 test-rmse:1.002869
## [124]    train-rmse:1.045085 test-rmse:1.001402
## [125]    train-rmse:1.043980 test-rmse:1.000072
## [126]    train-rmse:1.041901 test-rmse:0.998864
## [127]    train-rmse:1.041125 test-rmse:0.998482
## [128]    train-rmse:1.037061 test-rmse:0.995028
## [129]    train-rmse:1.034588 test-rmse:0.992739
## [130]    train-rmse:1.032864 test-rmse:0.991188
## [131]    train-rmse:1.030819 test-rmse:0.989549
## [132]    train-rmse:1.026703 test-rmse:0.985955
## [133]    train-rmse:1.024732 test-rmse:0.984038
## [134]    train-rmse:1.023262 test-rmse:0.982202
## [135]    train-rmse:1.018881 test-rmse:0.978220
## [136]    train-rmse:1.017364 test-rmse:0.976723
## [137]    train-rmse:1.015483 test-rmse:0.974993
## [138]    train-rmse:1.013167 test-rmse:0.972763
## [139]    train-rmse:1.011353 test-rmse:0.971544
## [140]    train-rmse:1.009608 test-rmse:0.969869
## [141]    train-rmse:1.006566 test-rmse:0.966828
```

```
## [142]    train-rmse:1.004525 test-rmse:0.964876
## [143]    train-rmse:1.003655 test-rmse:0.963932
## [144]    train-rmse:1.000628 test-rmse:0.960524
## [145]    train-rmse:0.996676 test-rmse:0.956816
## [146]    train-rmse:0.996098 test-rmse:0.956127
## [147]    train-rmse:0.994228 test-rmse:0.954639
## [148]    train-rmse:0.990461 test-rmse:0.950914
## [149]    train-rmse:0.989735 test-rmse:0.949812
## [150]    train-rmse:0.986842 test-rmse:0.947797
## [151]    train-rmse:0.984686 test-rmse:0.945694
## [152]    train-rmse:0.983000 test-rmse:0.944108
## [153]    train-rmse:0.981099 test-rmse:0.943231
## [154]    train-rmse:0.979203 test-rmse:0.941475
## [155]    train-rmse:0.977992 test-rmse:0.940297
## [156]    train-rmse:0.975571 test-rmse:0.938265
## [157]    train-rmse:0.973738 test-rmse:0.937037
## [158]    train-rmse:0.972491 test-rmse:0.936013
## [159]    train-rmse:0.970570 test-rmse:0.934623
## [160]    train-rmse:0.968296 test-rmse:0.932594
## [161]    train-rmse:0.966147 test-rmse:0.930525
## [162]    train-rmse:0.964317 test-rmse:0.928835
## [163]    train-rmse:0.961000 test-rmse:0.925837
## [164]    train-rmse:0.958314 test-rmse:0.923588
## [165]    train-rmse:0.957115 test-rmse:0.922401
## [166]    train-rmse:0.955667 test-rmse:0.921388
## [167]    train-rmse:0.953721 test-rmse:0.919674
## [168]    train-rmse:0.951551 test-rmse:0.917518
## [169]    train-rmse:0.948920 test-rmse:0.914825
## [170]    train-rmse:0.946041 test-rmse:0.911841
## [171]    train-rmse:0.944416 test-rmse:0.910090
## [172]    train-rmse:0.942236 test-rmse:0.908583
## [173]    train-rmse:0.940108 test-rmse:0.907131
## [174]    train-rmse:0.937583 test-rmse:0.905099
## [175]    train-rmse:0.934673 test-rmse:0.902544
## [176]    train-rmse:0.931444 test-rmse:0.899778
## [177]    train-rmse:0.929155 test-rmse:0.897509
## [178]    train-rmse:0.927110 test-rmse:0.895793
## [179]    train-rmse:0.925310 test-rmse:0.893317
## [180]    train-rmse:0.922617 test-rmse:0.890624
## [181]    train-rmse:0.920240 test-rmse:0.888340
## [182]    train-rmse:0.917727 test-rmse:0.886257
## [183]    train-rmse:0.917253 test-rmse:0.885176
## [184]    train-rmse:0.915168 test-rmse:0.883071
## [185]    train-rmse:0.914009 test-rmse:0.882061
## [186]    train-rmse:0.912060 test-rmse:0.879808
## [187]    train-rmse:0.910786 test-rmse:0.878309
## [188]    train-rmse:0.909209 test-rmse:0.876546
## [189]    train-rmse:0.906328 test-rmse:0.873930
## [190]    train-rmse:0.904873 test-rmse:0.872574
## [191]    train-rmse:0.903459 test-rmse:0.871795
## [192]    train-rmse:0.901217 test-rmse:0.869510
## [193]    train-rmse:0.900317 test-rmse:0.868433
## [194]    train-rmse:0.897286 test-rmse:0.865145
## [195]    train-rmse:0.895694 test-rmse:0.863634
```

```
## [196]    train-rmse:0.893252 test-rmse:0.861141
## [197]    train-rmse:0.891820 test-rmse:0.859649
## [198]    train-rmse:0.889220 test-rmse:0.857219
## [199]    train-rmse:0.887201 test-rmse:0.855087
## [200]    train-rmse:0.885379 test-rmse:0.853104
## [201]    train-rmse:0.883429 test-rmse:0.851252
## [202]    train-rmse:0.880177 test-rmse:0.848152
## [203]    train-rmse:0.877390 test-rmse:0.845695
## [204]    train-rmse:0.875623 test-rmse:0.843846
## [205]    train-rmse:0.873801 test-rmse:0.842558
## [206]    train-rmse:0.872103 test-rmse:0.841085
## [207]    train-rmse:0.870265 test-rmse:0.839578
## [208]    train-rmse:0.867292 test-rmse:0.836386
## [209]    train-rmse:0.865211 test-rmse:0.834793
## [210]    train-rmse:0.863276 test-rmse:0.832697
## [211]    train-rmse:0.861271 test-rmse:0.830880
## [212]    train-rmse:0.859940 test-rmse:0.829544
## [213]    train-rmse:0.859224 test-rmse:0.829109
## [214]    train-rmse:0.857888 test-rmse:0.827842
## [215]    train-rmse:0.854748 test-rmse:0.825048
## [216]    train-rmse:0.852002 test-rmse:0.822409
## [217]    train-rmse:0.851176 test-rmse:0.821815
## [218]    train-rmse:0.848915 test-rmse:0.819978
## [219]    train-rmse:0.848028 test-rmse:0.819173
## [220]    train-rmse:0.845899 test-rmse:0.816902
## [221]    train-rmse:0.844716 test-rmse:0.815932
## [222]    train-rmse:0.843605 test-rmse:0.814764
## [223]    train-rmse:0.841525 test-rmse:0.812600
## [224]    train-rmse:0.839547 test-rmse:0.810486
## [225]    train-rmse:0.839069 test-rmse:0.809784
## [226]    train-rmse:0.838444 test-rmse:0.809556
## [227]    train-rmse:0.837312 test-rmse:0.808728
## [228]    train-rmse:0.835741 test-rmse:0.807316
## [229]    train-rmse:0.832590 test-rmse:0.803250
## [230]    train-rmse:0.831374 test-rmse:0.801845
## [231]    train-rmse:0.830644 test-rmse:0.801284
## [232]    train-rmse:0.829371 test-rmse:0.800198
## [233]    train-rmse:0.826785 test-rmse:0.797554
## [234]    train-rmse:0.824094 test-rmse:0.795110
## [235]    train-rmse:0.822027 test-rmse:0.793180
## [236]    train-rmse:0.820409 test-rmse:0.792139
## [237]    train-rmse:0.819674 test-rmse:0.791502
## [238]    train-rmse:0.816972 test-rmse:0.789010
## [239]    train-rmse:0.813988 test-rmse:0.785863
## [240]    train-rmse:0.811312 test-rmse:0.782388
## [241]    train-rmse:0.809783 test-rmse:0.781024
## [242]    train-rmse:0.808563 test-rmse:0.780011
## [243]    train-rmse:0.806094 test-rmse:0.777335
## [244]    train-rmse:0.805237 test-rmse:0.776654
## [245]    train-rmse:0.803634 test-rmse:0.774829
## [246]    train-rmse:0.802519 test-rmse:0.773635
## [247]    train-rmse:0.801826 test-rmse:0.773010
## [248]    train-rmse:0.800675 test-rmse:0.772095
## [249]    train-rmse:0.799914 test-rmse:0.771179
```

```
## [250]    train-rmse:0.797308 test-rmse:0.768417
## [251]    train-rmse:0.796316 test-rmse:0.767352
## [252]    train-rmse:0.793856 test-rmse:0.764653
## [253]    train-rmse:0.791743 test-rmse:0.762567
## [254]    train-rmse:0.791003 test-rmse:0.762301
## [255]    train-rmse:0.788942 test-rmse:0.760139
## [256]    train-rmse:0.787742 test-rmse:0.759070
## [257]    train-rmse:0.785927 test-rmse:0.757486
## [258]    train-rmse:0.784754 test-rmse:0.756090
## [259]    train-rmse:0.782362 test-rmse:0.754039
## [260]    train-rmse:0.781914 test-rmse:0.753921
## [261]    train-rmse:0.780094 test-rmse:0.752691
## [262]    train-rmse:0.778729 test-rmse:0.751268
## [263]    train-rmse:0.777914 test-rmse:0.750506
## [264]    train-rmse:0.776727 test-rmse:0.749233
## [265]    train-rmse:0.775825 test-rmse:0.748148
## [266]    train-rmse:0.774760 test-rmse:0.746664
## [267]    train-rmse:0.773697 test-rmse:0.745456
## [268]    train-rmse:0.771832 test-rmse:0.743622
## [269]    train-rmse:0.770925 test-rmse:0.742817
## [270]    train-rmse:0.769148 test-rmse:0.741065
## [271]    train-rmse:0.768176 test-rmse:0.740123
## [272]    train-rmse:0.766237 test-rmse:0.738491
## [273]    train-rmse:0.764522 test-rmse:0.737412
## [274]    train-rmse:0.762812 test-rmse:0.735759
## [275]    train-rmse:0.761861 test-rmse:0.734821
## [276]    train-rmse:0.760164 test-rmse:0.733243
## [277]    train-rmse:0.759159 test-rmse:0.732434
## [278]    train-rmse:0.757191 test-rmse:0.731053
## [279]    train-rmse:0.755587 test-rmse:0.729759
## [280]    train-rmse:0.755069 test-rmse:0.729026
## [281]    train-rmse:0.753662 test-rmse:0.727610
## [282]    train-rmse:0.752893 test-rmse:0.727072
## [283]    train-rmse:0.751385 test-rmse:0.725269
## [284]    train-rmse:0.749377 test-rmse:0.723055
## [285]    train-rmse:0.748174 test-rmse:0.722082
## [286]    train-rmse:0.745949 test-rmse:0.719420
## [287]    train-rmse:0.744206 test-rmse:0.717980
## [288]    train-rmse:0.742439 test-rmse:0.715931
## [289]    train-rmse:0.741046 test-rmse:0.714538
## [290]    train-rmse:0.739363 test-rmse:0.713380
## [291]    train-rmse:0.737168 test-rmse:0.711460
## [292]    train-rmse:0.736599 test-rmse:0.710968
## [293]    train-rmse:0.735962 test-rmse:0.710670
## [294]    train-rmse:0.734826 test-rmse:0.709442
## [295]    train-rmse:0.732746 test-rmse:0.707165
## [296]    train-rmse:0.731869 test-rmse:0.706218
## [297]    train-rmse:0.729721 test-rmse:0.704300
## [298]    train-rmse:0.728602 test-rmse:0.702930
## [299]    train-rmse:0.727531 test-rmse:0.701806
## [300]    train-rmse:0.726580 test-rmse:0.701164
## [301]    train-rmse:0.725858 test-rmse:0.700388
## [302]    train-rmse:0.723688 test-rmse:0.698384
## [303]    train-rmse:0.722609 test-rmse:0.697215
```

```
## [304]    train-rmse:0.720867 test-rmse:0.695499
## [305]    train-rmse:0.718701 test-rmse:0.693353
## [306]    train-rmse:0.717462 test-rmse:0.692163
## [307]    train-rmse:0.716301 test-rmse:0.690820
## [308]    train-rmse:0.715236 test-rmse:0.689755
## [309]    train-rmse:0.713692 test-rmse:0.688536
## [310]    train-rmse:0.711912 test-rmse:0.687019
## [311]    train-rmse:0.710193 test-rmse:0.685679
## [312]    train-rmse:0.708828 test-rmse:0.684483
## [313]    train-rmse:0.706941 test-rmse:0.683460
## [314]    train-rmse:0.706054 test-rmse:0.682891
## [315]    train-rmse:0.704536 test-rmse:0.681564
## [316]    train-rmse:0.703495 test-rmse:0.680323
## [317]    train-rmse:0.701973 test-rmse:0.678425
## [318]    train-rmse:0.701423 test-rmse:0.677875
## [319]    train-rmse:0.700126 test-rmse:0.676354
## [320]    train-rmse:0.698831 test-rmse:0.675106
## [321]    train-rmse:0.697991 test-rmse:0.674364
## [322]    train-rmse:0.697307 test-rmse:0.673758
## [323]    train-rmse:0.695968 test-rmse:0.672263
## [324]    train-rmse:0.693881 test-rmse:0.670520
## [325]    train-rmse:0.692684 test-rmse:0.669381
## [326]    train-rmse:0.691881 test-rmse:0.668464
## [327]    train-rmse:0.690134 test-rmse:0.666395
## [328]    train-rmse:0.689142 test-rmse:0.665403
## [329]    train-rmse:0.688552 test-rmse:0.664810
## [330]    train-rmse:0.687817 test-rmse:0.663949
## [331]    train-rmse:0.686416 test-rmse:0.662535
## [332]    train-rmse:0.685160 test-rmse:0.661403
## [333]    train-rmse:0.684440 test-rmse:0.661207
## [334]    train-rmse:0.682171 test-rmse:0.659004
## [335]    train-rmse:0.681422 test-rmse:0.658293
## [336]    train-rmse:0.679171 test-rmse:0.656031
## [337]    train-rmse:0.678338 test-rmse:0.655335
## [338]    train-rmse:0.677026 test-rmse:0.654340
## [339]    train-rmse:0.676225 test-rmse:0.653193
## [340]    train-rmse:0.675391 test-rmse:0.652304
## [341]    train-rmse:0.674465 test-rmse:0.651662
## [342]    train-rmse:0.673067 test-rmse:0.650478
## [343]    train-rmse:0.671717 test-rmse:0.649411
## [344]    train-rmse:0.670406 test-rmse:0.648445
## [345]    train-rmse:0.669932 test-rmse:0.647897
## [346]    train-rmse:0.669593 test-rmse:0.647181
## [347]    train-rmse:0.668158 test-rmse:0.646252
## [348]    train-rmse:0.666871 test-rmse:0.645021
## [349]    train-rmse:0.665404 test-rmse:0.643485
## [350]    train-rmse:0.664445 test-rmse:0.642162
## [351]    train-rmse:0.662282 test-rmse:0.640471
## [352]    train-rmse:0.660002 test-rmse:0.638387
## [353]    train-rmse:0.658524 test-rmse:0.636852
## [354]    train-rmse:0.657514 test-rmse:0.635983
## [355]    train-rmse:0.656485 test-rmse:0.635134
## [356]    train-rmse:0.655182 test-rmse:0.633924
## [357]    train-rmse:0.653995 test-rmse:0.632798
```

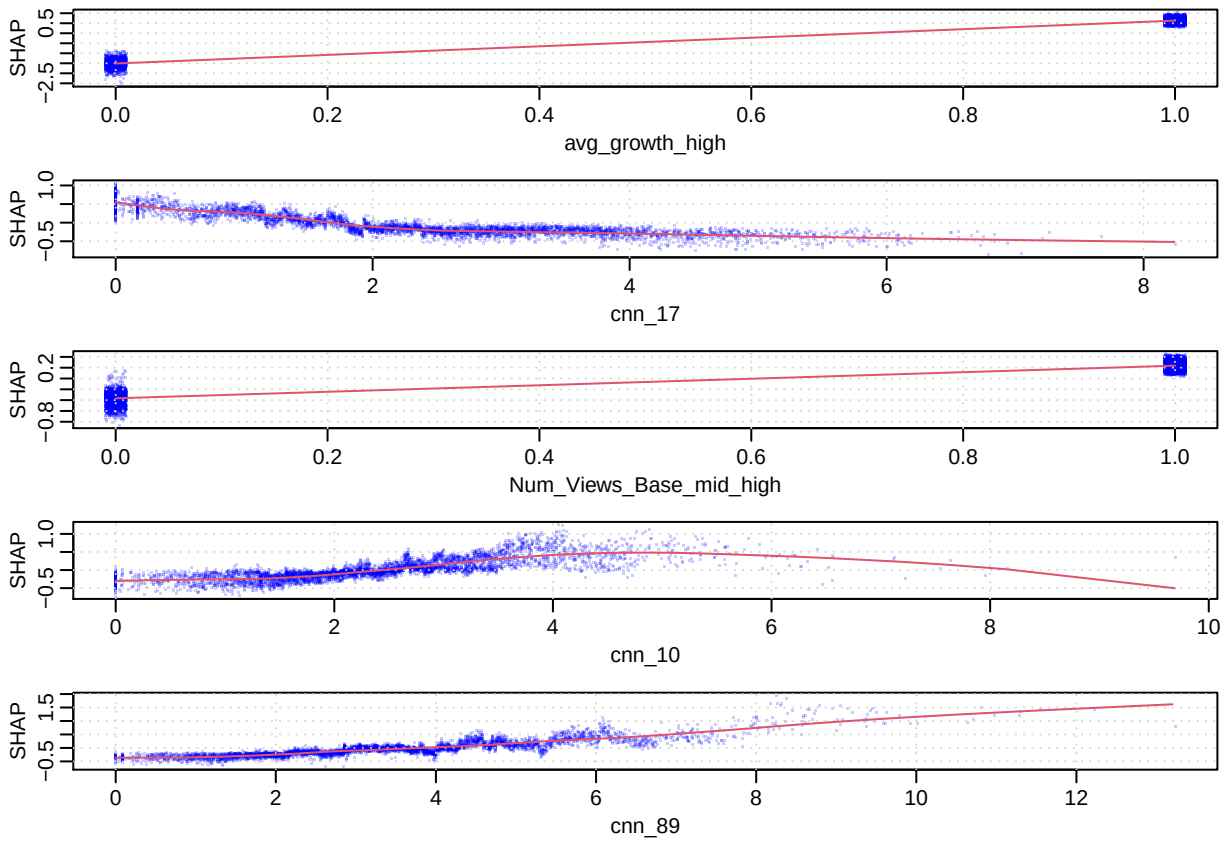
```
## [358]    train-rmse:0.652017 test-rmse:0.631039
## [359]    train-rmse:0.650585 test-rmse:0.629556
## [360]    train-rmse:0.649632 test-rmse:0.628392
## [361]    train-rmse:0.648926 test-rmse:0.627745
## [362]    train-rmse:0.647813 test-rmse:0.626550
## [363]    train-rmse:0.646688 test-rmse:0.625370
## [364]    train-rmse:0.645548 test-rmse:0.624341
## [365]    train-rmse:0.644513 test-rmse:0.623121
## [366]    train-rmse:0.643884 test-rmse:0.622650
## [367]    train-rmse:0.642333 test-rmse:0.621182
## [368]    train-rmse:0.640275 test-rmse:0.619074
## [369]    train-rmse:0.638806 test-rmse:0.617434
## [370]    train-rmse:0.638023 test-rmse:0.616571
## [371]    train-rmse:0.637346 test-rmse:0.616132
## [372]    train-rmse:0.636063 test-rmse:0.614996
## [373]    train-rmse:0.634696 test-rmse:0.613879
## [374]    train-rmse:0.634312 test-rmse:0.613464
## [375]    train-rmse:0.633438 test-rmse:0.612745
## [376]    train-rmse:0.632294 test-rmse:0.611645
## [377]    train-rmse:0.631493 test-rmse:0.611028
## [378]    train-rmse:0.630599 test-rmse:0.610184
## [379]    train-rmse:0.629387 test-rmse:0.609031
## [380]    train-rmse:0.627147 test-rmse:0.606702
## [381]    train-rmse:0.625359 test-rmse:0.604632
## [382]    train-rmse:0.623860 test-rmse:0.602966
## [383]    train-rmse:0.622682 test-rmse:0.602075
## [384]    train-rmse:0.621585 test-rmse:0.601282
## [385]    train-rmse:0.620168 test-rmse:0.599974
## [386]    train-rmse:0.618488 test-rmse:0.598457
## [387]    train-rmse:0.617000 test-rmse:0.597154
## [388]    train-rmse:0.615617 test-rmse:0.595611
## [389]    train-rmse:0.614860 test-rmse:0.594907
## [390]    train-rmse:0.614022 test-rmse:0.594117
## [391]    train-rmse:0.613195 test-rmse:0.593231
## [392]    train-rmse:0.611820 test-rmse:0.591842
## [393]    train-rmse:0.610731 test-rmse:0.590898
## [394]    train-rmse:0.609356 test-rmse:0.589679
## [395]    train-rmse:0.608451 test-rmse:0.588467
## [396]    train-rmse:0.607173 test-rmse:0.587559
## [397]    train-rmse:0.606470 test-rmse:0.586718
## [398]    train-rmse:0.605244 test-rmse:0.585830
## [399]    train-rmse:0.604646 test-rmse:0.585342
## [400]    train-rmse:0.603530 test-rmse:0.584198
## [401]    train-rmse:0.602623 test-rmse:0.583399
## [402]    train-rmse:0.601329 test-rmse:0.582330
## [403]    train-rmse:0.600210 test-rmse:0.581235
## [404]    train-rmse:0.598853 test-rmse:0.579847
## [405]    train-rmse:0.596860 test-rmse:0.578320
## [406]    train-rmse:0.595422 test-rmse:0.577435
## [407]    train-rmse:0.593431 test-rmse:0.575863
## [408]    train-rmse:0.593126 test-rmse:0.575553
## [409]    train-rmse:0.592376 test-rmse:0.575015
## [410]    train-rmse:0.591463 test-rmse:0.574083
## [411]    train-rmse:0.589531 test-rmse:0.572187
```

```
## [412]    train-rmse:0.588435 test-rmse:0.570693
## [413]    train-rmse:0.587846 test-rmse:0.570172
## [414]    train-rmse:0.586840 test-rmse:0.569250
## [415]    train-rmse:0.585770 test-rmse:0.568249
## [416]    train-rmse:0.584600 test-rmse:0.567228
## [417]    train-rmse:0.584084 test-rmse:0.566800
## [418]    train-rmse:0.582976 test-rmse:0.565719
## [419]    train-rmse:0.581313 test-rmse:0.564447
## [420]    train-rmse:0.580380 test-rmse:0.563696
## [421]    train-rmse:0.579559 test-rmse:0.562839
## [422]    train-rmse:0.579214 test-rmse:0.562393
## [423]    train-rmse:0.577814 test-rmse:0.561148
## [424]    train-rmse:0.577021 test-rmse:0.560858
## [425]    train-rmse:0.576355 test-rmse:0.560080
## [426]    train-rmse:0.574804 test-rmse:0.558384
## [427]    train-rmse:0.573869 test-rmse:0.557346
## [428]    train-rmse:0.573038 test-rmse:0.556761
## [429]    train-rmse:0.572355 test-rmse:0.556224
## [430]    train-rmse:0.571549 test-rmse:0.555790
## [431]    train-rmse:0.569792 test-rmse:0.554863
## [432]    train-rmse:0.567914 test-rmse:0.552897
## [433]    train-rmse:0.567219 test-rmse:0.552094
## [434]    train-rmse:0.566593 test-rmse:0.551508
## [435]    train-rmse:0.565767 test-rmse:0.550911
## [436]    train-rmse:0.564675 test-rmse:0.549855
## [437]    train-rmse:0.563199 test-rmse:0.548549
## [438]    train-rmse:0.561372 test-rmse:0.546901
## [439]    train-rmse:0.560999 test-rmse:0.546519
## [440]    train-rmse:0.559456 test-rmse:0.545209
## [441]    train-rmse:0.558581 test-rmse:0.543814
## [442]    train-rmse:0.557518 test-rmse:0.542661
## [443]    train-rmse:0.557074 test-rmse:0.542170
## [444]    train-rmse:0.556146 test-rmse:0.541380
## [445]    train-rmse:0.555155 test-rmse:0.540288
## [446]    train-rmse:0.553837 test-rmse:0.539421
## [447]    train-rmse:0.552988 test-rmse:0.538543
## [448]    train-rmse:0.552013 test-rmse:0.537776
## [449]    train-rmse:0.550649 test-rmse:0.536452
## [450]    train-rmse:0.548762 test-rmse:0.534509
## [451]    train-rmse:0.547439 test-rmse:0.533513
## [452]    train-rmse:0.546387 test-rmse:0.532687
## [453]    train-rmse:0.545457 test-rmse:0.532068
## [454]    train-rmse:0.544534 test-rmse:0.530893
## [455]    train-rmse:0.543255 test-rmse:0.529659
## [456]    train-rmse:0.541367 test-rmse:0.527945
## [457]    train-rmse:0.540737 test-rmse:0.527491
## [458]    train-rmse:0.539623 test-rmse:0.526348
## [459]    train-rmse:0.538655 test-rmse:0.525779
## [460]    train-rmse:0.538069 test-rmse:0.524983
## [461]    train-rmse:0.536708 test-rmse:0.523665
## [462]    train-rmse:0.536271 test-rmse:0.523052
## [463]    train-rmse:0.534859 test-rmse:0.521834
## [464]    train-rmse:0.533615 test-rmse:0.520462
## [465]    train-rmse:0.531947 test-rmse:0.518454
```

```
## [466]    train-rmse:0.531182 test-rmse:0.517446
## [467]    train-rmse:0.529546 test-rmse:0.515821
## [468]    train-rmse:0.529032 test-rmse:0.515244
## [469]    train-rmse:0.528007 test-rmse:0.513993
## [470]    train-rmse:0.526712 test-rmse:0.512654
## [471]    train-rmse:0.526046 test-rmse:0.511996
## [472]    train-rmse:0.525831 test-rmse:0.511723
## [473]    train-rmse:0.524558 test-rmse:0.510507
## [474]    train-rmse:0.524217 test-rmse:0.510266
## [475]    train-rmse:0.523509 test-rmse:0.509481
## [476]    train-rmse:0.522930 test-rmse:0.508750
## [477]    train-rmse:0.522051 test-rmse:0.508069
## [478]    train-rmse:0.520685 test-rmse:0.506886
## [479]    train-rmse:0.519806 test-rmse:0.505778
## [480]    train-rmse:0.519411 test-rmse:0.505310
## [481]    train-rmse:0.518468 test-rmse:0.504231
## [482]    train-rmse:0.517312 test-rmse:0.503379
## [483]    train-rmse:0.516010 test-rmse:0.502314
## [484]    train-rmse:0.515130 test-rmse:0.501371
## [485]    train-rmse:0.514165 test-rmse:0.500735
## [486]    train-rmse:0.513298 test-rmse:0.499875
## [487]    train-rmse:0.512533 test-rmse:0.499175
## [488]    train-rmse:0.511941 test-rmse:0.498510
## [489]    train-rmse:0.510847 test-rmse:0.497240
## [490]    train-rmse:0.509866 test-rmse:0.496288
## [491]    train-rmse:0.508944 test-rmse:0.495558
## [492]    train-rmse:0.508756 test-rmse:0.495413
## [493]    train-rmse:0.508055 test-rmse:0.494781
## [494]    train-rmse:0.507188 test-rmse:0.493545
## [495]    train-rmse:0.506118 test-rmse:0.492404
## [496]    train-rmse:0.504300 test-rmse:0.490865
## [497]    train-rmse:0.503278 test-rmse:0.489977
## [498]    train-rmse:0.502358 test-rmse:0.488780
## [499]    train-rmse:0.501250 test-rmse:0.487799
## [500]    train-rmse:0.500933 test-rmse:0.487536
```

Variable importance

```
xgb.plot.shap(data=trainm,model=xgb_fit,top_n=5)
```



Validate

```
valid_p <- predict(xgb_fit,valid_matrix)
xgb_RMSE <- RMSE(valid_p, valid$growth_2_6)
xgb_RMSE # validation RMSE
```

```
## [1] 0.4875357
```

```
valid_preds <- data.frame("obs"=valid$growth_2_6, "pred"=valid_p)
head(valid_preds,10) # check predictions
```

```
##         obs     pred
## 1  3.002473 3.711255
## 2  5.000000 5.355220
## 3  1.932232 2.687959
## 4  6.365432 6.046679
## 5  7.833333 7.839030
## 6  6.597612 6.407967
## 7  2.527267 2.592344
## 8  2.244444 2.094875
## 9  4.231683 3.625521
## 10 4.536059 3.837349
```

Predict on Test

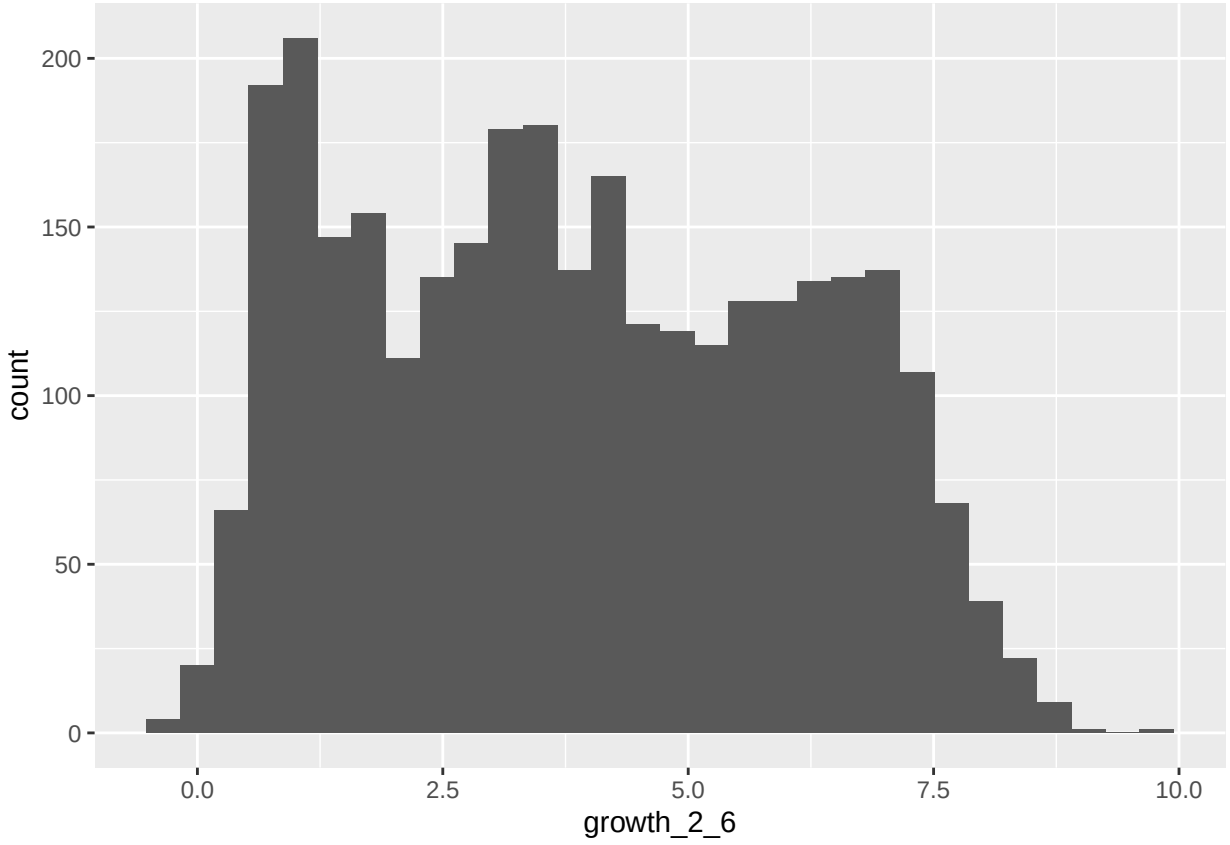
```
# format test for XGB
test <- test[subset]
testm <- as.matrix(test)
```

```
dtest <- xgb.DMatrix(data = testm)
# make prediction
preds <- predict(xgb_fit,newdata = dtest)
df <- data.frame("id" = test_id, "growth_2_6" = preds)
head(df)
```

```
##     id growth_2_6
## 1 7242   5.984253
## 2 7243   2.693341
## 3 7244   1.513965
## 4 7245   7.111295
## 5 7246   4.837350
## 6 7247   2.732717
```

```
# check distribution of predictions
ggplot(df) + geom_histogram(aes(x=growth_2_6))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Write to csv

```
write.csv(df,"sub_4f.csv", row.names = FALSE)
```