Boosting Prediction

Jihoon Han

- UW ID: 20673533

Summary

Before the summary

1. The final model I submitted for the "20673533.R" file had
   the hyperparameter nround = 10000. However, when I was writting the report
   the result came out to be nround = 9900. It is not a huge difference
   but just note that the nround value in "20673533.R" and "20673533.Rmd"
   is slightly different.

2. I followed the process that was shown in a website, thus I will have
   a citation for it

Pelkoja. (2018, July 04). Visual xgboost tuning with caret. Retrieved April 14, 2021, from https://www.kaggle.com/pelkoja/visual-xgboost-tuning-with-caret#tuning-xgbtree-with-caret

Preprocessing

Dealing with missing values and categorical variates

Made a new data frame that has the organized clean data,
categorical variables, and missing values by using the function "designTreatmentsZ" in the "vtreat" package. Note some columns in the data frame acts like a sparse matrix
(only having value 1 or 0)

Transformation

- I **log tranformed** the price variate.

New Variables

- new variables were added to dtrain and dtest.
  The new variables are made by using the function
  "designTreatmentsZ" in the "vtreat" package.
  They have a format of "variate_isBAD" or "variate_lev_category". The "variate_isBAD" is a column of 1 and 0's which is indicated as 1,
  if the variate has missing values.

The "variate_lev_category" is a column of 1 and 0's which is indicated as 1,
if the variate has the same category. (For more details look at the score frame given below)

Model Building

Used the "train" function with the method "xgbTree" in the "caret" package.

Used cross-validation fold of 5 to achieve RMSE and
to know how accurate the model is.

Ajusted the hyperparameters for the tune_grid for the "train" function.

Compared the RMSE of the tuned grid I tuned,
and the default grid in "caret::train" to conclude my final model.

Final Model

- The final model is obtained using the tuned grid having tuned hyper parameters,
  nrounds = 10000,
  eta = 0.005,
  max_depth = 6,
  gamma = 0,
  colsample_bytree = 0.4,
  min_child_weight = 1,
  subsample = 0.75

and the tuned control, with cross validation = 5.

Thus the final model is
(model <- caret::train(
x = input_x,
y = input_y,
trControl = tune_control,
tuneGrid = tuned_grid,
method = "xgbTree"
))

Where the input_x is a matrix containing all values in dtrain except for price.
input_y is a matrix containing price values.

For more details, look at the below process.

Also note that in the below process, the hyper parameter, nround is computed as 9900.
Not 10000.

1.Preprocessing

1.1 Loading data

```
load("project.Rdata")
```

1.2 Libraries used

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.4

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(vtreat)
```

```
## Warning: package 'vtreat' was built under R version 4.0.5

## Loading required package: wrapr

## Warning: package 'wrapr' was built under R version 4.0.5

##
## Attaching package: 'wrapr'

## The following object is masked from 'package:dplyr':
##
##     coalesce
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.0.5

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
set.seed(57)
```

1.3 Generate a dataframe that organizes clean data, categorical variables, and missing values.

```
plan_of_treat <- vtreat::designTreatmentsZ(
  dframe = dtrain,
  varlist = colnames(dtrain),
  codeRestriction = c("clean", "isBAD", "lev"),
  verbose = FALSE)
```

```r
score_frame <- plan_of_treat$scoreFrame %>%
  select(varName, origName, code)
```

```r
dtrain_treated <- vtreat::prepare(plan_of_treat, dtrain)

dtrain_treated$price <- log(dtrain_treated$price)

dim(dtrain_treated)
```

```
## [1] 4855  127
```

Now the dtrain_treated is a dataframe that has the organized clean data, categorical variables, and missing values with the price having log transformation. Since, we got the data to analyze, we want to use boosting and predict the price value.

To do so, we need to tune some hyper parameters.

2. Tuning the hyper parameters

Let's set the initial value of nrounds = 1000 that is one of the hyper parameter in caret::train. Also, let's set the tune_control to have a method cv (cross validation) with 5 folds.

```r
nrounds <- 1000
```

```r
input_x <- as.matrix(select(dtrain_treated, -price))
input_y <- dtrain_treated$price
```

The input_x is the matrix that is inputed for x in caret::train function,
and the input_y is the label that is inputed for y in caret::train function.

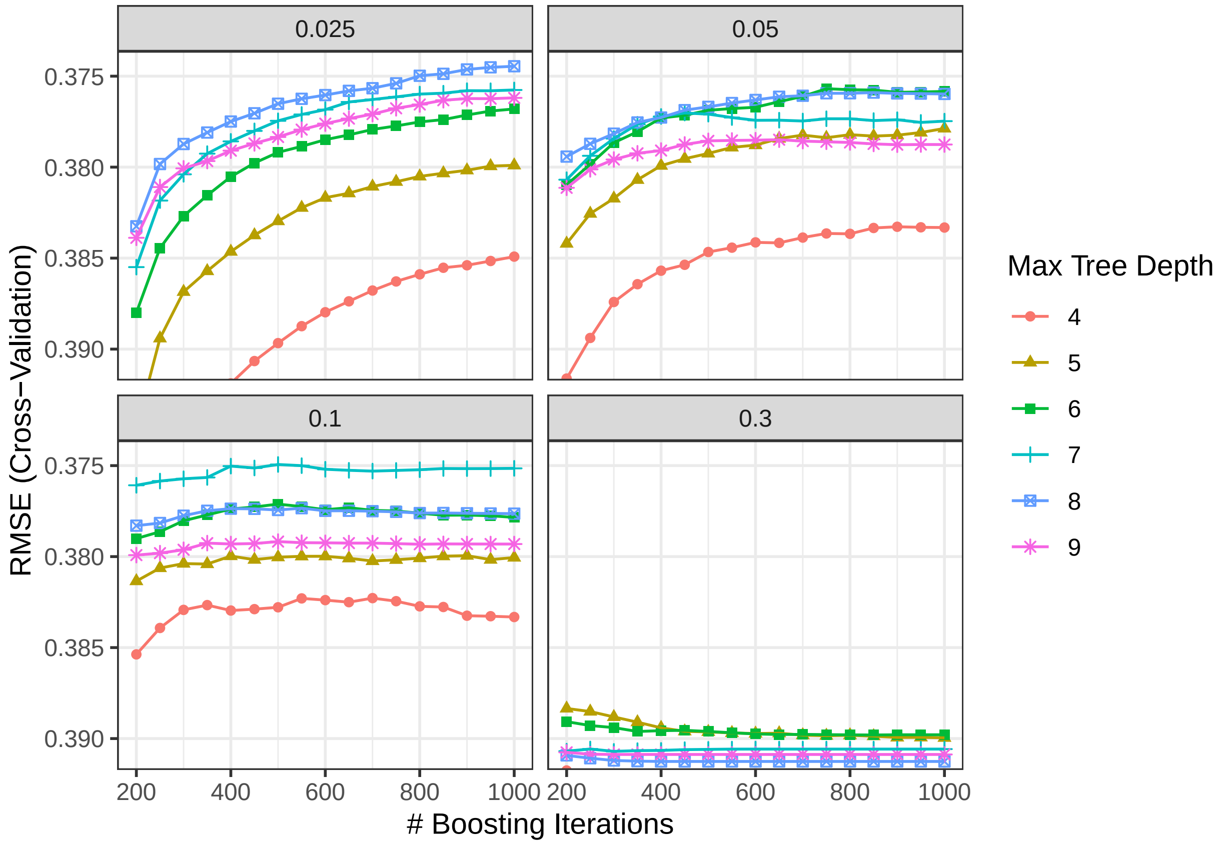
2.1. Tuning maximum tree depth and setting initial eta (learning rate)

```r
start_time <- Sys.time()
# tuning max depth
tune_grid_max_depth <- expand.grid(
  nrounds = seq(from = 200, to = nrounds, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(4, 5, 6, 7, 8, 9),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

tune_control <- caret::trainControl(
  method = "cv",
  number = 5,
  verboseIter = FALSE,
  allowParallel = TRUE
)

model_tuned <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid_max_depth, # tune grid to tune max_depth
  method = "xgbTree",
  verbose = FALSE
)
```

We want to visually see the results, thus we make a helper function to achieve visual representation of data.

```
# helper function for the plots
tuneplot <- function(x, probs = .90) {
  ggplot(x) +
    coord_cartesian(ylim = c(quantile(x$results$RMSE, probs = probs), min(x$results$RMSE))) +
    theme_bw()
}
```

```
tuneplot(model_tuned)
```




```
model_tuned$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 85    1000         8 0.025     0                1                1         1
```

We can see that the best tuned max depth is 6.
We can clearly see in the plot that maxdepth with 6 has lower RMSE than other maxdepth values among the eta values (0.025, 0.05, 0.1, 0.3)

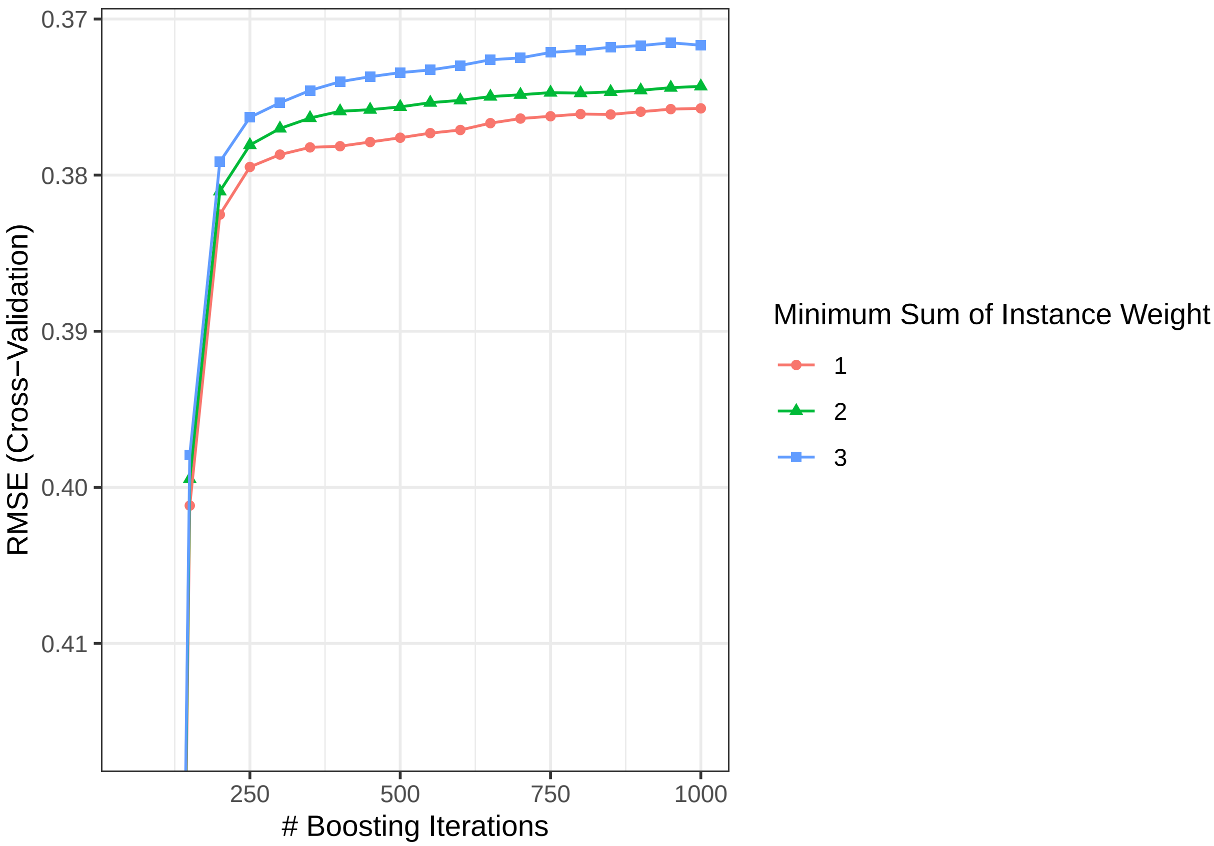
To conclude, we have our best tuned eta being 0.025 for now, and best tuned max depth being 6.

2.2. Tuning the Minimum Child Weight

```
# tuning Min child weight

# tune_grid_mcw is a tune_grid to tune min_child_weight
tune_grid_mcw <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
```

```
  eta = 0.025,
  max_depth = ifelse(model_tuned$bestTune$max_depth == 2,
    c(model_tuned$bestTune$max_depth:4),
    model_tuned$bestTune$max_depth - 1:model_tuned$bestTune$max_depth + 1),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = c(1, 2, 3),
  subsample = 1
)

model_tuned <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid_mcw, # put in the tune_grid_mcw
  method = "xgbTree",
  verbose = TRUE
)
```

```
tuneplot(model_tuned)
```




```
model_tuned$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 59     950         8 0.025     0                1                3         1
```

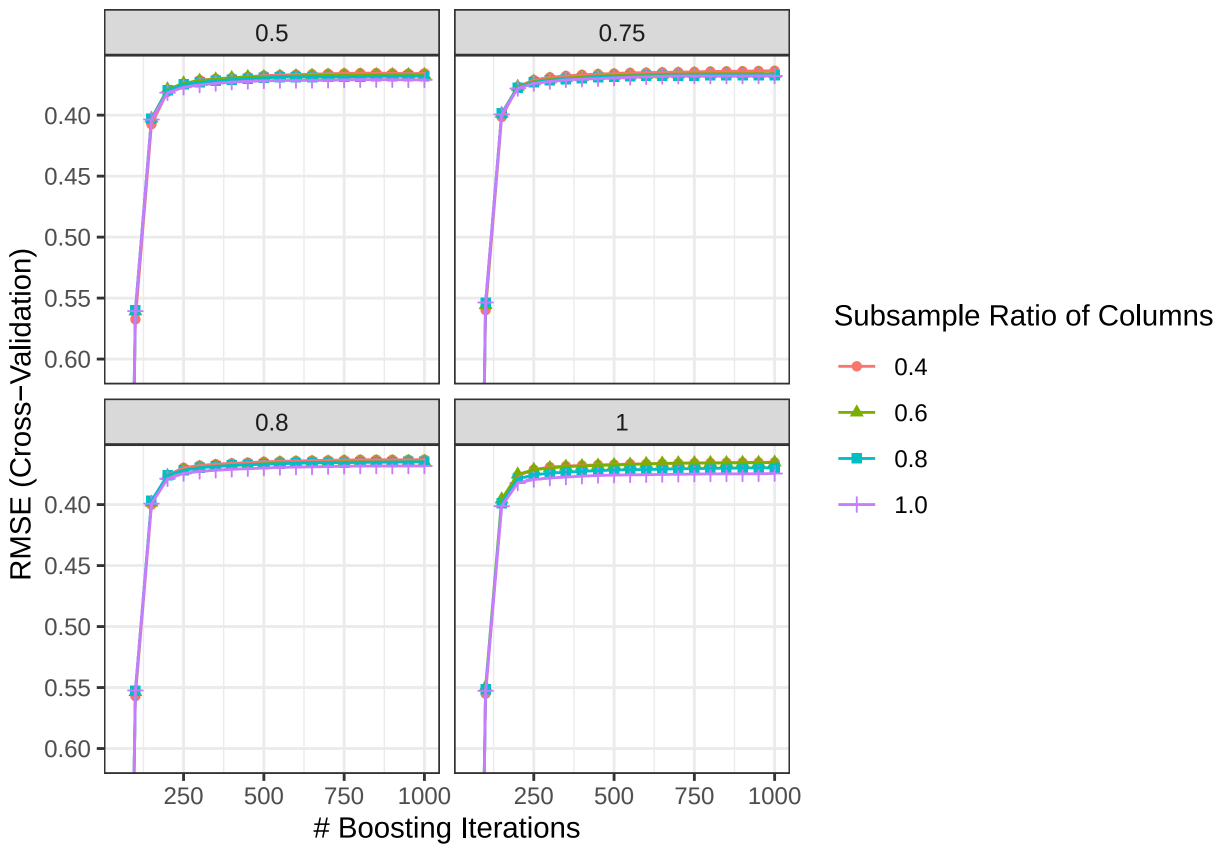
We can see that the best tuned minimum child weight is 1.
We can clearly see in the plot that minimum child weight with 1 has lower RMSE than other minimum child weight values.

To conclude, we have the minimum child weight as 1.

2.3. Tuning the colsample_bytree and the subsample proportion

```r
# tuning colsample bytree and subsample

# tune_grid_cols_sub is a tune_grid to tune the colsample_bytree
# and the subsample proportion
tune_grid_cols_sub <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
  eta = 0.025,
  max_depth = model_tuned$bestTune$max_depth,
  gamma = 0,
  colsample_bytree = c(0.4, 0.6, 0.8, 1.0),
  min_child_weight = model_tuned$bestTune$min_child_weight,
  subsample = c(0.5, 0.75, 0.8, 1.0)
)

model_tuned <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid_cols_sub,
  method = "xgbTree",
  verbose = TRUE
)
```

```r
tuneplot(model_tuned, probs = .95)
```




```r
model_tuned$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 60    1000         8 0.025     0              0.4                3       0.8
```

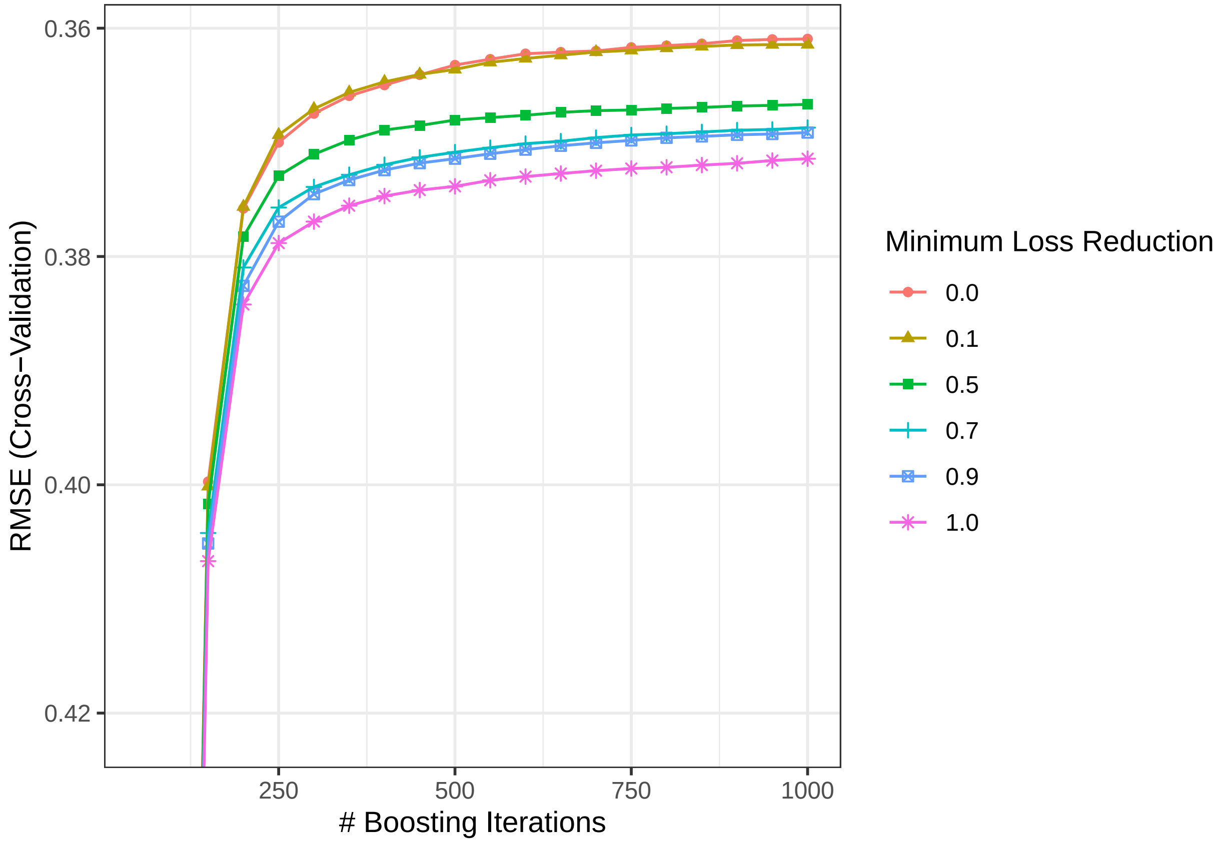
It is not quite clear to see what subsample is better among the 4 subsamples (0.5, 0.75, 0.8, 1). This is also the same issue for col_sample_bytree.
However, if we look at the model_tuned$bestTune,
we are able to identify that the best tune for col_sample_bytree and subsamples are 0.4 and 0.75, respectively.

To conclude, we have our best tuned col_sample_bytree as 0.4 and best tuned subsample as 0.75.

2.4. Tuning the gamma values

```
# tuning gamma

# tune_grid_gamma is a tune_grid to tune the gamma values
tune_grid_gamma <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
  eta = 0.025,
  max_depth = model_tuned$bestTune$max_depth,
  gamma = c(0, 0.1, 0.5, 0.7, 0.9, 1.0),
  colsample_bytree = model_tuned$bestTune$colsample_bytree,
  min_child_weight = model_tuned$bestTune$min_child_weight,
  subsample = model_tuned$bestTune$subsample
)

model_tuned <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid_gamma,
  method = "xgbTree",
  verbose = FALSE
)
```

```
tuneplot(model_tuned)
```




```r
model_tuned$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 20    1000         8 0.025     0              0.4                3       0.8
```

By looking at just the plot, it is debatable, rather rather 0 is a better tuned gamma value or 0.1 is a better tuned one.
Therefore, to check specifically, we look at the model_tuned$bestTune, and observe the value gamma = 0.

To conclude, we have our best tuned gamma value as 0.

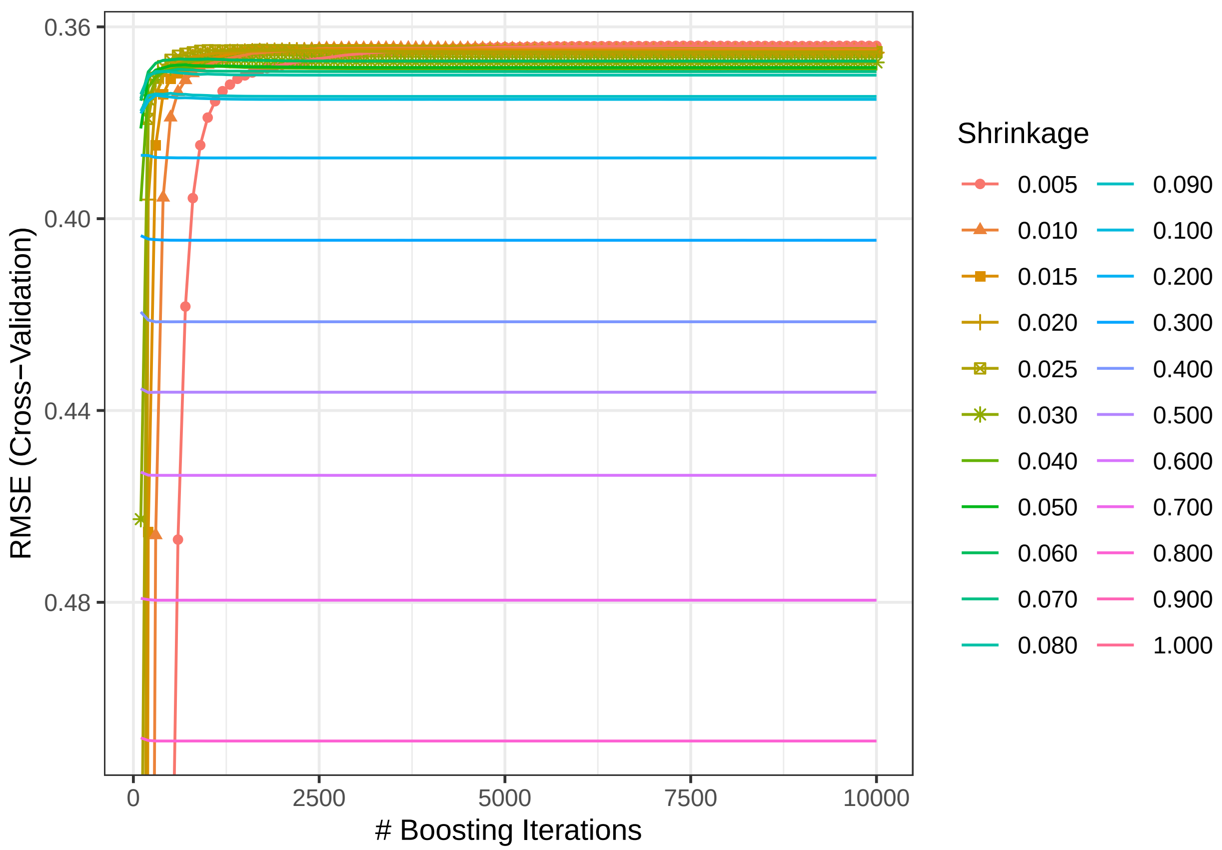
2.5. Tuning eta and nrounds

```r
# tuning nrounds and eta

# tune_grid_nrounds_eta is a tune_grid to tune the nrounds
# and eta values
tune_grid_nrounds_eta <- expand.grid(
  nrounds = seq(from = 100, to = 10000, by = 100),
  eta = c(0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04,
          0.05, 0.06, 0.07, 0.08, 0.09,
          0.10, 0.20, 0.30, 0.40, 0.50,
          0.60, 0.70, 0.80, 0.90, 1),
  max_depth = model_tuned$bestTune$max_depth,
  gamma = model_tuned$bestTune$gamma,
  colsample_bytree = model_tuned$bestTune$colsample_bytree,
  min_child_weight = model_tuned$bestTune$min_child_weight,
  subsample = model_tuned$bestTune$subsample
)
```

```r
model_tuned <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tune_grid_nrounds_eta,
  method = "xgbTree",
  verbose = TRUE
)
```

```r
tuneplot(model_tuned)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 22. Consider
## specifying shapes manually if you must have them.

## Warning: Removed 1600 rows containing missing values (geom_point).
```




```r
model_tuned$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
## 98    9800         8 0.005     0              0.4                3       0.8
```

```r
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 1.597994 hours
```

There are lots of eta values plotted in the above plot.
Therefore, we just look at the model_tuned$bestTune.
We can see that the best value for eta is 0.005 and the

best tuned value for nrounds is 9900.

Note that in the final model I submitted for "20673533.R", had an nround of 10000. However, when I ran the program in Rmarkdown with the same method I got nround of 9900.

3. Model of the all hyper parameters applied

```
(tuned_grid <- expand.grid(
        nrounds = model_tuned$bestTune$nrounds,
        eta = model_tuned$bestTune$eta,
        max_depth = model_tuned$bestTune$max_depth,
        gamma = model_tuned$bestTune$gamma,
        colsample_bytree = model_tuned$bestTune$colsample_bytree,
        min_child_weight = model_tuned$bestTune$min_child_weight,
        subsample = model_tuned$bestTune$subsample
        ))

(tuned_model <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tuned_grid,
  method = "xgbTree",
  verbose = TRUE
))
```

3.1 Comparison with the tuned_model and default grid

The grid with the tuned hyper parameters are as below.

```
(tuned_grid <- expand.grid(
        nrounds = model_tuned$bestTune$nrounds,
        eta = model_tuned$bestTune$eta,
        max_depth = model_tuned$bestTune$max_depth,
        gamma = model_tuned$bestTune$gamma,
        colsample_bytree = model_tuned$bestTune$colsample_bytree,
        min_child_weight = model_tuned$bestTune$min_child_weight,
        subsample = model_tuned$bestTune$subsample
        ))
```

```
##   nrounds   eta max_depth gamma colsample_bytree min_child_weight subsample
## 1    9800 0.005         8     0              0.4                3       0.8
```

And the default grid and trControl in caret::train looks like the followings.

```
grid_default <- expand.grid(
  nrounds = 100,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)
```

Now let's compare the two models. The tuned model, and the default model.

```r
(tuned_model <- caret::train(
  x = input_x,
  y = input_y,
  trControl = tune_control,
  tuneGrid = tuned_grid,
  method = "xgbTree",
  verbose = FALSE
))
```

```
## [06:57:44] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [06:58:47] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [06:59:49] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:00:52] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:01:55] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:02:58] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
```

```
## eXtreme Gradient Boosting
##
## 4855 samples
##  126 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3884, 3884, 3883, 3884, 3885
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3621969  0.7488608  0.241725
##
## Tuning parameter 'nrounds' was held constant at a value of 9800
##
## Tuning parameter 'min_child_weight' was held constant at a value of 3
##
## Tuning parameter 'subsample' was held constant at a value of 0.8
```

```r
(default_model <- caret::train(
  x = input_x, # tr_x is data frame, xgbTree needs matrix
  y = input_y,
  trControl = tune_control,
  tuneGrid = grid_default,
  method = "xgbTree",
  verbose = FALSE
))
```

```
## [07:04:08] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:04:09] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:04:09] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:04:10] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:04:11] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
## [07:04:11] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now deprecated in favor o
```

```
## eXtreme Gradient Boosting
##
## 4855 samples
##  126 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3882, 3885, 3884, 3885, 3884
```

```
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3887906  0.7119977  0.2656564
##
## Tuning parameter 'nrounds' was held constant at a value of 100
## Tuning
##  held constant at a value of 1
## Tuning parameter 'subsample' was held
##  constant at a value of 1
```

Now comparing the hyperparameter tuned, and default model, we
are able to observe that the tuned model has a lower RMSE with a value of
0.3636909, and the MAE is also lower than the default model with a value of
0.2435138.

Therefore, I thought this was an improvement by using tuned hyperparameters, and submitted the final model

3.2 Final model

To conclude, we get the final model being,

(tuned_grid <- expand.grid(
nrounds = 9900,
eta = 0.005,
max_depth = 6,
gamma = 0,
colsample_bytree = 0.4,
min_child_weight = 1,
subsample = 0.75
))

tune_control <- caret::trainControl(
method = "cv",
number = 5,
verboseIter = FALSE,
allowParallel = TRUE
)

(final_model <- caret::train(
x = input_x,
y = input_y,
trControl = tune_control,
tuneGrid = tuned_grid,
method = "xgbTree"
))

4. Citation

I looked over the methods and ideas about how to tune hyperparameters
visually in this site.

Pelkoja. (2018, July 04). Visual xgboost tuning with caret. Retrieved April 14, 2021, from https://www.kaggle.com/pelkoja/visual-xgboost-tuning-with-caret#tuning-xgbtree-with-caret