Lab 10

Tyron Samaroo

11:59PM May 11, 2020

In the first part of this lab, we will be joining three datasets in an effort to make a design matrix that predicts if a bill will be paid on time. Load up the three files:

```
pacman::p_load(tidyverse, magrittr, data.table, R.utils,ggplot2)
bills = fread("https://github.com/kapelner/QC_Math_390.4_Spring_2020/raw/master/labs/bills_dataset/bills
payments = fread("https://github.com/kapelner/QC_Math_390.4_Spring_2020/raw/master/labs/bills_dataset/pa
discounts = fread("https://github.com/kapelner/QC_Math_390.4_Spring_2020/raw/master/labs/bills_dataset/d
```

The unit we care about is the bill. The metric we care about is "paid in full" and it's binary. We would like to build the best design matrix we can and of course generate the y. Warning: this data is highly anonymized and there is likely zero signal! So don't expect to get predictive accuracy. The value of the exercise is in the practice. I think this may be one of the most useful exercises in the entire semester.

I will create the basic steps for you guys. First, join the three datasets in an intelligent way. You will need to examine the datasets beforehand.

```
head(bills)
```

```
##          id   due_date invoice_date   amount customer_id discount_id
## 1: 15163811 2017-02-12   2017-01-13 99490.77    14290629     5693147
## 2: 17244832 2016-03-22   2016-02-21 99475.73    14663516     5693147
## 3: 16072776 2016-08-31   2016-07-17 99477.03    14569622     7302585
## 4: 15446684 2017-05-29   2017-05-29 99478.60    14488427     5693147
## 5: 16257142 2017-06-09   2017-05-10 99678.17    14497172     5693147
## 6: 17244880 2017-01-24   2017-01-24 99475.04    14663516     5693147
```

```
setnames(bills, "amount", "tot_amount")
setnames(payments, "amount", "paid_amount")
head(payments)
```

```
##          id paid_amount transaction_date  bill_id
## 1: 15272980    99165.60       2017-01-16 16571185
## 2: 15246935    99148.12       2017-01-03 16660000
## 3: 16596393    99158.06       2017-06-19 16985407
## 4: 16596651    99175.03       2017-06-19 17062491
## 5: 16687702    99148.20       2017-02-15 17184583
## 6: 16593510    99153.94       2017-06-11 16686215
```

```
head(discounts)
```

```
##         id num_days pct_off days_until_discount
## 1: 5000000       20      NA                  NA
## 2: 5693147       NA       2                  NA
## 3: 6098612       20      NA                  NA
## 4: 6386294      120      NA                  NA
## 5: 6609438       NA       1                   7
```

```
## 6: 6791759       31       1                  NA
```

```r
bills_with_payments = merge(bills, payments, by.x ="id", by.y = "bill_id", all.x = TRUE)
bills_with_payments[, id.y := NULL]
bills_with_payments
```

```
##                id   due_date invoice_date tot_amount customer_id discount_id
##      1:   5000000 2016-07-31   2016-06-16   99480.18    12867871     7397895
##      2:   5693147 2017-05-11   2017-04-11   99528.76    12871311     7397895
##      3:   6098612 2016-01-15   2016-01-04   99477.35    13135347     7397895
##      4:   6386294 2016-12-30   2016-12-30   99479.31    12867871     7397895
##      5:   6609438 2017-05-07   2017-04-07   99477.20    12867871     7397895
##     ---
## 279114: 17619324 2017-02-02   2017-02-02   99478.67    14598456     5693147
## 279115: 17619327 2017-05-09   2017-04-09   99688.54    14475317     7708050
## 279116: 17619331 2017-05-05   2017-05-05   99484.81    14569203     7302585
## 279117: 17619334 2017-06-25   2017-05-11   99572.44    14579003     7302585
## 279118: 17619337 2016-10-22   2016-09-22   99475.44    14755451     7302585
##         paid_amount transaction_date
##      1:    99150.43       2016-07-01
##      2:    99220.42       2016-08-08
##      3:    99148.07       2016-08-03
##      4:    99154.67       2016-07-15
##      5:    99148.07       2016-08-03
##     ---
## 279114:          NA             <NA>
## 279115:          NA             <NA>
## 279116:          NA             <NA>
## 279117:          NA             <NA>
## 279118:    99148.84       2017-07-20
```

```r
bills_payments_discounts = merge(bills_with_payments, discounts, by.x = "discount_id", by.y = "id", all
bills_payments_discounts
```

```
##         discount_id       id   due_date invoice_date tot_amount customer_id
##      1:          NA  7639057 2017-05-27   2017-04-27   99475.01    13853808
##      2:          NA  7708050 2017-02-21   2017-01-22   99475.01    13853808
##      3:          NA  7772589 2017-01-28   2016-12-29   99475.01    13853808
##      4:          NA  7833213 2017-08-07   2017-06-08   99475.01    13853808
##      5:          NA  7944439 2016-10-21   2016-09-21   99475.01    13853808
##     ---
## 279114:     9077537 17320780 2016-01-22   2015-12-23   99476.66    15447467
## 279115:     9077537 17320789 2017-05-07   2017-05-07   99477.77    15447467
## 279116:     9094345 17313859 2017-01-17   2017-01-17   99490.42    15506929
## 279117:     9094345 17315622 2016-08-11   2016-07-12   99501.66    15509141
## 279118:     9094345 17323140 2014-09-03   2014-08-04   99504.91    15506929
##         paid_amount transaction_date num_days pct_off days_until_discount
##      1:          NA             <NA>       NA      NA                  NA
##      2:          NA             <NA>       NA      NA                  NA
##      3:          NA             <NA>       NA      NA                  NA
##      4:          NA             <NA>       NA      NA                  NA
##      5:          NA             <NA>       NA      NA                  NA
##     ---
## 279114:          NA             <NA>       45       0                  NA
## 279115:          NA             <NA>       45       0                  NA
```

```
## 279116:    99175.09       2017-06-26      365      NA                 NA
## 279117:    99194.80       2017-06-28      365      NA                 NA
## 279118:          NA             <NA>      365      NA                 NA
```

Now create the response metric "paid_in_full" and create the design matrix by ensuring the unit is bill. How should you featurize? Should you create some features? What type(s) should they be?

```
bills_data = bills_payments_discounts %>%
  group_by(id) %>%
  summarise(total_paid_amount = sum(paid_amount), customer_id = first(customer_id), discount_id = first
  mutate(total_paid_amount = ifelse(is.na(total_paid_amount), 0, total_paid_amount), paid_in_full = ifel
table(bills_data$paid_in_full, useNA = "always")
```

```
##
##      0      1   <NA>
## 199061  27373      0
```

Fit a tree to this data. Try to use `YARF` if you have it. If not, use the package `rpart`. Below is a guide to installing `YARF` and ensuring it works.

First, ensure you have the Java JDK installed. The JDK is NOT the JRE. The former allows you to compile Java programs and the latter allows you only to run Java programs. Then insure that `rJava` is installed and working. In other words, the following should work and give the same output from practice lecture 12. If it doesn't, try the code that is commented out to reinstall. Google errors. Frustration in libraries and platforms not working on your computer is unfortunately part of computer science and thus part of data science.

```
options(java.parameters = "-Xmx4000m")
pacman::p_load(rJava)
#if that doesn't work, use:
# install.packages("rJava", type = "source")
# library(rJava)
.jinit() #this initializes the JVM in the background and if this runs with no issues nor output, you pr
java_double = .jnew("java/lang/Double", 3.1415)
java_double
```

```
## [1] "Java-Object{3.1415}"
```

```
class(java_double)
```

```
## [1] "jobjRef"
## attr(,"package")
## [1] "rJava"
```

```
.jclass(java_double)
```

```
## [1] "java.lang.Double"
```

```
#call an instance method
.jcall(java_double, "I", "intValue") #java_double.intValue();
```

```
## [1] 3
```

```
#call a static method
J("java/lang/String", "valueOf", java_double) #String.valueOf(java_double);
```

```
## [1] "3.1415"
```

```
#J("java/lang/String", "valueOf", x) #some sort of alphanumeric code for the pointer address
```

It is important to have rJava working on your computer as a fair number of R packages really do make use of it. It's a good thing to have in your toolbox in general.

Now ensure that YARF is installed properly:

```
# pacman::p_install_gh("kapelner/YARF/YARFJARs", ref = "dev")
# pacman::p_install_gh("kapelner/YARF/YARF", ref = "dev")
pacman::p_load(YARF)
```

If that printed out "YARF can now make use of [n] cores", you are in business.

Now create a training-test split and make the tree model and provide oos performance metrics: create a confusion table and compute FDR and FOR.

```
prop_test = 0.20
test_indices = sample(1 : 10000, round((prop_test) * 6000))
bills_data_test = bills_data[test_indices, ]
y_test = bills_data_test$paid_in_full
X_test = bills_data_test
bills_data_test$paid_in_full = NULL


train_indices = setdiff(1 : 6000, test_indices)
bills_data_train = bills_data[train_indices, ]
y_train = bills_data_train$paid_in_full
X_train = bills_data_train
X_train$paid_in_full = NULL

n_train = nrow(X_train)


tree_model = YARFCART(X_train, y_train)
```

```
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 6 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```
y_hat_train = predict(tree_model,X_train)
y_hat_test = predict(tree_model, X_test)
```

```
## Warning in predict.YARF(tree_model, X_test): Prediction set column names did not match training set c
## Attempting to subset to training set columns.
```

```
oos_confusion_table = table(y_test, y_hat_test)
oos_confusion_table
```

```
##       y_hat_test
## y_test    0    1
##      0 1190    0
##      1    0   10
```

```
FDR = oos_confusion_table[1,2] / sum(oos_confusion_table[, 2])
FDR
```

```
## [1] 0
```

```
FOR =  oos_confusion_table[2,1] / sum(oos_confusion_table[, 1])
FOR
```

```
## [1] 0
```

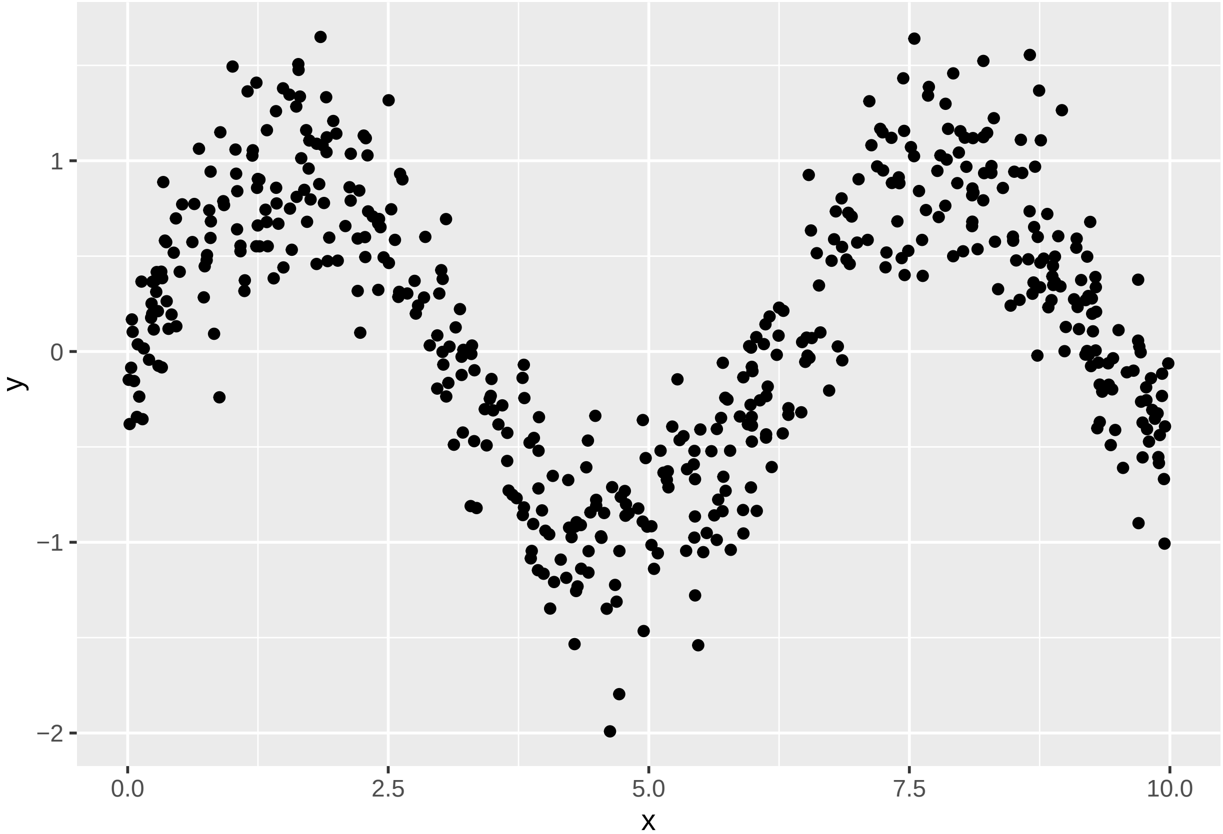
We are done with this unit.

Let's take a look at the simulated sine curve data from practice lecture 12. Below is the code for the data generating process:

```
rm(list = ls())
n = 500
sigma = 0.3
x_min = 0
x_max = 10
x = runif(n, x_min, x_max)
f_x = function(x){sin(x)}
y = f_x(x) + rnorm(n, 0, sigma)
```

Plot an example dataset of size 500:

```
ggplot(, aes(x,y)) + geom_point()
```



Locate the optimal node size hyperparameter for the regression tree model.

```
Nsim = 10
n_test = .25 * n
n_train = n - n_test
training_gs = list()
all_residuals = matrix(NA, nrow = Nsim, ncol = n_test)
for (nsim in 1 : Nsim){

  x_train = runif(n_train, x_min, x_max)
```

```r
  delta_train = rnorm(n_train, 0, sigma)
  y_train = sin(x_train) + delta_train

  g_model = YARFCART(data.frame(x = x_train), y_train,
                     bootstrap_indices = 1 : n_train, calculate_oob_error = FALSE)
  training_gs[[nsim]] = g_model


  x_test = runif(n_test, x_min, x_max)
  delta_test = rnorm(n_test, 0, sigma)
  y_test = sin(x_test) + delta_test
  y_hat_test = predict(g_model, data.frame(x = x_test))
  all_residuals[nsim, ] = y_test - y_hat_test
}
```

```
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
## YARF initializing with a fixed 1 trees...
## YARF after data preprocessed... 1 total features...
## Beginning YARF regression model construction...done.
```

```r
training_gs
```

```
## [[1]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
```

```
##
## [[2]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[3]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[4]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[5]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[6]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[7]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[8]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
```

```
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[9]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
##
## [[10]]
## YARF v1.0 for regression
## Missing data feature ON.
## 1 trees, training data n = 375 and p = 1
## Model construction completed within 0 minutes.
## Run the 'YARF_update_with_oob_results' function to get out of sample
## performance estimates using the out of bag predictions.
```

Plot the regression tree model with the optimal node size.

```
#TO-DO
```

Provide the bias-variance decomposition of this DGP fit with this model. It is a lot of code, but it is in the practice lectures.

```
#TO-DO
```

Load the boston housing data. Leave 25% of the observations oos for honest validation.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```
data(Boston)
Boston = Boston[sample(1:nrow(Boston)), ]
Boston_train = Boston[1 : 380, ]
Boston_test = Boston[381 : nrow(Boston), ]
```

Fit a linear model with all first-order interactions and provide std err of residuals in the test set.

```
mod = lm(medv ~ .*., Boston_train)
mod
```

```
##
## Call:
## lm(formula = medv ~ . * ., data = Boston_train)
##
## Coefficients:
##   (Intercept)          crim            zn         indus          chas
##    -1.570e+02    -2.738e+01    -1.290e-01    -2.630e+00     1.658e+01
##           nox            rm           age           dis           rad
##     4.038e+01     2.596e+01     1.191e+00    -4.082e+00     1.833e+00
##           tax       ptratio         black         lstat       crim:zn
```

```
##     3.611e-02      4.154e+00      5.130e-02      1.279e+00      3.615e-01
##    crim:indus      crim:chas       crim:nox        crim:rm       crim:age
##    -1.306e-01      3.568e+00     -1.744e+00      1.718e-01     -3.226e-03
##      crim:dis       crim:rad       crim:tax   crim:ptratio     crim:black
##    -1.033e-01     -8.672e-01      4.861e-02      8.981e-01     -4.023e-04
##    crim:lstat       zn:indus        zn:chas         zn:nox          zn:rm
##     2.979e-02     -1.885e-03     -8.192e-02      3.372e-01     -4.597e-03
##        zn:age         zn:dis         zn:rad         zn:tax     zn:ptratio
##     3.420e-05      4.831e-03     -1.925e-03      4.458e-04     -7.027e-03
##      zn:black       zn:lstat     indus:chas      indus:nox       indus:rm
##     1.661e-04     -1.340e-02     -4.501e-01      3.134e+00      3.852e-01
##     indus:age      indus:dis      indus:rad      indus:tax  indus:ptratio
##    -1.870e-03     -8.920e-02     -1.011e-02      6.415e-04     -7.946e-02
##   indus:black    indus:lstat       chas:nox        chas:rm       chas:age
##     9.587e-04      5.470e-03     -2.231e+01     -5.412e+00      2.786e-02
##      chas:dis       chas:rad       chas:tax   chas:ptratio     chas:black
##     2.518e+00     -8.986e-01      4.659e-02     -7.194e-01      7.492e-02
##    chas:lstat         nox:rm        nox:age        nox:dis        nox:rad
##    -2.674e-01      8.732e+00     -1.054e+00      1.361e+00     -3.728e-01
##       nox:tax    nox:ptratio      nox:black      nox:lstat         rm:age
##     1.081e-02     -4.372e+00     -4.293e-02      1.769e+00     -3.900e-02
##        rm:dis         rm:rad         rm:tax     rm:ptratio       rm:black
##     5.990e-01     -1.441e-01     -2.097e-02     -6.662e-01     -9.890e-03
##      rm:lstat        age:dis        age:rad        age:tax    age:ptratio
##    -2.614e-01     -2.948e-02      1.916e-02     -4.690e-04     -5.595e-03
##     age:black      age:lstat        dis:rad        dis:tax    dis:ptratio
##    -4.230e-04     -4.240e-03     -8.042e-03     -3.818e-03     -4.446e-02
##     dis:black      dis:lstat        rad:tax    rad:ptratio      rad:black
##     1.481e-03      2.045e-01     -6.523e-05     -6.302e-02      5.952e-04
##     rad:lstat    tax:ptratio      tax:black      tax:lstat  ptratio:black
##    -3.440e-02      6.086e-03      4.874e-06     -1.224e-03      3.392e-03
## ptratio:lstat    black:lstat
##    -1.581e-02     -9.859e-04
```

```r
yhat = predict(mod, Boston_test)
sd(Boston_test$medv - yhat)
```

```
## [1] 3.338034
```

Bag this algorithm with $M = 1000$ and provide std err of residuals in the test set.

```r
colnames(Boston) # last we need set as the y  dont encud
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```r
X = Boston_train
X$medv = NULL
y = Boston_train$medv
X_test = Boston_test
X_test$medv = NULL
M = 10  #I changed this because it talking way too long and keps on crashing on my computer
training_gs =  list()
for (m in 1:M) {
  bag_model = YARFBAG(X,y)
  y_hat_oos = predict(bag_model, Boston_test)
```

```
  training_gs[[m]] = bag_model
}
```

```
## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
```

```
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.

## YARF initializing with a fixed 500 trees...
## YARF after data preprocessed... 13 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

## Warning in predict.YARF(bag_model, Boston_test): Prediction set column names did not match training s
## Attempting to subset to training set columns.
```

```
training_gs
```

```
## [[1]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.2 minutes.
## OOB results on all observations:
##   R^2: 0.86806
##   RMSE: 3.438
##   MAE: 2.337
##   L2: 4490.83
##   L1: 887.91
##
## [[2]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.8713
##   RMSE: 3.395
##   MAE: 2.299
##   L2: 4380.42
##   L1: 873.59
##
## [[3]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.87144
##   RMSE: 3.393
##   MAE: 2.323
```

```
##   L2: 4375.79
##   L1: 882.88
##
## [[4]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.8683
##   RMSE: 3.435
##   MAE: 2.334
##   L2: 4482.49
##   L1: 886.88
##
## [[5]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.87155
##   RMSE: 3.392
##   MAE: 2.324
##   L2: 4371.79
##   L1: 883.08
##
## [[6]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.87027
##   RMSE: 3.409
##   MAE: 2.325
##   L2: 4415.53
##   L1: 883.55
##
## [[7]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.2 minutes.
## OOB results on all observations:
##   R^2: 0.86523
##   RMSE: 3.474
##   MAE: 2.362
##   L2: 4587.13
##   L1: 897.6
##
## [[8]]
## YARF v1.0 for regression
## Missing data feature ON.
```

```
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.86971
##   RMSE: 3.416
##   MAE: 2.342
##   L2: 4434.56
##   L1: 890
##
## [[9]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.19 minutes.
## OOB results on all observations:
##   R^2: 0.87424
##   RMSE: 3.356
##   MAE: 2.298
##   L2: 4280.33
##   L1: 873.31
##
## [[10]]
## YARF v1.0 for regression
## Missing data feature ON.
## 500 trees, training data n = 380 and p = 13
## Model construction completed within 0.21 minutes.
## OOB results on all observations:
##   R^2: 0.86772
##   RMSE: 3.442
##   MAE: 2.35
##   L2: 4502.42
##   L1: 893.11
```

What is your gain over the unbagged model? Why is there a gain?

The gain come from reducing the variance. The gain comes from randomly sampling the data into pieces and then training on each. There is also free validation that is provided. The model return a R^2 of 86% and a RSME of 3.394