Statistics 452: Statistical Learning and Prediction

Review Part 2: Predicting a HUI score

Brad McNeney

2018-11-26

Data

- One of the datasets given to the Stat 652 class includes 590 explanatory variables and a health utilities index score, called HUIDHSI.

This derived variable is a Health Utilities Index which provides a description of an individual's overall functional health, based on eight attributes: vision, hearing, speech, ambulation (ability to get around), dexterity (use of hands and fingers), emotion (feelings), cognition (memory and thinking) and pain (in HUP module). The version of the index used in CCHS is adapted from the HUI Mark 3 (HUI3). The index is designed to produce an overall health utility score. This multi-attribute utility index produces a score ranging from 1.000 (perfect health), through 0.000 (health status equal to death) to -0.360 (health status worse than death).

- I have unzipped the data file in my copy of the Project652 directory on github.

```
hs <- read.csv("../../Project652/HStrain.csv")
```

- 591 variables, grouped into 38 categories, indicated by the first three letters of the variable names:

```
cn <- colnames(hs)
table(substr(cn,start=1,stop=3))
```

```
##
## ADL ADM ALC CAG CCC CGE CIH CR1 CR2 DHH DPS DS2 EDU FAL GEN GEO HC2 HUI
##   4   4   5  45  31   8  27  34  19   9  33   4   2  15  10   2  19   1
## HUP HWT IAL IN2 LBF LON MED NUR OH3 OWN PA2 RET RPL SDC SLP SLS SMK SPA
##   1   5   6   8  19   4  33  12  27   2  49  32  19   6   1   6  21  24
## SSA TRA
##  25  19
```

```
head(sort(cn),n=4) # Activities of Daily Living
```

```
## [1] "ADL_04A" "ADL_06A" "ADLDCLS" "ADLDTOI"
```

Survey information not useful for prediction

- The variables that start with `ADM` are to do with administering the survey and are not useful for prediction.
- For example, `ADM_RNO` is a sequential record number, `ADM_N09` indicates whether the interview was by phone, in-person, etc.
- I will remove these.

```
library(dplyr)
hs <- select(hs,-starts_with("ADM"))
```

Summary variables

- Several categories of variable have an overall summary score, or classification, developed by survey experts.
- For example, the Activities of Daily Living (ADL) variables are summarized by `ADLDCLS` (page 158 of data dictionary):

ADLDCLS - Instrumental & Basic Activities of Daily Living Class. - Based on ADLDCLST and ADLDMEA. This variable is an overall summary measure of ratings of the ADL capacity-instrumental and physical dimensions.The instrument and the derived variable classification are developed from the activities of daily living component of the OARS Multidimensional Functional Assessment Questionnaire (OMFAQ). See documentation on derived variables.

```
summary(hs$ADLDCLS)
```

```
##  MILD IMPAIRMENT   MOD IMPAIRMENT   NO FUNC IMPAIR   SEV IMPAIRMENT
##             1098              258             8567               56
## TOTAL IMPAIRMENT
##               21
```

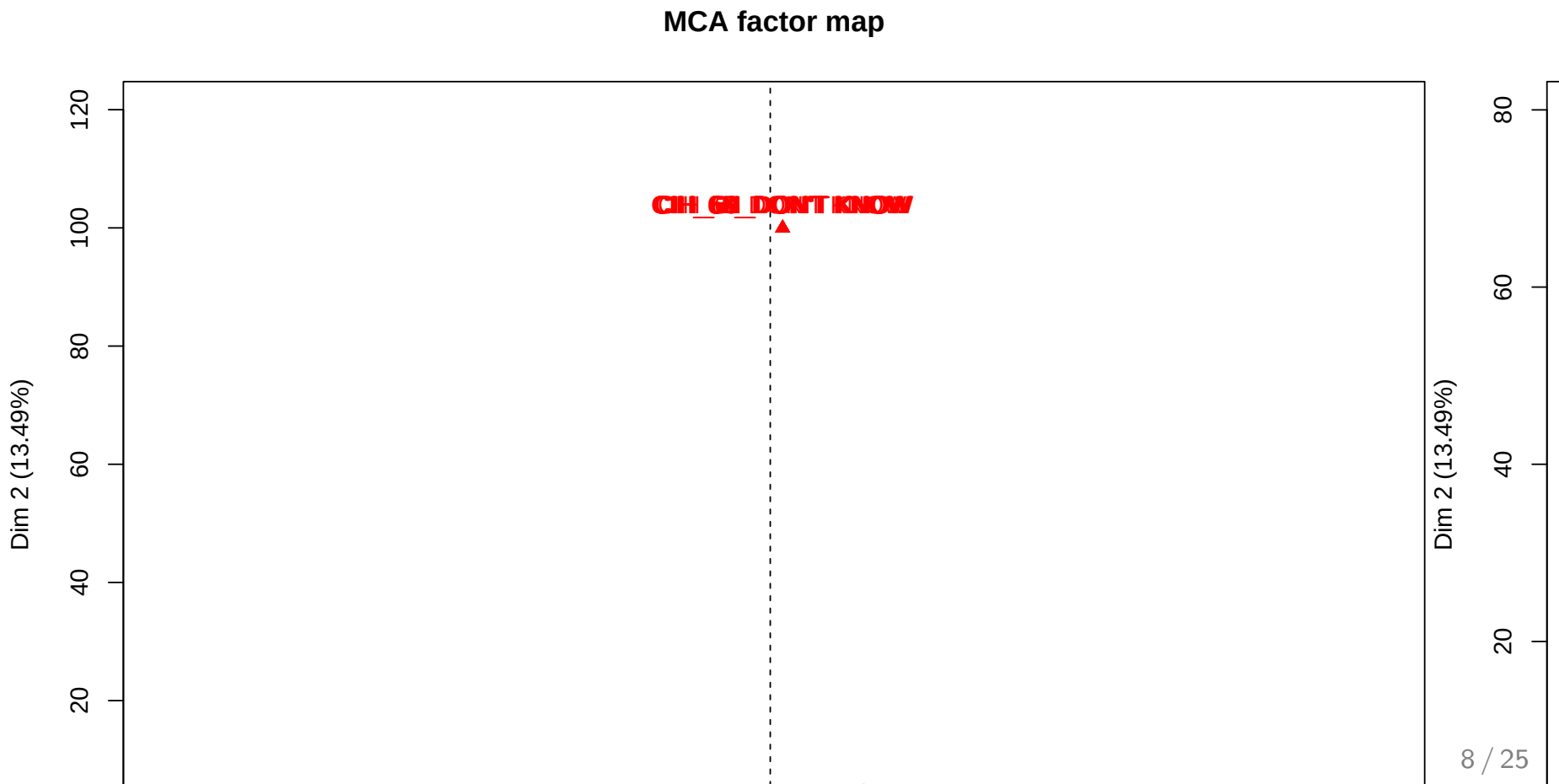
Choice of summary variable

- Some sets of variables do not have a single score, but may have several that could be useful.
- For example the caregive variables `CAG` tell us about care responsibilities (parent, spouse, neighbor, etc.)
  - `CAGDFAP` records the frequency of care (rarely, monthly, weekly, daily, etc.)
  - `CAGDIAP` records frequency and number of hours (daily - 1 hour, daily - 3 hours, etc.)

Making our own score

- Some groups of variables have no summary score.
- Can compute a few PCs from a group of survey questions.
- Example, `CIH` variables (questions about improving health).

```
library(FactoMineR)
res.mca <- MCA(select(hs,starts_with("CIH")))
```



```r
CIHPCs <- res.mca$ind$coord[,1:4] # first 4 explain 50%
colnames(CIHPCs) <- paste("CIH",colnames(CIHPCs))
```

My choices

```
hsred <- select(hs,
           ADLDCLS,ALCDTTM,CAGDFAP,CCCF1,CCCDCPD,
           CR1FRHC,CR2DTHC,CR2DFAR,DPSDSF,EDUDR04,
           FALG02,GENDHDI,GENDMHI,HC2FCOP,
           HUIDHSI, # response
           HUPDPAD,HWTGBMI,IN2GHH,LONDSCR,MEDF1,
           NURDHNR,PA2DSCR,SLP_02,SLSDCLS,
           SMKDSTY,SPAFPAR,starts_with("SSAD"))
hsred <- data.frame(hsred,CIHPCs)
```

Centre and Scale

- We will apply different statistical methods, some of which depend on scaling of the features.
- To make all results comparable, scale the features now.
- First have to convert factors to dummy variables with `model.matrix()`.
  - Leave out the intercept, though.

```
tem <- model.matrix(HUIDHSI ~ .,data=hsred)[,-1]
X <- as.data.frame(scale(tem))
Y <- hsred$HUIDHSI  # could also scale Y, but we won't
```

Training and test sets

- To compare methods we'll divide our 10000 observations into training and test sets.
- I'll go with a 70% training set.

```
set.seed(123)
n.train <- 7000
train <- sample(1:nrow(hs),replace=FALSE,size=n.train)
X.train <- X[train,]; Y.train <- Y[train]
X.test <- X[-train,]; Y.test <- Y[-train]
```

Subset selection

```
library(leaps)
rr <- regsubsets(X.train,Y.train,nvmax=40,
                 method="forward")
```

```
## Reordering variables and trying again:
```

```
ss <- summary(rr)
pbest <- which.min(ss$bic)
pbest
```

```
## [1] 29
```

```
# coef(rr,id=29)
```

- Important variables are from Activities of Daily Living, general health, satisfaction with life, smoking status, the first "improve health" PC, and some others.
- Also includes `HUPDPAD`, which should have been exlcuded from the dataset, because it is one of the variables used to create the response `HUIDHSI` (oops).

Test MSE of subset regression

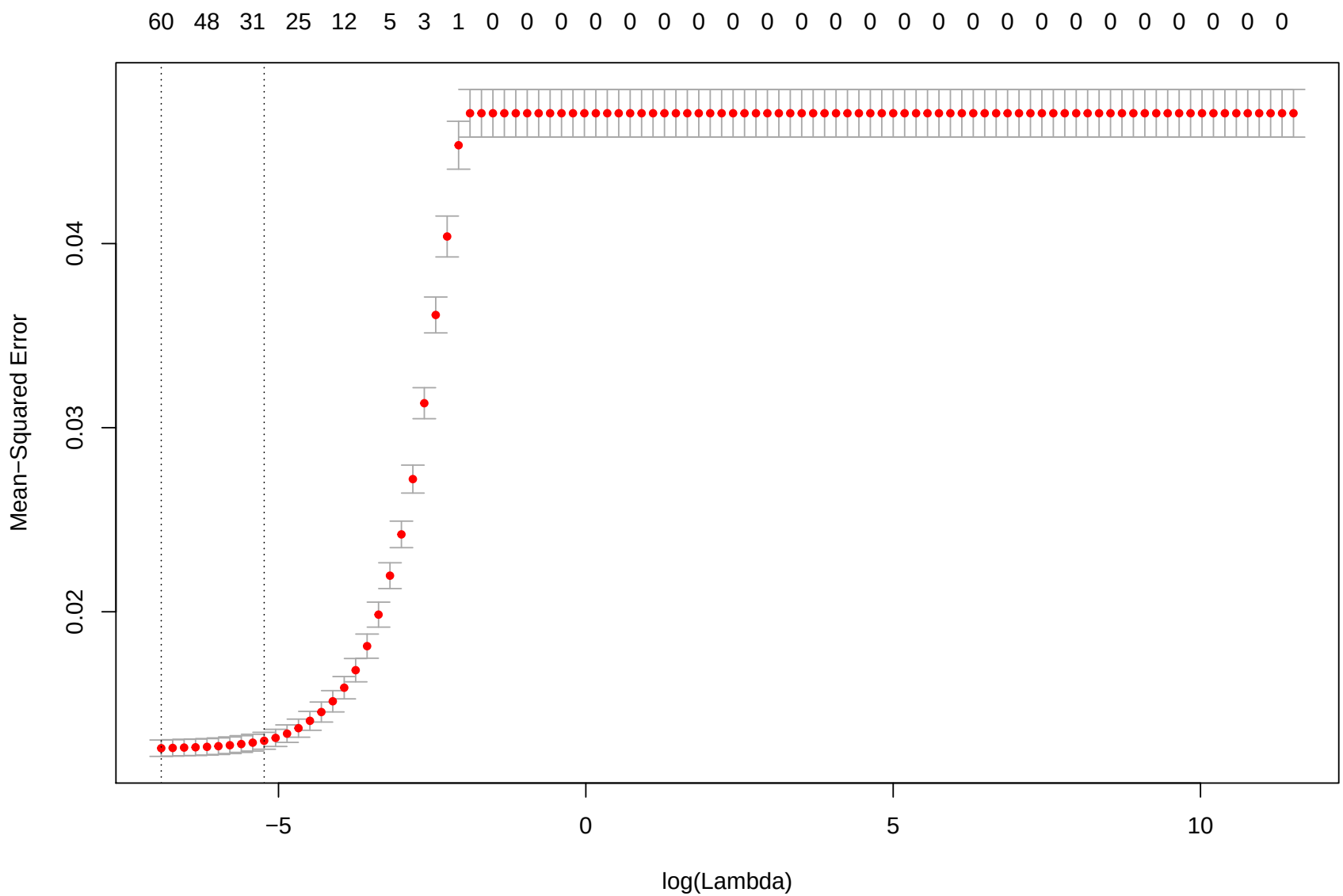
```
cols<- ss$which[pbest,-1] # don't include intercept
Xred <- as.matrix(X.test[,cols])
pred.test <- cbind(1,Xred) %*%  coef(rr,id=29)
mean((Y.test - pred.test)^2)
```

```
## [1] 0.04234925
```

Lasso

```
library(glmnet)
lambdas <- 10^{seq(from=-3,to=5,length=100)}
cv.lafit <- cv.glmnet(as.matrix(X.train),Y.train,alpha=1,lambda=lambdas)
```

```r
plot(cv.lafit)
```




```r
la.best.lam <- cv.lafit$lambda.1se
```

Lasso coefficients

- A similar (but not identical) set of non-zero coefficients.

```
ll <- glmnet(as.matrix(X.train),Y.train,alpha=1,lambda=la.best.lam)
coef(ll)
```

```
## 81 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                0.831988833
## ADLDCLSMOD IMPAIRMENT     -0.015224419
## ADLDCLSNO FUNC IMPAIR      0.032343545
## ADLDCLSSEV IMPAIRMENT     -0.011790791
## ADLDCLSTOTAL IMPAIRMENT   -0.004473601
## ALCDTTMOCCASIO. DRINKER    .
## ALCDTTMREGULAR DRINKER     .
## CAGDFAPOCC OR RARELY       .
## CAGDFAPREG BASIS DLY       .
## CAGDFAPREG BASIS LESS      .
## CAGDFAPREG BASIS MNTH      .
## CAGDFAPREG BASIS WK        .
## CCCF1HAS NO CHRON CON      0.004032078
## CCCDCPDNOT HAVE COPD       .
## CR1FRHCREC FORMAL H C     -0.001664231
## CR2DTHCDID NOT REC H C     0.012838025
## CR2DTHCFORMAL H C ONLY     .
## CR2DTHCINFORMAL H C ONL    .
## CR2DFAROCC OR RARELY       .
## CR2DFARREG BASIS DLY      -0.001881672
## CR2DFARREG BASIS LESS      .
## CR2DFARREG BASIS MNTH      .
## CR2DFARREG BASIS WK        .
## DPSDSF                    -0.001131551
## EDUDR04OTHER POST-SEC.     .
## EDUDR04POST-SEC. GRAD.     0.002437967
```

```r
pred.test <- predict(ll,as.matrix(X.test))
mean((Y.test-pred.test)^2)
```

```
## [1] 0.01378689
```

Random forests

- Computationally intensive, and computation grows with the number of features and observations.
- With many features and a large sample size, may need to filter some out, or need to reduce the sample size while we explore which features are important.

Use features found by lasso or subset selection

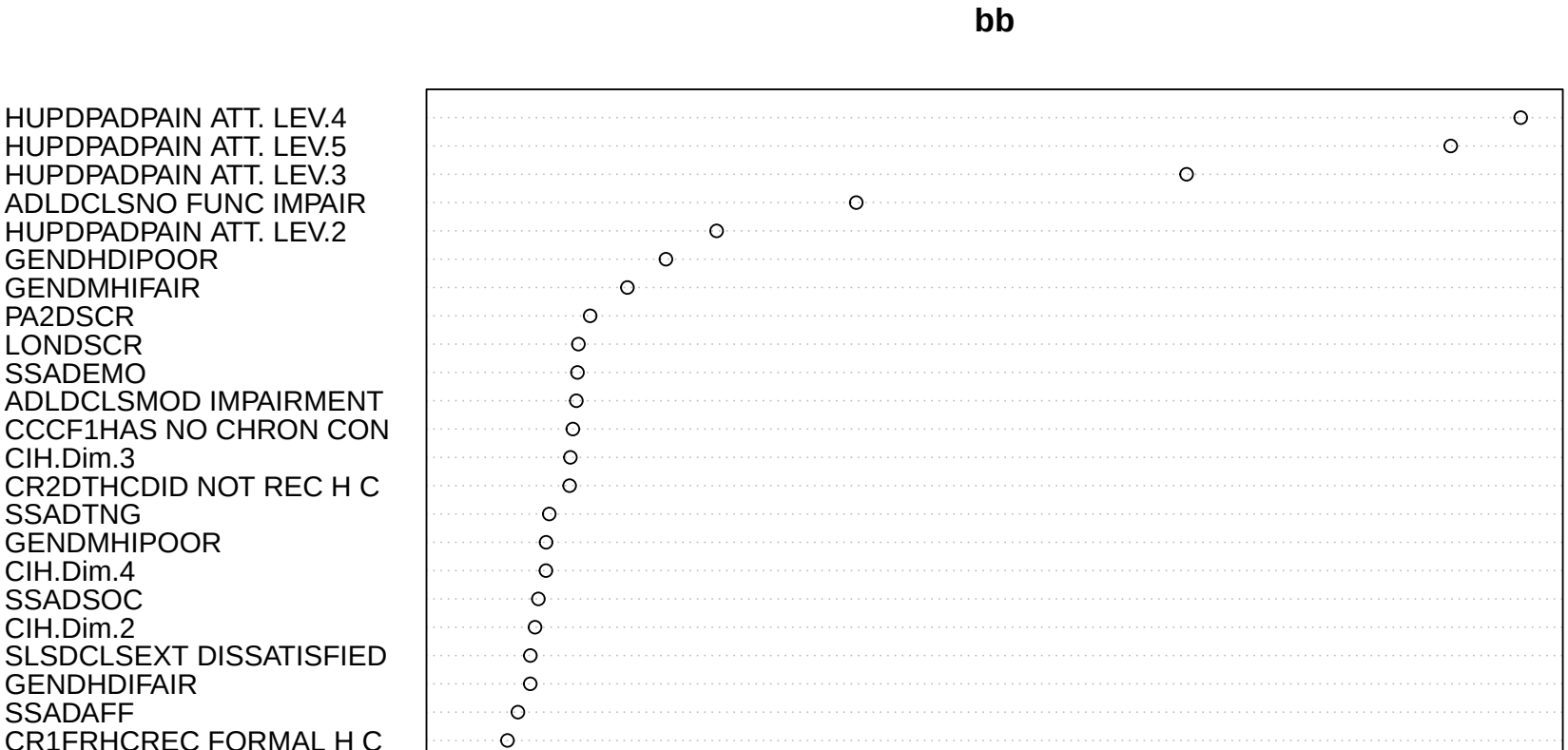
- Could do something like the following to use a subset of the features.
  - Here I pick out the features chosen by lasso

```
nonz <- (as.numeric(coef(ll))!=0)[-1] # rm intercept
hsred2.train <- data.frame(HUIDHSI=Y.train,X.train[,nonz])
```

Random forest with all features

- My laptop can handle all features, but I've chosen to build only 200 trees.

```
library(randomForest)
set.seed(1)
bb <- randomForest(X.train,y=Y.train,xtest=X.test,
          ytest=Y.test,ntree=200,
          mtry=sqrt(ncol(X.train)),importance=TRUE)
varImpPlot(bb,type=1) # HUPDPAD levels 4,5,3 important
```



```r
pred.test <- bb$test$predicted
mean((Y.test - pred.test)^2)
```

```
## [1] 0.0146929
```

Boosting

- Can specify interaction depth; I tried 1 and 2, with 2 performing better.
- The number of trees and the shrinkage factor are tuning parameters.
  - To limit computation I'm using 200 trees.
  - I've left the shrinkage at its default value.
  - I am told that the `caret` R package has useful tools for tuning boosting, but I've never tried it myself.

```
library(gbm)
hs.train <- data.frame(HUIDHSI=Y.train,X.train)
hboost <- gbm(HUIDHSI ~ ., data=hs.train,
              n.trees=200,interaction.depth=2,
              distribution="gaussian")
summary(hboost)
```

```r
library(gbm)
hs.test <- data.frame(HUIDHSI=Y.test,X.test)
pred.test <- predict(hboost,newdata=hs.test, n.trees=200,type="response")
mean((Y.test-pred.test)^2)
```

```
## [1] 0.01346971
```

The winner

- Boosting with interaction depth 2 gave the lowest MSE.
  - I didn't use many trees for random forest, and made almost no attempt to tune boosting.
- Fit the winner to all data – this is my $\hat{f}$ to predict the hold-out test data on Monday.

```
hs <- data.frame(HUIDHSI=Y,X)
winner <- gbm(HUIDHSI ~ ., data=hs,
              n.trees=200,interaction.depth=2,
              distribution="gaussian")
```