MACS 33002: Homework #3

Xin Feng

Due Monday, Feb 17th by 5pm

```
## -- Attaching packages ------------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Registered S3 method overwritten by 'mosaic':
##   method                           from
```

```
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum

##
## Attaching package: 'modelr'

## The following object is masked from 'package:broom':
##
##     bootstrap

## The following object is masked from 'package:mosaic':
##
##     resample

## The following object is masked from 'package:ggformula':
##
##     na.warn

## Loading required package: carData

##
```

```
## Attaching package: 'car'

## The following objects are masked from 'package:mosaic':
##
##     deltaMethod, logit

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':
##
##     nasa

## dummies-1.5.6 provided by Decision Patterns

##
## Attaching package: 'MASS'

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

##
## Attaching package: 'rcfss'

## The following object is masked from 'package:modelr':
##
##     mse

## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.

##
## Attaching package: 'yardstick'

## The following objects are masked from 'package:modelr':
##
##     mae, mape, rmse

## The following object is masked from 'package:readr':
##
##     spec
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

Decision Trees

1. Set up the data and store some things for later use:
   - Set seed
   - Load the data
   - Store the total number of features minus the biden feelings in object `p`
   - Set λ (shrinkage/learning rate) range from 0.0001 to 0.04, by 0.001

```
set.seed(123)
#Load the Data
defaultDataDir = "/Users/Liz/Desktop/problem-set-3-master 2/data"
fileName = "nes2008.csv"
fileLocation = file.path(defaultDataDir, fileName)
NESdta = read.csv(file = fileLocation, header = T, na.strings = "?")
#Store in Object p
p = 5
lamda = seq(from=0.0001,to=0.04, by=0.001)
```

2. (10 points) Create a training set consisting of 75% of the observations, and a test set with all remaining obs. **Note**: because you will be asked to loop over multiple λ values below, these training and test sets should only be integer values corresponding with row IDs in the data. This is a little tricky, but think about it carefully. If you try to set the training and testing sets as before, you will be unable to loop below.

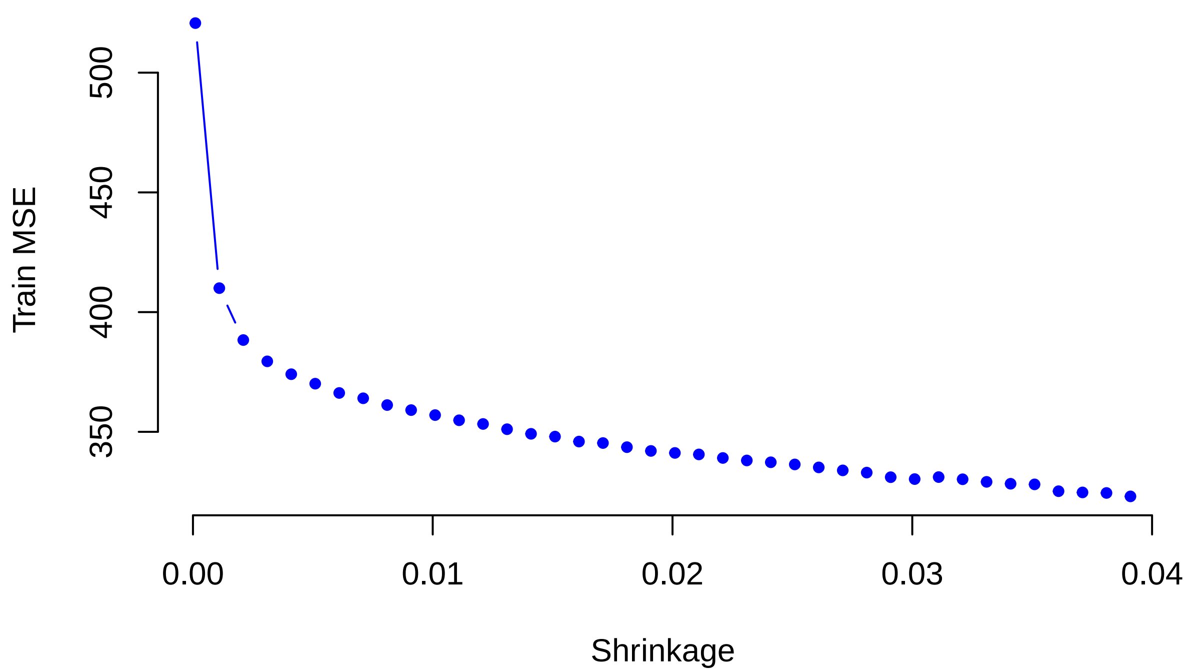
```
set.seed(1000)
num = seq(from=1, to=1807, by=1)
train=sample(1:nrow(NESdta),1355)
test = rep("NA", times = 452)
j = 1
for (i in 1:length(num)) {
  if(is.element(num[i], train) == FALSE){
    test[j] = num[i]
    j = j + 1
  }
}
test=sample(test)
```

3. (15 points) Create empty objects to store training and testing MSE, and then write a loop to perform boosting on the training set with 1,000 trees for the pre-defined range of values of the shrinkage parameter, λ. Then, plot the training set and test set MSE across shrinkage values.

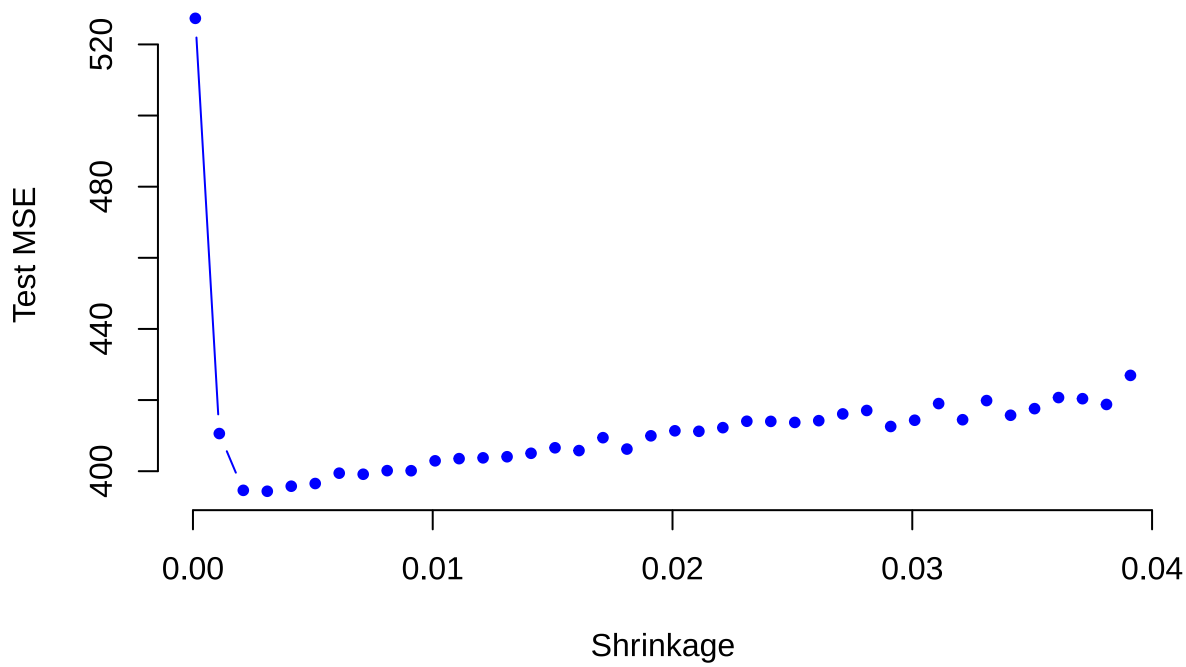
```
set.seed(1234)

library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
length.lamda <-  length(lamda)
train.errors <-  rep(NA, length.lamda)
test.errors <-  rep(NA, length.lamda)

for (i in 1:length.lamda) {
  boost.nes <- gbm(biden ~ .,
                   data=NESdta[train,],
                   distribution="gaussian",
                   n.trees=1000,
                   shrinkage=lamda[i],
                   interaction.depth = 4)
  train.pred <-  predict(boost.nes, newdata=NESdta[train,],n.trees=1000)
  test.pred <-  predict(boost.nes, newdata=NESdta[test,], n.trees=1000)
  train.errors[i] <-  mean((NESdta[train,]$biden - train.pred)^2)
  test.errors[i] <-  mean((NESdta[test,]$biden - test.pred)^2)
}

plot(lamda, train.errors, type="b",
     xlab="Shrinkage", ylab="Train MSE",
     col="blue", pch=20, bty = "n")
```




```r
plot(lamda, test.errors, type="b",
     xlab="Shrinkage", ylab="Test MSE",
     col="blue", pch=20, bty = "n")
```



4. (10 points) The test MSE values are insensitive to some precise value of λ as long as its small enough. Update the boosting procedure by setting λ equal to 0.01 (but still over 1000 trees). Report the test MSE and discuss the results. How do they compare?

```
set.seed(200)
boost.nes <- gbm(biden ~ .,
                 data=NESdta[train,],
                 distribution="gaussian",
                 n.trees=1000,
                 shrinkage=0.01,
                 interaction.depth = 4)
test.pred <-  predict(boost.nes, newdata=NESdta[test,], n.trees=1000)
mean((NESdta[test,]$biden - test.pred)^2)
```

```
## [1] 400.7
```

The means square error of lambda = 0.01 and 1000 trees is 400.7. This is at a relatively stable value, when lambda gets larger, the MSE fluctuates around 400.

5. (10 points) Now apply bagging to the training set. What is the test set MSE for this approach?

```
set.seed(300)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
bag.nes <-  randomForest(biden ~ ., data=NESdta[train,], ntree=1000, importance=TRUE)
bag.pred <-  predict(bag.nes, NESdta[test,])
mean((NESdta[test,]$biden - bag.pred)^2)
```

```
## [1] 400.7
```

The test MSE for this approach is 400.7.

6. (10 points) Now apply random forest to the training set. What is the test set MSE for this approach?

```
set.seed(400)
rf.nes <-  randomForest(biden ~ ., data=NESdta[train,], ntree=1000, importance=TRUE)
rf.pred <-  predict(rf.nes, NESdta[test,])
mean((NESdta[test,]$biden - rf.pred)^2)
```

```
## [1] 400
```

The test MSE for this approach is 400.

7. (5 points) Now apply linear regression to the training set. What is the test set MSE for this approach?

```
set.seed(500)
biden_fu <- lm(biden ~ female + age + educ + dem + rep, data = NESdta[train,])
(test_mse <- augment(biden_fu, newdata = NESdta[test,]) %>%
  mse(truth = biden, estimate = .fitted))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        386.
```

The test set MSE for this approach is 386.2.

8. (5 points) Compare test errors across all fits. Discuss which approach generally fits best and how you concluded this.

The boost(lamda = 0.01) MSE is 400.7. The bagging MSE is 400.7. The random forest MSE is 400. The linear regression MSE is 386.2. In this case, our linear regression fits the best since the MSE has the smallest value.

Support Vector Machines

Goal: Find a classification solution by tuning support vector classfiers that optimially predict whether customers buy either Cirtrus Hill (CH) or Minute Maid (MM) Orange Juice.

1. Create a training set with a random sample of size 800, and a test set containing the remaining observation.

```
set.seed(3)
tr <- sample(1:nrow(OJ), 800)
oj.train <- OJ[tr,]
oj.test <- OJ[-tr,]
```

2. (10 points) Fit a support vector classifier to the training data with `cost = 0.01`, with `Purchase` as the response and *all* other features as predictors. Discuss the results.

```
svm_linear <- svm(Purchase ~ ., data = oj.train,
                  kernel = 'linear',
                  cost = 0.01)
summary(svm_linear)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  434
##
##  ( 216 218 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

3. (5 points) Display the confusion matrix for the classification solution, and also report both the training and test set error rates.

```
require(caret)
```

```
## Loading required package: caret
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall
## The following object is masked from 'package:mosaic':
##
##     dotPlot
## The following object is masked from 'package:purrr':
##
##     lift
```

```
oj.pred <- predict(svm_linear, oj.test)
table(predicted = oj.pred, true = oj.test$Purchase)
```

```
##          true
## predicted  CH  MM
##        CH 139  22
##        MM  24  85
```

```
postResample(predict(svm_linear, oj.train), oj.train$Purchase)
```

```
## Accuracy    Kappa
##   0.8263   0.6294
```

```
postResample(predict(svm_linear, oj.test), oj.test$Purchase)
```

```
## Accuracy    Kappa
##   0.8296   0.6451
```

4. (10 points) Find an optimal cost in the range of 0.01 to 1000 (specific range values can vary; there is no set vector of range values you must use).

```
svm_linear_tune <-
  train(Purchase ~ ., data = oj.train, method = 'svmLinear2',trControl = trainControl(metho
svm_linear_tune
```

```
## Support Vector Machines with Linear Kernel
##
## 800 samples
##  17 predictor
##   2 classes: 'CH', 'MM'
##
## Pre-processing: centered (17), scaled (17)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 720, 720, 720, 720, 720, 720, ...
## Resampling results across tuning parameters:
##
##   cost      Accuracy  Kappa
##     0.010  0.8237    0.6229
##     5.273  0.8275    0.6323
##    10.535  0.8225    0.6218
```

```
##    15.798  0.8250    0.6277
##    21.061  0.8225    0.6228
##    26.323  0.8225    0.6232
##    31.586  0.8225    0.6232
##    36.848  0.8225    0.6230
##    42.111  0.8225    0.6230
##    47.374  0.8225    0.6230
##    52.636  0.8225    0.6230
##    57.899  0.8225    0.6225
##    63.162  0.8225    0.6225
##    68.424  0.8237    0.6244
##    73.687  0.8237    0.6244
##    78.949  0.8237    0.6244
##    84.212  0.8237    0.6244
##    89.475  0.8237    0.6244
##    94.737  0.8237    0.6244
##   100.000  0.8237    0.6244
##
## Accuracy was used to select the optimal model using the
##  largest value.
## The final value used for the model was cost = 5.273.
```

5. (10 points) Compute the optimal training and test error rates using this new value for cost. Display the confusion matrix for the classification solution, and also report both the training and test set error rates. How do the error rates compare? Discuss the results in substantive terms (e.g., how well did your optimally tuned classifer perform? etc.)

```r
postResample(predict(svm_linear_tune, oj.train), oj.train$Purchase)
```

```
## Accuracy    Kappa
##   0.8387   0.6577
```

```r
postResample(predict(svm_linear_tune, oj.test), oj.test$Purchase)
```

```
## Accuracy    Kappa
##   0.8370   0.6627
```

```r
oj.pred2 <- predict(svm_linear_tune, oj.test)
table(predicted = oj.pred, true = oj.test$Purchase)
```

```
##          true
## predicted  CH  MM
##        CH 139  22
##        MM  24  85
```

The accuracy of the optimal model gets better. The optimally tuned classifer performs pretty good with the type I error rate equals to 22 and type II error rate equals to 24.