Pset 3, Machine Learning

Rohen Shah

2/17/2020

Decision Trees

Question 1

```
set.seed(42)
nes <- read_csv("nes2008.csv")
```

```
## Parsed with column specification:
## cols(
##   biden = col_double(),
##   female = col_double(),
##   age = col_double(),
##   educ = col_double(),
##   dem = col_double(),
##   rep = col_double()
## )
```

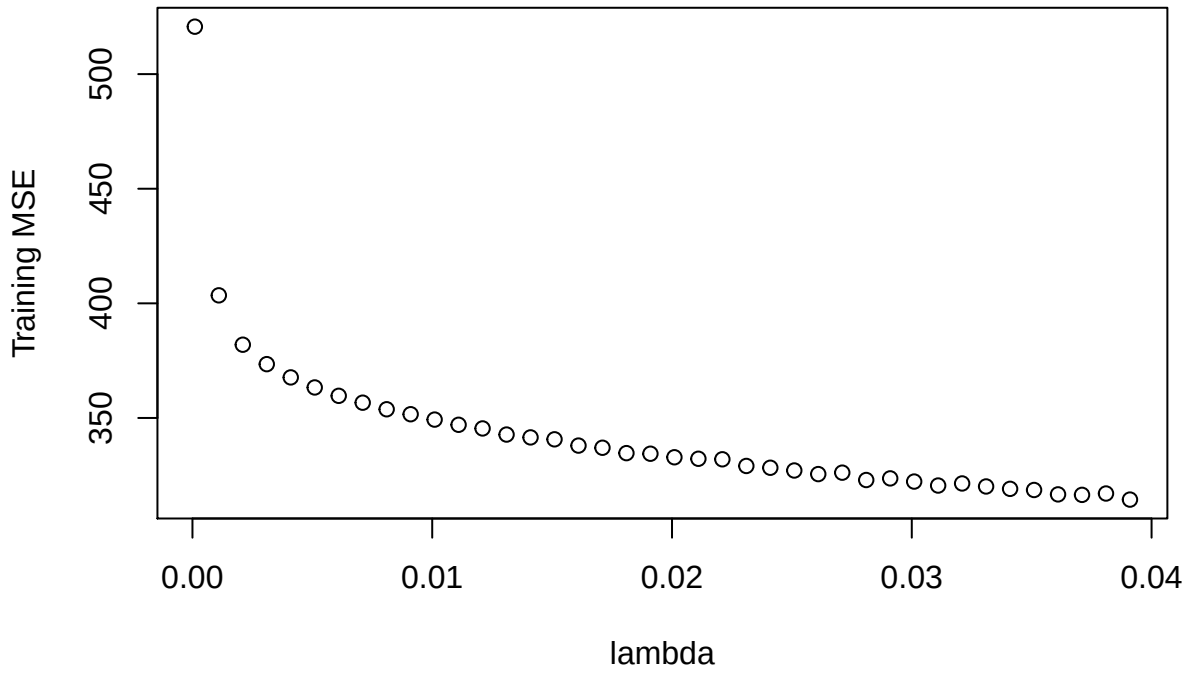
```
p <- dim(nes)[2]-1
lambda <- seq(from = .0001, to = .04, by =.001)
```

Question 2

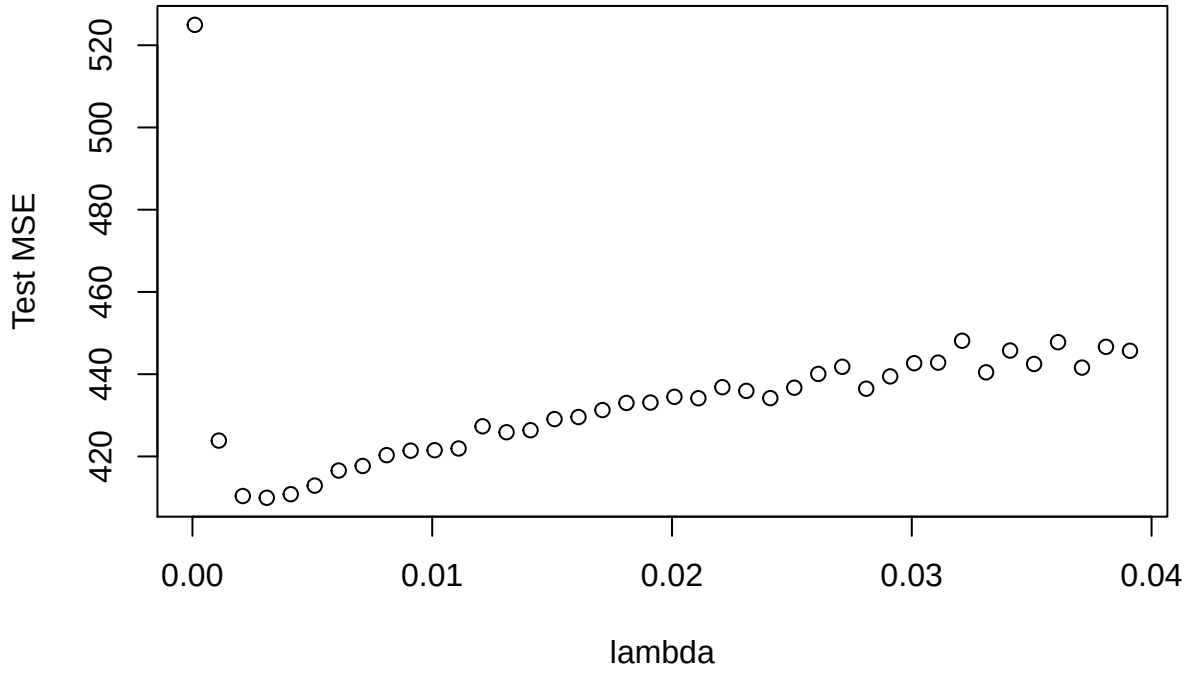
```
nes_split <- initial_split(data = nes, prop = 0.75)
nes_train <- training(nes_split)
nes_test <- testing(nes_split)
```

Question 3

```
MSE_train = c()
MSE_test = c()

for (i in lambda) {

boost.biden = gbm(biden ~ ., data = nes_train, distribution = "gaussian", n.trees=1000, shrinkage=i, in

PREDICT_train = predict(boost.biden, newdata = nes_train, n.trees=1000)
PREDICT_test = predict(boost.biden, newdata= nes_test, n.trees=1000)
MSE_train = append(MSE_train, MSE(y_pred = PREDICT_train, y_true = nes_train$biden))
MSE_test = append(MSE_test, MSE(y_pred = PREDICT_test, y_true = nes_test$biden))
}

plot(lambda, MSE_train, ylab="Training MSE", xlab="lambda")
```




```
plot(lambda, MSE_test, ylab="Test MSE", xlab="lambda")
```



Question 4

The test MSE here is 422.88. This looks very consistent with the MSE in the simulation from Q3 when looking at $\lambda = .01$ on the x-axis. In general from that graph, we can see that MSE ranges from 410 to 450 (if you disinclude the one outlier at 525) with an average of 434. So all in all, it does seem like the MSE doesn't vary much with our choice of λ, as long as it is small enough.

```
boost.biden = gbm(biden ~ ., data = nes_train, distribution = "gaussian", n.trees=1000, shrinkage=.01, i
Q4_predictions = predict(boost.biden, newdata= nes_test, n.trees=1000)
Q4_mse = MSE(y_pred = Q4_predictions, y_true = nes_test$biden)
Q4_mse
```

```
## [1] 422.1796
```

```
mean(MSE_test)
```

```
## [1] 434.0489
```

Question 5

The following code works on my computer and produced a bagging MSE of 455.87 However, for some reason it's not knitting with my document so I've included the code as a "comment" so the pdf at least renders.

```
# q5_bagging <- bagging(formula = biden ~ ., data = nes_train, nbagg=100, coob=TRUE, control = rpart.co
# Q5_predictions = predict(q5_bagging, newdata= nes_test)
# Q5_mse = MSE(y_pred = Q5_predictions, y_true = nes_test$biden)
# Q5_mse
```

Question 6

The test set MSE for this approach is 417.05

```
q6_rf <- randomForest(biden ~ ., data = nes_train)
q6_mse = MSE(y_pred = predict(q6_rf, newdata=nes_test), y_true = nes_test$biden)
q6_mse
```

```
## [1] 417.0507
```

Question 7

The test set MSE for this approach is 414.82

```
q7_reg <- lm(biden ~ ., data = nes_train)
q7_mse = MSE(y_pred = predict(q7_reg, newdata=nes_test), y_true = nes_test$biden)
q7_mse
```

```
## [1] 414.8206
```

Question 8

The lowest MSE happened to be the regression (414.82) in mine, followed by Random Forest (417.05). Boosting (422.88) was next, and the highest was Bagging (455.87). However, I notice that this might be a function of the seed I chose, and am hesitant to generalize this since conceptually we expect Random Forest and boosting to be better than linear regression.

Support Vector Machines

Question 1

```
set.seed(42)
oj_sample=sample(1:nrow(OJ),800)
oj_train <- OJ[oj_sample, ]
oj_test <- OJ[-oj_sample, ]
```

Question 2

Out of the 800 training observations, 439 were classified as support vectors (roughly equally for the 2 groups). This is relatively high, which means a lot of the observations might be borderline cases.

```
oj_svm <- svm(Purchase ~ ., data = oj_train, kernel = "linear", cost = 0.01, scale = TRUE)
summary(oj_svm)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj_train, kernel = "linear",
##     cost = 0.01, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  439
##
##  ( 219 220 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Question 3

The matrices are reported below, and as we can see there is a 16.5% error rate in the training classification and only a 6.25% error in the test classification

```
q3_train=table(pred = predict(oj_svm, newdata = oj_train), true = oj_train$Purchase)
q3_train
```

```
##     true
## pred  CH  MM
##   CH 428  78
##   MM  54 240
```

```
error_train = (q3_train[1,2]+q3_train[2,1])/800
error_train
```

```
## [1] 0.165
```

```
q3_test=table(pred = predict(oj_svm, newdata = oj_test), true = oj_test$Purchase)
q3_test
```

```
##     true
## pred  CH  MM
##   CH 150  29
##   MM  21  70
```

```
error_test = (q3_test[1,2]+q3_test[2,1])/800
error_test
```

```
## [1] 0.0625
```

Question 4

The optimal cost is 1000, according to my code below for the values that I tried for cost. It had 334 support vectors total.

```
q4_tuning <- tune(svm, Purchase ~ ., data = oj_train, kernel = "linear", ranges = list(cost = c(0.01, 0
summary(q4_tuning)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.17125
##
## - Detailed performance results:
##    cost   error dispersion
## 1 1e-02 0.17375 0.04910660
## 2 1e-01 0.17375 0.05219155
## 3 1e+00 0.17125 0.05172376
## 4 5e+00 0.17375 0.05118390
## 5 1e+01 0.17500 0.05137012
## 6 1e+02 0.17500 0.04823265
## 7 1e+03 0.18250 0.04533824
```

```
summary(q4_tuning$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = oj_train,
##     ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100, 1000)),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  1
##
## Number of Support Vectors:  335
##
##  ( 167 168 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Question 5

After replacing the cost with the optimal cost, the error rate in the training has decreased to 15.625% (as opposed to 16.5%) and that of the test set has decreased to 5.75%, down from 6.25%. Overall, this was a pretty good performance on the test set since the error is typical of what we would expect for a type 1 or 2 error in classical statistical theory. Additionally, optimizing the cost did lower the error rates, although not substantially (less than 1% improvement of error rate in both cases).

```
oj_svm <- svm(Purchase ~ ., data = oj_train, kernel = "linear", cost = 1000, scale = TRUE)

q5_train=table(pred = predict(oj_svm, newdata = oj_train), true = oj_train$Purchase)
q5_train
```

```
##     true
## pred  CH  MM
##   CH 427  70
##   MM  55 248
```

```
error_train = (q5_train[1,2]+q5_train[2,1])/800
error_train
```

```
## [1] 0.15625
```

```
q5_test=table(pred = predict(oj_svm, newdata = oj_test), true = oj_test$Purchase)
q5_test
```

```
##     true
## pred  CH  MM
##   CH 149  24
##   MM  22  75
```

```
error_test = (q5_test[1,2]+q5_test[2,1])/800
error_test
```

```
## [1] 0.0575
```