MAT 443 - HW8

Eric Agyemang

# Question 5

p= **c**(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)

There are two commmon ways to combine these results together into a single class prediction.One is amajority approach and the second is average approach.

# Majority Approach

**sum**(p **>=** 0.5) **> sum**(p **<** 0.5)

## [1] TRUE

The number of red predictions is greater than the number of green predictions based on a 50% threshold,thus

RED

# Average Approach

**mean**(p)

## [1] 0.45

The average of the probabilities is less than the 50% threshol,thus GREEN

# Question 8

1. splitting the data set into training and test set.

|  |
| --- |
| **library**(ISLR) **set.seed**(1)  train <-**sample**(1**:nrow**(Carseats), **nrow**(Carseats) **/** 2)  Carseats.train = Carseats[train, ]  Carseats.test = Carseats |

1. we now fit a regression tree to the training set.

**library**(tree) tree.carseats = **tree**(Sales**~**., data = Carseats.train) **summary**(tree.carseats)

##

## Regression tree:

## tree(formula = Sales ~ ., data = Carseats.train) ## Variables actually used in tree construction:

## [1] "ShelveLoc" "Price" "Age" "Advertising" "Income"

## [6] "CompPrice"

## Number of terminal nodes: 18 ## Residual mean deviance: 2.36 = 429.5 / 182 ## Distribution of residuals:

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## -4.2570 -1.0360 0.1024 0.0000 0.9301 3.9130

ploting the tree

**plot**(tree.carseats) **text**(tree.carseats, pretty= 0)

|

ShelveLoc: Bad,Medium

Price

< 120.5

< 50.5

Age

Price < 104.5

ShelveLoc: Bad

ertis

ing < 3.

5

Adv

e < 92

Pric

e < 85

Incom

:

Loc

Shelve

Bad

Co

m

pPr

ice < 107

Age

< 66.5

CompPr

ice < 148

10.5

Advertisi

ng

<

< 132

Price

Price

< 113

Price < 142.5

CompPrice < 133

8.284

10.400

6.922

9.117

7.512

9.882

5.059

5.206

7.131

4.627

6.418

7.031

4.780

2.249

12.080

8.778

11.010

6.973

Predicting error rate

pred.carseats = **predict**(tree.carseats, newdata = Carseats.test) **mean**((pred.carseats **-** Carseats.test**$**Sales)**^**2)

## [1] 3.14813 we may conclude that the Test MSE is about 4.15

c. we use cross-validation to determine the optimal level of the tree complexity.

cv.carseats<-**cv.tree**(tree.carseats) **plot**(cv.carseats**$**size, cv.carseats**$**dev, type = "b") tree.min<- **which.min**(cv.carseats**$**dev) **points**(tree.min, cv.carseats**$**dev[tree.min], col = "red",cex = 2, pch = 20)

5

10

15

1100

1300

1500

cv.carseats$dev

# cv.carseats$size

In this Particular case,the tree of size 8 is selected by cross-validation.We now prune the tree to obtain the 8-node tree.

prune.carseats= **prune.tree**(tree.carseats, best =8) **plot**(prune.carseats) **text**(prune.carseats,pretty = 0)

|

ShelveLoc: Bad,Medium

Price

< 120.5

Age

< 50.5

Pric

e < 92

Shelve

Loc: Bad

Age

< 66.5

Price < 113

9.278

8.628

5.059

6.820

5.654

3.303

12.080

8.792

yhat <- **predict**(prune.carseats, newdata = Carseats.test) **mean**((yhat **-** Carseats.test**$**Sales)**^**2)

## [1] 4.159743

We now see that prunning the tree will increase the test MSE to 4.2

d. using the bagging approach in orser to analyze the data set.

**library**(randomForest)

## randomForest 4.6-14

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

|  |
| --- |
| bag.carseats= **randomForest**(Sales**~**., data = Carseats.train,mtry=10, ntree = 500, importance= yhat.bag= **predict**(bag.carseats, newdata = Carseats.test) **mean**((yhat.bag **-** Carseats.test**$**Sales)**^**2) |

## TRUE)

## [1] 1.561924

**importance**(bag.carseats)

## %IncMSE IncNodePurity

|  |  |
| --- | --- |
| ## CompPrice 16.9874366 | 126.852848 |
| ## Income 3.8985402 | 78.314126 |
| ## Advertising 16.5698586 | 123.702901 |
| ## Population 0.6487058 | 62.328851 |
| ## Price 55.3976775 | 514.654890 |
| ## ShelveLoc 42.7849818 | 319.133777 |
| ## Age 20.5135255 | 185.582077 |
| ## Education 3.4615211 | 42.253410 |
| ## Urban -2.5125087 | 8.700009 |
| ## US 7.3586645 | 18.180651 |

We see that bagging decreases the Test MSE to 2.6.We also see that Price,ShelvecLoc and Age are the three most important predictors of sale

e. Using random forests to analyze the dataset.

|  |
| --- |
| rf.carseats= **randomForest**(Sales**~**., data = Carseats.train, mtry=3, ntree = 500, importance = yhat.rf= **predict**(rf.carseats, newdata = Carseats.test) **mean**((yhat.rf **-** Carseats.test**$**Sales)**^**2) |

## TRUE)

## [1] 1.995229

We obtain our m = square root of p, we have a test MSE OF 1.97

**importance**(rf.carseats)

## %IncMSE IncNodePurity

|  |  |
| --- | --- |
| ## CompPrice 7.443405 | 130.87552 |
| ## Income 3.227858 | 127.18662 |
| ## Advertising 13.388259 | 139.53499 |
| ## Population -1.031306 | 102.32154 |
| ## Price 36.616911 | 369.59534 |
| ## ShelveLoc 31.284175 | 233.49549 |
| ## Age 17.622273 | 206.09959 |
| ## Education 1.454555 | 70.41374 |
| ## Urban -1.864781 | 15.13225 |
| ## US 6.193082 | 35.74746 |

Again we conclude that,in this scenario, Price and ShelveLoc are the two most important variables.

## **Question 10**

We now use the boosting to predict salary in the hitters data set.

1. Removing the observations for whom the salary information and the log transform the salaries

Hitters <- **na.omit**(Hitters)

Hitters**$**Salary <- **log**(Hitters**$**Salary)

1. Creating training set of first 200 obseravtions and test set consisting of the remaining observations.

train = 1**:**200

Hitters.train = Hitters[train, ]

Hitters.test = Hitters[**-**train, ]

1. Now we perform boosting model on the training set

**library**(gbm)

## Loaded gbm 2.1.4

|  |
| --- |
| **set.seed**(1) pows <- **seq**(**-**10, **-**0.2, by = 0.1) lambdas <- 10**^**pows train.err <- **rep**(NA, **length**(lambdas)) **for** (i **in** 1**:length**(lambdas)) { boost.hitters <- **gbm**(Salary **~** ., data = Hitters.train, distribution = "gaussian", pred.train <- **predict**(boost.hitters, Hitters.train, n.trees = 1000) train.err[i] <- **mean**((pred.train **-** Hitters.train**$**Salary)**^**2)  } **plot**(lambdas, train.err, type = "b", xlab = "Shrinkage values", ylab = "Training MSE") |

n.trees = 1000, sh

0.0

0.1

0.2

0.3

0.4

0.5

0.6

0.0

0.2

0.4

0.6

0.8

Training MSE

# Shrinkage values

1. Plotting with diffeent type of shrinkage values on x-axis and corresponding test set MSE on the y-axis

|  |
| --- |
| **set.seed**(1) test.err <- **rep**(NA, **length**(lambdas)) **for** (i **in** 1**:length**(lambdas)) { boost.hitters <- **gbm**(Salary **~** ., data = Hitters.train, distribution = "gaussian", shrinkage = lambdas[i])  yhat <- **predict**(boost.hitters, Hitters.test, n.trees = 1000) |

n.trees = 1000,

|  |
| --- |
| test.err[i] <- **mean**((yhat **-** Hitters.test**$**Salary)**^**2)  } **plot**(lambdas, test.err, type = "b", xlab = "Shrinkage values", ylab = "Test MSE") |

0.0

0.1

0.2

0.3

0.4

0.5

0.6

0.3

0.4

0.5

0.6

Test MSE

Shrinkage values

**min**(test.err)

## [1] 0.2540265

lambdas[**which.min**(test.err)]

## [1] 0.07943282

We obtain the minimum test MSE to be 0.25 and Lambda = 0.079

1. Comparison of the test MSE of boosting

**library**(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

fit1 <- **lm**(Salary **~** ., data = Hitters.train) pred1 <- **predict**(fit1, Hitters.test) **mean**((pred1 **-** Hitters.test**$**Salary)**^**2)

## [1] 0.4917959

x <- **model.matrix**(Salary **~** ., data = Hitters.train)

x.test <- **model.matrix**(Salary **~** ., data = Hitters.test) y <- Hitters.train**$**Salary fit2 <- **glmnet**(x, y, alpha = 0) pred2 <- **predict**(fit2, s = 0.01, newx = x.test) **mean**((pred2 **-** Hitters.test**$**Salary)**^**2)

## [1] 0.4570283

We can see in this case the test MSE for boosting is lower than for linear regression and ridge regression.

f. Selecting variables that appear to be the most important predictors in the boosted model.

**library**(gbm)

|  |
| --- |
| boost.hitters <- **gbm**(Salary **~** ., data = Hitters.train, distribution = "gaussian", n.trees = **summary**(boost.hitters) |

1000, shrink

League

HmRun

CHits

Walks

0

5

10

15

20

# Relative influence

|  |  |
| --- | --- |
| ## | var rel.inf |
| ## CAtBat | CAtBat 20.8404970 |
| ## CRBI | CRBI 12.3158959 |
| ## Walks | Walks 7.4186037 |
| ## PutOuts | PutOuts 7.1958539 |
| ## Years | Years 6.3104535 |
| ## CWalks | CWalks 6.0221656 |
| ## CHmRun | CHmRun 5.7759763 |
| ## CHits | CHits 4.8914360 |
| ## AtBat | AtBat 4.2187460 |
| ## RBI | RBI 4.0812410 |
| ## Hits | Hits 4.0117255 |
| ## Assists | Assists 3.8786634 |
| ## HmRun | HmRun 3.6386178 |
| ## CRuns | CRuns 3.3230296 |
| ## Errors | Errors 2.6369128 |
| ## Runs | Runs 2.2048386 |
| ## Division | Division 0.5347342 |

## NewLeague NewLeague 0.4943540

## League League 0.2062551

g.

**set.seed**(1) bag.hitters <- **randomForest**(Salary **~** ., data = Hitters.train, mtry = 19, ntree = 500) yhat.bag <- **predict**(bag.hitters, newdata = Hitters.test) **mean**((yhat.bag **-** Hitters.test**$**Salary)**^**2)

## [1] 0.2299324

Now we conclude that, the test MSE for bagging is 0.23, which is slightly lower than the test MSE for boosting.