2020111142_谢嘉薪_Ass4

Xie Jiaxin, 2020111142

练习 & 第四次作业

1. This problem involves the `OJ` data set which is part of the `ISLR2` package.

```
library(ISLR2)
library(tree)
library(tidyverse)
data(OJ,package = 'ISLR2')
head(OJ)
```

	Purcha… <fct>	WeekofPurchase <dbl>	StoreID <dbl>	Price… <dbl>	Price… <dbl>	Disc… <dbl>	Disc… <dbl>	SpecialCH <dbl>	Special… <dbl>
1	CH	237	1	1.75	1.99	0.00	0.0	0	0
2	CH	239	1	1.75	1.99	0.00	0.3	0	1
3	CH	245	1	1.86	2.09	0.17	0.0	0	0
4	MM	227	1	1.69	1.69	0.00	0.0	0	0
5	CH	228	7	1.69	1.69	0.00	0.0	0	0
6	CH	230	7	1.69	1.99	0.00	0.0	0	1

6 rows | 1-10 of 19 columns

a. Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
set.seed(2020111142)
train <- sample(1:nrow(OJ), 800)
OJ.train <- OJ[train,]
OJ.test <- OJ[-train,]
```

b. Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
set.seed(2020111142)
tree.OJ <- tree(Purchase~., OJ.train)
summary(tree.OJ)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "SalePriceMM"   "ListPriceDiff" "PriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7801 = 618.7 / 793
## Misclassification error rate: 0.1575 = 126 / 800
```

> 树中用作内部节点的变量为：“LoyalCH” “SalePriceMM” “ListPriceDiff” “PriceDiff”

> 叶节点的数量为：7

> 训练误差为：0.1575

c. Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

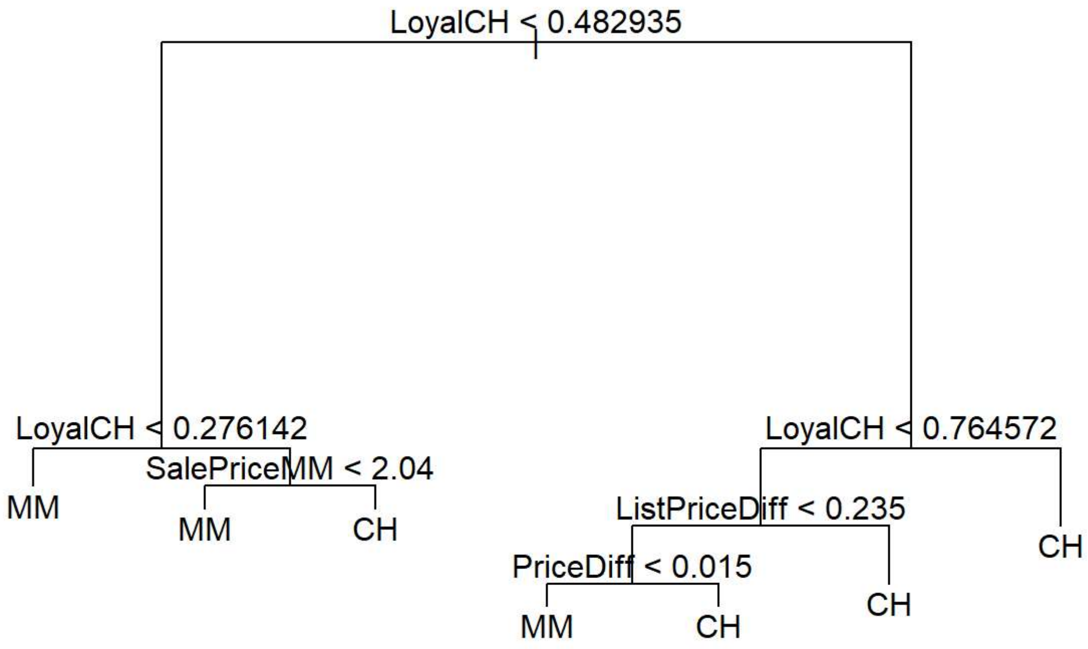
```
tree.OJ
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1066.00 CH ( 0.61500 0.38500 )
##    2) LoyalCH < 0.482935 302  336.30 MM ( 0.24503 0.75497 )
##      4) LoyalCH < 0.276142 160  120.60 MM ( 0.12500 0.87500 ) *
##      5) LoyalCH > 0.276142 142  188.60 MM ( 0.38028 0.61972 )
##       10) SalePriceMM < 2.04 75   80.28 MM ( 0.22667 0.77333 ) *
##       11) SalePriceMM > 2.04 67   92.15 CH ( 0.55224 0.44776 ) *
##    3) LoyalCH > 0.482935 498  439.00 CH ( 0.83936 0.16064 )
##      6) LoyalCH < 0.764572 240  286.10 CH ( 0.71667 0.28333 )
##       12) ListPriceDiff < 0.235 92  127.40 MM ( 0.47826 0.52174 )
##         24) PriceDiff < 0.015 51   61.79 MM ( 0.29412 0.70588 ) *
##         25) PriceDiff > 0.015 41   49.57 CH ( 0.70732 0.29268 ) *
##       13) ListPriceDiff > 0.235 148  117.20 CH ( 0.86486 0.13514 ) *
##      7) LoyalCH > 0.764572 258   97.07 CH ( 0.95349 0.04651 ) *
```

> 4. LoyalCH < 0.276142 160 120.60 MM (0.12500 0.87500) * 表示分类的准则为LoyalCH < 0.276142，节点中的观测值数量为160，偏差为120.6，节点的总体预测为MM，节点中观测值取MM的比例为0.125，取CH的比例为0.875

d. Create a plot of the tree, and interpret the results.nodes, and interpret the information displayed.

```
plot(tree.OJ)
text(tree.OJ, pretty = 0)
```



> 该图展示了树的结构，且每个节点上有其分类准则，每个叶节点对应的类别也展示出来了。同时可以看出LoyalCH是影响最大的变量，因为其节点所在位置靠近根部。

e. Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
tree.pred <- predict(tree.OJ,OJ.test,type = "class")
table(tree.pred, OJ.test$Purchase)
```

```
##
## tree.pred  CH  MM
##        CH 142  24
##        MM  19  85
```

```
error <- 1-mean(tree.pred == OJ.test$Purchase)
error
```

```
## [1] 0.1592593
```

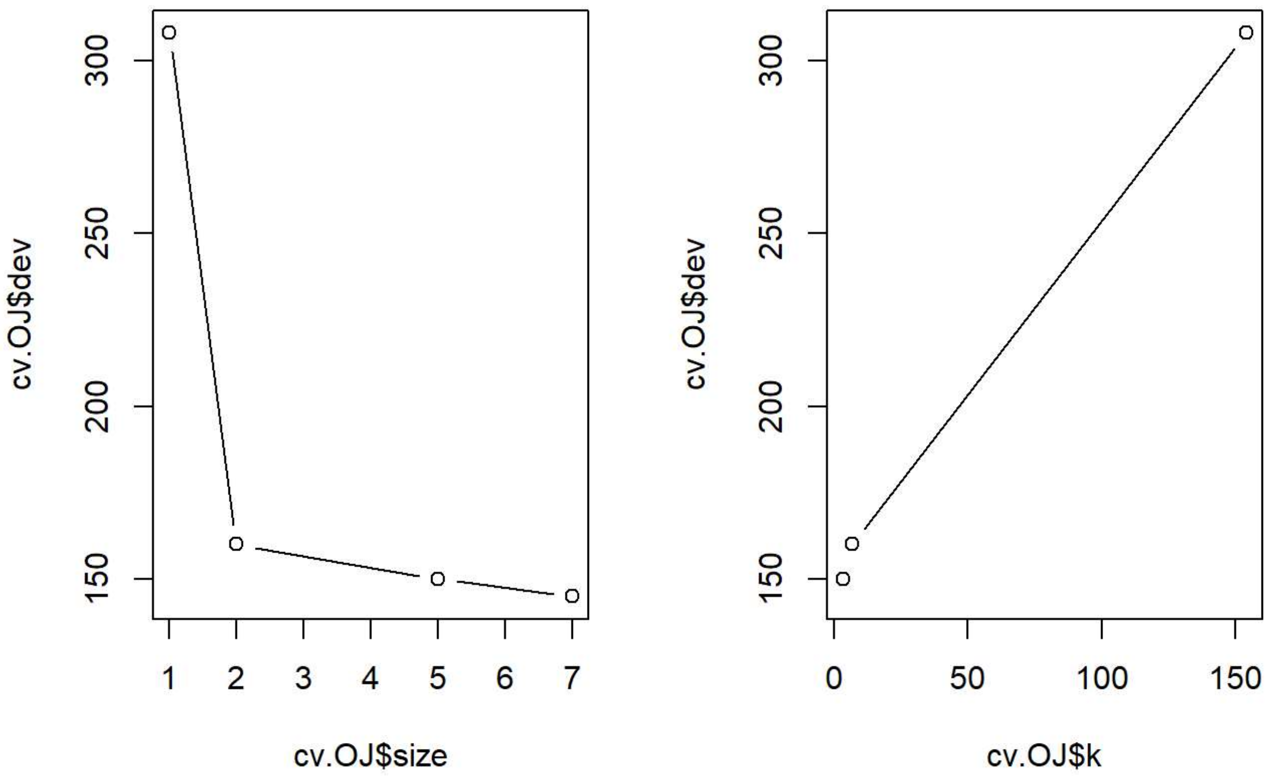
f. Apply the `cv.tree()` function to the training set in order to determine the optimal tree size.

```
set.seed(2020111142)
cv.OJ <- cv.tree(tree.OJ, FUN = prune.misclass)
cv.OJ
```

```
## $size
## [1] 7 5 2 1
##
## $dev
## [1] 145 150 160 308
##
## $k
## [1]  -Inf   3.5   7.0 154.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

g. Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

```
par(mfrow = c(1, 2))
plot(cv.OJ$size, cv.OJ$dev, type = "b")
plot(cv.OJ$k, cv.OJ$dev, type = "b")
```

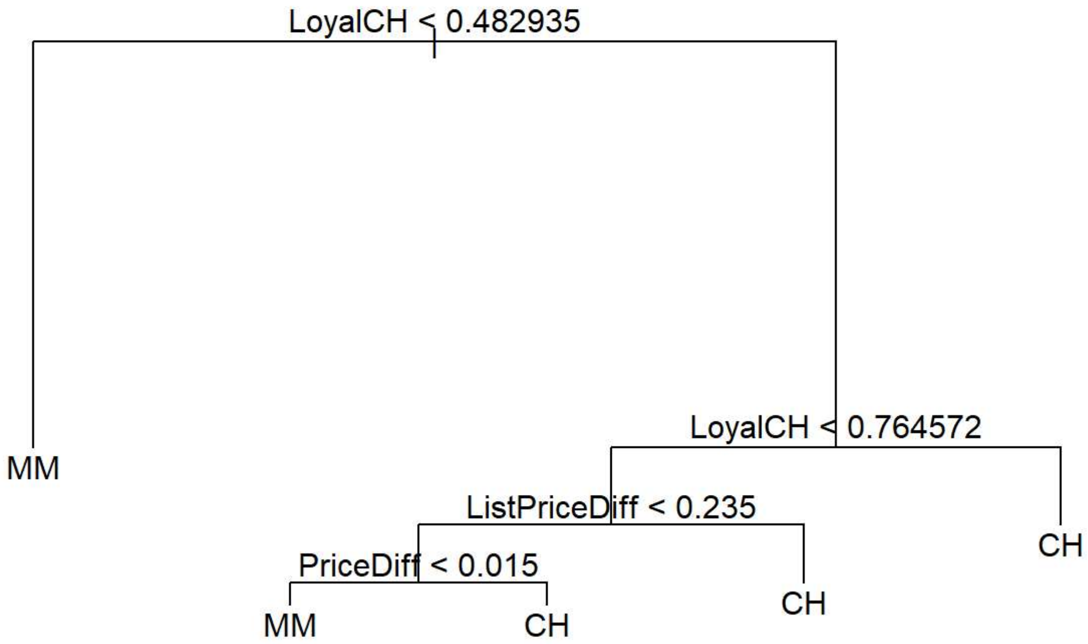


h. Which tree size corresponds to the lowest cross-validated classification error rate?

7

i. Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
prune.OJ <- prune.misclass(tree.OJ, best = 5)
plot(prune.OJ)
text(prune.OJ, pretty = 0)
```



j. Compare the training error rates between the pruned and unpruned trees. Which is higher?

```
summary(prune.OJ)
```

```
##
## Classification tree:
## snip.tree(tree = tree.OJ, nodes = 2L)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "ListPriceDiff" "PriceDiff"
## Number of terminal nodes:  5
## Residual mean deviance:  0.8327 = 662 / 795
## Misclassification error rate: 0.1662 = 133 / 800
```

pruned trees is higher

k. Compare the test error rates between the pruned and unpruned trees. Which is higher?

```
prune.tree.pred <- predict(prune.OJ,OJ.test,type = "class")
table(prune.tree.pred, OJ.test$Purchase)
```

```
##
## prune.tree.pred  CH  MM
##              CH 135  18
##              MM  26  91
```

```
error.prune <- 1-mean(prune.tree.pred == OJ.test$Purchase)
error.prune
```

```
## [1] 0.162963
```

> pruned trees is higher

2. We now use boosting to predict `Salary` in the `Hitters` data set.

```
data("Hitters")
head(Hitters$Salary)
```

```
## [1]    NA 475.0 480.0 500.0  91.5 750.0
```

a. Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
sum(is.na(Hitters))
```

```
## [1] 59
```

```
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```
Hitters <- Hitters %>%
  na.omit(Hitters) %>%
  mutate(Salary = log(Salary))
head(Hitters$Salary)
```

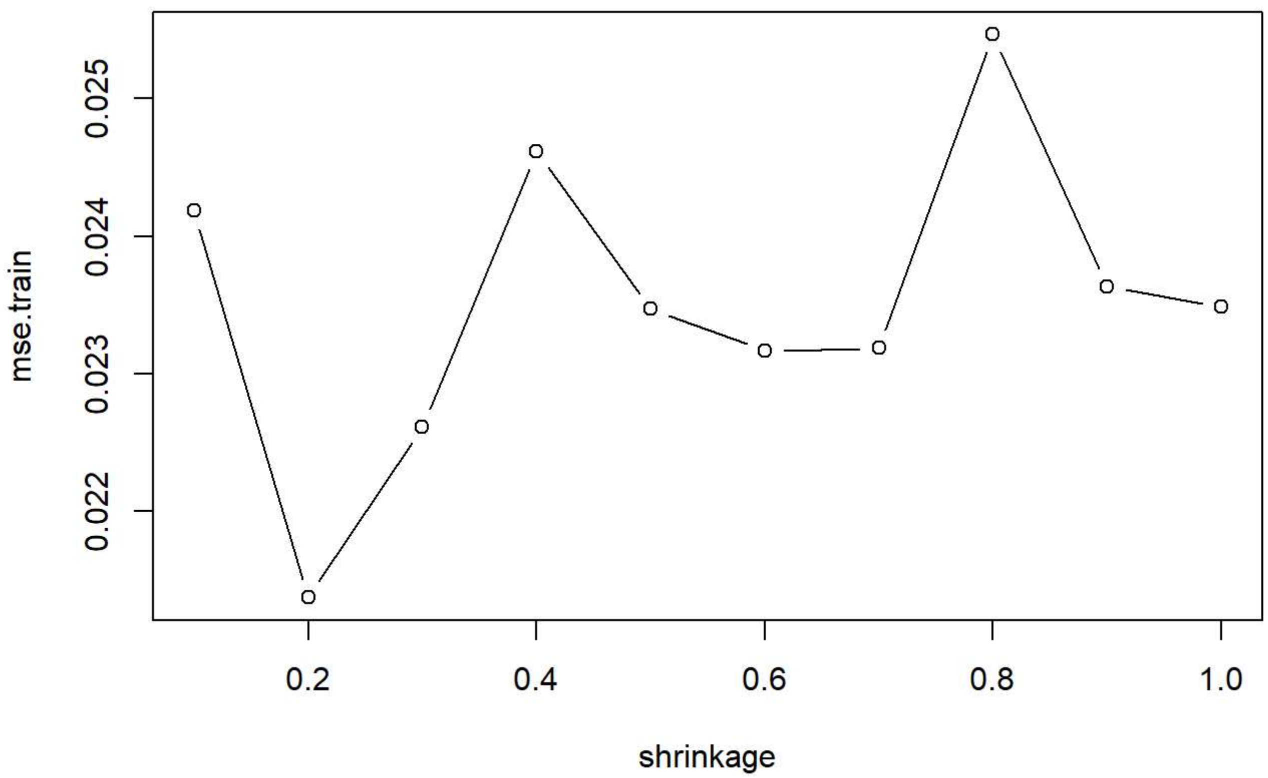
```
## [1] 6.163315 6.173786 6.214608 4.516339 6.620073 4.248495
```

b. Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
Hitters.train <- Hitters[1:200,]
Hitters.test <- Hitters[-(1:200),]
```

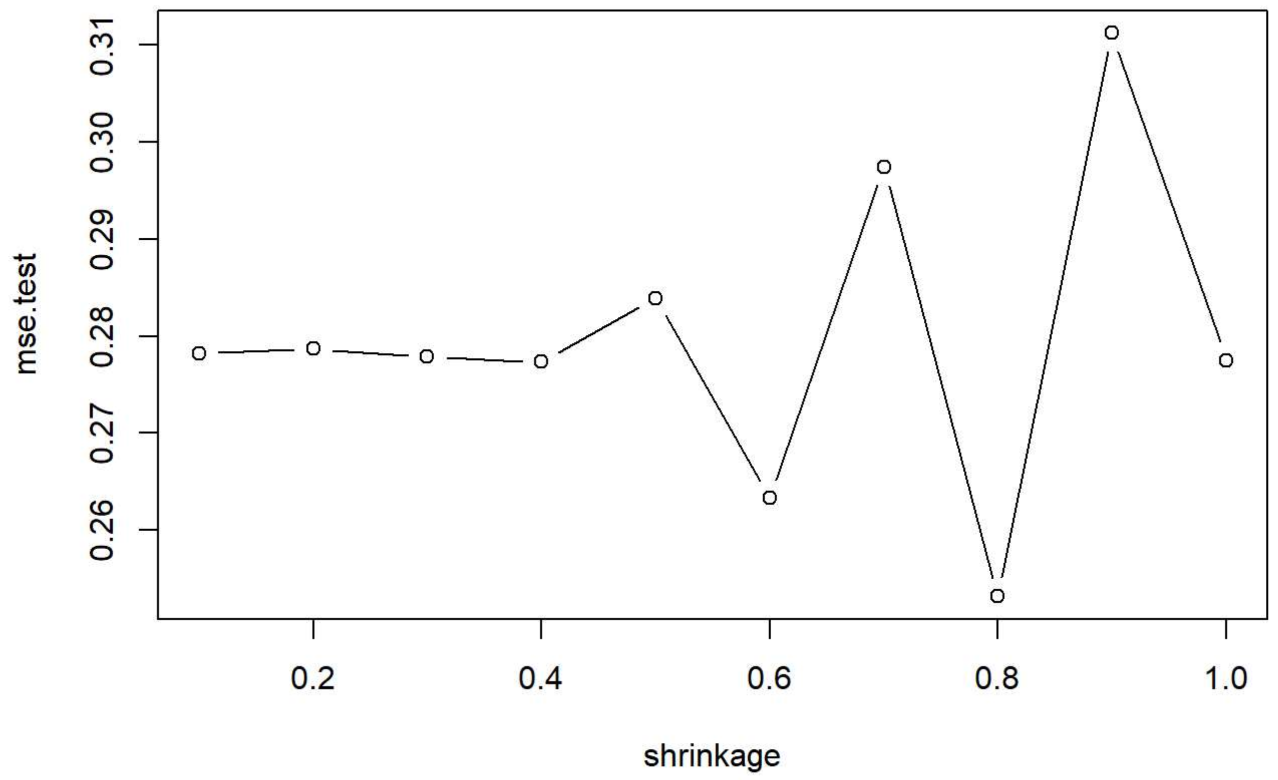
c. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

```
library(gbm)
set.seed(2020111142)

mse.train <- 0
mse.test <- 0
shrinkage <- seq(0.1,1,by=0.1)

for (i in 1:length(shrinkage)) {
  boost.Hitters <- gbm(Salary~., data = Hitters.train, distribution = "gaussian", n.trees = 100
0, shrinkage = 0.2)
  train.pred <- predict(boost.Hitters,data = Hitters.train)
  test.pred <- predict(boost.Hitters,newdata = Hitters.test)
  mse.train[i] <- mean((train.pred - Hitters.train$Salary)^2)
  mse.test[i] <- mean((test.pred - Hitters.test$Salary)^2)
}

plot(shrinkage,mse.train,type = "b")
```



d. Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```
plot(shrinkage,mse.test,type = "b")
```



e. Compare the test MSE of boosting to the test MSE that results from applying linear regression and LASSO.

```
library(glmnet)
lm.Hitters <- lm(Salary~., data = Hitters.train)
lm.pred <- predict(lm.Hitters,newdata = Hitters.test)
mse.lm <- mean((lm.pred - Hitters.test$Salary)^2)
mse.lm
```

```
## [1] 0.4917959
```

```
x <- model.matrix(Salary~., data = Hitters.train)[,-1]
x.test <- model.matrix(Salary~., data = Hitters.test)[,-1]
lasso.Hitters <- glmnet(x,y=Hitters.train$Salary,alpha = 1)
lasso.pred <- predict(lasso.Hitters,newx = x.test)
mse.lasso <- mean((lasso.pred - Hitters.test$Salary)^2)
mse.lasso
```

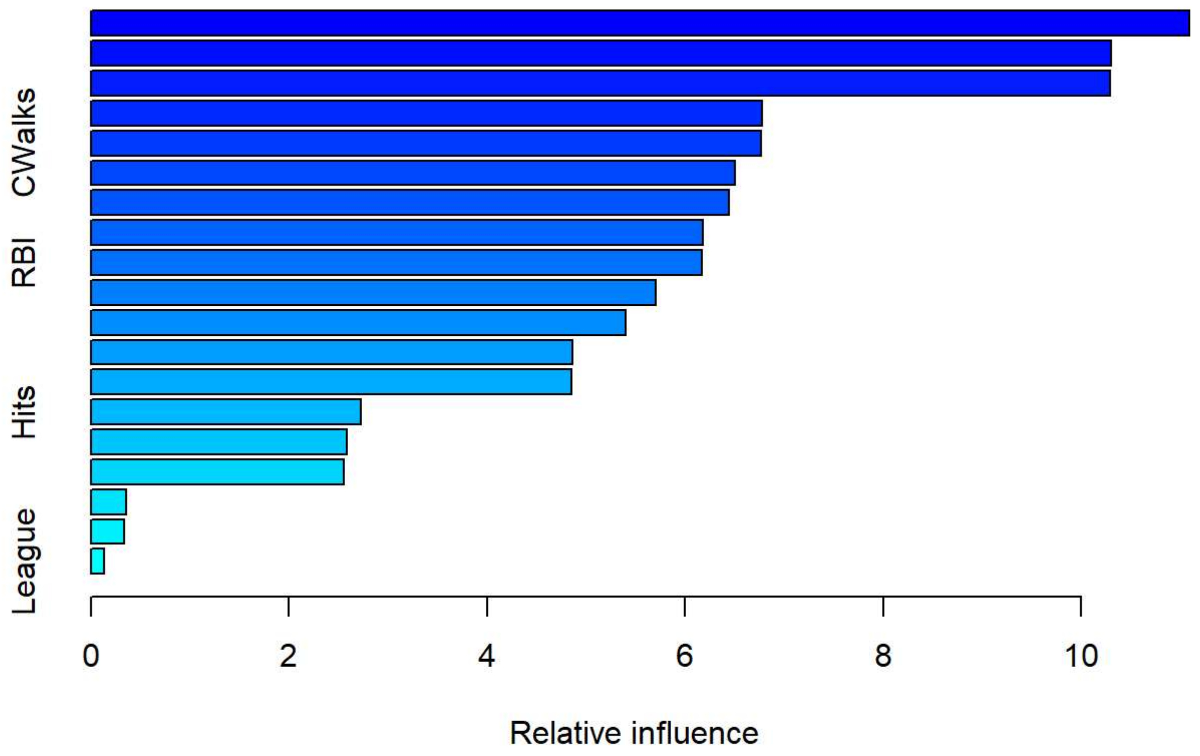
```
## [1] 0.4755605
```

```
best.mse <- min(mse.test,mse.lm,mse.lasso)
best.mse
```

```
## [1] 0.2531439
```

f. Which variables appear to be the most important predictors in the boosted model?

```
summary(boost.Hitters)
```



	var	rel.inf
	<chr>	<dbl>
CAtBat	CAtBat	11.0879773
PutOuts	PutOuts	10.2955331
CRuns	CRuns	10.2910112
CRBI	CRBI	6.7771754
CWalks	CWalks	6.7683361
Assists	Assists	6.4992090
Walks	Walks	6.4376112
CHits	CHits	6.1739339
RBI	RBI	6.1707069
CHmRun	CHmRun	5.7058854

1-10 of 19 rows Previous **1** 2 Next

g. Now apply bagging to the training set. What is the test set MSE for this approach?

```
library(randomForest)
set.seed(2020111142)
bag.Hitters <- randomForest(Salary~., data = Hitters.train, mtry = 19, ntree = 1000)
bag.pred <- predict(bag.Hitters, Hitters.test)
mse.bag <- mean((bag.pred - Hitters.test$Salary)^2)
mse.bag
```

```
## [1] 0.2269832
```