HW6

計財所 108071601 賴冠維

2020/12/22

```
## Warning: package 'magrittr' was built under R version 4.0.3
```

(8)

本題使用資料為$ISLR$裡的$Carseats$
有 400 observations 及 11 variables，變數解釋如下：
* Sales
Unit sales (in thousands) at each location
* CompPrice
Price charged by competitor at each location
* Income
Community income level (in thousands of dollars)
* Advertising
Local advertising budget for company at each location (in thousands of dollars)
* Population
Population size in region (in thousands)
* Price
Price company charges for car seats at each site
* ShelveLoc
A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site
* Age
Average age of the local population
* Education
Education level at each location
* Urban
A factor with levels No and Yes to indicate whether the store is in an urban or rural location
* US
A factor with levels No and Yes to indicate whether the store is in the US or not

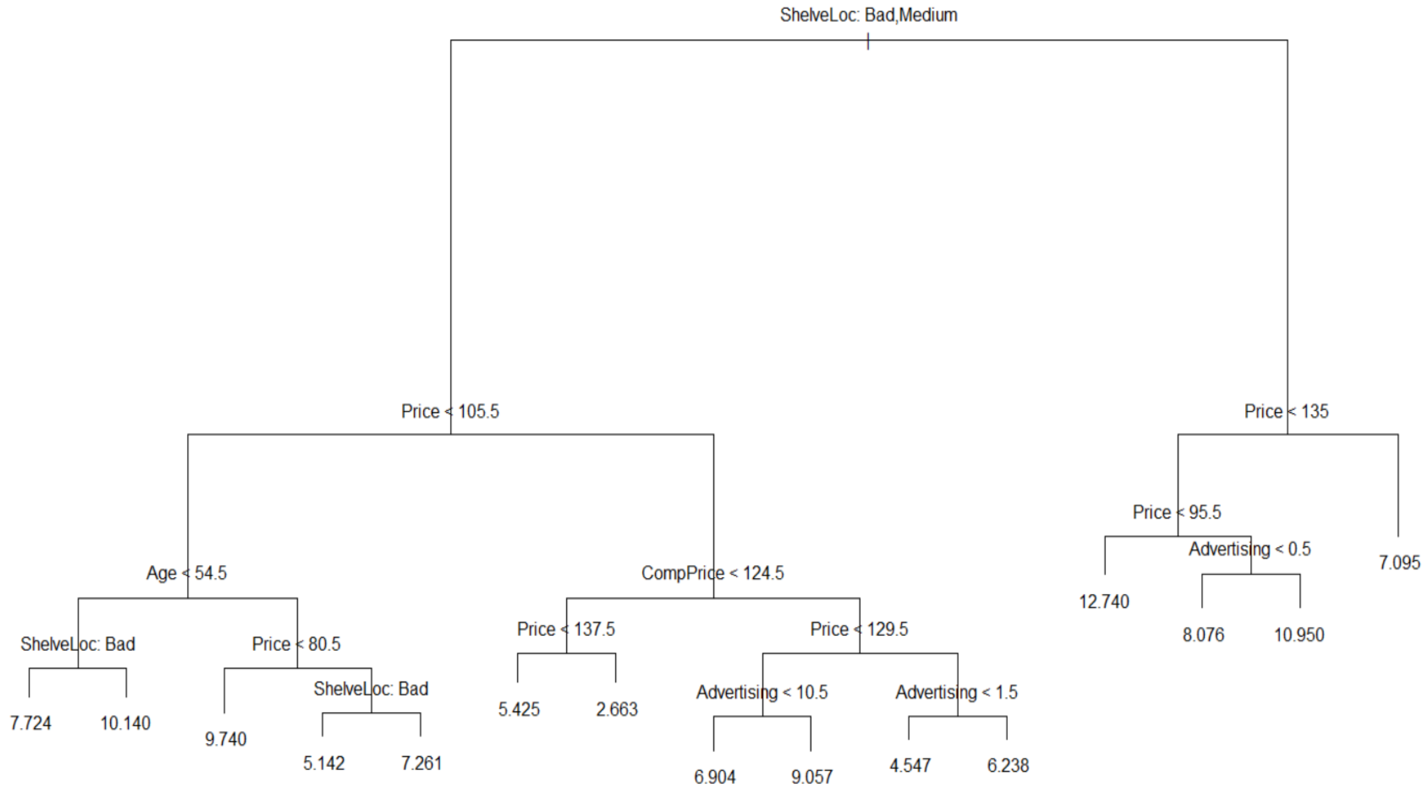
(a)

將$Carseats$以[80,20]比例分割成$Train, Test$

```
## Warning: package 'tree' was built under R version 4.0.3

## Train: 320 12 Test: 80 12
```

(b)

建立$Regression\ Tree$預測$Sales$，下圖為預測結果，節點左邊為Yes，右邊為No
模型裡僅用到$"ShelveLoc" "Price" "Age" "CompPrice" "Advertising"



一共有 15 個$terminal\ nodes$

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = train[, -12])
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "CompPrice"   "Advertising"
## Number of terminal nodes:  15
## Residual mean deviance:  2.641 = 805.4 / 305
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.54700 -1.08400 -0.09094  0.00000  1.17500  3.88500
```

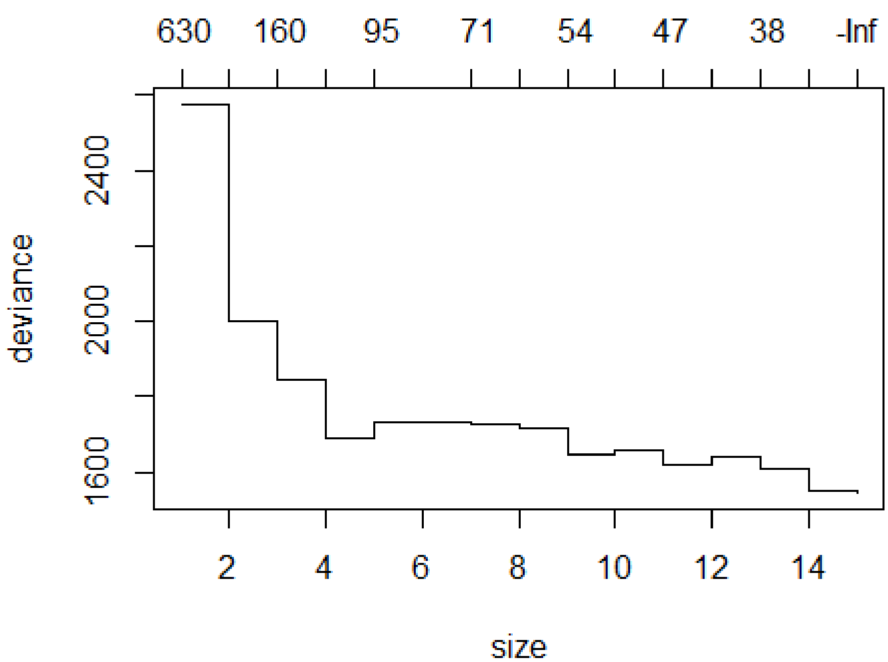
以$Train$建立迴歸樹後，導入$Test$資料，首先看到$Test\ RMSE$為 4.728284
將預測結果以是否大於 8 為界，分為$[Low, High]$
列出$ConfusionTable$，以及$Test\ Error\ Rate$為 0.2875

```
## [1] "Test Error MSE: 4.72828433855961"

##          actual
## predicted High Low
##      High   15   5
##      Low    18  42

## Test Error Rate: 0.2875
```

(c)

使用$Cross\ Validation$，選出具有最佳的預測力的樹複雜度
以$K = 10$，10 $Fold$進行$Cross\ Validation$，選到 15 個$Terminal\ Nodes$
由圖亦可看到誤差隨著$Size$增加而遞減至收斂
與上述模型相同，因此得到相同的$Test\ Error\ Rate$

```
## [1] "Best Size : 15"
```

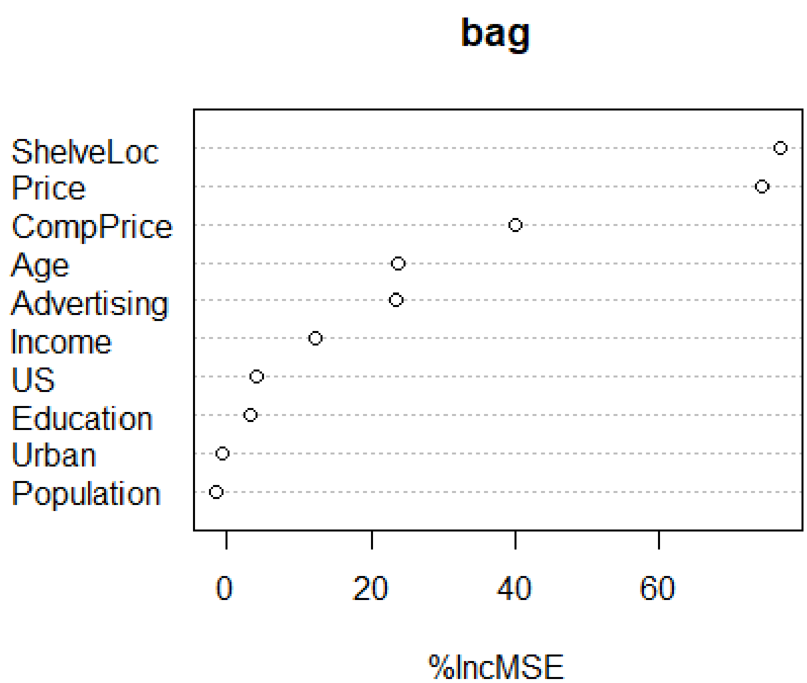



```
## [1] "Test Error MSE: 4.72828433855961"

##            actual
## predicted High Low
##      High   15   5
##      Low    18  42

## Test Error Rate: 0.2875
```

(d)

接下來以$Bagging$方式預測，$Bagging$與$Random Forest
僅差在建立子集(Subset)*時，是否也對*Columns*進行抽樣，因此以*random Forest
*套件進行即可以相同流程對*Test*估計*，RMSE*明顯較低並且從*Confusion
Table*及*Test Error Rate*亦可看出，預測表現較佳。由*Variance Importance Plot
*可看出個別變數對模型的貢獻度，發現*Price、SheleveLoc$這兩個變數解釋力為最佳。

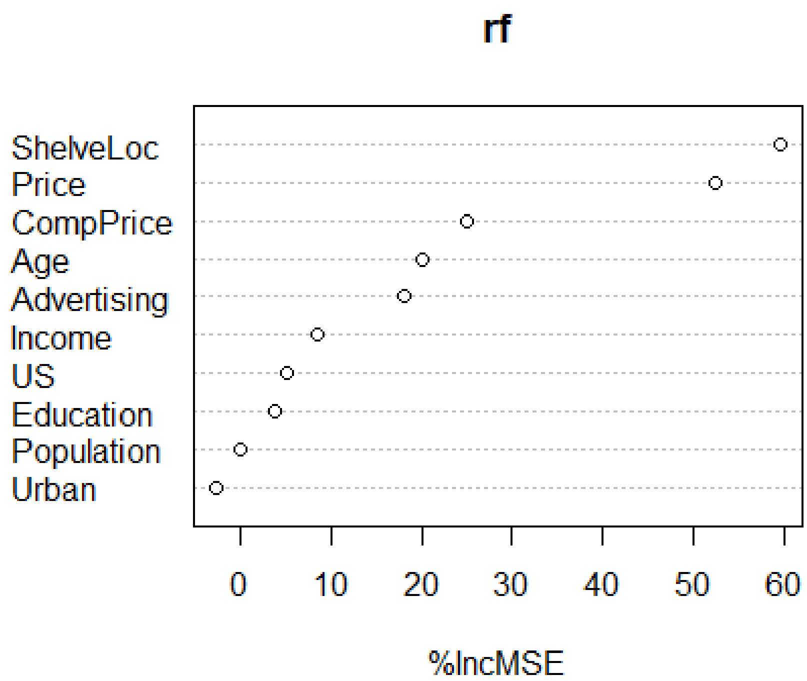
```
## Warning: package 'randomForest' was built under R version 4.0.3

## randomForest 4.6-14

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

## [1] "Test Error MSE: 2.4401977490189"

##           actual
## predicted High Low
##      High   22   4
##      Low    11  43

## Test Error Rate: 0.1875
```



(e)

接下來以$Random\ Forest$進行配適，選取$Column\ Split$以$p/3$為標準$= 4$
以相同步驟估計，最後可看到$Random\ Forest$表現比$Bagging$更佳。

```
## [1] "Test Error MSE: 2.32105249762642"

##           actual
## predicted High Low
##      High   23   4
##      Low    10  43

## Test Error Rate: 0.175
```



(9)

本題使用資料為*ISLR*裡的*OJ*
有 1070 observations 及 18 variables，變數解釋如下：
* Purchase
A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice
* WeekofPurchase
Week of purchase
* StoreID
Store ID
* PriceCH
Price charged for CH
* PriceMM
Price charged for MM
* DiscCH
Discount offered for CH
* DiscMM
Discount offered for MM
* SpecialCH
Indicator of special on CH
* SpecialMM
Indicator of special on MM
* LoyalCH
Customer brand loyalty for CH
* SalePriceMM
Sale price for MM
* SalePriceCH
Sale price for CH

* PriceDiff
Sale price of MM less sale price of CH
* Store7
A factor with levels No and Yes indicating whether the sale is at Store 7
* PctDiscMM
Percentage discount for MM
* PctDiscCH
Percentage discount for CH
* ListPriceDiff
List price of MM less list price of CH
* STORE
Which of 5 possible stores the sale occured at

(9)

(a)

把OJ資料，以前800筆為$Train$，剩下分為$Test$

```
## Train: 800 18 Test: 270 18
```

(b)

以$Purchase$為目標被解釋變數，其餘為解釋變數，建立類別樹
可看到共用到$“LoyalCH” “SalePriceMM” “ListPriceDiff” “PriceDiff” $
這幾個變數，最後的$Terminal\ Nodes$為 7，
$Misclassificationerrorrate$:為 0.1638

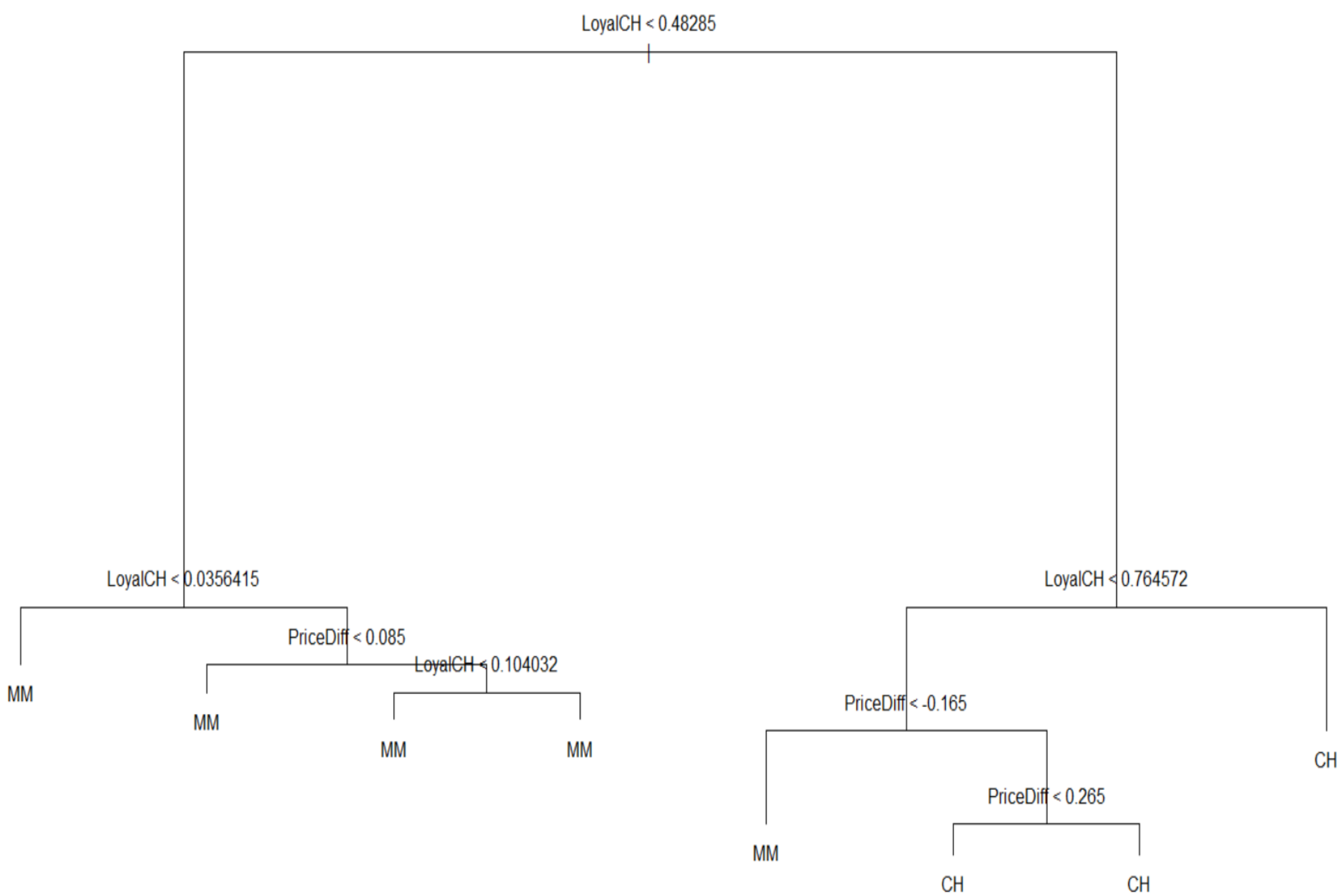
```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7916 = 626.9 / 792
## Misclassification error rate: 0.1762 = 141 / 800
```

以$Terminal\ Nodes$ (10)為例解釋，當$LoyalCH < 0.48285$，$LoyalCH > 0.0356415$，且$PriceDiff < 0.085$時，
在$TrainSet$裡共有 97 筆資料落在此$Nodes$，將此$Nodes$預測為MM，並且正確率為 0.82474

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1064.000 CH ( 0.61750 0.38250 )
##    2) LoyalCH < 0.48285 292  330.600 MM ( 0.25342 0.74658 )
##      4) LoyalCH < 0.0356415 54    9.959 MM ( 0.01852 0.98148 ) *
##      5) LoyalCH > 0.0356415 238  293.400 MM ( 0.30672 0.69328 )
##       10) PriceDiff < 0.085 97   90.040 MM ( 0.17526 0.82474 ) *
##       11) PriceDiff > 0.085 141  189.500 MM ( 0.39716 0.60284 )
##         22) LoyalCH < 0.104032 18    7.724 MM ( 0.05556 0.94444 ) *
##         23) LoyalCH > 0.104032 123  169.100 MM ( 0.44715 0.55285 ) *
##    3) LoyalCH > 0.48285 508  468.300 CH ( 0.82677 0.17323 )
##      6) LoyalCH < 0.764572 245  300.200 CH ( 0.69796 0.30204 )
##       12) PriceDiff < -0.165 33   31.290 MM ( 0.18182 0.81818 ) *
##       13) PriceDiff > -0.165 212  224.300 CH ( 0.77830 0.22170 )
##         26) PriceDiff < 0.265 116  145.300 CH ( 0.68103 0.31897 ) *
##         27) PriceDiff > 0.265 96   64.160 CH ( 0.89583 0.10417 ) *
##      7) LoyalCH > 0.764572 263  109.400 CH ( 0.94677 0.05323 ) *
```

(d)

由圖亦可看到上述結果



(e)

由$ConfusionTable$及$Test\ Error\ Rate$看出對 Test 的估計結果，
$Test\ Error\ Rate$為 0.1518519

```
##           actual
## predicted  CH  MM
##        CH 133  15
##        MM  26  96

## Test Error Rate: 0.1518519
```

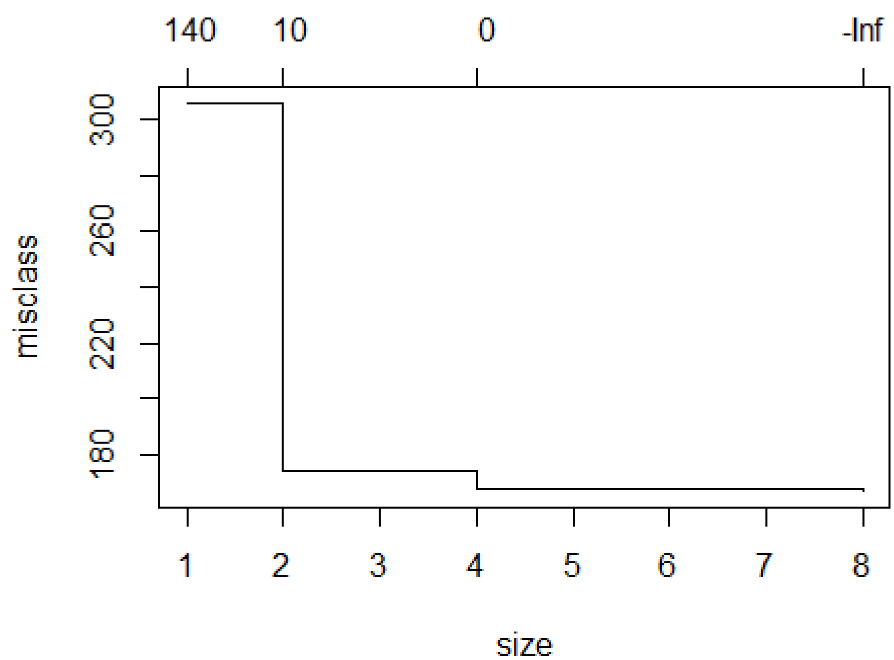
(f)

以$cv.tree$優化模型

```
##        Length Class  Mode
## size   4      -none- numeric
## dev    4      -none- numeric
## k      4      -none- numeric
## method 1      -none- character
```

(g)

可以看到優化結果顯示，$Nodes$為 8 時，有最佳的表現。



(h)

由上述結果，以$Terminal\ Nodes$為 8，進行配飾

```
## Best Size:  8
```

(i)

以上述優化結果與先前相同，因此並無改善。

```
##          actual
## predicted  CH  MM
##        CH 133  15
##        MM  26  96

## Test Error Rate: 0.1518519
```

(j)

優化前後模型並無差別，因此$Train\ Error\ Rate$並無改善。

```
## [1] "Train_Unpruned Error Rate: 0.17625"

## [1] "Train_Pruned Error Rate: 0.17625"
```

(k)

優化前後模型並無差別，因此$Test\ Error\ Rate$並無改善。

```
## [1] "Test_Unpruned Error Rate: 0.151851851851852"

## [1] "Test_Pruned Error Rate: 0.151851851851852"
```

(10)

本題使用資料為$ISLR$裡的$Hitters$
有 322 observations 及 20 variables，變數解釋如下：
* AtBat
Number of times at bat in 1986
* Hits
Number of hits in 1986
* HmRun
Number of home runs in 1986
* Runs
Number of runs in 1986
* RBI
Number of runs batted in in 1986
* Walks
Number of walks in 1986
* Years
Number of years in the major leagues
* CAtBat
Number of times at bat during his career
* CHits
Number of hits during his career

* CHmRun
Number of home runs during his career
* CRuns
Number of runs during his career
* CRBI
Number of runs batted in during his career
* CWalks
Number of walks during his career
* League
A factor with levels A and N indicating player's league at the end of 1986
* Division
A factor with levels E and W indicating player's division at the end of 1986
* PutOuts
Number of put outs in 1986
* Assists
Number of assists in 1986
* Errors
Number of errors in 1986
* Salary
1987 annual salary on opening day in thousands of dollars
* NewLeague
A factor with levels A and N indicating player's league at the beginning of 1987

(a)

首先將由遺漏值的資料刪去後，再將目標解釋變數$Salary$取對數

(b)

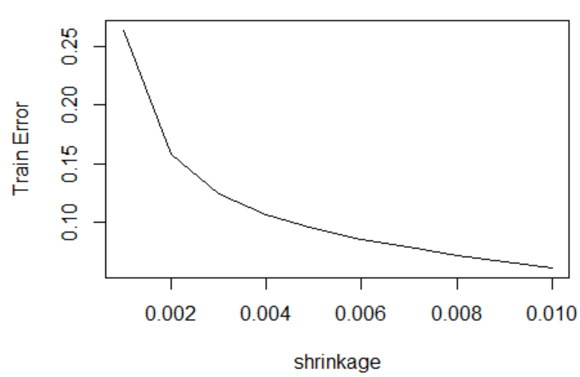
以前 200 筆為$Train$，剩下為$Test$

```
## Train: 200 20 Test: 63 20
```

(c)

取0.001 *到* 0.01之間 10 等分為$shrinkage$，可以看到$TrainError$隨$shrinkage$增加而降低。

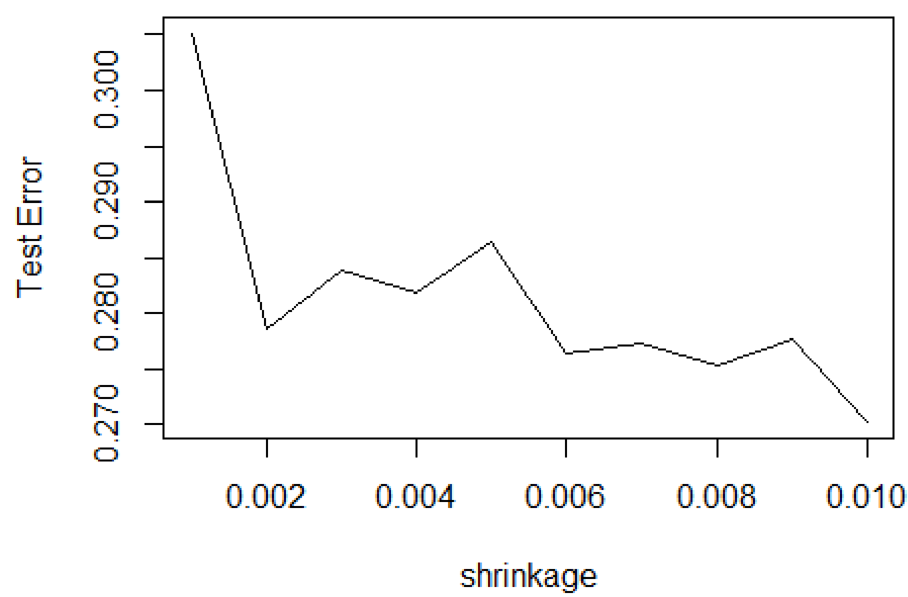
```
## Warning: package 'gbm' was built under R version 4.0.3

## Loaded gbm 2.1.8
```



(d)

以相同$Shrinkage\ Set$進行$Train$的配適，但所得到的$TestError$並不像$TrainError$穩健遞減，偶有反升的趨勢。

```
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
##
## Using 1000 trees...
```



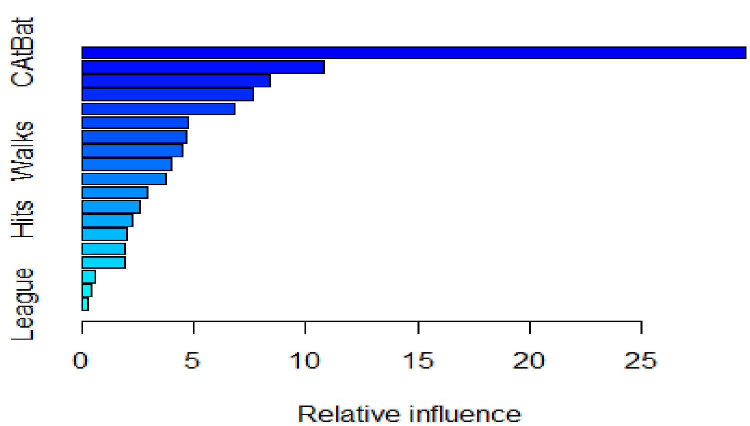
(e)

以多元回歸及 Lasso 方式估計與 Boosting 比較，可得下列之 Test MSE
明顯 Boosting 的 MSE 比另外兩個方法更佳。

```
## Warning: package 'glmnet' was built under R version 4.0.3

## Loading required package: Matrix

## Loaded glmnet 4.0-2

## [1] "Test Error MSE of LM: 0.491795937545494"

## [1] "Test Error MSE of Lasso: 0.457028250141243"

## Using 1000 trees...

## [1] "Test Error MSE of Boosting: 0.287606612725594"
```

(f)

由$Variance\ Importance\ Plot$與下表可得到$CAtBat$、$CRBI$為最有解釋力的前兩個變數。




```
##                 var    rel.inf
## CAtBat       CAtBat 29.6511477
## CWalks       CWalks 10.8355583
## CHits         CHits  8.3629778
## CRuns         CRuns  7.6238787
## CRBI           CRBI  6.8453865
## PutOuts     PutOuts  4.7338062
## Years         Years  4.6380926
## Walks         Walks  4.5147326
## AtBat         AtBat  3.9595985
## CHmRun       CHmRun  3.7707869
## Assists     Assists  2.9486224
## RBI             RBI  2.5755878
## Hits           Hits  2.2579686
## HmRun         HmRun  1.9932002
## Runs           Runs  1.9551249
## Errors       Errors  1.9504051
```

```
## NewLeague NewLeague  0.6191589
## Division   Division  0.4644946
## League       League  0.2994717
```

(g)

此處使用$Bagging$，發現$TestMSE$ 表現比$Boosting$更佳。

```
## [1] "Test Error MSE of Bagging: 0.234267450868388"
```

附錄(Code)

```
set.seed(123456)

X1 = rnorm(10000)

X2 = runif(10000)

X3 = rexp(10000)

X4 = rt(10000,10)

Y = 10*X1+15*X2+20*X3+25*X4

data= cbind(X1,X2,X3,X4,Y) %>% as.data.frame()

library(tree)

data = mtcars

controls = tree.control(nobs = 33,mincut = 5,minsize = 10)

tree.Y = tree(mpg~.,control = controls,data = data)

summary(tree.Y)

plot(tree.Y)

# 8

## (a)

library(ISLR)

library(tree)

data("Carseats")

Carseats$Sales_1 = as.factor(ifelse(Carseats$Sales <= 8, "Low", "High"))

set.seed(1234567)

index = sample(1:400,size = 320,replace = F)

train = Carseats[index,]

test = Carseats[-index,]

## (b)

model = tree(Sales~.,data = train[,-12])

plot(model)
```

```
text(model)

summary(model)

fit= predict(model,test[,-12])

sum((test$Sales - fit)^2)/nrow(test) #rmse

fit = ifelse(fit<=8,"Low","High")

table(predicted = fit, actual = test$Sales_1)

accuracy = function(actual, predicted) {

  mean(actual == predicted)

}

1-accuracy(predicted = fit, actual = test$Sales_1)

## (c)

set.seed(1234)

tree_cv = cv.tree(model, FUN = prune.tree,K = 10)

min_idx = which.min(tree_cv$dev)

tree_cv$size[min_idx]

tree_cv$dev/length(index)

# default plot

plot(tree_cv)

tree_prune = prune.tree(model,best = tree_cv$size[min_idx])

fit= predict(tree_prune,test)

sum((test$Sales - fit)^2)/nrow(test) #rmse

fit = ifelse(fit<=8,"Low","High")

table(predicted = fit, actual = test$Sales_1)

1-accuracy(predicted = fit, actual = test$Sales_1)

##(d)

library(randomForest)

bag = randomForest(Sales ~., data = train[,-12], mtry= 10,

         importance = TRUE, ntrees = 500)

fit= predict(bag,test[,-12])

sum((test$Sales - fit)^2)/nrow(test) #rmse

fit = ifelse(fit<=8,"Low","High")

table(predicted = fit, actual = test$Sales_1)

1-accuracy(predicted = fit, actual = test$Sales_1)
```

```
varImpPlot(bag, type = 1)

## (e)

library(randomForest)

rf = randomForest(Sales ~., data = train[,-12],mtry=4,

                    importance = TRUE, ntrees = 500)

fit= predict(rf,test[,-12])

sum((test$Sales - fit)^2)/nrow(test) #rmse

fit = ifelse(fit<=8,"Low","High")

table(predicted = fit, actual = test$Sales_1)

1-accuracy(predicted = fit, actual = test$Sales_1)

varImpPlot(rf, type = 1)

# (9)

## (a)

data(OJ)

index = sample(1:1070,800,replace = F)

train = OJ[index,]

test = OJ[-index,]

## (b)

tree = tree(Purchase~.,data = train)

summary(tree)

#(c)

tree

#(d)

plot(tree)

text(tree)

#(e)

fit=predict(tree,test,type = "class")

table(predicted = fit, actual = test$Purchase)

1- accuracy(predicted = fit, actual = test$Purchase)

#(f)

cv_tree = cv.tree(tree, FUN = prune.misclass)

#(g)

plot(cv_tree)
```

```
#(h)

min_idx = which.min(cv_tree$dev)

cv_tree$size[min_idx]

#(i)

tree_prune = prune.tree(tree,best = cv_tree$size[min_idx])

fit= predict(tree_prune,test,type="class")

table(predicted = fit, actual = test$Purchase)

1- accuracy(predicted = fit, actual = test$Purchase)

#(j)

fit= predict(tree,train,type="class")

fit_1= predict(tree_prune,train,type="class")

cat("Train_Unpruned Error Rate: ",1- accuracy(predicted = fit, actual = train$Purchase))

cat("Train_Pruned Error Rate: ",1- accuracy(predicted = fit_1, actual = train$Purchase))

#(k)

fit= predict(tree,test,type="class")

fit_1= predict(tree_prune,test,type="class")

cat("Test_Unpruned Error Rate: ",1- accuracy(predicted = fit, actual = test$Purchase))

cat("Test_Pruned Error Rate: ",1- accuracy(predicted = fit_1, actual = test$Purchase))

#(10)

## (a)

data("Hitters")

Hitters = na.omit(Hitters)

Hitters$Salary = log(Hitters$Salary)

## (b)

train = Hitters[1:200,]

test = Hitters[201:263,]

## (c)

library(gbm)

s = seq(0.001,0.01,by = 0.001)

t = sapply(1:10, function(a){

  boost = gbm(Salary ~ ., data = train, distribution = "gaussian",

                   n.trees = 1000, interaction.depth = 4, shrinkage = s[a])

  c(boost$shrinkage,tail(boost$train.error,1))
```

```
})
t = t(t) %>% as.data.frame()
names(t)= c("shrinkage","train_error")
plot(x=t$shrinkage,y=t$train_error,type="l",xlab = "shrinkage",ylab = "Train Error")
## (d)
s = seq(0.001,0.01,by = 0.001)
t = sapply(1:10, function(a){
  boost = gbm(Salary ~ ., data = train, distribution = "gaussian",
              n.trees = 1000, interaction.depth = 4, shrinkage = s[a])
  fit = predict(boost,test)
  mse = mean((fit-test$Salary)^2)
  c(boost$shrinkage,mse)
})
t = t(t) %>% as.data.frame()
names(t)= c("shrinkage","test_error")
plot(x=t$shrinkage,y=t$test_error,type="l",xlab = "shrinkage",ylab = "Test Error")
#(e)
library(glmnet)
lm = lm(Salary ~ ., data = train)
fit = predict(lm,test)
mean((fit - test$Salary)^2)
train_1 = model.matrix(Salary ~ ., data = train)
test_1 = model.matrix(Salary ~ ., data = test)
y = train$Salary
lasso = glmnet(train_1, y, alpha = 0)
fit = predict(lasso, s = 0.01, newx = test_1)
mean((fit - test$Salary)^2)
#(f)
summary(boost)
#(g)
bag <- randomForest(Salary ~ ., data = train, mtry = 19, ntree = 500)
fit <- predict(bag, newdata = test)
mean((fit - test$Salary)^2)
```