519095_Bosan_Individual2

2023-12-16

##Exercise 6.8: Problem 8 (parts e & f) ##e

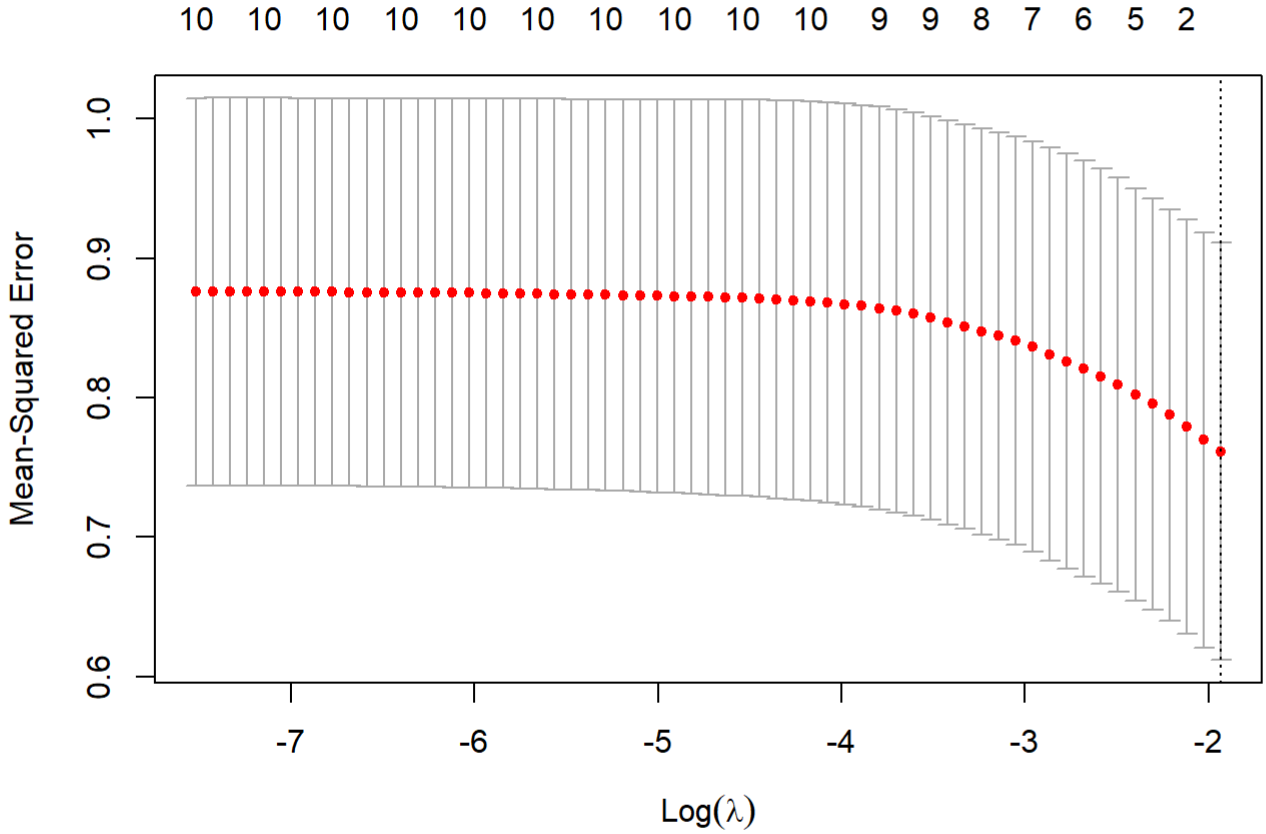
```
library(glmnet)
```

```
## Warning: 套件 'glmnet' 是用 R 版本 4.3.2 來建造的
```

```
## 載入需要的套件：Matrix
```

```
## Loaded glmnet 4.1-8
```

```
set.seed(15)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
y <- rnorm(n)

train_indices <- sample(1:n, size = 0.8 * n)
X_train <- X[train_indices, ]
y_train <- y[train_indices]
X_test <- X[-train_indices, ]
y_test <- y[-train_indices]

cv.out <- cv.glmnet(X_train, y_train, alpha = 1)

plot(cv.out)
```




```
best_lambda <- cv.out$lambda.min
lasso.mod <- glmnet(X_train, y_train, alpha = 1, lambda = best_lambda)
lasso.pred <- predict(lasso.mod, s = best_lambda, newx = X_test)

mse <- mean((lasso.pred - y_test)^2)
lasso.coef <- coef(lasso.mod, s = best_lambda)
print(lasso.coef)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -0.07362781
## V1           0.00000000
## V2           .
## V3           .
## V4           .
## V5           .
## V6           .
## V7           .
## V8           .
## V9           .
## V10          .
```

```
print(mse)
```

```
## [1] 0.9363548
```

```
##Lazzo can remove some predictors
```

#f

```
library(glmnet)
set.seed(15)


beta_0 <- 1.5
beta_7 <- 2.0
Y <- beta_0 + beta_7 * X[, 7] + rnorm(n)


Y_train <- Y[train_indices]
Y_test <- Y[-train_indices]


lasso_mod_Y <- glmnet(X_train, Y_train, alpha = 1, lambda = best_lambda)
lasso_pred_Y <- predict(lasso_mod_Y, s = best_lambda, newx = X_test)


mse_Y <- mean((lasso_pred_Y - Y_test)^2)


lasso_coef_Y <- coef(lasso_mod_Y, s = best_lambda)
print(lasso_coef_Y)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 1.5152170
## V1          0.8678231
## V2          .
## V3          .
## V4          .
## V5          .
## V6          .
## V7          1.8721727
## V8          .
## V9          .
## V10         .
```

```
print(mse_Y)
```

```
## [1] 0.05644347
```

```r
best_subset <- function(X, Y, size) {
  n <- ncol(X)
  best_score <- Inf
  best_model <- NULL

  for (i in 1:size) {
    combinations <- combn(n, i, simplify = FALSE)
    for (comb in combinations) {
      X_subset <- as.matrix(X[, comb, drop = FALSE])
      # SKIP glmnet IF ONE COL
      if (ncol(X_subset) == 1) {
        next
      }
      model <- glmnet(X_subset, Y, alpha = 1, lambda = best_lambda)
      pred <- predict(model, newx = X_subset, s = best_lambda)
      mse <- mean((pred - Y)^2)

      if (mse < best_score) {
        best_score <- mse
        best_model <- comb
      }
    }
  }

  return(list("model" = best_model, "score" = best_score))
}

best_subset_result <- best_subset(X_train, Y_train, p)
print(best_subset_result)
```

```
## $model
## [1] 1 3 5 7
##
## $score
## [1] 0.03717781
```

```
# The 1th, 3th, 5th, 7th predictors work best in this Lazzo model, with only 0.0372 MSE
```

##Exercise 8.4: Problem 8 (parts a, b, & c) ##Problem #8: In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative ##response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as ##a quantitative variable.

##(a) Split the data set into a training set and a test set.

```r
library(ISLR)
```

```
## Warning: 套件 'ISLR' 是用 R 版本 4.3.2 來建造的
```

```
set.seed(1)

train = sample(1:nrow(Carseats), nrow(Carseats)/2)
car_train = Carseats[train, ]
car_test = Carseats[-train,]
```

##(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

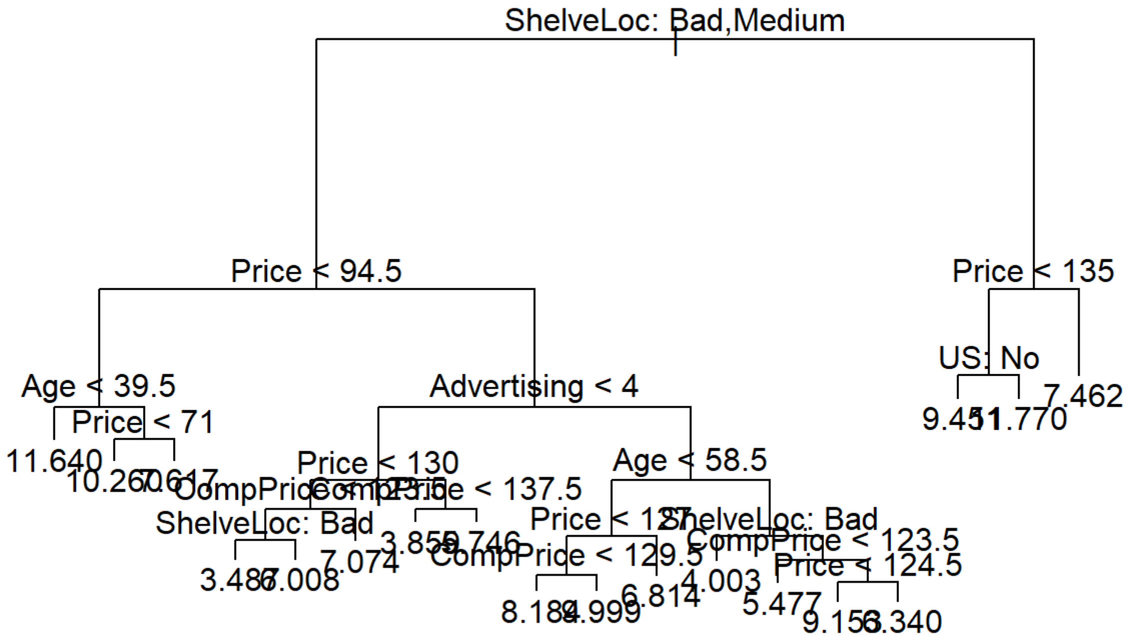
```
library(tree)
```

```
## Warning: 套件 'tree' 是用 R 版本 4.3.2 來建造的
```

```
# train the tree
tree_regression = tree(Sales~.,data = car_train)
summary(tree_regression)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = car_train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "CompPrice"
## [6] "US"
## Number of terminal nodes:  18
## Residual mean deviance:  2.167 = 394.3 / 182
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.88200 -0.88200 -0.08712  0.00000  0.89590  4.09900
```

```
plot(tree_regression)
text(tree_regression ,pretty =0)
```

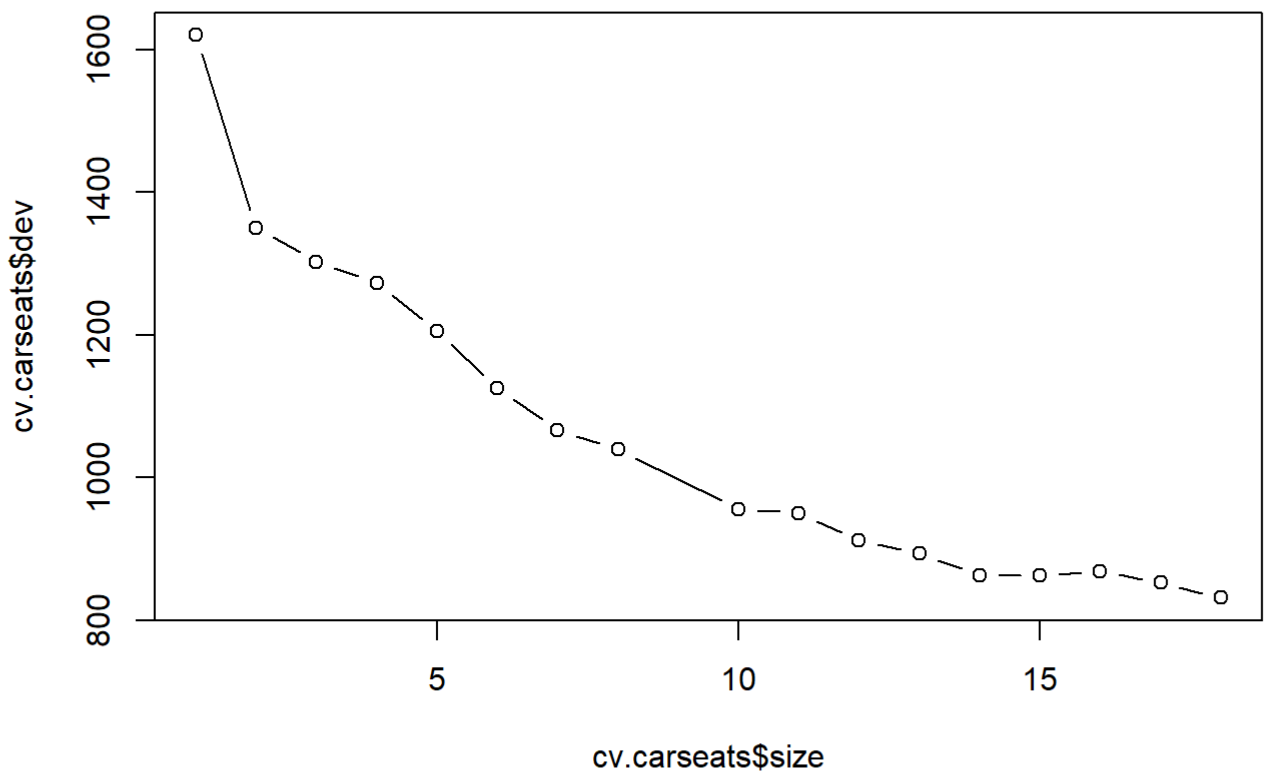


```
tree_prediction = predict(tree_regression, newdata=car_test)
tree_MSE <- mean((tree_prediction - car_test$Sales)^2)
tree_MSE
```

```
## [1] 4.922039
```

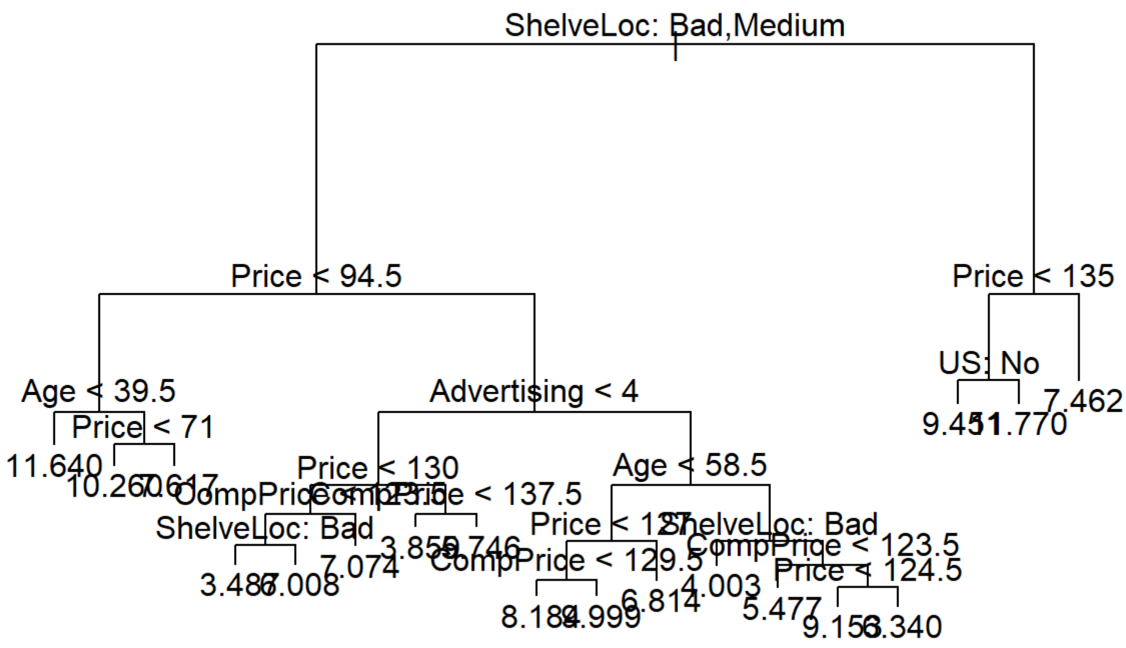
##(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
cv.carseats = cv.tree(tree_regression, FUN = prune.tree)
plot(cv.carseats$size, cv.carseats$dev, type = "b")
```



The graph above shows that deviation keep dropping when the size is 18 ## Thus, I picked 18 as the best size

```
prune_car = prune.tree(tree_regression, best = 18)
plot(prune_car)
text(prune_car, pretty = 0)
```




```
prune_prediction = predict(prune_car, newdata= car_test)

Prune_MSE = mean((prune_prediction - car_test$Sales)^2)
Prune_MSE
```

```
## [1] 4.922039
```

##The best size is 18. (Cross-Validation) ##In my test, the MSE remained the same after pruning the trees (remained 4.922039)

```

```

##Problem #8: In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable. # (d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```
library(randomForest)
```

```
## Warning: 套件 'randomForest' 是用 R 版本 4.3.2 來建造的
```

```
## randomForest 4.7-1.1
```

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

```
set.seed(88)
bagging_car = randomForest( Sales~., data = car_train, mtry = 10, importance = TRUE)
yhat_bagging = predict( bagging_car, newdata = car_test)
bagging_mse = mean((yhat_bagging - car_test$Sales)^2)
print(bagging_mse)
```

```
## [1] 2.618008
```

```
importance(bagging_car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   27.909312     172.84191
## Income       6.098099      92.62640
## Advertising 13.203302     100.80551
## Population  -2.264739      59.04411
## Price       56.594782     506.92540
## ShelveLoc   49.366388     369.15451
## Age         17.079423     157.21384
## Education    0.415839      44.67035
## Urban        1.710869      10.09492
## US           3.875395      17.49415
```

```
## The MSE using bagging is 2.618008
## Price's "%IncMSE" is 56.594782 and "it'sIncNodePurity" is 506.92540, which these both valu
es are the highest values.
## Thus, Price is the most important variable in this case.
```

(e)Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
set.seed(88)
randomforest_car = randomForest(Sales~., data = car_train, mtry = 3, importance = TRUE)
yhat_randomforest = predict(randomforest_car, newdata = car_test)
mse_randomforest = mean((yhat_randomforest - car_test$Sales)^2)
print(mse_randomforest)
```

```
## [1] 3.005177
```

```
importance(randomforest_car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   13.7741065     151.14646
## Income       2.8594620     125.83963
## Advertising  8.4696416     108.00899
## Population  -3.2377263     101.33362
## Price       35.7581303     389.53650
## ShelveLoc   34.7360147     293.78828
## Age         14.1990788     177.31774
## Education    2.0900556      76.26253
## Urban        0.3356185      16.59915
## US           5.1115996      33.41772
```

```
## The MSE using random forest is 3.005177 > 2.618008, there is no improvement using rf inste
ad of bagging in this case
## Price's "%IncMSE" is 56.594782 and "it'sIncNodePurity" is 506.92540, which these both valu
es are the highest values.
## Thus, Price is the most important variable in this case.
```

#Problem #10: We now use boosting to predict Salary in the Hitters data set. #(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
library(dplyr)
```

```
##
## 載入套件：'dplyr'
```

```
## 下列物件被遮斷自 'package:randomForest':
##
##     combine
```

```
## 下列物件被遮斷自 'package:stats':
##
##     filter, lag
```

```
## 下列物件被遮斷自 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data("Hitters")

Hitters %>%
  filter(!is.na(Salary)) %>%
  mutate(Salary = log(Salary)) -> Hitters

head(Hitters)
```

```
##                     AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Alan Ashby           315   81     7   24  38    39    14   3449   835     69
## -Alvin Davis          479  130    18   66  72    76     3   1624   457     63
## -Andre Dawson         496  141    20   65  78    37    11   5628  1575    225
## -Andres Galarraga     321   87    10   39  42    30     2    396   101     12
## -Alfredo Griffin      594  169     4   74  51    35    11   4408  1133     19
## -Al Newman            185   37     1   23   8    21     2    214    42      1
##                     CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Alan Ashby           321  414    375      N        W     632      43     10
## -Alvin Davis          224  266    263      A        W     880      82     14
## -Andre Dawson         828  838    354      N        E     200      11      3
## -Andres Galarraga      48   46     33      N        E     805      40      4
## -Alfredo Griffin      501  336    194      A        W     282     421     25
## -Al Newman             30    9     24      N        E      76     127      7
##                       Salary NewLeague
## -Alan Ashby         6.163315         N
## -Alvin Davis        6.173786         A
## -Andre Dawson       6.214608         N
## -Andres Galarraga 4.516339         N
## -Alfredo Griffin    6.620073         A
## -Al Newman          4.248495         A
```

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

```
hit_train <- Hitters[1:200,]
hit_test <- Hitters[-(1:200),]
```

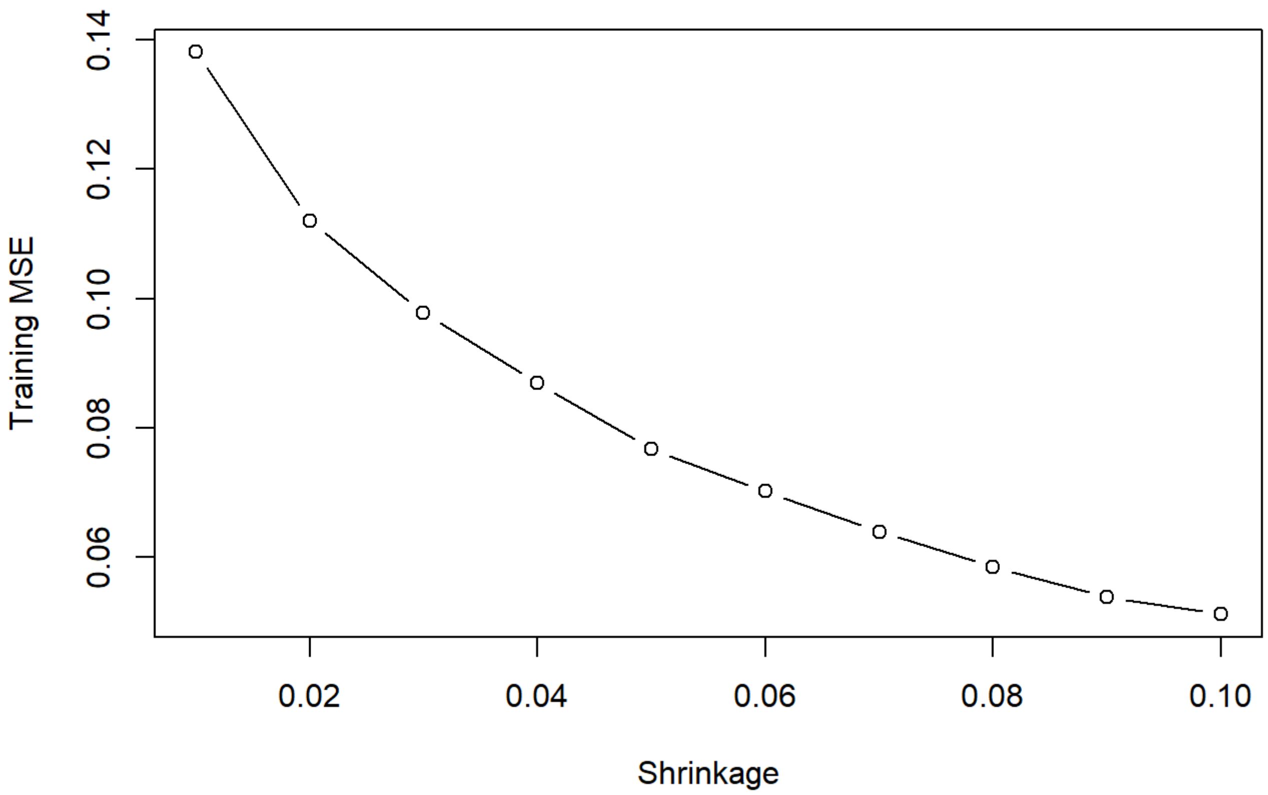
c. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

```
library(gbm)
```

```
## Warning: 套件 'gbm' 是用 R 版本 4.3.2 來建造的
```

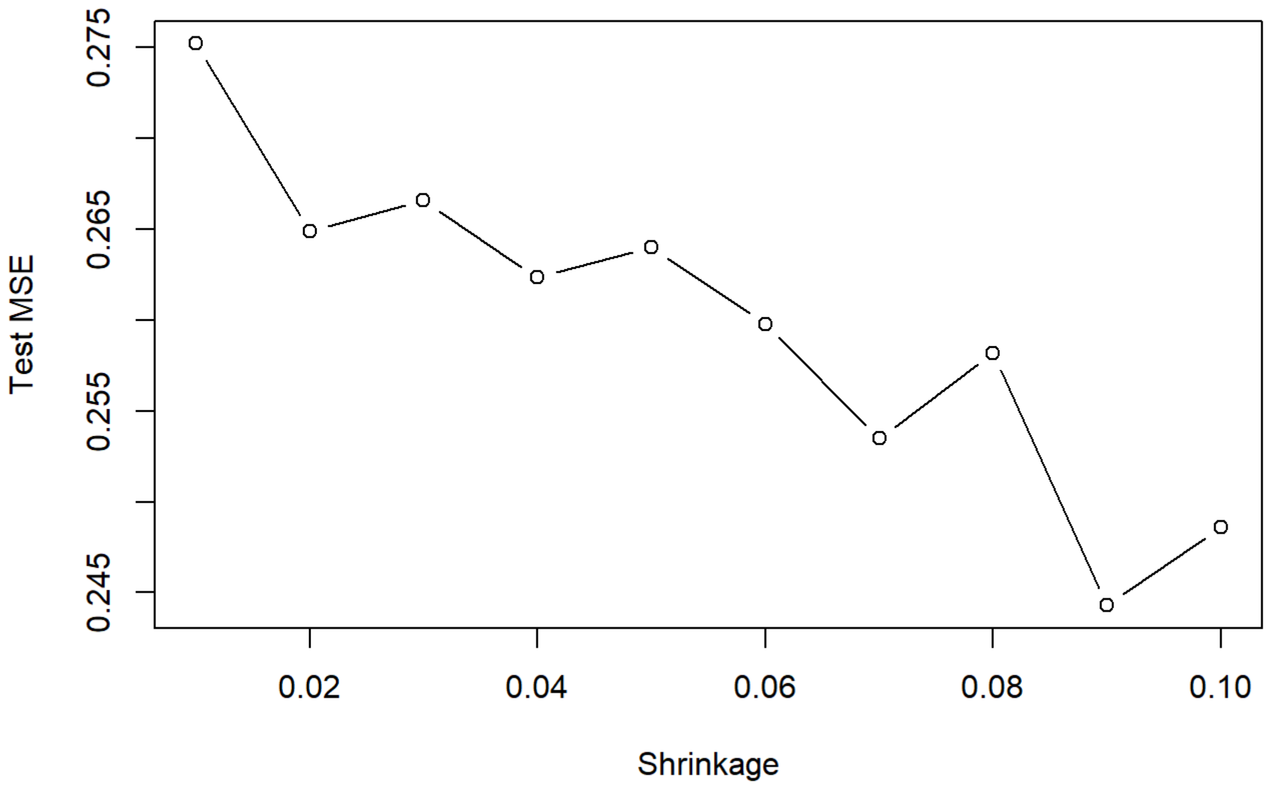
```
## Loaded gbm 2.1.8.1
```

```r
shrinkage_values <- seq(0.01, 0.1, by = 0.01)

#store every model's error
train_mse <- rep(NA, length(shrinkage_values))

#boosting
for (i in seq_along(shrinkage_values)) {
  set.seed(10)
  gbm_model <- gbm(Salary ~ ., data = hit_train, distribution = "gaussian",
                   n.trees = 1000, shrinkage = shrinkage_values[i])
  pred <- predict(gbm_model, hit_train, n.trees=1000)
  train_mse[i] <- mean((pred - hit_train$Salary)^2)
}


plot(shrinkage_values, train_mse, type="b", xlab="Shrinkage", ylab="Training MSE")
```



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

```r
library(gbm)

shrinkage_values <- seq(0.01, 0.1, by = 0.01)

test_mse <- rep(NA, length(shrinkage_values))

for (i in seq_along(shrinkage_values)) {
  set.seed(10)
  gbm_model <- gbm(Salary ~ ., data = hit_train, distribution = "gaussian",
                   n.trees = 1000, shrinkage = shrinkage_values[i])
  pred <- predict(gbm_model, hit_test, n.trees = 1000)
  test_mse[i] <- mean((pred - hit_test$Salary)^2)
}

plot(shrinkage_values, test_mse, type = "b", xlab = "Shrinkage", ylab = "Test MSE")
```




```r
boosted_mse = min(test_mse)
boosted_mse
```

```
## [1] 0.2442988
```

e. Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```r
library(glmnet)

lm = lm(Salary ~ ., data = hit_train)
lm_prediction = predict(lm, newdata = hit_test)
lm_mse = mean((lm_prediction - hit_test$Salary)^2)

x_train <- model.matrix(Salary ~ ., hit_train)[,-1]
y_train <- hit_train$Salary
x_test <- model.matrix(Salary ~ ., hit_test)[,-1]
y_test <- hit_test$Salary

lasso <- glmnet(x_train, y_train, alpha = 1, lambda = 0.1)
lasso_predictions <- predict(lasso, s = 0.1, newx = x_test)
lasso_test_mse <- mean((lasso_predictions - y_test)^2)

print(paste("Linear Test MSE:", lm_mse))
```

```
## [1] "Linear Test MSE: 0.491795937545494"
```

```r
print(paste("Lazzo Test MSE:", lasso_test_mse))
```

```
## [1] "Lazzo Test MSE: 0.43874517155745"
```

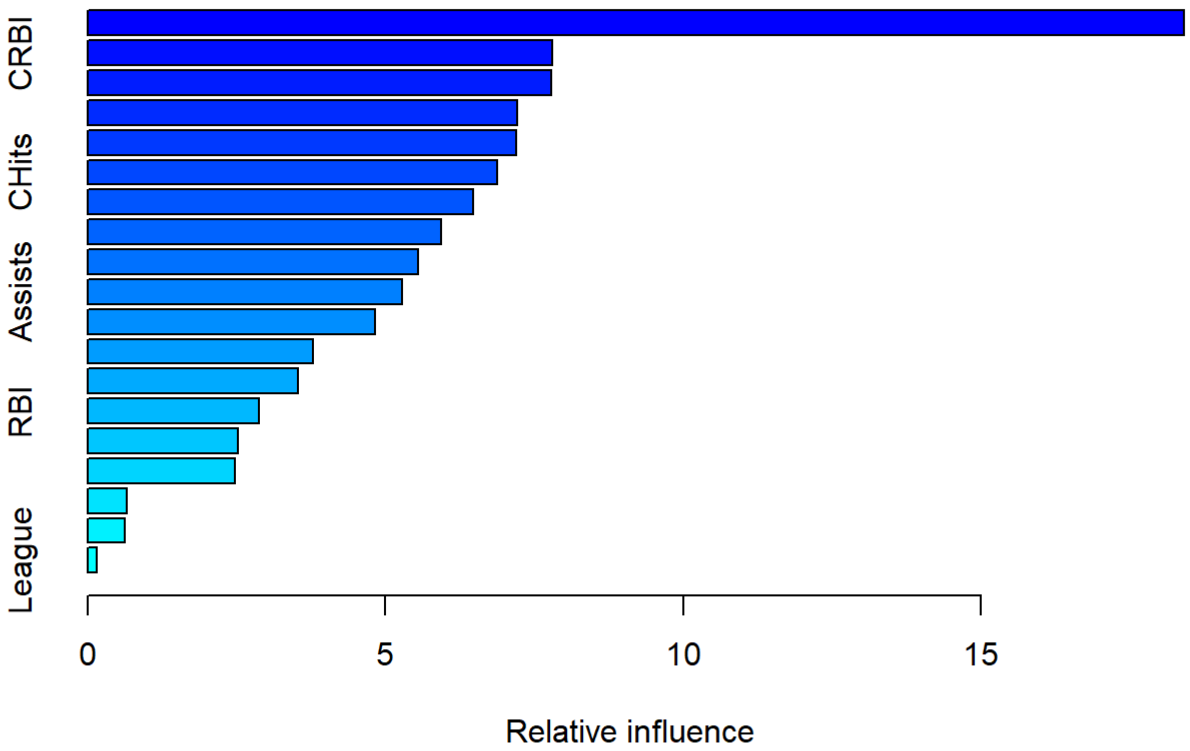
```r
print(paste("Boosted Test MSE:", boosted_mse))
```

```
## [1] "Boosted Test MSE: 0.244298771115224"
```

```r
#Boosted has the smallest MSE, 0.2443.
```

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

```r
boosted.model = gbm(Salary~., data = hit_train, distribution = "gaussian", n.trees = 1000, sh
rinkage=shrinkage_values[which.min(test_mse)])
summary(boosted.model)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 18.4102380
## CRBI           CRBI  7.8057942
## Years         Years  7.7893112
## PutOuts     PutOuts  7.2161375
## Walks         Walks  7.1898528
## CHits         CHits  6.8821517
## CWalks       CWalks  6.4770790
## CHmRun       CHmRun  5.9336103
## Hits           Hits  5.5554983
## Assists     Assists  5.2785393
## CRuns         CRuns  4.8314369
## HmRun         HmRun  3.7926550
## AtBat         AtBat  3.5392703
## RBI             RBI  2.8738555
## Errors       Errors  2.5164422
## Runs           Runs  2.4744086
## Division   Division  0.6613331
## NewLeague NewLeague  0.6234775
## League       League  0.1489086
```

```
#CAtBat is the most important variable( highest rel.inf )
```

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

```
bagging = randomForest(Salary~., data = hit_train, distribution = "gaussian", n.trees = 500,
shrinkage = lambdas[which.min(test.error)], mtry = 19, importance = TRUE)
bagging_prediction = predict(bagging, hit_test)

bagging_test_mse = mean((bagging_prediction - hit_test$Salary)^2)
bagging_test_mse
```

```
## [1] 0.2298841
```

```
#The test MSE is 0.2321575 in this approach
```

#Proble, 11.4 #Direct Mailing to Airline Customers. East-West Airlines has entered into a partnership with the wireless phone company Telcon to sell the latter's service via direct mail. The file EastWestAirlinesNN.csv Download EastWestAirlinesNN.csv contains a subset of a data sample of who has already received a test offer. About 13% accepted.

#You are asked to develop a model to classify East-West customers as to whether they purchase a wireless phone service contract (outcome variable Phone_Sale). This model will be used to classify additional customers.

#1. Run a neural net model on these data, using a single hidden layer with 5 nodes. Remember to first convert categorical variables into dummies and scale numerical predictor variables to a 0-1 (use function preprocess() with method="range" - see Chapter 7). Generate a deciles-wise lift chart for the training and validation sets. Interpret the meaning (in business terms) of the leftmost bar of the validation decile- wise lift chart.

```
library(caret)
```

```
## Warning: 套件 'caret' 是用 R 版本 4.3.2 來建造的
```

```
## 載入需要的套件：ggplot2
```

```
## Warning: 套件 'ggplot2' 是用 R 版本 4.3.2 來建造的
```

```
##
## 載入套件：'ggplot2'
```

```
## 下列物件被遮斷自 'package:randomForest':
##
##     margin
```

```
## 載入需要的套件：lattice
```

```
library(nnet)
library(ggplot2)
library(dplyr)

# Load your data
data <- read.csv("EastWestAirlinesNN.csv")
data <- na.omit(data)

# Convert categorical variables into dummy variables
dummies <- dummyVars("~ .", data = data)
data_transformed <- predict(dummies, newdata = data)

# Scale numerical predictor variables to a 0-1 range
preproc <- preProcess(data_transformed, method = "range")
data_scaled <- predict(preproc, data_transformed)


if (!is.data.frame(data_scaled)) {
  data_scaled <- as.data.frame(data_scaled)
}

if (!("Phone_sale" %in% names(data_scaled))) {
  data_scaled$Phone_sale <- data$Phone_sale
}

# Split the data into training and validation sets
set.seed(123) # for reproducibility
trainingIndex <- createDataPartition(data_scaled$Phone_sale, p = .8, list = TRUE)
trainingData <- data_scaled[trainingIndex[[1]], ]
validationData <- data_scaled[-trainingIndex[[1]], ]
```

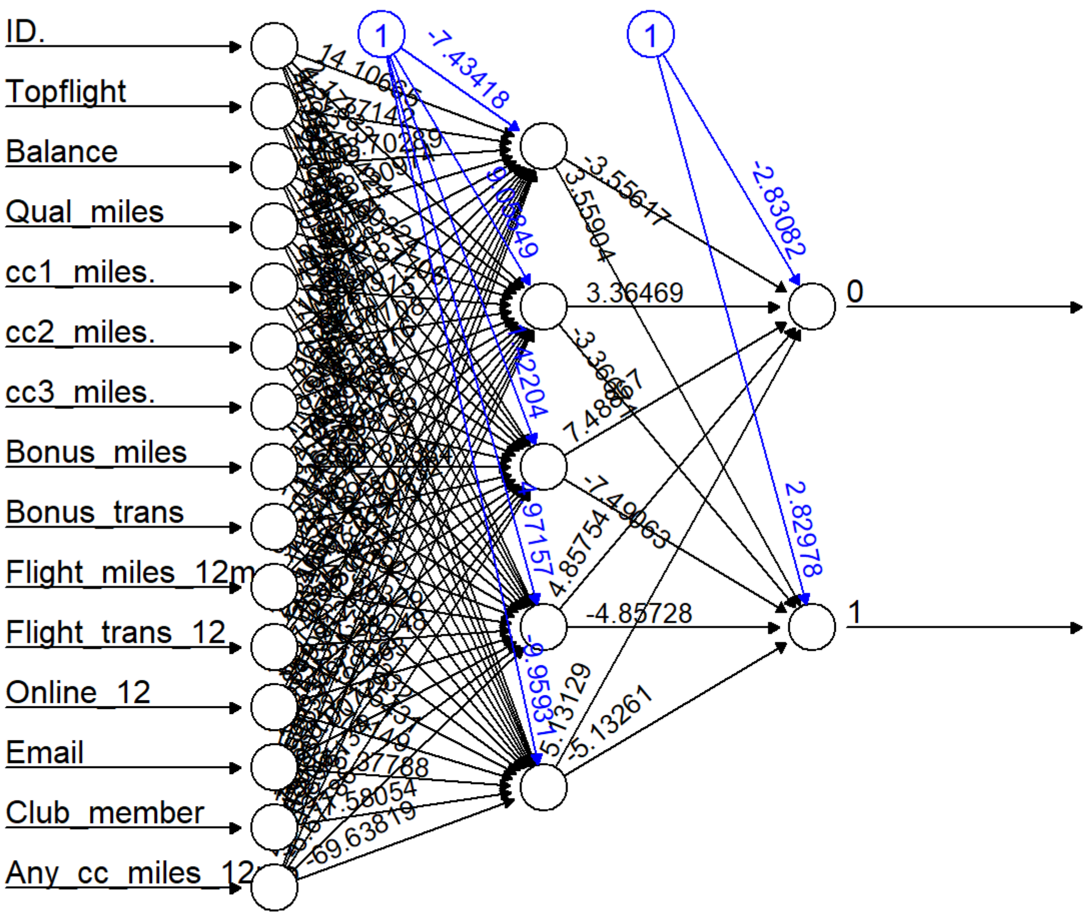
```
# Load necessary libraries
library(caret)
library(nnet)
library(ggplot2)
library(dplyr)
#install.packages("neuralnet")
library(neuralnet)
```

```
## Warning: 套件 'neuralnet' 是用 R 版本 4.3.2 來建造的
```

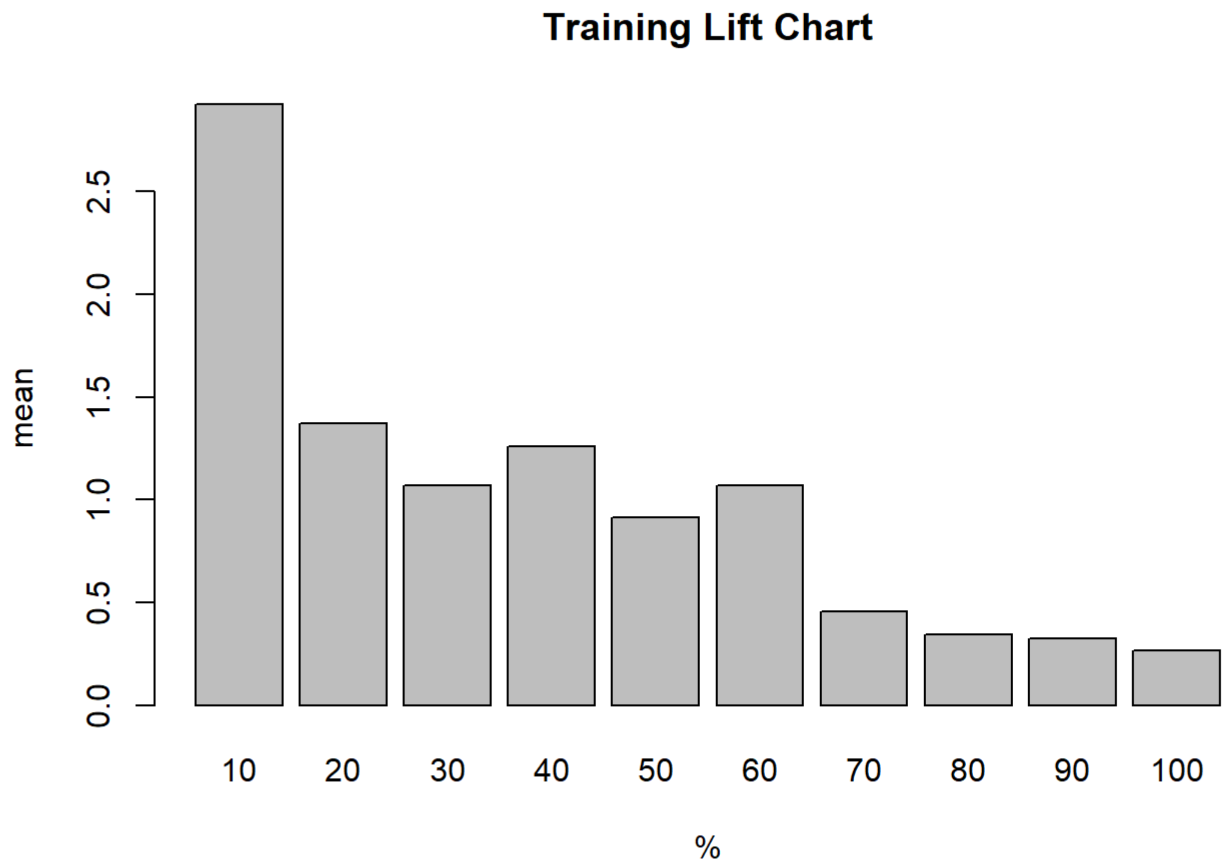
```
##
## 載入套件：'neuralnet'
```

```
## 下列物件被遮斷自 'package:dplyr':
##
##     compute
```

```
# Load your data
data <- read.csv("EastWestAirlinesNN.csv")
data <- na.omit(data)

# Convert categorical variables into dummy variables
dummies <- dummyVars("~ .", data = data)
data_transformed <- predict(dummies, newdata = data)
data_transformed <- as.data.frame(data_transformed)
data_transformed$Phone_sale <- as.factor(data$Phone_sale)


#scale 0-1
preproc <- preProcess(data_transformed[, -which(names(data_transformed) == "Phone_sale")], me
thod = "range")
data_scaled <- predict(preproc, data_transformed)


data_scaled <- as.data.frame(data_scaled)
data_scaled$Phone_sale <- as.factor(data$Phone_sale)

# Split the data into training and validation sets
set.seed(123) # for reproducibility
trainingIndex <- createDataPartition(data_scaled$Phone_sale, p = .8, list = TRUE)
trainingData <- data_scaled[trainingIndex[[1]], ]
validationData <- data_scaled[-trainingIndex[[1]], ]

nn_model <- neuralnet(Phone_sale ~ ., data = trainingData, hidden = 5, linear.output = FALSE)
plot( nn_model, rep = "best")
```



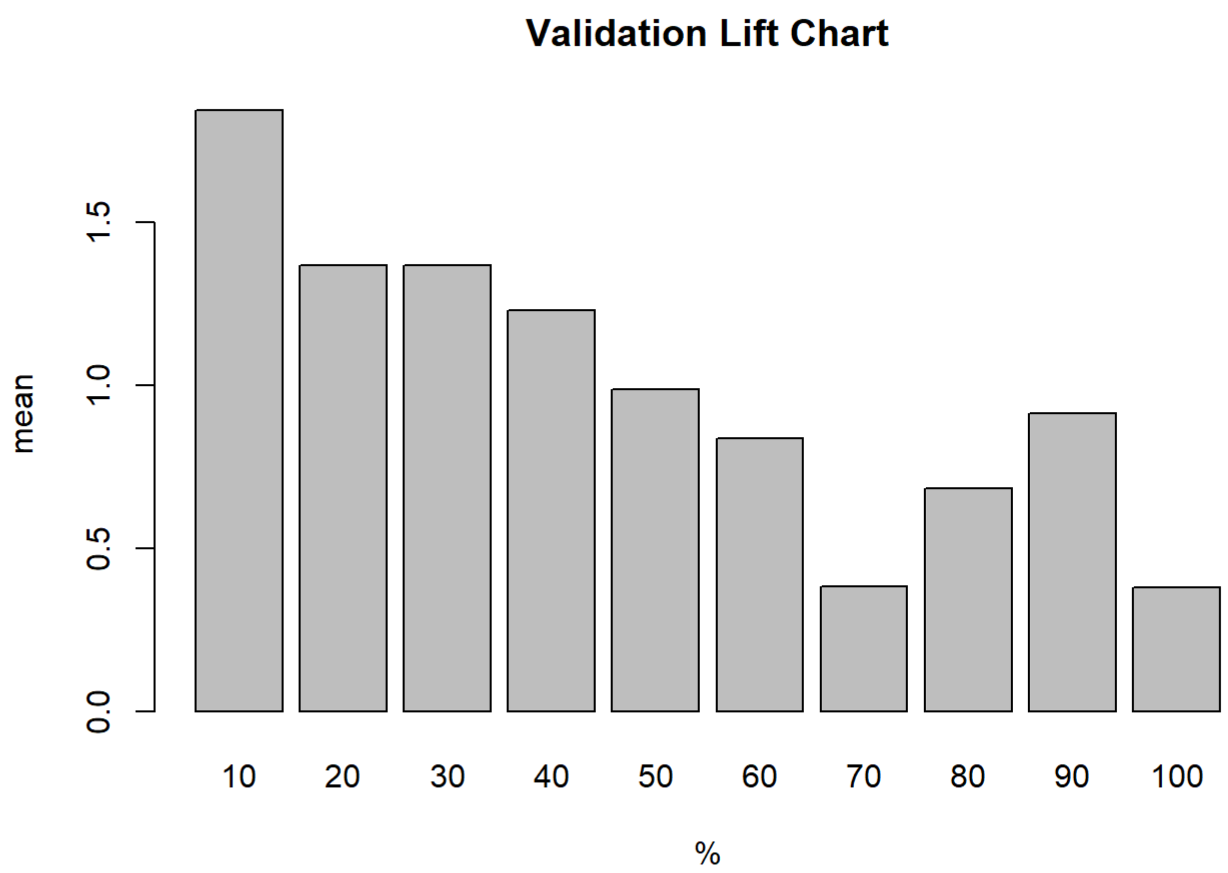
```
#install.packages("gains")
library(gains)
prediction <- predict(nn_model, trainingData, type = "raw")
actual_numeric <- as.numeric(as.character(trainingData$Phone_sale))
predicted_probabilities <- prediction[, 2]
gain <- gains(actual_numeric, predicted_probabilities)
barplot(gain$mean.resp / mean(actual_numeric), names.arg = gain$depth,
        xlab = "%", ylab = "mean", main = "Training Lift Chart")
```




```
## The first bar in a decile-wise lift chart represents the top 10% of cases predicted by a m
odel to be the most likely to have the positive outcome you're interested in.
## For example, this could represent a group of customers most likely to respond to a marketi
ng campaign, the most profitable segment, or those who are most likely to churn, depending on
the context of the model's objective.
```

##2. Comment on the difference between the training and validation lift charts.

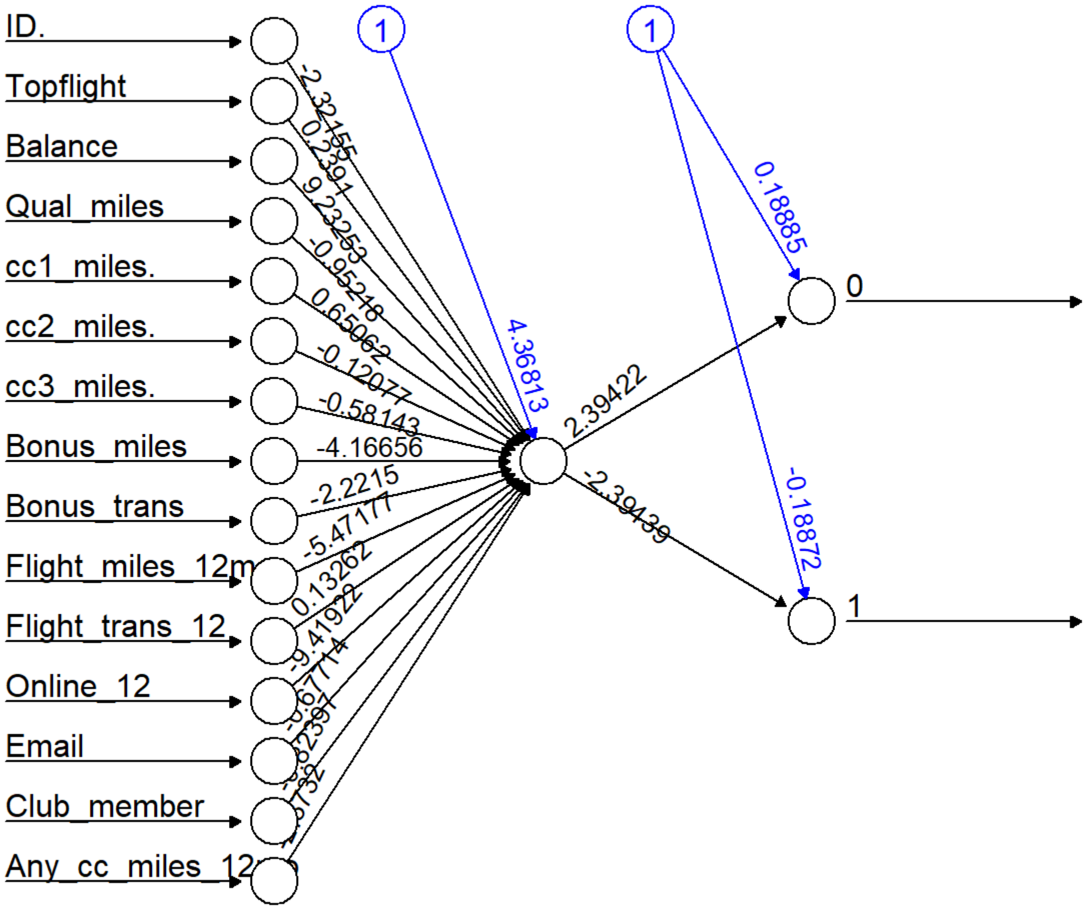
```
prediction_valid <- predict(nn_model, validationData, type = "raw")
actual_numeric_valid <- as.numeric(as.character(validationData$Phone_sale))
predicted_probabilities_valid <- prediction_valid[, 2]
gain_valid <- gains(actual_numeric_valid, predicted_probabilities_valid)
barplot(gain_valid$mean.resp / mean(actual_numeric_valid), names.arg = gain_valid$depth,
        xlab = "%", ylab = "mean", main = "Validation Lift Chart")
```




```
## The training lift chart shows a significantly higher lift in the first decile compared to
the validation lift chart. This suggests that the model may overfits to the training data.
```

3. Run a second neural net model on the data, this time setting the number of hidden nodes to 1. Comment now on the difference between this model and the model you ran earlier, and how overfitting might have affected results.

```
nn_model_2 <- neuralnet(Phone_sale ~ ., data = trainingData, hidden = 1, linear.output = FALS
E)
plot( nn_model_2, rep = "best")
```




```
error_nn_model <- as.character(nn_model$result.matrix[1,])
print(paste("hidden = 5, error =", error_nn_model))
```

```
## [1] "hidden = 5, error = 400.95224779195"
```

```
error_nn_model_2 <- as.character(nn_model_2$result.matrix[1,])
print(paste("hidden = 1, error =", error_nn_model_2))
```

```
## [1] "hidden = 1, error = 436.320567915564"
```

```
## when there is 5 hidden nodes, the error is lower
## however, having more node might overfit the data
```

4. What sort of information, if any, is provided about the effects of the various variables?

```
nn_model$result.matrix
```

```
##                                       [,1]
## error                         4.009522e+02
## reached.threshold             9.859623e-03
## steps                         1.412400e+04
## Intercept.to.1layhid1        -7.434181e+00
## ID..to.1layhid1               1.410665e+01
## Topflight.to.1layhid1         1.771420e+00
## Balance.to.1layhid1           1.137029e+02
## Qual_miles.to.1layhid1       -3.783097e+02
## cc1_miles..to.1layhid1        4.649399e-01
## cc2_miles..to.1layhid1       -1.009413e+01
## cc3_miles..to.1layhid1       -3.629226e+01
## Bonus_miles.to.1layhid1       1.189053e+02
## Bonus_trans.to.1layhid1      -1.049871e+02
## Flight_miles_12mo.to.1layhid1  9.350446e+01
## Flight_trans_12.to.1layhid1    2.241153e+01
## Online_12.to.1layhid1         3.511434e+02
## Email.to.1layhid1             2.258518e+00
## Club_member.to.1layhid1      -7.258084e+01
## Any_cc_miles_12mo.to.1layhid1  4.357676e+01
## Intercept.to.1layhid2         9.058490e+00
## ID..to.1layhid2               2.178335e+00
## Topflight.to.1layhid2        -5.704240e+00
## Balance.to.1layhid2           3.646032e+02
## Qual_miles.to.1layhid2       -4.358771e+02
## cc1_miles..to.1layhid2        8.219150e+01
## cc2_miles..to.1layhid2        1.330108e+01
## cc3_miles..to.1layhid2       -2.333176e+01
## Bonus_miles.to.1layhid2      -7.279435e+01
## Bonus_trans.to.1layhid2      -1.233009e+02
## Flight_miles_12mo.to.1layhid2  1.244364e+02
## Flight_trans_12.to.1layhid2    2.340551e+01
## Online_12.to.1layhid2         6.774487e+01
## Email.to.1layhid2            -1.073709e+01
## Club_member.to.1layhid2      -7.854633e+01
## Any_cc_miles_12mo.to.1layhid2 -1.748063e+01
## Intercept.to.1layhid3         1.422043e+00
## ID..to.1layhid3               4.212456e+00
## Topflight.to.1layhid3         6.328951e+01
## Balance.to.1layhid3           3.473118e+01
## Qual_miles.to.1layhid3       -2.376700e+01
## cc1_miles..to.1layhid3       -3.091388e+01
## cc2_miles..to.1layhid3        5.405492e+01
## cc3_miles..to.1layhid3        5.418715e+00
## Bonus_miles.to.1layhid3      -5.118938e+02
## Bonus_trans.to.1layhid3      -1.995063e+02
## Flight_miles_12mo.to.1layhid3 -8.970327e+02
## Flight_trans_12.to.1layhid3    3.593113e+02
## Online_12.to.1layhid3        -8.991200e+01
## Email.to.1layhid3            -4.136631e+01
## Club_member.to.1layhid3       4.390112e+01
## Any_cc_miles_12mo.to.1layhid3  3.845806e+01
## Intercept.to.1layhid4         4.971570e+00
## ID..to.1layhid4              -4.991784e+00
## Topflight.to.1layhid4         2.582949e+00
```

```
## Balance.to.1layhid4             1.397843e+01
## Qual_miles.to.1layhid4         -9.053788e+00
## cc1_miles..to.1layhid4          7.386216e+00
## cc2_miles..to.1layhid4         -1.700824e+00
## cc3_miles..to.1layhid4          4.081439e+01
## Bonus_miles.to.1layhid4        -3.269719e+00
## Bonus_trans.to.1layhid4         4.341923e+00
## Flight_miles_12mo.to.1layhid4 -3.632895e-01
## Flight_trans_12.to.1layhid4     1.128248e+01
## Online_12.to.1layhid4          -1.193627e+00
## Email.to.1layhid4              -7.786017e-03
## Club_member.to.1layhid4        -2.961337e+00
## Any_cc_miles_12mo.to.1layhid4 -6.677862e+00
## Intercept.to.1layhid5          -9.959312e+00
## ID..to.1layhid5                 4.395670e+01
## Topflight.to.1layhid5          -7.549098e+02
## Balance.to.1layhid5            -1.012107e+03
## Qual_miles.to.1layhid5          1.272584e+03
## cc1_miles..to.1layhid5         -1.167898e+03
## cc2_miles..to.1layhid5         -3.125285e+00
## cc3_miles..to.1layhid5         -1.021083e+03
## Bonus_miles.to.1layhid5         1.373793e+03
## Bonus_trans.to.1layhid5        -5.466323e+02
## Flight_miles_12mo.to.1layhid5 -6.726123e+02
## Flight_trans_12.to.1layhid5     2.393543e+02
## Online_12.to.1layhid5           7.791486e+00
## Email.to.1layhid5               1.637788e+01
## Club_member.to.1layhid5        -7.580537e+00
## Any_cc_miles_12mo.to.1layhid5 -6.963819e+01
## Intercept.to.0                 -2.830816e+00
## 1layhid1.to.0                  -3.556167e+00
## 1layhid2.to.0                   3.364686e+00
## 1layhid3.to.0                   7.488670e+00
## 1layhid4.to.0                   4.857535e+00
## 1layhid5.to.0                   5.131289e+00
## Intercept.to.1                  2.829779e+00
## 1layhid1.to.1                   3.559039e+00
## 1layhid2.to.1                  -3.366714e+00
## 1layhid3.to.1                  -7.490632e+00
## 1layhid4.to.1                  -4.857282e+00
## 1layhid5.to.1                  -5.132612e+00
```

```
#The variance generalized weights are provided
#If the absolute value of the weight is high, it means that the variable has great impact on the outcome
```

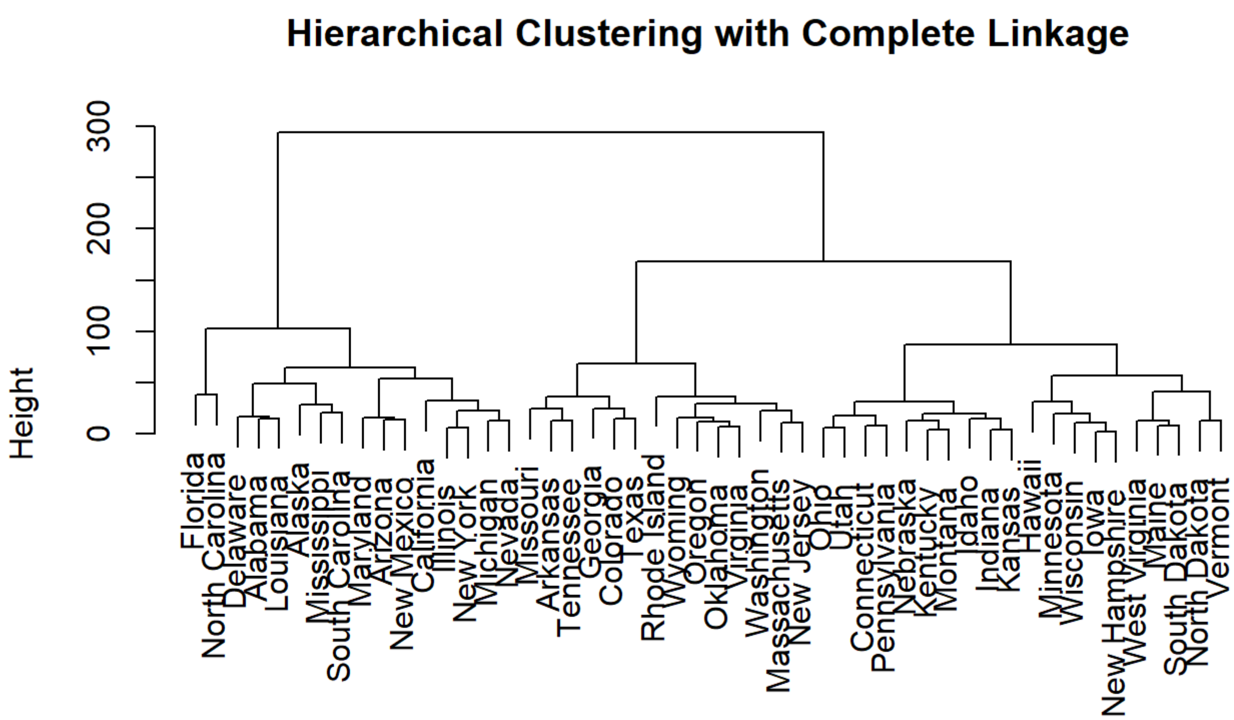
Exercise 10.7: Problem 9

ISLR p.417

9. Consider the USArrests data. We will now perform hierarchical clustering on the states.

a. Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
data("USArrests")
cluster_complete <- hclust(dist(USArrests), method = "complete")
plot(cluster_complete, main = "Hierarchical Clustering with Complete Linkage", sub = "", xlab = "")
```



b. Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
# From my perspective, I'll make split the data into three groups
cut_complete <- cutree(cluster_complete, k = 3)


clusters <- data.frame(State = names(cut_complete), Cluster = cut_complete)
clusters_sorted <- clusters[order(clusters$Cluster, clusters$State), ]
print(clusters_sorted)
```

```
##                           State Cluster
## Alabama              Alabama       1
## Alaska                Alaska       1
## Arizona              Arizona       1
## California        California       1
## Delaware            Delaware       1
## Florida              Florida       1
## Illinois            Illinois       1
## Louisiana          Louisiana       1
## Maryland            Maryland       1
## Michigan            Michigan       1
## Mississippi      Mississippi       1
## Nevada                Nevada       1
## New Mexico        New Mexico       1
## New York            New York       1
## North Carolina North Carolina       1
## South Carolina South Carolina       1
## Arkansas            Arkansas       2
## Colorado            Colorado       2
## Georgia              Georgia       2
## Massachusetts   Massachusetts       2
## Missouri            Missouri       2
## New Jersey        New Jersey       2
## Oklahoma            Oklahoma       2
## Oregon                Oregon       2
## Rhode Island    Rhode Island       2
## Tennessee          Tennessee       2
## Texas                  Texas       2
## Virginia            Virginia       2
## Washington        Washington       2
## Wyoming              Wyoming       2
## Connecticut      Connecticut       3
## Hawaii                Hawaii       3
## Idaho                  Idaho       3
## Indiana              Indiana       3
## Iowa                    Iowa       3
## Kansas                Kansas       3
## Kentucky            Kentucky       3
## Maine                  Maine       3
## Minnesota          Minnesota       3
## Montana              Montana       3
## Nebraska            Nebraska       3
## New Hampshire   New Hampshire       3
## North Dakota    North Dakota       3
## Ohio                    Ohio       3
## Pennsylvania    Pennsylvania       3
## South Dakota    South Dakota       3
## Utah                    Utah       3
## Vermont              Vermont       3
## West Virginia   West Virginia       3
## Wisconsin          Wisconsin       3
```

```
#Alabama, Alaska, Arizona, California, Delaware, Florida, Illinois, Louisiana, Maryland, Mich
igan,  Mississippi, Nevada, New Mexico, New York, Carolina North Carolina: Cluster 1

# Arkansas, Colorado, Georgia, Massachusetts, Missouri, New Jersey, Oklahoma,   Oregon, Tenne
ssee, Texas, Virginia, Washington  Washington  2

#Others are in the third cluster
```

c. Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
library(purrr)
```

```
## Warning: 套件 'purrr' 是用 R 版本 4.3.2 來建造的
```

```
##
## 載入套件：'purrr'
```

```
## 下列物件被遮斷自 'package:caret':
##
##     lift
```

```
scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)

scaled_cluster <- USArrests %>%
    map_df(scaling) %>%
    dist(method = 'euclidean') %>%
    hclust(method = 'complete')

scaled_cluster$labels <- row.names(USArrests)[scaled_cluster$order]
scaled_cluster$labels
```

```
##  [1] "South Dakota"   "West Virginia"  "North Dakota"   "Vermont"
##  [5] "Maine"          "Iowa"           "New Hampshire"  "Idaho"
##  [9] "Montana"        "Nebraska"       "Kentucky"       "Arkansas"
## [13] "Virginia"       "Wyoming"        "Missouri"       "Oregon"
## [17] "Washington"     "Delaware"       "Rhode Island"   "Massachusetts"
## [21] "New Jersey"     "Connecticut"    "Minnesota"      "Wisconsin"
## [25] "Oklahoma"       "Indiana"        "Kansas"         "Ohio"
## [29] "Pennsylvania"   "Hawaii"         "Utah"           "Colorado"
## [33] "California"     "Nevada"         "Florida"        "Texas"
## [37] "Illinois"       "New York"       "Arizona"        "Michigan"
## [41] "Maryland"       "New Mexico"     "Alaska"         "Alabama"
## [45] "Louisiana"      "Georgia"        "Tennessee"      "North Carolina"
## [49] "Mississippi"    "South Carolina"
```

d. What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
# Scaling the variables ensures that each variable contributes equally to the distance calcul
ations
# Without scaling, there might be dominating variables, and the clustering results might be s
kewed towards variables with    larger magnitudes.
# We should scale the variables before we compute the euclidean distances
# So that there will be no dominating variables, and the clustering process will be more inte
rpretable, as it removes the bias introduced by the scale of the variables.
```