ex4

Christopher Huong

2023-07-23

```r
library(mlbench)
library(caret)
```

```r
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
## We convert the 'x' data from a matrix to a data frame
## One reason is that this will give the columns names.
trainingData$x <- data.frame(trainingData$x)

## Look at the data using
## featurePlot(trainingData$x, trainingData$y)
## or other methods.

## This creates a list with a vector 'y' and a matrix
## of predictors 'x'. Also simulate a large test set to
## estimate the true error rate with good precision:
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)
```

Tune several models on these data. For example:

```r
set.seed(921)
knnModel <- train(x = trainingData$x,
  y = trainingData$y,
  method = "knn",
  preProc = c("center", "scale"),
  tuneLength = 10)

knnModel
```

```
## k-Nearest Neighbors
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
```

```
##    5  3.554898  0.4957356  2.877577
##    7  3.411610  0.5428199  2.748739
##    9  3.373158  0.5632258  2.709284
##   11  3.361960  0.5751777  2.699730
##   13  3.349038  0.5924464  2.692112
##   15  3.345493  0.6066887  2.697997
##   17  3.328564  0.6240135  2.687295
##   19  3.326839  0.6362744  2.690267
##   21  3.345087  0.6399141  2.707432
##   23  3.346978  0.6491588  2.710971
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 19.
```

```r
knnPred <- predict(knnModel, newdata = testData$x)

## The function 'postResample' can be used to get the test set
## perforamnce values
postResample(pred = knnPred, obs = testData$y)
```

```
##      RMSE  Rsquared       MAE
## 3.2286834 0.6871735 2.5939727
```

KNN test performance RMSE = 3.23, R2 = 0.69

Train SVM

```r
set.seed(111)
svm <- train(x=trainingData$x, y=trainingData$y,
             method = "svmRadial",
             preProc = c("center", "scale"),
             tuneLength=10)
# save(svm, file='svm.RData')
# load('svm.RData')

svm
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...
## Resampling results across tuning parameters:
##
##   C       RMSE      Rsquared   MAE
##     0.25  2.607382  0.7794330  2.047280
##     0.50  2.348922  0.7973097  1.839771
##     1.00  2.202259  0.8134827  1.723844
##     2.00  2.124218  0.8223866  1.660595
##     4.00  2.091074  0.8263118  1.636061
```

```
##     8.00  2.080000  0.8281388  1.631033
##    16.00  2.078276  0.8283645  1.630556
##    32.00  2.078276  0.8283645  1.630556
##    64.00  2.078276  0.8283645  1.630556
##   128.00  2.078276  0.8283645  1.630556
##
## Tuning parameter 'sigma' was held constant at a value of 0.06452729
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.06452729 and C = 16.
```

```r
svmpred <- predict(svm, newdata = testData$x)

postResample(pred = svmpred, obs = testData$y)
```

```
##      RMSE  Rsquared       MAE
## 2.0774147 0.8250716 1.5781103
```

SVM test performance RMSE = 2.08, R2 = 0.83

Train MARS

```r
set.seed(111)
mars <- train(x=trainingData$x, y=trainingData$y,
             method = "earth",
             preProc = c("center", "scale"),
             tuneGrid = expand.grid(degree = 1:2, nprune = seq(2,14,by=2)))
```

```
## Loading required package: earth

## Warning: package 'earth' was built under R version 4.2.3

## Loading required package: Formula

## Loading required package: plotmo

## Warning: package 'plotmo' was built under R version 4.2.3

## Loading required package: plotrix

## Loading required package: TeachingDemos

## Warning: package 'TeachingDemos' was built under R version 4.2.3
```

```r
save(mars, file='mars.RData')
load('mars.RData')


mars
```

```
## Multivariate Adaptive Regression Spline
##
## 200 samples
##  10 predictor
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 200, 200, 200, 200, 200, 200, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE      Rsquared   MAE
##   1        2      4.466096  0.2176708  3.681105
##   1        4      2.795955  0.6913984  2.208750
##   1        6      2.379241  0.7751782  1.893483
##   1        8      1.834890  0.8661336  1.456156
##   1       10      1.769366  0.8754725  1.386847
##   1       12      1.762974  0.8766241  1.377984
##   1       14      1.799064  0.8719125  1.410912
##   2        2      4.451476  0.2225173  3.670933
##   2        4      2.794891  0.6909875  2.195961
##   2        6      2.367890  0.7762309  1.879506
##   2        8      1.820892  0.8674435  1.444558
##   2       10      1.587148  0.8992738  1.257393
##   2       12      1.511719  0.9087747  1.195093
##   2       14      1.531998  0.9074012  1.209270
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 12 and degree = 2.
```

```r
marspred <- predict(mars, newdata = testData$x)

postResample(pred = marspred, obs = testData$y)
```

```
##      RMSE  Rsquared       MAE
## 1.3227340 0.9291489 1.0524686
```

MARS test performance RMSE = 1.32, R2 = 0.93

Check which predicts were in important in the model

```r
varImp(mars)
```

```
## earth variable importance
##
##    Overall
## X1  100.00
## X4   75.40
## X2   49.00
## X5   15.72
## X3    0.00
```

The MARS model gives the best performance and did indeed choose the best predictors (X1,X4,X2,X5)