Exercise 4 R Markdown

Santanu Mukherjee

07/23/2022

R Markdown

7.2

7.2. **Friedman (1991)** introduced several benchmark data sets create by simulation.One of these simulations used the following nonlinear equation to create data:

y = 10 sin($\pi x_1 x_2$) + 20$(x_3 - 0.5)^2$ + 10x_4 + 5x_5 + N(0, σ^2)

where the x values are random variables uniformly distributed between [0, 1] (there are also 5 other non-informative variables also created in the simulation). The package **mlbench** contains a function called **mlbench.friedman1** that simulates these data:

Which models appear to give the best performance? Does MARS select the informative predictors (those named X1–X5)?

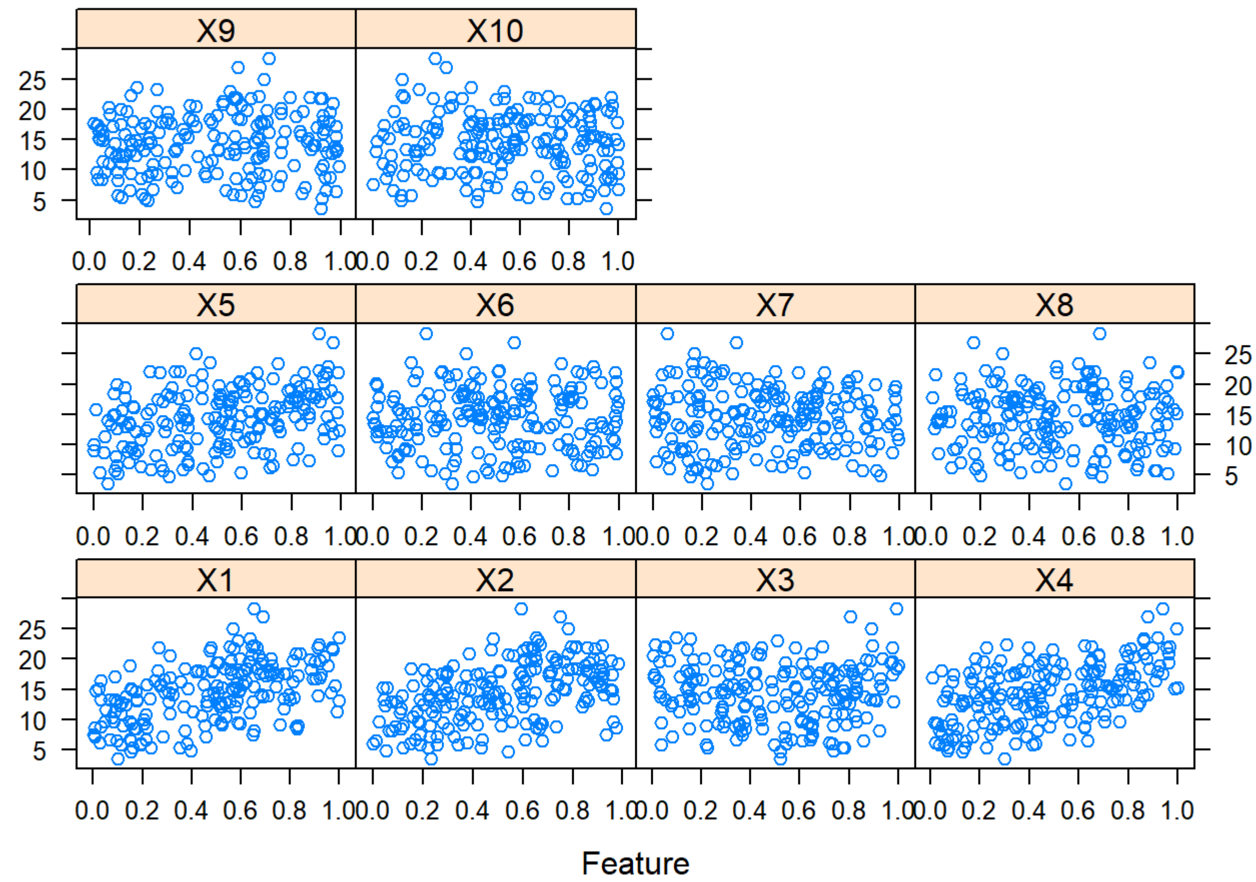
```
library(mlbench)
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)

## 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:

KNN Model

```r
library(caret)
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  3.466085  0.5121775  2.816838
##    7  3.349428  0.5452823  2.727410
##    9  3.264276  0.5785990  2.660026
##   11  3.214216  0.6024244  2.603767
##   13  3.196510  0.6176570  2.591935
##   15  3.184173  0.6305506  2.577482
##   17  3.183130  0.6425367  2.567787
##   19  3.198752  0.6483184  2.592683
##   21  3.188993  0.6611428  2.588787
##   23  3.200458  0.6638353  2.604529
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 17.
```

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

## The function 'postResample' can be used to get the test set
## performance values

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

```
##      RMSE  Rsquared       MAE
## 3.2040595 0.6819919 2.5683461
```

```
knnPR = postResample(pred=knnPred, obs=testData$y)
rmses = c(knnPR[1])
r2s = c(knnPR[2])
methods = c("KNN")
```

Neural Net (NN) Model

```
#
nn.grid = expand.grid( .decay=c(0,0.01,0.1), .size=1:10, .bag=FALSE )
set.seed(0)
nnet.model = train(x=trainingData$x, y=trainingData$y, method="nnet", preProc=c("center", "scale"),
                  linout=TRUE,trace=FALSE,MaxNWts=10 * (ncol(trainingData$x)+1) + 10 + 1, maxit=500)

nnet.pred = predict(nnet.model, newdata=testData$x)
nnet.pr = postResample(pred=nnet.pred, obs=testData$y)
rmses = c(rmses,nnet.pr[1])
r2s = c(r2s,nnet.pr[2])
methods = c(methods,"NN")
```

MARS (Multivariate Adaptive Regression Splines) Model

```
#
mars.grid = expand.grid(.degree=1:2, .nprune=2:38)
set.seed(0)
mars.model = train(x=trainingData$x, y=trainingData$y, method="earth", preProc=c("center", "scale"), tuneGrid=mars.grid)

mars.pred = predict(mars.model, newdata=testData$x)
marsPR = postResample(pred=mars.pred, obs=testData$y)
rmses = c(rmses,marsPR[1])
r2s = c(r2s,marsPR[2])
methods = c(methods,"MARS")
```

Support Vector Machine

```
#
set.seed(0)
svm.model = train(x=trainingData$x, y=trainingData$y, method="svmRadial", preProc=c("center", "scale"), tuneLength=20)

svm.pred = predict(svm.model, newdata=testData$x)
svmPR = postResample(pred=svm.pred, obs=testData$y)
rmses = c(rmses,svmPR[1])
r2s = c(r2s,svmPR[2])
methods = c(methods,"SVM")
```

Final results from all the models

```
#
res = data.frame( rmse=rmses, r2=r2s )
rownames(res) = methods

# Order the dataframe so that the best results are at the bottom:
#
res = res[ order( -res$rmse ), ]
print( "Final Results from all the models" )
```

```
## [1] "Final Results from all the models"
```

```
print( res )
```

```
##          rmse        r2
## KNN  3.204059 0.6819919
## NN   2.649316 0.7177210
## SVM  2.059719 0.8279547
## MARS 1.322734 0.9291489
```

After tuning several models, **MARS model** appear to give the best performance with the **highest** R^2 and **lowest** $RMSE$.

Variable importance for MARS Model

```r
# Lets see what variables are most important for MARS model:
varImp(mars.model)
```

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

The above table shows the importance of the variables (from the most important to the least important) for MARS model.