DATA 624: PREDICTIVE ANALYTICS HW 8

Gabriel Campos

Last edited April 07, 2024

Library

```
library(AppliedPredictiveModeling)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(earth)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Loading required package: plotrix
```

```
library(fpp3)
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.5 --
```

```
## v tibble      3.2.1     v tsibbledata 0.4.1
## v dplyr       1.1.4     v feasts      0.3.1
## v tidyr       1.3.1     v fable       0.3.3
## v lubridate   1.9.3     v fabletools  0.4.0
## v tsibble     1.1.4
```

```
## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
```

```
## x dplyr::lag()         masks stats::lag()
## x fabletools::MAE()    masks caret::MAE()
## x fabletools::RMSE()   masks caret::RMSE()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
library(dplyr)
library(ggplot2)
library(mlbench)
library(RANN)
library(tidyr)
```

Description

Do problems 7.2 and 7.5 in Kuhn and Johnson. There are only two but they have many parts. Please submit both a link to your Rpubs and the .rmd file.

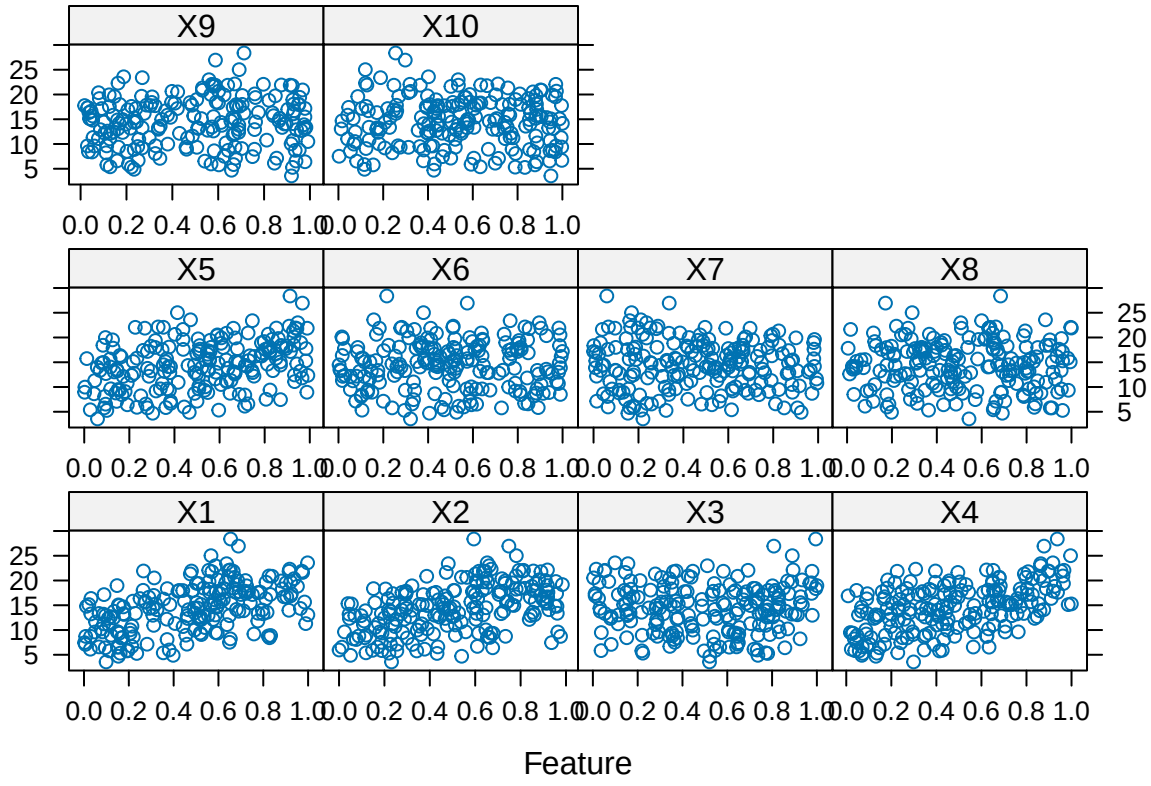
7.2

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

$y = 10sin(\pi 1 x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0, \sigma^2)$

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:

```
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)
```

i

Example

Tune several models on these data. For example:

```
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.
```

RMSE was used to select the optimal model using the smallest value. The final value used for the model was k = 19.

```
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
## 3.2040595 0.6819919 2.5683461
```

MARS & MARS TUNED

```
mars_Grid <- expand.grid(.degree = 1:2, .nprune = 2:38)
set.seed(100)

mars_Fit <- earth(trainingData$x, trainingData$y)

mars_Pred <- predict(mars_Fit, newdata = testData$x)
postResample(pred = mars_Pred, obs = testData$y)
```

MARS

```
##      RMSE  Rsquared       MAE
## 1.8136467 0.8677298 1.3911836
```

```
marsTuned_Grid <- expand.grid(.degree = 1:2, .nprune = 2:38)
set.seed(100)

marsTuned <- train(trainingData$x, trainingData$y, method = "earth",
                   tuneGrid = marsTuned_Grid,
                   trControl = trainControl(method = "cv"))

marsTune_Pred <- predict(marsTuned, newdata = testData$x)
postResample(pred = marsTune_Pred, obs = testData$y)
```

MARS Tuned

```
##      RMSE  Rsquared       MAE
## 1.1589948 0.9460418 0.9250230
```

SVM Tuned

```
svmRTuned <- train(trainingData$x, trainingData$y,
                   method = "svmRadial",
                   preProcess = c("center", "scale"),
                   tuneLength = 15,
                   trControl = trainControl(method = "cv"))

svmPred <- predict(svmRTuned, newdata = testData$x)
postResample(svmPred, testData$y)
```

```
##      RMSE  Rsquared       MAE
## 2.0741473 0.8255848 1.5755185
```

ii

Which models appear to give the best performance?

MAR tuned model with an R-squared value of 0.9460 looks like its the best performing.

iii

Does MARS select the informative predictors (those named X1–X5)?

Yes it selects for informative predictors

7.5

Exercise 6.3 describes data for a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several nonlinear regression models.

Load Data

```
data(ChemicalManufacturingProcess)
```

data imputation, data splitting & pre-processing

impute

```
imputed_df <- preProcess(ChemicalManufacturingProcess, "knnImpute")
imputed_full_df <- predict(imputed_df, ChemicalManufacturingProcess)
```

```
val_low <- nearZeroVar(imputed_full_df)
#remove low frequency columns using baser df[row,columns]
chem_df <- imputed_full_df[,-val_low]
```

split

```
chem_index <- createDataPartition(chem_df$Yield , p=.8, list=F)
```

```
chem_train <-  chem_df[chem_index,]
chem_test <- chem_df[-chem_index,]
```

```
chem_marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)
```

(a)

Which nonlinear regression model gives the optimal resampling and test set performance?

MARS Tuned

```
chem_MarsTuned <- train(Yield~. ,
                  data = chem_train,
                   method = "earth",
                   tuneGrid = chem_marsGrid,
                   trControl = trainControl(method = "cv"))

chem_MarsTunePred <- predict(chem_MarsTuned,  chem_test)
postResample(chem_MarsTunePred, chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 0.6437705 0.5394597 0.4739508
```

SVM Tuned

```
chem_SVMTuned <- train(Yield~. ,
                   data = chem_test,
                    method = "svmRadial",
                    tuneLength = 15,
                    trControl = trainControl(method = "cv"))

chem_SVMTunePred <- predict(chem_SVMTuned,  chem_test)
postResample(chem_SVMTunePred, chem_test$Yield)
```

```
##       RMSE   Rsquared        MAE
## 0.09178983 0.99520810 0.09087154
```

KNN

```
knnModel <- train(Yield~.,
                     data = chem_test,
                     method = "knn",
                     preProc = c("center", "scale"),
                     tuneLength = 10)

knnPred <- predict(knnModel,  chem_test)
postResample(pred = knnPred, obs = chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 0.7631649 0.4717384 0.6279725
```

Neural Network

```
nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
                        .size = c(1:10), .bag = FALSE)

nnetTune <- train(Yield~.,
                   data = chem_train,
                   method = "avNNet",
                   tuneGrid = nnetGrid,
                   trControl = trainControl(method = "cv"),
                   linout = TRUE,trace = FALSE,
                   MaxNWts = 10 * (ncol(chem_train) + 1) + 10 + 1,
                   maxit = 500)
```

```
nnetPred <- predict(nnetTune,  chem_test)
postResample(predict(nnetTune,  chem_test), chem_test$Yield)
```

```
##      RMSE  Rsquared       MAE
## 0.6338060 0.6117156 0.4841039
```

The SVM tuned is the best performing RMSE and R squared with values of RMSE=0.09178983 , and R squared=0.99520810

(b)

Which predictors are most important in the optimal nonlinear regression model? Do either the biological or process variables dominate the list?How do the top ten important predictors compare to the top ten predictors from the optimal linear model?

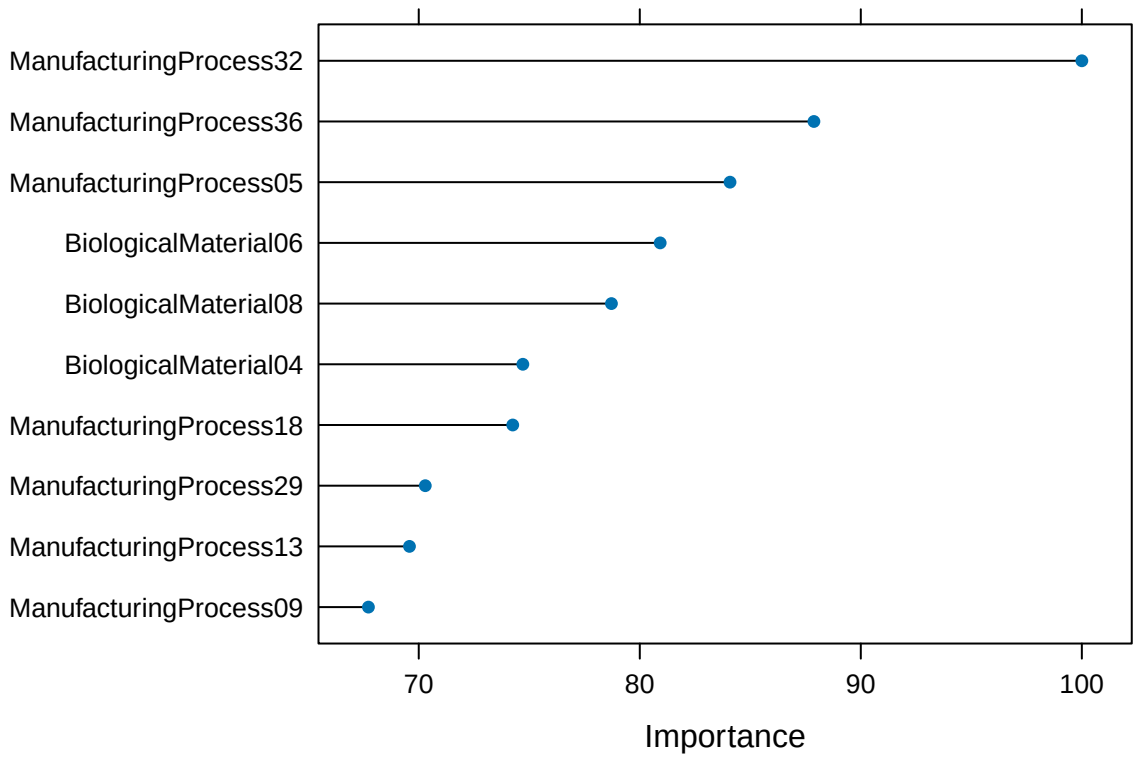
Of the 20 `ManufacturingProcess` predictors 12 are `BiologicalMaterial` and 8 are `ManufacturingProcess`. In the top 10, 6 are `ManufacturingProcess` in with 1rst through 3rd being `ManufacturingProcess32`, `ManufacturingProcess36`, and `ManufacturingProcess05`. Although neither dominates, Manufacturing-Process does hold a greater influence overall, when compared to the ratio of optimal nonlinear vs optimal linear, in optimal linear, ManuracturingProcess importance is greater.

```
plot(varImp(chem_SVMTuned), top=10)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 0.019411
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 0.44646
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 1.669e-16
```



(c)

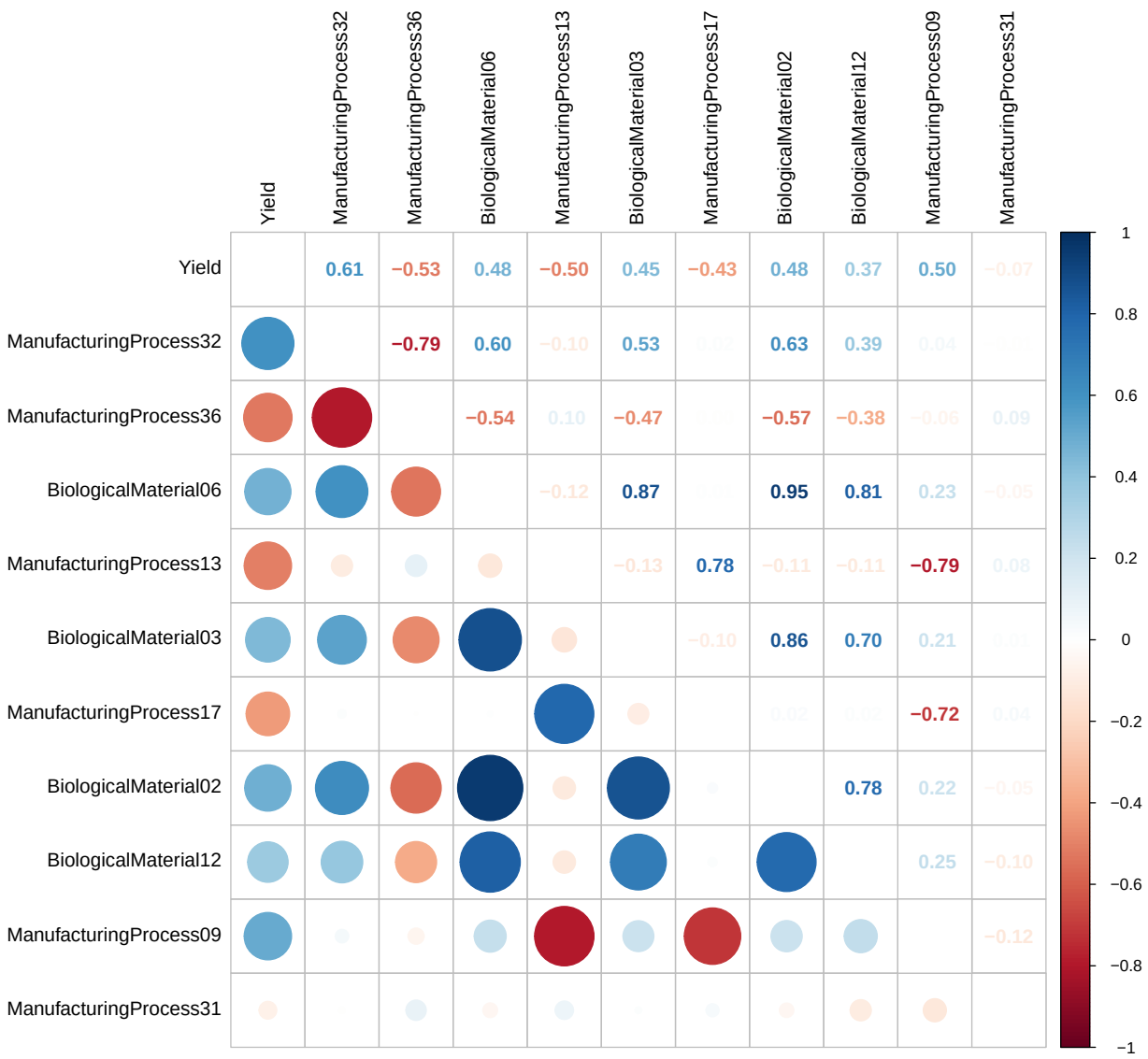
Explore the relationships between the top predictors and the response for the predictors that are unique to the optimal nonlinear regression model. Do these plots reveal intuition about the biological or process predictors and their relationship with yield?

- Corrplot used to explore relationship
- Yield response variable doesn't have strong correlations with most of the important predictors noted previously.
- Strongest positive correlations exist with `ManufacturingProcess32` and `ManufacturingProcess09` (0.61 and 0.50).
- The strongest negative correlations exist with `ManufacturingProcess36` and `ManufacturingProcess13` (-0.53 and -0.50, respectively).
- No strong correlations with Yeild and the important Biological Material predictors identified in the previous step.

```
corr_vals <- chem_df %>%
  dplyr::select('Yield', 'ManufacturingProcess32','ManufacturingProcess36',
         'BiologicalMaterial06','ManufacturingProcess13',
         'BiologicalMaterial03','ManufacturingProcess17',
         'BiologicalMaterial02','BiologicalMaterial12',
         'ManufacturingProcess09','ManufacturingProcess31')

corr_plot_vals <- cor(corr_vals)

corrplot.mixed(corr_plot_vals, tl.col = 'black', tl.pos = 'lt',
         upper = "number", lower="circle")
```


Yield
ManufacturingProcess32
ManufacturingProcess36
BiologicalMaterial06
ManufacturingProcess13
BiologicalMaterial03
ManufacturingProcess17
BiologicalMaterial02
BiologicalMaterial12
ManufacturingProcess09
ManufacturingProcess31
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1