KJ Chapter 6 Problem 3

Jonathan Hernandez

March 5, 2019

6.3. A chemical manufacturing process for a pharmaceutical product was discussed in Sect. 1.4. In this problem, the objective is to understand the relationship between biological measurements of the raw materials (predictors), measurements of the manufacturing process (predictors), and the response of product yield. Biological predictors cannot be changed but can be used to assess the quality of the raw material before processing. On the other hand, manufacturing process predictors can be changed in the manufacturing process. Improving product yield by 1% will boost revenue by approximately one hundred thousand dollars per batch:
(a) Start R and use these commands to load the data:
> library(AppliedPredictiveModeling)
> data(chemicalManufacturing)

The matrix processPredictors contains the 57 predictors (12 describing the input biological material and 45 describing the process predictors) for the 176 manufacturing runs. yield contains the percent yield for each run.

(b) A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in these missing values (e.g., see Sect. 3.8).

(c) Split the data into a training and a test set, pre-process the data, and tune a model of your choice from this chapter. What is the optimal value of the performance metric?

(d) Predict the response for the test set. What is the value of the performance metric and how does this compare with the resampled performance metric on the training set?

(e) Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list?

(f) Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in future runs of the manufacturing process?

- For (a), let's begin by importing the libraries in question (and any other libraries) and examine the data in detail:

```
library(AppliedPredictiveModeling)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

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

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

```
library(e1071)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
data(ChemicalManufacturingProcess)
#summary(ChemicalManufacturingProcess)
```

- For (b), we will replace NA value with the median using the built-in function impute() from the 'e1071' package. Replacing missing values with the median rather than the mean because the median is not a bias statistic and varies.
- Also, let's check for zero variance (while using the train() function in caret package, R reported some variables had zero variance which in this case, we'll remove then apply training and test sets.

```
# Split into X and y matrix and vector respectively
Chem_yield <- ChemicalManufacturingProcess$Yield # y vector (response)
Chem_predictors <- ChemicalManufacturingProcess[, 2:58] # X matrix (explanatory vars)
Chem_predictors <- impute(Chem_predictors) # apply median to NA values for predictors
sum(!complete.cases(Chem_predictors))
```

```
## [1] 0
```

```
# when evaluating lm model using train() for the first time,
# R complained that there were some
# predictors with zero variance, let's see if there is a way to remove them
zero_var <- nearZeroVar(Chem_predictors)
str(zero_var) # output will show which column has the zero variance
```

```
##  int 7
```

```r
# extract features that don't have zero variance in the training set
Chem_predictors <- Chem_predictors[, -zero_var]
```

- (c) First, start by splitting the training and test set using a 70/30 rule: 70% of the random observations go into the training set and the remaining 30% go into the test set. Use different regression techniques like linear regression, ridge regression and LASSO regression and enet

```r
set.seed(123) # set seed for reproducible result
# create the data partition 70/30 training/test set
# data_split contains the row indicies to use in the training set, the rest in
# the test set
data_split <- createDataPartition(Chem_yield, p = 0.7)
#data_split

# Training set X and y
chem_train_X <- Chem_predictors[data_split$Resample1, ]
chem_train_y <- Chem_yield[data_split$Resample1]

# Test set X and y
chem_test_X <- Chem_predictors[-data_split$Resample1, ]
chem_test_y <- Chem_yield[-data_split$Resample1]

# store as matricies the training data and center the response variable
chem_train_X <- chem_train_X %>% as.matrix()
chem_train_y <- chem_train_y %>% scale(center = TRUE, scale = FALSE) %>% as.matrix()

# Ridge Regression (penalization L_2)
# get optimal lambda to minimize the MSE
chem_ridge_lambda <- cv.glmnet(chem_train_X, chem_train_y, alpha = 0, standardize = TRUE)

# now fit ridge regression model using optimal lambda
chem_ridge_model <- glmnet(chem_train_X, chem_train_y, alpha = 0, standardize = TRUE,
                           lambda = chem_ridge_lambda$lambda.min)

chem_lasso_lambda <- cv.glmnet(chem_train_X, chem_train_y, alpha = 1, standardize = TRUE)

# now fit LASSO regression model using optimal lambda
chem_lasso_model <- glmnet(chem_train_X, chem_train_y, alpha = 1, standardize = TRUE,
                           lambda = chem_lasso_lambda$lambda.min)

# Enet regression
enet_control_lambda <- cv.glmnet(chem_train_X, chem_train_y, alpha = 0.5, standardize = TRUE)

# now fit enet regression model using optimal lambda
chem_enet_model <- glmnet(chem_train_X, chem_train_y, alpha = 0.5, standardize = TRUE,
                           lambda = enet_control_lambda$lambda.min)
```

- There were 3 models (regularization) that were applied. I decided to use the MSE as a metric for picking the best model to fit the data. Looking at the λ values that minimize the MSE for each model, they turned out to be the following. And it turns out that using LASSO is the winner with the smallest MSE

```
models_lambda <- data.frame(Lambda = c(chem_ridge_lambda$lambda.min,
                                       chem_lasso_lambda$lambda.min,
                                       enet_control_lambda$lambda.min),
                            MSE = c(min(chem_ridge_lambda$cvm),
                                    min(chem_lasso_lambda$cvm),
                                    min(enet_control_lambda$cvm)),
                            ModelName = c("Ridge", "LASSO", "enet"))
models_lambda
```

```
##        Lambda      MSE ModelName
## 1 10.18589242 2.704456     Ridge
## 2  0.09499397 1.501134     LASSO
## 3  0.20851154 1.522729      enet
```

- (d) Let's make predictions using the test data using each of the 3 penalization methods using predict() and compare the MSE's between the test and training sets. Also the R^2 value will also be examined

```
# predicting with the test data for ridge regression
# let's also see the MSE as well
chem_predict_ridge <- predict(chem_ridge_model, newx=chem_test_X)
chem_predict_ridge_mse <- mean((chem_test_y - chem_predict_ridge)^2)

# predicting with the test data for lasso regression
# let's also see the MSE as well
chem_predict_lasso <- predict(chem_lasso_model, newx=chem_test_X)
chem_predict_lasso_mse <- mean((chem_predict_lasso - chem_test_y)^2)

# predicting with the test data for lasso regression
# let's also see the MSE as well
chem_predict_enet <- predict(chem_enet_model, newx=chem_test_X)
chem_predict_enet_mse <- mean((chem_test_y - chem_predict_enet)^2)
```

- Results below

```
pred_test_results <- data.frame(Test_MSE = c(chem_predict_ridge_mse,
                                             chem_predict_lasso_mse,
                                             chem_predict_enet_mse),
                                ModelName = c("Ridge", "LASSO", "enet"))
pred_test_results
```

```
##   Test_MSE ModelName
## 1 1585.496     Ridge
## 2 1595.496     LASSO
## 3 1596.444      enet
```

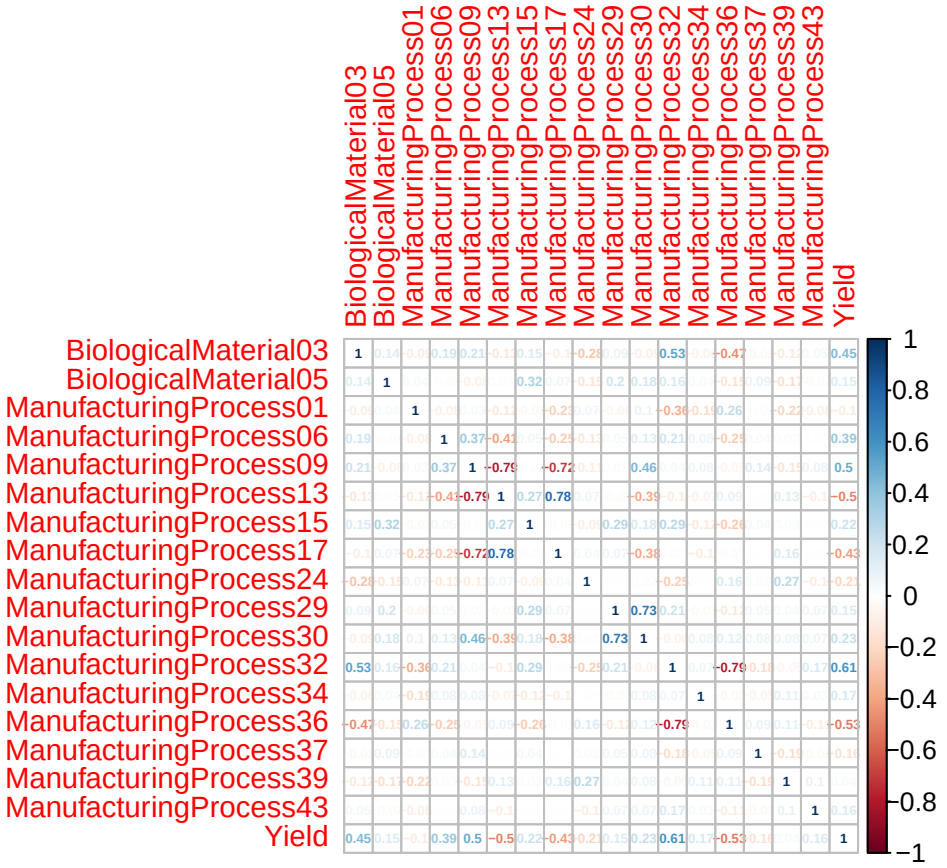
- So we see that the MSE for the test dataset when using our models has a much higher MSE than when we fit the training data with regularization. This may be that even though the regularization models fit the data well, overfitting may have occured for the training data and it cannot recognize very well future data.
- For (e), picking the LASSO model, let's look at the coefficients and see what was left and see how many biological predictors and how many process predictors exist and which has more, biological or process.

```
lasso_coef <- coef(chem_lasso_model)[,1]
lasso_coef_nonzero <- lasso_coef[which(lasso_coef != 0)]
```

- Looking at the coefficients of the LASSO model that are not 0, we see that there are more process predictors that biological predictors (15 process predictors and only 2 biological predictors)
- For (f), let's look at the correlation for all variables as well as compare across different models

```
# first get the names of the predictor we trained for our LASSO model using a
# lambda of about 0.094
vars_lasso <- names(lasso_coef_nonzero)[-1]

# now take those columns and select them from the original dataset and do a correlation
# plot amongst them
corrplot(cor(ChemicalManufacturingProcess[, c(vars_lasso, "Yield")],
             use = "pairwise.complete.obs"), tl.cex = 0.9, method = "number",
         number.cex = 0.4)
```



- Looking at the correlation plot of the variables we selected from the LASSO model, the response variable Yield doesn't have much strong correlation with the other explanatory variables. Variables such as ManufacturingProcess32, 36, 09 and 13 show slightly okay correlation in Yield.
- Let's do a simple linear regression plot of Yield vs ManufacturingProcess32 as that feautre has the highest correlation value.

```r
lm_yield <- lm(Yield ~ ManufacturingProcess32, data = ChemicalManufacturingProcess)
summary(lm_yield)
```

```
##
## Call:
## lm(formula = Yield ~ ManufacturingProcess32, data = ChemicalManufacturingProcess)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0376 -0.9921 -0.0826  0.8849  4.3881
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              7.21119    3.26242    2.21   0.0284 *
## ManufacturingProcess32  0.20803    0.02058   10.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.469 on 174 degrees of freedom
## Multiple R-squared:  0.3701, Adjusted R-squared:  0.3664
## F-statistic: 102.2 on 1 and 174 DF,  p-value: < 2.2e-16
```

- The most correlated variable for Yield shows that manufacturingProcess32 has a well-low p-value for its estimated coefficient but a low R^2 value.
- These variables may be the more better features to use in predicting yield or a more non-linear approach may be required.