Data 624: Homework II

Group Two

Vinicio Haro
Sang Yoon (Andy) Hwang
Julian McEachern
Jeremy O'Brien
Bethany Poulin

16 December 2019

Contents

Dependencies

```r
# Processing
libraries("mice", "AppliedPredictiveModeling", "tidyverse",
    "caret", "pls", "recipes", "corrplot")

# Formatting Libraries

libraries("default", "knitr", "kableExtra")

# Plotting Libraries
libraries("ggplot2", "grid", "ggfortify")
```

Assignment 1

Kuhn and Johnson 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:

```
data("ChemicalManufacturingProcess")
```

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).

There are 28 predictor variables with 106 missing values within the `ChemicalManufacturingProcess` (CMP) dataset. The `mice` function from the `mice` package can be used to impute multivariate missing data. The method applies a unique model to each variable to conduct multiple imputations. After running the function, we apply the `complete` function to fill in the missing data. There are now 0 missing values within the dataset.

(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?

For this question, we decided to use a partial least squares (pls) method to model the data. We created evaluation method using the `createDataPartition` from the `caret` package, where 80% of the CMP data was randomly selected for training and 20% for testing purposes. We built two models to compare the effect pre-processing and tuning had on the fitting the data. Our pre-processing methods involved applying a principal component analysis (pca). In doing so, we recognized one of our variables, `BiologicalMaterial08`, had near-zero variance so we applied a recipe to filter that variable. We also applied training controls to resample the data over 5 folds.

The pre-processing components included the following:

```
Created from 144 samples and 56 variables
```

```
Pre-processing:
  - centered (56)
  - ignored (0)
  - principal component signal extraction (56)
  - scaled (56)

PCA needed 25 components to capture 95 percent of the variance
```

We found that this approach helped improve model accuracy metrics. The optimal value of the performance metric for our model with transformations was 1.53, whereas the optimal value for the model without pre-processing was 1.77.

PLS Model - No Pre-Processing

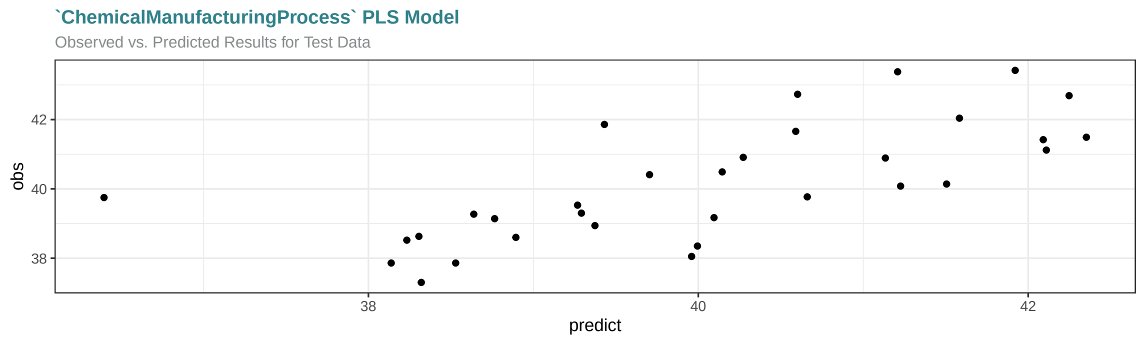
```
Partial Least Squares

144 samples
 57 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 144, 144, 144, 144, 144, 144, ...
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared    MAE
  1      1.771420  0.12809343  1.417675
  2      2.158868  0.11869755  1.496848
  3      3.581090  0.09634691  1.752045
  4      6.930771  0.08060499  2.207104
  5      5.680859  0.15421739  1.915612

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 1.
```

PLS Model - With Pre-Processing

```
Partial Least Squares

144 samples
 56 predictor

Pre-processing: principal component signal extraction (56), centered
 (56), scaled (56)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 114, 116, 116, 115, 115
Resampling results across tuning parameters:
  ncomp  RMSE      Rsquared   MAE
  1      1.519177  0.4249250  1.160602
  2      1.793880  0.4441069  1.153173
  3      1.525966  0.5180790  1.083496
  4      1.475197  0.5362954  1.062071
  5      1.511131  0.5224443  1.068680

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 4.
```

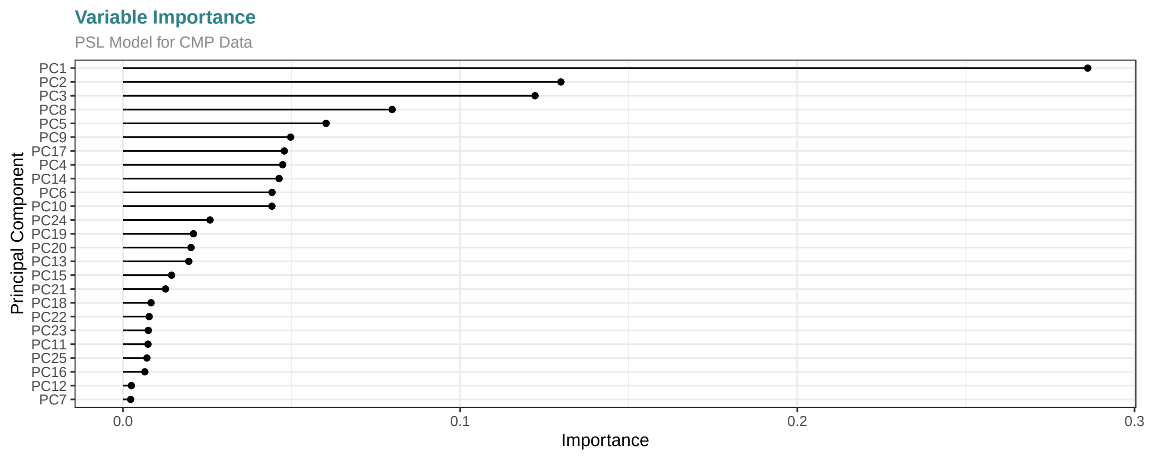
(d). Predict the response for the test set. What is the value of the performance metric and how does this compare with the resampled performance metric on the training set?

The RMSE for the test data set was 1.22, which was slightly higher than when we fitted the model on the train data (RMSE: 1.53). We compared the test predictions with the observed data below:



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

The `varImp` function allows us to see the importance of our model's principal components. Biological predictors make up 8, or 80%, of our top 10 variables. The following plot shows the ranks of importance of our model's principal components:



(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?

Below, we looked at the correlation between our top predictor variables with our response variable:

BiologicalMaterial01	0.3427280
BiologicalMaterial03	0.4827276
BiologicalMaterial02	0.5100752
BiologicalMaterial09	0.1107648
BiologicalMaterial10	0.1691056
BiologicalMaterial12	0.4463825
ManufacturingProcess07	-0.0345109
BiologicalMaterial05	0.1626980
BiologicalMaterial06	0.5315807
ManufacturingProcess03	-0.1325062

Both our top manufacturing predictor variables have a negative correlation coefficient which would be beneficial to examine to improve yield predictions in future models.

R Code

```r
# (6.3a) Load data
data("ChemicalManufacturingProcess")

# (6.3b) Calculate NA Values
CMP_NA <- ChemicalManufacturingProcess %>% select(-Yield) %>%
    summarise_all(funs(sum(is.na(.)))) %>% t() %>%
    as.data.frame() %>% rownames_to_column("Predictor") %>%
    filter(V1 > 0)

# Impute NA Values
CMP_Impute <- mice(ChemicalManufacturingProcess, m = 5,
    printFlag = F)
CMP_DF <- mice::complete(CMP_Impute, 2)

# Verify
CMP_Verify <- sum(is.na(CMP_DF))

# (6.3c) Set random seed
set.seed(50)

# Create Partition for Train/Test Splits
trainingRows <- createDataPartition(CMP_DF$Yield, p = 0.8,
    list = FALSE)

# Split Train/Test Data
train <- CMP_DF[trainingRows, ]
test <- CMP_DF[-trainingRows, ]

# Create model function
fit_pls_1 <- train(Yield ~ ., data = train, method = "pls",
    tuneLength = 5)

# Pre-Process Recipe
rec <- recipes::recipe(CMP_DF, Yield ~ .)
rec <- rec %>% step_nzv(all_predictors(), options = list(freq_cut = 95/5,
    unique_cut = 10))
prep_rec = prep(rec, training = CMP_DF)
CMP_DF_TF = bake(prep_rec, CMP_DF)

# Create Partition for Train/Test Splits
trainingRows <- createDataPartition(CMP_DF_TF$Yield,
    p = 0.8, list = FALSE)

# Split Train/Test Data
train <- CMP_DF_TF[trainingRows, ]
test <- CMP_DF_TF[-trainingRows, ]

# Create model function
fit_pls_2 <- train(Yield ~ ., data = train, method = "pls",
    preProcess = "pca", trControl = trainControl(method = "cv",
        number = 5), tuneLength = 5)
```

```r
RMSE_1 <- fit_pls_1$results$RMSE[1]
RMSE_2 <- fit_pls_2$results$RMSE[3]

# (6.3d) Predict Response
predict <- predict(fit_pls_2, test)

# Performance Metrics
test_predict_result <- data.frame(obs = test$Yield,
    pred = predict)

# summarize results
test_predict_accuracy <- defaultSummary(test_predict_result)
RMSE_3 <- test_predict_accuracy[[1]]

# plot
plot <- ggplot(test_predict_result, aes(obs, predict)) +
    labs(title = "`ChemicalManufacturingProcess` PLS Model",
        subtitle = "Observed vs. Predicted Results for Test Data") +
    geom_point() + coord_flip() + theme_bw() + theme()

# (6.3e)
varimp1 <- varImp(fit_pls_2, scale = F, useModel = T)

varimp2 <- varimp1$importance %>% rownames_to_column("Principal Component")

plot <- ggplot(varimp2, aes(x = reorder(`Principal Component`,
    Overall), y = Overall)) + geom_point() + geom_segment(aes(x = `Principal Component`,
    xend = `Principal Component`, y = 0, yend = Overall)) +
    labs(title = "Variable Importance", subtitle = "PSL Model for CMP Data",
        x = "Principal Component", y = "Importance") +
    coord_flip() + theme_bw() + theme()

varimp_pc <- train %>% select(-Yield) %>% select(1,
    3, 2, 8, 9, 11, 18, 5, 6, 14)
bio <- varimp_pc %>% select(starts_with("Bio")) %>%
    ncol()

# (6.3f)
top <- varimp_pc %>% colnames()
cor <- cor(train[, top], train$Yield, method = "pearson")
```