Team 2 - Homework Two

Assignment 2: KJ 7.2; KJ 7.5

Juliann McEachern

10/23/19

Dependencies

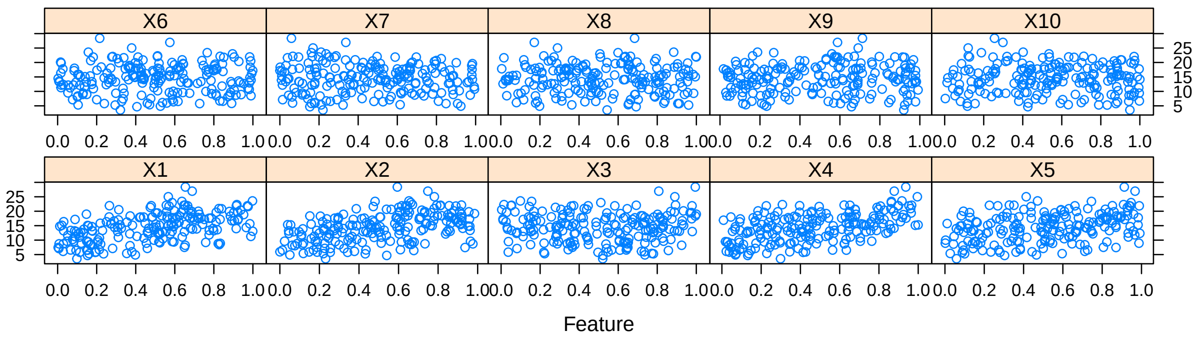
```r
# predictive modeling
libraries("mlbench", "caret", "mice", "AppliedPredictiveModeling",
    "recipes", "tibble", "tidyverse")

# Formatting Libraries
libraries("default", "knitr", "kableExtra")

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

(1) Kuhn & Johnson 7.2

Friedman (1991) introduced several benchmark data sets create by simulation. One of these simulations used the following nonlinear equation to create data: $y = 10\text{sin}(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0, \sigma^2)$; 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:

```r
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
trainingData$x <- data.frame(trainingData$x)
featurePlot(trainingData$x, trainingData$y)
```



```r
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)
```

(a) Tune several models on these data.

For example:

Train set model & performance:

```
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  3.533813  0.5130609  2.910827
   7  3.429971  0.5461330  2.818732
   9  3.401852  0.5637178  2.775645
  11  3.338443  0.5938918  2.728206
  13  3.315336  0.6142508  2.700334
  15  3.310544  0.6284303  2.697967
  17  3.306122  0.6423117  2.698210
  19  3.323482  0.6487590  2.718881
  21  3.327365  0.6585681  2.718610
  23  3.335849  0.6635003  2.725054

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 17.
```

Test set performance values:

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

Model 1:

Train set model & performance:

```
Linear Regression

200 samples
 10 predictor

Pre-processing: principal component signal extraction (10), centered
 (10), scaled (10)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
Resampling results:

  RMSE      Rsquared   MAE
  2.432147  0.7653831  1.951699

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Test set performance values:

```
     RMSE  Rsquared       MAE 
2.6970680 0.7084666 2.0600540 
```

Model 2:

Train set model & performance:

```
Partial Least Squares 

200 samples
 10 predictor

Pre-processing: principal component signal extraction (10), centered
 (10), scaled (10) 
Resampling: Cross-Validated (10 fold, repeated 5 times) 
Summary of sample sizes: 180, 180, 180, 180, 180, 180, ... 
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared   MAE     
  1      2.522945  0.7537245  2.006956
  2      2.421026  0.7743143  1.930347
  3      2.425394  0.7738312  1.934872
  4      2.426279  0.7735851  1.936843
  5      2.426467  0.7736047  1.937165

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 2.
```

Test set performance values:

```
    RMSE Rsquared      MAE 
2.685591 0.710292 2.052676 
```

Model 3:

Train set model & performance:

```
Multivariate Adaptive Regression Spline 

200 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times) 
Summary of sample sizes: 180, 180, 180, 180, 180, 180, ... 
Resampling results:

  RMSE      Rsquared   MAE     
  1.681637  0.8915313  1.318973
```

```
Tuning parameter 'nprune' was held constant at a value of 10

Tuning parameter 'degree' was held constant at a value of 1
```

Test set performance values:

```
    RMSE Rsquared      MAE 
1.776575 0.872700 1.358367 
```

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

The MARS model has the lowest RMSE accuracy scores for both our training and test sets. This model appeared to give the best performance.

Table 1: Model Performance

	RMSE	**RSquared**	**MAE**
knnTrain	3.3105	3.3105	3.3105
knnTest	3.2041	0.6820	2.5683
lmTrain	2.4321	0.7654	1.9517
lmTest	2.6971	0.7085	2.0601
plsTrain	2.4210	0.7743	1.9303
plsTest	2.6856	0.7103	2.0527
marsTrain	**1.6816**	**0.8915**	**1.3190**
marsTest	**1.7766**	**0.8727**	**1.3584**

In addition, the MARS model selected the important indicator variables: X1-X5.

Table 2: MARS Model - Variable Importance

	Overall
X1	100.00
X4	84.07
X2	66.86
X5	44.68
X3	33.51
X6	7.48
X7	0.00
X8	0.00
X9	0.00
X10	0.00

(2) Kuhn & Johnson 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.

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

We trained four models on the chemical manufacturing process data: Support Vector Machines with Radial Basis Function Kernel (SVM), Bayesian Ridge Regression (Model Averaged), k-Nearest Neighbors (KNN), and Multivariate Adaptive Regression Spline (MARS).

We found that the SVM and Bayesian Ridge approach produced the lowest train accuracy score. However, he train accuracy for the Baysian Ridge was lower than the test accuracy, indicating that model may have been overfitted to the training data. MARS also produced similiarly low accuracy scores. MARS test RMSE was slightly lower than SVM's test accuracy measures, however SVM outperformed MARS with the training accuracy. There was a smaller difference between the train and test accuracy with the SVM method, thus we choose this as our optimal model for resampling and test set performance.

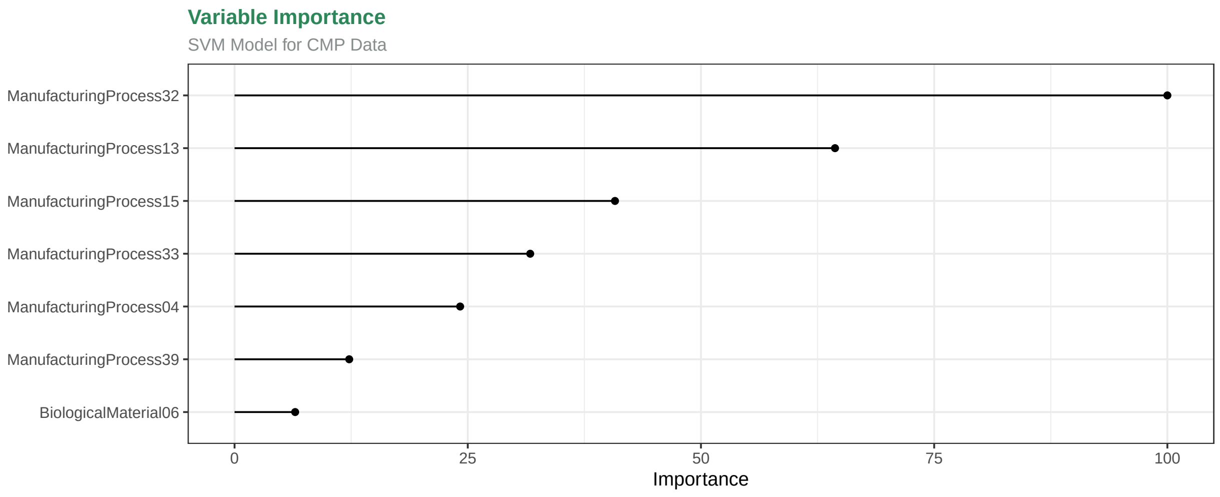
Table 3: Model Performance

	RMSE	Rsquared	MAE
svmTrain	**1.1598**	**0.6344**	**0.9360**
svmTest	**1.0019**	**0.6278**	**0.7822**
knnTrain	1.5595	1.5595	1.5595
knnTest	1.5650	0.1221	1.2830
bma_rrTrain	1.1676	0.6212	0.9559
bma_rrTest	1.4768	0.3817	0.9591
marsTrain	1.2477	0.5982	1.0095
marsTest	0.9359	0.6672	0.7253

(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?

The `caret` package does not allow the `varImp` function to work on SVM models, thus we used MARS to evaluate predictor importance of our optimal nonlinear regression models as these MARS ans SVM had similiar accuracy performance.

In our homework from Chapter 6, our PLS linear model identified mostly biological process variables as the important predictors. This differs from the non-linear important predictors identified in our MARS model. The MARS method calculated variable importance for 7 indicators, of which 6 were manufacturing variables.



(c) Explore the relationships between the top predictors and the response for the predictors that are unique to the optimal nonlinear regression model.

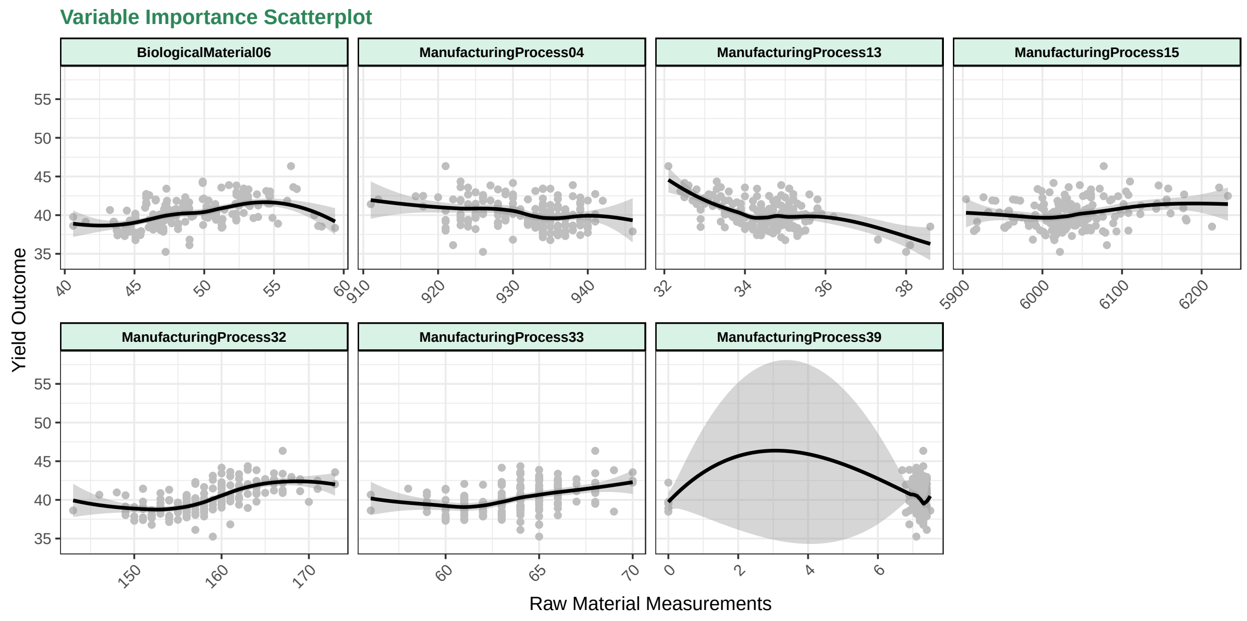
Do these plots reveal intuition about the biological or process predictors and their relationship with yield?

The following shows correlation between our top predictor and response variables using our MARS model.

Table 4: Correlation

Variable	Correlation
ManufacturingProcess32	0.6061176
BiologicalMaterial06	0.4766767
ManufacturingProcess33	0.4130360
ManufacturingProcess15	0.2888848
ManufacturingProcess39	0.0420715
ManufacturingProcess04	-0.2424601
ManufacturingProcess13	-0.5280717

We can use a scatterplot to further look at their relationship. None of the variables exibit a strong linear pattern when examined against yield. This depiction could help explain why variable importance results from our selected linear model varied greatly from our non-linear model.



R Code

```
# (7.2 example)
set.seed(200)
trainingData <- mlbench.friedman1(200, sd = 1)
trainingData$x <- data.frame(trainingData$x)
featurePlot(trainingData$x, trainingData$y)
testData <- mlbench.friedman1(5000, sd = 1)
testData$x <- data.frame(testData$x)
knnModel <- train(x = trainingData$x, y = trainingData$y,
    method = "knn", preProc = c("center", "scale"),
```

```r
    tuneLength = 10)
knnPred <- predict(knnModel, newdata = testData$x)
knnPerf <- postResample(pred = knnPred, obs = testData$y)

# (7.2a) Model 1
lmModel <- train(x = trainingData$x, y = trainingData$y,
    method = "lm", preProc = "pca", trControl = trainControl(method = "repeatedcv",
        repeats = 5))
lmPred <- predict(lmModel, newdata = testData$x)
lmPerf <- postResample(pred = lmPred, obs = testData$y)

# Model 2
plsModel <- train(x = trainingData$x, y = trainingData$y,
    method = "pls", preProc = "pca", tuneLength = 5,
    trControl = trainControl(method = "repeatedcv",
        repeats = 5))
plsPred <- predict(plsModel, newdata = testData$x)
plsPerf <- postResample(pred = plsPred, obs = testData$y)

# Model 3
marsModel <- train(x = trainingData$x, y = trainingData$y,
    method = "earth", tuneGrid = expand.grid(degree = 1,
        nprune = 10), trControl = trainControl(method = "repeatedcv",
        repeats = 5))
marsPred <- predict(marsModel, newdata = testData$x)
marsPerf <- postResample(pred = marsPred, obs = testData$y)

# (7.2b)
performance_table <- rbind(knnTrain = c(RMSE = knnModel$results$RMSE[6],
    RSquared = knnModel$results$RMSE[6], MAE = knnModel$results$RMSE[6]),
    knnTest = knnPerf, lmTrain = c(RMSE = lmModel$results$RMSE,
        RSquared = lmModel$results$Rsquared, MAE = lmModel$results$MAE),
    lmTest = lmPerf, plsTrain = c(plsModel$results$RMSE[2],
        plsModel$results$Rsquared[2], plsModel$results$MAE[2]),
    plsTest = plsPerf, marsTrain = c(marsModel$results$RMSE,
        marsModel$results$Rsquared, marsModel$results$MAE),
    marsTest = marsPerf) %>% kable(caption = "Model Performance",
    digits = 4) %>% kable_styling() %>% row_spec() %>%
    row_spec(row = 7:8, background = "#d9f2e6")

marsImp <- varImp(marsModel)

marsImptbl <- marsImp$importance %>% kable(caption = "MARS Model - Variable Importance",
    digits = 2) %>% kable_styling()


# (7.5a) Models FOR FINAL HW SUBMISSION, DO NOT
# REPEAT HW2-1 STEPS: JUST CALL VARIABLES FROM
# PRIOR ASSIGNMENT.
data(ChemicalManufacturingProcess)
CMP_Impute <- mice(ChemicalManufacturingProcess, m = 5,
    printFlag = F)
CMP_DF <- mice::complete(CMP_Impute, 2)
```

```r
# Set random seed
set.seed(1)

# 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, ]

# 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, ]

# Non-Linear Model 1: Support Vector Machines with
# Radial Basis Function Kernel
svmModel <- train(Yield ~ ., data = train, method = "svmRadial",
    preProcess = "pca", trControl = trainControl(method = "cv",
        number = 5), tuneLength = 5)
svmPred <- predict(svmModel, newdata = test)
svmPerf <- postResample(pred = svmPred, obs = test$Yield)

# Non-Linear Model 2: Bayesian Ridge Regression
# (Model Averaged)
bma_rrModel <- train(Yield ~ ., data = train, method = "blassoAveraged",
    trControl = trainControl(method = "cv", number = 5),
    tuneLength = 5)
bma_rrPred <- predict(bma_rrModel, newdata = test)
bma_rrPerf <- postResample(pred = bma_rrPred, obs = test$Yield)

# Non-Linear Model 3: k-Nearest Neighbors (KNN)
knnModel <- train(Yield ~ ., data = train, method = "knn",
    trControl = trainControl(method = "cv", number = 5),
    tuneLength = 5)
knnPred <- predict(knnModel, newdata = test)
knnPerf <- postResample(pred = knnPred, obs = test$Yield)

# Non-Linear Model 4: Multivariate Adaptive
# Regression Spline (MARS)
marsModel <- train(Yield ~ ., data = train, method = "earth",
    tuneGrid = expand.grid(degree = 1, nprune = 10),
    trControl = trainControl(method = "repeatedcv",
        repeats = 5))
marsPred <- predict(marsModel, newdata = test)
marsPerf <- postResample(pred = marsPred, obs = test$Yield)

# Performance
performance_table2 <- rbind(svmTrain = c(svmModel$results$RMSE[5],
    svmModel$results$Rsquared[5], svmModel$results$MAE[5]),
    svmTest = svmPerf, knnTrain = c(RMSE = knnModel$results$RMSE[2],
        RSquared = knnModel$results$RMSE[2], MAE = knnModel$results$RMSE[2]),
    knnTest = knnPerf, bma_rrTrain = c(RMSE = bma_rrModel$results$RMSE,
        RSquared = bma_rrModel$results$Rsquared, MAE = bma_rrModel$results$MAE),
    bma_rrTest = bma_rrPerf, marsTrain = c(marsModel$results$RMSE,
        marsModel$results$Rsquared, marsModel$results$MAE),
    marsTest = marsPerf) %>% kable(caption = "Model Performance",
    digits = 4) %>% kable_styling() %>% row_spec() %>%
    row_spec(row = 1:2, background = "#d9f2e6")


# (7.5b)
marsImp <- varImp(marsModel)

mars_most_important <- as.data.frame(marsImp$importance) %>%
    rownames_to_column("Variable") %>% filter(Overall >
    0)

plot2 <- ggplot(mars_most_important, aes(x = reorder(Variable,
    Overall), y = Overall)) + geom_point() + geom_segment(aes(x = Variable,
    xend = Variable, y = 0, yend = Overall)) + labs(title = "Variable Importance",
    subtitle = "SVM Model for CMP Data", x = "", y = "Importance") +
    coord_flip() + theme_bw() + theme()

# (7.5c)
top <- mars_most_important$Variable
cor <- cor(train[, top], train$Yield, method = "pearson")
cor_tbl <- cor %>% as.data.frame() %>% rownames_to_column("Variable") %>%
    rename(Correlation = V1) %>% arrange(desc(Correlation)) %>%
    kable(caption = "Correlation") %>% kable_styling()
plot3 <- CMP_DF[, top] %>% cbind(Yield = CMP_DF$Yield) %>%
    gather(key = "Variable", value = "Value", -Yield) %>%
    ggplot(aes(Value, Yield, color = Variable)) + geom_point(color = "grey") +
    geom_smooth(stat = "smooth", color = "black", method = "loess") +
    facet_wrap(~Variable, scales = "free_x", nrow = 2) +
    labs(title = "Variable Importance Scatterplot",
        y = "Yield Outcome", x = "Raw Material Measurements") +
    theme_bw() + theme(legend.position = "none", axis.text.x = element_text(angle = 45,
    hjust = 1))
```