Team 2 - Homework Two

Assignment 2: KJ 7.2; KJ 7.5

Vinicio

DATE

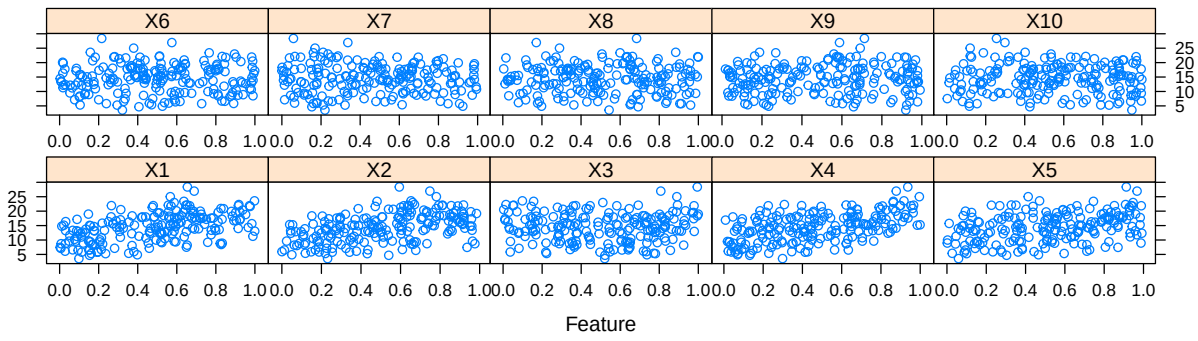
Dependencies

```
# predictive modeling
libraries("mlbench", "caret", "AppliedPredictiveModeling",
    "impute")
# Formatting Libraries
libraries("default", "knitr", "kableExtra", "dplyr")
# 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: $y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0, \sigma^2)$; 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:**

```
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)
```

(a) 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
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)
```

Model 1: MARS Regression: MARS, otherwise known as multivariate adapitive regression splines is a non parametric regression technique that automatically captures non-linearity and interaction between predictors. The basic MARS model has the following form:

$$\hat{f} = \sum_{i=1}^{k} c_i B_i(x)$$

The model computes the sum of basis functions B multiplied by constant coefficients c.The basis function can either be a constant, a hinge function, or a product of hinge functions. By definition, a hinge function is a piecewise function that converge at a point known as a knot.

```
Multivariate Adaptive Regression Spline

200 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE
  1        2      4.468111  0.2087388  3.7431452
  1        4      2.641758  0.7407360  2.0924332
  1        6      2.311051  0.8060173  1.7934162
  1        8      1.673679  0.8930817  1.3185331
  1       10      1.651006  0.8944561  1.2866820
  1       12      1.645767  0.8967129  1.2744893
  1       14      1.694342  0.8883776  1.3184117
  2        2      4.468111  0.2087388  3.7431452
  2        4      2.641758  0.7407360  2.0924332
  2        6      2.367878  0.7871842  1.8467180
  2        8      1.702625  0.8886624  1.2987304
  2       10      1.479107  0.9165083  1.1564345
  2       12      1.313741  0.9314083  1.0452799
  2       14      1.266703  0.9371059  0.9956994

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 14 and degree = 2.
```

Model 2:

SVM: SVM, also known as support vector machine is a method that can be applied to classification and regression tasks. On a high level, SVM creates a hyperplane in n dimensional space. This hyperplane acts

like a classification boundary which can be linear or non linear. This boundary classifies information from a feature space.

```
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 8

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0632601638659396

Number of Support Vectors : 152

Objective Function Value : -72.2262
Training error : 0.009161
```

Model 3:

KNN: This model is provided to use through the literature. KNn, also known as k-nearest neighbor is a method that can be applied to classification and regression problems. At a high level, kNN is a using an applied version of the Euclidean distance. The technique classifies a feature based on the k measure of the nearest neighbor features.

```
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  3.525127  0.5098546  2.857808
   7  3.422232  0.5417690  2.779895
   9  3.373772  0.5644477  2.740805
  11  3.356367  0.5783349  2.707764
  13  3.325697  0.5997964  2.669882
  15  3.305349  0.6188194  2.664031
  17  3.290196  0.6345102  2.653819
  19  3.290632  0.6441826  2.658689
  21  3.298473  0.6505305  2.673455
  23  3.299563  0.6589666  2.672668

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

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

	Overall
X1	100.00000
X4	85.13408
X2	69.22036
X5	49.27524
X3	39.95037
X6	0.00000
X7	0.00000
X8	0.00000
X9	0.00000
X10	0.00000

Before we take on which model performed best, lets examine if the mars model contained predictors X1-X5. Using variable importance, we can generate a list of features ranked by importance. From our table of important features, we can confirm that mars model selected features X1-X5 with X1 being the most important.

We turn to address the problem of model performance. We built a MARS model and SVM model. We were provided with a KNN model by the literature.

	KNN_metrics
RMSE	3.2040595
Rsquared	0.6819919
MAE	2.5683461

	MARS_metrics
RMSE	1.1722635
Rsquared	0.9448890
MAE	0.9324923

	SVM_metrics
RMSE	2.0552496
Rsquared	0.8288511
MAE	1.5595559

It is clear that the MARS model is the best performing. It captures over 90 percent of the data variability in addition to having the lowest RMSE of the three models.

(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. Manufactuing process 03 is missing 8.52 percent of entires. There are more predictors missing less than 3 percent of their entries. This is an ideal situation to impute variables. The impute package is not available in CRAN. We need to install it directly from BiocManager. We utilize knn method to impute missing values across all variables with missing data. We essentially use k nearest neighbors toimpute the missing values. For each variable with missing data, we use Euclidean distance to identify the k nearest neighbors. If we are missing a coordinate to compute the distance, the package uses the average distance from the closest non missing coordinates. This package assumes that not all variables are missing data.
>
> (a) Which nonlinear regression model gives the optimal resampling and test set performance? MARS Model with performance metrics

	MARS_Metrics
RMSE	1.0127496
Rsquared	0.6369266
MAE	0.8222455

SVMmodel with performance metrics

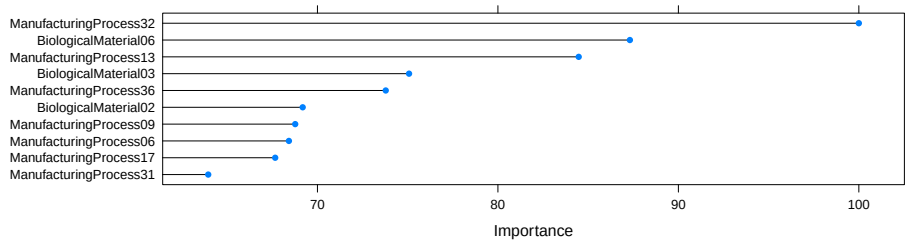
	SVM_Metrics
RMSE	0.9421919
Rsquared	0.7047197
MAE	0.7921343

KNN model and performance metrics

	SVM_Metrics
RMSE	1.1971397
Rsquared	0.4732448
MAE	0.9028693

Radial SVM method proves to be the best model in our case. This model has the lower RMSE while not showing a major decrease in the percentage of data variabiity captured.

(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?



Manufacturing process 32 is the most important predictor in our model. Within the top 10 mostimportant predictors, Manufactuing predictors outnumber Biological Predictors. From the linear model, we had Manufacturing Process 36 as the most important predictor followed by BiologicalMaterial03. Overall, the linear process is still dominated by manufacturing process predictors.

(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?

	Yield	ManufacturingProcess32	BiologicalMaterial06	ManufacturingProcess13	Mar
Yield	1.0000000	0.6083321	0.4781634	-0.5036797	
ManufacturingProcess32	0.6083321	1.0000000	0.6005958	-0.1012068	
BiologicalMaterial06	0.4781634	0.6005958	1.0000000	-0.1218676	
ManufacturingProcess13	-0.5036797	-0.1012068	-0.1218676	1.0000000	
ManufacturingProcess09	0.5034705	0.0410030	0.2300597	-0.7913537	
ManufacturingProcess36	-0.5250028	-0.7774848	-0.5106558	0.0660408	
BiologicalMaterial03	0.4450860	0.5318574	0.8723637	-0.1336953	

ManufacturingProcess13 and ManufacturingProcess36 have the strongest inverse relationship with Yield. This could suggets that something needs to be done to these features since they seem to contribute more to decrease in Yield. If these features they have an adverse effect on yield, then that impacts revenue potential. ManufacturingProcess32 has the strongest positive effect on Yield. Perhaps ManufacturingProcess13 can be

used as a benchmark to suggest change in other manufacturing predictors. ## R Code

```
# insert all code here (7.2a)
set.seed(101)
marsGrid <- expand.grid(degree = 1:2, nprune = seq(2,
    14, by = 2))
mars_mod <- train(x = trainingData$x, y = trainingData$y,
    method = "earth", tuneGrid = marsGrid, trControl = trainControl(method = "cv"))

set.seed(102)
svm_mod <- train(x = trainingData$x, y = trainingData$y,
    method = "svmRadial", tuneLength = 14, trControl = trainControl(method = "cv"))
svm_mod$finalModel
mars_mod
# (7.2b)
dt <- as.matrix(varImp(mars_mod)$importance)

kable(dt)

knn_pred <- predict(knn_mod, newdata = testData$x)
kp <- postResample(pred = knn_pred, obs = testData$y)

mars_Pred <- predict(mars_mod, testData$x)
mp <- postResample(pred = mars_Pred, obs = testData$y)

svm_pred <- predict(svm_mod, newdata = testData$x)
sp <- postResample(pred = svm_pred, obs = testData$y)

kd <- as.data.frame(as.matrix(kp))
colnames(kd) <- c("KNN_metrics")
md <- as.data.frame(as.matrix(mp))
colnames(md) <- c("MARS_metrics")
sd <- as.data.frame(as.matrix(sp))
colnames(sd) <- c("SVM_metrics")

kx <- kd$KNN_metrics
mx <- kd$mars_metrics
sx <- kd$SVM_metrics




kable(kd)
kable(md)
kable(sd)
# (7.5a)
data("ChemicalManufacturingProcess")
cd <- ChemicalManufacturingProcess

# Package to use knn imputing if
# (!requireNamespace('BiocManager', quietly =
# TRUE)) install.packages('BiocManager')
```

```r
library(impute)
cd2 <- impute.knn(as.matrix(cd))
cd2 <- as.data.frame(cd2$data)

# partition data into test and train
set.seed(20)
train_row_partition <- createDataPartition(cd2$Yield,
    p = 0.8, list = FALSE)
X_train <- cd2[train_row_partition, -1]
y_train <- cd2[train_row_partition, 1]
X_test <- cd2[-train_row_partition, -1]
y_test <- cd2[-train_row_partition, 1]

mars_mod2 <- earth(X_train, y_train)
mars_pred2 <- as.data.frame(predict(mars_mod2, newdata = X_test))

act <- as.data.frame(y_test)

colnames(mars_pred2) <- c("predicted")
colnames(act) <- c("actual")

mars_metric <- cbind(mars_pred2, act)
meat1 <- as.data.frame(postResample(mars_metric$actual,
    mars_metric$predicted))
colnames(meat1) <- c("MARS_Metrics")

kable(meat1)

# code
set.seed(102)
svm_mod2 <- train(x = X_train, y = y_train, method = "svmRadial",
    tuneLength = 14, trControl = trainControl(method = "cv"))

svm_pred2 <- as.data.frame(predict(svm_mod2, newdata = X_test))

act2 <- as.data.frame(y_test)

colnames(svm_pred2) <- c("predicted")
colnames(act2) <- c("actual")

svm_metric <- cbind(svm_pred2, act2)
meat2 <- as.data.frame(postResample(svm_metric$actual,
    svm_metric$predicted))
colnames(meat2) <- c("SVM_Metrics")

kable(meat2)

knn_mod2 <- train(x = X_train, y = y_train, method = "knn",
    preProc = c("center", "scale"), tuneLength = 10)

knn_pred2 <- as.data.frame(predict(knn_mod2, newdata = X_test))

act3 <- as.data.frame(y_test)
```

```r
colnames(knn_pred2) <- c("predicted")
colnames(act3) <- c("actual")

knn_metric <- cbind(knn_pred2, act3)
meat3 <- as.data.frame(postResample(knn_metric$actual,
    knn_metric$predicted))
colnames(meat3) <- c("SVM_Metrics")

kable(meat3)


# (7.5b) svm_mod2$finalModel$coefficients
key_features <- varImp(svm_mod2)
plot(key_features, top = 20)
# (7.5c)
imp_train <- cd2 %>% select(Yield, ManufacturingProcess32,
    BiologicalMaterial06, ManufacturingProcess13, ManufacturingProcess09,
    ManufacturingProcess36, BiologicalMaterial03)
cdt <- as.data.frame(cor(imp_train))

kable(cdt)
```