DATA624 Predictive Analytics

Group 3

**TEAM:**

## **Kuiete Tchoupou Alain**

## **Littlejohn Jeffrey**

## **Malhotra Samriti**

## **Mishra Rajwant**

## **Ng Jimmy**

Homework Assignment 2

Table of Contents

[Kuiete Tchoupou Alain 1](#__RefHeading___Toc10790_1911604043)

[Littlejohn Jeffrey 1](#__RefHeading___Toc10792_1911604043)

[Malhotra Samriti 1](#__RefHeading___Toc10794_1911604043)

[Mishra Rajwant 1](#__RefHeading___Toc10796_1911604043)

[Ng Jimmy 1](#__RefHeading___Toc10798_1911604043)

[Kuhn & Johnson 6 5](#__RefHeading___Toc5770_3750042978)

[HW\_KJ\_6.3 5](#__RefHeading___Toc6530_19116040431)

[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: 5](#__RefHeading___Toc5650_3750042978)

[1. 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. 5](#__RefHeading___Toc5652_3750042978)

[2. 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). 5](#__RefHeading___Toc5654_3750042978)

[3. 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? 12](#__RefHeading___Toc5656_3750042978)

[4. 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? 27](#__RefHeading___Toc5658_3750042978)

[5. Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list? 28](#__RefHeading___Toc5660_3750042978)

[6. 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? 36](#__RefHeading___Toc5662_3750042978)

[Kuhn & Johnson 7 38](#__RefHeading___Toc6528_1911604043)

[HW\_KJ\_7.2 38](#__RefHeading___Toc6530_1911604043)

[Tune several models on these data. For example: 39](#__RefHeading___Toc6275_1911604043)

[MARS 41](#__RefHeading___Toc6277_1911604043)

[SVM 42](#__RefHeading___Toc6279_1911604043)

[NNET 43](#__RefHeading___Toc6281_1911604043)

[Which models appear to give the best performance? Does MARS select the informative predictors (those named X1-X5)? 44](#__RefHeading___Toc6283_1911604043)

[HW\_KJ\_7.5 48](#__RefHeading___Toc6532_1911604043)

[(a). Which nonlinear regression model gives the optimal resampling and test set performance? 48](#__RefHeading___Toc6287_1911604043)

[(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? 53](#__RefHeading___Toc6289_1911604043)

[(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? 54](#__RefHeading___Toc6291_1911604043)

[Kuhn & Johnson 8 59](#__RefHeading___Toc6534_1911604043)

[HW\_KJ\_8.1 59](#__RefHeading___Toc6536_1911604043)

[8.1. Recreate the simulated data from Exercise 7.2: 59](#__RefHeading___Toc6295_1911604043)

[(a) Fit a random forest model to all of the predictors, then estimate the variable importance scores: 59](#__RefHeading___Toc6297_1911604043)

[Did the random forest model significantly use the uninformative predictors (V6 – V10)? 60](#__RefHeading___Toc6299_1911604043)

[(b) Now add an additional predictor that is highly correlated with one of the informative predictors. For example: 61](#__RefHeading___Toc6301_1911604043)

[Fit another random forest model to these data. Did the importance score for V1 change? What happens when you add another predictor that is also highly correlated with V1? 61](#__RefHeading___Toc6303_1911604043)

[(c) Use the cforest function in the party package to fit a random forest model using conditional inference trees. The party package function varimp can calculate predictor importance. The conditional argument of that function toggles between the traditional importance measure and the modified version described in Strobl et al. (2007). Do these importances show the same pattern as the traditional random forest model? 62](#__RefHeading___Toc6305_1911604043)

[(d) Repeat this process with different tree models, such as boosted trees and Cubist. Does the same pattern occur? 64](#__RefHeading___Toc6307_1911604043)

[HW\_KJ\_8.2 68](#__RefHeading___Toc6538_1911604043)

[8.2. Use a simulation to show tree bias with different granularities. 68](#__RefHeading___Toc6309_1911604043)

[HW\_KJ\_8.3 70](#__RefHeading___Toc6540_1911604043)

[8.3. In stochastic gradient boosting the bagging fraction and learning rate will govern the construction of the trees as they are guided by the gradient. Although the optimal values of these parameters should be obtained through the tuning process, it is helpful to understand how the magnitudes of these parameters affect magnitudes of variable importance. Figure 8.24 provides the variable importance plots for boosting using two extreme values for the bagging fraction (0.1 and 0.9) and the learning rate (0.1 and 0.9) for the solubility data. The left-hand plot has both parameters set to 0.1, and the right-hand plot has both set to 0.9: 70](#__RefHeading___Toc10802_1911604043)

[(a) Why does the model on the right focus its importance on just the first few of predictors, whereas the model on the left spreads importance across more predictors? 70](#__RefHeading___Toc10804_1911604043)

[(b) Which model do you think would be more predictive of other samples? 71](#__RefHeading___Toc6542_1911604043)

[(c) How would increasing interaction depth affect the slope of predictor importance for either model in Fig. 8.24 71](#__RefHeading___Toc6317_1911604043)

[HW\_KJ\_8.7 71](#__RefHeading___Toc10806_1911604043)

[8.7. Refer to Exercises 6.3 and 7.5 which describe a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several tree-based models: 71](#__RefHeading___Toc11603_1911604043)

[Preprocessing 72](#__RefHeading___Toc11605_1911604043)

[Fitting Models 73](#__RefHeading___Toc11607_1911604043)

[Bagged Cart 73](#__RefHeading___Toc11609_1911604043)

[Random Forest 73](#__RefHeading___Toc11611_1911604043)

[Cubist 73](#__RefHeading___Toc11613_1911604043)

[Stochastic Gradient Boosting 75](#__RefHeading___Toc11615_1911604043)

[Prediction 75](#__RefHeading___Toc11617_1911604043)

[Performance 75](#__RefHeading___Toc11619_1911604043)

[Summary 75](#__RefHeading___Toc11621_1911604043)

[77](#__RefHeading___Toc5664_3750042978)

[(a) Which tree-based regression model gives the optimal resampling and test set performance? 77](#__RefHeading___Toc11623_1911604043)

[(b) Which predictors are most important in the optimal tree-based regression model? Do either the biological or process variables dominate the list? How do the top 10 important predictors compare to the top 10 predictors from the optimal linear and nonlinear models? 78](#__RefHeading___Toc11625_1911604043)

[(c) Plot the optimal single tree with the distribution of yield in the terminal nodes. Does this view of the data provide additional knowledge about the biological or process predictors and their relationship with yield? 79](#__RefHeading___Toc11627_1911604043)

[Kuhn & Johnson 10 80](#__RefHeading___Toc10808_1911604043)

[HW\_KJ\_10 80](#__RefHeading___Toc6530_19116040432)

[Market basket analysis / recommender systems 81](#__RefHeading___Toc11485_1911604043)

[Introduction 81](#__RefHeading___Toc11487_1911604043)

[You assignment is to use R to mine the data for association rules. You should report support, confidence and lift and your top 10 rules by lift. Turn in as you would the other problems from HA and KJ. Due 07/18/20 with the packaged set, HW #2. 81](#__RefHeading___Toc11489_1911604043)

[MARKET BASKET ANALYSIS 82](#__RefHeading___Toc11491_1911604043)

[Data preprocessing and exploring 82](#__RefHeading___Toc11493_1911604043)

[Association rules for the grocery store 86](#__RefHeading___Toc6180_3750042978)

[Create some rules 87](#__RefHeading___Toc11497_1911604043)

# Kuhn & Johnson 6

## HW\_KJ\_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. 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. |
| **library**(AppliedPredictiveModeling)  **library**(VIM)  **library**(caret) |
| data(ChemicalManufacturingProcess) |
| **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)**. |
| We wil user kNN imputation method to impute the missing values from VIM package. |
| summary(ChemicalManufacturingProcess) |
| ## Yield BiologicalMaterial01 BiologicalMaterial02 BiologicalMaterial03  ## Min. :35.25 Min. :4.580 Min. :46.87 Min. :56.97  ## 1st Qu.:38.75 1st Qu.:5.978 1st Qu.:52.68 1st Qu.:64.98  ## Median :39.97 Median :6.305 Median :55.09 Median :67.22  ## Mean :40.18 Mean :6.411 Mean :55.69 Mean :67.70  ## 3rd Qu.:41.48 3rd Qu.:6.870 3rd Qu.:58.74 3rd Qu.:70.43  ## Max. :46.34 Max. :8.810 Max. :64.75 Max. :78.25  ##  ## BiologicalMaterial04 BiologicalMaterial05 BiologicalMaterial06  ## Min. : 9.38 Min. :13.24 Min. :40.60  ## 1st Qu.:11.24 1st Qu.:17.23 1st Qu.:46.05  ## Median :12.10 Median :18.49 Median :48.46  ## Mean :12.35 Mean :18.60 Mean :48.91  ## 3rd Qu.:13.22 3rd Qu.:19.90 3rd Qu.:51.34  ## Max. :23.09 Max. :24.85 Max. :59.38  ##  ## BiologicalMaterial07 BiologicalMaterial08 BiologicalMaterial09  ## Min. :100.0 Min. :15.88 Min. :11.44  ## 1st Qu.:100.0 1st Qu.:17.06 1st Qu.:12.60  ## Median :100.0 Median :17.51 Median :12.84  ## Mean :100.0 Mean :17.49 Mean :12.85  ## 3rd Qu.:100.0 3rd Qu.:17.88 3rd Qu.:13.13  ## Max. :100.8 Max. :19.14 Max. :14.08  ##  ## BiologicalMaterial10 BiologicalMaterial11 BiologicalMaterial12  ## Min. :1.770 Min. :135.8 Min. :18.35  ## 1st Qu.:2.460 1st Qu.:143.8 1st Qu.:19.73  ## Median :2.710 Median :146.1 Median :20.12  ## Mean :2.801 Mean :147.0 Mean :20.20  ## 3rd Qu.:2.990 3rd Qu.:149.6 3rd Qu.:20.75  ## Max. :6.870 Max. :158.7 Max. :22.21  ##  ## ManufacturingProcess01 ManufacturingProcess02 ManufacturingProcess03  ## Min. : 0.00 Min. : 0.00 Min. :1.47  ## 1st Qu.:10.80 1st Qu.:19.30 1st Qu.:1.53  ## Median :11.40 Median :21.00 Median :1.54  ## Mean :11.21 Mean :16.68 Mean :1.54  ## 3rd Qu.:12.15 3rd Qu.:21.50 3rd Qu.:1.55  ## Max. :14.10 Max. :22.50 Max. :1.60  ## NA's :1 NA's :3 NA's :15  ## ManufacturingProcess04 ManufacturingProcess05 ManufacturingProcess06  ## Min. :911.0 Min. : 923.0 Min. :203.0  ## 1st Qu.:928.0 1st Qu.: 986.8 1st Qu.:205.7  ## Median :934.0 Median : 999.2 Median :206.8  ## Mean :931.9 Mean :1001.7 Mean :207.4  ## 3rd Qu.:936.0 3rd Qu.:1008.9 3rd Qu.:208.7  ## Max. :946.0 Max. :1175.3 Max. :227.4  ## NA's :1 NA's :1 NA's :2  ## ManufacturingProcess07 ManufacturingProcess08 ManufacturingProcess09  ## Min. :177.0 Min. :177.0 Min. :38.89  ## 1st Qu.:177.0 1st Qu.:177.0 1st Qu.:44.89  ## Median :177.0 Median :178.0 Median :45.73  ## Mean :177.5 Mean :177.6 Mean :45.66  ## 3rd Qu.:178.0 3rd Qu.:178.0 3rd Qu.:46.52  ## Max. :178.0 Max. :178.0 Max. :49.36  ## NA's :1 NA's :1  ## ManufacturingProcess10 ManufacturingProcess11 ManufacturingProcess12  ## Min. : 7.500 Min. : 7.500 Min. : 0.0  ## 1st Qu.: 8.700 1st Qu.: 9.000 1st Qu.: 0.0  ## Median : 9.100 Median : 9.400 Median : 0.0  ## Mean : 9.179 Mean : 9.386 Mean : 857.8  ## 3rd Qu.: 9.550 3rd Qu.: 9.900 3rd Qu.: 0.0  ## Max. :11.600 Max. :11.500 Max. :4549.0  ## NA's :9 NA's :10 NA's :1  ## ManufacturingProcess13 ManufacturingProcess14 ManufacturingProcess15  ## Min. :32.10 Min. :4701 Min. :5904  ## 1st Qu.:33.90 1st Qu.:4828 1st Qu.:6010  ## Median :34.60 Median :4856 Median :6032  ## Mean :34.51 Mean :4854 Mean :6039  ## 3rd Qu.:35.20 3rd Qu.:4882 3rd Qu.:6061  ## Max. :38.60 Max. :5055 Max. :6233  ## NA's :1  ## ManufacturingProcess16 ManufacturingProcess17 ManufacturingProcess18  ## Min. : 0 Min. :31.30 Min. : 0  ## 1st Qu.:4561 1st Qu.:33.50 1st Qu.:4813  ## Median :4588 Median :34.40 Median :4835  ## Mean :4566 Mean :34.34 Mean :4810  ## 3rd Qu.:4619 3rd Qu.:35.10 3rd Qu.:4862  ## Max. :4852 Max. :40.00 Max. :4971  ##  ## ManufacturingProcess19 ManufacturingProcess20 ManufacturingProcess21  ## Min. :5890 Min. : 0 Min. :-1.8000  ## 1st Qu.:6001 1st Qu.:4553 1st Qu.:-0.6000  ## Median :6022 Median :4582 Median :-0.3000  ## Mean :6028 Mean :4556 Mean :-0.1642  ## 3rd Qu.:6050 3rd Qu.:4610 3rd Qu.: 0.0000  ## Max. :6146 Max. :4759 Max. : 3.6000  ##  ## ManufacturingProcess22 ManufacturingProcess23 ManufacturingProcess24  ## Min. : 0.000 Min. :0.000 Min. : 0.000  ## 1st Qu.: 3.000 1st Qu.:2.000 1st Qu.: 4.000  ## Median : 5.000 Median :3.000 Median : 8.000  ## Mean : 5.406 Mean :3.017 Mean : 8.834  ## 3rd Qu.: 8.000 3rd Qu.:4.000 3rd Qu.:14.000  ## Max. :12.000 Max. :6.000 Max. :23.000  ## NA's :1 NA's :1 NA's :1  ## ManufacturingProcess25 ManufacturingProcess26 ManufacturingProcess27  ## Min. : 0 Min. : 0 Min. : 0  ## 1st Qu.:4832 1st Qu.:6020 1st Qu.:4560  ## Median :4855 Median :6047 Median :4587  ## Mean :4828 Mean :6016 Mean :4563  ## 3rd Qu.:4877 3rd Qu.:6070 3rd Qu.:4609  ## Max. :4990 Max. :6161 Max. :4710  ## NA's :5 NA's :5 NA's :5  ## ManufacturingProcess28 ManufacturingProcess29 ManufacturingProcess30  ## Min. : 0.000 Min. : 0.00 Min. : 0.000  ## 1st Qu.: 0.000 1st Qu.:19.70 1st Qu.: 8.800  ## Median :10.400 Median :19.90 Median : 9.100  ## Mean : 6.592 Mean :20.01 Mean : 9.161  ## 3rd Qu.:10.750 3rd Qu.:20.40 3rd Qu.: 9.700  ## Max. :11.500 Max. :22.00 Max. :11.200  ## NA's :5 NA's :5 NA's :5  ## ManufacturingProcess31 ManufacturingProcess32 ManufacturingProcess33  ## Min. : 0.00 Min. :143.0 Min. :56.00  ## 1st Qu.:70.10 1st Qu.:155.0 1st Qu.:62.00  ## Median :70.80 Median :158.0 Median :64.00  ## Mean :70.18 Mean :158.5 Mean :63.54  ## 3rd Qu.:71.40 3rd Qu.:162.0 3rd Qu.:65.00  ## Max. :72.50 Max. :173.0 Max. :70.00  ## NA's :5 NA's :5  ## ManufacturingProcess34 ManufacturingProcess35 ManufacturingProcess36  ## Min. :2.300 Min. :463.0 Min. :0.01700  ## 1st Qu.:2.500 1st Qu.:490.0 1st Qu.:0.01900  ## Median :2.500 Median :495.0 Median :0.02000  ## Mean :2.494 Mean :495.6 Mean :0.01957  ## 3rd Qu.:2.500 3rd Qu.:501.5 3rd Qu.:0.02000  ## Max. :2.600 Max. :522.0 Max. :0.02200  ## NA's :5 NA's :5 NA's :5  ## ManufacturingProcess37 ManufacturingProcess38 ManufacturingProcess39  ## Min. :0.000 Min. :0.000 Min. :0.000  ## 1st Qu.:0.700 1st Qu.:2.000 1st Qu.:7.100  ## Median :1.000 Median :3.000 Median :7.200  ## Mean :1.014 Mean :2.534 Mean :6.851  ## 3rd Qu.:1.300 3rd Qu.:3.000 3rd Qu.:7.300  ## Max. :2.300 Max. :3.000 Max. :7.500  ##  ## ManufacturingProcess40 ManufacturingProcess41 ManufacturingProcess42  ## Min. :0.00000 Min. :0.00000 Min. : 0.00  ## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:11.40  ## Median :0.00000 Median :0.00000 Median :11.60  ## Mean :0.01771 Mean :0.02371 Mean :11.21  ## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:11.70  ## Max. :0.10000 Max. :0.20000 Max. :12.10  ## NA's :1 NA's :1  ## ManufacturingProcess43 ManufacturingProcess44 ManufacturingProcess45  ## Min. : 0.0000 Min. :0.000 Min. :0.000  ## 1st Qu.: 0.6000 1st Qu.:1.800 1st Qu.:2.100  ## Median : 0.8000 Median :1.900 Median :2.200  ## Mean : 0.9119 Mean :1.805 Mean :2.138  ## 3rd Qu.: 1.0250 3rd Qu.:1.900 3rd Qu.:2.300  ## Max. :11.0000 Max. :2.100 Max. :2.600  ## |
| impu\_data <- kNN(ChemicalManufacturingProcess, imp\_var = FALSE) |
| summary((ChemicalManufacturingProcess$ManufacturingProcess02)) |
| ## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  ## 0.00 19.30 21.00 16.68 21.50 22.50 3  summary(impu\_data$ManufacturingProcess02)  ## Min. 1st Qu. Median Mean 3rd Qu. Max.  ## 0.00 19.30 21.00 16.76 21.50 22.50 |
| data.frame("OLD"=ChemicalManufacturingProcess$ManufacturingProcess02,  "Imputed"=impu\_data$ManufacturingProcess02) |
| ## OLD Imputed  ## 1 NA 21.0  ## 2 0.0 0.0  ## 3 0.0 0.0  ## 171 20.0 20.0  ## 172 21.5 21.5  ## 173 21.5 21.5  ## 174 20.4 20.4  ## 175 21.6 21.6  ## 176 20.8 20.8 |
| 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? |
| n <- nrow(impu\_data)  i.training <- sort(sample(n,round(n\*0.8)))  L.training <- impu\_data[i.training,]  L.test <- impu\_data[-i.training,]  X\_train <- L.training[,-1]  Y\_train <- L.training[,1]  X\_test <- L.test[,-1]  Y\_test <- L.test[,1]  ctrl <- trainControl(method = "cv", number = 10)  model\_lm <- lm(Yield~.,data=L.training )  *# model\_lm <- train(x = X\_train, y = Y\_train,*  *# method = "lm",*  *# trControl = ctrl)*  summary(model\_lm) |
| ##  ## Call:  ## lm(formula = Yield ~ ., data = L.training)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -2.28499 -0.52689 -0.06211 0.46025 2.39991  ##  ## Coefficients: (1 not defined because of singularities)  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) 2.651e+01 1.877e+02 0.141 0.88805  ## BiologicalMaterial01 1.206e-01 3.955e-01 0.305 0.76123  ## BiologicalMaterial02 -1.502e-01 1.536e-01 -0.978 0.33083  ## BiologicalMaterial03 2.429e-01 3.046e-01 0.797 0.42746  ## BiologicalMaterial04 -2.693e-01 6.728e-01 -0.400 0.68997  ## BiologicalMaterial05 1.853e-01 1.306e-01 1.419 0.15962  ## BiologicalMaterial06 9.591e-03 3.966e-01 0.024 0.98077  ## BiologicalMaterial07 -1.303e+00 1.610e+00 -0.809 0.42054  ## BiologicalMaterial08 5.734e-01 7.742e-01 0.741 0.46096  ## BiologicalMaterial09 -1.516e+00 1.749e+00 -0.867 0.38838  ## BiologicalMaterial10 7.633e-01 1.743e+00 0.438 0.66247  ## BiologicalMaterial11 -1.038e-01 1.073e-01 -0.967 0.33612  ## BiologicalMaterial12 5.672e-01 7.572e-01 0.749 0.45585  ## ManufacturingProcess01 -8.544e-03 1.139e-01 -0.075 0.94040  ## ManufacturingProcess02 8.416e-02 7.105e-02 1.184 0.23957  ## ManufacturingProcess03 -5.721e+00 6.493e+00 -0.881 0.38077  ## ManufacturingProcess04 7.225e-02 3.505e-02 2.062 0.04234 \*  ## ManufacturingProcess05 -1.814e-03 4.795e-03 -0.378 0.70609  ## ManufacturingProcess06 2.687e-02 4.900e-02 0.548 0.58494  ## ManufacturingProcess07 -1.952e-01 2.435e-01 -0.802 0.42495  ## ManufacturingProcess08 5.108e-03 3.012e-01 0.017 0.98651  ## ManufacturingProcess09 4.573e-01 2.359e-01 1.939 0.05591 .  ## ManufacturingProcess10 4.069e-01 5.565e-01 0.731 0.46668  ## ManufacturingProcess11 -1.413e-02 8.191e-01 -0.017 0.98628  ## ManufacturingProcess12 6.041e-05 1.372e-04 0.440 0.66072  ## ManufacturingProcess13 -5.274e-01 5.125e-01 -1.029 0.30640  ## ManufacturingProcess14 9.779e-05 1.235e-02 0.008 0.99370  ## ManufacturingProcess15 1.282e-02 1.164e-02 1.101 0.27383  ## ManufacturingProcess16 -2.510e-06 5.880e-04 -0.004 0.99660  ## ManufacturingProcess17 4.222e-01 4.214e-01 1.002 0.31937  ## ManufacturingProcess18 2.580e-03 6.353e-03 0.406 0.68570  ## ManufacturingProcess19 2.936e-04 1.314e-02 0.022 0.98223  ## ManufacturingProcess20 -3.056e-03 1.254e-02 -0.244 0.80808  ## ManufacturingProcess21 NA NA NA NA  ## ManufacturingProcess22 -2.074e-02 5.274e-02 -0.393 0.69514  ## ManufacturingProcess23 -3.326e-02 1.050e-01 -0.317 0.75231  ## ManufacturingProcess24 -2.625e-02 2.939e-02 -0.893 0.37415  ## ManufacturingProcess25 7.353e-03 1.657e-02 0.444 0.65828  ## ManufacturingProcess26 -2.355e-03 1.321e-02 -0.178 0.85901  ## ManufacturingProcess27 -8.809e-03 1.333e-02 -0.661 0.51059  ## ManufacturingProcess28 -8.921e-02 3.946e-02 -2.261 0.02636 \*  ## ManufacturingProcess29 7.778e-01 1.014e+00 0.767 0.44515  ## ManufacturingProcess30 3.282e-01 6.625e-01 0.495 0.62164  ## ManufacturingProcess31 2.028e-02 1.303e-01 0.156 0.87666  ## ManufacturingProcess32 2.492e-01 7.869e-02 3.167 0.00215 \*\*  ## ManufacturingProcess33 -2.971e-01 1.516e-01 -1.959 0.05337 .  ## ManufacturingProcess34 -6.569e-01 3.285e+00 -0.200 0.84198  ## ManufacturingProcess35 -3.854e-03 2.036e-02 -0.189 0.85036  ## ManufacturingProcess36 -1.438e+02 3.774e+02 -0.381 0.70424  ## ManufacturingProcess37 -6.288e-01 3.762e-01 -1.672 0.09834 .  ## ManufacturingProcess38 -2.244e-01 2.887e-01 -0.777 0.43927  ## ManufacturingProcess39 1.337e-01 1.701e-01 0.786 0.43409  ## ManufacturingProcess40 2.368e-02 8.224e+00 0.003 0.99771  ## ManufacturingProcess41 3.963e-01 5.755e+00 0.069 0.94526  ## ManufacturingProcess42 2.085e-01 2.551e-01 0.817 0.41602  ## ManufacturingProcess43 1.726e-01 1.359e-01 1.270 0.20761  ## ManufacturingProcess44 -1.528e+00 1.500e+00 -1.019 0.31117  ## ManufacturingProcess45 1.408e+00 6.672e-01 2.110 0.03784 \*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  ##  ## Residual standard error: 1.078 on 84 degrees of freedom  ## Multiple R-squared: 0.7974, Adjusted R-squared: 0.6623  ## F-statistic: 5.902 on 56 and 84 DF, p-value: 2.242e-13 |
| The train function generates a resampling estimate of performance. Because the training set size is not small, 10-fold cross-validation should produce reasonable estimates of model performance. The function trainControl specifies the type of resampling: |
| ctrl <- trainControl(method = "cv", number = 10)  model\_lm1 <- train(x = X\_train, y = Y\_train, method = "lm", trControl = ctrl)  model\_lm1 |
| ## Linear Regression  ##  ## 141 samples  ## 57 predictor  ##  ## No pre-processing  ## Resampling: Cross-Validated (10 fold)  ## Summary of sample sizes: 127, 126, 125, 128, 129, 127, ...  ## Resampling results:  ##  ## RMSE Rsquared MAE  ## 3.874202 0.3936565 1.920977  ##  ## Tuning parameter 'intercept' was held constant at a value of TRUE |
| xyplot(Y\_train ~ predict(model\_lm1),  *## plot the points (type = 'p') and a background grid ('g')*  type = c("p", "g"),  xlab = "Predicted", ylab = "Observed")      xyplot(resid(model\_lm1) ~ predict(model\_lm1),  type = c("p", "g"),  xlab = "Predicted", ylab = "Residuals  *# To build a smaller model without predictors with extremely high correlations,*  corThresh <- .9  tooHigh <- findCorrelation(cor(X\_train), corThresh)  print(paste0(names(X\_train)[tooHigh]))    ## [1] "BiologicalMaterial02" "BiologicalMaterial04" "BiologicalMaterial12"  ## [4] "ManufacturingProcess29" "ManufacturingProcess42" "ManufacturingProcess44"  ## [7] "ManufacturingProcess25" "ManufacturingProcess27" "ManufacturingProcess31"  ## [10] "ManufacturingProcess40"  corrPred <- names(X\_train)[tooHigh]  X\_train\_no\_cor <- X\_train[, -tooHigh]  X\_test\_no\_cor <- X\_test[, -tooHigh]  model\_lm1\_no\_cor <- train(X\_train\_no\_cor, Y\_train, method = "lm",  trControl = ctrl)  model\_lm1\_no\_cor  ## Linear Regression  ##  ## 141 samples  ## 47 predictor  ##  ## No pre-processing  ## Resampling: Cross-Validated (10 fold)  ## Summary of sample sizes: 128, 127, 127, 127, 126, 127, ...  ## Resampling results:  ##  ## RMSE Rsquared MAE  ## 2.658052 0.4926409 1.458777  ##  ## Tuning parameter 'intercept' was held constant at a value of TRUE  xyplot(Y\_train ~ predict(model\_lm1\_no\_cor),  *## plot the points (type = 'p') and a background grid ('g')*  type = c("p", "g"),  xlab = "Predicted", ylab = "Observed")    xyplot(resid(model\_lm1\_no\_cor) ~ predict(model\_lm1\_no\_cor),  type = c("p", "g"),  xlab = "Predicted", ylab = "Residuals")    *#PLS*  *# Using train perform to perfrom pre-process and tuning together. The function first preprocess the training set by centering it and scaling it. Then the function uses 10-fold cross validation to try the number of components, i.e. latent variables, of the PLS model from 1 to 20.*  model\_pls\_no\_cor <- train(x=X\_train\_no\_cor, y=Y\_train,  method = "pls",  tuneLength = 20,  metric='Rsquared',  trControl = ctrl,  preProc = c("center", "scale"))  ## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =  ## 10, : These variables have zero variances: BiologicalMaterial07  model\_pls\_no\_cor  ## Partial Least Squares  ##  ## 141 samples  ## 47 predictor  ##  ## Pre-processing: centered (47), scaled (47)  ## Resampling: Cross-Validated (10 fold)  ## Summary of sample sizes: 126, 125, 127, 127, 128, 128, ...  ## Resampling results across tuning parameters:  ##  ## ncomp RMSE Rsquared MAE  ## 1 1.458709 0.4574974 1.148228  ## 2 1.347703 0.5148911 1.064615  ## 3 1.577812 0.5502737 1.136405  ## 4 1.774332 0.5349251 1.171156  ## 5 1.961094 0.5019444 1.219957  ## 6 2.074273 0.4997302 1.231290  ## 7 2.270451 0.4878229 1.303499  ## 8 2.292993 0.4897226 1.302224  ## 9 2.295947 0.4873463 1.317192  ## 10 2.213082 0.4881721 1.300682  ## 11 2.136101 0.4873956 1.280935  ## 12 2.059281 0.4839662 1.275996  ## 13 1.993511 0.4821788 1.266493  ## 14 1.968057 0.4834612 1.259621  ## 15 1.974853 0.4835267 1.256889  ## 16 2.051947 0.4802843 1.279162  ## 17 2.173141 0.4776259 1.313999  ## 18 2.282961 0.4764214 1.343129  ## 19 2.339001 0.4759227 1.360926  ## 20 2.396062 0.4758143 1.380584  ##  ## Rsquared was used to select the optimal model using the largest value.  ## The final value used for the model was ncomp = 3.  summary(model\_pls\_no\_cor)  ## Data: X dimension: 141 47  ## Y dimension: 141 1  ## Fit method: oscorespls  ## Number of components considered: 3  ## TRAINING: % variance explained  ## 1 comps 2 comps 3 comps  ## X 15.57 26.63 33.63  ## .outcome 52.96 62.81 69.02  *#enet*  *# The optimal Lasso model had fraction = 0.25 and lambda = 0.1*  enetGrid <- expand.grid(.lambda = c(0, 0.01, .1),  .fraction = seq(.05, 1, length = 20))  model\_ener\_no\_cor <- train(x=X\_train\_no\_cor, y=Y\_train,  method = "enet",  tuneGrid = enetGrid,  trControl = ctrl,  preProc = c("center", "scale"))  model\_ener\_no\_cor  ## Elasticnet  ##  ## 141 samples  ## 47 predictor  ##  ## Pre-processing: centered (47), scaled (47)  ## Resampling: Cross-Validated (10 fold)  ## Summary of sample sizes: 125, 127, 128, 127, 126, 126, ...  ## Resampling results across tuning parameters:  ##  ## lambda fraction RMSE Rsquared MAE  ## 0.00 0.05 1.503639 0.5579717 1.2286939  ## 0.00 0.10 1.297466 0.5707639 1.0613770  ## 0.00 0.15 1.242583 0.5639709 1.0165484  ## 0.00 0.20 1.206247 0.5856567 0.9767223  ## 0.00 0.25 1.188698 0.5979767 0.9728722  ## 0.00 0.30 1.307138 0.5537706 1.0302272  ## 0.00 0.35 1.589036 0.5353523 1.1150142  ## 0.00 0.40 1.855539 0.5349656 1.1859978  ## 0.00 0.45 2.121417 0.5330365 1.2553891  ## 0.00 0.50 2.392626 0.5341373 1.3315064  ## 0.00 0.55 2.423332 0.5269313 1.3509160  ## 0.00 0.60 2.480140 0.5121562 1.3758467  ## 0.00 0.65 2.609282 0.5019211 1.4171859  ## 0.00 0.70 2.721160 0.4968153 1.4525080  ## 0.00 0.75 2.787234 0.4927105 1.4769178  ## 0.00 0.80 2.835616 0.4889234 1.4983123  ## 0.00 0.85 2.868804 0.4855544 1.5150220  ## 0.00 0.90 2.873294 0.4822253 1.5232851  ## 0.00 0.95 2.932410 0.4761127 1.5487505  ## 0.00 1.00 2.987264 0.4708093 1.5742194  ## 0.01 0.05 1.560605 0.5438803 1.2712212  ## 0.01 0.10 1.359169 0.5774719 1.1115770  ## 0.01 0.15 1.264602 0.5658312 1.0378194  ## 0.01 0.20 1.231676 0.5693867 1.0059973  ## 0.01 0.25 1.208905 0.5834280 0.9814489  ## 0.01 0.30 1.196519 0.5914192 0.9817102  ## 0.01 0.35 1.192284 0.5970780 0.9852138  ## 0.01 0.40 1.376182 0.5458714 1.0582050  ## 0.01 0.45 1.616335 0.5350300 1.1293663  ## 0.01 0.50 1.814523 0.5325775 1.1833742  ## 0.01 0.55 1.975520 0.5330850 1.2235537  ## 0.01 0.60 2.163280 0.5342524 1.2680065  ## 0.01 0.65 2.313993 0.5358659 1.3058653  ## 0.01 0.70 2.326406 0.5316054 1.3234823  ## 0.01 0.75 2.384033 0.5214064 1.3480633  ## 0.01 0.80 2.425499 0.5140739 1.3646067  ## 0.01 0.85 2.463973 0.5092787 1.3775205  ## 0.01 0.90 2.524694 0.5051815 1.3969604  ## 0.01 0.95 2.576200 0.5012725 1.4168724  ## 0.01 1.00 2.619690 0.4965698 1.4356840  ## 0.10 0.05 1.630812 0.5125344 1.3260770  ## 0.10 0.10 1.469636 0.5699834 1.1977413  ## 0.10 0.15 1.352863 0.5716689 1.1042990  ## 0.10 0.20 1.283034 0.5648607 1.0553452  ## 0.10 0.25 1.257926 0.5558212 1.0275066  ## 0.10 0.30 1.232324 0.5688482 1.0008282  ## 0.10 0.35 1.212503 0.5818975 0.9896883  ## 0.10 0.40 1.204527 0.5879443 0.9913823  ## 0.10 0.45 1.298352 0.5543888 1.0396517  ## 0.10 0.50 1.489219 0.5390807 1.1024189  ## 0.10 0.55 1.653772 0.5327360 1.1507042  ## 0.10 0.60 1.791104 0.5350562 1.1849283  ## 0.10 0.65 1.928248 0.5364567 1.2182559  ## 0.10 0.70 2.038903 0.5366090 1.2448179  ## 0.10 0.75 2.081921 0.5359402 1.2537526  ## 0.10 0.80 2.125166 0.5348600 1.2626802  ## 0.10 0.85 2.161835 0.5342180 1.2827505  ## 0.10 0.90 2.205681 0.5312068 1.3009388  ## 0.10 0.95 2.239149 0.5288216 1.3146651  ## 0.10 1.00 2.273258 0.5266555 1.3282257  ##  ## RMSE was used to select the optimal model using the smallest value.  ## The final values used for the model were fraction = 0.25 and lambda = 0.  test\_model <- **function**(modelName,predData){  options(warn=-1) *#turn off warnings*  predicted\_result <- predict(modelName, predData)  options(warn=1)  *#We can collect the observed and predicted values into a data frame, then use*  *# the caret function defaultSummary to estimate the test set performance*  DT\_model\_lm\_pred <- data.frame(obs=Y\_test,pred=predicted\_result)  **return**(defaultSummary(DT\_model\_lm\_pred))  } 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? model\_lm1$results[,2:4]  ## RMSE Rsquared MAE  ## 1 3.874202 0.3936565 1.920977  test\_model(model\_lm1,X\_test)  ## RMSE Rsquared MAE  ## 1.2776049 0.6016466 1.0914990  model\_lm1\_no\_cor$results[,2:4]  ## RMSE Rsquared MAE  ## 1 2.658052 0.4926409 1.458777  test\_model(model\_lm1\_no\_cor,X\_test)  ## RMSE Rsquared MAE  ## 5.2092899 0.1077397 1.8131285  model\_pls\_no\_cor$results[3,2:4]  ## RMSE Rsquared MAE  ## 3 1.577812 0.5502737 1.136405  test\_model(model\_pls\_no\_cor,X\_test\_no\_cor)  ## RMSE Rsquared MAE  ## 2.5693673 0.2106991 1.2427024  model\_ener\_no\_cor$results[2,2:4]  ## fraction RMSE Rsquared  ## 21 0.05 1.560605 0.5438803  test\_model(model\_ener\_no\_cor,X\_test\_no\_cor)  ## RMSE Rsquared MAE  ## 1.0986265 0.6303881 0.8695010 e. Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list? model\_pls\_no\_cor$finalModel$coefficients  ## , , 1 comps  ##  ## .outcome  ## BiologicalMaterial01 0.097921794  ## BiologicalMaterial03 0.112976021  ## BiologicalMaterial05 0.039507121  ## BiologicalMaterial06 0.125979373  ## BiologicalMaterial07 -0.017960263  ## BiologicalMaterial08 0.104575302  ## BiologicalMaterial09 0.022375691  ## BiologicalMaterial10 0.058967561  ## BiologicalMaterial11 0.093643560  ## ManufacturingProcess01 -0.026024522  ## ManufacturingProcess02 -0.056769738  ## ManufacturingProcess03 -0.022925631  ## ManufacturingProcess04 -0.080727962  ## ManufacturingProcess05 0.025352928  ## ManufacturingProcess06 0.110811530  ## ManufacturingProcess07 -0.014799697  ## ManufacturingProcess08 -0.004834193  ## ManufacturingProcess09 0.143373389  ## ManufacturingProcess10 0.058072402  ## ManufacturingProcess11 0.107618486  ## ManufacturingProcess12 0.106959424  ## ManufacturingProcess13 -0.137707621  ## ManufacturingProcess14 0.006475541  ## ManufacturingProcess15 0.068858608  ## ManufacturingProcess16 -0.010268284  ## ManufacturingProcess17 -0.123629494  ## ManufacturingProcess18 -0.048005973  ## ManufacturingProcess19 0.030650399  ## ManufacturingProcess20 -0.059514779  ## ManufacturingProcess21 -0.017065995  ## ManufacturingProcess22 0.007680759  ## ManufacturingProcess23 -0.031422387  ## ManufacturingProcess24 -0.068441059  ## ManufacturingProcess26 0.009375918  ## ManufacturingProcess28 0.078438694  ## ManufacturingProcess30 0.068051715  ## ManufacturingProcess32 0.168867220  ## ManufacturingProcess33 0.117768879  ## ManufacturingProcess34 0.058178851  ## ManufacturingProcess35 -0.049168790  ## ManufacturingProcess36 -0.145120997  ## ManufacturingProcess37 -0.056099467  ## ManufacturingProcess38 -0.022486550  ## ManufacturingProcess39 0.007206070  ## ManufacturingProcess41 -0.012057725  ## ManufacturingProcess43 0.057812519  ## ManufacturingProcess45 0.009861988  ##  ## , , 2 comps  ##  ## .outcome  ## BiologicalMaterial01 0.035039544  ## BiologicalMaterial03 0.093121935  ## BiologicalMaterial05 0.027842222  ## BiologicalMaterial06 0.096088407  ## BiologicalMaterial07 -0.048631187  ## BiologicalMaterial08 0.045711350  ## BiologicalMaterial09 -0.040732999  ## BiologicalMaterial10 -0.008988251  ## BiologicalMaterial11 0.041736913  ## ManufacturingProcess01 0.002288708  ## ManufacturingProcess02 -0.005885621  ## ManufacturingProcess03 -0.039285219  ## ManufacturingProcess04 -0.029611651  ## ManufacturingProcess05 -0.009938794  ## ManufacturingProcess06 0.164621519  ## ManufacturingProcess07 -0.028663643  ## ManufacturingProcess08 0.001069445  ## ManufacturingProcess09 0.228209502  ## ManufacturingProcess10 0.069724268  ## ManufacturingProcess11 0.160750581  ## ManufacturingProcess12 0.137029524  ## ManufacturingProcess13 -0.220816613  ## ManufacturingProcess14 0.004047956  ## ManufacturingProcess15 0.089018919  ## ManufacturingProcess16 -0.015223573  ## ManufacturingProcess17 -0.226382162  ## ManufacturingProcess18 -0.098899458  ## ManufacturingProcess19 0.004750463  ## ManufacturingProcess20 -0.114781058  ## ManufacturingProcess21 -0.073895298  ## ManufacturingProcess22 0.024715355  ## ManufacturingProcess23 -0.029296060  ## ManufacturingProcess24 -0.063950774  ## ManufacturingProcess26 -0.007531911  ## ManufacturingProcess28 0.019325986  ## ManufacturingProcess30 0.095930179  ## ManufacturingProcess32 0.239755794  ## ManufacturingProcess33 0.128161505  ## ManufacturingProcess34 0.143997697  ## ManufacturingProcess35 -0.048541903  ## ManufacturingProcess36 -0.198286359  ## ManufacturingProcess37 -0.127980696  ## ManufacturingProcess38 -0.002127119  ## ManufacturingProcess39 0.064680607  ## ManufacturingProcess41 -0.030393587  ## ManufacturingProcess43 0.071583391  ## ManufacturingProcess45 0.074998801  ##  ## , , 3 comps  ##  ## .outcome  ## BiologicalMaterial01 -0.002007753  ## BiologicalMaterial03 0.087224878  ## BiologicalMaterial05 0.061952201  ## BiologicalMaterial06 0.091301809  ## BiologicalMaterial07 -0.083547419  ## BiologicalMaterial08 0.009992303  ## BiologicalMaterial09 -0.113334909  ## BiologicalMaterial10 -0.049986287  ## BiologicalMaterial11 0.012383089  ## ManufacturingProcess01 0.022413248  ## ManufacturingProcess02 -0.003415089  ## ManufacturingProcess03 -0.049372017  ## ManufacturingProcess04 0.061610800  ## ManufacturingProcess05 -0.049979822  ## ManufacturingProcess06 0.185156813  ## ManufacturingProcess07 -0.071127409  ## ManufacturingProcess08 -0.011296484  ## ManufacturingProcess09 0.247356375  ## ManufacturingProcess10 0.026764566  ## ManufacturingProcess11 0.146172779  ## ManufacturingProcess12 0.097175892  ## ManufacturingProcess13 -0.214648788  ## ManufacturingProcess14 0.076668712  ## ManufacturingProcess15 0.175883045  ## ManufacturingProcess16 -0.032176626  ## ManufacturingProcess17 -0.238607051  ## ManufacturingProcess18 -0.043636251  ## ManufacturingProcess19 0.084899237  ## ManufacturingProcess20 -0.089335265  ## ManufacturingProcess21 -0.102500702  ## ManufacturingProcess22 0.041977074  ## ManufacturingProcess23 -0.042819155  ## ManufacturingProcess24 -0.075858876  ## ManufacturingProcess26 -0.020903995  ## ManufacturingProcess28 -0.042986246  ## ManufacturingProcess30 0.078158754  ## ManufacturingProcess32 0.370295915  ## ManufacturingProcess33 0.173413626  ## ManufacturingProcess34 0.232679553  ## ManufacturingProcess35 -0.060512965  ## ManufacturingProcess36 -0.300299385  ## ManufacturingProcess37 -0.228961548  ## ManufacturingProcess38 0.025135247  ## ManufacturingProcess39 0.145554088  ## ManufacturingProcess41 -0.037893733  ## ManufacturingProcess43 0.091703875  ## ManufacturingProcess45 0.177924191  *# it appears that ManufacturingProcess are more important. Alternatively, varImp function can be used to rank the importance of predictors:*  varImp(model\_ener\_no\_cor)  ## loess r-squared variable importance  ##  ## only 20 most important variables shown (out of 47)  ##  ## Overall  ## ManufacturingProcess32 100.00  ## ManufacturingProcess13 87.97  ## BiologicalMaterial06 86.60  ## ManufacturingProcess17 77.05  ## ManufacturingProcess09 74.17  ## ManufacturingProcess36 73.83  ## BiologicalMaterial03 65.55  ## ManufacturingProcess11 56.44  ## ManufacturingProcess06 55.44  ## ManufacturingProcess30 50.38  ## ManufacturingProcess33 48.60  ## BiologicalMaterial11 46.36  ## BiologicalMaterial08 41.69  ## ManufacturingProcess12 40.07  ## BiologicalMaterial01 38.45  ## ManufacturingProcess15 37.70  ## ManufacturingProcess20 33.13  ## BiologicalMaterial09 31.47  ## ManufacturingProcess18 30.71  ## ManufacturingProcess26 27.22  varImp(model\_pls\_no\_cor)  ##  ## Attaching package: 'pls'  ## The following object is masked from 'package:caret':  ##  ## R2  ## The following object is masked from 'package:stats':  ##  ## loadings  ## pls variable importance  ##  ## only 20 most important variables shown (out of 47)  ##  ## Overall  ## ManufacturingProcess32 100.00  ## ManufacturingProcess36 84.00  ## ManufacturingProcess09 81.07  ## ManufacturingProcess13 77.11  ## ManufacturingProcess17 72.00  ## BiologicalMaterial06 65.65  ## ManufacturingProcess33 61.92  ## ManufacturingProcess06 60.98  ## BiologicalMaterial08 59.13  ## ManufacturingProcess11 58.86  ## BiologicalMaterial03 57.89  ## ManufacturingProcess12 57.82  ## BiologicalMaterial01 56.10  ## BiologicalMaterial11 52.30  ## ManufacturingProcess04 49.24  ## ManufacturingProcess28 47.04  ## ManufacturingProcess34 40.61  ## ManufacturingProcess15 39.70  ## ManufacturingProcess37 38.91  ## BiologicalMaterial10 36.34  Looking at only 3 comps, The Manufacturing Process seems to have the most importance, as generally their scores are higher than the Biological Materials. ManufacturingProcess32 has the highest score at 0.3687089330.  The evaluation on the test sets seems to suggest that the PLS model is best, with R^2 = 0.7202954 Here we noted that when we apply all the models on not correalted data then RMSE and Rsquared for bothe test and train PLS model is better compare to other model. Train: RMSE : 1.666406 Rsquared :0.4722788 TEST: RMSE : 1.0391511 Rsquared :0.7202954  13 out of the 20 in the list are ManufacturingProcess predictors, which makes it more important than BiologicalMaterial. |
| 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? |
| We can compare the non-zero coefficients, Elastic net is a linear regression model. The coefficients directly explain how the predictors affect the target. Positive coefficients improve the yield, while negative coefficients decrease the yield. |
| coeffs <- elasticnet::predict.enet(model\_ener\_no\_cor$finalModel, s=model\_ener\_no\_cor$bestTune[1, "fraction"], type="coef", mode="fraction")$coefficients |
| We can compare the non-zero coefficients by taking their absolute value, and then sorting them: |
| coeffs.sorted <- abs(coeffs)  coeffs.sorted <- coeffs.sorted[coeffs.sorted>0]  (coeffs.sorted <- sort(coeffs.sorted, decreasing = T)) |
| ## ManufacturingProcess32 ManufacturingProcess09 ManufacturingProcess45  ## 0.760119012 0.651939513 0.217011390  ## ManufacturingProcess36 ManufacturingProcess37 BiologicalMaterial05  ## 0.156031955 0.156003773 0.147387404  ## ManufacturingProcess34 ManufacturingProcess17 ManufacturingProcess13  ## 0.142832613 0.124789127 0.110417254  ## ManufacturingProcess06 ManufacturingProcess15 ManufacturingProcess04  ## 0.102493521 0.092675846 0.049124108  ## ManufacturingProcess07 ManufacturingProcess11 ManufacturingProcess43  ## 0.030563966 0.024677763 0.022354828  ## ManufacturingProcess24 ManufacturingProcess30 ManufacturingProcess01  ## 0.022072382 0.010941377 0.006358633 |
| coeffs.mp <- coeffs[names(coeffs.sorted[grep('ManufacturingProcess', names(coeffs.sorted))])]  coeffs.mp[coeffs.mp>0] |
| ## ManufacturingProcess32 ManufacturingProcess09 ManufacturingProcess45  ## 0.760119012 0.651939513 0.217011390  ## ManufacturingProcess34 ManufacturingProcess06 ManufacturingProcess15  ## 0.142832613 0.102493521 0.092675846  ## ManufacturingProcess04 ManufacturingProcess11 ManufacturingProcess43  ## 0.049124108 0.024677763 0.022354828  ## ManufacturingProcess30 ManufacturingProcess01  ## 0.010941377 0.006358633  coeffs.mp[coeffs.mp<0]  ## ManufacturingProcess36 ManufacturingProcess37 ManufacturingProcess17  ## -0.15603196 -0.15600377 -0.12478913  ## ManufacturingProcess13 ManufacturingProcess07 ManufacturingProcess24  ## -0.11041725 -0.03056397 -0.02207238  For the ManufacturingProcess having the negative coefficients, we would change the process so that it would decrease the Yeald. Similarly ManufacturingProcess with surge in coefficients would help in increasitng the yeald. |

# Kuhn & Johnson 7

## HW\_KJ\_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(*π**x*1*x*2)+20(*x*3−0.5)2+10*x*4+5*x*5+*N*(0,*σ*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: |
| **library**(mlbench)  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) |
|  |
| **Tune several models on these data. For example:** **library**(caret)  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.*  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)  *## RMSE Rsquared MAE*  *## 3.2040595 0.6819919 2.5683461* |
| **MARS** arsGrid <- expand.grid(degree = 1:2, nprune = seq(2, 14,by = 2))  marsModel <- train(x = trainingData$x,  y = trainingData$y,  method = 'earth',  tuneGrid = marsGrid,  trControl = trainControl(method = "cv"))  ggplot(marsModel) + labs(title = "MARS Cross-Validated RMSE Profile") + theme\_classic() |
|  |
| **SVM** svmModel <- train(x = trainingData$x,  y = trainingData$y,  method = 'svmRadial',  tuneLength = 14,  trControl = trainControl(method = "cv"))  ggplot(svmModel) + labs(title = "SVM Cross-Validated RMSE Profile") + theme\_gray() |
|  |
| **NNET** nnetGrid <- expand.grid(.size = c(1:10), .decay = c(0, 0.01, .1))  nnetModel <- train(trainingData$x,  trainingData$y,  method = "nnet",  tuneGrid = nnetGrid,  trControl = trainControl(method="cv"),  preProc = c("center", "scale"),  linout = TRUE,  trace = FALSE,  MaxNWts = 10 \* (ncol(trainingData$x) + 1) + 10 + 1,  maxit = 500)  ggplot(nnetModel) + labs(title = "NNET Cross-Validated RMSE Profile") + theme\_linedraw() |
|  |
| **Which models appear to give the best performance? Does MARS select the informative predictors (those named X1-X5)?** |
| As displayed in the summary table, MARS appears to be supreme comparing to KNN, SVM and NNET in terms of RMSE, R squared and MAE on the test set. The variable importance plot for MARS indicates that the variables X1 through 5 are the most important. X1 appears to be the most important variable. MARS appears to have successfully selected the most informative predictors and outperformed the other models in this exercise.  *# Prediction*  *# knnPred is already given to us from the textbook*  marsPred <- predict(marsModel, newdata = testData$x)  svmPred <- predict(svmModel, newdata = testData$x)  nnetPred <- predict(nnetModel, newdata = testData$x)  *# Performance*  knnPerformance <- postResample(pred = knnPred, obs = testData$y)  marsPerformance <- postResample(pred = marsPred, obs = testData$y)  svmPerformance <- postResample(pred = svmPred, obs = testData$y)  nnetPerformance <- postResample(pred = nnetPred, obs = testData$y)  *# Summary*  rbind(  *# KNN*  "KNN Train" = c("RMSE" = max(knnModel$results$RMSE),  "RSquared" = max(knnModel$results$RMSE),  "MAE" = max(knnModel$results$RMSE)),  "KNN Test" = knnPerformance,  *# MARS*  "MARS Train" = c("RMSE" = max(marsModel$results$RMSE),  "RSquared" = max(marsModel$results$Rsquared),  "MAE" = max(marsModel$results$MAE)),  "MARS Test" = marsPerformance,  *# SVM*  "SVM Train" = c(max(svmModel$results$RMSE),  max(svmModel$results$Rsquared),  max(svmModel$results$MAE)),  "SVM Test" = svmPerformance,  *# NNET*  "NNET Train" = c(max(nnetModel$results$RMSE),  max(nnetModel$results$Rsquared),  max(nnetModel$results$MAE)),  "NNET Test" = nnetPerformance) %>%  kable(caption = "Model Performance", digits = 3) %>%  kable\_styling() |
|  |



|  |
| --- |
| *# Variable Importance*  marsImportance <- caret::varImp(marsModel)  marsImportance$importance %>%  as.data.frame() %>%  tibble::rownames\_to\_column() %>%  dplyr::mutate(name = forcats::fct\_inorder(rowname)) %>%  arrange(desc(Overall)) %>%  head(10) %>%  ggplot(., aes(x = reorder(name, Overall), y = Overall)) +  geom\_point() +  geom\_segment(aes(x = name, xend = name, y = 0, yend = Overall)) +  ggtitle("Variable Importance: MARS for Simulated Data Set") +  labs(x = "Variable", y = "Importance") +  coord\_flip() +  theme\_minimal() |
|  |
|  |

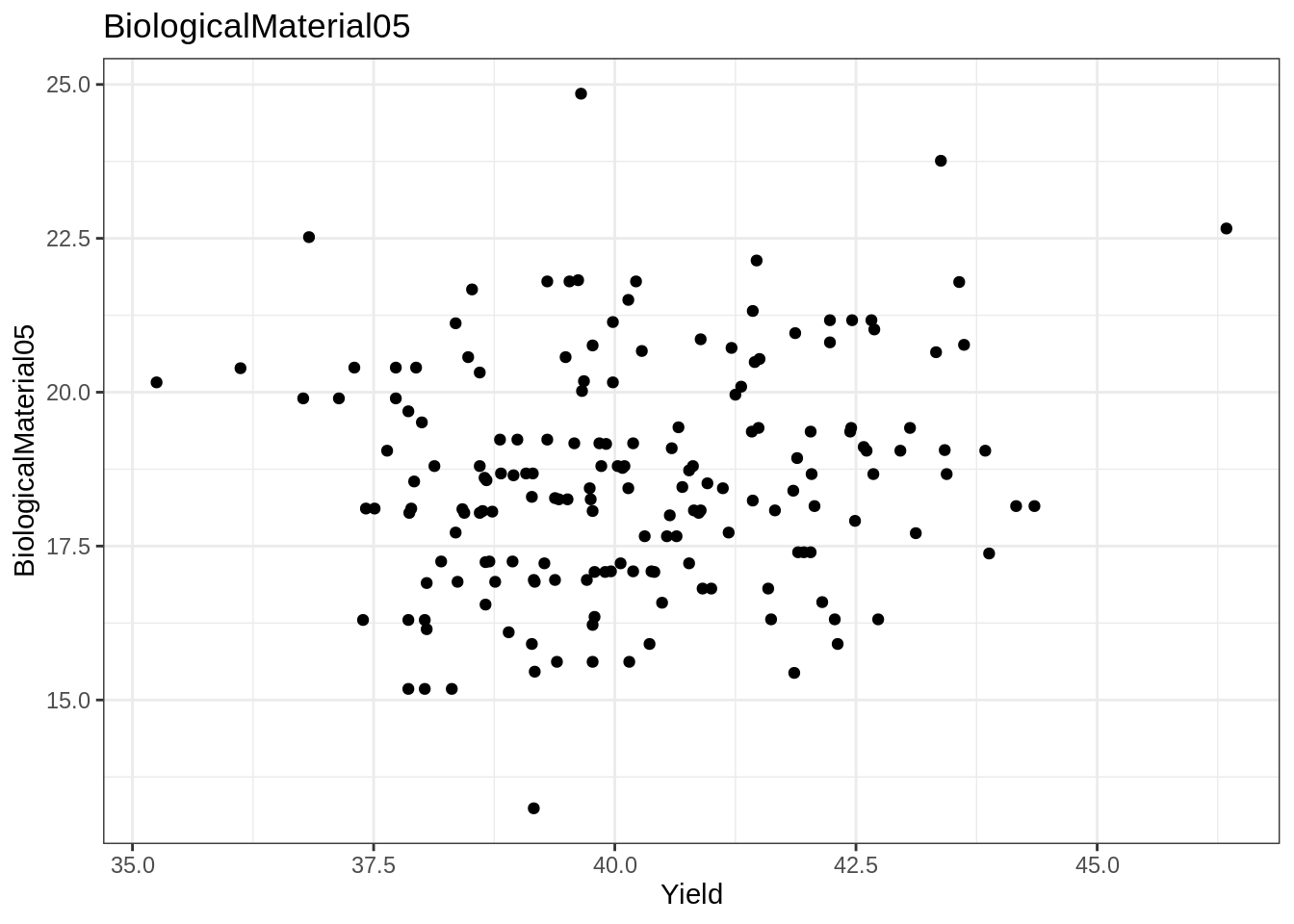
## HW\_KJ\_7.5

|  |
| --- |
| Exercise 6.3 describes data for a chemical manufacturing process. Use the same data imputation, data splitting, and preprocessing steps as before and train several nonlinear regression models. **(a). Which nonlinear regression model gives the optimal resampling and test set performance?** *# data prep - using the same methodology from exercise 6.3*  data(ChemicalManufacturingProcess)  impu\_data <- kNN(ChemicalManufacturingProcess, imp\_var = FALSE)  n <- nrow(impu\_data)  i.training <- sort(sample(n,round(n\*0.8)))  L.training <- impu\_data[i.training,]  L.test <- impu\_data[-i.training,]  *# check if any missing left*  **if**(!any(colSums(is.na(impu\_data)) %>% unlist >0)){print("there is no more missing after successful imputation")}  *## [1] "there is no more missing after successful imputation"*  *# dim*  sapply(list(impu\_data, L.training, L.test), dim) %>%  as.data.frame %>%  cbind(dim = c("# of row", "# of column")) %>%  dplyr::select(dim, imput\_data = V1, L.training = V2, L.test = V3)  *## dim imput\_data L.training L.test*  *## 1 # of row 176 141 35*  *## 2 # of column 58 58 58*  *# KNN*  knnModel2 <- train(Yield~.,  data = L.training,  method = "knn",  preProc = c("center", "scale"),  tuneLength = 10)  *# MARS*  marsGrid2 <- expand.grid(degree = 1:2, nprune = 1:20)  marsModel2 <- train(Yield~.,  data = L.training,  method = "earth",  tuneGrid = marsGrid2,  trControl = trainControl(method = "cv"))  *# SVM*  svmModel2 <- train(Yield~.,  data = L.training,  method = "svmRadial",  preProc = c("center", "scale"),  tuneLength = 14,  trControl = trainControl(method = "cv"))  *# NNET*  nnetGrid2 <- expand.grid(decay = c(0, 0.01, .1), size = c(1:10))  nnetModel2 <- train(Yield~.,  data = L.training,  method = "nnet",  tuneGrid = nnetGrid2,  trControl = trainControl(method = "cv"),  preProc = c("center", "scale"),  linout = TRUE,  trace = FALSE,  maxit = 500)  *# Prediction*  knnPred2 <- predict(knnModel2, newdata = L.test)  marsPred2 <- predict(marsModel2, newdata = L.test)  svmPred2 <- predict(svmModel2, newdata = L.test)  nnetPred2 <- predict(nnetModel2, newdata = L.test)  *# Performance*  knnPerformance2 <- postResample(pred = knnPred2, obs = L.test$Yield)  marsPerformance2 <- postResample(pred = marsPred2, obs = L.test$Yield)  svmPerformance2 <- postResample(pred = svmPred2, obs = L.test$Yield)  nnetPerformance2 <- postResample(pred = nnetPred2, obs = L.test$Yield)  *# Summary*  rbind(  *# KNN*  "KNN Train" = c("RMSE" = max(knnModel2$results$RMSE),  "RSquared" = max(knnModel2$results$Rsquared),  "MAE" = max(knnModel2$results$MAE)),  "KNN Test" = knnPerformance2,  *# MARS*  "MARS Train" = c("RMSE" = max(marsModel2$results$RMSE),  "RSquared" = max(marsModel2$results$Rsquared),  "MAE" = max(marsModel2$results$MAE)),  "MARS Test" = marsPerformance2,  *# SVM*  "SVM Train" = c(max(svmModel2$results$RMSE),  max(svmModel2$results$Rsquared),  max(svmModel2$results$MAE)),  "SVM Test" = svmPerformance2,  *# NNET*  "NNET Train" = c(max(nnetModel2$results$RMSE),  max(nnetModel2$results$Rsquared),  max(nnetModel2$results$MAE)),  "NNET Test" = nnetPerformance2) %>%  kable(caption = "Model Performance on ChemicalManufacturing Data", digits = 3) %>%  kable\_styling() |
| OLE-object |

|  |
| --- |
| **(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?** |
| svmImportance2 <- caret::varImp(svmModel2)  svmImportance2$importance %>%  as.data.frame() %>%  tibble::rownames\_to\_column() %>%  dplyr::mutate(name = forcats::fct\_inorder(rowname)) %>%  arrange(desc(Overall)) %>%  head(10) %>%  ggplot(., aes(x = reorder(name, Overall), y = Overall)) +  geom\_point() +  geom\_segment(aes(x = name, xend = name, y = 0, yend = Overall)) +  ggtitle("Variable Importance: SVM for ChemicalManufacturing Data") +  labs(x = "Variable", y = "Importance") +  coord\_flip() +  theme\_minimal() |
|  |
| **(c). Explore the relationships between the top predictors and the response for the predictors that are unique to the optimal nonlinear regression model. Do these plots reveal intuition about the biological or process predictors and their relationship with yield?** In general, the SVM model pointed to the same direction, conclusion as the linear model (Elastic net) as both indicated that the ManufacturingProcess variables seemed to be more important, e.g. ManufacturingProcess32, 09 tops the list. However, the SVM found that several biological variables important that were not found in the linear model and vice versa, e.g. BiologicalMaterial02 was found in SVM only, whereas BiologicalMaterial05 was found in the linear model only. It seemed that both variables positively correlated with Yield, but the relationship between BiologicalMaterial02 and Yield seemed to be much more obvious and stronger than 05.  features <- svmImportance2$importance %>%  as.data.frame() %>%  tibble::rownames\_to\_column() %>%  dplyr::mutate(name = forcats::fct\_inorder(rowname)) %>%  arrange(desc(Overall)) %>%  head(10) %>%  .$rowname %>%  as.vector  subset <- impu\_data %>%  dplyr::select(all\_of(c("Yield", features)))  corSubset <- cor(subset) %>%  as.matrix %>%  as.data.frame %>%  tibble::rownames\_to\_column(., "variable") %>%  arrange(desc(Yield)) %>%  dplyr::select(variable, Yield)  corSubset %>%  kable(caption = "Correlation") %>%  kable\_styling() |
|  |

|  |
| --- |
| *# BiologicalMaterial02*  impu\_data %>%  dplyr::select(Yield, BiologicalMaterial02) %>%  ggplot(., aes(Yield, BiologicalMaterial02)) +  geom\_point() +  ggtitle("BiologicalMaterial02") +  theme\_classic() |
|  |
|  |

|  |
| --- |
| *# BiologicalMaterial05*  impu\_data %>%  dplyr::select(Yield, BiologicalMaterial05) %>%  ggplot(., aes(Yield, BiologicalMaterial05)) +  geom\_point() +  ggtitle("BiologicalMaterial05") +  theme\_bw() |



# Kuhn & Johnson 8

## HW\_KJ\_8.1

|  |
| --- |
| **library**(AppliedPredictiveModeling) 8.1. Recreate the simulated data from Exercise 7.2: **library**(mlbench)  set.seed(200)  simulated <- mlbench.friedman1(200, sd = 1)  simulated <- cbind(simulated$x, simulated$y)  simulated <- as.data.frame(simulated)  colnames(simulated)[ncol(simulated)] <- "y" (a) Fit a random forest model to all of the predictors, then estimate the variable importance scores: **library**(randomForest)  ## randomForest 4.6-14  ## Type rfNews() to see new features/changes/bug fixes.  **library**(caret)  rfmodel1 <- randomForest(y ~ ., data = simulated,  importance = TRUE,  ntree = 1000)  rfImp1 <- varImp(rfmodel1, scale = FALSE)  rfImp1  ## Overall  ## V1 8.732235404  ## V2 6.415369387  ## V3 0.763591825  ## V4 7.615118809  ## V5 2.023524577  ## V6 0.165111172  ## V7 -0.005961659  ## V8 -0.166362581  ## V9 -0.095292651  ## V10 -0.074944788 Did the random forest model significantly use the uninformative predictors (V6 – V10)? The importance values of predictors (V6 - V10) are either very low and positive or very low and negative. This means that the random forest model really did not use those variables.  varImpPlot(rfmodel1) |
|  |
| (b) Now add an additional predictor that is highly correlated with one of the informative predictors. For example: simulated2 <- simulated  simulated2$duplicate1 <- simulated2$V1 + rnorm(200) \* .1  cor(simulated2$duplicate1, simulated2$V1)  ## [1] 0.9460206 Fit another random forest model to these data. Did the importance score for V1 change? What happens when you add another predictor that is also highly correlated with V1? rfmodel2 <- randomForest(y ~ ., data = simulated2,  importance = TRUE,  ntree = 1000)  rfImp2 <- varImp(rfmodel2, scale = FALSE)  rfImp2  ## Overall  ## V1 5.69119973  ## V2 6.06896061  ## V3 0.62970218  ## V4 7.04752238  ## V5 1.87238438  ## V6 0.13569065  ## V7 -0.01345645  ## V8 -0.04370565  ## V9 0.00840438  ## V10 0.02894814  ## duplicate1 4.28331581  The variable V1 importance value has been affected, It importance value has decreased. The model chose randomly which variable to use among the two highly correlated variable. There are two variables with the same type of information. This introduces instability in the model coefficients. |
|  |
| (c) Use the cforest function in the party package to fit a random forest model using conditional inference trees. The party package function varimp can calculate predictor importance. The conditional argument of that function toggles between the traditional importance measure and the modified version described in Strobl et al. (2007). Do these importances show the same pattern as the traditional random forest model? **library**(party)  bagCtrl <- cforest\_control(mtry = ncol(simulated) - 1)  baggedTree <- cforest(y ~ ., data = simulated)  **library**(caret)  crfImp <- varimp(baggedTree, conditional = TRUE)  crfImp  ## V1 V2 V3 V4 V5 V6  ## 5.471457361 5.166826657 0.020994281 6.689072245 1.256076719 0.004925215  ## V7 V8 V9 V10  ## -0.008184439 -0.009141850 -0.012594617 -0.013200299  crf.importance <- data.frame(var = names(crfImp), y =crfImp)  **library**(dplyr)  crf.imp <- crf.importance %>% arrange(desc(y))  crf.imp  ## var y  ## V4 V4 6.689072245  ## V1 V1 5.471457361  ## V2 V2 5.166826657  ## V5 V5 1.256076719  ## V3 V3 0.020994281  ## V6 V6 0.004925215  ## V7 V7 -0.008184439  ## V8 V8 -0.009141850  ## V9 V9 -0.012594617  ## V10 V10 -0.013200299  crf.imp %>% dplyr::mutate(var = reorder(var, y)) %>%  ggplot(aes(x = var, y= y)) +  geom\_point(show.legend = FALSE) +  labs(x = "Important Variables", y = NULL) +  coord\_flip() |
|  |
| (d) Repeat this process with different tree models, such as boosted trees and Cubist. Does the same pattern occur? **Boosted trees models**  **library**(gbm)  ## Loaded gbm 2.1.5  gbmModel <- gbm(y ~ ., data = simulated, distribution = "gaussian")  gbmImp <- varImp(gbmModel, numTrees = 100, scale = FALSE)  gbmImp  ## Overall  ## V1 4071.65288  ## V2 3550.71836  ## V3 1156.48989  ## V4 4416.45306  ## V5 1791.01067  ## V6 81.00046  ## V7 0.00000  ## V8 0.00000  ## V9 0.00000  ## V10 0.00000  The most important predictors are V1, V2, V3, V4, V5 The V6 predictor is less important than the first four V7, V8, V9, and V10 have not been used by the model  summary(gbmModel)  ## var rel.inf  ## V4 V4 29.3114601  ## V1 V1 27.0230635  ## V2 V2 23.5656846  ## V5 V5 11.8867193  ## V3 V3 7.6754823  ## V6 V6 0.5375902  ## V7 V7 0.0000000  ## V8 V8 0.0000000  ## V9 V9 0.0000000  ## V10 V10 0.0000000  The same pattern is repeated. The important variables are V1-V5  **Random Forest Model**  rfmodel2 <- randomForest(y ~ ., data = simulated2,  importance = TRUE,  ntree = 1000)  rfImp2 <- varImp(rfmodel2, scale = FALSE)  rfImp2  ## Overall  ## V1 5.49787495  ## V2 6.23713508  ## V3 0.63801571  ## V4 7.00094583  ## V5 2.03385902  ## V6 0.15776399  ## V7 0.01694912  ## V8 -0.05427651  ## V9 -0.02525938  ## V10 -0.07556684  ## duplicate1 4.66387457  Random Forest presents the same pattern as previous where variables V6-V7 are not important.  **Cubist Model**  simulatedX <- dplyr::select(simulated, V1:V10)  **library**(Cubist)  cubistModel <- cubist(simulatedX, simulated$y, committees = 100, neighbors = 0.01)  cubistImp <- varImp(cubistModel, scale = FALSE)  cubistImp  ## Overall  ## V1 71.5  ## V3 47.0  ## V2 58.5  ## V4 48.0  ## V5 33.0  ## V6 13.0  ## V7 0.0  ## V8 0.0  ## V9 0.0  ## V10 0.0  cubistTuned <- train(simulatedX, simulated$y, method = "cubist", )  varImp(cubistTuned)  ## cubist variable importance  ##  ## Overall  ## V1 100.00  ## V2 75.69  ## V4 68.06  ## V3 58.33  ## V5 55.56  ## V6 15.28  ## V9 0.00  ## V8 0.00  ## V10 0.00  ## V7 0.00  Cubist present the same pattern as the GBM model We observe practically the same pattern. the variables (V6 - V10)are not important, though not use by all those models. |

## HW\_KJ\_8.2

|  |
| --- |
| 8.2. Use a simulation to show tree bias with different granularities. **Generate random variables Predictors with different granularities Generate also the response variable colinear to some of them**  X1 <- sample(c(3, 5), 100, replace = TRUE)  X2 <- sample(c(3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41), 100, replace = TRUE)  X3 <- sample(10:1000/10, 100)  y <- 7\*X1 + 3\*X2 + rnorm(100)  df <- data.frame(X1, X2, X3, y)  head(df)  ## X1 X2 X3 y  ## 1 5 3 80.1 44.95411  ## 2 3 19 54.6 76.88111  ## 3 3 11 45.0 51.92110  ## 4 5 7 54.5 55.32196  ## 5 3 5 51.2 34.74864  ## 6 3 19 66.0 76.96352  **Fit a model and Compare the predictors importance**  rfBias <- randomForest(y ~ ., data = df,  importance = TRUE,  ntree = 100)  varImportance <- varImp(rfBias, scale = FALSE)  varImportance  ## Overall  ## X1 125.24888  ## X2 1536.94859  ## X3 6.56273  The predictor X2 with more unique values is more used by the random forest model than the predictor X2 with less unique values. But the variable X3 with more distinct values that is non correlated and has no relation with the response variable was also used by the model. |

## HW\_KJ\_8.3

|  |
| --- |
| 8.3. In stochastic gradient boosting the bagging fraction and learning rate will govern the construction of the trees as they are guided by the gradient. Although the optimal values of these parameters should be obtained through the tuning process, it is helpful to understand how the magnitudes of these parameters affect magnitudes of variable importance. Figure 8.24 provides the variable importance plots for boosting using two extreme values for the bagging fraction (0.1 and 0.9) and the learning rate (0.1 and 0.9) for the solubility data. The left-hand plot has both parameters set to 0.1, and the right-hand plot has both set to 0.9: |
| (a) Why does the model on the right focus its importance on just the first few of predictors, whereas the model on the left spreads importance across more predictors? The gradient boosting model employs the greedy strategy of choosing the optimal weak learner at each stage. Ridgeway (2007) suggests that small values of the learning parameter (< 0.01) work best. High values of learning rate will result in increasing the contribution of few variables at each stage of the process and then shrinking the number of important variables.  Learning rate controls the fraction of the predictions of each tree being added. A higher learning rate means that larger fraction of each tree’s predictions are added to the final prediction. This effectively means that a higher learning rate increases the dependent / correlation structure. More of the same predictors will be selected among the trees. This is why the right-hand plot has its importance focus on just the first few of the predictors, and look very steep.  Bagging fraction is the fraction of data being used in each iteration of the trees. When you have a small bagging fraction, say 0.1, on each iteration just 10% of the full data is randomly sampled. So each tree may be built using very different dataset. Since the dataset are very different, the trees will be splitting very differently from each other. On the contrast, when you have large bagging fraction, say 0.9, essentially on each iteration the trees are seeing the same dataset - they will likely split similarly. This means that larger bagging fraction increases the dependent / correlated structure in the boosting trees. Therefore, the right-hand plot with a larger bagging fraction has its importance focus on just the first few of the predictors. |
| (b) Which model do you think would be more predictive of other samples? Greedy models are less likely to select the optimal global model and are prone to over-fitting. Stochastic models reduce prediction variance. Therefore, the less greedy model on the left (with a 0.1 learning rate) that is also more random (due to only selecting 0.1 of the training set observations to propose the next tree in the expansion) would be more predictive of other samples.  Learning rate and bagging fraction are important parameters to control the overfitting of the gradient boosting model that requires tuning. A smaller learning rate and bagging fraction leads to better generalization ability over unseen samples. If I have to guess, the model with 0.1 learning rate and bagging fraction will be more predictive of out of bag samples. However, since this invovles a trade off between bias-variance, I can only confirm using cross validation or a test set. (c) How would increasing interaction depth affect the slope of predictor importance for either model in Fig. 8.24 The author comment stated that the larger value of shrinkage has an impact on reducting RMSE for all choises of tree depth and number of trees.  Figure 8.20 presents the cross-validated RMSE results for boosted trees across tuning parameters of tree depth (1–7), number of trees (100–1,000), and shrinkage (0.01 or 0.1); the bagging fraction in this illustration was fixed at 0.5. When examining this figure, the larger value of shrinkage (right-hand plot) has an impact on reducing RMSE for all choices of tree depth and number of trees. Also, RMSE decreases as tree depth increases when shrinkage is 0.01. The same pattern holds true for RMSE when shrinkage is 0.1 and the number of trees is less than 300. Using the one-standard-error rule, the optimal boosted tree has depth 3 with 400 trees and shrinkage of 0.1. |

## HW\_KJ\_8.7

|  |
| --- |
| **8.7. Refer to Exercises 6.3 and 7.5 which describe a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several tree-based models:** data("ChemicalManufacturingProcess") **Preprocessing** *# data prep - using the same methodology from exercise 6.3*  data(ChemicalManufacturingProcess)  impu\_data <- kNN(ChemicalManufacturingProcess, imp\_var = FALSE)  n <- nrow(impu\_data)  i.trainset <- sort(sample(n,round(n\*0.8)))  L.trainset <- impu\_data[i.trainset,]  L.testset <- impu\_data[-i.trainset,]  *# check if any missing left*  **if**(!any(colSums(is.na(impu\_data)) %>% unlist >0)){print("there is no more missing after successful imputation")}  *## [1] "there is no more missing after successful imputation"*  *# dim*  sapply(list(impu\_data, L.trainset, L.testset), dim) %>%  as.data.frame %>%  cbind(dim = c("# of row", "# of column")) %>%  dplyr::select(dim, imput\_data = V1, L.trainset = V2, L.testset = V3)  *## dim imput\_data L.trainset L.testset*  *## 1 # of row 176 141 35*  *## 2 # of column 58 58 58*  X\_train <- L.trainset[,-1]  y\_train <- L.trainset[,1]  X\_test <- L.testset[,-1]  y\_test <- L.testset[,1] **Fitting Models****Bagged Cart** **library**(ipred)  **library**(e1071)  baggedCart <- train(X\_train, y\_train, method = "treebag") Random Forest **library**(randomForest)  *## randomForest 4.6-14*  *## Type rfNews() to see new features/changes/bug fixes.*  *##*  *## Attaching package: 'randomForest'*  *## The following object is masked from 'package:dplyr':*  *##*  *## combine*  *## The following object is masked from 'package:ggplot2':*  *##*  *## margin*  randomForest <- train(x = X\_train,  y = y\_train,  method = "rf",  tuneLength = 10,  trControl = trainControl(method = "cv")) Cubist **library**(Cubist)  cubeModel <- train(X\_train,  y\_train,  method = "cubist")  cubeModel  *## Cubist*  *##*  *## 141 samples*  *## 57 predictor*  *##*  *## No pre-processing*  *## Resampling: Bootstrapped (25 reps)*  *## Summary of sample sizes: 141, 141, 141, 141, 141, 141, ...*  *## Resampling results across tuning parameters:*  *##*  *## committees neighbors RMSE Rsquared MAE*  *## 1 0 1.947816 0.3081246 1.4003082*  *## 1 5 1.924566 0.3151805 1.3686542*  *## 1 9 1.935386 0.3099767 1.3825032*  *## 10 0 1.292321 0.5262281 0.9936424*  *## 10 5 1.262906 0.5447284 0.9651846*  *## 10 9 1.273433 0.5386535 0.9740627*  *## 20 0 1.257526 0.5408880 0.9612249*  *## 20 5 1.220897 0.5638929 0.9248384*  *## 20 9 1.234052 0.5558137 0.9362896*  *##*  *## RMSE was used to select the optimal model using the smallest value.*  *## The final values used for the model were committees = 20 and neighbors = 5.* Stochastic Gradient Boosting **library**(gbm)  *## Loaded gbm 2.1.5*  gbmModel <- train(X\_train,  y\_train,  method = "gbm",  tuneGrid = expand.grid(n.trees = c(50, 100, 150, 200),  interaction.depth = c(1,5,10,15),  shrinkage = c(0.01, 0.1,0.5),  n.minobsinnode = c(5,10,15)),  verbose = FALSE) Prediction cartPred <- predict(baggedCart, newdata = X\_test)  rfPred <- predict(randomForest, newdata = X\_test)  cubPred <- predict(cubeModel, newdata = X\_test)  gbmPred <- predict(gbmModel, newdata = X\_test) Performance cartPerformance <- postResample(pred = cartPred, obs = y\_test)  rfPerformance <- postResample(pred = rfPred, obs = y\_test)  cubPerformance <- postResample(pred = cubPred, obs = y\_test)  gbmPerformance <- postResample(pred = gbmPred, obs = y\_test) Summary rbind(  *# Bagged Cart*  "Cart Train" = c("RMSE" = max(baggedCart$results$RMSE),  "RSquared" = max(baggedCart$results$Rsquared),  "MAE" = max(baggedCart$results$MAE)),  "Cart Test" = cartPerformance,  *# Random Forest*  "RF Train" = c("RMSE" = max(randomForest$results$RMSE),  "RSquared" = max(randomForest$results$Rsquared),  "MAE" = max(randomForest$results$MAE)),  "RF Test" = rfPerformance,  *# Cubist*  "Cubist Train" = c(max(cubeModel$results$RMSE),  max(cubeModel$results$Rsquared),  max(cubeModel$results$MAE)),  "Cubist Test" = cubPerformance,  *# GBM model*  "GBM Train" = c(max(gbmModel$results$RMSE),  max(gbmModel$results$Rsquared),  max(gbmModel$results$MAE)),  "GBM Test" = gbmPerformance) %>%  kable(caption = "Model Performance on ChemicalManufacturing Data", digits = 3) %>%  kable\_styling() |

# OLE-object

|  |
| --- |
| **(a) Which tree-based regression model gives the optimal resampling and test set performance?** With RMSE, Cubist has the best result on testing set but a worse result on training set. It is the only one that can explain the data at 82%. |
| **(b) Which predictors are most important in the optimal tree-based regression model? Do either the biological or process variables dominate the list? How do the top 10 important predictors compare to the top 10 predictors from the optimal linear and nonlinear models?** cubistImportance <- caret::varImp(cubeModel)  cubistImportance$importance %>%  as.data.frame() %>%  tibble::rownames\_to\_column() %>%  dplyr::mutate(name = forcats::fct\_inorder(rowname)) %>%  arrange(desc(Overall)) %>%  head(10) %>%  ggplot(., aes(x = reorder(name, Overall), y = Overall)) +  geom\_point() +  geom\_segment(aes(x = name, xend = name, y = 0, yend = Overall)) +  ggtitle("Variable Importance: SVM for ChemicalManufacturing Data") +  labs(x = "Variable", y = "Importance") +  coord\_flip() +  theme\_minimal() |

# 

|  |
| --- |
| Manufacturing Process dominate biological Material but not totally. There is an alternance variable importance between the two type of variables. We obserserve with variable importance on Cubist model that the first two important variable are Manufacturing Process, the second two are Biological Material and so on. |
| **(c) Plot the optimal single tree with the distribution of yield in the terminal nodes. Does this view of the data provide additional knowledge about the biological or process predictors and their relationship with yield?** **library**(partykit)  ## Loading required package: libcoin  ## Loading required package: mvtnorm  **library**(rpart)  rpartTree <- rpart(Yield~., data = L.trainset)  rpart\_tree <-as.party(rpartTree)  plot(rpart\_tree) |
| The variables Manufacturing Process dominate the graph. They are mostly used by the model. |

## Market basket analysis / recommender systems

|  |
| --- |
| Introduction I am assigning one simple problem on market basket analysis / recommender systems.  Imagine 10000 receipts sitting on your table. Each receipt represents a transaction with items that were purchased. The receipt is a representation of stuff that went into a customer’s basket – and therefore ‘Market Basket Analysis’.  That is exactly what the Groceries Data Set contains: a collection of receipts with each line representing 1 receipt and the items purchased. Each line is called a transaction and each column in a row represents an item.  Here is the dataset = GroceryDataSet.csv (comma separated file) You assignment is to use R to mine the data for association rules. You should report support, confidence and lift and your top 10 rules by lift. Turn in as you would the other problems from HA and KJ. Due 07/18/20 with the packaged set, HW #2. |

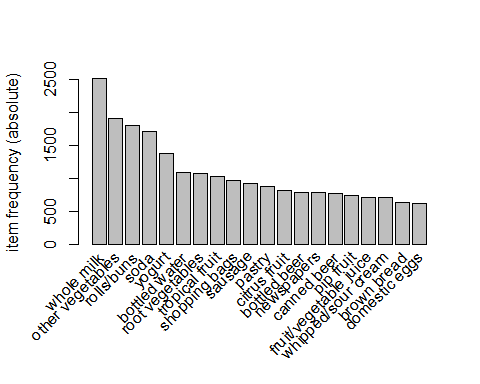
|  |
| --- |
| MARKET BASKET ANALYSIS |

|  |  |
| --- | --- |
| Data preprocessing and exploring ## citrus.fruit semi.finished.bread margarine ## 1 tropical fruit yogurt coffee ## 2 whole milk  ## 3 pip fruit yogurt cream cheese  ## 4 other vegetables whole milk condensed milk ## 5 whole milk butter yogurt ## 6 rolls/buns  ## ready.soups | ## 1  ## 2  ## 3 meat spreads  ## 4 long life bakery product  ## 5 rice abrasive cleaner  ## 6 |

### 

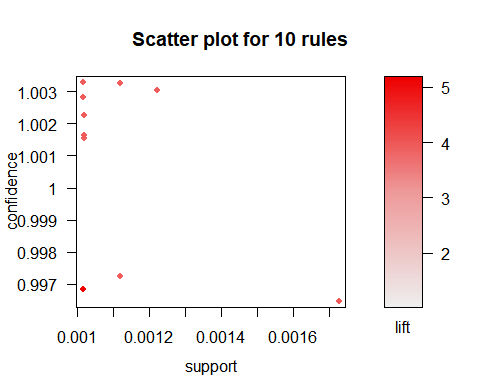
|  |
| --- |
| ## citrus.fruit semi.finished.bread margarine  ## sausage : 825 :2159 :3802  ## whole milk : 717 whole milk : 654 whole milk : 506  ## frankfurter : 580 other vegetables: 550 other vegetables: 415  ## tropical fruit : 482 root vegetables : 383 rolls/buns : 293  ## other vegetables: 460 rolls/buns : 378 yogurt : 289  ## citrus fruit : 452 tropical fruit : 355 soda : 229  ## (Other) :6318 (Other) :5355 (Other) :4300 |
| ## ready.soups X X.1  ## :5101 :6105 :6960  ## whole milk : 315 rolls/buns : 176 soda : 150  ## other vegetables: 254 soda : 168 rolls/buns : 146  ## rolls/buns : 238 yogurt : 160 shopping bags: 107  ## soda : 211 whole milk : 149 bottled water: 95  ## yogurt : 202 shopping bags: 145 yogurt : 93  ## (Other) :3513 (Other) :2931 (Other) :2283  ## X.2 X.3 X.4  ## :7605 :8150 :8588  ## soda : 120 shopping bags: 76 soda : 61  ## shopping bags: 107 bottled water: 68 shopping bags : 56  ## rolls/buns : 92 newspapers : 66 fruit/vegetable juice: 55  ## newspapers : 68 rolls/buns : 59 bottled water : 54  ## domestic eggs: 57 soda : 59 newspapers : 51  ## (Other) :1785 (Other) :1356 (Other) : 969  ## X.5 X.6  ## :8938 :9184  ## shopping bags : 49 shopping bags: 40  ## soda : 39 newspapers : 36  ## fruit/vegetable juice: 34 pastry : 27  ## newspapers : 33 bottled water: 25  ## bottled water : 26 napkins : 23  ## (Other) : 715 (Other) : 499  ## X.7 X.8  ## :9366 :9483  ## soda : 30 soda : 24  ## shopping bags : 19 shopping bags : 18  ## chocolate : 17 fruit/vegetable juice: 16  ## fruit/vegetable juice: 17 napkins : 14  ## napkins : 17 newspapers : 14  ## (Other) : 368 (Other) : 265  ## X.9 X.10  ## :9561 :9638  ## shopping bags : 18 shopping bags : 16  ## fruit/vegetable juice: 17 napkins : 13  ## newspapers : 14 fruit/vegetable juice: 11  ## soda : 14 hygiene articles : 11  ## napkins : 11 candy : 9  ## (Other) : 199 (Other) : 136  ## X.11 X.12  ## :9693 :9739  ## shopping bags : 11 napkins : 8  ## napkins : 9 chocolate : 5  ## chocolate : 8 newspapers : 5  ## hygiene articles : 6 candy : 4  ## long life bakery product: 6 fruit/vegetable juice: 4  ## (Other) : 101 (Other) : 69  ## X.13 X.14 X.15  ## :9768 :9782 :9796  ## candy : 5 detergent : 4 bottled beer : 3  ## chocolate : 5 fruit/vegetable juice: 4 napkins : 3  ## napkins : 5 shopping bags : 4 pot plants : 3  ## newspapers : 3 chocolate : 3 candy : 2  ## bottled water: 2 bottled water : 2 hygiene articles: 2  ## (Other) : 46 (Other) : 35 (Other) : 25  ## X.16 X.17  ## :9805 :9816  ## napkins : 4 napkins : 2  ## fruit/vegetable juice : 2 baking powder : 1  ## house keeping products: 2 bottled beer : 1  ## hygiene articles : 2 cleaner : 1  ## candles : 1 cling film/bags: 1  ## (Other) : 18 (Other) : 12  ## X.18 X.19  ## :9820 :9826  ## waffles : 2 bottled beer : 2  ## chocolate marshmallow: 1 bottled water: 1  ## cling film/bags : 1 cake bar : 1  ## dental care : 1 coffee : 1  ## dog food : 1 flour : 1  ## (Other) : 8 (Other) : 2  ## X.20 X.21  ## :9827 :9827  ## chocolate : 2 chocolate : 1  ## fruit/vegetable juice : 1 female sanitary products: 1  ## liquor (appetizer) : 1 long life bakery product: 1  ## long life bakery product: 1 margarine : 1  ## pasta : 1 rum : 1  ## white wine : 1 (Other) : 2  ## X.22 X.23  ## :9828 :9829  ## abrasive cleaner : 1 chocolate : 1  ## chocolate : 1 hygiene articles: 1  ## hygiene articles : 2 napkins : 2  ## long life bakery product: 1 sugar : 1  ## specialty fat : 1  ##  ## X.24 X.25 X.26  ## :9830 :9833 :9833  ## cooking chocolate : 1 skin care: 1 hygiene articles: 1  ## house keeping products: 2  ## soups : 1  ##  ##  ##  ## X.27  ## :9833  ## candles: 1  ## |

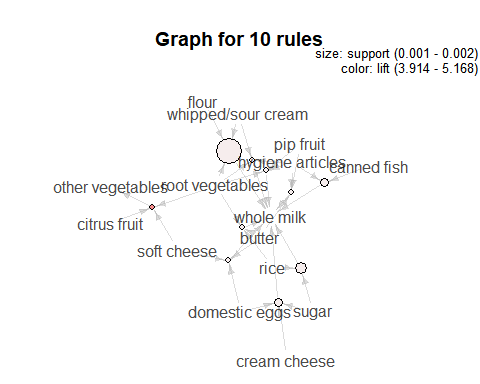
|  |
| --- |
| Association rules for the grocery store The grocery dataset shows the items that have been bought togheter. Now we going to see how often they are bought together , and rules.  We can then see how many transactions we have and what they are.  ## [1] "Description of the transactions"  ## transactions in sparse format with ## 9835 transactions (rows) and ## 169 items (columns)  ## transactions as itemMatrix in sparse format with ## 9835 rows (elements/itemsets/transactions) and ## 169 columns (items) and a density of 0.02609146  ##  ## most frequent items: ## whole milk other vegetables rolls/buns soda  ## 2513 1903 1809 1715  ## yogurt (Other)  ## 1372 34055  ##  ## element (itemset/transaction) length distribution: ## sizes ## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  ## 2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55  ## 16 17 18 19 20 21 22 23 24 26 27 28 29 32  ## 46 29 14 14 9 11 4 6 1 1 1 1 3 1  ##  ## Min. 1st Qu. Median Mean 3rd Qu. Max.  ## 1.000 2.000 3.000 4.409 6.000 32.000  ##  ## includes extended item information - examples: ## labels ## 1 abrasive cleaner ## 2 artif. sweetener ## 3 baby cosmetics  We see 9835 transactions, this is the number of rows as well, and 169 items.  The summary gives us some useful information:   * density: The percentage of non-empty cells in the sparse matrix. In another word, the total number of items that was purchased divided by the total number of possible items in that matrix. * Transactions sizes There are 2159 transactions with one item, 1643 with 2 items, 1299 with 3 items, 1299 with items. This indicates that most customers buy small number of items on each purchase. * The data distribution is right skewed. * Let’s have a look item freqnency plot. |

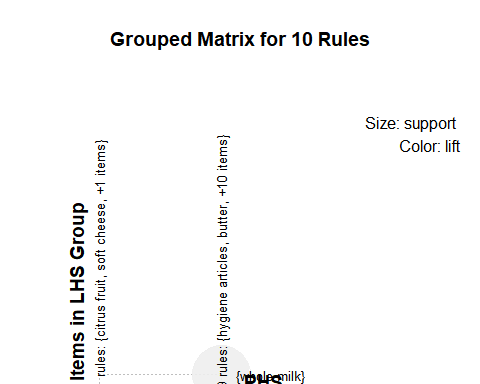


|  |
| --- |
| Create some rules  * We use the Apriori algorithm in arules library to mine frequent itemsets and association rules. The algorithm employs level-wise search for frequent itemsets. * We pass supp=0.001 and conf=0.8 to return all the rules have a support of at least 0.1% and confidence of at least 80%. * We sort the rules by decreasing confidence. * Have a look the summary of the rules.   ## Apriori ##  ## Parameter specification: ## confidence minval smax arem aval originalSupport maxtime support minlen ## 0.8 0.1 1 none FALSE TRUE 5 0.001 1 ## maxlen target ext ## 10 rules FALSE |

|  |
| --- |
| ##  ## Algorithmic control: ## filter tree heap memopt load sort verbose ## 0.1 TRUE TRUE FALSE TRUE 2 TRUE ##  ## Absolute minimum support count: 9  ##  ## set item appearances ...[0 item(s)] done [0.00s]. ## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s]. ## sorting and recoding items ... [157 item(s)] done [0.00s]. ## creating transaction tree ... done [0.00s]. ## checking subsets of size 1 2 3 4 5 6 done [0.02s]. ## writing ... [410 rule(s)] done [0.00s]. ## creating S4 object ... done [0.00s].  ## set of 410 rules ##  ## rule length distribution (lhs + rhs):sizes ## 3 4 5 6  ## 29 229 140 12  ##  ## Min. 1st Qu. Median Mean 3rd Qu. Max.  ## 3.000 4.000 4.000 4.329 5.000 6.000  ##  ## summary of quality measures: ## support confidence lift count  ## Min. :0.001017 Min. :0.8000 Min. : 3.131 Min. :10.00  ## 1st Qu.:0.001017 1st Qu.:0.8333 1st Qu.: 3.312 1st Qu.:10.00  ## Median :0.001220 Median :0.8462 Median : 3.588 Median :12.00  ## Mean :0.001247 Mean :0.8663 Mean : 3.951 Mean :12.27  ## 3rd Qu.:0.001322 3rd Qu.:0.9091 3rd Qu.: 4.341 3rd Qu.:13.00  ## Max. :0.003152 Max. :1.0000 Max. :11.235 Max. :31.00  ##  ## mining info: ## data ntransactions support confidence ## transactions 9835 0.001 0.8   * The number of rules: 9835. * The distribution of rules by length: Most rules are 4 items long. * The summary of quality measures: ranges of support, confidence, and lift. * The information on the data mining: total data mined, and minimum parameters we set earlier.   let’s inspect top 10.  ## lhs rhs support confidence lift count ## [1] {rice,  ## sugar} => {whole milk} 0.001220132 1 3.913649 12 ## [2] {canned fish,  ## hygiene articles} => {whole milk} 0.001118454 1 3.913649 11 ## [3] {butter,  ## rice,  ## root vegetables} => {whole milk} 0.001016777 1 3.913649 10 ## [4] {flour,  ## root vegetables,  ## whipped/sour cream} => {whole milk} 0.001728521 1 3.913649 17 ## [5] {butter,  ## domestic eggs,  ## soft cheese} => {whole milk} 0.001016777 1 3.913649 10 ## [6] {citrus fruit,  ## root vegetables,  ## soft cheese} => {other vegetables} 0.001016777 1 5.168156 10 ## [7] {butter,  ## hygiene articles,  ## pip fruit} => {whole milk} 0.001016777 1 3.913649 10 ## [8] {hygiene articles,  ## root vegetables,  ## whipped/sour cream} => {whole milk} 0.001016777 1 3.913649 10 ## [9] {hygiene articles,  ## pip fruit,  ## root vegetables} => {whole milk} 0.001016777 1 3.913649 10 ## [10] {cream cheese,  ## domestic eggs,  ## sugar} => {whole milk} 0.001118454 1 3.913649 11   * 100% customers who bought {canned fish,hygiene articles} end up bought {whole milk} as well. * 100% customers who bought {rice, sugar} end up bought {whole milk} as well. |
| Below are the plot of these top 10 rules. |







In this post, we have learned how to Perform Market Basket Analysis in R and how to interpret the results.

reference: [R and Data Mining](http://www.rdatamining.com/examples/association-rules)