EX6

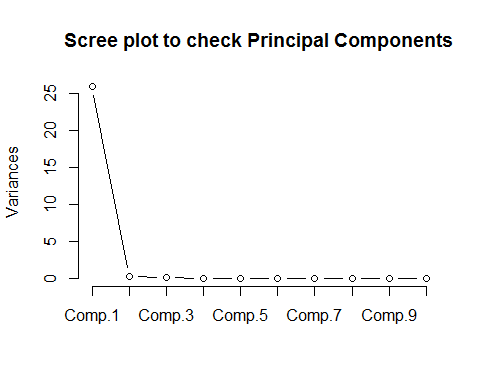
Chathura J Gunasekara

Thursday, October 16, 2014

1. a. Load the data

1. Use PCA to determine the effective dimension of the data which is number of PCA Componenents. There are two PCA compnenets which is the effective dimention.

## [1] Using Scree Plot to check number of effective compents



Effective Dimension = 2

## PC1 PC2  
## [1,] -4.3037 -0.4040  
## [2,] 0.9983 0.4865  
## [3,] -7.1414 1.4104  
## [4,] -1.8086 1.1541  
## [5,] 0.9883 -1.1921  
## [6,] 5.5620 0.2265

c.Using the created 2 PCs a traing and testing set is created.

## V1 PC1 PC2  
## 1 22.5 -4.3037 -0.4040  
## 2 40.1 0.9983 0.4865  
## 3 8.4 -7.1414 1.4104  
## 4 5.9 -1.8086 1.1541  
## 5 25.5 0.9883 -1.1921  
## 6 42.7 5.5620 0.2265

1. Ordinary Linear Regression

##   
## Call:  
## lm(formula = V1 ~ ., data = newdata)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -21.36 -7.01 -3.23 5.55 30.95   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 18.142 0.763 23.78 < 2e-16 \*\*\*  
## PC1 0.563 0.077 7.31 5.5e-12 \*\*\*  
## PC2 -2.713 0.776 -3.49 0.00058 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 11.2 on 212 degrees of freedom  
## Multiple R-squared: 0.236, Adjusted R-squared: 0.229   
## F-statistic: 32.8 on 2 and 212 DF, p-value: 3.9e-13

## [1] Testing data

## RMSE Rsquared   
## 11.2484 0.1804

|  |  |  |
| --- | --- | --- |
| LM | Training Data | Testing Data |
| RMSE | 11.2 | 11.2484 |
| R-squared | 0.236 | 0.1804 |

1. Robust Linear Regresssion

##   
## Call: rlm(formula = V1 ~ ., data = newdata)  
## Residuals:  
## Min 1Q Median 3Q Max   
## -22.82 -5.36 -1.90 6.61 32.47   
##   
## Coefficients:  
## Value Std. Error t value  
## (Intercept) 16.897 0.714 23.649   
## PC1 0.598 0.072 8.286   
## PC2 -3.147 0.727 -4.327   
##   
## Residual standard error: 8.45 on 212 degrees of freedom

## [1] Testing data

## RMSE Rsquared   
## 11.2518 0.1778

|  |  |  |
| --- | --- | --- |
| RLM | Training Data | Testing Data |
| RMSE | 8.45 | 11.2518 |
| R-squared | - | 0.1778 |

1. Partial Least Squares

##   
## Attaching package: 'pls'  
##   
## The following object is masked from 'package:caret':  
##   
## R2  
##   
## The following object is masked from 'package:stats':  
##   
## loadings

## Data: X dimension: 215 2   
## Y dimension: 215 1  
## Fit method: kernelpls  
## Number of components considered: 2  
## TRAINING: % variance explained  
## 1 comps 2 comps  
## X 99.02 100.00  
## V1 19.32 23.63

## [1] Testing data

## , , 1 comps  
##   
## V1  
## 12 9.787  
## 14 20.600  
## 15 22.573  
## 19 18.874  
## 24 25.275  
##   
## , , 2 comps  
##   
## V1  
## 12 12.24  
## 14 19.71  
## 15 22.05  
## 19 21.43  
## 24 25.75

## Partial Least Squares   
##   
## 215 samples  
## 2 predictor  
##   
## Pre-processing: centered, scaled   
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 215, 215, 215, 215, 215, 215, ...   
##   
## Resampling results  
##   
## RMSE Rsquared RMSE SD Rsquared SD  
## 11 0.2 0.7 0.06   
##   
## Tuning parameter 'ncomp' was held constant at a value of 1  
## Testing set:

## RMSE Rsquared

## 13.2599847 0.0688547

|  |  |  |
| --- | --- | --- |
| PLS | Training Data | Testing Data |
| RMSE | 11 | 13.2599 |
| R-squared | 0.2 | 0. 0688547 |

iv.ridge regresssion

## Loading required package: lars  
## Loaded lars 1.2

## 12 14 15 19 24 25   
## 12.78 20.09 22.56 22.31 26.52 20.78

## Ridge Regression   
##   
## 161 samples  
## 2 predictor  
##   
## Pre-processing: centered, scaled   
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 161, 161, 161, 161, 161, 161, ...   
##   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared RMSE SD Rsquared SD  
## 0.000 11 0.3 0.7 0.08   
## 0.007 11 0.3 0.7 0.08   
## 0.014 11 0.3 0.7 0.08   
## 0.021 11 0.3 0.7 0.08   
## 0.029 11 0.3 0.7 0.08   
## 0.036 11 0.3 0.7 0.08   
## 0.043 11 0.3 0.7 0.08   
## 0.050 11 0.3 0.7 0.08   
## 0.057 11 0.3 0.7 0.08   
## 0.064 11 0.3 0.7 0.08   
## 0.071 11 0.3 0.7 0.08   
## 0.079 11 0.3 0.7 0.08   
## 0.086 11 0.3 0.7 0.08   
## 0.093 11 0.3 0.7 0.08   
## 0.100 11 0.3 0.7 0.08   
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.1.

# For Testing set

## RMSE Rsquared

13.41521188 0.06744405

|  |  |  |
| --- | --- | --- |
| PLS | Training Data | Testing Data |
| RMSE | 11 | 13.41521188 |
| R-squared | 0.3 | 0.06744405 |

v.lasso model

## [1] "s" "fraction" "mode" "fit"

## 12 14 15 19 24 25   
## 17.07 18.82 19.14 18.54 19.57 18.46

## PC1 PC2   
## 0.09093 0.00000

|  |  |  |
| --- | --- | --- |
| Lasso | Training Data | Testing Data |
| RMSE |  |  |
| R-squared |  |  |

vi. Elasticnet Model

#Call:

#enet(x = as.matrix(train[, 2:3]), y = train$V1, lambda = 0.01,

# normalize = TRUE)

#Sequence of moves:

# PC1 PC2

#Var 1 2 3

#Step 1 2 3

Train:

RMSE Rsquared

11.9736733 0.2735589

Test:

# RMSE Rsquared

# 13.39399783 0.04264157

|  |  |  |
| --- | --- | --- |
| Elasticnet | Training Data | Testing Data |
| RMSE | 11.9736733 | 13.39399783 |
| R-squared | 0.2735589 | 0.04264157 |

(d)For this data set non of the models are siginificantly better or worse than others.A robust linear model with PCA should work because its interpretablity.

1. Robust Linear model will be used because it is easier to implement and give higer R-squred values for the fit and lower RMSE for testing data.

2. a.Loading data

1. Remove the near zero variance predictors.

388

## [1] New Dimension of the fingerprint is:

## [1] 165 388

Effective Dimension is 388

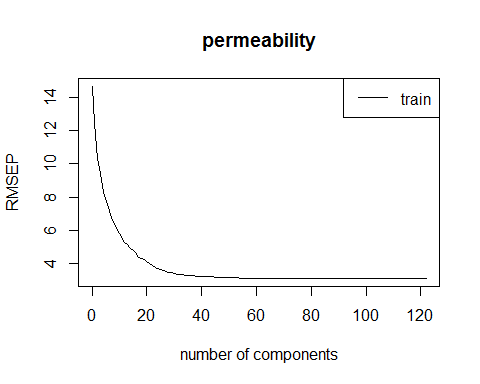
C. Split the data and train a PLS model

75 % for Training set

25% for Testing set

**Preprocessing = "center", and "scale"**

## Data: X dimension: 123 388   
## Y dimension: 123 1  
## Fit method: kernelpls  
## Number of components considered: 122



40 variables are optimal according the above graph as it will explain 91% of the variation in the data.

## Partial Least Squares   
##   
## 123 samples  
## 388 predictors  
##   
## Pre-processing: centered, scaled   
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 123, 123, 123, 123, 123, 123, ...   
##   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared RMSE SD Rsquared SD  
## 1 13 0.3 2 0.1   
## 2 12 0.4 2 0.1   
## 3 12 0.4 2 0.1   
## 4 12 0.4 2 0.1   
## 5 12 0.4 2 0.2   
## 6 12 0.4 2 0.1   
## 7 12 0.4 2 0.1   
## 8 12 0.4 2 0.1   
## 9 12 0.4 2 0.1   
## 10 12 0.4 2 0.2   
## 11 12 0.4 2 0.2   
## 12 12 0.4 2 0.2   
## 13 12 0.4 2 0.2   
## 14 12 0.4 2 0.2   
## 15 13 0.4 2 0.2   
## 16 13 0.4 2 0.2   
## 17 13 0.4 2 0.2   
## 18 13 0.4 2 0.2   
## 19 14 0.4 2 0.2   
## 20 14 0.4 2 0.2   
## 21 14 0.4 2 0.2   
## 22 14 0.3 3 0.2   
## 23 14 0.3 3 0.2   
## 24 15 0.3 3 0.2   
## 25 15 0.3 3 0.2   
## 26 15 0.3 3 0.2   
## 27 15 0.3 3 0.2   
## 28 15 0.3 3 0.2   
## 29 16 0.3 3 0.2   
## 30 16 0.3 3 0.1   
## 31 16 0.3 3 0.1   
## 32 16 0.3 3 0.1   
## 33 16 0.3 3 0.1   
## 34 17 0.3 3 0.1   
## 35 17 0.3 3 0.1   
## 36 17 0.2 3 0.1   
## 37 17 0.2 3 0.1   
## 38 17 0.2 3 0.1   
## 39 18 0.2 4 0.1   
## 40 18 0.2 4 0.1   
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 6.

d.

predTest <- predict(plsFit, test, ncomp=40)

plsSum <- data.frame(obs = test$permeability, pred = predTest)

e. i.Apply robust liner model

## Robust Linear Model   
##   
## 123 samples  
## 388 predictors  
##   
## Pre-processing: principal component signal extraction, scaled, centered   
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 123, 123, 123, 123, 123, 123, ...   
##   
## Resampling results  
##   
## RMSE Rsquared RMSE SD Rsquared SD  
## 13 0.3 2 0.1   
##   
##

f. No. Because the sample size is small. The models are not good enough.

3.

a.Load Data

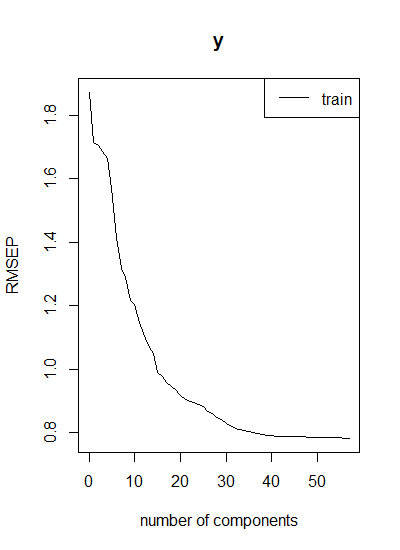
b.Impute missing values

library(impute)

x.imputed <- impute.knn(as.matrix(x), k=5)

dataset <-cbind(y,x.imputed$data)

Split the data in to training and testing set. Train a PLS model, from the above figure about 40 components have optimal RMSE.



d.

plsFitPred <- predict(plsFit,test,ncomp=40)

plsSum <- data.frame(obs = test$y, pred = plsFitPred)

obs y.40.comps

3 42.03 43.00921

5 42.49 41.29671

6 43.57 41.82107

10 42.45 42.44659

12 42.68 42.24415

15 41.50 40.47215

32 41.87 40.94658

36 40.87 42.11370

39 42.23 43.38682

55 39.77 41.08004

57 39.14 39.65448

59 42.31 42.06472

60 40.49 41.19709

66 42.03 41.49470

68 41.85 39.63101

69 39.71 39.95932

71 39.16 39.08638

76 38.76 40.18474

82 41.25 40.59070

85 40.91 39.45312

90 39.27 37.59883

92 39.17 39.69051

95 40.77 40.98456

96 39.86 39.78961

100 37.73 38.69308

101 37.30 38.62855

107 39.42 38.91600

115 41.86 39.61187

116 42.15 37.92068

117 43.88 38.53108

118 39.58 38.55666

122 40.66 39.75220

129 42.73 41.84246

135 37.86 39.50506

136 38.03 41.07014

142 40.31 40.43579

144 40.64 40.17808

146 38.13 37.74891

148 39.14 38.70629

154 37.51 38.18449

164 38.67 42.02360

165 38.42 40.61114

167 38.82 37.52129

172 39.66 36.82774

e.

library(caret)

plsImp <- varImp(plsFit,useModel=TRUE,scale=FALSE)

plsImp

Overall

BiologicalMaterial01 0.0037636251

BiologicalMaterial02 0.0061627042

BiologicalMaterial03 0.0074523767

BiologicalMaterial04 0.0056757466

BiologicalMaterial05 0.0043282008

BiologicalMaterial06 0.0064829052

BiologicalMaterial07 0.0096930901

BiologicalMaterial08 0.0061004835

BiologicalMaterial09 0.0032605497

BiologicalMaterial10 0.0054176680

BiologicalMaterial11 0.0037373796

BiologicalMaterial12 0.0053204056

ManufacturingProcess01 0.0043895915

ManufacturingProcess02 0.0028928196

ManufacturingProcess03 0.0016489279

ManufacturingProcess04 0.0032304816

ManufacturingProcess05 0.0018557470

ManufacturingProcess06 0.0055780143

ManufacturingProcess07 0.0048561644

ManufacturingProcess08 0.0042153086

ManufacturingProcess09 0.0099172028

ManufacturingProcess10 0.0021014055

ManufacturingProcess11 0.0041500451

ManufacturingProcess12 0.0001293519

ManufacturingProcess13 0.0072739916

ManufacturingProcess14 0.0047148999

ManufacturingProcess15 0.0043143962

ManufacturingProcess16 0.0001571240

ManufacturingProcess17 0.0081821859

ManufacturingProcess18 0.0016796166

ManufacturingProcess19 0.0013170724

ManufacturingProcess20 0.0018220593

ManufacturingProcess21 0.0029222467

ManufacturingProcess22 0.0028299021

ManufacturingProcess23 0.0046532756

ManufacturingProcess24 0.0064236747

ManufacturingProcess25 0.0045760453

ManufacturingProcess26 0.0047087295

ManufacturingProcess27 0.0034934057

ManufacturingProcess28 0.0029032736

ManufacturingProcess29 0.0092933008

ManufacturingProcess30 0.0052514294

ManufacturingProcess31 0.0029302038

ManufacturingProcess32 0.0095202745

ManufacturingProcess33 0.0094482146

ManufacturingProcess34 0.0026804695

ManufacturingProcess35 0.0046265651

ManufacturingProcess36 0.1180427124

ManufacturingProcess37 0.0107504515

ManufacturingProcess38 0.0040080219

ManufacturingProcess39 0.0088174535

ManufacturingProcess40 0.0038981436

ManufacturingProcess41 0.0039788277

ManufacturingProcess42 0.0079372437

ManufacturingProcess43 0.0044322238

ManufacturingProcess44 0.0065351804

ManufacturingProcess45 0.0140503775

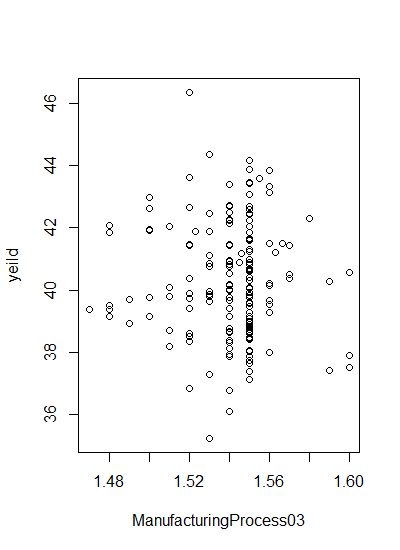
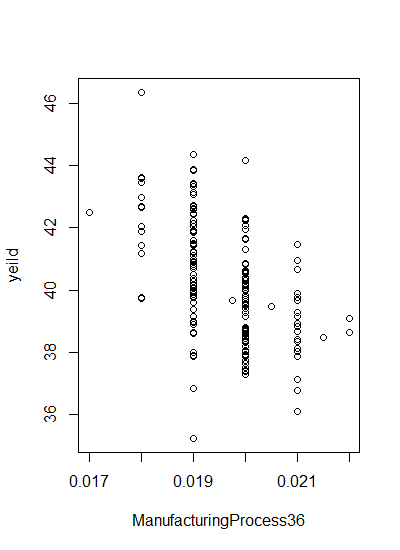
ManufacturingProcess variables dominate the list with higher weighted sums.

f.

dataset <- as.data.frame(dataset)

plot(dataset$ManufacturingProcess36,dataset$y,ylab="yeild",xlab="ManufacturingProcess36")

plot(dataset$ManufacturingProcess03,dataset$y,ylab="yeild",xlab="ManufacturingProcess03")



ManufacturingProcess36 has a high overall weight and when we plot yeild and ManufacturingProcess36 has a high correlation can be seen. But for ManufacturingProcess03 ,

it is low. So information about ManufacturingProcess36 is helpfull in improving the yeild.