> # Read Desharnais77 public dataset from promise uottawa repository

> desharnais <- read.table("DesharnaisPromise.csv",

+ sep = ",", header = TRUE)

> desha <- subset(desharnais,Project!=38)

> # continue to remove 38 , 44, 66, 75 because not enough data in these projects

> desha <- subset(desha,Project!=44)

> desha <- subset(desha,Project!=66)

> desha <- subset(desha,Project!=75)

> ### because ? in some fields under these columns

> # has to specify as numeric

> desha$TeamExp<-as.numeric(desha$TeamExp)

> desha$ManagerExp<-as.numeric(desha$ManagerExp)

> # Keep only Project+Effor + 8 independent var, remove project, yearEnd, PointsNnAdjust

> desha <- desha[,c(1,2,3,5:10,12)]

> ### randomly keep 6 project rows , remaining 71 rows, keep as Training set

> deshaTrain <- subset(desha,Project!=69)

> deshaTrain <- subset(deshaTrain,Project!=72)

> deshaTrain <- subset(deshaTrain,Project!=73)

> deshaTrain <- subset(deshaTrain,Project!=74)

> deshaTrain <- subset(deshaTrain,Project!=70)

> deshaTrain <- subset(deshaTrain,Project!=71)

> ### keep the 6 rows in Test dataset

> deshaTest <- subset(desha,Project>68 & Project <75)

> ### remove first column Project from both Train and Test sets

> deshaTrain <- deshaTrain[,c(2:10)]

> deshaTest <- deshaTest[,c(2:10)]

> deshaTrainY = deshaTrain$Effort

> deshaTestY = deshaTest$Effort

> #remove Effort from X set

> deshaTrainX = deshaTrain[,c(1:3,5:9)]

> deshaTestX = deshaTest[,c(1:3,5:9)]

> solTestY = deshaTestY

> solTrainY = deshaTrainY

> solTrainX = deshaTrainX

> solTestX = deshaTestX

> solTrainXtrans = deshaTrainX

> solTestXtrans = deshaTestX

> #try Kuhn ch6 Linear Regression and cousins

> library(AppliedPredictiveModeling)

>

> library(lattice)

> ### Some initial plots of the data

>

> xyplot(solTrainY ~ solTrainX$Entities, type = c("p", "g"),

+ ylab = "Effort",

+ main = "(a)",

+ xlab = "Entities")

> xlab = "Entities")

Error: unexpected ')' in " xlab = "Entities")"

> xyplot(solTrainY ~ solTrainX$Transactions, type = c("p", "g"),

+ ylab = "Effort",

+ xlab = "Trans")

> library(corrplot)

> corrplot::corrplot(cor(solTrainXtrans),

+ order = "hclust",

+ tl.cex = .8)

> ### Section 6.2 Linear Regression

> set.seed(100)

> indx <- createFolds(solTrainY, returnTrain = TRUE)

> ctrl <- trainControl(method = "cv", index = indx)

> set.seed(100)

> lmTune0 <- train(x = solTrainXtrans, y = solTrainY,

+ method = "lm",

+ trControl = ctrl)

Warning messages:

1: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

2: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

3: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

4: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

5: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

6: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

7: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

8: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

9: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

10: In predict.lm(modelFit, newdata) :

prediction from a rank-deficient fit may be misleading

>

> lmTune0

Linear Regression

71 samples

8 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results:

RMSE Rsquared MAE

2890.465 0.5785532 2196.331

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

> tooHigh <- findCorrelation(cor(solTrainXtrans), .8)

> trainXfiltered <- solTrainXtrans[, -tooHigh]

> testXfiltered <- solTestXtrans[, -tooHigh]

>

> set.seed(100)

> lmTune <- train(x = trainXfiltered, y = solTrainY,

+ method = "lm",

+ trControl = ctrl)

>

> lmTune

Linear Regression

71 samples

7 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results:

RMSE Rsquared MAE

2890.465 0.5785532 2196.331

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

> ### Save the test set results in a data frame

> testResults <- data.frame(obs = solTestY,

+ Linear\_Regression = predict(lmTune, testXfiltered))

>

> ### Section 6.4 Penalized Models

> ## There is now a simple ridge regression method.

> ridgeGrid <- expand.grid(lambda = seq(0, .1, length = 15))

>

> set.seed(100)

> ridgeTune <- train(x = solTrainXtrans, y = solTrainY,

+ method = "ridge",

+ tuneGrid = ridgeGrid,

+ trControl = ctrl,

+ preProc = c("center", "scale"))

> ridgeTune

Ridge Regression

71 samples

8 predictor

Pre-processing: centered (8), scaled (8)

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results across tuning parameters:

lambda RMSE Rsquared MAE

0.000000000 2890.465 0.5785532 2196.331

0.007142857 2888.209 0.5785053 2194.353

0.014285714 2886.145 0.5784456 2192.489

0.021428571 2884.262 0.5783747 2190.730

0.028571429 2882.551 0.5782936 2189.071

0.035714286 2881.001 0.5782030 2187.507

0.042857143 2879.605 0.5781035 2186.031

0.050000000 2878.356 0.5779959 2184.928

0.057142857 2877.246 0.5778807 2183.979

0.064285714 2876.269 0.5777586 2183.100

0.071428571 2875.418 0.5776301 2182.286

0.078571429 2874.688 0.5774957 2181.535

0.085714286 2874.075 0.5773560 2180.843

0.092857143 2873.572 0.5772113 2180.207

0.100000000 2873.176 0.5770621 2180.015

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was lambda = 0.1.

>

> print(update(plot(ridgeTune), xlab = "Penalty"))

>

> enetGrid <- expand.grid(lambda = c(0, 0.01, .1),

+ fraction = seq(.05, 1, length = 20))

> set.seed(100)

> enetTune <- train(x = solTrainXtrans, y = solTrainY,

+ method = "enet",

+ tuneGrid = enetGrid,

+ trControl = ctrl,

+ preProc = c("center", "scale"))

> enetTune

Elasticnet

71 samples

8 predictor

Pre-processing: centered (8), scaled (8)

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results across tuning parameters:

lambda fraction RMSE Rsquared MAE

0.00 0.05 3781.445 0.4774543 2897.896

0.00 0.10 3612.075 0.4744326 2775.538

0.00 0.15 3468.811 0.4715431 2666.927

0.00 0.20 3347.110 0.4861316 2561.272

0.00 0.25 3244.115 0.4987823 2454.566

0.00 0.30 3157.932 0.5095587 2351.560

0.00 0.35 3109.099 0.5112419 2286.918

0.00 0.40 3063.767 0.5194054 2223.961

0.00 0.45 3002.281 0.5258196 2149.721

0.00 0.50 2952.815 0.5312533 2105.361

0.00 0.55 2912.710 0.5345911 2071.218

0.00 0.60 2887.708 0.5377282 2053.587

0.00 0.65 2877.876 0.5414518 2056.394

0.00 0.70 2872.878 0.5450629 2070.708

0.00 0.75 2880.365 0.5491048 2090.107

0.00 0.80 2880.525 0.5561382 2109.901

0.00 0.85 2877.498 0.5630281 2132.197

0.00 0.90 2877.763 0.5692102 2151.567

0.00 0.95 2882.121 0.5743494 2171.476

0.00 1.00 2890.465 0.5785532 2196.331

0.01 0.05 3775.924 0.4771276 2893.166

0.01 0.10 3602.607 0.4742934 2767.855

0.01 0.15 3457.072 0.4740470 2656.453

0.01 0.20 3331.914 0.4893973 2546.622

0.01 0.25 3224.900 0.5033552 2434.731

0.01 0.30 3143.893 0.5121229 2334.687

0.01 0.35 3096.806 0.5130265 2272.420

0.01 0.40 3047.235 0.5222897 2203.988

0.01 0.45 2987.568 0.5282235 2136.716

0.01 0.50 2941.262 0.5335214 2097.890

0.01 0.55 2902.600 0.5370697 2064.325

0.01 0.60 2882.463 0.5412163 2054.508

0.01 0.65 2872.640 0.5453195 2061.153

0.01 0.70 2873.603 0.5488218 2078.843

0.01 0.75 2877.503 0.5546609 2097.882

0.01 0.80 2877.027 0.5614698 2123.640

0.01 0.85 2875.636 0.5683257 2143.777

0.01 0.90 2879.171 0.5726055 2161.278

0.01 0.95 2882.686 0.5757769 2175.688

0.01 1.00 2887.361 0.5784828 2193.594

0.10 0.05 3776.662 0.4761908 2894.425

0.10 0.10 3606.099 0.4728194 2770.888

0.10 0.15 3460.673 0.4858844 2659.634

0.10 0.20 3329.462 0.5023268 2547.790

0.10 0.25 3215.785 0.5144576 2434.547

0.10 0.30 3128.804 0.5192791 2329.358

0.10 0.35 3086.498 0.5132872 2270.611

0.10 0.40 3045.858 0.5204056 2206.946

0.10 0.45 2992.566 0.5277486 2141.742

0.10 0.50 2942.843 0.5348631 2098.919

0.10 0.55 2897.722 0.5428827 2063.062

0.10 0.60 2870.208 0.5473699 2046.631

0.10 0.65 2860.246 0.5495669 2043.683

0.10 0.70 2855.142 0.5512024 2053.279

0.10 0.75 2859.843 0.5529347 2069.603

0.10 0.80 2868.114 0.5565056 2089.174

0.10 0.85 2872.394 0.5611505 2119.123

0.10 0.90 2869.849 0.5669576 2139.941

0.10 0.95 2869.656 0.5724440 2159.434

0.10 1.00 2873.176 0.5770621 2180.015

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were fraction = 0.7 and lambda

= 0.1.

> plot(enetTune)

> testResults$Enet <- predict(enetTune, solTestXtrans)

> ### Section 6.3 Partial Least Squares

>

> ## Run PLS and PCR on solubility data and compare results

> # removed tune grid ncomp =... to use default , max number of components

> set.seed(100)

> plsTune <- train(x = solTrainXtrans, y = solTrainY,

+ method = "pls",

+ trControl = ctrl)

> plsTune

Partial Least Squares

71 samples

8 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results across tuning parameters:

ncomp RMSE Rsquared MAE

1 3089.316 0.4631649 2191.559

2 3139.772 0.4676062 2233.136

3 3140.972 0.4690654 2305.236

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was ncomp = 1.

> testResults$PLS <- predict(plsTune, solTestXtrans)

>

> set.seed(100)

> pcrTune <- train(x = solTrainXtrans, y = solTrainY,

+ method = "pcr",

+ trControl = ctrl)

> pcrTune

Principal Component Analysis

71 samples

8 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 64, 64, 64, 64, 64, 64, ...

Resampling results across tuning parameters:

ncomp RMSE Rsquared MAE

1 3106.318 0.4623076 2201.208

2 3141.561 0.4672719 2233.862

3 3185.003 0.4387136 2286.805

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was ncomp = 1.

> plsResamples <- plsTune$results

> plsResamples$Model <- "PLS"

> pcrResamples <- pcrTune$results

> pcrResamples$Model <- "PCR"

> plsPlotData <- rbind(plsResamples, pcrResamples)

> xyplot(RMSE ~ ncomp,

+ data = plsPlotData,

+ #aspect = 1,

+ xlab = "# Components",

+ ylab = "RMSE (Cross-Validation)",

+ auto.key = list(columns = 2),

+ groups = Model,

+ type = c("o", "g"))

> plsImp <- varImp(plsTune, scale = FALSE)

> plot(plsImp, top = 25, scales = list(y = list(cex = .95)))