QSARdata Analysis

Jeremy Ash

June 20, 2017

Recreating Krstajic et al. 2014

I am attempting to recreate the analysis in "Cross-validation pitfalls when selecting and assessing regression and classification models" (Krstajic et al. 2014). THis paper has 70 citations and introduces methods for repeated cross validation (e.g., nested CV where another CV loop is used to perform model tunning within each outer CV iteration). The QSARdata package is used to perform the analysis, and the methods used for filtering descriptors are described well enough that I can recreate them.

Analysis of the AquaticTox data set

"AquaticTox contains negative log of toxic activity for 322 compounds. It was described and compiled by He and Jurs ... There are 220 MOE 2D descriptors for each compound. However, during pre-processing we removed 30 descriptors with near zero variation and 6 descriptors that were linear combinations of others, leaving 184 descriptors for model building."

```
dim(AquaticTox_moe2D)
```

```
## [1] 322 221
```

```
rownames(AquaticTox_moe2D) <- make.unique(as.character(AquaticTox_moe2D[, 1]))
AquaticTox_moe2D <- AquaticTox_moe2D[, -1]
length(nearZeroVar(AquaticTox_moe2D))
```

```
## [1] 30
```

```
AquaticTox_moe2D <- AquaticTox_moe2D[-nearZeroVar(AquaticTox_moe2D)]
length(findLinearCombos(AquaticTox_moe2D)$remove)
```

```
## [1] 6
```

```
AquaticTox_moe2D <- AquaticTox_moe2D[-findLinearCombos(AquaticTox_moe2D)$remove]
dim(AquaticTox_moe2D)
```

```
## [1] 322 184
```

```
AquaticTox_moe2D <- cbind(AquaticTox_Outcome$Activity, AquaticTox_moe2D)
head(AquaticTox_moe2D[, 1:3])
```

```
##                                                    AquaticTox_Outcome$Activity
## (d)-limonene                                                              5.29
## 111-trichloro-2-methyl-2-propanolol(chlorobytanol)                        3.12
## 111-trichloroethane                                                       3.40
## 1122-tetrachloroethane                                                    3.92
## 112-trichloroethane                                                       3.21
## 11-dichloroethylene(vinylidene                                            2.84
##                                                    moeGao_Abra_L
## (d)-limonene                                               4.729
## 111-trichloro-2-methyl-2-propanolol(chlorobytanol)         4.226
## 111-trichloroethane                                        2.790
## 1122-tetrachloroethane                                 3.636
## 112-trichloroethane                                    3.078
## 11-dichloroethylene(vinylidene                         2.225
##                                                  moeGao_Abra_R
## (d)-limonene                                           0.512
## 111-trichloro-2-methyl-2-propanolol(chlorobytanol)     0.527
## 111-trichloroethane                                    0.383
## 1122-tetrachloroethane                                 0.496
## 112-trichloroethane                                    0.401
## 11-dichloroethylene(vinylidene                         0.412
```

```
aq.process <- AquaticTox_moe2D
```

I set the ridge regression lambda value and ncomp to the optimal values found by a cross validated grid search in the paper. I also used 10 fold cross validation as they did, and 5 repeats (they used 50). Folds were also assigned at random for one of their methods. Since their CV protocol matches ours closely, their results should match ours. However, neither ridge regression nor PLS have performance that is as good as what was found by the paper. We have effectively the same performance with Lasso.

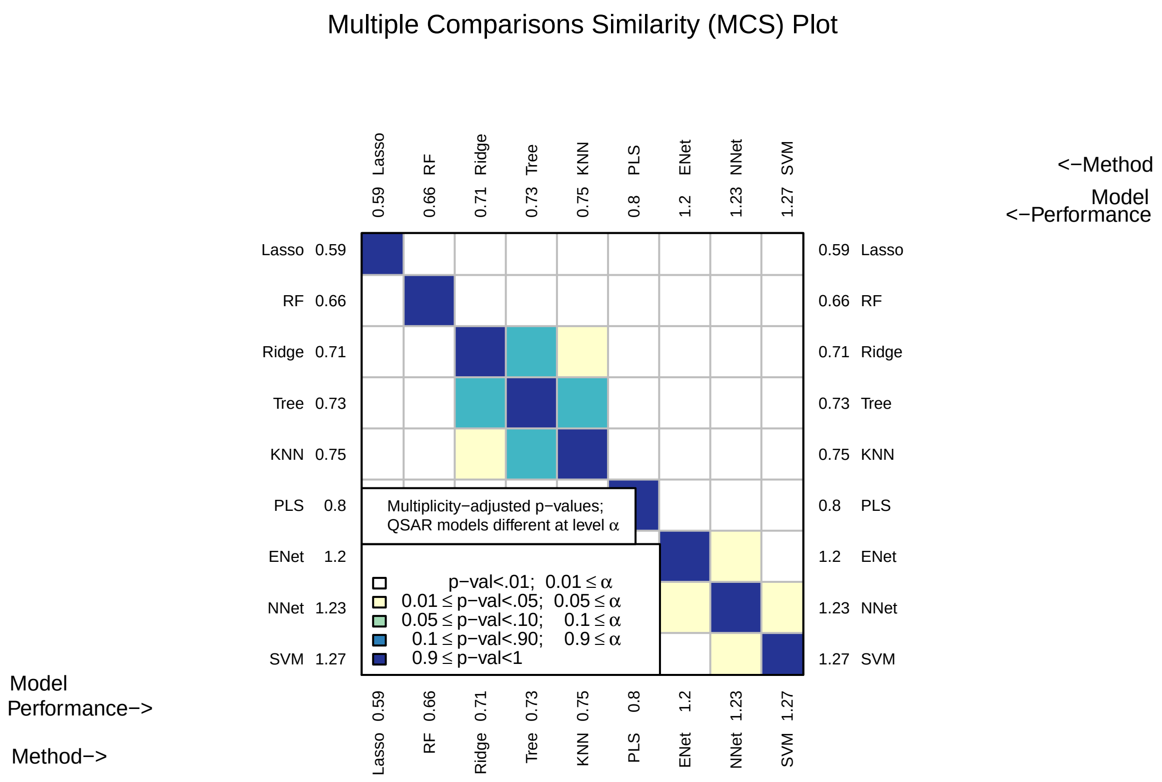
They said that they used the sum of squared residuals for their Cross Validation loss function. I guess this means CV SSE, I used RMSE and converted to SSE for comparison.

```
user.params <- MakeModelDefaults(nrow(aq.process), ncol(aq.process) - 1, F, 10)
user.params$Ridge$lambda <- .05325
user.params$PLS$ncomp <- 13

cml.tune <- ModelTrain(aq.process, ids = F, user.params = user.params,
                       models = c("Ridge", "Lasso", "RF",
                                  "Tree", "KNN", "NNet",
                                  "SVM", "PLS", "ENet"),
                       nsplits = 5, seed.in = 1:5)

cml.tune$params$Ridge
cml.tune$params$PLS

CombineSplits(cml.tune, metric = "RMSE")
```



```r
# pretty sure that the performance measure they report in table 3 is RMSE
(.59^2) * 322
```

```
## [1] 112.0882
```

The PLS model fit with caret instead

```r
colnames(aq.process)[1] <- "activity"
plsGrid <-  expand.grid(ncomp = c(1, 3, 13))

ctrl <- trainControl(method = "cv", number = 10, repeats = 3)

usingMC <-  train(activity ~ .,
                  data = aq.process,
                  method = "pls",
                  trControl = ctrl,
                  tuneGrid = plsGrid)
```

```
## Loading required package: pls
```

```
## Warning: package 'pls' was built under R version 3.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
```

```
##
##     loadings
```

```
usingMC
```

```
## Partial Least Squares
##
## 322 samples
## 184 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 290, 290, 289, 290, 290, 290, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared
##    1     0.9320087  0.5130940
##    3     0.7408084  0.6863987
##   13     0.6214240  0.7757041
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was ncomp = 13.
```

Analysis of the AquaticTox data set

```
head(MP_Descriptors[1:6])
```

```
##   diameter petitjean petitjeanSC radius  VDistEq  VDistMa
## 1        5 0.4000000   0.6666667      3 2.037476 6.011166
## 2        9 0.4444444   0.8000000      5 2.954872 8.805204
## 3       10 0.5000000   1.0000000      5 3.083532 8.211762
## 4        7 0.4285714   0.7500000      4 2.616827 7.313269
## 5        7 0.4285714   0.7500000      4 2.609669 6.833154
## 6        7 0.4285714   0.7500000      4 2.575820 7.324709
```

```
length(nearZeroVar(MP_Descriptors))
```

```
## [1] 11
```

```
MP_Descriptors <- MP_Descriptors[-nearZeroVar(MP_Descriptors)]
dim(MP_Descriptors)
```

```
## [1] 4401  191
```

```
length(findLinearCombos(MP_Descriptors)$remove)
```

```
## [1] 22
```

```
MP_Descriptors <- MP_Descriptors[-findLinearCombos(MP_Descriptors)$remove]
dim(MP_Descriptors)
```

```
## [1] 4401  169
```

```
MP_Descriptors <- cbind(MP_Outcome, MP_Descriptors)
MP.descriptors <- MP_Descriptors

user.params <- MakeModelDefaults(nrow(MP.descriptors), ncol(MP.descriptors) - 1, F, 10)
```

```r
user.params$Ridge$lambda <- .0549
user.params$PLS$ncomp <- 47

cml.tune$params$Ridge
```

```
##    lambda
## 1 0.05325
```

```r
cml.tune$params$PLS
```

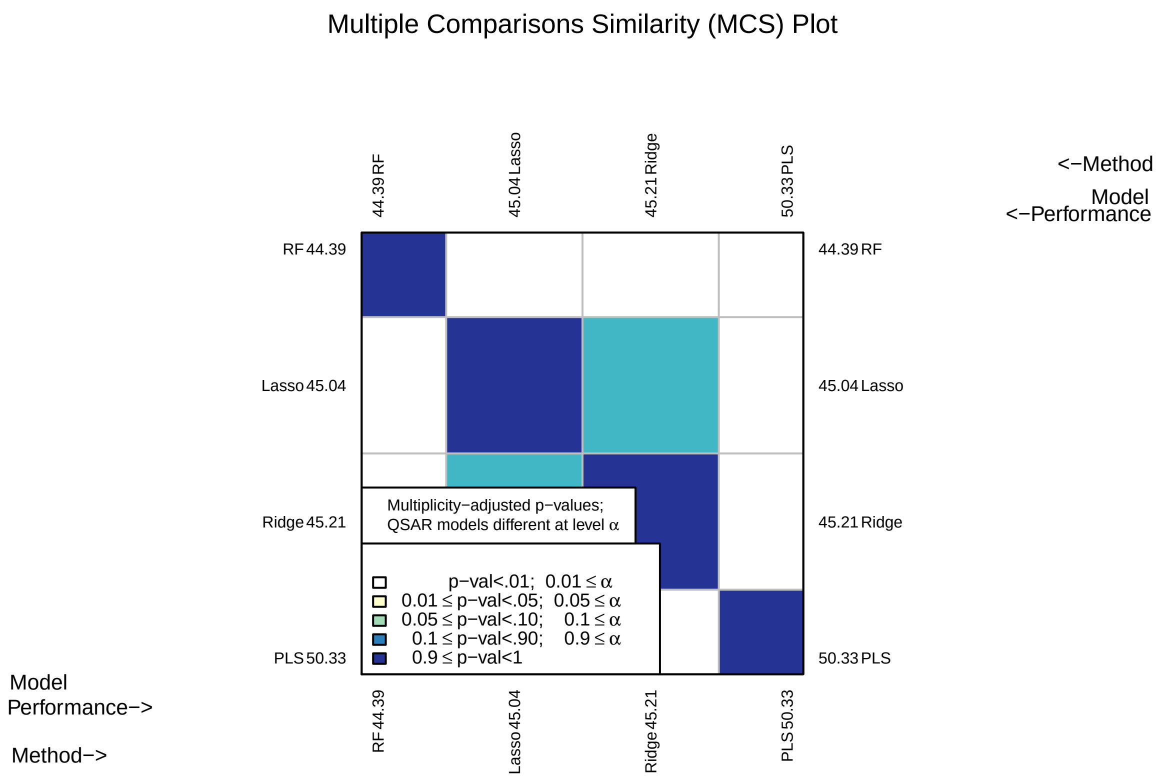
```
##   ncomp
## 1    13
```

The ridge performance is comparable, PLS is still a bit worse.

```r
cml.tune <- ModelTrain(MP_Descriptors, ids = F, user.params = user.params,
                      models = c("Ridge", "Lasso", "RF",
                                 "PLS"),
                      nsplits = 3, seed.in = 1:3)

cml.tune$params$Ridge
cml.tune$params$PLS

CombineSplits(cml.tune, metric = "RMSE")
```



Future Directions

Perhaps we should generate boxplots for the sum of squared residuals for each CV split the way they did. I could modify the performance function in Chemmodlab so that the data necessary is output. I have been wanting to compute a SE for the performance measures anyways.