Retrieved from the ‘AppliedPredictiveModeling’ package

https://cran.r-project.org/web/packages/AppliedPredictiveModeling/

`solubility` *Solubility Data*

Description

Tetko et al. (2001) and Huuskonen (2000) investigated a set of compounds with corresponding experimental solubility values using complex sets of descriptors. They used linear regression and neural network models to estimate the relationship between chemical structure and solubility. For our analyses, we will use 1267 compounds and a set of more understandable descriptors that fall into one of three groups: 208 binary "fingerprints" that indicate the presence or absence of a particular chemical sub-structure, 16 count descriptors (such as the number of bonds or the number of Bromine atoms) and 4 continuous descriptors (such as molecular weight or surface area).

Usage

```
data(solubility)
```

Value

`solTrainX`	training set predictors in their natural units.
`solTrainXtrans`	training set predictors after transformations for skewness and centering/scaling.
`solTrainY`	a vector of log10 solubility values for the training set.
`solTestX`	test set predictors in their natural units.
`solTestXtrans`	test set predictors after the same transformations used on the training set are applied.
`solTestY`	a vector of log10 solubility values for the training set.

Source

Tetko, I., Tanchuk, V., Kasheva, T., and Villa, A. (2001). Estimation of aqueous solubility of chemical compounds using E-state indices. *Journal of Chemical Information and Computer Sciences*, 41(6), 1488-1493.

Huuskonen, J. (2000). Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *Journal of Chemical Information and Computer Sciences*, 40(3), 773-777.

Examples

```
data(solubility)

library(caret)

### Cross-validation splits used in the book:
set.seed(100)
indx <- createFolds(solTrainY, returnTrain = TRUE)
```

```
### To re-create the transformed version of the data:
## Not run:
## Find the predictors that are not fingerprints
contVars <- names(solTrainX)[!grepl(”FP”, names(solTrainX))]
## Some have zero values, so we need to add one to them so that
## we can use the Box-Cox transformation. Alternatively, we could
## use the Yeo-Johnson transformation without altering the data.
contPredTrain <- solTrainX[,contVars] + 1
contPredTest  <-  solTestX[,contVars] + 1

pp <- preProcess(contPredTrain, method = ”BoxCox”)
contPredTrain <- predict(pp, contPredTrain)
contPredTest  <- predict(pp, contPredTest)

## Reassemble the fingerprint data with the transformed values.
trainXtrans <- cbind(solTrainX[,grep(”FP”, names(solTrainX))], contPredTrain)
testXtrans  <- cbind( solTestX[,grep(”FP”, names(solTestX))],  contPredTest)

all.equal(trainXtrans, solTrainXtrans)
all.equal(testXtrans, solTestXtrans)

## End(Not run)
```