Package 'RobustPrediction'

December 16, 2024

Type Package

Title Robust Tuning and Training for Cross-Source Prediction

Version 0.1.7

Maintainer Yuting He <yutingh19@gmail.com>

Description Provides robust parameter tuning and model training for predictive models applied across data sources where the data distribution varies slightly from source to source. This package implements three primary tuning methods: cross-validation-based internal tuning, external tuning, and the 'RobustTuneC' method. External tuning includes a conservative option where parameters are tuned internally on the training data and validating on an external dataset, providing a slightly pessimistic estimate. It supports Lasso, Ridge, Random Forest, Boosting, and Support Vector Machine classifiers. Currently, only binary classification is supported. The response variable must be the first column of the dataset and a factor with exactly two levels. The tuning methods are based on the paper by Nicole Ellenbach, Anne-Laure Boulesteix, Bernd Bischl, Kristian Unger, and Roman Hornung (2021) ``Improved Outcome Prediction Across Data Sources Through Robust Parameter Tuning" <doi:10.1007/s00357-020-09368-z>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

LazyData true

Depends R (>= 3.5.0)

Imports glmnet, mboost, mlr, ranger, e1071, pROC

URL https://github.com/Yuting-He/RobustPrediction

Author Yuting He [aut, cre],
Nicole Ellenbach [ctb],
Roman Hornung [ctb]

Repository CRAN

Date/Publication 2024-12-16 16:50:11 UTC

Contents

`RobustPrediction` *Package Title: Robust Tuning and Training for Cross-Source Prediction*

Description

This package provides robust parameter tuning and predictive modeling techniques, useful for situations where prediction across different data sources is important and the data distribution varies slightly from source to source.

Details

The 'RobustPrediction' package helps users build and tune classifiers using the methods 'RobustTuneC' method, internal, or external tuning method. The package supports the following classifiers: boosting, lasso, ridge, random forest, and support vector machine(SVM). It is intended for scenarios where parameter tuning across data sources is important.

The 'RobustPrediction' package provides comprehensive tools for robust parameter tuning and predictive modeling, particularly for cross-source prediction tasks.

The package includes functions for tuning model parameters using three methods: - **Internal tuning**: Standard cross-validation on the training data to select the best parameters. - **External

tuning**: Parameter tuning based on an external dataset that is independent of the training data. This method has two variants controlled by the `estperf` argument: - **Standard external tuning (`estperf = FALSE`)**: Parameters are tuned directly using the external dataset. This is the default approach and provides a straightforward method for selecting optimal parameters based on external data. - **Conservative external tuning (`estperf = TRUE`)**: Internal tuning is first performed on the training data, and then the model is evaluated on the external dataset. This approach provides a more conservative (slightly pessimistic) AUC estimate, as described by Ellenbach et al. (2021). For the most accurate performance evaluation, it is recommended to use a second external dataset. - **RobustTuneC**: A method designed to combine internal and external tuning for better performance in cross-source scenarios.

The package supports Lasso, Ridge, Random Forest, Boosting, and SVM classifiers. These models can be trained and tuned using the provided methods, and the package includes the model's AUC (Area Under the Curve) value to help users evaluate prediction performance.

It is particularly useful when the data to be predicted comes from a different source than the training data, where variability between datasets may require more robust parameter tuning techniques. The methods provided in this package may help reduce overfitting the training data distribution and improve model generalization across different data sources.

Dependencies

This package requires the following packages: `glmnet`, `mboost`, `mlr`, `pROC`, `ranger`.

Author(s)

Maintainer: Yuting He <yutingh19@gmail.com>

Other contributors:

- Nicole Ellenbach [contributor]
- Roman Hornung [contributor]

References

Ellenbach, N., Boulesteix, A.-L., Bischl, B., Unger, K., & Hornung, R. (2021). Improved outcome prediction across data sources through robust parameter tuning. *Journal of Classification*, *38*, 212-231. <doi:10.1007/s00357-020-09368-z>.

See Also

Useful links:

- https://github.com/Yuting-He/RobustPrediction

Examples

```
# Example usage:
data(sample_data_train)
data(sample_data_extern)
res <- tuneandtrain(sample_data_train, sample_data_extern, tuningmethod = "robusttunec",
  classifier = "lasso")
```

`sample_data_extern` *Sample External Validation Data Subset*

Description

This dataset, named 'sample_data_extern', is a subset of publicly available microarray data from the HG-U133PLUS2 chip. It contains expression levels of 200 genes across 50 samples, used primarily as an external validation set in robust feature selection studies. The data has been sourced from the ArrayExpress repository and has been referenced in several research articles.

Usage

```
sample_data_extern
```

Format

A data frame with 50 observations and 201 variables, including:

y Factor. The response variable.

236694_at Numeric. Expression level of gene 236694_at.

222356_at Numeric. Expression level of gene 222356_at.

1554125_a_at Numeric. Expression level of gene 1554125_a_at.

232823_at Numeric. Expression level of gene 232823_at.

205766_at Numeric. Expression level of gene 205766_at.

1560446_at Numeric. Expression level of gene 1560446_at.

202565_s_at Numeric. Expression level of gene 202565_s_at.

234887_at Numeric. Expression level of gene 234887_at.

209687_at Numeric. Expression level of gene 209687_at.

221592_at Numeric. Expression level of gene 221592_at.

1570123_at Numeric. Expression level of gene 1570123_at.

241368_at Numeric. Expression level of gene 241368_at.

243324_x_at Numeric. Expression level of gene 243324_x_at.

224046_s_at Numeric. Expression level of gene 224046_s_at.

202775_s_at Numeric. Expression level of gene 202775_s_at.

216332_at Numeric. Expression level of gene 216332_at.

1569545_at Numeric. Expression level of gene 1569545_at.

205946_at Numeric. Expression level of gene 205946_at.

203547_at Numeric. Expression level of gene 203547_at.

243239_at Numeric. Expression level of gene 243239_at.

234245_at Numeric. Expression level of gene 234245_at.

210832_x_at Numeric. Expression level of gene 210832_x_at.

224549_x_at Numeric. Expression level of gene 224549_x_at.

236628_at Numeric. Expression level of gene 236628_at.

214848_at Numeric. Expression level of gene 214848_at.

1553015_a_at Numeric. Expression level of gene 1553015_a_at.

1554199_at Numeric. Expression level of gene 1554199_at.

1557636_a_at Numeric. Expression level of gene 1557636_a_at.

1558511_s_at Numeric. Expression level of gene 1558511_s_at.

1561713_at Numeric. Expression level of gene 1561713_at.

1561883_at Numeric. Expression level of gene 1561883_at.

1568720_at Numeric. Expression level of gene 1568720_at.

1569168_at Numeric. Expression level of gene 1569168_at.

1569443_s_at Numeric. Expression level of gene 1569443_s_at.

1570103_at Numeric. Expression level of gene 1570103_at.

200916_at Numeric. Expression level of gene 200916_at.

201554_x_at Numeric. Expression level of gene 201554_x_at.

202371_at Numeric. Expression level of gene 202371_at.

204481_at Numeric. Expression level of gene 204481_at.

205831_at Numeric. Expression level of gene 205831_at.

207061_at Numeric. Expression level of gene 207061_at.

207423_s_at Numeric. Expression level of gene 207423_s_at.

209896_s_at Numeric. Expression level of gene 209896_s_at.

212646_at Numeric. Expression level of gene 212646_at.

214068_at Numeric. Expression level of gene 214068_at.

217727_x_at Numeric. Expression level of gene 217727_x_at.

221103_s_at Numeric. Expression level of gene 221103_s_at.

221785_at Numeric. Expression level of gene 221785_at.

224207_x_at Numeric. Expression level of gene 224207_x_at.

228257_at Numeric. Expression level of gene 228257_at.

228877_at Numeric. Expression level of gene 228877_at.

231173_at Numeric. Expression level of gene 231173_at.

231328_s_at Numeric. Expression level of gene 231328_s_at.

231639_at Numeric. Expression level of gene 231639_at.

232221_x_at Numeric. Expression level of gene 232221_x_at.

232349_x_at Numeric. Expression level of gene 232349_x_at.

232849_at Numeric. Expression level of gene 232849_at.

233601_at Numeric. Expression level of gene 233601_at.

234403_at Numeric. Expression level of gene 234403_at.

234585_at Numeric. Expression level of gene 234585_at.

234650_at Numeric. Expression level of gene 234650_at.

234897_s_at Numeric. Expression level of gene 234897_s_at.

236071_at Numeric. Expression level of gene 236071_at.

236689_at Numeric. Expression level of gene 236689_at.

238551_at Numeric. Expression level of gene 238551_at.

239414_at Numeric. Expression level of gene 239414_at.

241034_at Numeric. Expression level of gene 241034_at.

241131_at Numeric. Expression level of gene 241131_at.

241897_at Numeric. Expression level of gene 241897_at.

242611_at Numeric. Expression level of gene 242611_at.

244805_at Numeric. Expression level of gene 244805_at.

244866_at Numeric. Expression level of gene 244866_at.

32259_at Numeric. Expression level of gene 32259_at.

1552264_a_at Numeric. Expression level of gene 1552264_a_at.

1552880_at Numeric. Expression level of gene 1552880_at.

1553186_x_at Numeric. Expression level of gene 1553186_x_at.

1553372_at Numeric. Expression level of gene 1553372_at.

1553438_at Numeric. Expression level of gene 1553438_at.

1554299_at Numeric. Expression level of gene 1554299_at.

1554362_at Numeric. Expression level of gene 1554362_at.

1554491_a_at Numeric. Expression level of gene 1554491_a_at.

1555098_a_at Numeric. Expression level of gene 1555098_a_at.

1555990_at Numeric. Expression level of gene 1555990_at.

1556034_s_at Numeric. Expression level of gene 1556034_s_at.

1556822_s_at Numeric. Expression level of gene 1556822_s_at.

1556824_at Numeric. Expression level of gene 1556824_at.

1557278_s_at Numeric. Expression level of gene 1557278_s_at.

1558603_at Numeric. Expression level of gene 1558603_at.

1558890_at Numeric. Expression level of gene 1558890_at.

1560791_at Numeric. Expression level of gene 1560791_at.

1561083_at Numeric. Expression level of gene 1561083_at.

1561364_at Numeric. Expression level of gene 1561364_at.

1561553_at Numeric. Expression level of gene 1561553_at.

1562523_at Numeric. Expression level of gene 1562523_at.

1562613_at Numeric. Expression level of gene 1562613_at.

1563351_at Numeric. Expression level of gene 1563351_at.

1563473_at Numeric. Expression level of gene 1563473_at.

1566780_at Numeric. Expression level of gene 1566780_at.

1567257_at Numeric. Expression level of gene 1567257_at.

1569664_at Numeric. Expression level of gene 1569664_at.

1569882_at Numeric. Expression level of gene 1569882_at.

1570252_at Numeric. Expression level of gene 1570252_at.

201089_at Numeric. Expression level of gene 201089_at.

201261_x_at Numeric. Expression level of gene 201261_x_at.

202052_s_at Numeric. Expression level of gene 202052_s_at.

202236_s_at Numeric. Expression level of gene 202236_s_at.

202948_at Numeric. Expression level of gene 202948_at.

203080_s_at Numeric. Expression level of gene 203080_s_at.

203211_s_at Numeric. Expression level of gene 203211_s_at.

203218_at Numeric. Expression level of gene 203218_at.

203236_s_at Numeric. Expression level of gene 203236_s_at.

203347_s_at Numeric. Expression level of gene 203347_s_at.

203960_s_at Numeric. Expression level of gene 203960_s_at.

204609_at Numeric. Expression level of gene 204609_at.

204806_x_at Numeric. Expression level of gene 204806_x_at.

204949_at Numeric. Expression level of gene 204949_at.

204979_s_at Numeric. Expression level of gene 204979_s_at.

205823_at Numeric. Expression level of gene 205823_at.

205902_at Numeric. Expression level of gene 205902_at.

205967_at Numeric. Expression level of gene 205967_at.

206186_at Numeric. Expression level of gene 206186_at.

207151_at Numeric. Expression level of gene 207151_at.

207379_at Numeric. Expression level of gene 207379_at.

207440_at Numeric. Expression level of gene 207440_at.

207883_s_at Numeric. Expression level of gene 207883_s_at.

208277_at Numeric. Expression level of gene 208277_at.

208280_at Numeric. Expression level of gene 208280_at.

209224_s_at Numeric. Expression level of gene 209224_s_at.

209561_at Numeric. Expression level of gene 209561_at.

209630_s_at Numeric. Expression level of gene 209630_s_at.

210118_s_at Numeric. Expression level of gene 210118_s_at.

210342_s_at Numeric. Expression level of gene 210342_s_at.

211566_x_at Numeric. Expression level of gene 211566_x_at.

211756_at Numeric. Expression level of gene 211756_at.

212170_at Numeric. Expression level of gene 212170_at.

212494_at Numeric. Expression level of gene 212494_at.

213118_at Numeric. Expression level of gene 213118_at.

214475_x_at Numeric. Expression level of gene 214475_x_at.

214834_at Numeric. Expression level of gene 214834_at.

215718_s_at Numeric. Expression level of gene 215718_s_at.

216283_s_at Numeric. Expression level of gene 216283_s_at.

217206_at Numeric. Expression level of gene 217206_at.

217557_s_at Numeric. Expression level of gene 217557_s_at.

217577_at Numeric. Expression level of gene 217577_at.

218152_at Numeric. Expression level of gene 218152_at.

218252_at Numeric. Expression level of gene 218252_at.

219714_s_at Numeric. Expression level of gene 219714_s_at.

220506_at Numeric. Expression level of gene 220506_at.

220889_s_at Numeric. Expression level of gene 220889_s_at.

221204_s_at Numeric. Expression level of gene 221204_s_at.

221795_at Numeric. Expression level of gene 221795_at.

222048_at Numeric. Expression level of gene 222048_at.

223142_s_at Numeric. Expression level of gene 223142_s_at.

223439_at Numeric. Expression level of gene 223439_at.

223673_at Numeric. Expression level of gene 223673_at.

224363_at Numeric. Expression level of gene 224363_at.

224512_s_at Numeric. Expression level of gene 224512_s_at.

224690_at Numeric. Expression level of gene 224690_at.

224936_at Numeric. Expression level of gene 224936_at.

225334_at Numeric. Expression level of gene 225334_at.

225713_at Numeric. Expression level of gene 225713_at.

225839_at Numeric. Expression level of gene 225839_at.

226041_at Numeric. Expression level of gene 226041_at.

226093_at Numeric. Expression level of gene 226093_at.

226543_at Numeric. Expression level of gene 226543_at.

227695_at Numeric. Expression level of gene 227695_at.

228295_at Numeric. Expression level of gene 228295_at.

228548_at Numeric. Expression level of gene 228548_at.

229234_at Numeric. Expression level of gene 229234_at.

229658_at Numeric. Expression level of gene 229658_at.

229725_at Numeric. Expression level of gene 229725_at.

230252_at Numeric. Expression level of gene 230252_at.

230471_at Numeric. Expression level of gene 230471_at.

231149_s_at Numeric. Expression level of gene 231149_s_at.

231556_at Numeric. Expression level of gene 231556_at.

231754_at Numeric. Expression level of gene 231754_at.

232011_s_at Numeric. Expression level of gene 232011_s_at.

233030_at Numeric. Expression level of gene 233030_at.

234161_at Numeric. Expression level of gene 234161_at.

235050_at Numeric. Expression level of gene 235050_at.

235094_at Numeric. Expression level of gene 235094_at.

235278_at Numeric. Expression level of gene 235278_at.

235671_at Numeric. Expression level of gene 235671_at.

235952_at Numeric. Expression level of gene 235952_at.

236158_at Numeric. Expression level of gene 236158_at.

236181_at Numeric. Expression level of gene 236181_at.

237055_at Numeric. Expression level of gene 237055_at.

237768_x_at Numeric. Expression level of gene 237768_x_at.

238897_at Numeric. Expression level of gene 238897_at.

239160_at Numeric. Expression level of gene 239160_at.

239998_at Numeric. Expression level of gene 239998_at.

240254_at Numeric. Expression level of gene 240254_at.

240612_at Numeric. Expression level of gene 240612_at.

240692_at Numeric. Expression level of gene 240692_at.

240822_at Numeric. Expression level of gene 240822_at.

240842_at Numeric. Expression level of gene 240842_at.

241331_at Numeric. Expression level of gene 241331_at.

241598_at Numeric. Expression level of gene 241598_at.

241927_x_at Numeric. Expression level of gene 241927_x_at.

242405_at Numeric. Expression level of gene 242405_at.

Details

This dataset was extracted from a larger dataset available on ArrayExpress and is used as an external validation set for feature selection tasks and other machine learning applications in bioinformatics.

Source

The original dataset can be found on ArrayExpress: https://www.ebi.ac.uk/arrayexpress

References

Ellenbach, N., Boulesteix, A.L., Bischl, B., et al. (2021). Improved Outcome Prediction Across Data Sources Through Robust Parameter Tuning. *Journal of Classification*, 38, 212–231. doi:10.1007/s00357020093682.

Hornung, R., Causeur, D., Bernau, C., Boulesteix, A.L. (2017). Improving cross-study prediction through addon batch effect adjustment or addon normalization. *Bioinformatics*, 33(3), 397–404. doi:10.1093/bioinformatics/btw650.

Examples

```
# Load the dataset
data(sample_data_extern)

# View the first few rows of the dataset
head(sample_data_extern)

# Summary of the dataset
summary(sample_data_extern)
```

`sample_data_train` *Sample Training Data Subset*

Description

This dataset, named 'sample_data_train', is a subset of publicly available microarray data from the HG-U133PLUS2 chip. It contains expression levels of 200 genes across 50 samples, used primarily as a training set in robust feature selection studies. The data has been sourced from the ArrayExpress repository and has been referenced in several research articles.

Usage

```
sample_data_train
```

Format

A data frame with 50 observations and 201 variables, including:

y Factor. The response variable.

236694_at Numeric. Expression level of gene 236694_at.

222356_at Numeric. Expression level of gene 222356_at.

1554125_a_at Numeric. Expression level of gene 1554125_a_at.

232823_at Numeric. Expression level of gene 232823_at.

205766_at Numeric. Expression level of gene 205766_at.

1560446_at Numeric. Expression level of gene 1560446_at.

202565_s_at Numeric. Expression level of gene 202565_s_at.

234887_at Numeric. Expression level of gene 234887_at.

209687_at Numeric. Expression level of gene 209687_at.

221592_at Numeric. Expression level of gene 221592_at.

1570123_at Numeric. Expression level of gene 1570123_at.

241368_at Numeric. Expression level of gene 241368_at.

243324_x_at Numeric. Expression level of gene 243324_x_at.

224046_s_at Numeric. Expression level of gene 224046_s_at.

202775_s_at Numeric. Expression level of gene 202775_s_at.

216332_at Numeric. Expression level of gene 216332_at.

1569545_at Numeric. Expression level of gene 1569545_at.

205946_at Numeric. Expression level of gene 205946_at.

203547_at Numeric. Expression level of gene 203547_at.

243239_at Numeric. Expression level of gene 243239_at.

234245_at Numeric. Expression level of gene 234245_at.

210832_x_at Numeric. Expression level of gene 210832_x_at.

224549_x_at Numeric. Expression level of gene 224549_x_at.

236628_at Numeric. Expression level of gene 236628_at.

214848_at Numeric. Expression level of gene 214848_at.

1553015_a_at Numeric. Expression level of gene 1553015_a_at.

1554199_at Numeric. Expression level of gene 1554199_at.

1557636_a_at Numeric. Expression level of gene 1557636_a_at.

1558511_s_at Numeric. Expression level of gene 1558511_s_at.

1561713_at Numeric. Expression level of gene 1561713_at.

1561883_at Numeric. Expression level of gene 1561883_at.

1568720_at Numeric. Expression level of gene 1568720_at.

1569168_at Numeric. Expression level of gene 1569168_at.

1569443_s_at Numeric. Expression level of gene 1569443_s_at.

1570103_at Numeric. Expression level of gene 1570103_at.

200916_at Numeric. Expression level of gene 200916_at.

201554_x_at Numeric. Expression level of gene 201554_x_at.

202371_at Numeric. Expression level of gene 202371_at.

204481_at Numeric. Expression level of gene 204481_at.

205831_at Numeric. Expression level of gene 205831_at.

207061_at Numeric. Expression level of gene 207061_at.

207423_s_at Numeric. Expression level of gene 207423_s_at.

209896_s_at Numeric. Expression level of gene 209896_s_at.

212646_at Numeric. Expression level of gene 212646_at.

214068_at Numeric. Expression level of gene 214068_at.

217727_x_at Numeric. Expression level of gene 217727_x_at.

221103_s_at Numeric. Expression level of gene 221103_s_at.

221785_at Numeric. Expression level of gene 221785_at.

224207_x_at Numeric. Expression level of gene 224207_x_at.

228257_at Numeric. Expression level of gene 228257_at.

228877_at Numeric. Expression level of gene 228877_at.

231173_at Numeric. Expression level of gene 231173_at.

231328_s_at Numeric. Expression level of gene 231328_s_at.

231639_at Numeric. Expression level of gene 231639_at.

232221_x_at Numeric. Expression level of gene 232221_x_at.

232349_x_at Numeric. Expression level of gene 232349_x_at.

232849_at Numeric. Expression level of gene 232849_at.

233601_at Numeric. Expression level of gene 233601_at.

234403_at Numeric. Expression level of gene 234403_at.

234585_at Numeric. Expression level of gene 234585_at.

234650_at Numeric. Expression level of gene 234650_at.

234897_s_at Numeric. Expression level of gene 234897_s_at.

236071_at Numeric. Expression level of gene 236071_at.

236689_at Numeric. Expression level of gene 236689_at.

238551_at Numeric. Expression level of gene 238551_at.

239414_at Numeric. Expression level of gene 239414_at.

241034_at Numeric. Expression level of gene 241034_at.

241131_at Numeric. Expression level of gene 241131_at.

241897_at Numeric. Expression level of gene 241897_at.

242611_at Numeric. Expression level of gene 242611_at.

244805_at Numeric. Expression level of gene 244805_at.

244866_at Numeric. Expression level of gene 244866_at.

32259_at Numeric. Expression level of gene 32259_at.

1552264_a_at Numeric. Expression level of gene 1552264_a_at.

1552880_at Numeric. Expression level of gene 1552880_at.

1553186_x_at Numeric. Expression level of gene 1553186_x_at.

1553372_at Numeric. Expression level of gene 1553372_at.

1553438_at Numeric. Expression level of gene 1553438_at.

1554299_at Numeric. Expression level of gene 1554299_at.

1554362_at Numeric. Expression level of gene 1554362_at.

1554491_a_at Numeric. Expression level of gene 1554491_a_at.

1555098_a_at Numeric. Expression level of gene 1555098_a_at.

1555990_at Numeric. Expression level of gene 1555990_at.

1556034_s_at Numeric. Expression level of gene 1556034_s_at.

1556822_s_at Numeric. Expression level of gene 1556822_s_at.

1556824_at Numeric. Expression level of gene 1556824_at.

1557278_s_at Numeric. Expression level of gene 1557278_s_at.

1558603_at Numeric. Expression level of gene 1558603_at.

1558890_at Numeric. Expression level of gene 1558890_at.

1560791_at Numeric. Expression level of gene 1560791_at.

1561083_at Numeric. Expression level of gene 1561083_at.

1561364_at Numeric. Expression level of gene 1561364_at.

1561553_at Numeric. Expression level of gene 1561553_at.

1562523_at Numeric. Expression level of gene 1562523_at.

1562613_at Numeric. Expression level of gene 1562613_at.

1563351_at Numeric. Expression level of gene 1563351_at.

1563473_at Numeric. Expression level of gene 1563473_at.

1566780_at Numeric. Expression level of gene 1566780_at.

1567257_at Numeric. Expression level of gene 1567257_at.

1569664_at Numeric. Expression level of gene 1569664_at.

1569882_at Numeric. Expression level of gene 1569882_at.

1570252_at Numeric. Expression level of gene 1570252_at.

201089_at Numeric. Expression level of gene 201089_at.

201261_x_at Numeric. Expression level of gene 201261_x_at.

202052_s_at Numeric. Expression level of gene 202052_s_at.

202236_s_at Numeric. Expression level of gene 202236_s_at.

202948_at Numeric. Expression level of gene 202948_at.

203080_s_at Numeric. Expression level of gene 203080_s_at.

203211_s_at Numeric. Expression level of gene 203211_s_at.

203218_at Numeric. Expression level of gene 203218_at.

203236_s_at Numeric. Expression level of gene 203236_s_at.

203347_s_at Numeric. Expression level of gene 203347_s_at.

203960_s_at Numeric. Expression level of gene 203960_s_at.

204609_at Numeric. Expression level of gene 204609_at.

204806_x_at Numeric. Expression level of gene 204806_x_at.

204949_at Numeric. Expression level of gene 204949_at.

204979_s_at Numeric. Expression level of gene 204979_s_at.

205823_at Numeric. Expression level of gene 205823_at.

205902_at Numeric. Expression level of gene 205902_at.

205967_at Numeric. Expression level of gene 205967_at.

206186_at Numeric. Expression level of gene 206186_at.

207151_at Numeric. Expression level of gene 207151_at.

207379_at Numeric. Expression level of gene 207379_at.

207440_at Numeric. Expression level of gene 207440_at.

207883_s_at Numeric. Expression level of gene 207883_s_at.

208277_at Numeric. Expression level of gene 208277_at.

208280_at Numeric. Expression level of gene 208280_at.

209224_s_at Numeric. Expression level of gene 209224_s_at.

209561_at Numeric. Expression level of gene 209561_at.

209630_s_at Numeric. Expression level of gene 209630_s_at.

210118_s_at Numeric. Expression level of gene 210118_s_at.

210342_s_at Numeric. Expression level of gene 210342_s_at.

211566_x_at Numeric. Expression level of gene 211566_x_at.

211756_at Numeric. Expression level of gene 211756_at.

212170_at Numeric. Expression level of gene 212170_at.

212494_at Numeric. Expression level of gene 212494_at.

213118_at Numeric. Expression level of gene 213118_at.

214475_x_at Numeric. Expression level of gene 214475_x_at.

214834_at Numeric. Expression level of gene 214834_at.

215718_s_at Numeric. Expression level of gene 215718_s_at.

216283_s_at Numeric. Expression level of gene 216283_s_at.

217206_at Numeric. Expression level of gene 217206_at.

217557_s_at Numeric. Expression level of gene 217557_s_at.

217577_at Numeric. Expression level of gene 217577_at.

218152_at Numeric. Expression level of gene 218152_at.

218252_at Numeric. Expression level of gene 218252_at.

219714_s_at Numeric. Expression level of gene 219714_s_at.

220506_at Numeric. Expression level of gene 220506_at.

220889_s_at Numeric. Expression level of gene 220889_s_at.

221204_s_at Numeric. Expression level of gene 221204_s_at.

221795_at Numeric. Expression level of gene 221795_at.

222048_at Numeric. Expression level of gene 222048_at.

223142_s_at Numeric. Expression level of gene 223142_s_at.

223439_at Numeric. Expression level of gene 223439_at.

223673_at Numeric. Expression level of gene 223673_at.

224363_at Numeric. Expression level of gene 224363_at.

224512_s_at Numeric. Expression level of gene 224512_s_at.

224690_at Numeric. Expression level of gene 224690_at.

224936_at Numeric. Expression level of gene 224936_at.

225334_at Numeric. Expression level of gene 225334_at.

225713_at Numeric. Expression level of gene 225713_at.

225839_at Numeric. Expression level of gene 225839_at.

226041_at Numeric. Expression level of gene 226041_at.

226093_at Numeric. Expression level of gene 226093_at.

226543_at Numeric. Expression level of gene 226543_at.

227695_at Numeric. Expression level of gene 227695_at.

228295_at Numeric. Expression level of gene 228295_at.

228548_at Numeric. Expression level of gene 228548_at.

229234_at Numeric. Expression level of gene 229234_at.

229658_at Numeric. Expression level of gene 229658_at.

229725_at Numeric. Expression level of gene 229725_at.

230252_at Numeric. Expression level of gene 230252_at.

230471_at Numeric. Expression level of gene 230471_at.

231149_s_at Numeric. Expression level of gene 231149_s_at.

231556_at Numeric. Expression level of gene 231556_at.

231754_at Numeric. Expression level of gene 231754_at.

232011_s_at Numeric. Expression level of gene 232011_s_at.

233030_at Numeric. Expression level of gene 233030_at.

234161_at Numeric. Expression level of gene 234161_at.

235050_at Numeric. Expression level of gene 235050_at.

235094_at Numeric. Expression level of gene 235094_at.

235278_at Numeric. Expression level of gene 235278_at.

235671_at Numeric. Expression level of gene 235671_at.

235952_at Numeric. Expression level of gene 235952_at.

236158_at Numeric. Expression level of gene 236158_at.

236181_at Numeric. Expression level of gene 236181_at.

237055_at Numeric. Expression level of gene 237055_at.

237768_x_at Numeric. Expression level of gene 237768_x_at.

238897_at Numeric. Expression level of gene 238897_at.

239160_at Numeric. Expression level of gene 239160_at.

239998_at Numeric. Expression level of gene 239998_at.

240254_at Numeric. Expression level of gene 240254_at.

240612_at Numeric. Expression level of gene 240612_at.

240692_at Numeric. Expression level of gene 240692_at.

240822_at Numeric. Expression level of gene 240822_at.

240842_at Numeric. Expression level of gene 240842_at.

241331_at Numeric. Expression level of gene 241331_at.

241598_at Numeric. Expression level of gene 241598_at.

241927_x_at Numeric. Expression level of gene 241927_x_at.

242405_at Numeric. Expression level of gene 242405_at.

Details

This dataset was extracted from a larger dataset available on ArrayExpress. It is used as a training set for feature selection tasks and other machine learning applications in bioinformatics.

Source

The original dataset can be found on ArrayExpress: https://www.ebi.ac.uk/arrayexpress

References

Ellenbach, N., Boulesteix, A.L., Bischl, B., et al. (2021). Improved Outcome Prediction Across Data Sources Through Robust Parameter Tuning. *Journal of Classification*, 38, 212–231. doi:10.1007/s00357020093682.

Hornung, R., Causeur, D., Bernau, C., Boulesteix, A.L. (2017). Improving cross-study prediction through addon batch effect adjustment or addon normalization. *Bioinformatics*, 33(3), 397–404. doi:10.1093/bioinformatics/btw650.

Examples

```
# Load the dataset:
data(sample_data_train)

# Dimension of the dataset:
dim(sample_data_train)

# View the first rows of the dataset:
head(sample_data_train)
```

---

tuneandtrain *Tune and Train Classifier*

Description

This function tunes and trains a classifier using a specified tuning method. Depending on the method chosen, the function will either perform RobustTuneC, external tuning, or internal tuning.

Usage

```
tuneandtrain(data, dataext = NULL, tuningmethod, classifier, ...)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable, which must be a factor for classification tasks. The remaining columns should be the predictor variables. Ensure that the data is properly formatted, with no missing values.
`dataext`	A data frame containing the external validation data, required only for the tuning methods "robusttunec" and "ext". Similar to the ‘data‘ argument, the first column should be the response variable (factor), and the remaining columns should be the predictors. If ‘tuningmethod = "int"‘, this parameter is ignored.
`tuningmethod`	A character string specifying which tuning approach to use. Options are: • "robusttunec": Uses robust tuning that combines internal and external validation for parameter selection. • "ext": Uses external validation data for tuning the parameters. • "int": Internal cross-validation is used to tune the parameters without any external data.
`classifier`	A character string specifying which classifier to use. Options include: • "boosting": Boosting algorithms for improving weak classifiers. • "rf": Random Forest for robust decision tree-based models. • "lasso": Lasso regression for feature selection and regularization. • "ridge": Ridge regression for regularization. • "svm": Support Vector Machines for high-dimensional classification.
`...`	Additional parameters to be passed to the specific tuning and training functions. These can include options such as the number of trees for Random Forest, the number of folds for cross-validation, or hyperparameters specific to the chosen classifier.

Value

A list containing the results of the tuning and training process, which typically includes:

- Best hyperparameters selected during the tuning process.
- The final trained model.
- Performance metrics (AUC) on the training or validation data, depending on the tuning method.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage: Robust tuning with Ridge classifier
result_boosting <- tuneandtrain(sample_data_train, sample_data_extern,
```

```
  tuningmethod = "robusttunec", classifier = "ridge")
result_boosting$best_lambda
result_boosting$best_model
result_boosting$final_auc

# Example usage: Internal cross-validation with Lasso classifier
result_lasso <- tuneandtrain(sample_data_train, tuningmethod = "int",
  classifier = "lasso", maxit = 120000, nlambda = 200, nfolds = 5)
result_lasso$best_lambda
result_lasso$best_model
result_lasso$final_auc
result_lasso$active_set_Train
```

`tuneandtrainExt` *Tune and Train Classifier by Tuning Method Ext*

Description

This function tunes and trains a classifier using an external validation dataset. Based on the specified classifier, the function selects and runs the appropriate tuning and training process. The external validation data is used to optimize the model's hyperparameters and improve generalization performance across datasets.

Usage

```
tuneandtrainExt(data, dataext, classifier, ...)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables. Ensure that the data is properly formatted, with no missing values.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables. The external data is used for tuning hyperparameters to avoid overfitting on the training data.
`classifier`	A character string specifying the classifier to use. Must be one of the following: • "boosting" for gradient boosting models. • "rf" for Random Forest. • "lasso" for Lasso regression (for feature selection and regularization). • "ridge" for Ridge regression (for regularization). • "svm" for Support Vector Machines (SVM).
`...`	Additional arguments to pass to the specific classifier function. These may include hyperparameters such as the number of trees for Random Forest, regularization parameters for Lasso/Ridge, or kernel settings for SVM.

Value

A list containing the results from the classifier's tuning and training process. The returned object typically includes:

- `best_model`: The final trained model using the best hyperparameters.
- `best_hyperparams`: The optimal hyperparameters found during the tuning process.
- `final_auc`: Performance metrics (AUC) of the final model.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with Lasso
result_lasso <- tuneandtrainExt(sample_data_train, sample_data_extern, classifier = "lasso",
  maxit = 120000, nlambda = 100)
result_lasso$best_lambda
result_lasso$best_model
result_lasso$final_auc
result_lasso$active_set_Train

# Example usage with Ridge
result_ridge <- tuneandtrainExt(sample_data_train, sample_data_extern,
  classifier = "ridge", maxit = 120000, nlambda = 100)
result_ridge$best_lambda
result_ridge$best_model
result_ridge$final_auc
```

`tuneandtrainExtBoost` *Tune and Train External Boosting*

Description

This function tunes and trains a Boosting classifier using the `mboost::glmboost` function. It provides two strategies for tuning the number of boosting iterations (`mstop`) based on the `estperf` argument:

- When `estperf = FALSE` (default): Hyperparameters are tuned using the external validation dataset. The `mstop` value that gives the highest AUC on the external dataset is selected as the best model. However, no AUC value is returned in this case, as per best practices.
- When `estperf = TRUE`: Hyperparameters are tuned internally using the training dataset. The model is then validated on the external dataset to provide a conservative (slightly pessimistic) AUC estimate.

Usage

```
tuneandtrainExtBoost(
  data,
  dataext,
  estperf = FALSE,
  mstop_seq = seq(5, 1000, by = 5),
  nu = 0.1
)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
dataext	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
estperf	A logical value indicating whether to use internal tuning with external validation (TRUE) or external tuning (FALSE). Default is FALSE.
mstop_seq	A numeric vector specifying the sequence of boosting iterations to evaluate. Default is seq(5, 1000, by = 5).
nu	A numeric value specifying the learning rate for boosting. Default is 0.1.

Value

A list containing the following components:

- best_mstop: The optimal number of boosting iterations determined during the tuning process.
- best_model: The trained Boosting model using the selected mstop.
- est_auc: The AUC value evaluated on the external dataset. This is only returned when estperf = TRUE, providing a conservative (slightly pessimistic) estimate of the model's performance.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with external tuning (default)
mstop_seq <- seq(50, 500, by = 50)
result <- tuneandtrainExtBoost(sample_data_train, sample_data_extern,
  mstop_seq = mstop_seq, nu = 0.1)
print(result$best_mstop)         # Optimal mstop
print(result$best_model)         # Trained Boosting model
# Note: est_auc is not returned when estperf = FALSE

# Example usage with internal tuning and external validation
```

```
result_internal <- tuneandtrainExtBoost(sample_data_train, sample_data_extern,
  estperf = TRUE, mstop_seq = mstop_seq, nu = 0.1)
print(result_internal$best_mstop) # Optimal mstop
print(result_internal$best_model) # Trained Boosting model
print(result_internal$est_auc)    # AUC on external validation dataset
```

`tuneandtrainExtLasso` *Tune and Train External Lasso*

Description

This function tunes and trains a Lasso classifier using the `glmnet` package. It provides two strategies for tuning hyperparameters based on the `estperf` argument:

- When `estperf = FALSE` (default): Hyperparameters are tuned using the external validation dataset. The lambda value that gives the highest AUC on the external dataset is selected as the best model. However, no AUC value is returned in this case, as per best practices.
- When `estperf = TRUE`: Hyperparameters are tuned internally using the training dataset. The model is then validated on the external dataset to provide a conservative (slightly pessimistic) AUC estimate.

Usage

```
tuneandtrainExtLasso(
  data,
  dataext,
  estperf = FALSE,
  maxit = 120000,
  nlambda = 100
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`estperf`	A logical value indicating whether to use internal tuning with external validation (`TRUE`) or external tuning (`FALSE`). Default is `FALSE`.
`maxit`	An integer specifying the maximum number of iterations. Default is 120000.
`nlambda`	An integer specifying the number of lambda values to use in the Lasso model. Default is 100.

Value

A list containing the following components:

- `best_lambda`: The optimal lambda value determined during the tuning process.
- `best_model`: The trained Lasso model using the selected lambda value.
- `est_auc`: The AUC value evaluated on the external dataset. This is only returned when `estperf = TRUE`, providing a conservative (slightly pessimistic) estimate of the model's performance.
- `active_set_Train`: The number of active coefficients (non-zero) in the model trained on the training dataset.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with external tuning (default)
result <- tuneandtrainExtLasso(sample_data_train, sample_data_extern, maxit = 120000, nlambda = 100)
print(result$best_lambda)
print(result$best_model)
print(result$active_set_Train)

# Example usage with internal tuning and external validation
result_internal <- tuneandtrainExtLasso(sample_data_train, sample_data_extern,
  estperf = TRUE, maxit = 120000, nlambda = 100)
print(result_internal$best_lambda)
print(result_internal$best_model)
print(result_internal$est_auc)
print(result_internal$active_set_Train)
```

---

tuneandtrainExtRF *Tune and Train External Random Forest*

Description

This function tunes and trains a Random Forest classifier using the `ranger` package. It provides two strategies for tuning the `min.node.size` parameter based on the `estperf` argument:

- When `estperf = FALSE` (default): Hyperparameters are tuned using the external validation dataset. The `min.node.size` value that gives the highest AUC on the external dataset is selected as the best model. However, no AUC value is returned in this case, as per best practices.
- When `estperf = TRUE`: Hyperparameters are tuned internally using the training dataset. The model is then validated on the external dataset to provide a conservative (slightly pessimistic) AUC estimate.

Usage

```
tuneandtrainExtRF(data, dataext, estperf = FALSE, num.trees = 500)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
dataext	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
estperf	A logical value indicating whether to use internal tuning with external validation (TRUE) or external tuning (FALSE). Default is FALSE.
num.trees	An integer specifying the number of trees in the Random Forest. Default is 500.

Value

A list containing the following components:

- best_min_node_size: The optimal min.node.size value determined during the tuning process.
- best_model: The trained Random Forest model using the selected min.node.size.
- est_auc: The AUC value evaluated on the external dataset. This is only returned when estperf = TRUE, providing a conservative (slightly pessimistic) estimate of the model's performance.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with external tuning (default)
result <- tuneandtrainExtRF(sample_data_train, sample_data_extern, num.trees = 500)
print(result$best_min_node_size)  # Optimal min.node.size
print(result$best_model)          # Trained Random Forest model
# Note: est_auc is not returned when estperf = FALSE

# Example usage with internal tuning and external validation
result_internal <- tuneandtrainExtRF(sample_data_train, sample_data_extern,
  estperf = TRUE, num.trees = 500)
print(result_internal$best_min_node_size)  # Optimal min.node.size
print(result_internal$best_model)          # Trained Random Forest model
print(result_internal$est_auc)             # AUC on external validation dataset
```

tuneandtrainExtRidge *Tune and Train External Ridge*

Description

This function tunes and trains a Ridge classifier using the `glmnet` package. It provides two strategies for tuning the regularization parameter `lambda` based on the `estperf` argument:

- When `estperf = FALSE` (default): Hyperparameters are tuned using the external validation dataset. The `lambda` value that gives the highest AUC on the external dataset is selected as the best model. However, no AUC value is returned in this case, as per best practices.
- When `estperf = TRUE`: Hyperparameters are tuned internally using the training dataset. The model is then validated on the external dataset to provide a conservative (slightly pessimistic) AUC estimate.

Usage

```
tuneandtrainExtRidge(
  data,
  dataext,
  estperf = FALSE,
  maxit = 120000,
  nlambda = 100
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`estperf`	A logical value indicating whether to use internal tuning with external validation (`TRUE`) or external tuning (`FALSE`). Default is `FALSE`.
`maxit`	An integer specifying the maximum number of iterations. Default is 120000.
`nlambda`	An integer specifying the number of lambda values to use in the Ridge model. Default is 100.

Value

A list containing the following components:

- `best_lambda`: The optimal `lambda` value determined during the tuning process.
- `best_model`: The trained Ridge model using the selected `lambda`.
- `est_auc`: The AUC value evaluated on the external dataset. This is only returned when `estperf = TRUE`, providing a conservative (slightly pessimistic) estimate of the model's performance.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with external tuning (default)
result <- tuneandtrainExtRidge(sample_data_train, sample_data_extern, maxit = 120000, nlambda = 100)
print(result$best_lambda)       # Optimal lambda
print(result$best_model)        # Final trained model
# Note: est_auc is not returned when estperf = FALSE

# Example usage with internal tuning and external validation
result_internal <- tuneandtrainExtRidge(sample_data_train, sample_data_extern,
  estperf = TRUE, maxit = 120000, nlambda = 100)
print(result_internal$best_lambda)  # Optimal lambda
print(result_internal$best_model)   # Final trained model
print(result_internal$est_auc)      # AUC on external validation dataset
```

---

tuneandtrainExtSVM *Tune and Train External SVM*

Description

This function tunes and trains a Support Vector Machine (SVM) classifier using the `mlr` package. It provides two strategies for tuning the cost parameter based on the `estperf` argument:

- When `estperf = FALSE` (default): Hyperparameters are tuned using the external validation dataset. The `cost` value that gives the highest AUC on the external dataset is selected as the best model. However, no AUC value is returned in this case, as per best practices.
- When `estperf = TRUE`: Hyperparameters are tuned internally using the training dataset. The model is then validated on the external dataset to provide a conservative (slightly pessimistic) AUC estimate.

Usage

```
tuneandtrainExtSVM(
  data,
  dataext,
  estperf = FALSE,
  kernel = "linear",
  cost_seq = 2^(-15:15),
  scale = FALSE
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.

dataext	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
estperf	A logical value indicating whether to use internal tuning with external validation (TRUE) or external tuning (FALSE). Default is FALSE.
kernel	A character string specifying the kernel type to be used in the SVM. Default is "linear".
cost_seq	A numeric vector specifying the sequence of cost values to evaluate. Default is 2^(-15:15).
scale	A logical value indicating whether to scale the predictor variables. Default is FALSE.

Value

A list containing the following components:

- best_cost: The optimal cost value determined during the tuning process.
- best_model: The trained SVM model using the selected cost.
- est_auc: The AUC value evaluated on the external dataset. This is only returned when estperf = TRUE, providing a conservative (slightly pessimistic) estimate of the model's performance.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with external tuning (default)
result <- tuneandtrainExtSVM(sample_data_train, sample_data_extern, kernel = "linear",
  cost_seq = 2^(-15:15), scale = FALSE)
print(result$best_cost)        # Optimal cost
print(result$best_model)       # Final trained model
# Note: est_auc is not returned when estperf = FALSE

# Example usage with internal tuning and external validation
result_internal <- tuneandtrainExtSVM(sample_data_train, sample_data_extern,
  estperf = TRUE, kernel = "linear", cost_seq = 2^(-15:15), scale = FALSE)
print(result_internal$best_cost)  # Optimal cost
print(result_internal$best_model) # Final trained model
print(result_internal$est_auc)    # AUC on external validation dataset
```

tuneandtrainInt *Tune and Train by tuning method Int*

Description

This function tunes and trains a specified classifier using internal cross-validation. The classifier is specified by the 'classifier' argument, and the function delegates to the appropriate tuning and training function based on this choice.

Usage

```
tuneandtrainInt(data, classifier, ...)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
classifier	A character string specifying the classifier to use. Must be one of 'boosting', 'rf', 'lasso', 'ridge', 'svm'.
...	Additional arguments to pass to the specific classifier function.

Value

A list containing the results from the specific classifier's tuning and training process. The list typically includes:

- best_hyperparams: The best hyperparameters selected by cross-validation.
- best_model: The final trained model using the selected hyperparameters.
- final_auc: Cross-validation results (AUC).

Examples

```
# Load sample data
data(sample_data_train)

# Example usage with Lasso
result_lasso <- tuneandtrainInt(sample_data_train, classifier = "lasso",
  maxit = 120000, nlambda = 100)
result_lasso$best_lambda
result_lasso$best_model
result_lasso$final_auc
result_lasso$active_set_Train

# Example usage with Ridge
result_ridge <- tuneandtrainInt(sample_data_train, classifier = "ridge",
  maxit = 120000, nlambda = 100)
result_ridge$best_lambda
```

```
result_ridge$best_model
result_ridge$final_auc
```

`tuneandtrainIntBoost` *Tune and Train Internal Boosting*

Description

This function tunes and trains a Boosting classifier using the `mboost` package. The function evaluates a sequence of boosting iterations on the training dataset using internal cross-validation and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainIntBoost(data, mstop_seq = seq(5, 1000, by = 5), nu = 0.1)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`mstop_seq`	A numeric vector of boosting iterations to be evaluated. Default is a sequence from 5 to 1000 with a step of 5.
`nu`	A numeric value for the learning rate. Default is 0.1.

Details

This function performs K-fold cross-validation on the training dataset, where the number of boosting iterations (`mstop`) is tuned to maximize the AUC. The optimal number of boosting iterations is selected, and the final model is trained on the entire training dataset.

Value

A list containing the best number of boosting iterations (‘best_mstop‘) and the final Boosting classifier model (‘best_model‘).

Examples

```
# Load sample data
data(sample_data_train)

# Example usage
mstop_seq <- seq(5, 5000, by = 5)
result <- tuneandtrainIntBoost(sample_data_train, mstop_seq, nu = 0.1)
result$best_mstop
result$best_model
```

`tuneandtrainIntLasso` *Tune and Train Internal Lasso*

Description

This function tunes and trains a Lasso classifier using the `glmnet` package. The function performs internal cross-validation to evaluate a sequence of lambda (regularization) values and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainIntLasso(data, maxit = 120000, nlambda = 200, nfolds = 5)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`maxit`	An integer specifying the maximum number of iterations. Default is 120000.
`nlambda`	An integer specifying the number of lambda values to use in the Lasso model. Default is 200.
`nfolds`	An integer specifying the number of folds for cross-validation. Default is 5.

Details

This function trains a logistic Lasso model on the training dataset using cross-validation. The lambda value that results in the highest AUC during cross-validation is chosen as the best model, and the final model is trained on the full training dataset with this optimal lambda value.

Value

A list containing the best lambda value (‘best_lambda‘), the final trained model (‘best_model‘), and the number of active coefficients (‘active_set_Train‘).

Examples

```
# Load sample data
data(sample_data_train)

# Example usage
result <- tuneandtrainIntLasso(sample_data_train, maxit = 120000, nlambda = 200, nfolds = 5)
result$best_lambda
result$best_model
result$active_set_Train
```

tuneandtrainIntRF *Tune and Train Internal Random Forest*

Description

This function tunes and trains a Random Forest classifier using the ranger package with internal cross-validation. The function evaluates a sequence of min.node.size values on the training dataset and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainIntRF(data, num.trees = 500, nfolds = 5, seed = 123)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
num.trees	An integer specifying the number of trees in the Random Forest. Default is 500.
nfolds	An integer specifying the number of folds for cross-validation. Default is 5.
seed	An integer specifying the random seed for reproducibility. Default is 123.

Details

Random Forest constructs multiple decision trees and aggregates their predictions. The min.node.size parameter controls the minimum number of samples in each terminal node, affecting model complexity. This function performs cross-validation within the training dataset to evaluate the impact of different min.node.size values. The min.node.size value that results in the highest AUC is selected as the best model.

Value

A list containing the best ‘min.node.size‘ value (‘best_min_node_size‘) and the final trained model (‘best_model‘).

Examples

```
# Load sample data
data(sample_data_train)

# Example usage
result <- tuneandtrainIntRF(sample_data_train, num.trees = 500, nfolds = 5, seed = 123)
result$best_min_node_size
result$best_model
```

tuneandtrainIntRidge *Tune and Train Internal Ridge*

Description

This function tunes and trains a Ridge classifier using the `glmnet` package. The function evaluates a sequence of lambda (regularization) values using internal cross-validation and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainIntRidge(
  data,
  maxit = 120000,
  nlambda = 200,
  nfolds = 5,
  seed = 123
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`maxit`	An integer specifying the maximum number of iterations. Default is 120000.
`nlambda`	An integer specifying the number of lambda values to use in the Ridge model. Default is 200.
`nfolds`	An integer specifying the number of folds for cross-validation. Default is 5.
`seed`	An integer specifying the random seed for reproducibility. Default is 123.

Details

The function trains a logistic Ridge regression model on the training dataset and performs cross-validation to select the best lambda value. The lambda value that gives the highest AUC on the training dataset during cross-validation is chosen as the best model.

Value

A list containing the best lambda value (‘best_lambda‘) and the final trained model (‘best_model‘).

Examples

```
# Load sample data
data(sample_data_train)

# Example usage
result <- tuneandtrainIntRidge(sample_data_train, maxit = 120000,
```

```
  nlambda = 200, nfolds = 5, seed = 123)
result$best_lambda
result$best_model
```

---

tuneandtrainIntSVM *Tune and Train Internal SVM*

Description

This function tunes and trains a Support Vector Machine (SVM) classifier using the `mlr` package. The function evaluates a sequence of cost values using internal cross-validation and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainIntSVM(
  data,
  kernel = "linear",
  cost_seq = 2^(-15:15),
  scale = FALSE,
  nfolds = 5,
  seed = 123
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`kernel`	A character string specifying the kernel type to be used in the SVM. Default is "linear".
`cost_seq`	A numeric vector of cost values to be evaluated. Default is '2^(-15:15)'.
`scale`	A logical indicating whether to scale the predictor variables. Default is FALSE.
`nfolds`	An integer specifying the number of folds for cross-validation. Default is 5.
`seed`	An integer specifying the random seed for reproducibility. Default is 123.

Details

In Support Vector Machines, the `cost` parameter controls the trade-off between achieving a low training error and a low testing error. This function trains an SVM model on the training dataset, performs cross-validation, and selects the cost value that results in the highest AUC. The final model is then trained using the optimal cost value, and the performance is reported based on the AUC.

Value

A list containing the best cost value ('best_cost') and the final trained model ('best_model').

Examples

```
# Load sample data
data(sample_data_train)

# Example usage
result <- tuneandtrainIntSVM(
  sample_data_train,
  kernel = "linear",
  cost_seq = 2^(-15:15),
  scale = FALSE,
  nfolds = 5,
  seed = 123
)
result$best_cost
result$best_model
```

---

tuneandtrainRobustTuneC

Tune and Train Classifier by Tuning Method RobustTuneC

Description

This function tunes and trains a specified classifier using the "RobustTuneC" method and the provided data.

Usage

```
tuneandtrainRobustTuneC(data, dataext, classifier, ...)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`classifier`	A character string specifying the classifier to use. Must be one of the following: • "boosting" for Boosting classifiers. • "rf" for Random Forest. • "lasso" for Lasso regression. • "ridge" for Ridge regression. • "svm" for Support Vector Machines.
`...`	Additional arguments to pass to the specific classifier function.

Value

A list containing the results from the specific classifier's tuning and training process, the returned object typically includes:

- `best_hyperparams`: The best hyperparameters selected through the RobustTuneC method.
- `best_model`: The final trained model based on the best hyperparameters.
- `final_auc`: Performance metrics (AUC) of the final model.

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with Lasso
result_lasso <- tuneandtrainRobustTuneC(sample_data_train, sample_data_extern, classifier = "lasso",
  maxit = 120000, nlambda = 100)
result_lasso$best_lambda
result_lasso$best_model
result_lasso$final_auc
result_lasso$active_set_Train

# Example usage with Ridge
result_ridge <- tuneandtrainRobustTuneC(sample_data_train, sample_data_extern,
  classifier = "ridge", maxit = 120000, nlambda = 100)
result_ridge$best_lambda
result_ridge$best_model
result_ridge$final_auc
```

---

tuneandtrainRobustTuneCBoost

Tune and Train RobustTuneC Boosting

Description

This function tunes and trains a Boosting classifier using the `mboost::glmboost` function and the "RobustTuneC" method. The function performs K-fold cross-validation on the training dataset and evaluates a sequence of boosting iterations (`mstop`) based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainRobustTuneCBoost(
  data,
  dataext,
  K = 5,
  mstop_seq = seq(5, 1000, by = 5),
  nu = 0.1
)
```

Arguments

data	Training data as a data frame. The first column should be the response variable.
dataext	External validation data as a data frame. The first column should be the response variable.
K	Number of folds to use in cross-validation. Default is 5.
mstop_seq	A sequence of boosting iterations to consider. Default is a sequence starting at 5 and increasing by 5 each time, up to 1000.
nu	Learning rate for the boosting algorithm. Default is 0.1.

Details

After cross-validation, the best mstop value is selected based on the AUC, and the final Boosting model is trained using this optimal mstop. The external validation dataset is then used to calculate the final AUC and assess the model performance.

Value

A list containing the best number of boosting iterations (‘best_mstop‘), the final trained model (‘best_model‘), and the chosen c value(‘best_c‘).

Examples

```
# Load the sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage with the sample data
mstop_seq <- seq(50, 500, by = 50)
result <- tuneandtrainRobustTuneCBoost(sample_data_train, sample_data_extern, mstop_seq = mstop_seq)
result$best_mstop
result$best_model
result$best_c
```

---

tuneandtrainRobustTuneCLasso

Tune and Train RobustTuneC Lasso

Description

This function tunes and trains a Lasso classifier using the glmnet package and the "RobustTuneC" method. The function uses K-fold cross-validation to evaluate a sequence of lambda (regularization) values and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainRobustTuneCLasso(
  data,
  dataext,
  K = 5,
  maxit = 120000,
  nlambda = 100
)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
dataext	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
K	Number of folds to use in cross-validation. Default is 5.
maxit	Maximum number of iterations. Default is 120000.
nlambda	The number of lambda values to use for cross-validation. Default is 100.

Details

This function trains a logistic Lasso model using the training dataset and validates it through cross-validation. After selecting the best lambda value based on the training data, the model is then applied to an external validation dataset to compute the final AUC. The lambda value that results in the highest AUC on the external validation dataset is chosen as the best model.

Value

A list containing the best lambda value (‘best_lambda‘), the final trained model (‘best_model‘), the number of active coefficients (‘active_set_Train‘), and the chosen c value(‘best_c‘).

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage
result <- tuneandtrainRobustTuneCLasso(sample_data_train, sample_data_extern,
  K = 5, maxit = 120000, nlambda = 100)
result$best_lambda
result$best_model
result$best_c
```

tuneandtrainRobustTuneCRF

Tune and Train RobustTuneC Random Forest

Description

This function tunes and trains a Random Forest classifier using the `ranger` package and the "RobustTuneC" method. The function uses K-fold cross-validation to evaluate different `min.node.size` values on the training dataset and selects the best model based on the Area Under the Curve (AUC).

Usage

```
tuneandtrainRobustTuneCRF(data, dataext, K = 5, num.trees = 500)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`K`	Number of folds to use in cross-validation. Default is 5.
`num.trees`	An integer specifying the number of trees to grow in the Random Forest. Default is 500.

Details

Random Forest constructs multiple decision trees and aggregates their predictions. The `min.node.size` parameter controls the minimum number of samples in each terminal node, affecting model complexity. This function evaluates the `min.node.size` values through cross-validation and then applies the best model to an external validation dataset. The `min.node.size` value that results in the highest AUC on the validation dataset is selected.

Value

A list containing the best minimum node size (‘best_min_node_size‘), the final trained model (‘best_model‘), and the chosen c value(‘best_c‘).

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage
result <- tuneandtrainRobustTuneCRF(sample_data_train, sample_data_extern, K = 5, num.trees = 500)
```

```
result$best_min_node_size
result$best_model
result$best_c
```

`tuneandtrainRobustTuneCRidge` *Tune and Train RobustTuneC Ridge*

Description

This function tunes and trains a Ridge classifier using the `glmnet` package with the "RobustTuneC" method. The function evaluates a sequence of lambda (regularization) values using K-fold cross-validation (K specified by the user) on the training dataset and selects the best model based on Area Under the Curve (AUC).

Usage

```
tuneandtrainRobustTuneCRidge(
  data,
  dataext,
  K = 5,
  maxit = 120000,
  nlambda = 100
)
```

Arguments

`data`	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`dataext`	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
`K`	Number of folds to use in cross-validation. Default is 5.
`maxit`	Maximum number of iterations. Default is 120000.
`nlambda`	The number of lambda values to use for cross-validation. Default is 100.

Details

The function first performs K-fold cross-validation on the training dataset to select the best lambda value based on AUC. Then, the model is further validated on an external dataset, and the lambda value that provides the best performance on the external dataset is chosen as the final model. The Ridge regression is fitted using the selected lambda value, and the final model's performance is evaluated using AUC on the external validation dataset.

Value

A list containing the best lambda value (‘best_lambda‘), the final trained model (‘best_model‘), and the chosen c value(‘best_c‘).

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage
result <- tuneandtrainRobustTuneCRidge(sample_data_train, sample_data_extern,
  K = 5, maxit = 120000, nlambda = 100)
result$best_lambda
result$best_model
result$best_c
```

---

tuneandtrainRobustTuneCSVM

Tune and Train RobustTuneC Support Vector Machine (SVM)

Description

This function tunes and trains a Support Vector Machine (SVM) classifier using the "RobustTuneC" method. It performs K-fold cross-validation (with K specified by the user) to select the best model based on the Area Under the Curve (AUC) metric.

Usage

```
tuneandtrainRobustTuneCSVM(
  data,
  dataext,
  K = 5,
  seed = 123,
  kernel = "linear",
  cost_seq = 2^(-15:15),
  scale = FALSE
)
```

Arguments

data	A data frame containing the training data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.
dataext	A data frame containing the external validation data. The first column should be the response variable (factor), and the remaining columns should be the predictor variables.

K	Number of folds to use in cross-validation. Default is 5.
seed	An integer specifying the random seed for reproducibility. Default is 123.
kernel	A character string specifying the kernel type to be used in the SVM. It can be "linear", "polynomial", "radial", or "sigmoid". Default is "linear".
cost_seq	A numeric vector of cost values to be evaluated. Default is '2^(-15:15)'.
scale	A logical value indicating whether to scale the predictor variables. Default is 'FALSE'.

Details

In Support Vector Machines, the cost parameter controls the trade-off between achieving a low training error and a low testing error. This function trains an SVM model on the training dataset, performs cross-validation to evaluate different cost values, and selects the one that yields the highest AUC. The final model is trained using the optimal cost value, and its performance is reported using the AUC metric on the external validation dataset.

Value

A list containing the best cost value ('best_cost'), the final trained model ('best_model'), and the chosen c value('best_c').

Examples

```
# Load sample data
data(sample_data_train)
data(sample_data_extern)

# Example usage
result <- tuneandtrainRobustTuneCSVM(sample_data_train, sample_data_extern, K = 5, seed = 123,
                                     kernel = "linear", cost_seq = 2^(-15:15), scale = FALSE)
result$best_cost
result$best_model
result$best_c
```

Index