Machine Learning with R at LRZ: Introduction to mlr

Spam E-mail Database

Description

A data set collected at Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam. In addition to this class label there are 57 variables indicating the frequency of certain words and characters in the e-mail.

Format

A data frame with 4601 observations and 58 variables.

The first 48 variables contain the frequency of the variable name (e.g., business) in the e-mail. If the variable name starts with num (e.g., num650) the it indicates the frequency of the corresponding number (e.g., 650). The variables 49-54 indicate the frequency of the characters ';', '(', '[', '!', '\$', and '\#'. The variables 55-57 contain the average, longest and total run-length of capital letters. Variable 58 indicates the type of the mail and is either `"nonspam"` or `"spam"`, i.e. unsolicited commercial e-mail.

Details

The data set contains 2788 e-mails classified as `"nonspam"` and 1813 classified as `"spam"`.

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... This collection of spam e-mails came from the collectors' postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Source

- Creators: Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt at Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304
- Donor: George Forman (gforman at nospam hpl.hp.com) 650-857-7835

These data have been taken from the UCI Repository Of Machine Learning Databases at http://www.ics.uci.edu/~mlearn/MLRepository.html

References

T. Hastie, R. Tibshirani, J.H. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

Exercise

a) Create a binary classifcation task from the spam data

```
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
data(spam, package = "kernlab")
spam.task = makeClassifTask(id = "spam", data = spam, target = "type", positive = "spam")
spam.task
```

```
## Supervised task: spam
## Type: classif
## Target: type
## Observations: 4601
## Features:
##    numerics     factors     ordered functionals
##          57           0           0           0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
## Has coordinates: FALSE
## Classes: 2
## nonspam    spam
##    2788    1813
## Positive class: spam
```

b) List all learners that could be trained on `spam.task`

```
listLearners(spam.task, warn.missing.packages = FALSE)
```

```
##                   class                              name  short.name
## 1           classif.ada                      ada Boosting         ada
## 2   classif.adaboostm1                   ada Boosting M1  adaboostm1
## 3 classif.bartMachine Bayesian Additive Regression Trees bartmachine
## 4      classif.binomial               Binomial Regression    binomial
## 5      classif.boosting                   Adabag Boosting      adabag
## 6           classif.bst                 Gradient Boosting         bst
##         package    type installed numerics factors ordered missings weights
## 1    ada,rpart classif      TRUE     TRUE    TRUE   FALSE    FALSE   FALSE
## 2        RWeka classif      TRUE     TRUE    TRUE   FALSE    FALSE   FALSE
## 3  bartMachine classif      TRUE     TRUE    TRUE   FALSE     TRUE   FALSE
## 4        stats classif      TRUE     TRUE    TRUE   FALSE    FALSE    TRUE
## 5 adabag,rpart classif      TRUE     TRUE    TRUE   FALSE     TRUE   FALSE
## 6    bst,rpart classif      TRUE     TRUE   FALSE   FALSE    FALSE   FALSE
##    prob oneclass twoclass multiclass class.weights featimp oobpreds
## 1  TRUE    FALSE     TRUE      FALSE         FALSE   FALSE    FALSE
## 2  TRUE    FALSE     TRUE       TRUE         FALSE   FALSE    FALSE
## 3  TRUE    FALSE     TRUE      FALSE         FALSE   FALSE    FALSE
## 4  TRUE    FALSE     TRUE      FALSE         FALSE   FALSE    FALSE
## 5  TRUE    FALSE     TRUE       TRUE         FALSE    TRUE    FALSE
## 6 FALSE    FALSE     TRUE      FALSE         FALSE   FALSE    FALSE
##   functionals single.functional    se lcens rcens icens
## 1       FALSE             FALSE FALSE FALSE FALSE FALSE
## 2       FALSE             FALSE FALSE FALSE FALSE FALSE
## 3       FALSE             FALSE FALSE FALSE FALSE FALSE
```

```
## 4       FALSE             FALSE FALSE FALSE FALSE FALSE
## 5       FALSE             FALSE FALSE FALSE FALSE FALSE
## 6       FALSE             FALSE FALSE FALSE FALSE FALSE
## ... (#rows: 78, #cols: 24)
```

c) Select a learner you like and create it. If you want to can change its hyperparameters

```
lrn = makeLearner("classif.rpart", predict.type = "prob")
```

d) Create an index set of train and test indicies. The test set should have 1000 observations.

d*) Ensure that the fraction between `"spam"` and `"nonspam"` is the training and test set is the same as in the full dataset.

```
n = getTaskSize(spam.task)
test.inds = sample(1:n, size = 1000)
train.inds = setdiff(1:n, test.inds)
head(test.inds)
```

```
## [1] 4467 2970  740 1431 3859  515
```

```
head(train.inds)
```

```
## [1] 1 2 4 5 6 7
```

e) Train your model on the train subset of the spam data and predict on the test subset.

```
mod = train(lrn, spam.task, subset = train.inds)
preds = predict(mod, spam.task, subset = test.inds)
print(mod)
```

```
## Model for learner.id=classif.rpart; learner.class=classif.rpart
## Trained on: task.id = spam; obs = 3601; features = 57
## Hyperparameters: xval=0
```

```
print(preds)
```

```
## Prediction: 1000 observations
## predict.type: prob
## threshold: nonspam=0.50,spam=0.50
## time: 0.01
##        id   truth prob.nonspam prob.spam response
## 4467 4467 nonspam      0.92990    0.0701  nonspam
## 2970 2970 nonspam      0.92990    0.0701  nonspam
## 740   740    spam      0.16552    0.8345     spam
## 1431 1431    spam      0.19799    0.8020     spam
## 3859 3859 nonspam      0.92990    0.0701  nonspam
## 515   515    spam      0.04021    0.9598     spam
## ... (#rows: 1000, #cols: 5)
```

f) Evaluate the performance of your model based on accuracy and area under the curve.

```
perf = performance(preds, measures = list(acc, auc))
perf
```

```
##    acc    auc
## 0.8850 0.8992
```

g) Try to find a model with an AUC of at least 98%.

- Try different models
- Change hyperparameters

- Have a closer look at the feature and try to find transformations or combination of features that improve your model's performance

```
lrn2 = makeLearner("classif.randomForest", predict.type = "prob")
mod = train(lrn2, spam.task, subset = train.inds)
preds = predict(mod, spam.task, subset = test.inds)
performance(preds, measures = auc)
```

```
##    auc
## 0.9846
```