COSC 5557: Practical ML

Warm Up Exercise

Abiodun Awosola

2024-03-02

Loading Basic Packages

```
knitr::opts_chunk$set(comment = NA) # removes '##' from outputs

# .Rprofile   (# allows universal package installation)
options(repos = c(
  mlrorg = "https://mlr-org.r-universe.dev",
  CRAN = "https://cloud.r-project.org/"
))


library(mlr3verse)
library(mlr3learners)

lgr::get_logger("mlr3")$set_threshold("warn")
```

Creating Tasks and Learners

```
# Imports Data

wine_data <- read.table("winequality-white.csv", sep = ";",
                        check.names = TRUE, header=T)

str(wine_data)
```

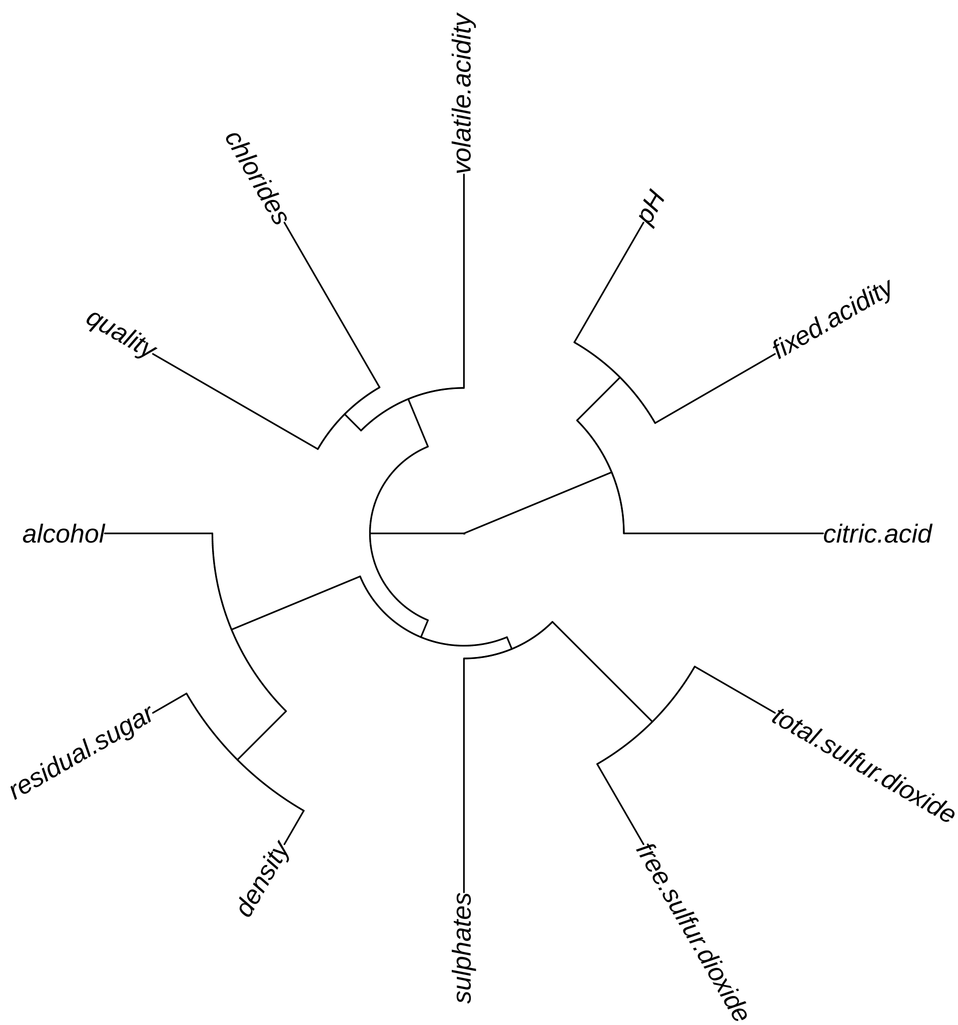
```
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

```r
# Creates mlr3 task;
# target is the column to be learnt

wine_tsk = as_task_classif(wine_data, target = "quality")

print(wine_tsk)
```

```
<TaskClassif:wine_data> (4898 x 12)
* Target: quality
* Properties: multiclass
* Features (11):
  - dbl (11): alcohol, chlorides, citric.acid, density, fixed.acidity,
    free.sulfur.dioxide, pH, residual.sugar, sulphates,
    total.sulfur.dioxide, volatile.acidity
```

Phylogenetic tree

```r
# phylogenetic tree

library(ClustOfVar)
library(ape)
vctree <- hclustvar(wine_data)
plot(as.phylo(vctree), type = "fan")
```




```r
# creates learner
# 2 equivalent calls:

learner_1 = mlr_learners$get("classif.rpart")
learner_1 = lrn("classif.rpart")
print(learner_1)
```

```
<LearnerClassifRpart:classif.rpart>: Classification Tree
* Model: -
```

```
* Parameters: xval=0
* Packages: mlr3, rpart
* Predict Types:  [response], prob
* Feature Types: logical, integer, numeric, factor, ordered
* Properties: importance, missings, multiclass, selected_features,
  twoclass, weights
```

Train and Predict

```r
# trains learner on subset of task
learner_1$train(wine_tsk, row_ids = 1:3918)

# this is what the decision tree looks like
print(learner_1$model)
```

```
n= 3918

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 3918 2237 6 (0.0048 0.035 0.3 0.43 0.19 0.04 0.0013)
   2) alcohol< 10.85 2560 1438 6 (0.0043 0.044 0.41 0.44 0.091 0.011 0.00039)
     4) volatile.acidity>=0.2425 1466  703 5 (0.0048 0.063 0.52 0.36 0.046 0.0041 0.00068)
       8) alcohol< 9.75 897  353 5 (0.0045 0.057 0.61 0.3 0.03 0.0011 0) *
       9) alcohol>=9.75 569  310 6 (0.0053 0.072 0.38 0.46 0.072 0.0088 0.0018)
        18) free.sulfur.dioxide< 17.5 108   53 5 (0.0093 0.2 0.51 0.23 0.046 0 0) *
        19) free.sulfur.dioxide>=17.5 461  227 6 (0.0043 0.041 0.36 0.51 0.078 0.011 0.0022) *
     5) volatile.acidity< 0.2425 1094  501 6 (0.0037 0.018 0.27 0.54 0.15 0.02 0) *
   3) alcohol>=10.85 1358  799 6 (0.0059 0.019 0.1 0.41 0.37 0.094 0.0029)
     6) alcohol< 12.55 1100  612 6 (0.0064 0.02 0.12 0.44 0.33 0.081 0.0018) *
     7) alcohol>=12.55 258  124 7 (0.0039 0.016 0.031 0.28 0.52 0.15 0.0078) *
```

```r
# predicts using observations from task
prediction = learner_1$predict(wine_tsk, row_ids = 3919:4898)
print(prediction)
```

```
<PredictionClassif> for 980 observations:
    row_ids truth response
       3919     7        7
       3920     6        7
       3921     6        6
---
       4896     6        6
       4897     7        7
       4898     6        6
```

Evaluation

Scoring the Prediction object with some metrics. And take a deeper look by inspecting the confusion matrix.

```r
head(as.data.table(mlr_measures))
```

```
              key                          label task_type          packages
1:            aic   Akaike Information Criterion      <NA>              mlr3
```

```
2:            bic Bayesian Information Criterion      <NA>              mlr3
3:    classif.acc        Classification Accuracy   classif mlr3,mlr3measures
4:    classif.auc       Area Under the ROC Curve   classif mlr3,mlr3measures
5:   classif.bacc              Balanced Accuracy   classif mlr3,mlr3measures
6: classif.bbrier             Binary Brier Score   classif mlr3,mlr3measures
   predict_type task_properties
1:         <NA>
2:         <NA>
3:     response
4:         prob        twoclass
5:     response
6:         prob        twoclass
```

```r
scores = prediction$score(msr("classif.acc"))
print(scores)
```

```
classif.acc
  0.5877551
```

```r
scores = prediction$score(msrs(c("classif.acc", "classif.ce")))
print(scores)
```

```
classif.acc  classif.ce
  0.5877551   0.4122449
```

Confusion matrix

```r
cm = prediction$confusion
print(cm)
```

```
        truth
response   3   4   5   6   7   8   9
       3   0   0   0   0   0   0   0
       4   0   0   0   0   0   0   0
       5   0  10 121  55   0   0   0
       6   1  15 144 414 110  13   0
       7   0   0   1  48  41   7   0
       8   0   0   0   0   0   0   0
       9   0   0   0   0   0   0   0
```

Key to understand the confusion matrix

- 5 was predicted as 6 144 times. (response, truth) = (6,5)

Changing Hyperparameters

The `Learner` contains information about all parameters that can be configured, including data type, constraints, defaults, etc. The hyperparameters can be changed either during construction of later through an active binding.

```r
as.data.table(learner_1$param_set)[, .(id, class, lower, upper, nlevels)]
```

```
                id    class lower upper nlevels
 1:             cp ParamDbl     0     1     Inf
 2:     keep_model ParamLgl    NA    NA       2
 3:     maxcompete ParamInt     0   Inf     Inf
```

```
 4:        maxdepth ParamInt     1    30      30
 5:    maxsurrogate ParamInt     0   Inf     Inf
 6:       minbucket ParamInt     1   Inf     Inf
 7:        minsplit ParamInt     1   Inf     Inf
 8: surrogatestyle ParamInt     0     1       2
 9:    usesurrogate ParamInt     0     2       3
10:            xval ParamInt     0   Inf     Inf
```

```r
learner_2 = lrn("classif.rpart", predict_type = "prob", minsplit = 50)
learner_2$param_set$values$minsplit = 50
```

Resampling

`Resampling` repeats the train-predict-score loop and collects all results in a nice 'data.table::data.table()'.

```r
cv10 = rsmp("cv", folds = 10)
rr = resample(wine_tsk, learner_1, cv10)
print(rr)
```

```
<ResampleResult> with 10 resampling iterations
   task_id    learner_id resampling_id iteration warnings errors
 wine_data classif.rpart            cv         1        0      0
 wine_data classif.rpart            cv         2        0      0
 wine_data classif.rpart            cv         3        0      0
 wine_data classif.rpart            cv         4        0      0
 wine_data classif.rpart            cv         5        0      0
 wine_data classif.rpart            cv         6        0      0
 wine_data classif.rpart            cv         7        0      0
 wine_data classif.rpart            cv         8        0      0
 wine_data classif.rpart            cv         9        0      0
 wine_data classif.rpart            cv        10        0      0
```

```r
rr$score(msrs(c("classif.acc", "classif.ce")))[, .(iteration, task_id, learner_id, resampling_id, classi
```

```
    iteration   task_id    learner_id resampling_id classif.ce
 1:         1 wine_data classif.rpart            cv  0.4714286
 2:         2 wine_data classif.rpart            cv  0.5000000
 3:         3 wine_data classif.rpart            cv  0.4795918
 4:         4 wine_data classif.rpart            cv  0.4387755
 5:         5 wine_data classif.rpart            cv  0.5204082
 6:         6 wine_data classif.rpart            cv  0.4836735
 7:         7 wine_data classif.rpart            cv  0.4061224
 8:         8 wine_data classif.rpart            cv  0.4571429
 9:         9 wine_data classif.rpart            cv  0.4580777
10:        10 wine_data classif.rpart            cv  0.4723926
```

```r
# gets all predictions nicely concatenated in a table
prediction = rr$prediction()
as.data.table(prediction)
```

```
      row_ids truth response
   1:       2     6        5
   2:       6     6        6
   3:      17     6        5
   4:      43     6        5
   5:      52     7        6
```

```
  ---
4894:    4838     6         7
4895:    4845     6         6
4896:    4848     7         6
4897:    4865     5         5
4898:    4898     6         6
```

```
# The confusion matrix for entire prediction
cm = prediction$confusion
print(cm)
```

```
        truth
response    3    4    5    6    7    8    9
       3    0    0    0    0    0    0    0
       4    0    0    0    0    0    0    0
       5    5   85  787  439   41    1    0
       6   14   74  661 1640  664  129    3
       7    1    4    9  119  175   45    2
       8    0    0    0    0    0    0    0
       9    0    0    0    0    0    0    0
```

Populating the learner dictionary

`mlr3learners` ships out with a dozen different popular `Learners`. They can be listed from the dictionary. If more were desired, an extension package, `mlr3extralearners`, could be installed from GitHub. Importantly, after loading `mlr3extralearners`, the dictionary increases in size.

```
head(as.data.table(mlr_learners)[, c("key", "packages")])
```

```
                  key                        packages
1: classif.AdaBoostM1 mlr3,mlr3extralearners,RWeka
2:        classif.C50   mlr3,mlr3extralearners,C50
3:        classif.IBk mlr3,mlr3extralearners,RWeka
4:        classif.J48 mlr3,mlr3extralearners,RWeka
5:       classif.JRip mlr3,mlr3extralearners,RWeka
6:        classif.LMT mlr3,mlr3extralearners,RWeka
```

```
library(mlr3extralearners)
print(as.data.table(mlr_learners)[, c("key", "packages")])
```

```
                  key                                                    packages
  1: classif.AdaBoostM1                            mlr3,mlr3extralearners,RWeka
  2:        classif.C50                              mlr3,mlr3extralearners,C50
  3:        classif.IBk                            mlr3,mlr3extralearners,RWeka
  4:        classif.J48                            mlr3,mlr3extralearners,RWeka
  5:       classif.JRip                            mlr3,mlr3extralearners,RWeka
 ---
172:        surv.ranger                 mlr3,mlr3proba,mlr3extralearners,ranger
173:         surv.rfsrc mlr3,mlr3proba,mlr3extralearners,randomForestSRC,pracma
174:          surv.rpart                      mlr3,mlr3proba,rpart,distr6,survival
175:            surv.svm            mlr3,mlr3proba,mlr3extralearners,survivalsvm
176:        surv.xgboost                mlr3,mlr3proba,mlr3extralearners,xgboost
```

Benchmarking multiple learners

The `benchmark` function can conveniently compare ‘r ref(“Learner”, “Learners”) on the same dataset(s).

```
learners = list(learner_1, learner_2, lrn("classif.randomForest"))
grid = benchmark_grid(wine_tsk, learners, cv10)
bmr = benchmark(grid)
print(bmr)
```

```
<BenchmarkResult> of 30 rows with 3 resampling runs
 nr   task_id           learner_id resampling_id iters warnings errors
  1 wine_data        classif.rpart            cv    10        0      0
  2 wine_data        classif.rpart            cv    10        0      0
  3 wine_data classif.randomForest            cv    10        0      0
```

```
print(bmr$aggregate(measures = msrs(c("classif.acc", "classif.ce"))))
```

```
   nr   task_id           learner_id resampling_id iters classif.acc classif.ce
1:  1 wine_data        classif.rpart            cv    10   0.5320521  0.4679479
2:  2 wine_data        classif.rpart            cv    10   0.5320521  0.4679479
3:  3 wine_data classif.randomForest            cv    10   0.7051863  0.2948137
Hidden columns: resample_result
```