Uczenie maszynowe z pakietem mlr

Mateusz Staniak
30.01.2018

Materiały

- Szybkie wprowadzenie: **winietka**
- Bogata dokumentacja: **mlr**
- Poprzednie warsztaty STWUR: **Github/STWUR**, **Github/STWUR**
- Warsztaty z konferencji Why R: **Github/pzawistowski**
- Inne...

Dlaczego mlr?

```
library(randomForest)
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
```

Work flow

- Przygotowanie danych (preprocessing)
- Zadanie (task)
- Wybór modelu (benchmark)
- Strojenie parametrów (tuning)
- Ocena modelu

Dane

```
library(mlr)
library(dplyr)
library(ggplot2)

mieszkania  <- read.csv("~/Dokumenty/Projekty/eRementarz4/mieszkania_wroclaw_ceny.csv")

head(mieszkania)
```

```
  n_pokoj metraz cena_m2  rok pietro pietro_maks   dzielnica
1       3  89.00    5270 2007      2           2      Krzyki
2       4 163.00    6687 2002      2           2   Psie Pole
3       3  52.00    6731 1930      1           2 Srodmiescie
4       4  95.03    5525 2016      1           2      Krzyki
5       4  88.00    5216 1930      3           4 Srodmiescie
6       2  50.00    5600 1915      3           4      Krzyki
```

Przygotowanie danych

Typowe zadania:

- standaryzacja danych - `normalizeFeatures`
- łączenie mało licznych poziomów zmiennych jakościowych - `mergeSmallFactorLevels`
- wybranie części obserewacji - `subsetTask`
- imputacja danych brakujących - `impute`
- i inne...

Zadania (task)

- Obsługiwane klasy problemów

```
makeClassifTask()
makeRegrTask()
makeClusterTask()
makeCostSensTask()
makeMultilabelTask()
makeSurvTask()
```

Nasz problem

```
m2_task <- makeRegrTask(id = "mieszkanie",
                        data = mieszkania,
                        target = "cena_m2")
```

- Alternatywnie

```
m2_task_ndz <- makeRegrTask(id = "mieszkanie_ndz",
                            data = select(mieszkania, -dzielnica),
                            target = "cena_m2")
```

Uwaga

- `fixup.data = "warn"`: czyszczenie danych (aktualnie tylko usuwanie pustych poziomów)
- `check.data =` TRUE: sprawdzanie poprawności danych (aktualnie: NA i puste poziomy zmiennej odpowiedzi)

Metody uczenia (learner)

- Ogromna liczba dostępnych metod

```
listLearners(obj = "regr")[1:6, c(1, 3:4)]
```

```
              class  short.name package
1     regr.cforest     cforest   party
2       regr.ctree       ctree   party
3    regr.cvglmnet    cvglmnet  glmnet
4 regr.featureless featureless     mlr
5     regr.gausspr     gausspr kernlab
6         regr.glm         glm   stats
```

- Jak radzi sobie zwykła regresja liniowa?

```
reg_lm <- makeLearner("regr.lm")
```

- A jak inne popularne metody?

```
reg_rf <- makeLearner("regr.randomForest")
reg_nnet <- makeLearner("regr.nnet")
```

- Inaczej:

```
lrns <- makeLearners(c("lm", "randomForest", "nnet"),
                     type = "regr")
```

- takie wywołania tworzą obiekty typu `Learner`
- metody są zaimplementowane w odpowiednich pakietach - `mlr` jest nakładką

- różne metody - różne wsparcie dla brakujących wartości, wag itd

- Uwaga: w ten sposób wszystkie hiperparametry mają ustawione wartości domyślne

Informacje o metodzie

```
getLearnerProperties(reg_rf)
```

```
[1] "numerics" "factors"  "ordered"  "se"       "oobpreds" "featimp"
```

```
getLearnerParamSet(reg_rf)
```

```
                    Type  len       Def                 Constr Req
ntree            integer    -       500               1 to Inf   -
se.ntree         integer    -       100               1 to Inf   Y
se.method       discrete    - jackknife bootstrap,jackknife,sd   Y
se.boot          integer    -        50               1 to Inf   -
mtry             integer    -         -               1 to Inf   -
replace          logical    -      TRUE                      -   -
strata           untyped    -         -                      -   -
sampsize    integervector <NA>         -               1 to Inf   -
nodesize         integer    -         5               1 to Inf   -
maxnodes         integer    -         -               1 to Inf   -
importance       logical    -     FALSE                      -   -
localImp         logical    -     FALSE                      -   -
nPerm            integer    -         1            -Inf to Inf   -
proximity        logical    -     FALSE                      -   -
oob.prox         logical    -         -                      -   Y
do.trace         logical    -     FALSE                      -   -
keep.forest      logical    -      TRUE                      -   -
```

```
keep.inbag        logical     -     FALSE                       -    -
            Tunable Trafo
ntree          TRUE     -
se.ntree       TRUE     -
se.method      TRUE     -
se.boot        TRUE     -
mtry           TRUE     -
replace        TRUE     -
strata        FALSE     -
sampsize       TRUE     -
nodesize       TRUE     -
maxnodes       TRUE     -
importance     TRUE     -
localImp       TRUE     -
nPerm          TRUE     -
proximity     FALSE     -
oob.prox      FALSE     -
do.trace      FALSE     -
keep.forest   FALSE     -
keep.inbag    FALSE     -
```

Ustawianie hiperparametrów

- przy tworzeniu learnera:

```
reg_rf2 <- makeLearner("regr.randomForest",
                       par.vals = list(ntree = 1000))
```

- po utworzeniu learnera:

```
reg_rf2 <- setHyperPars(reg_rf, ntree = 1000)
```

```
getHyperPars(reg_rf2)
```

```
$ntree
[1] 1000
```

Porównywanie modeli

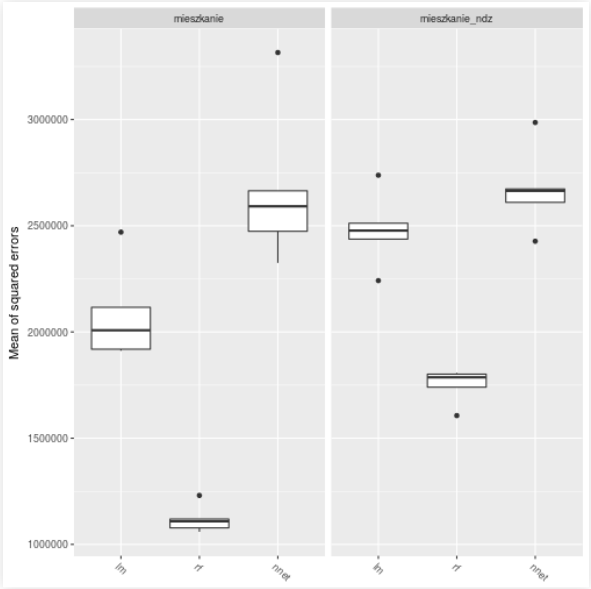
```
porownanie <- benchmark(learners = list(reg_lm, reg_rf, reg_nnet),
                        tasks = list(m2_task, m2_task_ndz),
                        resampling = cv5)
save(porownanie, file = "porownanie.rda")
```

```
load("porownanie.rda")
porownanie
```

```
         task.id        learner.id mse.test.mean
1     mieszkanie           regr.lm       2084932
2     mieszkanie regr.randomForest       1119241
3     mieszkanie          regr.nnet       2674150
4 mieszkanie_ndz           regr.lm       2481207
5 mieszkanie_ndz regr.randomForest       1748378
6 mieszkanie_ndz          regr.nnet       2672237
```

Wyniki porównania

```
# getBMRAggrPerformances(porownanie)
# getBMRPerformances(porownanie)
plotBMRBoxplots(porownanie)
```



Różne kryteria

```
listMeasures(obj = "regr")
```

```
 [1] "rsq"          "rae"          "rmsle"        "mse"          "rrse"
 [6] "medse"        "mae"          "timeboth"     "timepredict" "medae"
[11] "featperc"     "mape"         "rmse"         "kendalltau"   "sse"
[16] "arsq"         "expvar"       "msle"         "sae"          "timetrain"
[21] "spearmanrho"
```

Wytrenowanie pojedynczego modelu

- podstawowe wywołanie:

```
m2_rf <- train(reg_rf2, m2_task)
```

- uwaga: można samodzielnie zdefiniować zbiór uczący

```
uczacy <- sample(1:nrow(mieszkania), floor(0.7*nrow(mieszkania)))
testowy <- setdiff(1:nrow(mieszkania), uczacy)
```

```
m2_rf_czesc <- train(reg_rf2, m2_task, subset = uczacy)
```

- przewidywane wartości:

```
pred <- predict(m2_rf, task = m2_task)
head(getPredictionResponse(pred))
```

```
[1] 5353.419 6598.072 6426.211 5378.307 5241.464 5547.251
```

```
pred2 <- predict(m2_rf_czesc, newdata = mieszkania[testowy, ])
head(getPredictionResponse(pred2))
```

```
[1] 5383.544 6566.714 5296.223 4756.715 7551.108 5929.577
```

Strojenie parametrów

```
all_params <- makeParamSet(
  makeDiscreteParam("mtry", values = 2:5),
  makeDiscreteParam("nodesize", values = seq(5, 45, by = 10))
)
```

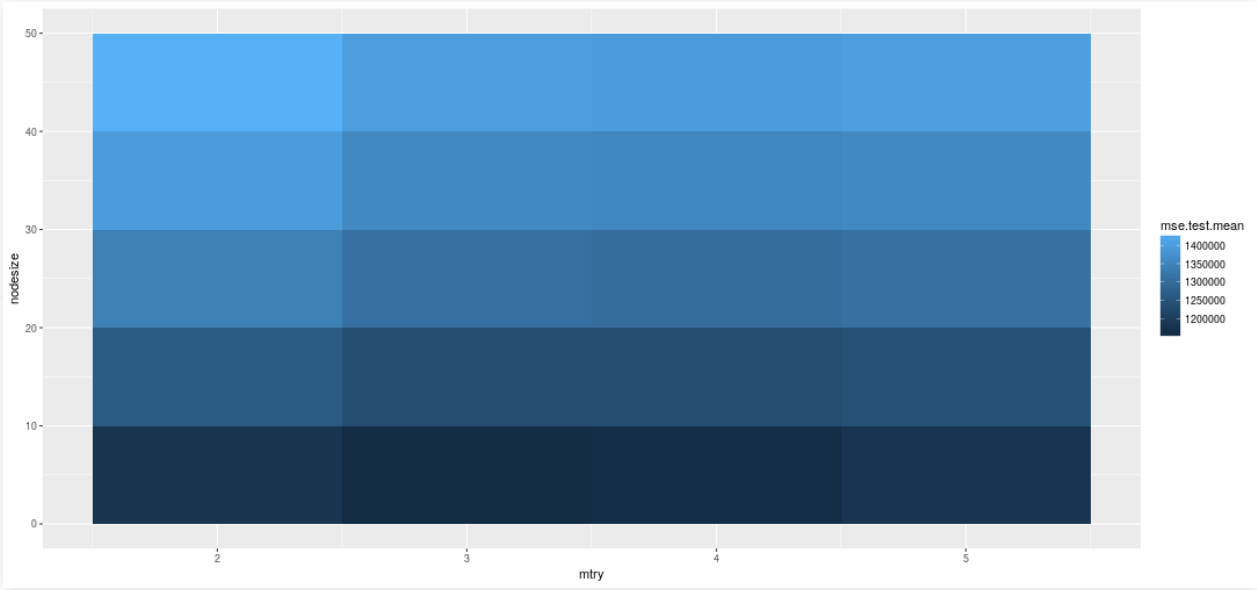
```
m2_params <- tuneParams(reg_rf2, task = m2_task,
                        resampling = cv3,
                        par.set = all_params,
                        control =  makeTuneControlGrid())
```

Wynik

```
m2_params
```

```
Tune result:
Op. pars: mtry=3; nodesize=5
mse.test.mean=1.15e+06
```

```
par_data <- generateHyperParsEffectData(m2_params)
plotHyperParsEffect(par_data, x = "mtry", y = "nodesize", z = "mse.test.mean", plot.type =
"heatmap")
```




```
reg_rf2 <- setHyperPars(reg_rf2, mtry = 3)
```

```
m2_rf2 <- train(reg_rf2, m2_task)
```

Inne kontrolki

```
makeTuneControlCMAES()
makeTuneControlIrace()
makeTuneControlRandom()
```

Cena mieszkania

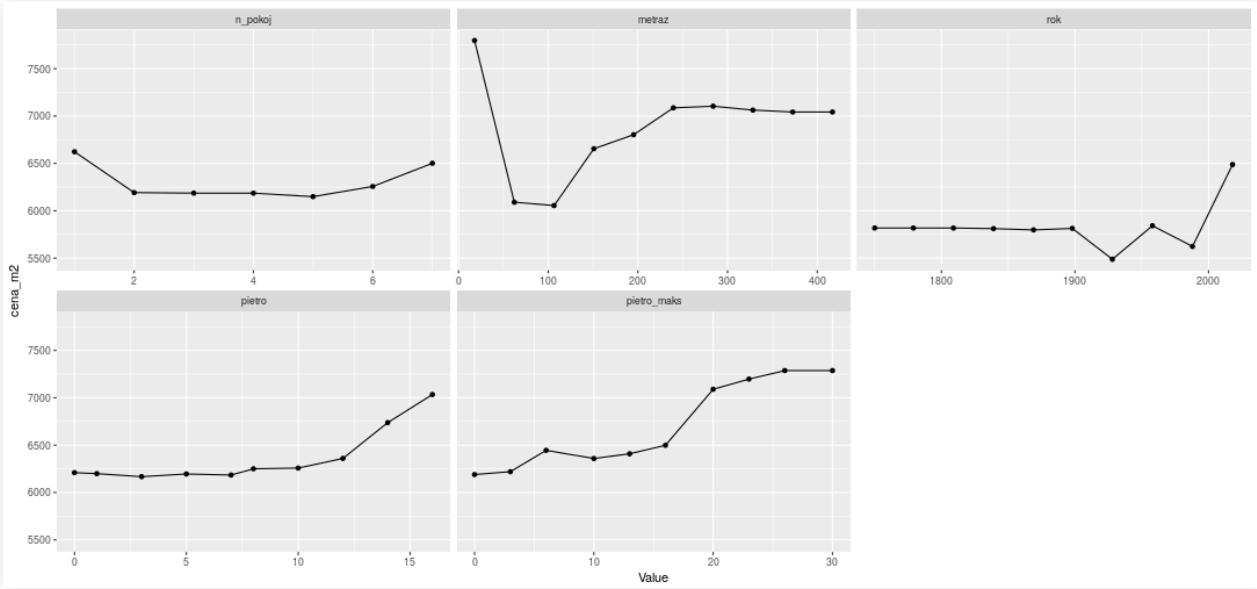
```
moje <- data.frame(n_pokoj = 3L,
                   metraz = 60.00,
                   rok = 2010L,
                   pietro = 3L,
                   pietro_maks = 5L,
                   dzielnica = "Srodmiescie")
moje$dzielnica <- factor(moje$dzielnica,
                         levels = levels(mieszkania$dzielnica))
predict(m2_rf2, newdata = moje)
```

```
Prediction: 1 observations
predict.type: response
threshold:
time: 0.09
  response
1 7503.985
```

Wizualizacja modelu

```
pdp <- generatePartialDependenceData(m2_rf2,
                                     m2_task,
                                     features = colnames(mieszkania)[-c(3, 7)])
```

```
plotPartialDependence(pdp)
```



Podziękowanie

Zachęta

Projekt

Konkurs

- Nagroda: wejściówka na konferencję Why R? 2018
- Co należy zrobić: podać dalej post zapowiadający kolejne spotkanie STWUR-a!

Why R? 2018

- 2-4.07 we Wrocławiu
- Organizowany przez STWUR przy współpracy ze społecznościami R-owymi z innych części Polski
- Międzynarodowe wydarzenie (goście m.in. z Niemiec i Czech)
- Why R? 2017: ponad 200 uczestników, warsztaty, hackathon i wiele wykładów z różnych dziedzin
- Nastawiony na machine learning

STWUR

- [https://www.facebook.com/stwur]
- [https://www.meetup.com/pl-PL/Wroclaw-R-Users-Group]
- [https://stwur.github.io/STWUR/]

Po warsztatach spotykamy się w Cybermachinie!

Podsumowanie

- Tworzenie zadania: `makeRegrTask`, `makeClassifTask` itd
- Metoda uczenia: `makeLearner`, `makeLearners`
- Porównanie kilku modeli: `benchmark`
- Ustawianie hiperparametrów: `setHyperPars`
- Wytrenowanie pojedynczego modelu: `train`
- Strojenie parametrów: `tune` - uzupełnić

Pytania, zadania, problemy

1. Porównaj działanie swojego ulubionego modelu np. z przedstawionymi modelami. Wybierz odpowiednie kryterium.

2. Czy przekształcenie zmiennej objaśnianej może poprawić predykcje modelu? (Skośność? Obserwacje odstające?)

3. Czy umiesz zaproponować nowe zmienne, które poprawią dokładność wybranego modelu?

4. Jakie są parametry (hiperparametry) w najlepiej sprawdzającym się modelu? Spróbuj wybrać optymalne wartości dla nich.

5. Jaką cenę metra kwadratowego przewiduje wytrenowany przez Ciebie model dla mieszkania Twoich marzeń?

Bonus

1. Jak wygląda (brzegowa) zależność ceny m^2 od wieku mieszkania w wybranym przez Ciebie modelu?

2. Sensownym pytaniem jest, które mieszkania zaklasyfikujemy jako osiągalne cenowo dla przeciętnego Wrocławianina. Przyjmując za próg 300 tys. zł, dodaj zmienną oznaczającą, czy mieszkanie jest dość tanie, stwórz zadanie klasyfikacji dla tej zmiennej i wytrenuj wybrany model.