Kombajn do uczenia maszynowego: MLR w praktyce

dr inż. Paweł Zawistowski

Instytut Informatyki, WEiTI, PW / Adform

Źródło obrazu: https://cdn.pixabay.com/photo/2013/06/16/13/41/cotton-harvester-139651_1280.jpg

Plan warsztatu

- Do czego służy i co potrafi pakiet MLR?
- Przygotowanie danych/sformułowanie zadania.
- Uczenie modelu, strojenie parametrów.
- Wizualizacje i rozszerzanie MLR
- Mini kaggle.

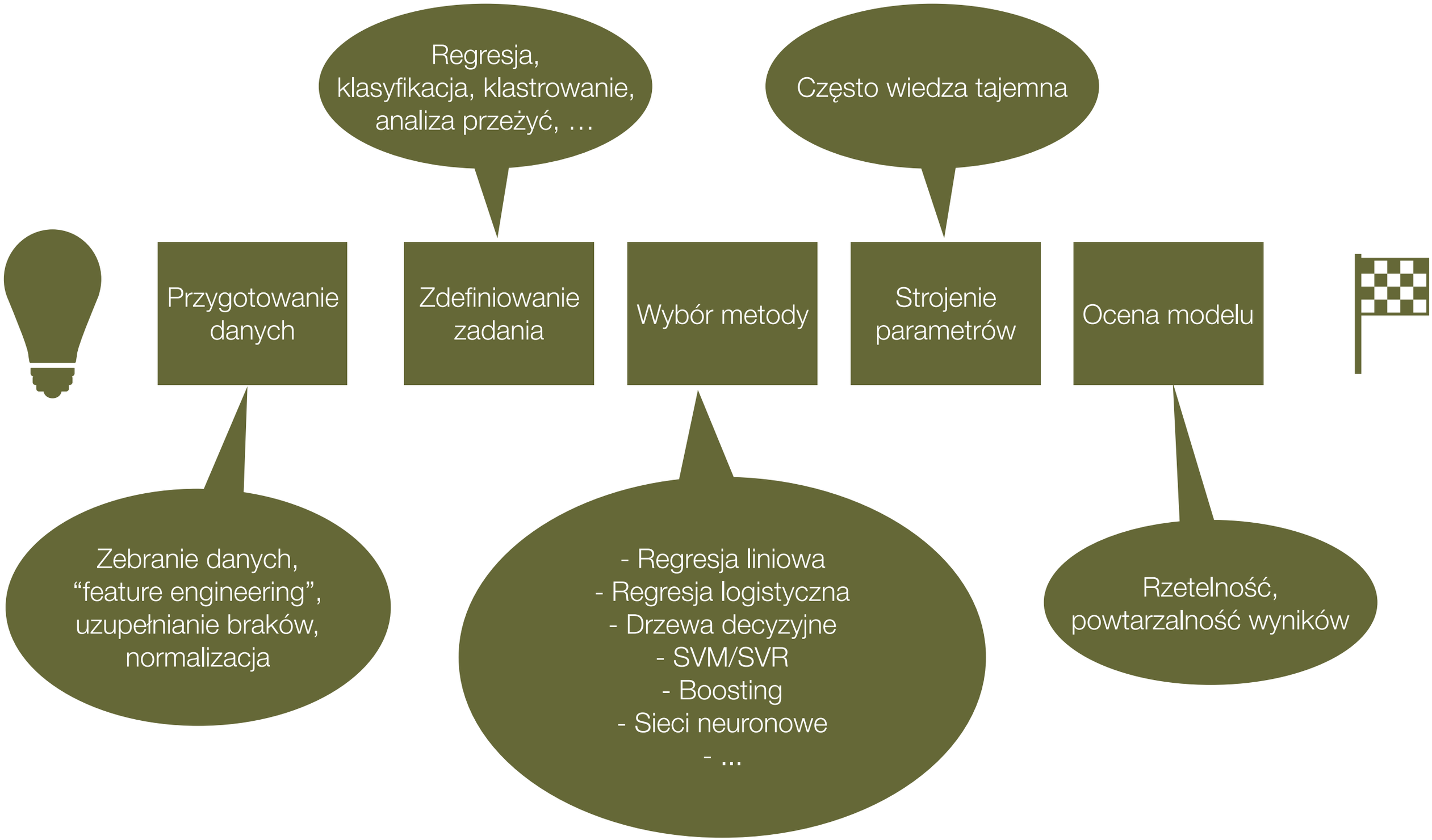
- W trakcie warsztatu **przerwa** 15:30 - 16:00.

Do czego służy i co potrafi pakiet MLR?

Źródło obrazu: https://cdn.pixabay.com/photo/2013/06/16/13/41/cotton-harvester-139651_1280.jpg

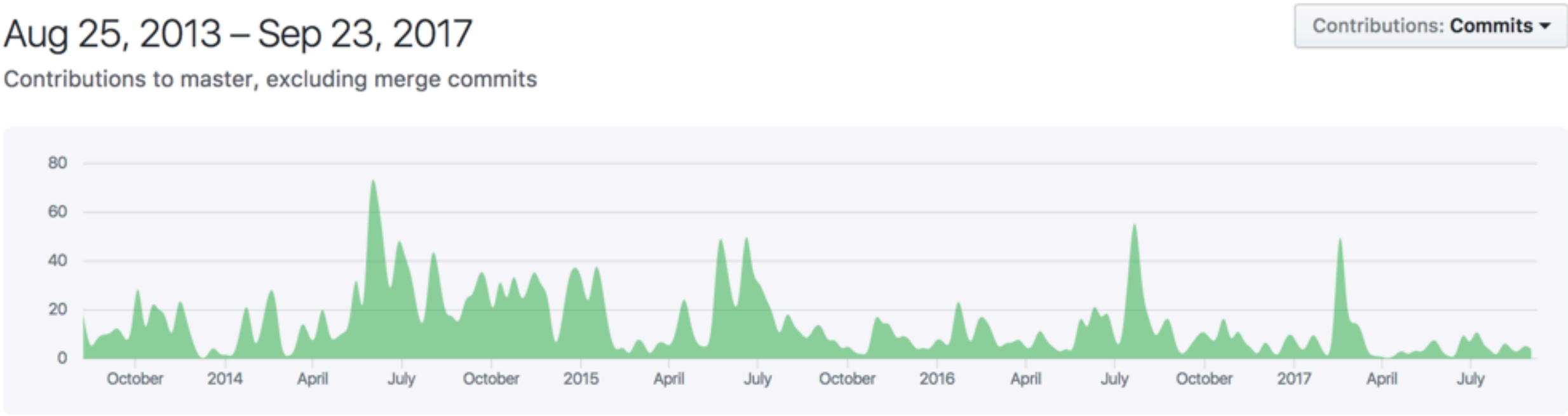
MLR **ułatwia modelowanie** ujednolicając sposób korzystania z wielu innych pakietów.

Tworzenie modelu z lotu ptaka



mlR : Machine Learning in R

- Pierwszy commit sierpień 2013.
- > 3800 commits
- 47 osób ma swoje kontrybucje



Classification and Regression Training

- Na GitHub od 2014, istnieje znacznie dłużej v2.* w 2007.
- > 1500 commits
- 55 osób ma swoje kontrybucje

Co potrafi MLR?

Motywujący przykład

```
library(mlr); library(                                    )

parallelMap::parallelStartMu                    (level = 'mlr.resamp
results = makeLearner("classif.knn") %>%
          makeTuneWrapper(resampling = cv3,
                          measures = acc,
                          par.set = makeParamSet(makeInteg                    er=1, upper=20)),
                          control = makeTuneControlGrid()) %>%
                          resampling = cv3,
                          measures = list(acc, setAggregation(                    tes
para
calc                                                    sults$pred)
```

Wybór metody

Walidacja krzyżowa

Obliczenia Równoległe ⌣ parametrów

Zadanie klasyfikacji

Walidacja krzyżowa

MLR - co potrafi?

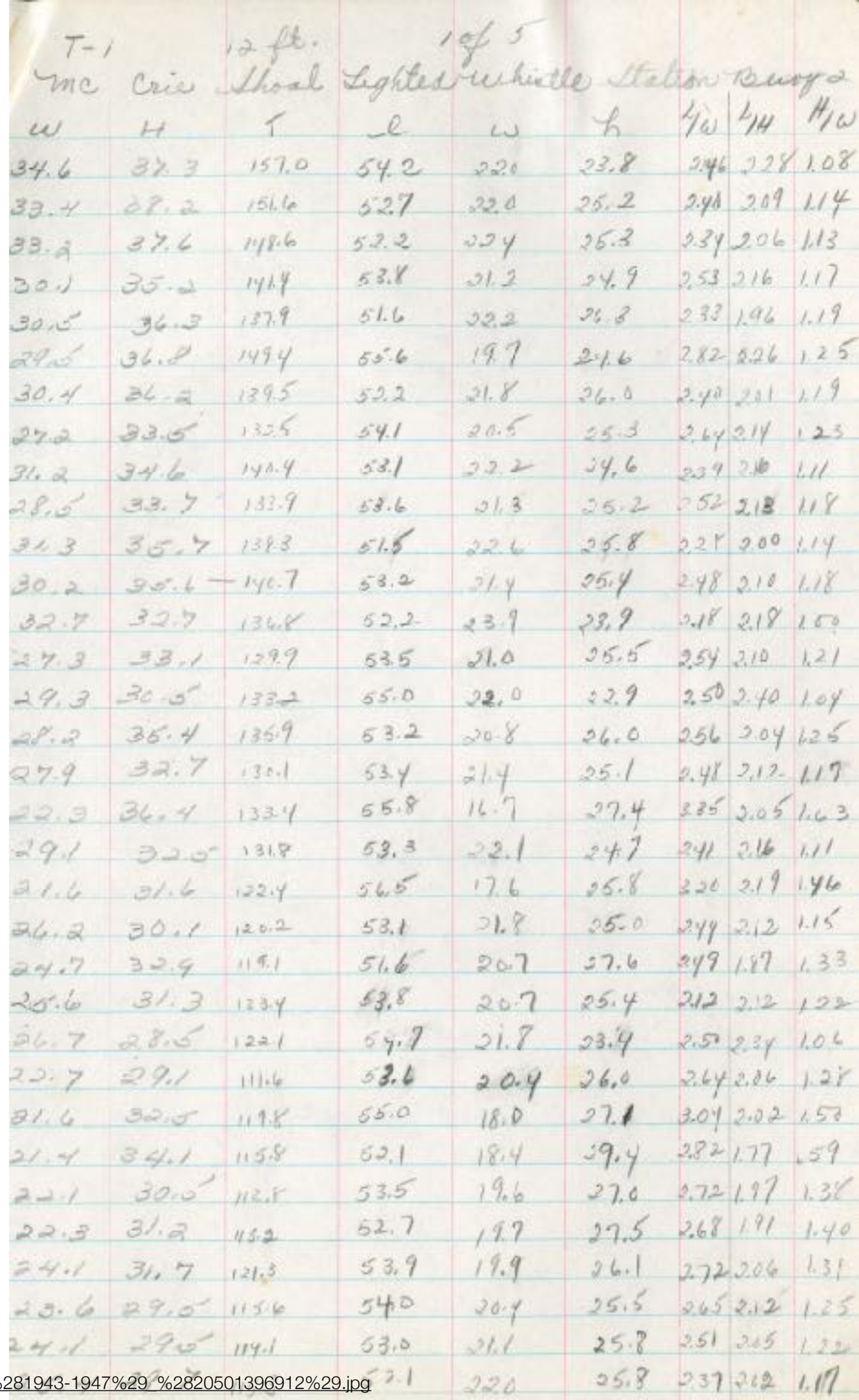
- Do rozwiązywania zadań: klasyfikacji, regresji, klastrowania i analizy przeżycia.
- Zwięzły interfejs wykorzystujący klasy S3.
- Opis zadań i modeli za pomocą właściwości.
- Możliwość kodowania typów danych i ich ograniczeń.
- Bootstrapping, walidacja krzyżowa, próbkowanie - również zagnieżdżone.
- Wizualizacje: krzywe ROC, predykcje.
- Możliwość tworzenia “benchmarków”.
- Strojenie parametrów przy pomocy różnych strategii optymalizacyjnych - np. F-racing.
- Selekcja zmiennych.
- Wsparcie dla ważenia przypadków/nierównomiernego rozkładu klas.
- Wbudowane wsparcie dla zrównoleglania.
- …

Przygotowanie do warsztatu

- Materiały z warsztatu znajdują się w repozytorium:

```
git clone \
https://github.com/pzawistowski/mlr-workshop.git
```

Przygotowanie danych



T-1 1 of 5

Mc W	Cric H	12 ft. Shoal T	Lighted l	Whistle w	Station h	Buoy 4/w	Buoy 4/H	Buoy H/w
34.6	37.3	157.0	54.2	22.0	23.8	2.46	2.28	1.08
33.4	28.2	151.6	52.7	22.0	25.2	2.48	2.09	1.14
33.9	37.6	148.6	52.2	22.4	25.3	2.34	2.06	1.13
30.1	35.2	141.4	53.8	21.2	24.9	2.53	2.16	1.17
30.5	36.3	137.9	51.6	22.2	26.8	2.33	1.96	1.19
29.5	36.8	149.4	55.6	19.7	24.6	2.82	2.26	1.25
30.4	26.2	139.5	52.2	21.8	26.0	2.40	2.21	1.19
27.2	33.5	133.5	54.1	20.5	25.3	2.64	2.14	1.23
31.2	34.6	141.4	53.1	22.2	24.6	2.39	2.16	1.11
28.5	33.7	133.9	53.6	21.3	25.2	2.52	2.13	1.18
31.3	35.7	134.3	51.6	22.6	25.8	2.28	2.00	1.14
30.2	35.6	140.7	53.2	21.4	25.4	2.48	2.10	1.18
32.7	32.7	136.8	52.2	23.9	23.9	2.18	2.18	1.00
27.3	33.1	129.9	53.5	21.0	25.5	2.54	2.10	1.21
29.3	30.5	133.2	55.0	22.0	22.9	2.50	2.40	1.04
28.2	35.4	135.9	53.2	20.8	26.0	2.56	2.04	1.25
27.9	32.7	130.1	53.4	21.4	25.1	2.48	2.12	1.17
22.3	36.4	133.4	55.8	16.7	27.4	3.35	2.05	1.63
29.1	32.0	131.8	53.3	22.1	24.7	2.41	2.16	1.11
21.6	31.6	122.4	56.5	17.6	25.8	3.20	2.19	1.46
26.2	30.1	120.2	53.1	21.8	25.0	2.44	2.12	1.15
24.7	32.9	119.1	51.6	20.7	27.6	2.49	1.87	1.33
25.6	31.3	123.4	53.8	20.7	25.4	2.12	2.12	1.22
26.7	28.5	122.1	54.7	21.8	23.4	2.50	2.34	1.06
22.7	29.1	111.6	53.6	20.4	26.0	2.64	2.86	1.28
21.6	30.5	119.8	55.0	18.0	27.1	3.04	2.02	1.58
21.4	34.1	115.8	52.1	18.4	29.4	2.82	1.77	1.59
22.1	30.0	112.5	53.5	19.6	27.0	2.72	1.97	1.38
22.3	31.2	115.2	52.7	19.7	27.5	2.68	1.91	1.40
24.1	31.7	121.3	53.9	19.9	26.1	2.72	2.06	1.31
23.6	29.5	115.6	54.0	20.4	25.5	2.65	2.12	1.25
24.1	29.5	114.1	53.0	21.1	25.8	2.51	2.05	1.22
			52.1	22.0	25.8	2.37	2.02	1.17

Źródło obrazu: https://upload.wikimedia.org/wikipedia/commons/8/8d/Buoy_fouling_survey%2C_research_data_%2815%29_%281943-1947%29_%2820501396912%29.jpg

Zadania wspierane przez MLR

- Konkretne zadanie, wraz z jego danymi, reprezentuje obiekt klasy Task.
- Tworzy się go przy pomocy jednej z funkcji:
  - makeClassifTask, makeMultilabelTask - klasyfikacja binarna/ wieloklasowa
  - makeClusterTask - klastrowanie,
  - makeRegrTask - regresja,
  - makeSurvTask - analiza przeżycia,
  - makeCostSensTask - klasyfikacja z różnymi kosztami pomyłek.

Tworzenie zadania regresji

```
makeRegrTask(id =“foo”, data, target, weights = NULL,
  blocking = NULL, fixup.data = "warn", check.data = TRUE)
```

```
fixup.data - usuwanie pustych poziomów ze zmiennych typu “factor”
check.data - sprawdzanie zmiennej celu - obecnie pod kątem brakujących wartości i “pustych” poziomów
```

Tworzenie zadania klasyfikacji z różnymi kosztami pomyłek

```
data(iris, package = "datasets")
cost <- matrix(runif(150 * 3, 0, 2000), 150) * (1 - diag(3))[iris$Species, ]
iris$Species <- NULL
costiris.task <- makeCostSensTask("cost-sensitive iris-example", data = iris, cost = cost)
```

Macierz kosztów pomyłek dla poszczególnych wierszy

	V1	V2	V3
1	0.00000	1707.1007420	1919.29621
2	0.00000	990.6031210	364.47402
3	0.00000	224.3249393	1214.94884
4	0.00000	1500.1604022	895.25275
5	0.00000	172.1161953	1423.55333
6	0.00000	544.0438990	1272.51073
7	0.00000	1609.5329621	768.48458
8	0.00000	1086.8785055	1920.57162
9	0.00000	1618.9780566	1874.59364

Przekształcenia danych

- capLargeValues - usuwanie/zamiana wartości odstających,
- createDummyFeatures - “one hot encoding”,
- dropFeatures - usuwanie atrybutów,
- joinClassLevels - łączenie “małych” klas w większe,
- mergeSmallFactorLevels - łączenie rzadko spotykanych wartości atrybutów,
- normalizeFeatures - normalizacja atrybutów,
- removeConstantFeatures - usuwanie jednowartościowych atrybutów,
- subsetTask - usuwanie obserwacji/atrybutów,
- selekcja tech,
- uzupełanianie brakujących danych.

01_PrzygotowanieDanych.Rmd

Zadanie 1

Uczenie modelu, strojenie parametrów

Źródło obrazu: https://upload.wikimedia.org/wikipedia/commons/e/e5/Comp._rack_%28Supernatural%29.jpg

Metody uczące

- MLR jest tylko nakładką - implementacje są w osobnych pakietach.

- Reprezentowane przez obiekty klasy Learner

- Tworzone przez metodę makeLearner

```
classif.lrn = makeLearner("classif.randomForest", predict.type = "prob",
fix.factors.prediction = TRUE)

regr.lrn = makeLearner("regr.gbm", par.vals = list(n.trees = 500, interaction.depth
= 3))

surv.lrn = makeLearner("surv.coxph", id = "cph")

cluster.lrn = makeLearner("cluster.kmeans", centers = 5)

multilabel.lrn = makeLearner(“multilabel.rFerns”)
```

Właściwości metod

- Rodzaj zadania do jakiego przystosowana jest dana metoda.
- Obsługiwane typy danych wejściowych.
- Wsparcie dla brakujących danych i wag.

Wspierane metody

- Pomocna funkcja: listLearners

Class / Short Name / Name	Packages	Num.	Fac.	Ord.	NAs	Weights	Props	Note
classif.ada *ada* ada Boosting	ada rpart	X	X				prob twoclass	`xval` has been set to `0` by default for speed.
classif.bartMachine *bartmachine* Bayesian Additive Regression Trees	bartMachine	X	X		X		prob twoclass	`use_missing_data` has been set to `TRUE` by default to allow missing data support.
classif.binomial *binomial* Binomial Regression	stats	X	X			X	prob twoclass	Delegates to `glm` with freely choosable binomial link function via learner parameter `link`. We set 'model' to FALSE by default to save memory.

Ustawianie parametrów metody

- Podczas tworzenia obiektu Learner:

```
makeLearner("regr.gbm", par.vals = list(n.trees = 500, interaction.depth = 3))
makeLearner("regr.gbm", n.trees = 500, interaction.depth = 3)
```

- Na utworzonym wcześniej obiekcie

```
lrn = makeLearner(“regr.gbm”)
lrn = setHyperPars(lrn, n.trees = 500, interaction.depth = 3)
```

- W świecie tidyverse

```
lrn = makeLearner(“regr.gbm”) %>%
        setHyperPars(n.trees = 500, interaction.depth = 3)
```

Strojenie parametrów

- Ustalanie przestrzeni przeszukiwania

```
par.set = makeParamSet(
    makeDiscreteParam("S", values = c("linear","ranked"))
    , makeDiscreteVectorParam(id = "C", values = c('dp','radial','exp'),)
    , makeNumericParam("scal.lambda", lower = 0.0, upper = 1.0)
    , makeFunctionParam(“weights”,
        default = function(y, w.par){ 0.01+dnorm(y,sd=abs(w.par))})
    , makeIntegerParam("optim.maxit", lower=10, upper = 100000)
    , makeLogicalParam("verbose", default = FALSE, tunable = FALSE)
)
```

- Strategie optymalizacji - obiekty TuneControl

Strategie optymalizacji

- `makeTuneControlDesign` – podajemy ramkę danych z parametrami
- `makeTuneControlGrid` – przeszukiwanie po hipersiatce
- `makeTuneControlRandom` – przeszukiwanie po losowe
- `makeTuneControlCMAES` – zastosowanie algorytmu CMA-ES
- `makeTuneControlGenSA` – symulowane wyżarzanie

- `makeTuneControlIrace` – metoda “iterated F-Racing”

Uczenie modelu

- “Zwyczajne”:

```
model <-train(lrn, task)
```

- Z dodatkowym próbkowaniem:

```
results =  lrn %>%  resample(task = tsk,
                   resampling = cv3,
                   measures = list(acc, setAggregation(acc, test.sd)))
```

- W ramach benchmarku:

```
benchmarkResults <- benchmark(list(lrn1, lrn2), list(task1, task2),
                   resampling = cv3,
                   measures = list(acc, setAggregation(acc, test.sd)))
```

Próbkowanie

- Tworzenie przy pomocy metody makeResampleDesc
- Dostępne strategie:
  - CV - walidacja krzyżowa,
  - RepCV - powtarzana walidacja krzyżowa,
  - LOO - “leave-one-out”,
  - Bootstrap - bootstrapping,
  - Subsample - losowo wybrany zbiór testowy,
  - Holdout - ustalony zbiór testowy.

Próbkowanie c.d.

- Są też “gotowe” strategie:
  - hout (testowanie na 1/3 danych),
  - cv2,
  - cv3 ,
  - cv5,
  - cv10

“Gotowych” miar błędu jest wiele

- featperc
- timetrain
- timepredict
- timeboth
- sse
- mse
- rmse
- medse
- sae
- mae
- medae
- rsq
- expvar
- arsq
- rrse
- rae
- mape
- msle
- rmsle
- kendalltau
- spearmanrho
- mmce
- acc
- ber
- multiclass.aunp
- multiclass.au1u
- multiclass.au1p
- multiclass.brier
- logloss
- ssr
- qsr
- lsr
- kappa
- wkappa
- auc
- brier
- brier.scaled
- bac
- tp
- tn
- fp
- fn
- tpr
- tnr
- fpr
- fnr
- ppv
- npv
- fdr
- mcc
- f1
- gmean
- gpr
- multilabel.hamloss
- multilabel.subset01
- multilabel.f1
- multilabel.acc
- multilabel.ppv
- multilabel.tpr
- cindex
- meancosts
- mcp
- db
- dunn
- G1
- G2
- silhouette

Można również tworzyć własne

```
err = function(task, model, pred, extra.args)
  mean(abs(pred$data$response - pred$data$truth))

makeMeasure(id = “my.err”
, minimize = TRUE
, properties = c("regr", “response")
, fun = err)
```

Obliczenia równoległe

- Proces uczenia można przyspieszyć poprzez zrównoleglenie.
- MLR korzysta do tego z pakietu parallelMap:

  parallelMap::parallelStartMulticore(level = 'mlr.resample')

- Wspierane poziomy:
  - mlr.resample - instancje próbkowania równolegle,
  - mlr.benchmark - pojedyncze eksperymenty równolegle,
  - mlr.tuneParams - obliczenia dla pojedynczych zestawów parametrów równolegle (zależy od strategii optymalizacji),
  - mlr.selectFeatures - pojedyncze zestawy atrybutów równolegle,
  - mlr.ensemble - w przypadku modeli złożonych - np. bagging.

02_UczenieModelu.Rmd

Zadanie 2

Rozszerzanie MLR

Źródło obrazu: https://cdn.pixabay.com/photo/2013/06/16/13/41/cotton-harvester-139651_1280.jpg

Własne rozszerzenia MLR

- Możemy dodawać własne:
  - miary jakości/błędu,
  - funkcje do uzupełniania danych,
  - algorytmy selekcji cech - tzw. metody filtrujące,
  - metody uczenia.

Własna metoda ucząca

Trzeba napisać:

1. definicję obiektu dla makeLearner,
2. funkcję uczącą nasz model,
3. funkcję do predykcji na podstawie gotowego modelu,
4. opcjonalnie - rejestracja metody.

03_RozszerzanieMlr.Rmd

Zadanie 3

Diagnostyka i wizualizacja wyników

Źródło obrazu: https://cdn.pixabay.com/photo/2013/06/16/13/41/cotton-harvester-139651_1280.jpg

04_Wizualizacje.Rmd

Zadanie 4

05_MiniKaggle.Rmd

Zadanie 5

Dziękuję za uczestnictwo!

Źródło obrazu: https://cdn.pixabay.com/photo/2013/06/16/13/41/cotton-harvester-139651_1280.jpg