05.Mini kaggle

Zadanie

Na podstawie atrybutów opisujących ubrania w sklepie odzieżowym należy przewidzieć ich przyszłą sprzedaż - binarny atrybut `Recommendation`.

```
## Tworzymy zadanie na podstawie pliku csv
readSalesTask <- function(filePath){
  df <- readr::read_csv(filePath,na = "NA") %>%  mutate_if(is.character, as.factor)
  makeClassifTask(data = df, target = "Recommendation")
}

salesTask <- readSalesTask('data/dress-sales.train.csv')
salesTask
```

```
## Supervised task: df
## Type: classif
## Target: Recommendation
## Observations: 450
## Features:
## numerics  factors  ordered
##        1       11        0
## Missings: TRUE
## Has weights: FALSE
## Has blocking: FALSE
## Classes: 2
##   0   1
## 264 186
## Positive class: 0
```

Przygotowanie danych

```
## Uzupełniamy brakujące dane
salesTaskImputation <- salesTask %>% impute(classes = list(factor = imputeConstant('Unknown')))

## Dodajemy zmienne wskaźnikowe
salesTaskPreprocessed <- salesTaskImputation$task %>% createDummyFeatures()
```

Uczenie

```
## Klasyfikator k-NN ze strojeniem parametrów
knnLrn <- makeLearner("classif.knn") %>%
          makeTuneWrapper(resampling = cv3,
                          measures = acc,
                          par.set = makeParamSet(makeIntegerParam("k", lower=1, upper=5)),
                          control = makeTuneControlGrid())
## Regresja logistyczna
logregLrn <- makeLearner("classif.logreg")
```

```
## Lista metod, które porównamy
learners <- list(knnLrn, logregLrn)

parallelMap::parallelStartMulticore(level = 'mlr.resample')
benchmarkResults <- benchmark(learners, list(salesTaskPreprocessed),
                  resampling = cv3,
                  measures = list(acc, setAggregation(acc, test.sd)))
parallelMap::parallelStop()

getBMRAggrPerformances(benchmarkResults, as.df = T)
```

```
##   task.id        learner.id acc.test.mean acc.test.sd
## 1      df classif.knn.tuned     0.5577778  0.02523959
## 2      df    classif.logreg     0.5777778  0.04438885
```

Finalny model

```
finalModel <-train(logregLrn, salesTaskPreprocessed)
```

```
## Mając gotowy model dokonujemy predykcji testowych i je wysyłamy mailem

salesSubmissionTask <- readSalesTask('data/dress-sales.submission.csv') %>%
                          reimpute(desc = salesTaskImputation$desc) %>%
                          createDummyFeatures()

predictions <- predict(finalModel, salesSubmissionTask)

submit(predictions)
```

Kilka pomysłów

1. Dostrojenie parametrów regresji logistycznej.
2. Zastosowanie xgboost (`regr.xgboost`).
3. Inne metody uzupełniania brakujących danych.
4. Zastosowanie normalizacji.
5. Bagging któregoś z modeli.
6. ...