xgboost

```
library(mlr3)
library(mlr3learners)
library(mlr3filters)
library(mlr3pipelines)
library(mlr3tuning)
library(paradox)
library(tidyverse)
library(ggplot2)
library(future)
library(future.apply)
```

```
future::plan("multiprocess")
```

```
## Warning: Strategy 'multiprocess' is deprecated in future (>= 1.20.0). Instead,
## explicitly specify either 'multisession' or 'multicore'. In the current R
## session, 'multiprocess' equals 'multisession'.
```

xgboost Upload Rate Prediction

Reading the Data

First, the data has to be read from the .csv file:

```
data_dir = "../datasets/"

dataset_ul = read_csv(
  str_c(data_dir, "dataset_ul.csv"),
  col_types = cols(scenario=col_factor())
)

dataset_ul_prediction = dataset_ul %>% select(
  scenario,
  velocity_mps,
  acceleration_mpss,
  rsrp_dbm,
  rsrq_db,
  rssnr_db,
  cqi,
  ta,
  payload_mb,
  f_mhz,
  throughput_mbits
)

# remove missing values
dataset_ul_prediction = dataset_ul_prediction %>% drop_na()
glimpse(dataset_ul_prediction)
```

```
## Rows: 6,168
## Columns: 11
## $ scenario          <fct> campus, campus, campus, campus, campus, campus, c...
## $ velocity_mps      <dbl> 11.80, 11.83, 11.70, 11.45, 11.49, 7.93, 8.15, 8....
## $ acceleration_mpss <dbl> 0.13, 0.03, 0.06, -0.32, -0.26, 0.23, 0.24, 0.32,...
## $ rsrp_dbm          <dbl> -99, -85, -121, -84, -97, -96, -74, -108, -111, -...
## $ rsrq_db           <dbl> -9, -5, -15, -6, -12, -12, -5, -9, -13, -11, -6, ...
## $ rssnr_db          <dbl> -1, 22, -8, 11, -2, 5, 29, 2, 6, 11, 13, 16, -3, ...
## $ cqi               <dbl> 8, 10, 4, 13, 9, 5, 15, 2, 6, 15, 12, 9, 6, 11, 1...
## $ ta                <dbl> 9, 7, 63, 4, 7, 7, 4, 21, 16, 7, 4, 4, 7, 16, 4, ...
## $ payload_mb        <dbl> 1.0, 4.0, 6.0, 2.0, 6.0, 5.0, 4.0, 10.0, 0.1, 7.0...
## $ f_mhz             <dbl> 1750, 1720, 1770, 1720, 1750, 1750, 1720, 1770, 1...
## $ throughput_mbits  <dbl> 4.66, 24.52, 1.29, 14.86, 3.97, 6.52, 16.27, 3.18...
```

Next, a prediction task has to be created to work with mlr3. The goal is to predict the variable `throughput_mbits`.

```
task = TaskRegr$new(
  id = "ul_prediction",
  backend = dataset_ul_prediction,
  target = "throughput_mbits"
)
task
```

```
## <TaskRegr:ul_prediction> (6168 x 11)
## * Target: throughput_mbits
## * Properties: -
## * Features (10):
##   - dbl (9): acceleration_mpss, cqi, f_mhz, payload_mb, rsrp_dbm,
##     rsrq_db, rssnr_db, ta, velocity_mps
##   - fct (1): scenario
```

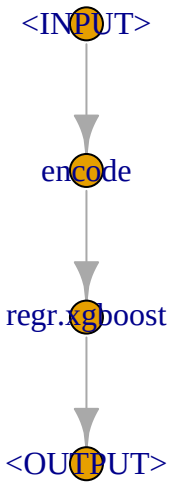
Creating the Prediction Pipeline

The next step is to create the prediction pipeline. It will consist of a factor encoding followed by the xgboost learner. The method for factor encoding is one-hot encoding. We put the pipeline creation inside of a function so that we can easily create untrained copies of the pipeline later when we need them.

```
make_pipeline = function() {
  factor_encoding = po(
    "encode",
    method = "one-hot",
    affect_columns = selector_type("factor")
  )
  xgboost = lrn("regr.xgboost")
  pipe = factor_encoding %>>% xgboost
  return(pipe)
}
```

Let's see what the pipeline looks like:

```
pipe = make_pipeline()
pipe$plot(html=FALSE)
```



Now the pipeline has to be converted to a learner so it can be used during training and prediction:

```
learner = GraphLearner$new(pipe)
```

Parameter Tuning

First, we have to define the set of parameters we use to tune the learner:

```
parameters = ParamSet$new(list(
  ParamInt$new("regr.xgboost.nrounds", lower=10, upper=500),
  ParamDbl$new("regr.xgboost.gamma", lower=0, upper=10),
  ParamInt$new("regr.xgboost.max_depth", lower=1, upper=10),
  ParamDbl$new("regr.xgboost.min_child_weight", lower=1, upper=100)
))
```

Next, we specify the tuning algorithm. For now, we use grid search:

```
tuner = tnr("grid_search", resolution=10)
```

Now all that is left is putting together the parts using the AutoTuner class. The resulting object is a learner that can automatically tune its parameters using the algorithm we specified.

```
tuned_learner = AutoTuner$new(
  learner = learner,
  resampling = rsmp("cv", folds = 5),
  measure = msr("regr.mae"),
  search_space = parameters,
  terminator = trm("evals", n_evals=100),
```

```
  tuner = tuner
)
```

Training the Learner

```
result = resample(
  task = task,
  learner = tuned_learner,
  resampling = rsmp("cv", folds=5)
)
```

Check if everything went well:

```
print(result)
```

```
## <ResampleResult> of 5 iterations
## * Task: ul_prediction
## * Learner: encode.regr.xgboost.tuned
## * Warnings: 0 in 0 iterations
## * Errors: 0 in 0 iterations
```

Prediction Results

```
# get r^2
result$aggregate(msr("regr.rsq"))
```

```
##  regr.rsq
## 0.8248892
```

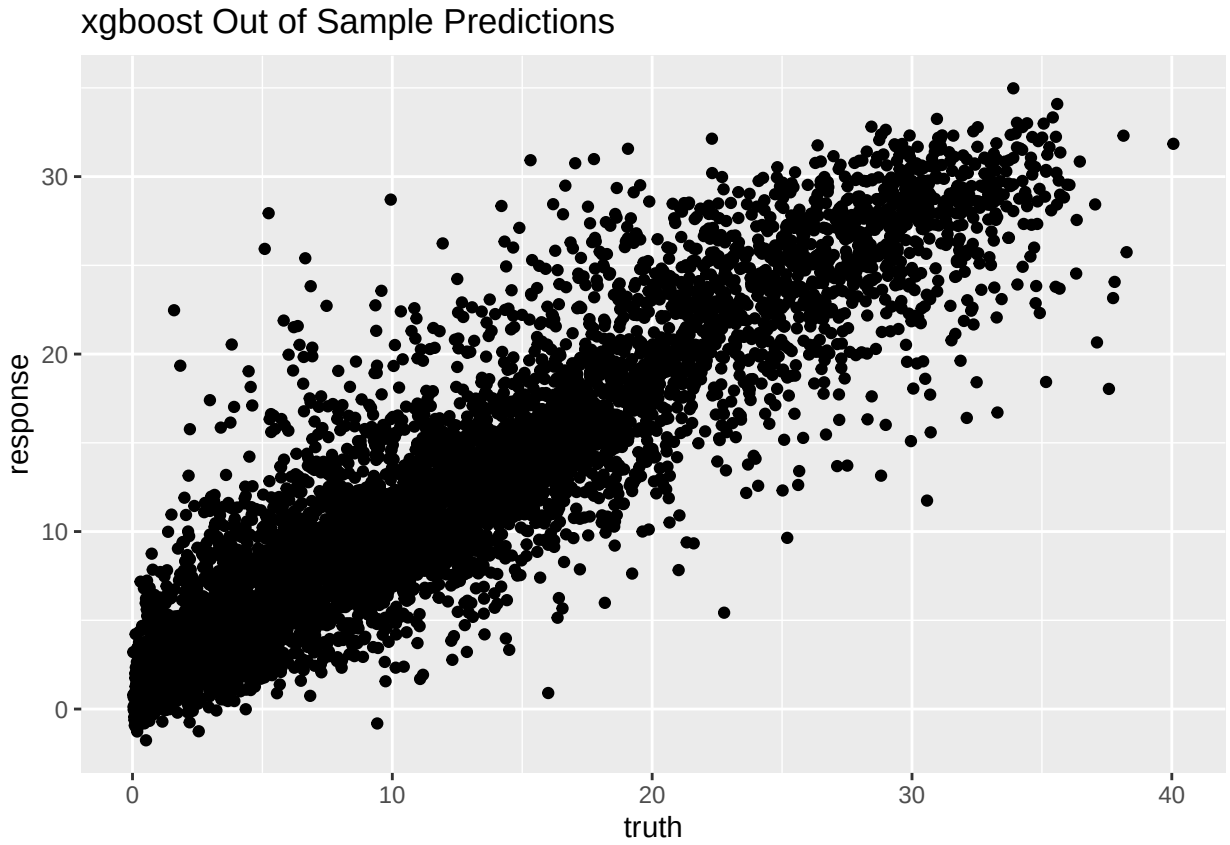
```
# get MSE
result$aggregate(msr("regr.mse"))
```

```
## regr.mse
## 13.66756
```

```
# get MAE
result$aggregate(msr("regr.mae"))
```

```
## regr.mae
## 2.668817
```

```
predictions = as.data.table(result$prediction())
ggplot(predictions) +
  geom_point(aes(x=truth, y=response)) +
  ggtitle("xgboost Out of Sample Predictions")
```



Feature Importance

Create an untrained learner with default parameters:

```
learner_default = GraphLearner$new(
  make_pipeline()
)
learner_default$param_set$values = mlr3misc::insert_named(
  learner_default$param_set$values,
  list(regr.xgboost.nrounds=100)
)
```

```
filter_permutation = flt("permutation",
  learner = learner_default,
  resampling = rsmp("holdout", ratio=0.8),
  measure = msr("regr.mae"),
  standardize = TRUE,
  nmc=10
)
filter_permutation$calculate(task)
```

```
filter_permutation_results = as.data.table(filter_permutation)
filter_permutation_results
```

```
##               feature      score
##  1:        payload_mb 2.23278346
##  2:             f_mhz 1.05008456
```

```
##  3:          rsrp_dbm 0.52315205
##  4:                ta 0.16514134
##  5:          scenario 0.14307804
##  6:      velocity_mps 0.09584147
##  7:               cqi 0.08013609
##  8:           rsrq_db 0.07383586
##  9:          rssnr_db 0.06093882
## 10: acceleration_mpss 0.02112324
```

```
ggplot(filter_permutation_results) +
  geom_bar(aes(x = reorder(feature, -score), y = score), stat="identity") +
  xlab("feature") +
  ylab("MAE difference") +
  scale_x_discrete(guide = guide_axis(angle = 20)) +
  ggtitle("Permutation Feature Importance")
```