Drivendata Cleaning time

Tomasz Waleń

March 2019

Objective

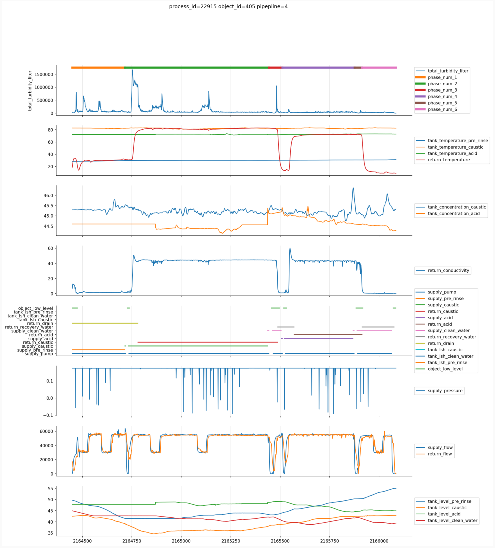
The contest is devoted to modeling of clean-in-place installations.

Given time-series with readings from various sensors the predict total turbidity in the last part of final rinse.

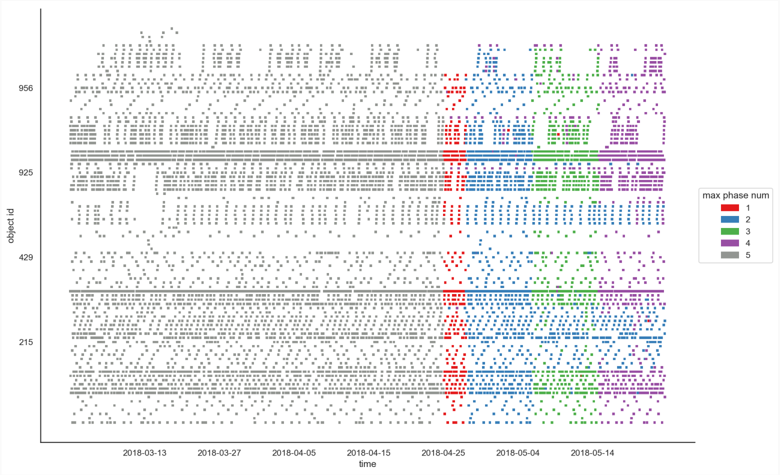
Train / test data

	train	test
# of series	5021	2967
# of obj	94	88
min. time	2018-02-21 17:42	2018-04-25 10:57
max. time	2018-04-25 12:02	2018-05-24 14:26

Sample process – process_id=22915

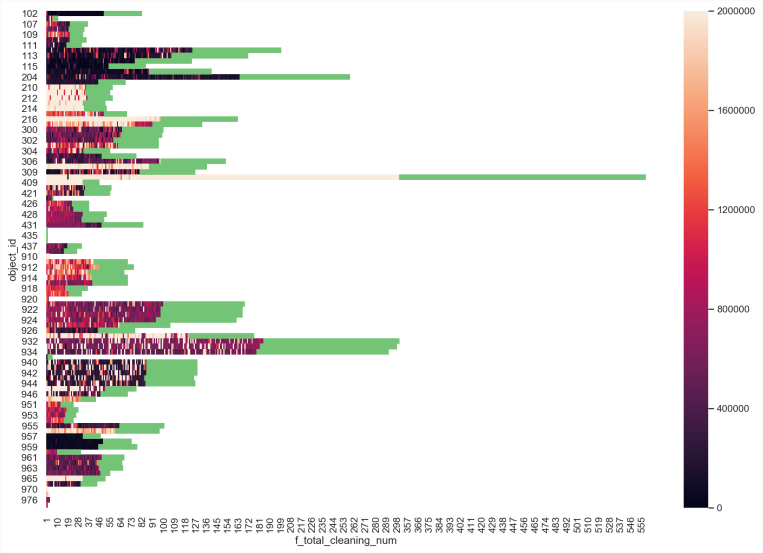


Train / test data



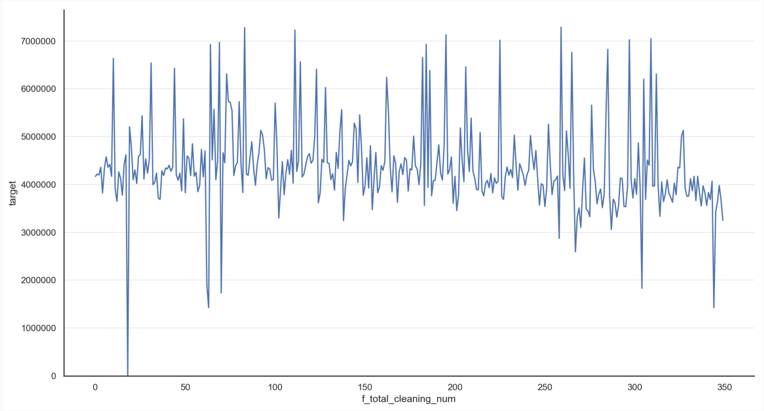
Train / test split is time based. Later processes have more data compared to early ones.

Train / test data - target values



Target values are from range $[360, 1.48 \cdot 10^8]$. Green markers - test set

Train / test data - targets for object_id=405



Test set

- it has different distribution than the training set (number of available phases)
- every object present in the test set is also present in the training set
- there is single pipeline (12) present in the training set, not present in test set

Approach

- split problem into 7 sub-problems:
  - given processes recipe X and max available phase number Y train model for all available data compatible with X/Y
  - considered recipes X: 1001, 1100, 1111
  - considered phases: 1, 2, 3, 4
  - there are 7 possible combinations in the test set
- for each segment create 2 models (one LGB and one Keras)
- the results are blended using weighted mean (weights are calculated based on the results from the validation set)

Features

- sizes of each phases
- aggregate values of input parameters in each phase
- target mean values, aggregated by object id, pipeline

Validation

- train/validate split is time-based (similar to train/test split), usually I was using processes $<$ 2018-04-23 for training, and processes $>=$ 2018-04-23 for validation
- based on the size of the segment in the test set the weight is computer (# of processes / total size of test set)
- the resulting validation is a weighted sum of validations in each segment

Code

- `src/features/build_features.py` contains code for calculating features
- `src/models/contest_models.py` contains code for training models
- for segmenting the `ContestSegmentationLayer` is used
- for blending results and generating submodels `ContestBlendedModelV1` is used
- for scaling features the `ContestScalingLayer` is used