# II. Code submission and result reproducibility

The model is very similar to the example model <http://drivendata.co/blog/rinse-over-run-benchmark>. The total model consists of 4 single lightgbm models which are stored in the file lightgbm\_Allmendinger.py.

All process steps, such as reading the data, preparing the data, fitting the model and generating the predictions (submit file) are in this file.

As a development environment, I use Windows 10, Python 2.7.  
To run the script i use the anaconda package Spyder.

To run the script (lightgbm\_Allmendinger.py), the libraries

1. Pathlib
2. Numpy
3. Pandas
4. Lightgbm (conda install -c conda-forge lightgbm)

must be installed.

It is important that for lightgbm the latest version is installed (training with objective 'mape' should be run without error)

The raw data (recipe\_metadata.csv, submission\_format.csv, test\_values.csv, train\_labels.csv, train\_values.csv) must be in the subdirectory **data**.

If you execute the script (lightgbm\_Allmendinger.py) the raw data are read in, prepared, the model is fitted; the forecasts for the test data are generated and written to the disk in the file with name 'submission\_GMBlight\_27.csv'.

# III. Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally?  
   I finished the HFT - Stuttgart University of Applied Sciences 1990. Since 25 years I work as a Simulation Engineer for the Company PPI-Informatik. I use discrete event simulation and optimization methods to analyse and optimize production- and logistic processes.
2. High level summary of your approach: what did you do and why?  
   I build a separate simple gradient boost model for each of the 4 cases:   
   process\_id with phases = 1  
   process\_id with phases <=2   
   process\_id with phases <=3   
   process\_id with phases <=4  
   To prediction the final\_rinse\_total\_turbidity\_liter in the test data, for process\_id in the test data the maximum number of phases are determined and the prediction was made with the corresponding phase-model.   
   With this approach we can utilize the (the few) trainings data very well.
3. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.
   1. Train lightgbm model with objective 'mape'  
      params ={ 'boosting\_type': 'gbdt',   
       'learning\_rate': 0.01,   
       **'objective': 'mape',**   
       'num\_leaves': 49,  
       'bagging\_freq': 5,   
      …  
       }
   2. Defined the features *object\_id* and *pipeline* as categorical feature in lightgbm:  
      dtrain = lgb.Dataset(x\_train, np.log1p(y\_train), free\_raw\_data=False, categorical\_feature=[0,1])
   3. Use The MAPE metric as early stopping in lightgbm.cv:  
      def lgb\_mean\_absolute\_max\_error(preds, train\_data):  
      labels = np.expm1(train\_data.get\_label())  
      d=np.maximum(np.abs(labels), np.array([T]\*len(labels)))  
      return 'mame', np.mean(np.abs(labels - np.expm1(preds)) / d), False
   4. Use for process\_id with phases <= i (i=1,2,3,4) a own model  
      for phase in np.arange(4)+1:  
       X = train\_values[train\_values.phase <= phase].copy()  
       x\_train = create\_feature\_matrix(X, phase)
   5. Transform y = response=*final\_rinse\_total\_turbidity\_liter* with log(.)+1:

dtrain = lgb.Dataset(x\_train**, np.log1p(y\_train),** free\_raw\_data=False, categorical\_feature=[0,1]).   
A power transformation also works with similar results (np.power(y, 1/10)

1. What are some other things you tried that didn’t necessarily make it into the final workflow (quick overview)?  
   I experimented with deep neural networks with object function “mae” (best public score 0.3444 private score 3.71.   
   I've tried also dimensional reduction (PCA) and stacking but without success.
2. Did you use any tools for data preparation or exploratory data analysis that aren’t listed in your code submission?  
   No, I have not used any other tools.
3. How did you evaluate performance of the model other than the provided metric, if at all?  
   From my point of view, the metric was the key to success. I did not use any other metrics to evaluate the model.
4. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?   
   For the given training data, the model provides stable results, even if the input data changes slightly. The calculations run without problems on my Laptop DELL Inspiron 5759 with 16 GB Ram and a i7-6500U CPU with 2.6 GHz processor and Windows 10) .
5. Do you have any useful charts, graphs, or visualizations from the process?   
   For feature engineering I only used the feature importance charts from lightGBM to decide which derived feature can lead to improvements.   
   To get familiar with the problem, I loaded the data into Excel and “played” with it.
6. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?   
   If it's just about winning a contest, I'd try to use other methods that optimize the given Metric MAPE. I would also try to develop more “time series” related features.   
   If the goal is to develop an application, I'd look at how different metrics affect performance and accordingly to this, I would engineer the features.