1. **Who are you (mini-bio) and what do you do professionally?**

Riccardo is a Data Scientist at BCG gamma. He focuses on building machine learning models at scale for data driven decision making.

Riccardo has experience with Agile development and he is a certified scrum master.

Before BCG Riccardo worked as data scientist for an energy company with focus on energy price predictions. He worked also for an IT consulting Firm building data driven applications for finance, retail and aerospace industries.

1. **High level summary of your approach: what did you do and why?**

I have trained a two-layer stacked model of ligthGBM. The model has been trained on the maximum available granularity (i.e. time series level) and then aggregated with a weighted median to obtain predictions with process granularity.

1. **Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.**
   1. **generation of the aggregated validation set**

Different folds are subsampled by phase in order to obtain a similar distribution of the testset and thus a reliable validation set.

**def** generate\_train\_cuts(data, mode=**''**):  
 eliminations = {**'el1'**: [**'final\_rinse'**],  
 **'el2'**: [**'acid'**, **'final\_rinse'**],  
 **'el3'**: [**'intermediate\_rinse'**, **'acid'**, **'final\_rinse'**],  
 **'el4'**: [**'caustic'**, **'intermediate\_rinse'**, **'acid'**, **'final\_rinse'**],  
 }  
  
 df\_list = []  
 process\_list = data.process\_id.unique()  
 **for** el, list\_el **in** eliminations.items():  
 **if** el == **'el4'**:  
 \_, process\_el4 = train\_test\_split(process\_list, shuffle=**False**, test\_size=0.35)  
 df = data[data[**'process\_id'**].isin(process\_el4)]  
 df = df[~df[**'phase'**].isin(list\_el)]  
 **else**:  
 df = data[~data[**'phase'**].isin(list\_el)]  
 print(mode + **' shape for '**, el, **':'**, df.shape)  
 df\_list.append(group\_level1\_preds(df))  
 agg\_df = pd.concat(df\_list, ignore\_index=**True**)  
 **return** agg\_df

* 1. **Aggregate the predictions:**

In this part of code, predictions (one for each measurement line) are aggregated using a weighted median (in order to optimize MAPE)

**def** group\_level1\_preds(filtered\_data):  
 df\_list = []  
 model\_list = [x **for** x **in** filtered\_data.columns **if** x.startswith(**'m'**)]  
 filtered\_data[**'logdur'**] = np.log(1 + 0.1 \* filtered\_data[**'duration'**] / 10000)  
  
  
  
 **for** col **in** [**'ml2'**]:  
 df = filtered\_data.groupby(**'process\_id'**).apply(**lambda** x: pd.Series({  
 col + **'\_wgmedianlogdur'**: weighted.median(x[col], x[**'logdur'**] / x[col]),  
 }))  
 df\_list.append(df)  
  
 **if 'final\_rinse\_total\_turbidity\_liter' in** filtered\_data.columns:  
 filtered\_data[**'score'**] = mapeu(filtered\_data[**'final\_rinse\_total\_turbidity\_liter'**], filtered\_data[**'ml2'**])  
  
 df = filtered\_data[filtered\_data.groupby(**'process\_id'**).score.transform(**'min'**) == filtered\_data[**'score'**]]  
 df[**'ml2best'**] = df[**'ml2'**]  
 df = filtered\_data.groupby(**'process\_id'**).apply(**lambda** x: pd.Series({  
 **'final\_rinse\_total\_turbidity\_liter'**: x[**'final\_rinse\_total\_turbidity\_liter'**].max(),  
  
 }))  
 df\_list.append(df)  
  
 **return** pd.concat(df\_list, axis=1)

* 1. **Optimize on weighted L1**

In this part of the code a weight vector is set up. In this way lightGBM can use it for the optimization

**if** usewg:  
 print(**'using wg'**)  
 *# take the labels* div\_train = y\_true\_dict[train\_dataset].copy()  
 *# set labels<290000 to 290000* div\_train[div\_train < 290000] = 290000  
 div\_val = y\_true\_dict[val\_dataset].copy()  
 div\_val[div\_val < 290000] = 290000  
 *#create ligthgbm vector with weigts* d\_train = lgbm.Dataset(x\_dict[train\_dataset], y\_dict[train\_dataset], weight=1 / div\_train)  
 d\_valid = lgbm.Dataset(x\_dict[val\_dataset], y\_dict[val\_dataset], weight=1 / div\_val)

1. **What are some other things you tried that didn’t necessarily make it into the final workflow (quick overview)?**

* Train models at process\_id aggregation level
* Catboost + Xgboost
* A lot of feature engineering

1. **Did you use any tools for data preparation or exploratory data analysis that aren’t listed in your code submission?**

No

1. **How did you evaluate performance of the model other than the provided metric, if at all?**

The predictions in the out folds + the data subsampling by phase (see 3.1) seem to provide and efficient method to validate the model

1. **Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?**

The code runs without any problem on a laptop with 32 GB of ram. The main issue is the execution time (It take 36 hours to run on my laptop).

If the size of the dataset changes consider a re-tune of the models parameters

1. **Do you have any useful charts, graphs, or visualizations from the process?**

No, the process is implemented without visualization

1. **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?**

* Add object\_id metadata
* Data collection of less standardized processes
* Have a better understanding of the process in order to create explicit features
* Create a model based on a RNN on LSTM in order to provide a better optimization for the aggregation part (see 3.2)