# III. Model documentation and write-up

You can respond to these questions either in an e-mail or as an attached file (any common document format is acceptable such as plain text, PDF, DOCX, etc.) **Please number your responses.**

1. Who are you (mini-bio) and what do you do professionally?  
   I am a multidisciplinary data scientist with a passion for building predictive models to help, automate business process workflows in different business functions of IT industry . I have 7 years of experience in this field and mostly worked in data science projects pertaining to contact center, retail, manufacturing and energy industries .

|  |
| --- |
| **If you are on a team, please complete this block for each member of the team.** |

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

Sampling: Sample the train dataset to make it look more like the test dataset

Data preprocessing: I didn’t do much as I have planned to use tree based models. I have log transformed the response variable to make it more normally distributed.

Feature Engineering:

1.local features: features based on summary statistics(min,max,median,mean) on the given features/phase

2.global features: features on summary statistics(min,max,median,mean) on the given features without considering the phase

3. final rinse features: summary statistics of final rinse features in train dataset which is groupded by object\_id and then later merged with test dataset with object\_id

4. interaction features: two-way interaction of numeric features

5. Features created using response variable

Model:

A single lightgbm model with the above mentioned features(easy and faster to train, less preprocessing and handles NA )

Regression type: Quantile regression

Validation strategy: 10-fold cross-validation, hyperparameters are hand tuned

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

1. def groupby\_(vals,col\_name,train,test):

dd = train[['final\_rinse\_total\_turbidity\_liter']+[vals[0]]].groupby(vals[0]).mean().reset\_index()

dd.columns = [vals[0]]+[ col\_name]

dd[col\_name] = np.log(dd[col\_name])

#dd['turbidity\_median'] =dd['turbidity\_median']

train = train.merge(dd,on=vals[0],how="left")

test = test.merge(dd,on=vals[0],how="left")

dd = train[['final\_rinse\_total\_turbidity\_liter']+[vals[1]]].groupby(vals[1]).mean().reset\_index()

dd.columns = [vals[1]]+[ col\_name+"\_pipeline"]

dd[col\_name+"\_pipeline"] = np.log(dd[col\_name+"\_pipeline"])

#dd['turbidity\_median'] =dd['turbidity\_median']

train = train.merge(dd,on=vals[1],how="left")

test = test.merge(dd,on=vals[1],how="left")

return train test

Helps to create features using response variable w.r.t object\_id that considerably improved the performance of the model.

2. agg\_fns = ["mean","median","std","min","max"]

#agg\_fns = ["median","std","min","max"]

count=0

for feat in real\_val\_feats:

print(feat)

zz = train[['object\_id',feat]].groupby(['object\_id']).agg(agg\_fns).reset\_index()

zz.columns = zz.columns.droplevel(0)

zz["range"] = zz["max"]-zz["min"]

zz.columns = ['object\_id']+[feat+"\_"+k for k in agg\_fns+["range"]]

if count==0:

count+=1

train\_agg = zz

else:

train\_agg = train\_agg.merge(zz,on=['object\_id'],how = "left")

Helps to create aggregate features in a single go.

3.Setting objective function to Quantile improved the score tremendously as the response distribution has long tails to the right and also adjusting alpha of quantile regression helped in improving the score further

params = {

'objective': 'Quantile',

'boosting': 'gbdt',

'learning\_rate': 0.006,

'verbose': 1,

'num\_leaves': 65,

'bagging\_fraction': 0.85,

'bagging\_freq': 1,

'bagging\_seed': 12345,

'feature\_fraction': 0.6,

'feature\_fraction\_seed': 123,

'max\_bin': 150,

'max\_depth':7,

'num\_rounds': ROUNDS,

'min\_sum\_hessian\_in\_leaf':0.25,

'min\_data\_in\_leaf':20,

"alpha":0.3,

"min\_data\_in\_bin":30

}

1. What are some other things you tried that didn’t necessarily make it into the final workflow (quick overview)?  
   - models based on catboost and xgboost

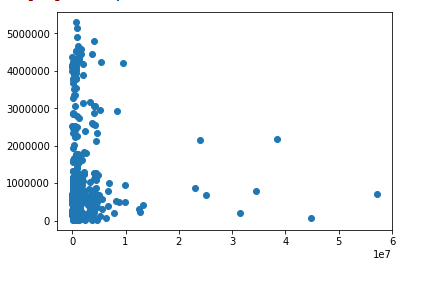
* Models trained using Huber loss, Fair loss,Mape as objective function
* Custom MAPE based objective functions
* Individual models on different phases
* Features extracted using FFT

1. Did you use any tools for data preparation or exploratory data analysis that aren’t listed in your code submission?  
   No. Numpy and Pandas are the two packages I used extensively
2. How did you evaluate performance of the model other than the provided metric, if at all?

I have created the provided metric as a custom metric in lighgbm and used it to early stop the models

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.)?  
   A system with atleast 8Gb of ram would handle the script well.
2. Do you have any useful charts, graphs, or visualizations from the process?

Model built doesn’t do very good for large response variables.



X\_axis = actual response , Y\_axis = Predicted response

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?

Since the dataset is time based, I would try some of the deep learning techniques (1D convolution , LSTM) to build an end-end model.