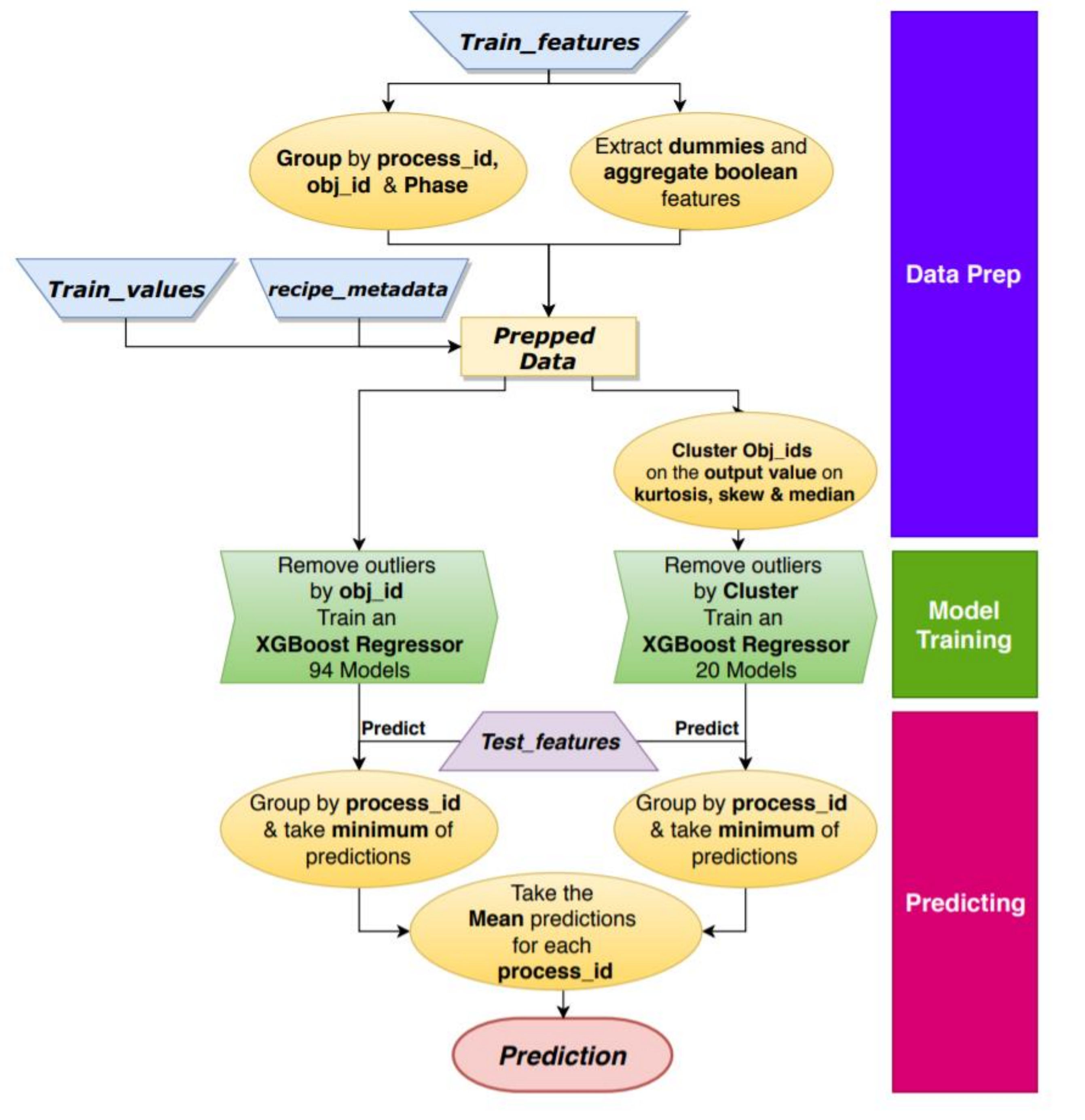
DRIVEN DATA: RINSE OVER RUN

THE BI SHARPS:

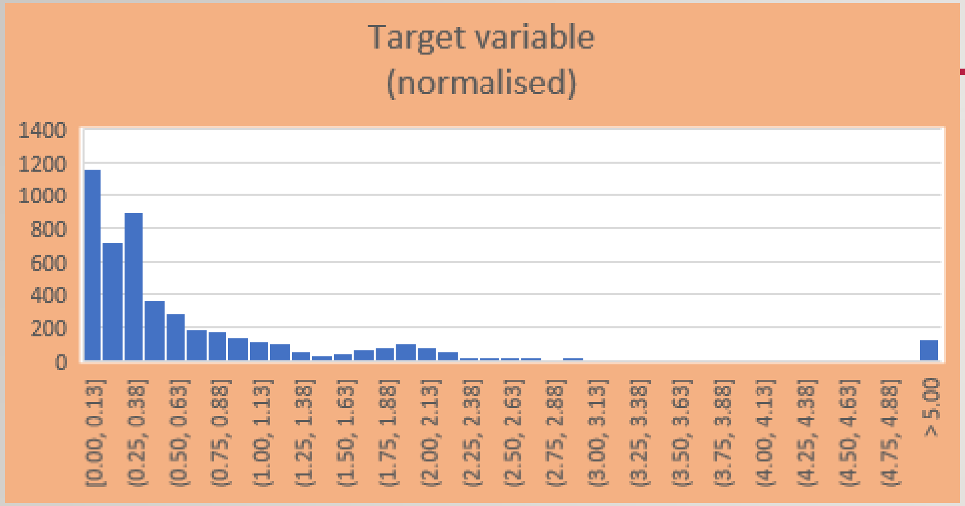
DAVID BELTON & PAT WALSH

OUR PROCESS

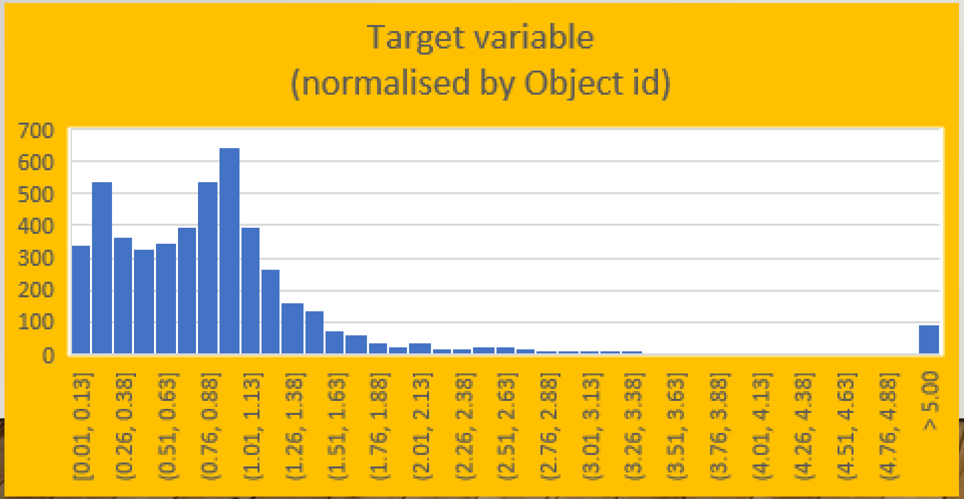


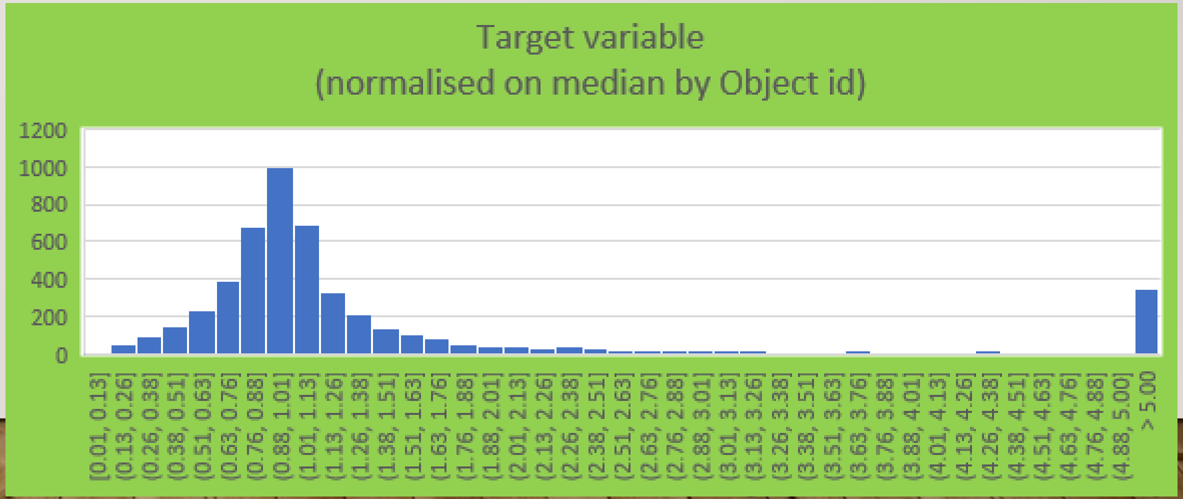
Train_features
Group by process_id, obj_id & Phase
Extract dummies and aggregate boolean features
Train_values
recipe_metadata
Prepped Data
Cluster Obj_ids on the output value on kurtosis, skew & median
Remove outliers by obj_id Train an XGBoost Regressor 94 Models
Remove outliers by Cluster Train an XGBoost Regressor 20 Models
Predict
Test_features
Predict
Group by process_id & take minimum of predictions
Group by process_id & take minimum of predictions
Take the Mean predictions for each process_id
Prediction
Data Prep
Model Training
Predicting

VARIANCE OF THE TARGET VARIABLE



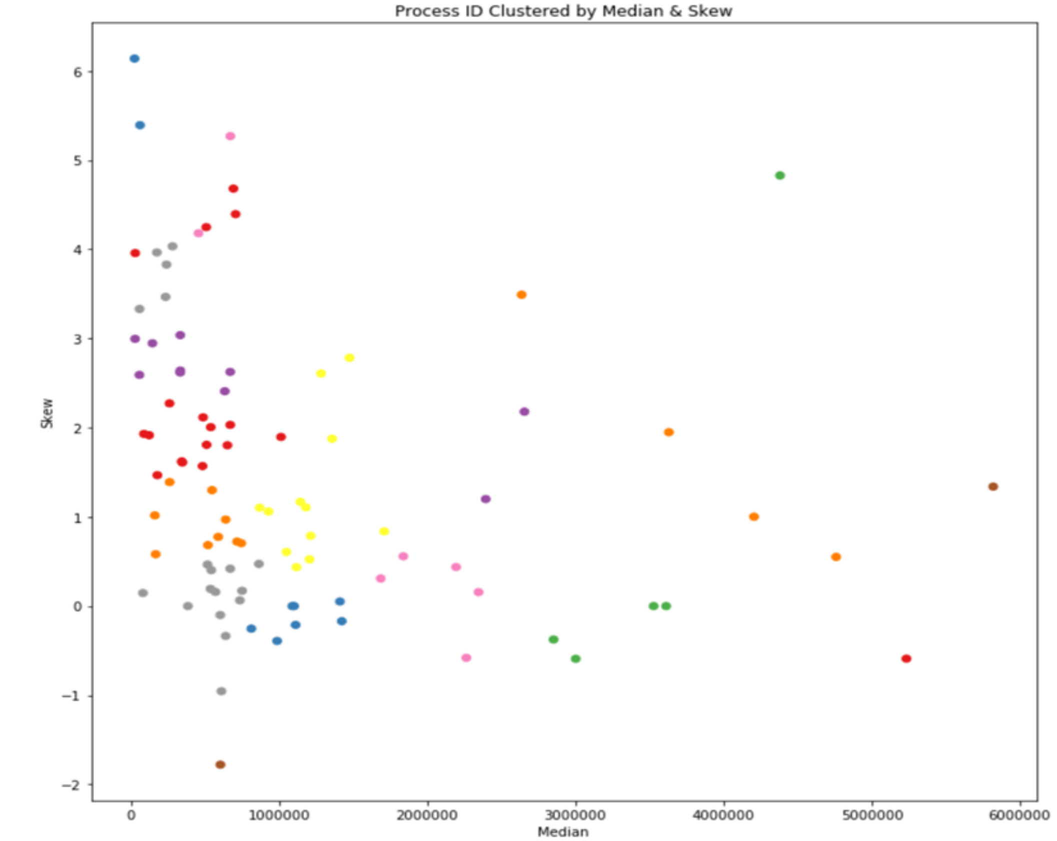
- When we look across all processes we see a large variance in the target variable
- The variance in Target is less when looking at individual object_ids





CLUSTERING SIMILAR OBJECT IDS

- With many object_ids having few processes associated with them, we clustered similar object_ids using the Kmeans algorithm
- We clustered based on the target's normalised median, skew & kurtosis.
- This gave us larger samples of similarly distributed outputs to train on



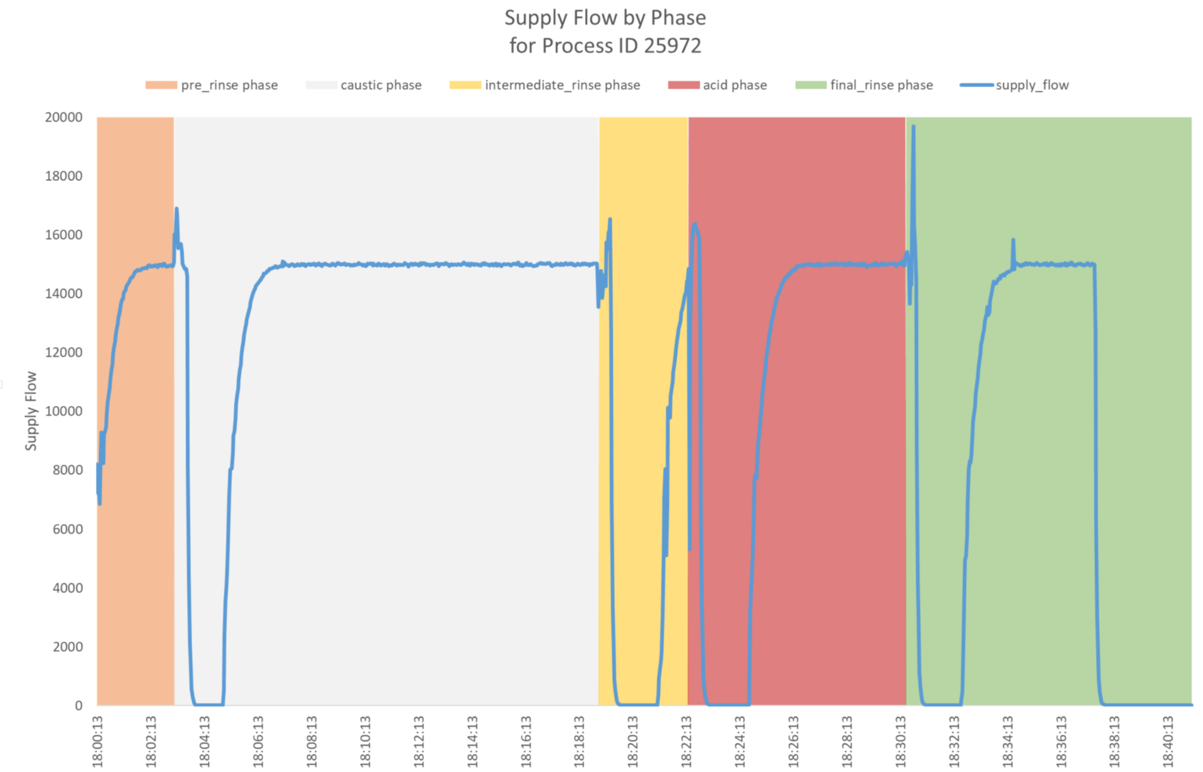
DATA PREP: GROUPING BY PHASE

- We built features based on summary stats for each of the numeric variables:
    - Mean, std, min, max & median
- For our initial models, we built features for each of these summary stats split by phase, as in the example in table below:

Process_id	Median Supply Flow pre_rinse Phase	Median Supply Flow caustic Phase
25972	14,762	14,985

- We found that by creating a new record for each phase instead and using a dummy variable worked much better, ie. Grouping by process_id and phase. Example of this final dataset is below:

Process_id	Phase	Median Supply Flow	Pre_rinse dummy	caustic dummy
25972	Pre_rinse	14,762	1	0
25972	caustic	14,985	0	1

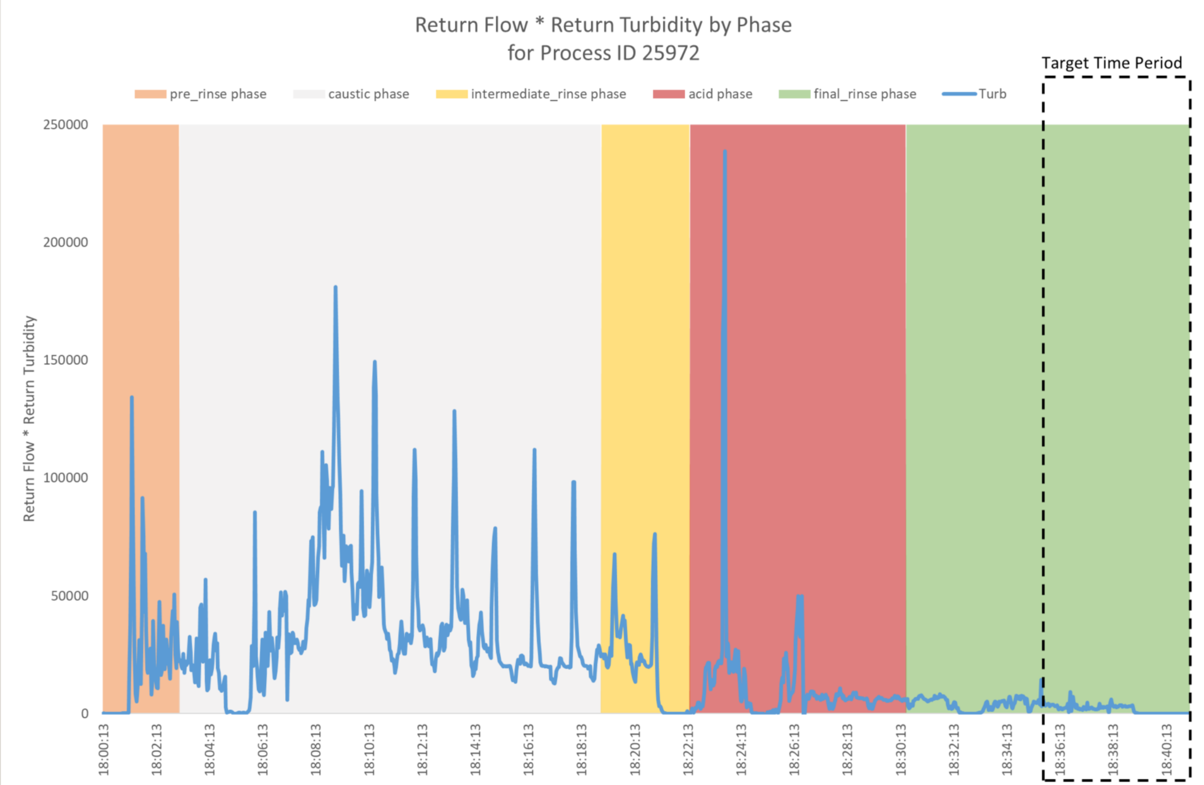

Supply Flow by Phase for Process ID 25972

DATA PREP:
FEATURE ENGINEERING

- To interpret the Boolean columns such as “return_recovery_water” we used a percentage True column sum/count by phase. (0.09 feature importance)
- We also created a variable

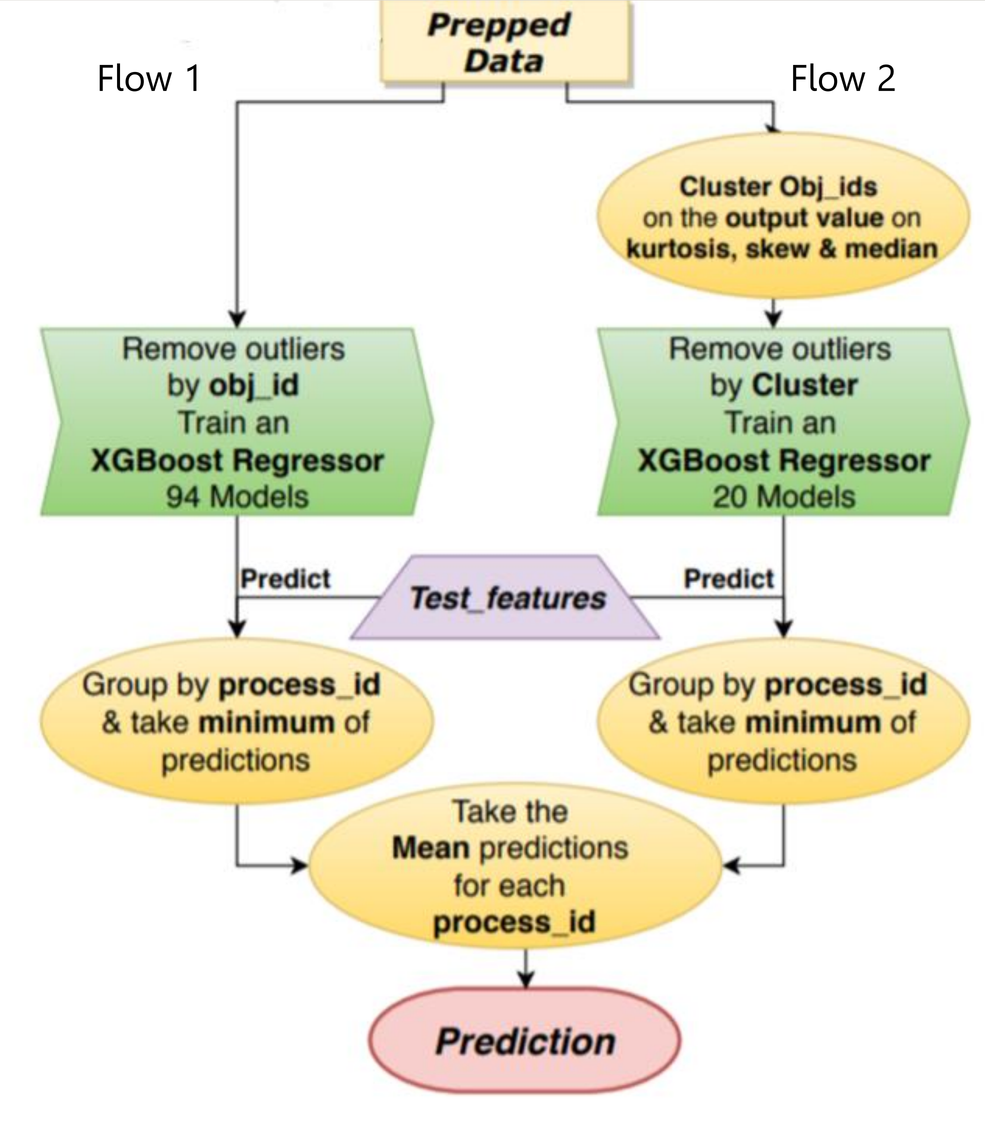
$turb = return\_flow * return\_turbidity$

We aggregated this in the same way as the other numeric columns (0.07 feature importance)

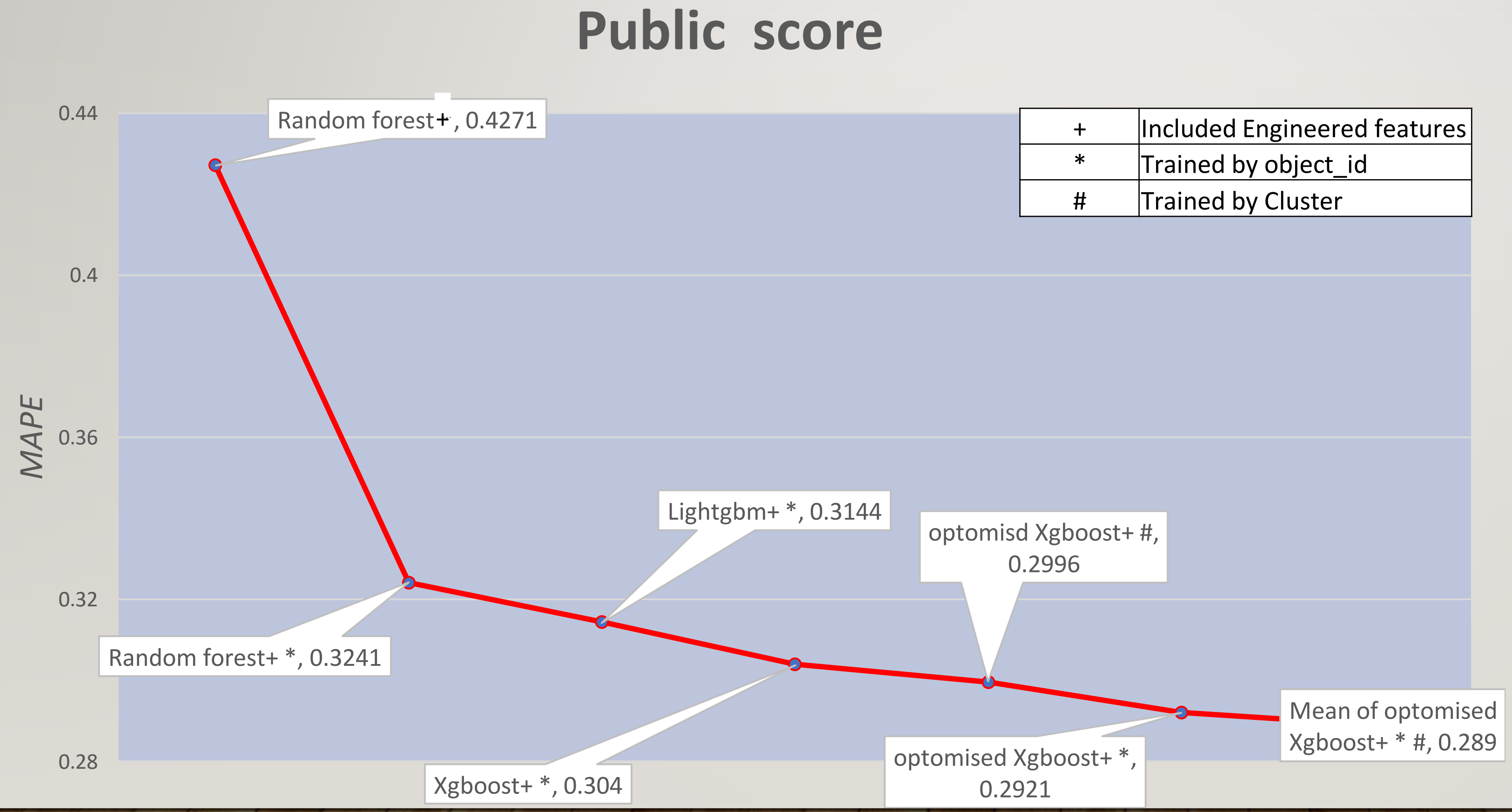


MODELLING

- We had 2 pipeline flows for our modelling process
- For the first flow we built a separate model for each object_id
- For the second we built a separate model for each cluster of similar object_ids
- The average of the output from each flow was used as our overall prediction


Prepped Data
Flow 1
Flow 2
Cluster Obj_ids on the output value on kurtosis, skew & median
Remove outliers by obj_id Train an XGBoost Regressor 94 Models
Remove outliers by Cluster Train an XGBoost Regressor 20 Models
Predict
Test_features
Predict
Group by process_id & take minimum of predictions
Group by process_id & take minimum of predictions
Take the Mean predictions for each process_id
Prediction

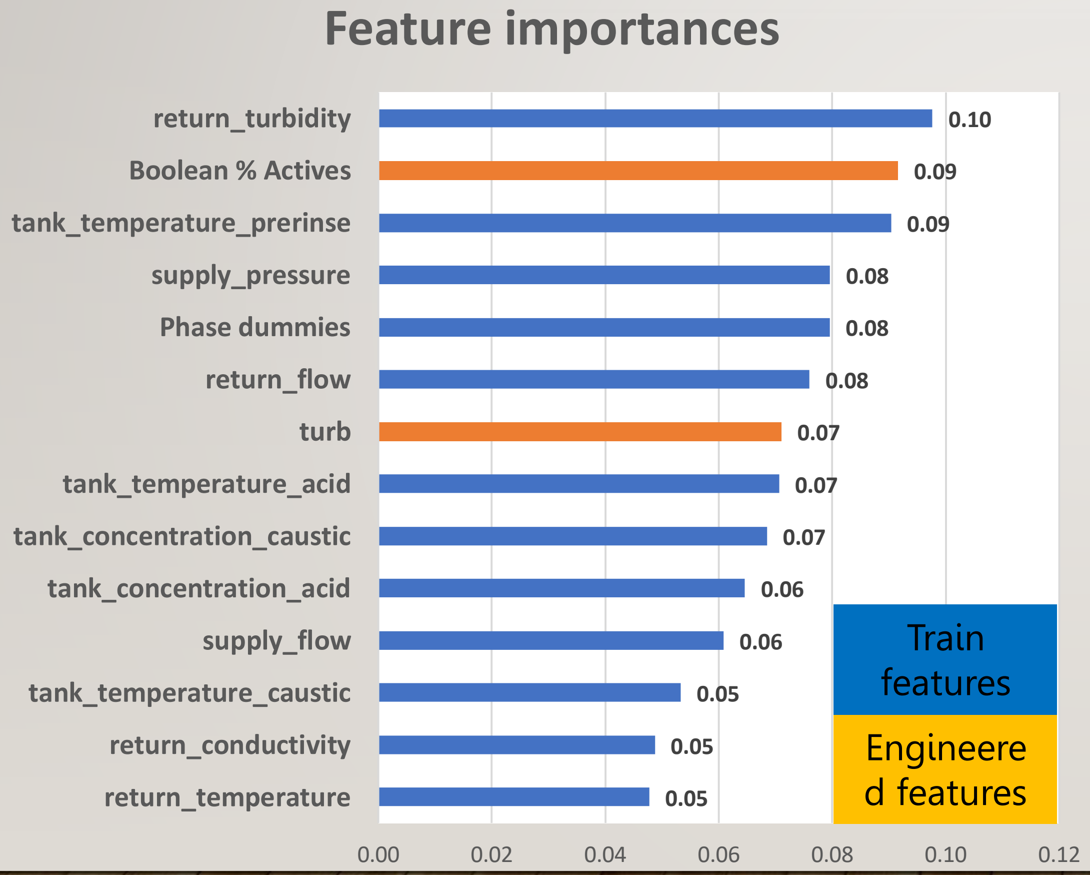
MODEL COMPARISON



+	Included Engineered features
*	Trained by object_id
#	Trained by Cluster

- We achieved the largest improvement in score when, instead of building one model, we built a model for each object_id or cluster of object_ids (0.4271 to 0.3241)
- Boosting based models outperformed random forest
- A blend of a xgboost model per object_id and xgboost model per cluster gave us our best score

FEATURE IMPORTANCE

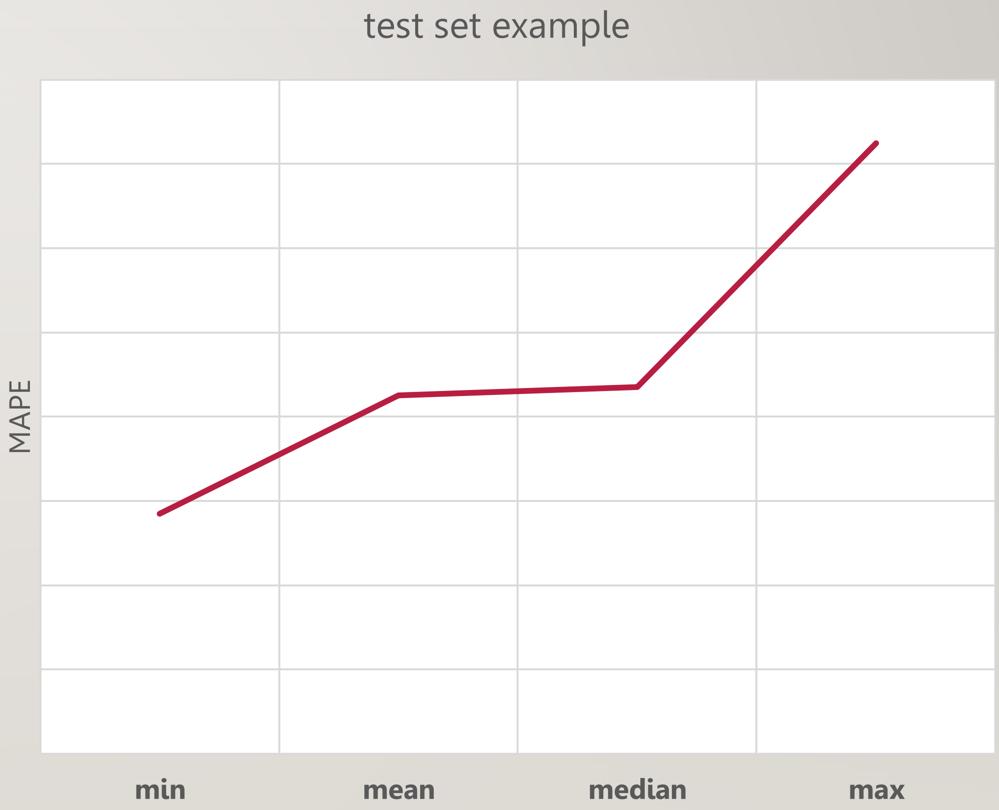


- Return turbididty, the % of time valves are open and temperature of the pre_rinse fluid tank are the strongest indicators

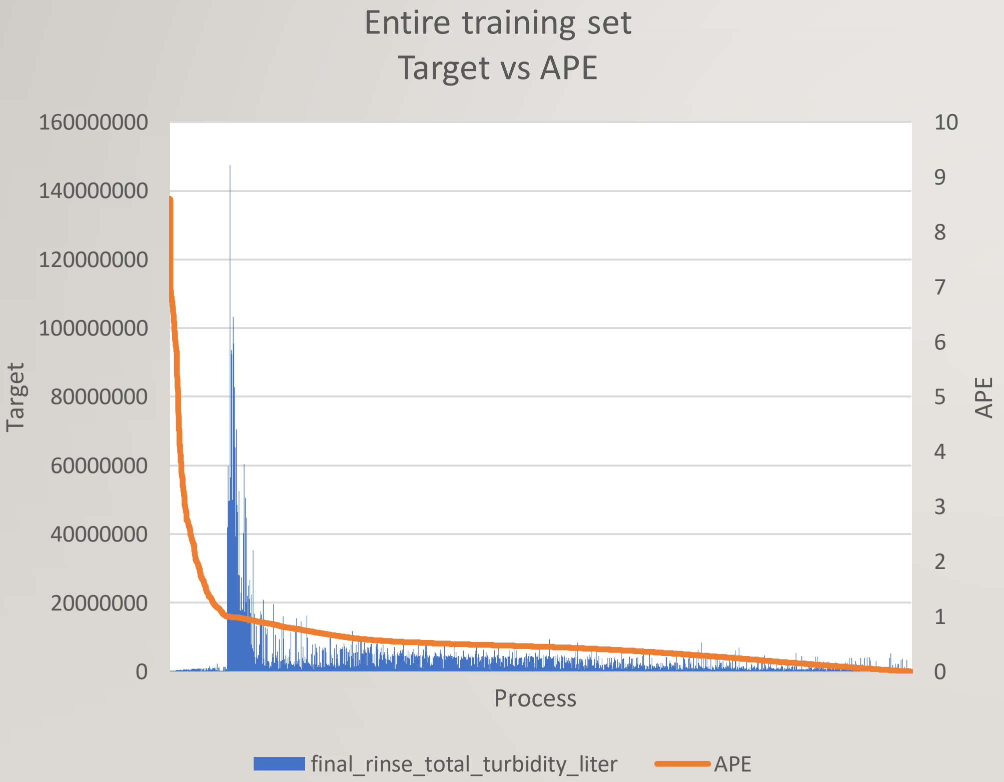
- Return conductivity and temperature are less prominent in the importance, which was a surprise, intuitively we would associate water conductivity with suspended particles and hence turbidity.

CHOOSING A PREDICTION

- Since we grouped by process_id and phase, we made a prediction for each process_id and phase. We needed to determine which one prediction should be used for the process_id
- Using the prediction from a particular phase did not prove fruitful. We found that the minimum prediction for each process_id was the best performing.
- This could be partially due to the MAPE metric preferring under prediction to over predicting the target



COMPARISON: TARGET VS PREDICTED



- Our Method performed worst on small target values as APE punishes over prediction
- It also performed poorly on high outliers but the APE metric was more forgiving of these
- Optimising for these small target values was our area of most focus at the close of the competition but we didn't make a breakthrough