March 18th, 2019

Sustainable Industry: Rinse Over Run Modeling Report

Arindam Bhattacharya

Executive Summary

End-to-end overview of analytical process

Problem Statement

- Goal: increased resource efficiency when cleaning food and beverage industrial equipment while maintaining high cleaning standards
- Solution approach: build model that accurately predicts the *total residue** during the final rinse phase using historical cleaning data
- Produce empirical, model-derived insights into the factors that drive increased turbidity during the final rinse phase

Data Preparation and Feature Engineering

- Extensive feature engineering to distill time-series data into a set of representative characteristics during supply and return phases
- Features primarily generated at a phase-level of granularity or lower, allowing for straightforward simulation of mid-process predictions
- Data cleaning included outlier detection and minor data imputation

Modeling

- Four separate models, one corresponding to the end of each phase
- **LightGBM boosted tree model** with custom loss function
- Training/validation splits created **using walk-forward/rolling origin** techniques; training data ranges from first 44 to 56 days of available data
- Estimates of generalization error obtained through weighted average of validation set errors across all four models

Performance and Insights

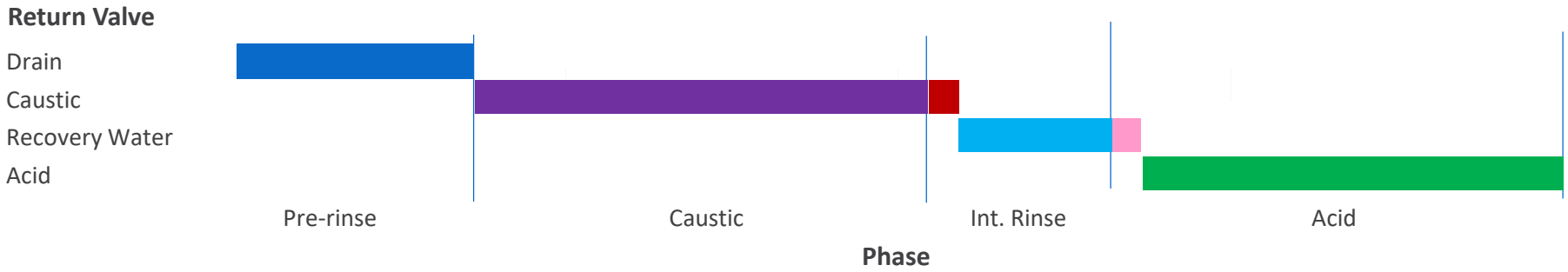
- Best average MAPE on holdout test data (private leaderboard): **27.4%**
- Model-driven insights derived using SHAP (Shapley Additive Explanations)
- Primary factor that influences total final phase residue: the specific type of object being cleaned, or object_id
- Additional key factors: total weighted turbidity and residue, minimum temperature, minimum conductivity, recipe type

**Residue* is our simplified nomenclature for the response variable during modeling, “total_turbidity_liter” (total quantity of turbidity * outgoing flow).

Feature Engineering

Phase subdivision, time-series aggregations

Return valve states during a typical cleaning process



Phase	Return Phase
Pre-rinse	Pre-Rinse-Drain
Caustic	Caustic-Caustic
Int. Rinse	Int-Rinse-Caustic
Int. Rinse	Int-Rinse-Recovery-Water
Acid	Acid-Recovery-Water
Acid	Acid-Acid

The above diagram illustrates the danger of creating aggregated features at the phase level, as multiple cleaning agents can be "returned" or "supplied" during a particular phase (pink and red bars).
Consequently, **a key initial step was to sub-divide phases into "return phases" and "supply phases"** to account for the actual cleaning agent being returned during the phase.
Return-phase and supply-phase combinations that occurred relatively infrequently were bucketed as "other" to reduce overfitting.

Return phase-level features

- Total residue
- Total weighted turbidity*
- Minimum conductivity
- Duration

*Weighted by normalized return flow; details/rationale are presented in Insights section

Supply phase-level features

- Total supply flow of cleaning agent
- Minimum supply pressure
- Duration

Phase-level features

- Total residue during end of phase
- Total weighted turbidity* during end of phase
- Minimum return temperature
- Proportion of phase duration during which caustic tank was empty
- Proportion of phase duration during which liquid was present in the cleaning object

Other features

- Object ID
- Recipe Type

Modeling

Modeling and validation techniques

Primary Modeling Technique: Gradient Boosted Trees (LightGBM)

A state-of-the-art implementation of gradient-boosted decision trees with several attractive features:

- Excellent predictive power
- Efficient implementation → short training time (5-10 seconds per model, depending on number of features) → ability to rapidly iterate when developing new features, tuning hyperparameters, etc.
- Supports custom loss functions and evaluation metrics
- Robust and efficient handling of categorical data – finds optimal splits without the need for alternate encodings (e.g. one-hot)
- SHAP (model interpretation technique, discussed later in Insights section) implements a special high-speed, exact algorithm in C++ for LightGBM models that reduces the time required to interpret model output

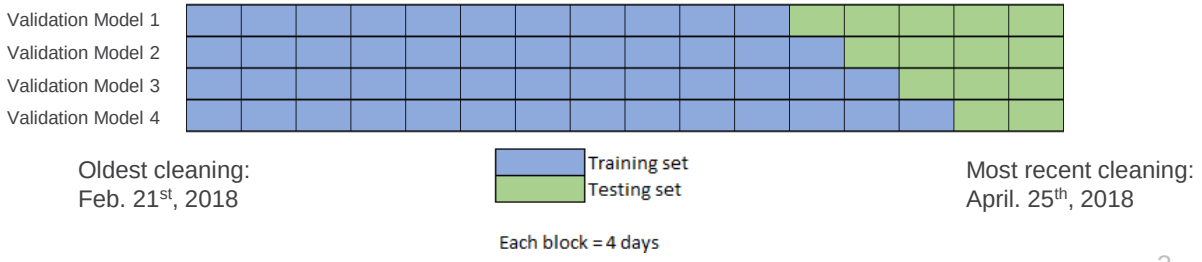
Methodology details

Loss function: $L(\hat{y}, y) = \frac{|\hat{y}_i - y_i|}{\max(|y_i|, 290000)}$

- Directly minimize performance metric
- LightGBM typically uses both gradient and Hessian to build trees
- Gradient = $\frac{\text{sgn}(\hat{y}_i - y_i)}{\max(|y_i|, 290000)}$, Hessian = 0 (where defined)
- Implemented by modifying source code for LightGBM's built-in MAPE loss function, which only uses the gradient to build trees
  (custom loss function functionality in Python utilizes a second-order approximation and requires a non-zero Hessian)

Walk-forward/rolling origin validation sets

- K-fold CV not appropriate for time-series due to data leakage
- Training sets ranged from 44 to 56 days of data (64 days of data in total)
- Four validation sets per model; average estimated error across all four validation sets to obtain estimated generalization (test set) error



Modeling

Multiple model rationale and additional methodology details

Additional methodology details

Data cleaning and preparation

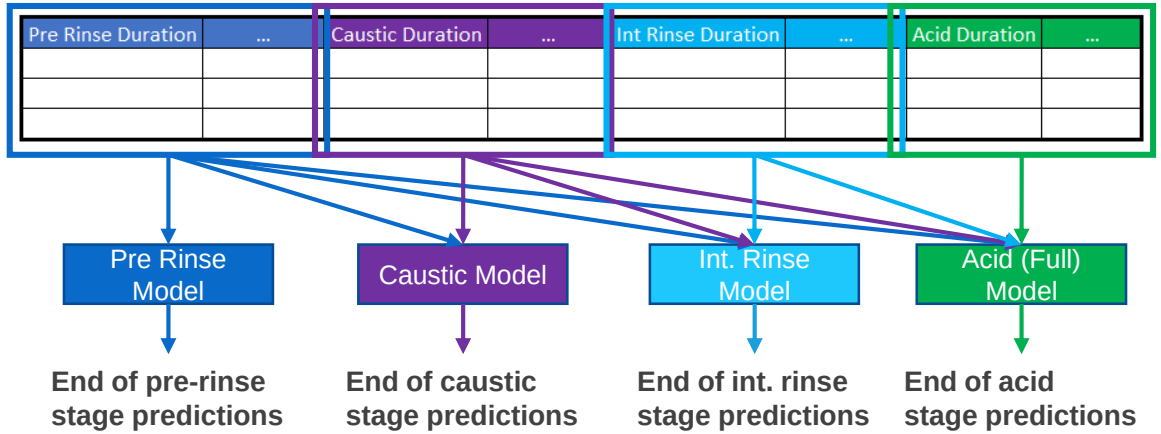
- Set all negative return and supply flows to zero
- Compute object_id-level median statistics for several features, for use during feature normalization (rationale explained in Insights)
- Identify and remove outlier records along total process duration (e.g less than 20 seconds total) and response value (e.g. total final rinse residue < 1000) dimensions using quantile-based thresholds

Intentionally avoided techniques

- Some commonly used modeling techniques were unsuitable for this analysis, given the importance of model reproducibility and interpretability
- *Stacking/ensembling*: very difficult to determine contributions of variables to final output when multiple models are combined
- *PCA, autoencoders, etc.:* dimensionality reduction techniques which result in new, significantly less interpretable predictors
- *Model tuning techniques that involve randomness into the model-building process*, such as selecting a random subset of features before building each tree (LightGBM parameter “feature_fraction”); can lead to inconsistent model interpretation across multiple runs

Simulating Mid-Process Predictions with Multiple Models

Although a single LightGBM model could make predictions at any point in the cleaning process, its performance would be greatly hampered by the considerable missing data in the later stages (e.g. 70% of the acid-phase features). To maximize predictive performance, we built four separate models corresponding to the four phases, simulating a mid-process prediction by excluding the appropriate predictors:



Additional benefits of this approach include:

- Distinct estimates of generalization error for each type of prediction
- Full utilization of the entire training set for all four estimates of test set error
- Visibility into the unique factors driving each type of prediction

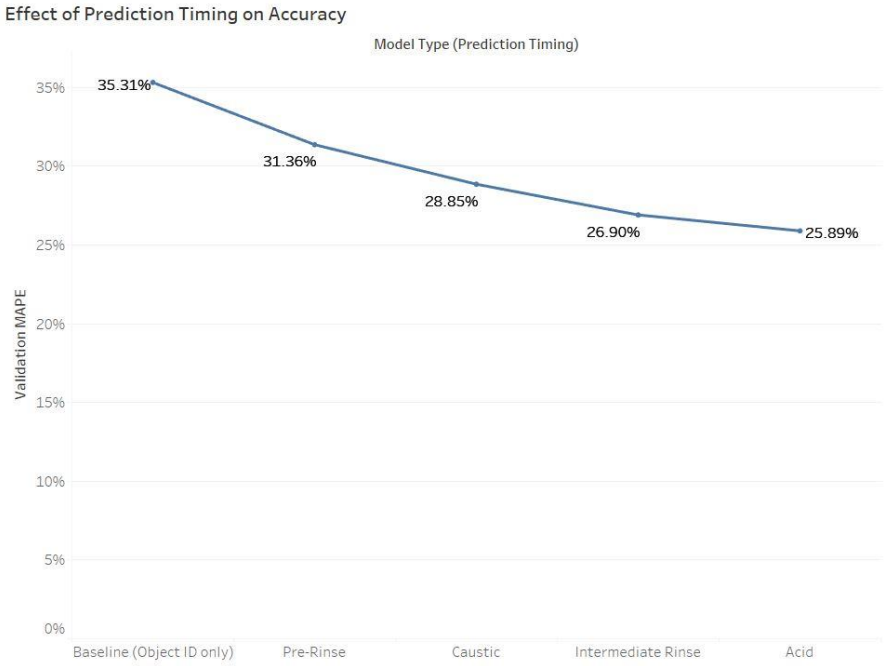
Performance

Prediction accuracy

Best test set MAPE (private leaderboard): 27.4%

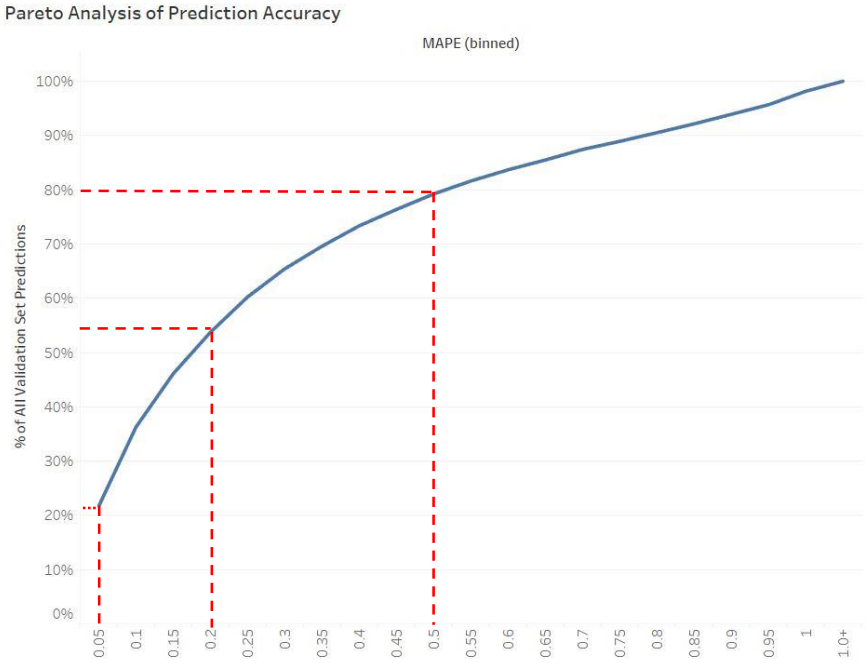
Since we do not have the actual response values from the test set, we can use the predictions made on the validation sets as a proxy for the purposes of evaluating model prediction quality

Effect of Prediction Timing



As expected, the later in the cleaning process the prediction is made, the more accurate the model prediction will be

Pareto Analysis



Approximately **20%** of validation set predictions were within **5%** of the observed response

The **majority** of validation set predictions were within **20%** of the observed response

Approximately **80%** of validation set predictions were within **50%** of the observed response

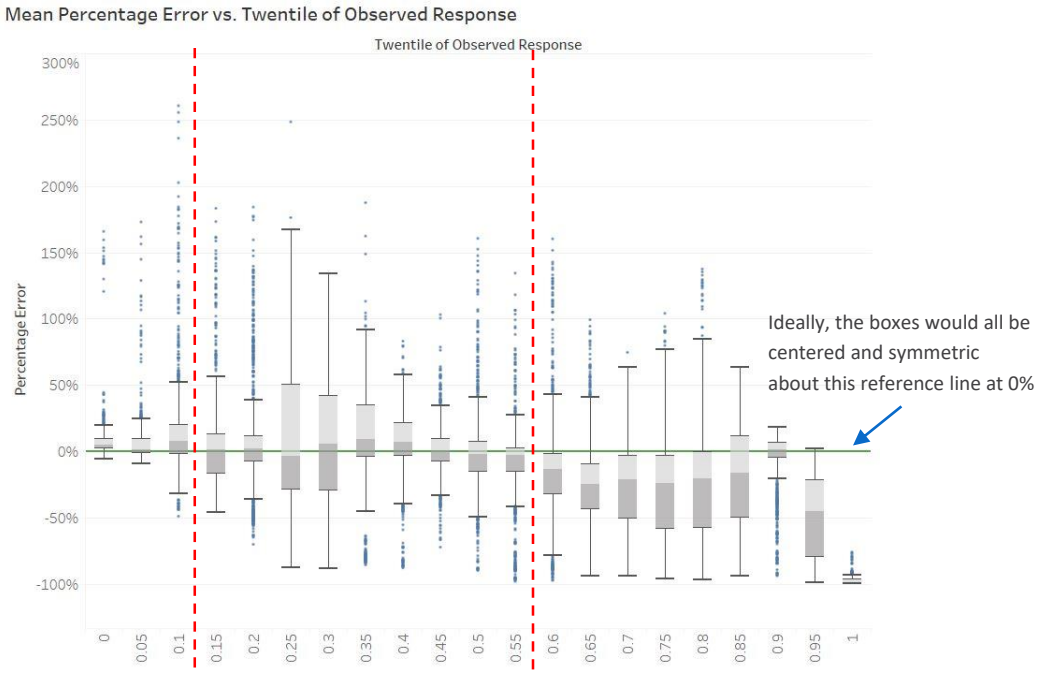
Performance

Prediction characteristics and patterns

Best test set MAPE (private leaderboard): 27.4%

Since we do not have the actual response values from the test set, we can use the predictions made on the validation sets as a proxy for the purposes of evaluating model prediction quality

Prediction Biases

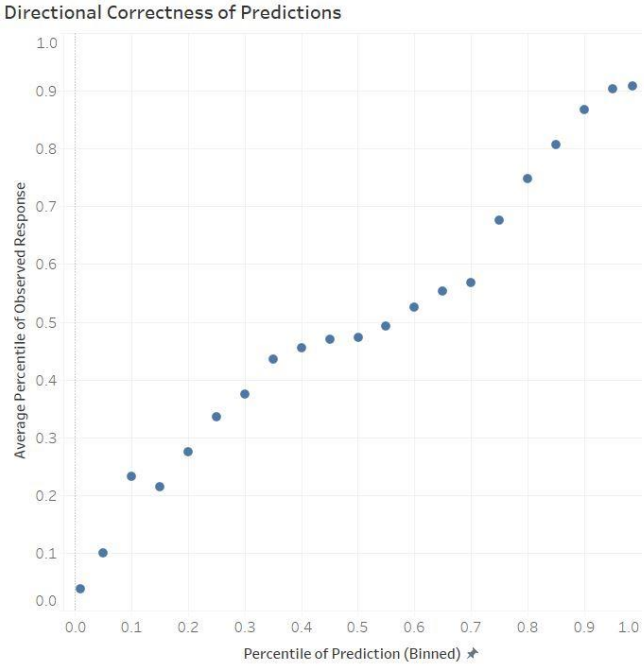


Ideally, the boxes would all be centered and symmetric about this reference line at 0%

The model slightly over-predicts the smallest observed values due to the "max(290000,…)" adjustment made to MAPE

The model significantly under-predicts the largest observed values
This is typical of models that minimize MAPE, due to the asymmetric costs of over- and under-prediction

Directional Correctness



Despite the clear systematic over- and under-bias at the extremes of the observed values, the predictions show good directional correctness (**Spearman's ρ = 0.80**)

**Twentile* is a shorthand for quantiles/deciles with 20 bins, each containing 5% of the data.

Insights

SHAP overview, most important predictor

Primary model interpretation technique: SHAP (Shapley Additive Explanations)

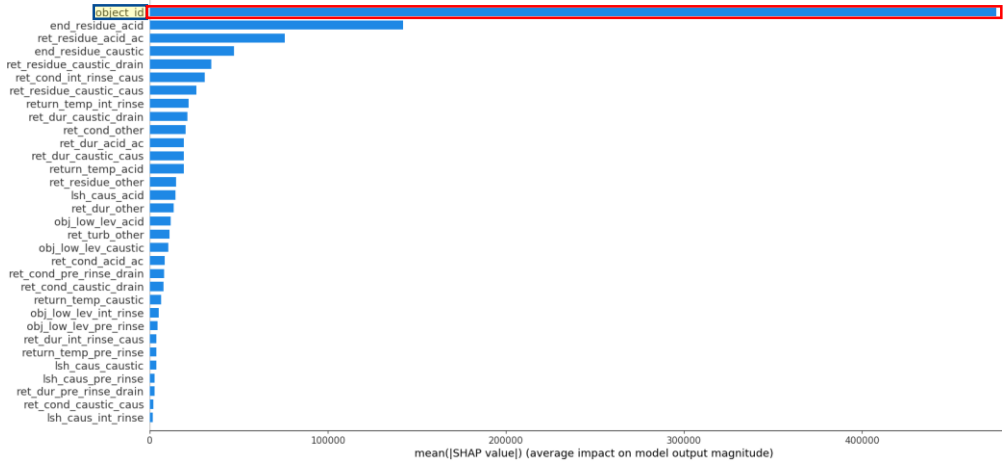
- Rooted in the concept of a “Shapley value” from coalitional game theory, which attempts to distribute the “gains” or “payout” from a cooperative game to its players in a “fair” manner
- Adapted to the context of black-box ML models by treating the features as players and the difference between the prediction and global average of the observed response as the payout
- Shapley value of a predictor can be interpreted as the average marginal contribution to the payout made by that predictor across all possible “predictor coalitions” which include it
- Has several unique and attractive theoretical properties, such as decomposing the payouts into contributions made by each feature in the model (efficiency), which means estimated effects of predictors are unique to each prediction
- However, it is very computationally expensive; SHAP mitigates this downside by using a special weighting kernel that allows the SHAP values to be computed using weighted linear regression

Key feature that drives model performance: object_id

- Most important model feature by a very large margin across all four models
- Basic EDA, such as a box-and-whisker plot, reveals stark differences in observed response values across different objects; not an unexpected result
- Baseline model with only object_id achieved a test set MAPE of 35.3%; this result is in the ballpark of the subsequent derived-feature models, again reinforcing how critical the feature is

Additional feature engineering based on implications of key feature

- The overwhelming importance of object_id as a feature suggests that subsequent modeling patterns may be object-specific in nature
- Consequently, several initial features were centered using object-level aggregations (median of training set only), which led to further improvements in model performance



Mean SHAP values for acid (full) model, training set = first 56 days of data; object_id was similarly dominant in terms of impact on predictions in other models as well

Original Features	Centering Method
Minimum conductivity, Minimum supply pressure	Subtract object-level median
Return flow	Divide by median
Residue (turbidity * return flow)	Turbidity * normalized return flow, which can be interpreted as a pseudo-normalized residue or a **weighted turbidity**

Insights

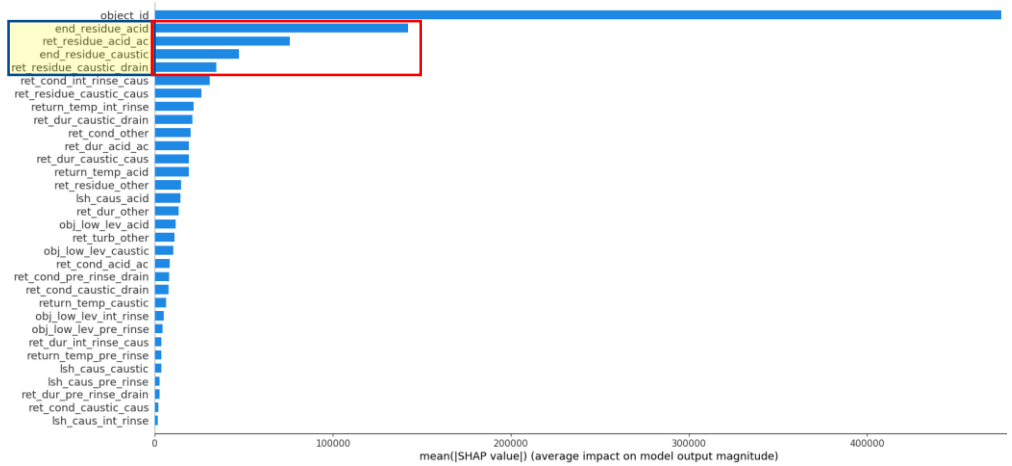
Residue and turbidity-derived predictors

Second-most important class of predictors: residue and weighted turbidity

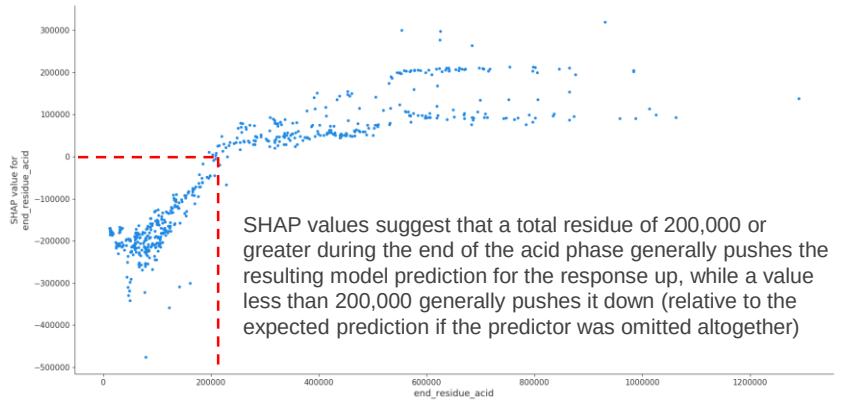
- These predictors account for most of the improvement in model performance over the baseline model (only object_id)
- Weighted turbidity had more predictive power in first three stages; total residue produced the best results in the acid stage (full model)
- Residue and turbidity were calculated in two different manners: for the entire return phase, and for just the end of the phase (last 80 seconds)

High residue/weighted turbidity is a strong indicator of high residue during the final rinse

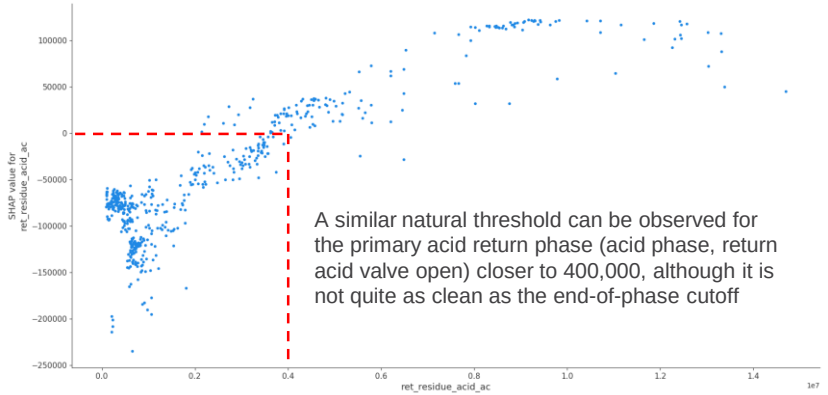
- The more total residue that flows out during (and especially at the end of) an earlier cleaning phase, the more total residue one can expect during the final rinse phase
- Effect appears to be strongest for caustic and acid phases



Residue produced the best results in the acid model; weighted turbidity was similarly important in the other three models



SHAP values suggest that a total residue of 200,000 or greater during the end of the acid phase generally pushes the resulting model prediction for the response up, while a value less than 200,000 generally pushes it down (relative to the expected prediction if the predictor was omitted altogether)



A similar natural threshold can be observed for the primary acid return phase (acid phase, return acid valve open) closer to 400,000, although it is not quite as clean as the end-of-phase cutoff

SHAP dependence plots for total residue during last 80 seconds of acid phase (left) and entire acid-acid return phase (right)
Each point corresponds to the SHAP value for a single validation set prediction

Insights

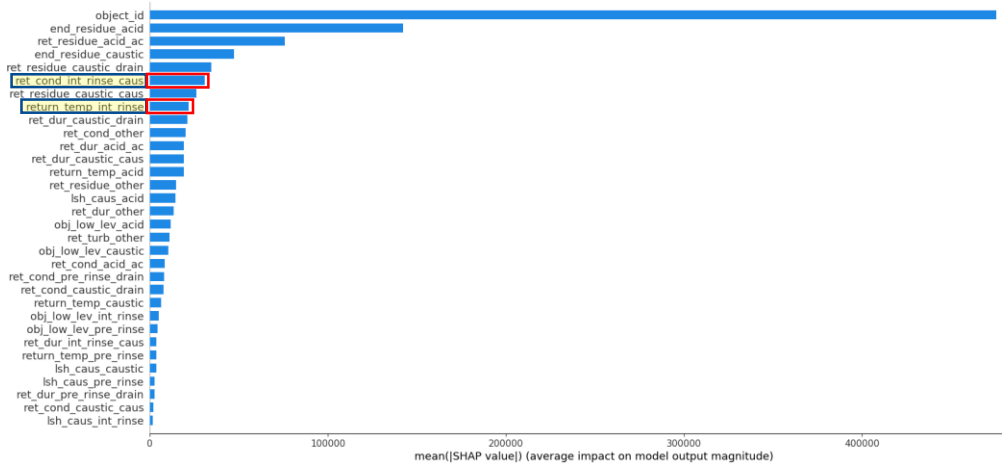
Other noteworthy predictors

Conductivity (minimum)

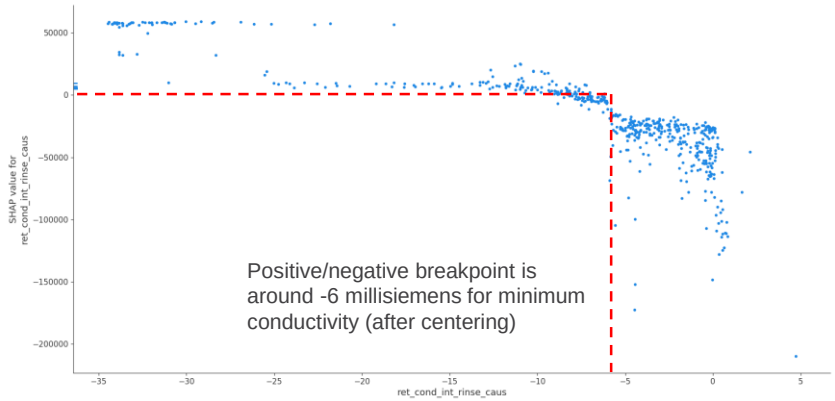
- Primary predictive power comes from considering this feature when the caustic return valve is open during the intermediate rinse phase (usually at the beginning)
- Centered by object_id; negative value indicates the minimum conductivity during the return phase was atypically low for that particular object
- Inverse effect – the lower the minimum conductivity, the more residue we predict during the final rinse – consistent with physical intuition (one would expect a high concentration of solid particulates to reduce the conductivity of the cleaning agent)

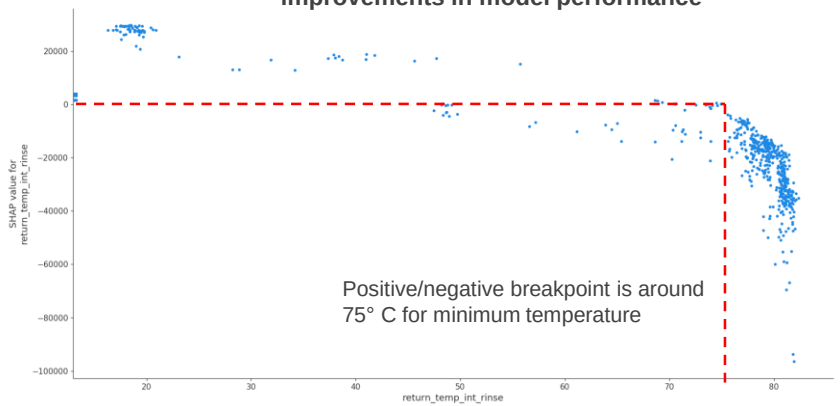
Temperature (minimum)

- Very similar in nature to conductivity – primarily useful when considered during intermediate rinse, pronounced inverse effect on final rinse residue, consistent with intuition
- Unlike conductivity, not normalized prior to model building



Minimum conductivity (top) and minimum temperature (bottom) during intermediate rinse phase also resulted in non-trivial improvements in model performance





SHAP dependence plots for minimum conductivity during intermediate rinse return-phase, caustic valve open (left) and minimum temperature during intermediate rinse phase (right)

Conclusion

Miscellaneous insights and predictors, future work

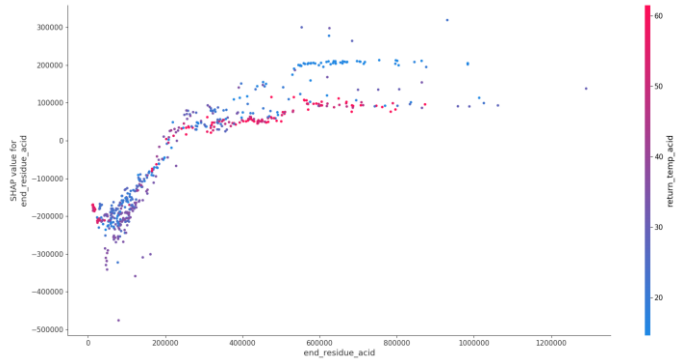
Miscellaneous Insights

Other model predictors

- Recipe type very important in pre-rinse and caustic models; 3rd largest SHAP importance value for both
- Features derived from phase duration, obj_low_level, and tank_lsh_caustic also modestly improved predictions

Higher-order interactions between predictors

- SHAP values can be used to identify interactions between predictors; Python implementation has built-in estimation of strongest interactions for a given predictor
- Provides deeper insight into how predictors can amplify or dampen each others' effects on the response



Second-order dependence plot between end-of-phase total residue and min. temperature during acid phase. Striking visual patterns that illustrate the interaction effects are readily apparent.

Miscellaneous Predictors

In addition to the predictors that made it into the final model, several predictors which did not appear to improve predictive accuracy were thoroughly investigated:

- Measures of volatility/variability for previously discussed features (e.g. standard deviation, coefficient of variation, maximum rolling range over some fixed window, etc.)
- Features derived from tank level and temperature; probably did not help due to relatively low variability
- Temporal features such as day of week, hour of day, etc.
- Relationships between supply and return flow (e.g. ratio of supply flow to lagged return flow)

Although these features did not seem to improve model performance, they may have an impact on final rinse residue that only becomes apparent with deeper exploration (i.e. other derived features, higher-order interactions) and/or additional data

Future Work

Operationalization

- Building a real-time tool for operators to make enhanced decisions on final rinse run time using model predictions
- Tool can leverage SHAP values to increase confidence in recommendations by illuminating the key factors driving each prediction

Additional Data

- The importance of object_id naturally suggests a number of related features that characterize each object
- Brand, dimensions, primary functionality (i.e. a mixer, granulator, etc.), company that uses the equipment, what types of food are processed in the equipment, etc.
- These characteristics could greatly improve how well the predictive models generalize to newer equipment with no or little historical data

Alternative Modeling Approaches

- Alternative performance metrics such as L_1 or L_2 loss could help alleviate the systematic biases that are inherent to MAPE-minimizing models
- Alternative responses such as "time until rolling average turbidity drops below *some threshold* during final rinse phase" could lead to model-driven predictions that are more actionable for operators