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?

We are part of the data science team at Contiamo. Our company offers a data science platform, and the data science team is responsible for the on-boarding of clients with projects to demonstrate the capabilities of the platform.

- **Nicolas Gautier** has background in experimental physics. After working as a physical simulation software salesman, he joined Contiamo where he is responsible for customer on-boarding, delivering proof of concepts, and advanced process analytics research.
- **Ian Solliec** has an applied mathematics background. He worked in finance for almost a decade, developing pricing and risk models, before joining Contiamo.
- **Brandon Williams** has a mathematics background. He has worked at tech startups in a variety of industries such as healthcare and insurance.

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

- **Data preparation**: We truncated phases in the training data in order to match the phase distribution of the test set, and we split the training data into a training set and a validation set.
- **Feature engineering** (and selection): We transformed timeseries data into features, so that the resulting structured data could be fed to standard machine-learning algorithms.
- **Modeling**: We decided to model the log of the target variable, as it was better distributed (even if this complicated the choice of an objective function). We then tried a variety of algorithms, settling on Gradient Boosted Trees based on its good initial performance.
- **Fine-tuning**: We tried to fine-tune the model's hyper-parameters, but there was little improvement (the error decreased by a mere 0.5%). This is a good sign, as it points to the inherent robustness of the algorithm.

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

We find it hard to answer this question. We believe that the most impactful code we wrote is the code we don't show: all the exploration and experimentation that led to our model being what it is. Everything that we left out, rather than what we left in, which ends up being relatively simple.

If anything, the investment we made in making feature calculation fast helped ensure faster iteration. However, this is just standard pandas gymnastics, and there is nothing particular to highlight (rather, the first throw is usually sub-optimal). Similarly, having access to good hardware made experimentation faster and more likely to yield results.

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

- **More feature engineering**: we tried many ways to summarize timeseries properties (e.g. FFT coefficients), but nothing beyond the obvious statistics (first, last, mean, max, min, standard deviation…) made a difference in the model.
- **Recurrent neural networks**: we tried to feed the timeseries directly to LSTM models (long short term memory) but we believe this is not a fruitful approach.

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

After building and exploring timeseries features manually, we discovered a library which automates the process: TSFRESH. Even though it did not uncover new, useful features, we intend to re-use it in the future.

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

Remember that we modeled the **log of the target**. In this space, the official evaluation metric compares the exponentials of the estimate and the target variable.

- Optimizing the MAPE on the log-target directly gave surprisingly good results (even as measured by the official, exponential metric).
- Optimizing for the official metric failed, as it had vanishing gradients due to the presence of the exponential terms.
- As a compromise, we found that optimizing for a quantile objective function (specifically, the 30% quantile) gave the best results.
- Intuitively, the quantile objective function penalized deviations to the right of the target more than deviations to the left. This mimicked the convexity of the exponential error, but being linear, it avoided vanishing gradients.

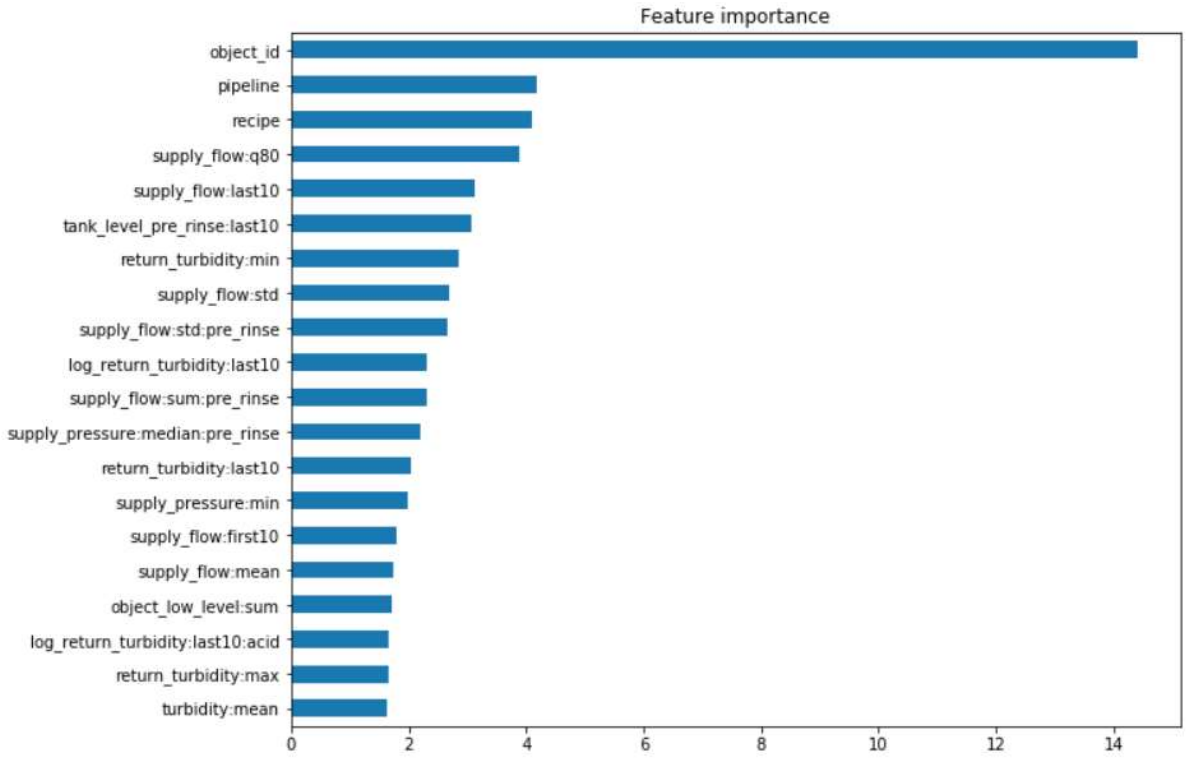
7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

Not that we are aware of. The final model trains on CPU about as fast as on GPU.

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

Top 20 features of our best model

The most important features in general were categorical (metadata). The most important timeseries was the supply flow. (Note on naming: if a feature does not contain the name of a specific phase, it was calculated on the entire timeseries.)

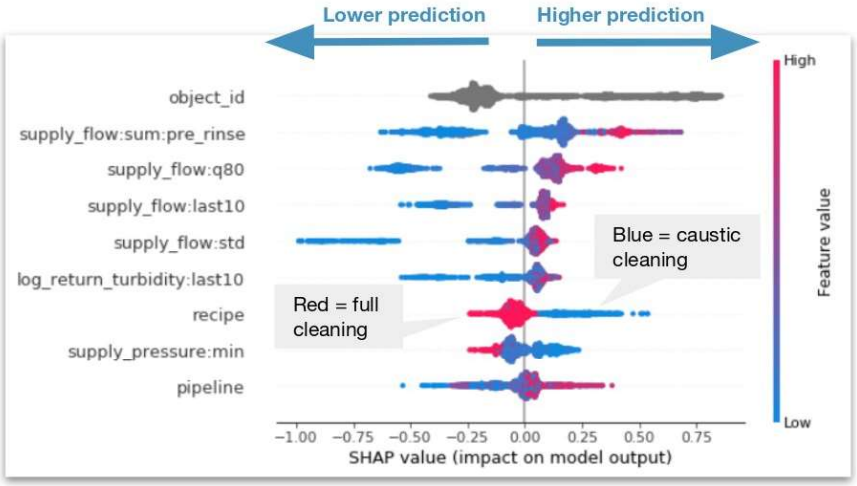


SHAP values

We like SHAP values as an inspection tool for black-box models such as the Gradient Boosted Trees we used. (For more information: A unified approach to interpreting model predictions.)

We found that:

- Object id generally has the strongest impact on the prediction of final total turbidity.
- The caustic recipe (without acid phase, blue dots) contributes to a higher prediction.
- As a rule of thumb, higher pipeline numbers contribute to a higher prediction.
- Generally, higher values of the supply flow contribute to a higher prediction.



9. 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?

   - **Prune the model** down to the minimal number of features that still provide reasonable accuracy (say, no more than 2% worse than the best model). This would improve the robustness of the model at little cost, and make it easier to understand which factors drive final turbidity.

   - Train the model to provide a specific estimate **after each phase** in the cleaning process. As a result, we could:
     - Have more training data (all processes would be truncated at the relevant phase, and included in the training data);
     - Gain a better understanding of how our estimates improves after each phase, as more data from the cleaning process flows in.

   - Additional features:
     - During the competition we requested (and were happy to get!) the **planned recipe** (i.e. sequence of phases). This certainly helped: it was the second or third most important feature of our model.
     - One feature we thought might be useful would be the **duration of the target time-period** in the final rinse (if that is known in advance).
     - Finally, since the object id was the most important feature in our model, having additional **meta-data on the objects** could help refine the model.