Supplementary Information

Active learning enabled reactor characterization for mass transfer in aerobic oxidation reactions

Ajit Vikram,[*] Keith A. Mattern, [*] and Shane T. Grosser

S1. Model Architecture, Input Feature Representation, and Training

The machine learning models used in this study was implemented using open-source packages for Python. Linear Regression and Decision Tree Regression models were designed using the scikit learn package.[1] For Decision Tree models, the maximum depth of the tree was limited to 5 and the minimum number of samples required for the split was kept at 10% of the training dataset size. Linear Tree Models (LTM) that combine linear regression and decision trees were based on Linear-Tree package.[2] For LTM, the maximum tree depth and the minimum number of splits was kept same as for the decision tree models. Additionally, Linear Regression model was used as the base estimator for the split in the LTM models. Neural network-based models were designed using modules from the Keras package.[3] The neural network model was designed with 2 fully connected hidden layers, each with 20 nodes, and using Rectified Linear Units (RELU) as the activation function. Batch size and epochs for the model training was 32 and 1000 respectively. For all the models, mean squared error was used as the loss criteria. For Monet-Carlo Dropout Neural Networks (MC-DNN), a dropout mask with a dropout probability of 0.2 was added to each layer in the hidden layer of the neural network. This mask was applied during both the training epochs and the model inference to avoid overfitting during training and extract model prediction distribution during inference. For MC-DNN model uncertainty calculation, the model was evaluated 200 times, each with a randomized dropout mask and the 95% confidence interval was used to get the distribution of k_La prediction from the ensemble of all NN models evaluated during the Monte-Carlo inference. The general architecture and the code implementation of MC-DNN was similar to the one described in our previous work.[4]

To train the models, the data was split into 80-20 split for training and testing. All hyperparameter tuning for the models was performed and evaluated using the training dataset only. For k-fold cross validation, all possible 5 folds of 80-20 split was prepared from the entire dataset and then the models were evaluated for each fold. The training dataset for k_La prediction was represented by 6 critical input features. These include 5 continuous input features (sparger size, fill volume of the reactor vessel, temperature, agitation rate, VVM: volumetric flow rate of air per reactor volume), and 1 categorical input feature (reactor geometry / scale). Reactor configuration features such as the specific reactor geometry, shape of the agitator blade, number of baffles (like temperature and dissolved oxygen probes) and their relative position in the reactor are all combined in this single categorical feature. One-hot encoding technique is used to represent this categorical feature as a numerical input to the model.

Data transformation for improved linearity and normality: For all models except standard Linear Regression (LR) model, log-transformation was applied to the train-test dataset prior to data normalization. Figure 2b in the manuscript shows that the improved data normality and linearity leads to slight improvement in accuracy on test dataset when log-transformed linear regression model (LL) is compared with the LR models. Figure S1 shows the distribution of all continuous inputs and output to the model with and without the log transformation. Additional transformation techniques like box-cox were also explored but did not yield any significant improvement in terms of normalization of the data and the accuracy of the model.

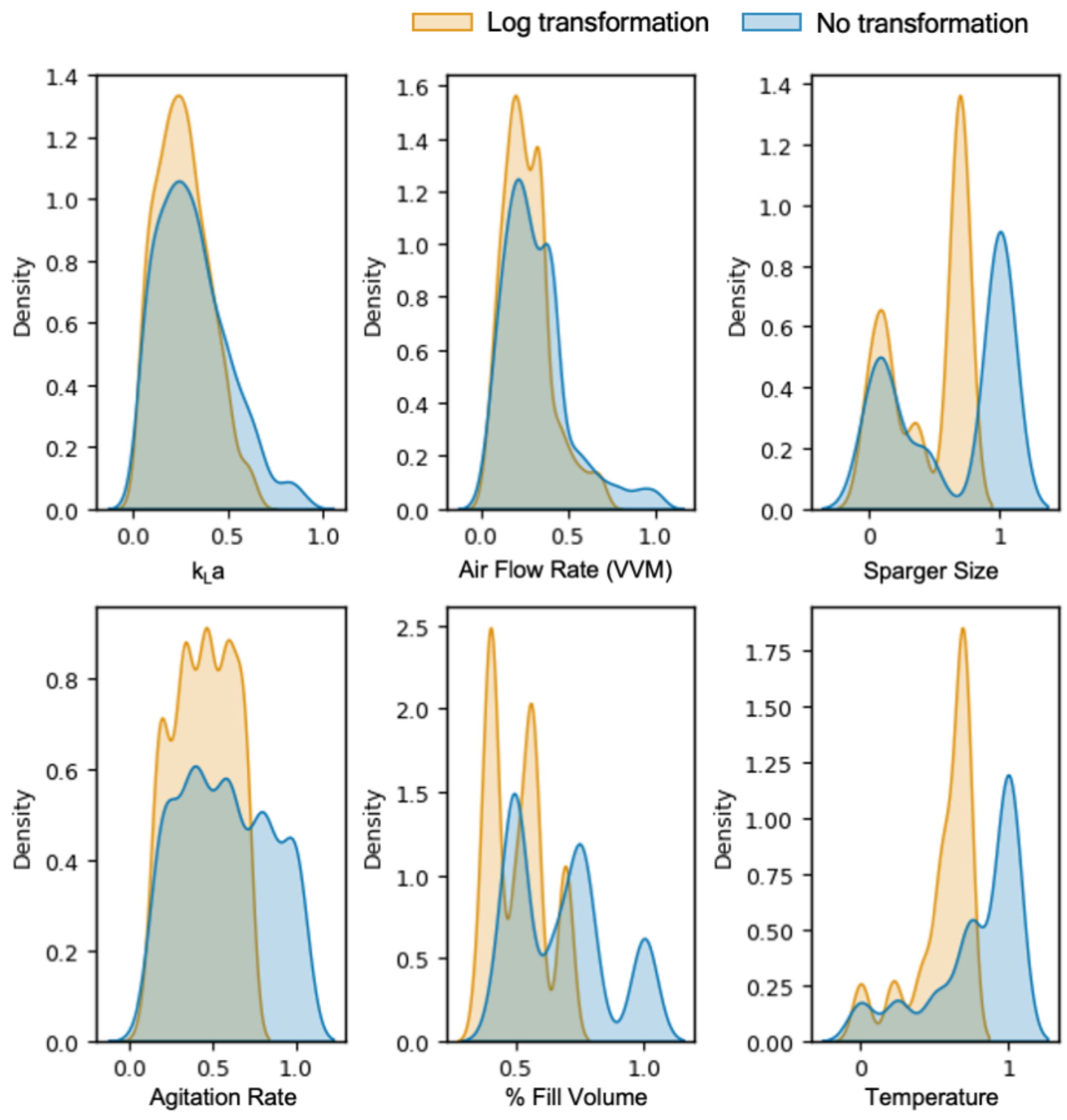
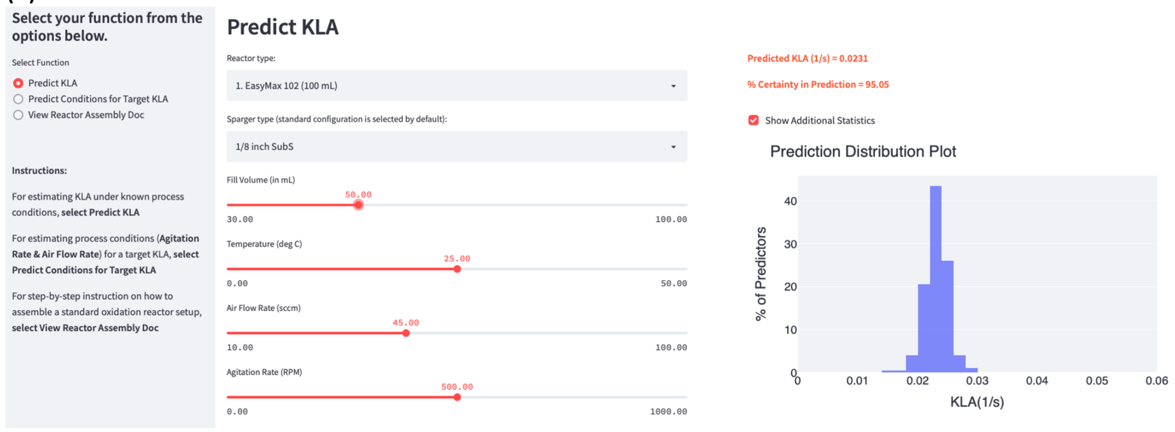


Figure S1. *Density plot showing the distribution of data prior to the transformation and after log transformation of the training dataset. The parameters on the x axis are shown on a normalized scale.*

S2. Web App Deployment of the Machine Learning Model for k_La Prediction

We utilized the Streamlit package[5] to deploy the trained ML model as an interactive web application (Figure S2). Streamlit provides an intuitive open-source framework that simplifies the integration of Python scripts into a user-friendly interface. By leveraging its straightforward and well documented API, features such as data input, model inference, and result visualization can be seamlessly incorporated. The examples for all functionalities and the widgets used in the deployed web app are available with the Streamlit package documentation.[5] Figure S2 shows the screenshot of the two key features of this web app: (i) prediction of k_La along with the uncertainty in the model prediction are provided based on the user input of the process conditions and choice of reactor configurations (Figure S2a); and (ii) identifying the range of operating conditions that can yield a desired k_La value (based on user input) for the process (Figure S2b).

(a)



Select your function from the options below.

Select Function

[✓ Predict KLA]
Predict Conditions for Target KLA
View Reactor Assembly Doc

Instructions:

For estimating KLA under known process conditions, **select Predict KLA**

For estimating process conditions (**Agitation Rate & Air Flow Rate**) for a target KLA, **select Predict Conditions for Target KLA**

For step-by-step instruction on how to assemble a standard oxidation reactor setup, **select View Reactor Assembly Doc**

Predict KLA

Reactor type: 1. EasyMax 102 (100 mL)

Sparger type (standard configuration is selected by default): 1/8 inch SubS

Fill Volume (in mL): 50.00 (30.00 – 100.00)

Temperature (deg C): 25.00 (0.00 – 50.00)

Air Flow Rate (sccm): 45.00 (10.00 – 100.00)

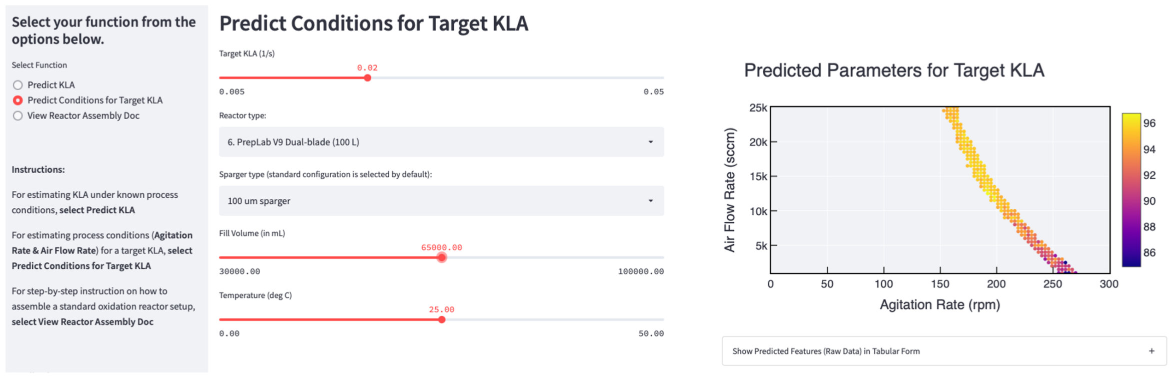
Agitation Rate (RPM): 500.00 (0.00 – 1000.00)

Predicted KLA (1/s) = 0.0231

% Certainty in Prediction = 95.05

[✓ Show Additional Statistics]

(b)



Select your function from the options below.

Select Function

Predict KLA
[✓ Predict Conditions for Target KLA]
View Reactor Assembly Doc

Instructions:

For estimating KLA under known process conditions, **select Predict KLA**

For estimating process conditions (**Agitation Rate & Air Flow Rate**) for a target KLA, **select Predict Conditions for Target KLA**

For step-by-step instruction on how to assemble a standard oxidation reactor setup, **select View Reactor Assembly Doc**

Predict Conditions for Target KLA

Target KLA (1/s): 0.02 (0.005 – 0.05)

Reactor type: 6. PrepLab V9 Dual-blade (100 L)

Sparger type (standard configuration is selected by default): 100 um sparger

Fill Volume (in mL): 65000.00 (30000.00 – 100000.00)

Temperature (deg C): 25.00 (0.00 – 50.00)

Predicted Parameters for Target KLA
Air Flow Rate (sccm)
Agitation Rate (rpm)
5k
10k
15k
20k
25k
0
50
100
150
200
250
300
86
88
90
92
94
96
Show Predicted Features (Raw Data) in Tabular Form

Figure S2. *Screenshot from web app deployment of the* k_La *predictor models for the lab scientists. Two features of how the developed ML model can be used for process development: (a)Prediction of* k_La *under known process conditions, and reactor configurations; and (b) Identifying the range of process conditions that can yield a desired* k_La *for the process.*

S3. SHAP analysis of the model

Shapley Additive explanations (SHAP) was used to interpret the trained ML model in terms of quantifying the impact of each parameter on the predicted k_La value. For the SHAP analysis, we utilized the open-source shap package for python (using the standard shap.Explainer class).[6, 7] To provide a comprehensive evaluation of the model's behavior, we employed Latin Hypercube Sampling (LHS) to generate the input data for SHAP analysis. LHS is a statistical method that ensures space-filling and stratified sampling of the feature space., which helps in capturing a representative distribution of the input features. This approach thus enhances the reliability of the SHAP values by covering the feature space more uniformly. Figure S3 shows the shap values for each input feature for different reactor configurations and scales. These results supplement Figure 4 of the manuscript that only shows shap values across two different reactor configurations. As evident, the importance and impact of different features varies significantly across different reactor configurations / scale, thus highlighting the benefits of ML-based models over the existing empirical models for predicting k_La.

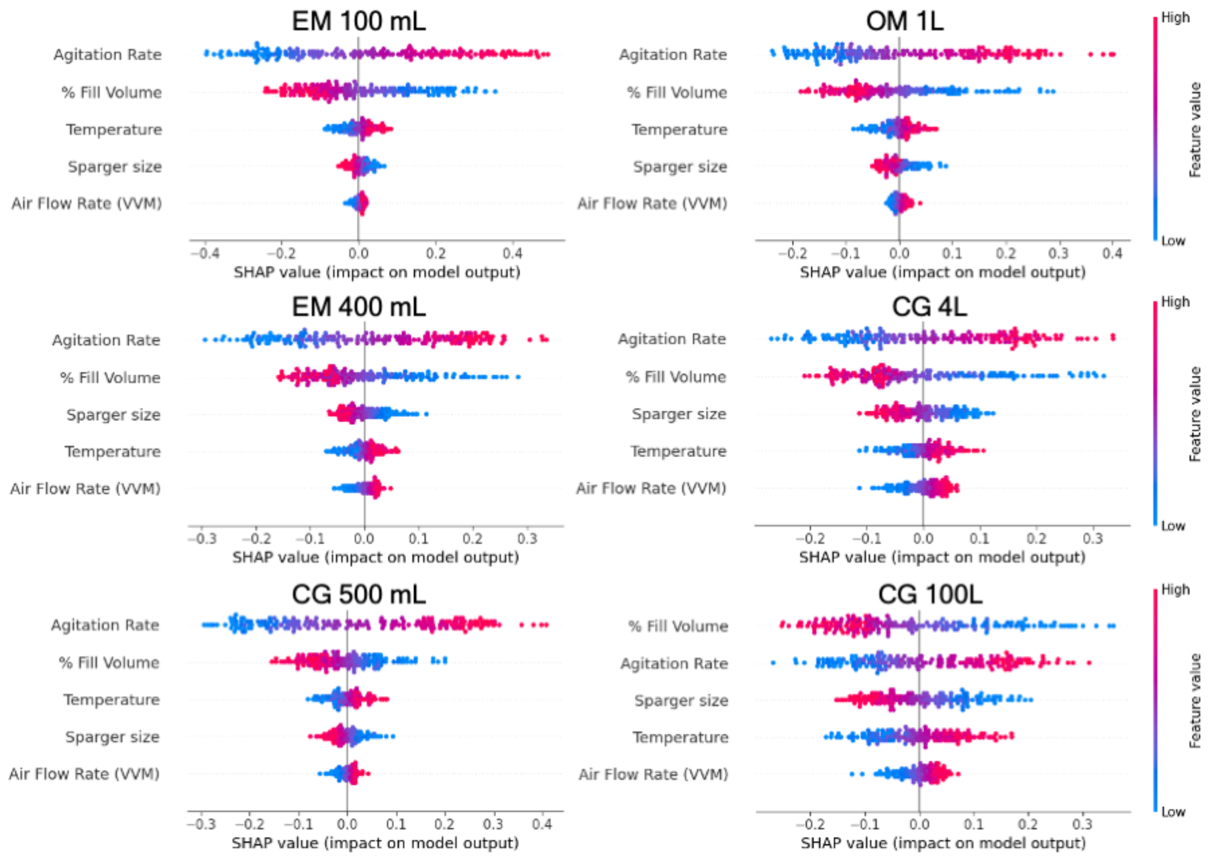


Figure S3. *Shapley-additive explanations (SHAP) values for each parameter to estimate the impact of each input feature on* k_La *prediction across scales ranging from 100 mL lab scale reactor to 100 L kilo scale reactors (EasyMax (EM)100 mL, EasyMax (EM) 400mL, Chemglass (CG) 500 mL, OptiMax (OM) 1L, Chemglass (CG) 4L, Chemglass (CG) 100L). This figure supplements the information provided in Figure 4 of the manuscript.*

Supplementary Information References

1. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *J Mach Learn Res*, 2011, **12**, 2825-2830.
2. M. Cerliani, Linear Tree Model, https://github.com/cerlymarco/linear-tree).
3. Keras API, https://keras.io/api/models/model/).
4. A. Vikram, K. Brudnak, A. Zahid, M. Shim and P. J. A. Kenis, *Nanoscale*, 2021, **13**, 17028-17039.
5. Streamlit, https://docs.streamlit.io).
6. S. Lundberg and S.-I. Lee, *Journal*, 2017, DOI: 10.48550/arXiv.1705.07874, arXiv:1705.07874.
7. SHAP Documentation, https://github.com/shap/shap/tree/master).