An Artificial Neural Network for Simulation of an Upflow Anaerobic Filter Wastewater Treatment Process

Complementary information

Mark McCormick

University of Lausanne
Faculty of Business and Economics (HEC Lausanne)
Department of Information Systems (DESI)
Neuroheuristic research group

June 29, 2022

1 Summary

The complementary information includes the data and the methods used to prepare the datasets. Descriptive statistics describing the raw and the transformed reference data are shown. The process of generating mock data is described in detail. The polynomial, multi-layer perceptron and long short-term memory models are described in detail. Additional results that support method development and the conclusions are reported.

2 Material and Methods

Table S1 shows 21 sets of values of influent and effluent total solids (respectively TS_i and TS_e in g.L^{-1}) and calorific values (respectively CV_i and CV_e in kJ.g^{-1}). The calorific values of 11 data points that were derived from measured total solids are shown in bold text. The calorific values of the remaining 10 data points were derived from the measured TS_i and TS_e. These values were used to calculate the sample calorific value reduction (ΔCV) in kJ.L^{-1} as follows:

$$\Delta CV = (TS_i \times CV_i) - (TS_e \times CV_e) \tag{S1}$$

In addition to ΔCV, the fixed bed temperature T (in °C) and the influent flow rate Q (in L.day^{-1}), are also indicated in Table S1.

As described in the article text, these data were used to make the regression equation.

$$\widehat{\Delta CV} = a \cdot k_{\mathrm{B}} \cdot \frac{T}{Q} + C \tag{S2}$$

where k_{B} is the Boltzmann constant ($k_{\mathrm{B}} = 1.380649 \times 10^{-23}$ J.K^{-1}), a is a constant in days^{-1}, and C is a constant in J.L^{-1}. This equation can also be rewritten as:

$$\widehat{\Delta CV} = b \cdot \frac{T}{Q} + C \tag{S3}$$

Given the 21 points observed, we compute a linear regression of Equation S3 (package SciPy optimize curve fit) with $\widehat{\Delta CV}$ expressed in J.L^{-1}

The result yields a slope $b = 65.48$ and an intercept $C = 178.39$ Then, knowing temperature T and influent flow rate Q it is possible to estimate $\widehat{\Delta CV}$ with the equation:

$$\widehat{\Delta CV} = 65.48 \cdot \frac{T}{Q} + 178.39 \tag{S4}$$

A noise vector was generated to add supposed effects that Equation S4 does not account for. The noise vector was obtained by adding vectors obtained from the mean (μ) and standard deviation (σ) of the ratio of measured temperature T and influent flow rate Q according to Equation S5 :

$$noise = 0.5 \cdot \mu \cdot \frac{T}{Q} + 1.2 \cdot \sigma \tag{S5}$$

The noise vector was then added to the vector obtained from taking the ratio of the measured temperature T and influent flow rate Q. Figure S2 shows the noise vector. Equation S6 was then used to generate the surrogate data points.

$$\widehat{\Delta CV} = 65.48 \cdot (\frac{T}{Q} + noise) + 178.39 \tag{S6}$$

The regression equation was used to generate surrogate data from measured temperature and influent flow rates. Thus, the resulting series of 170 data samples of calorific value reduction — referred to as the *reference dataset* — included direct and supplemental surrogate data.

Table S1: Daily average measurements derived from the experimental UAF bioreactor of influent (TS_i) and effluent (TS_e) total solids, influent (CV_i) and effluent (CV_e) calorific values, fixed bed temperature (T) and influent flow rate (Q). The daily calorific value reduction (ΔCV) was derived following Equation S1.

Day	TS_i (kg.L^{-1})	CV_i (kJ.kg^{-1})	TS_e (kg.L^{-1})	CV_e (kJ.kg^{-1})	ΔCV (kJ.L^{-1})	T (°C)	Q (L.day^{-1})
17-MAY-2017	0.07	5.88	0.06	1.78	0.3048	22.6	142.6
24-MAY-2017	0.09	5.34	0.06	1.99	0.3612	24.0	151.2
31-MAY-2017	0.08	5.25	0.06	1.79	0.3126	22.4	144.0
07-JUN-2017	0.06	5.92	0.05	1.90	0.2661	19.2	196.4
14-JUN-2017	0.08	**7.41**	0.06	**2.74**	0.4284	29.1	223.2
21-JUN-2017	0.08	5.62	0.05	1.72	0.3636	35.9	144.0
28-JUN-2017	0.05	**5.10**	0.03	**2.67**	0.1749	23.9	220.2
05-JUL-2017	0.05	5.55	0.05	1.84	0.1966	27.4	216.0
12-JUL-2017	0.06	5.68	0.05	1.85	0.2483	23.1	194.4
19-JUL-2017	0.06	**5.68**	0.05	**2.15**	0.2333	30.4	165.6
26-JUL-2017	0.03	**6.06**	0.05	**1.65**	0.0993	23.1	316.8
02-AUG-2017	0.03	5.88	0.04	1.86	0.1020	32.1	309.6
09-AUG-2017	0.06	**5.86**	0.05	**1.57**	0.2731	23.1	288.0
16-AUG-2017	0.07	**5.68**	0.06	**0.88**	0.3448	23.9	312.6
23-AUG-2017	0.06	5.64	0.05	1.86	0.2454	22.3	367.2
20-SEP-2017	0.06	5.76	0.04	1.71	0.2772	23.1	230.2
27-SEP-2017	0.08	**4.20**	0.06	**1.24**	0.2616	20.6	385.2
18-OCT-2017	0.05	5.62	0.04	1.69	0.2303	15.8	408.0
25-OCT-2017	0.07	**6.10**	0.05	**2.36**	0.3202	14.3	309.6
12-NOV-2017	0.07	6.50	0.06	3.50	0.2385	14.6	316.0
26-NOV-2017	0.08	**6.50**	0.07	**4.30**	0.2405	12.0	448.0

As shown in Figure S1, the CV reduction increased with the ratio of influent temperature to flow.

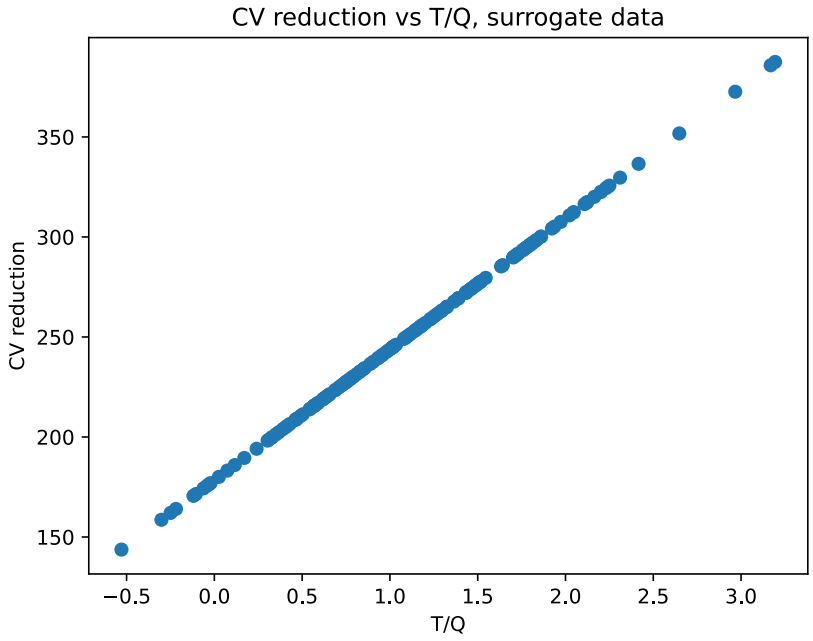

Figure S1: **Ratio of CV reduction versus temperature/influent flow**

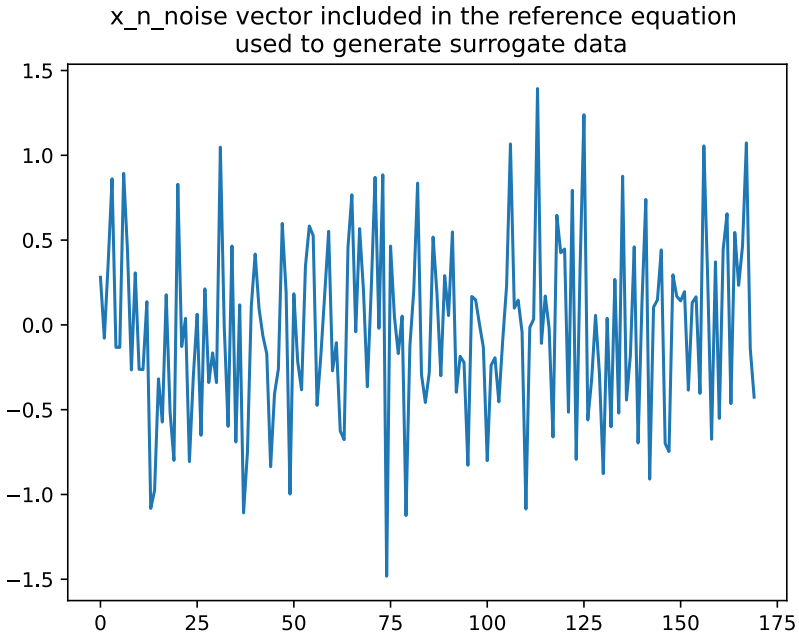

Figure S2: **Noise vector included in the regression equation used to generate surrogate data.**

Mock data generation

2.0.1 Packing equivalent spherical diameter (ESD)

The selected mock packing equivalent spherical diameters $ESD = [4, 12, 36]$ with the middle level the same as the the ESD of the experimental bioreactor and the other values 3 times smaller than and 3 times larger than the experimental value. Larger packaging material leads to an increase in void space between individual pieces of packing material. An increase in the effectiveness of the treatment, expressed in terms of biogas production, results from better mixing that is achieved when the void space volume increases [4]. To include the effect of large void space, the coefficient increases exponentially with the influent flow rate when the value of ESD is 36 mm. In a study of a bioreactor containing 3.5 m tubular packing, the efficiency of the biological reactions was limited by substrate transfer that is very sensitive to substrate loading and methane flow rate [6]. It is also known that inadequate mixing of the sludge bed occurs at very low flow rates, and that very high flow rates lead to shear and biomass loss [8, 6]. In the dataset from the mock experiments, these effects were created using coefficients that included the ratio of the influent flow rate to the maximum influent flow rate and by setting a threshold value below which effectiveness is reduced due to very poor mixing.

2.0.2 Packing material type (MAT)

Based on the relative measured activity of biofilms on samples of polyvinyl chloride (PVC) chips, torrefied wood chips (TWD), and polyurethane foam (PUF) taken from the experimental bioreactor, the coefficients for the factor MAT were set to respectively $c_{MAT(PVC)} = 1.0$, $c_{MAT(TWD)} = 2.7$, and $c_{MAT(PUF)} = 11.0$. Since biogas production improves mixing [4], a coeficient increased effectiveness when the material type was polyurethane foam. Polyurethane foam packing effectiveness is reduced at low flow rates due to biomass clogging the interstial space. PVC packing effectiveness is reduced at high flow rates due to biomass shear loss. Consequently during mock experiments the value of $c_{MAT(PVC)}$ was further reduced when the influent flow rate was greater than 80% of the maximum.

2.0.3 Fixed bed height to diameter ratio (HDR)

Since the optimal HDR was not known, three levels —$HDR = [0.5, 1.8, 4.0]$— were selected arbitrarily with the middle level the same as the the HDR of the experimental bioreactor. It has been reported that most of the organic substances are consumed in the bottom part of the bioreactor [2]. This implies that the bed is too long when the organic load is low and that a well mixed long bed has the highest capacity for CV reduction (ΔCV). Consequently, the value of the coeficient was made to increase with the flow rate to an upper limit that was proportional to the bioreactor height to diameter ratio.

As described in the article, the predictors were set according to an experimental plan that was based on the Taguchi L9 design with the 4th factor (predictor) set to the experimental influent flow rate.

The 4 predictors were:

- Equivalent spherical diameter (ESD)
- Material type (MAT)
- Height to diameter ratio (HDR)
- Influent flow rate (Qin)

Figure S3 below shows the values of the predictors used in the 9 mock experiments in sequence.

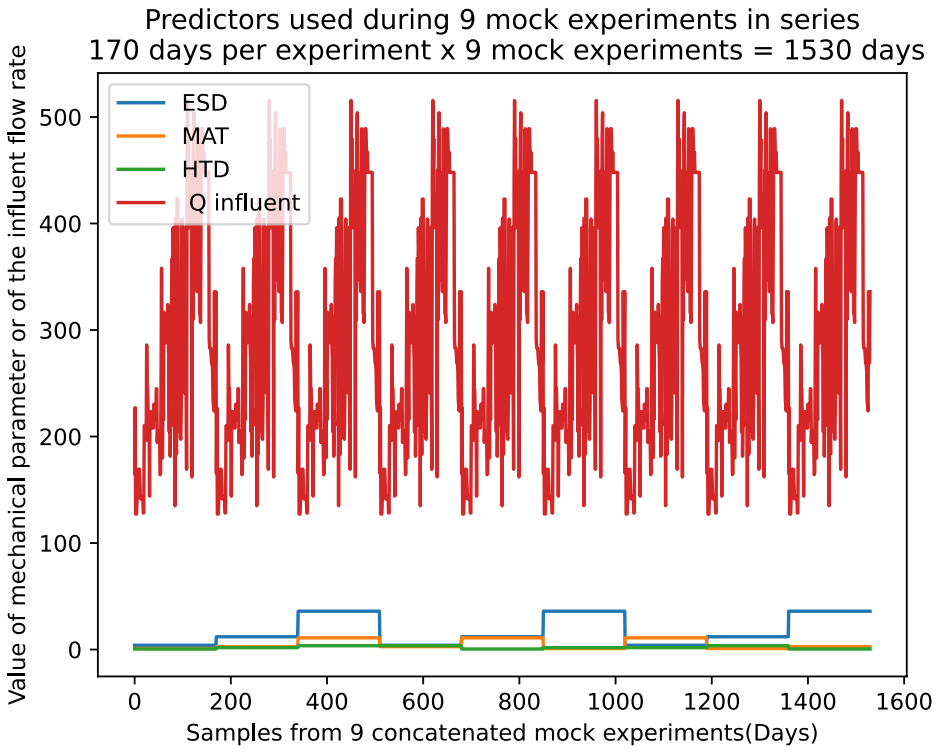

Figure S3: **Value of the predictors. 9 mock experiments in sequence**
Predictors: Sphere diameter (ESD), Material type (MAT), Height to diameter ratio (HDR), Influent flow rate (Q influent).

An *ad hoc* function was used to generate the calorific value reduction from the sum of the ESD, MAT and HDR components. The daily calorific value reduction was obtained from the product of the influent flow rate and the predictor effects. The *ad hoc* function is shown in the article and below.

$$CV_{red} = \sum_{i=1}^{170} C_i(S_i * Q_i + M_i * Q_i + H_i * Q_i)$$

where,

CV_{red} = a vector of calorific value reductions for a single mock experiment

Q_i = the influent flow rate on the ith day

C_i = the reference change in the calorific value of the influent stream on the ith day

S_i = the spherical diameter effects on calorific value reduction on the ith day

M_i = the material type effects on calorific value reduction on the ith day

H_i = the height to diameter effects on calorific value reduction on the ith day

The following Python code was used to generate mock data using the *ad hoc* function.

```
    S=[] # ESD component of total CV reduction
M=[] # MAT component of total CV reduction
H=[] # HDR component of total CV reduction
Qmax = np.max(IFR_ref_all) #516 l/day Maximum infuent flow rate
i=0
h=0
SF = 1550 # Scale factor to adjust the value predicted by each of the 3 features.
SC = 1 # sphere diameter adjustment factor
SC4 = 0.4 # Threshold below which the CVred increases exponentially in proportion to flow rate when Q
is below SC4*Qmax and SD = 4 because small SD is well adapted to low flow rates.
# Predictors. 9 experiments with values from the range described in the MDPI article text.
exper_lit = [(4.0, 1.0, 0.5), (12.0, 2.7, 1.8), (36.0, 11.0, 3.6), (4.0, 2.7, 3.6), (12.0, 11.0, 0.5), (36.0, 1.0, 1.8),
(4.0, 11.0, 1.8), (12.0, 1.0, 3.6), (36.0, 2.7, 0.5)]
for h in range(0,len(exper_lit)): # Mock experiments kjl x l/day = kJ/day
for i in range(len(IFR_1)):

    S.append(np.multiply((CV_ref_kjl[i]),( # aprox 0 - 0,5 kjl
1+
np.where(exper_lit[h][0]==12.0, SF*0.33*IFR_1[i]/Qmax, 0)+
np.where(exper_lit[h][0]==36.0 and IFR_1[i] > 0.8*Qmax , SF*0.1, 0)+
np.where(exper_lit[h][0]==36.0 and IFR_1[i] > SC4*Qmax , 0.8*SF*0.33*(1-(-IFR_1[i]/Qmax)), 0)+
np.where(exper_lit[h][0]==36.0 and IFR_1[i] < SC4*Qmax , 0.1*SF*0.33*np.exp(-IFR_1[i]/Qmax), 0) +
np.where(exper_lit[h][0]==36.0 and IFR_1[i] < 0.45*Qmax , -SF*0.05, 0) +
np.where(exper_lit[h][0]==4.0 and IFR_1[i] < SC4*Qmax , 0.4*SF*0.33*np.exp(-IFR_1[i]/Qmax), 0)+
np.where(exper_lit[h][0]==4.0 and IFR_1[i] < 0.8*Qmax , 2*(SF*4/12)*0.33*IFR_1[i]/Qmax, 0)+
np.where(exper_lit[h][0]==4.0 and IFR_1[i] > 0.8*Qmax , 0.01*SF*0.33*(1-np.exp(IFR_1[i]/Qmax)),0)+
np.where(exper_lit[h][0]==4.0 and IFR_1[i] < 0.45*Qmax , -SF*0.01, 0)+
0)*SC)*1)

    M.append(np.multiply((CV_ref_kjl[i]),(
1+
np.where(exper_lit[h][1]==2.7, SF*0.33*IFR_1[i]/Qmax, 0) +
np.where(exper_lit[h][1]==11.0 and IFR_1[i]> 0.8*Qmax, SF*0.2, 0)+
np.where(exper_lit[h][1]==11.0 and IFR_1[i]> 0.3*Qmax, 0.3*SF*0.33*np.sin((3.14/2)*(IFR_1[i]/Qmax)), 0)
+
np.where(exper_lit[h][1]==11.0 and IFR_1[i]< 0.3*Qmax, -0.1*SF*0.33*np.exp(-IFR_1[i]/Qmax), 0)+
np.where(exper_lit[h][1]==11.0 and IFR_1[i]< 0.3*Qmax, -0.05*SF*0.33, 0)+
np.where(exper_lit[h][1]==1.0 and IFR_1[i]< SC4*Qmax, 0.5*SF*0.33*np.exp(-IFR_1[i]/Qmax), 0)+
np.where(exper_lit[h][1]==1.0 and IFR_1[i]< 0.8*Qmax, 11*(SF*1/11)*0.33*IFR_1[i]/Qmax, 0)+
np.where(exper_lit[h][1]==1.0 and IFR_1[i]> 0.8*Qmax, 0.03*SF*0.33*(-np.exp(-IFR_1[i]/Qmax)),0)+
0)*SC)*1)

    H.append(np.multiply((CV_ref_kjl[i]), (
1+
np.where(exper_lit[h][2]==1.8, SF*0.33*IFR_1[i]/Qmax, 0) +
np.where(exper_lit[h][2] == 0.5 and IFR_3[i]<Qmax*0.5/3.6, SF*0.33*IFR_3[i]/Qmax, 0)+
np.where(exper_lit[h][2] == 0.5 and IFR_3[i]>0.5/3.6, 0.4*SF*0.33*IFR_3[i]/Qmax, 0)+
np.where(exper_lit[h][2] == 3.6 and IFR_3[i]< Qmax*1.8/3.6, SF*0.33*IFR_3[i]/Qmax, 0)+
np.where(exper_lit[h][2] == 3.6 and IFR_3[i]>Qmax*1.8/3.6, 0.8*SF*0.33*IFR_3[i]/Qmax, 0) +
0)*SC)*1)

    IFR_Mtest = np.concatenate((np.asarray(S), np.asarray(M), np.asarray(H)), axis=1).T
```

The contribution of each predictor and the total CV reduction on each day are shown separately in the

figure S4 below.

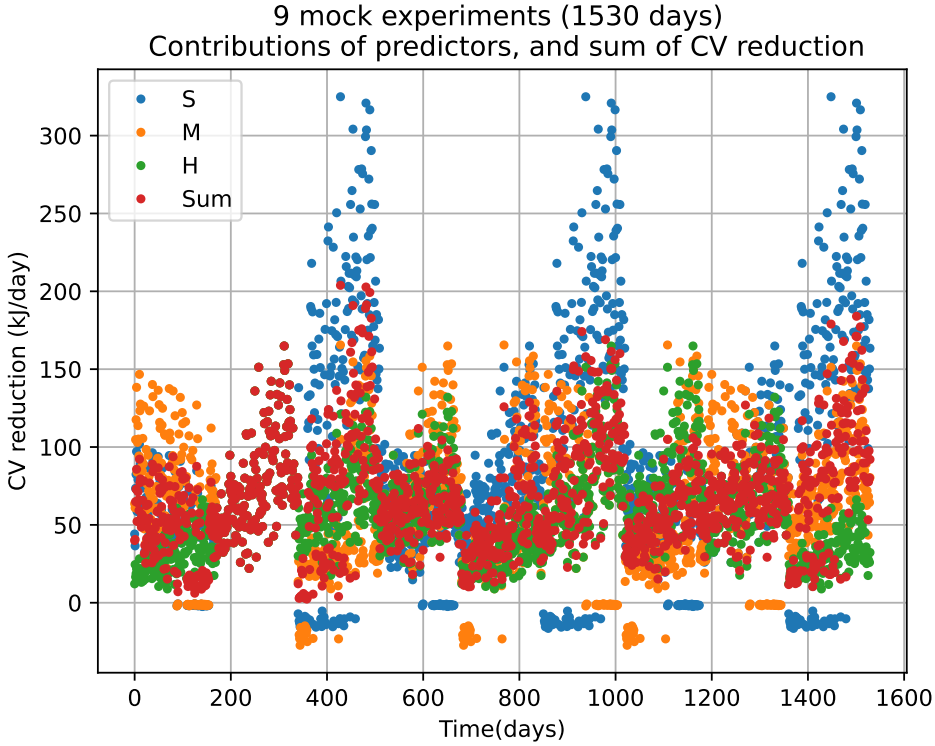

Figure S4: **Calorific value reduction (ΔCV).** Components: Sphere diameter (S), Material type (M), Height to diameter ratio (H), Sum of the component contributions (Sum)

The histogram below (Figure S5) shows the range of calorific value reductions and the difference between the 9 mock experiments.

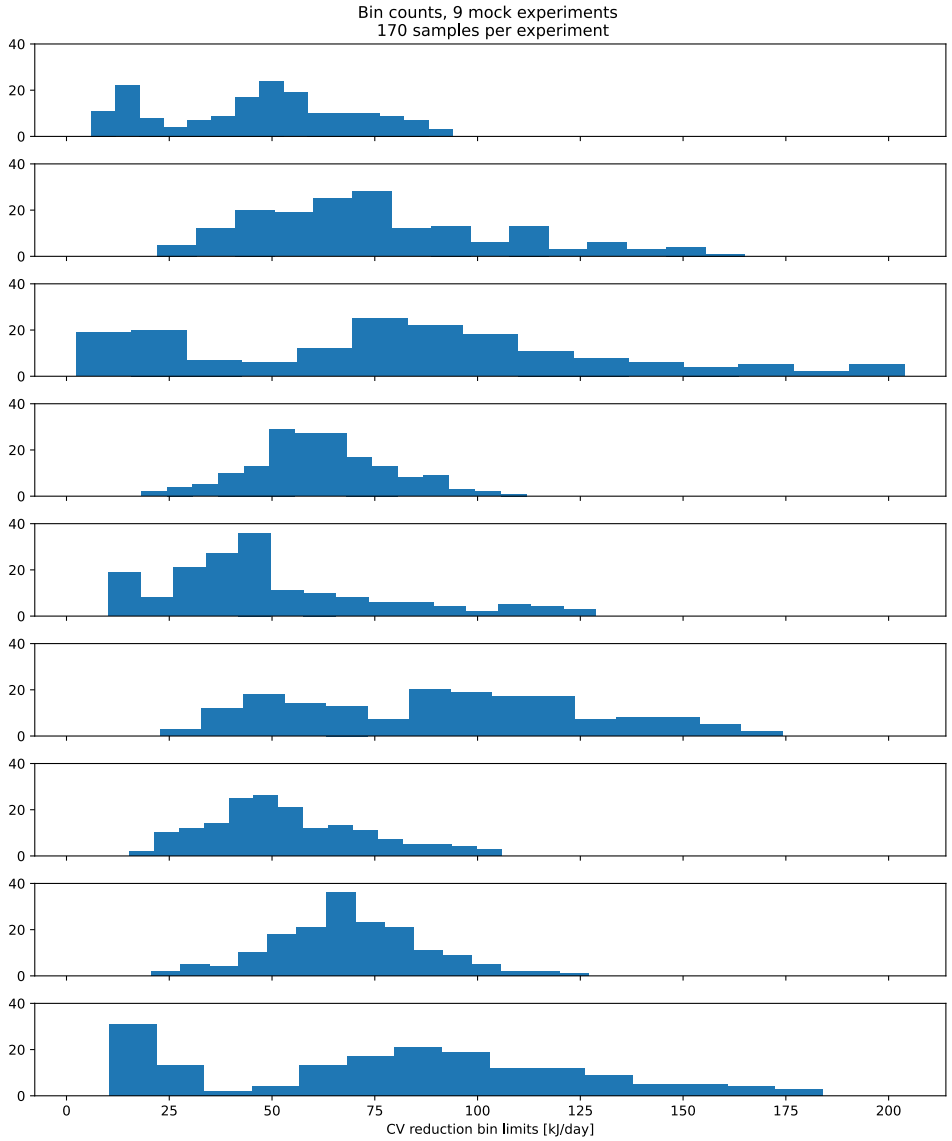

Figure S5: **Binned calorific value reduction (ΔCV).** 9 experiments, 170 sample values per experiment. Experiment numbers 1 - 9 (from top to bottom)

For the purpose of comparaison, response data was generated from the predictors used in the experiment using the mock response data generation function. The predictors were: ESD=12, MAT=2.7, HDR=1.8

with influent flow the same as the experiment. Figure S6 shows a plot of experimentally aquired data verses mock data.

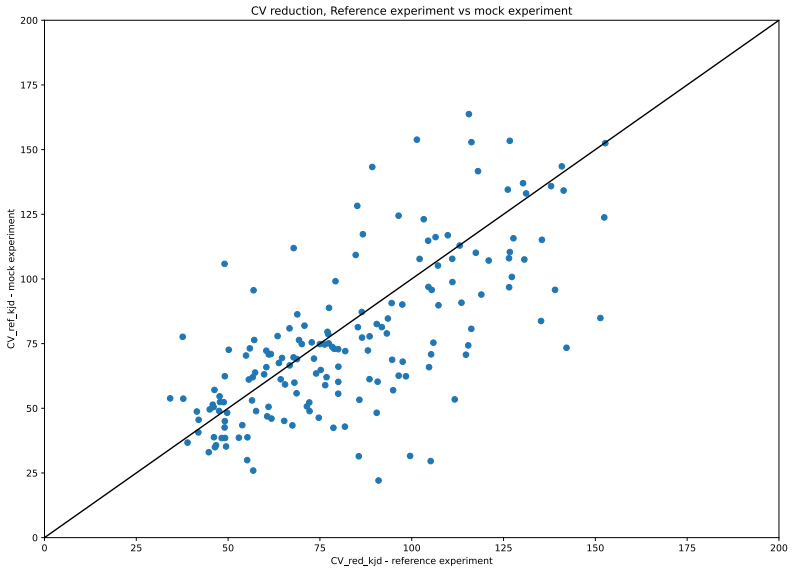

Figure S6: **CV reduction, reference experiment versus data generated using the mock response data generation function.**

2.0.4 Transformation of the response data

Different transformations of the response data were evaluated to obtain a more normally shaped distribution of the response data. The Yeo-Johnson transformation was found to yield the distribution that was most like a normal distribution. The histograms of the raw and the transformed response data are shown in Figure S7. The cumulative CVreduction for the different transformations is shown in Figure S8.

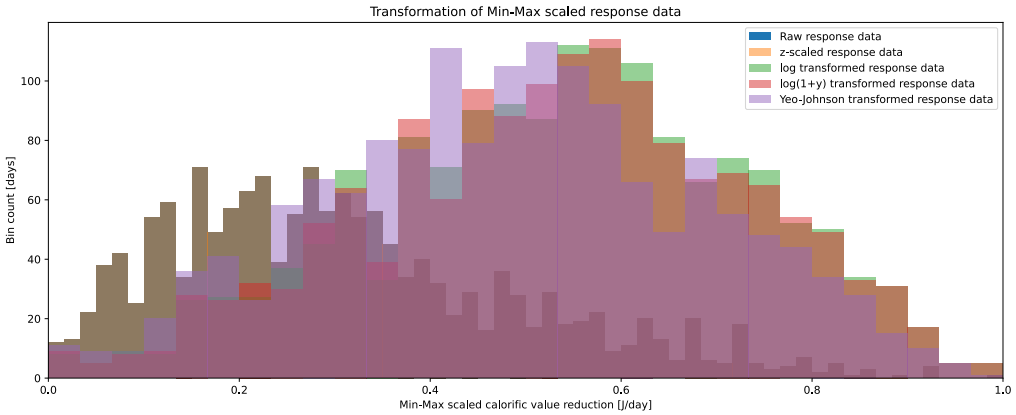

Figure S7: **Mock data transformation** Distribution of calorific value reductions 9 mock experiments, 1530 data points

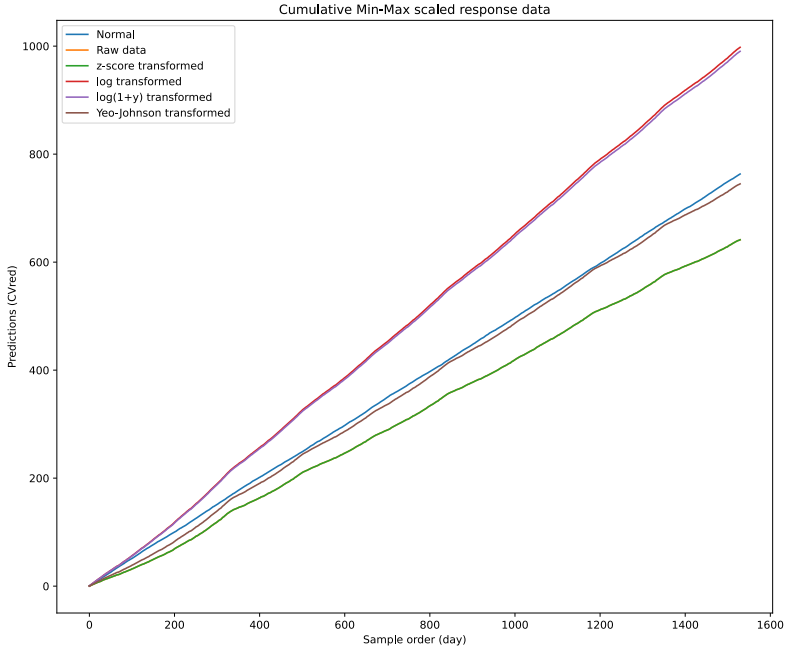

Figure S8: **Mock data transformation** Cumulative CV reduction, 9 mock experiments, 1530 data points

2.0.5 Analysis of time dependency

Autocorrelation evaluates the similarity between current and lagged data points in a time series dataset. The Durbin-Watson test was performed to determine if there is autocorrelation in the series of daily calorific value reduction. The null hypothesis in no autocorrelation. The value of the calculated test statistic lies between 0 and 4. A value of 2 indicates no autocorrelation. Values below 2 indicate positive autocorrelation. Values above 2 indicate negative autocorrelation. The calculated D-W test statistic was 0.146. The maximum p-value of the correlation coefficient of each time lag at the 95% level of confidence was 7.91×10^{-10}. Since the D-W test statistic value is less than 2, the time series of daily calorific value reduction is positively autocorrelated The autocorrelation function was also calculated for 100 days.

Using the statsmodels library for python [7] autocorrelation was analyzed for each day up to 100 days. The results are shown in Figure S9. The autocorrelation of the current day is always 1. The results located in the shaded area are significant (alpha = 0.05). The autocorrelation coefficient falls to and remains less than 0.5 after a 1-day lag and there is a small positive or negative autocorrelation of the time series data after day 16.

The low VIF values shown in Table S2 indicate that the predictors are not correlated and consequently none of the coefficients are redundant and all the coefficients can be used in the polynomial regression model. Additionally, this result shows the independence of the predictors and their suitability for use in the dataset used to train and validate the ANN models.

The observation that the response data are autocorrelated implies that the data might have time dependent features. Consequently, we decided to further study time series features by using an LSTM model and by assessing the effect of data shuffling prior to training. Additionally to further evaluate time dependency, separate datasets were built using both shuffled and unshuffled data for training and testing. If time series features are important, then then the LSTM model is expected to lead to more accurate predictions than the polynomial and the MLP models.

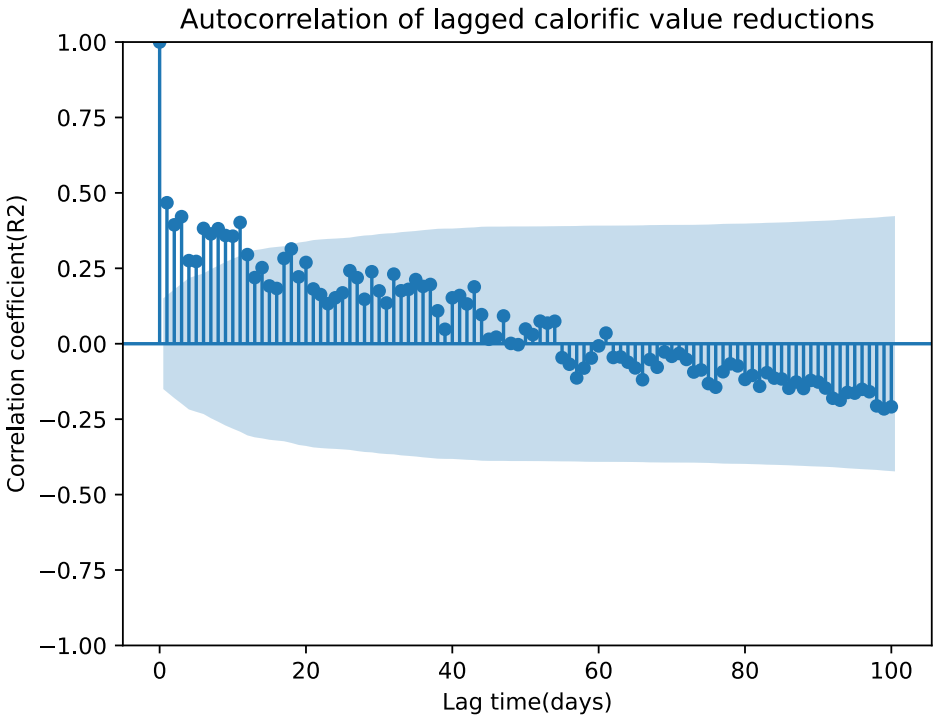

Figure S9: **Autocorrelation of lagged calorific value reduction** The lags are plotted on the horizontal and the correlations are plotted on the vertical axis.

Table S2: Variance Inflation Factors (VIF) for each predictor. ESD: equivalent spherical diameter; MAT: packing material; HDR: bed height/diameter ratio; Q: Influent flow rate.

	VIF
ESD	2.31
MAT	2.05
HDR	2.80
Q	3.85

Polynomial model

The polynomial model was made using the sklearn linear_model package. The leave-one-out methout was used to randomly remove a set of predictors and responses during each replication The equation is shown below where Wr is the vector of model coefficients and Ir is the y-intercept.

yhat_poly = X_test_mm[k,0]*Wr[0] + X_test_mm[k,1]*Wr[1] + X_test_mm[k,2]*Wr[2] + X_test_mm[k,3]*Wr[3]+Ir

MLP model

Each hidden layer unit input was propagated to every unit of the subsequent hidden layer dense units using the following propagation rule:.

$$s_k^p = \sum_{i=1}^{L} w_{ik} y_i^p + b_k$$

where s_k^p is the vector propagated to unit k of the subsequent layer, L is the number of hidden layer units, w_{ik} is the value of the weight of the connection between unit i of the current layer and unit k of the subsequent layer, y_i^p is the output of the unit, and b_k is the value of the bias of this connection [1].

The error during training was calculated with the Mean Squared Error (MSE) loss function.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Where N, is the training batch size, y_i is the true value and $\hat{y}_i$ is the predicted value. The architecture of the MLP ANN that was used for subsequent simulations is shown in Figure S10 below. Note: The final model had 4 hidden layers with 256 units per hidden layer.

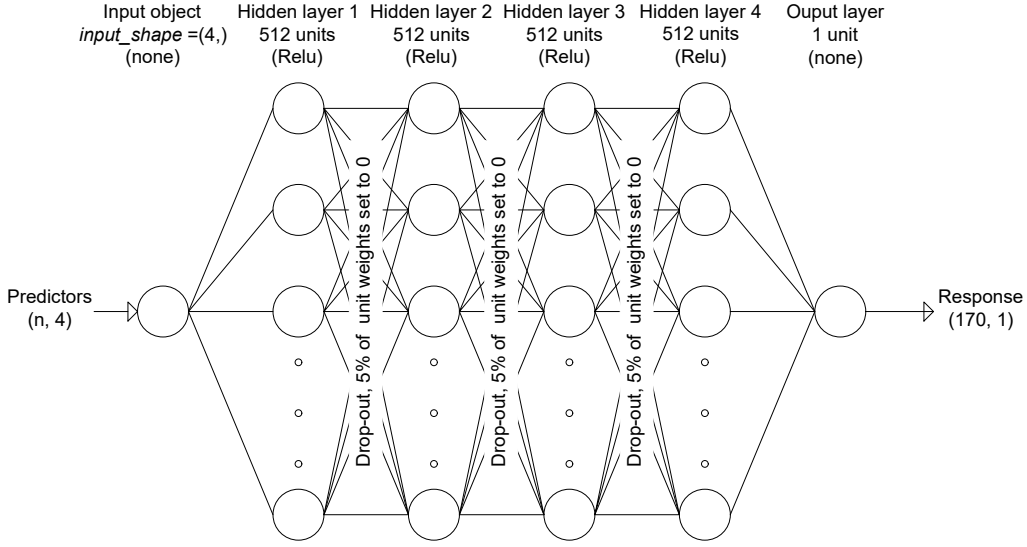

Figure S10: **Architecture of the x-layer MLP used for simulation**

Choosing the number of MLP hidden layers and units

To determine the number of hidden layers and hidden layer units, all possible MLP architectures having from 1 to 13 hidden layers and from 8 to 4096 units per hidden layer were built, trained and evaluated. The

prediction accuracy of the validation tests was evaluated in terms of the coefficient of determination (R^2) and the slope of the regression line. A value of 1 of both the coefficient of determination (R^2) and the slope indicates the best fit of the predicted to the true values. Consequently, a high value of the sum of R^2 and the slope indicates high accuracy. The results are shown in Figure S11 and in Figure S12. Model building, training and evaluation was repeated at least 10 times each with random initialisation of the model weights. We observed that the models having between 64 and 2048 units per layer had the highest accuracy and that using between 2 and 6 hidden layers yields similar results.

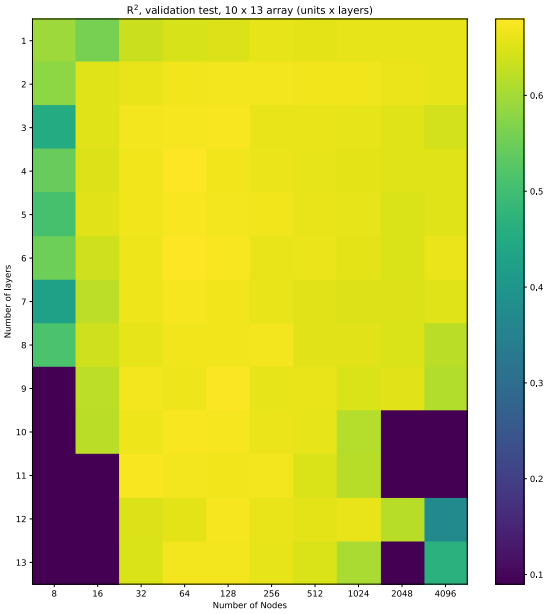

Figure S11: **Validation test R-squared for different numbers of hidden layers and numbers of units per hidden layer**

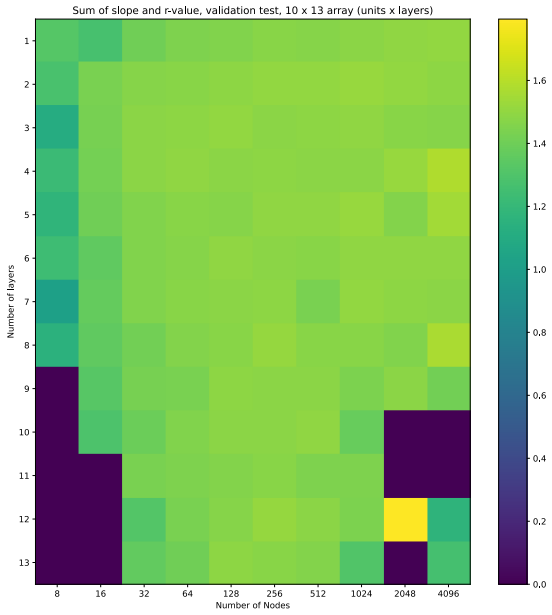

Figure S12: **Validation test sum of R-squared and slope of the regression line for different numbers of hidden layers and numbers of units per hidden layer**

LSTM model

In addition to the forward and backward propagation of errors that is a characteristic of all neural networks, recurrent neural networks (RNN) have feedback connections between neurons. Having feedback connections makes it possible to model persitant relations between data objects in a time series thereby giving RNNs the capacity for memory. The LSTM neural network is an improved class of RNN that has memory cells that feature input and output gates that protect the memory cell from perturbations due to, respectively, currently irrelevant inputs and memory cell contents [3]. Neural network architectures containing LSTM cells have demonstrated excellent ability for use in predictive models of time series data sets. The LSTM model was built using Python 3.7.6 and the Keras 2.3.0 API running on top of the TensorFlow 2.0 library for machine learning. The sequential model architecture was used. The model input was defined as a 3D tensor with shape [batch size, timestep, features] where *batch_size* refers to the number of days input to the model before updating the LSTM memory cell weights, *timestep* is the number of equal length sequences of data making up a single batch, and *features* refers to the 4 predictors (ESD, MAT, HDR, Q). One epoch was defined as a passage of all 1530 samples (days). The input tensor was fed directly to an LSTM layer comprised of 128 units. Following the Keras recommendations, the LSTM layer unit used the *tanh* activation function and the *sigmoid* recurrent activation function. The memory cell layer was updated between mini-batches of length *batch_size* as defined by the input shape definition. The *stateful* argument was set to *True* so that the layer weights were used as initial weights in the subsequent mini-batch. The *return_sequences* argument was set to *False* because the model contained a single hidden layer.The remaining hyperparameter arguments were set to default values. The output layer had a single *Dense* unit without activation.

Since LSTM layers are capable of using features that vary over long time periods, we tested input *batch_size* arguments between 1 and 170 days during training. The compiled regression models were trained using a mini-batch gradient descent back propagation learning algorithm. The error during training was

calculated with the Mean Squared Error (MSE) loss function. For the purpose of comparison, separate models were built using both shuffled and unshuffled data for training and testing.

I found that using unshuffled time-series data and a memory cell training `time_step` of 10 days yielded the best results.

The input and output gate activation functions adjust independently of the real rate and values of error information flow through the memory cell. Memory cell learning is the result of automatic scaling of the values by the input and output gates during model training. Having gates makes it possible to superimpose, onto the real-time error flow, selected errors from previous time steps that might be important to accurately modelling the time series data. An update layer combines the input and output gate vector values with the cell state. These functions make it possible to account for time series features of the data such as long time lags.

The LSTM cell is described in the figure S13 and the text below adapted from Olah ([5]).

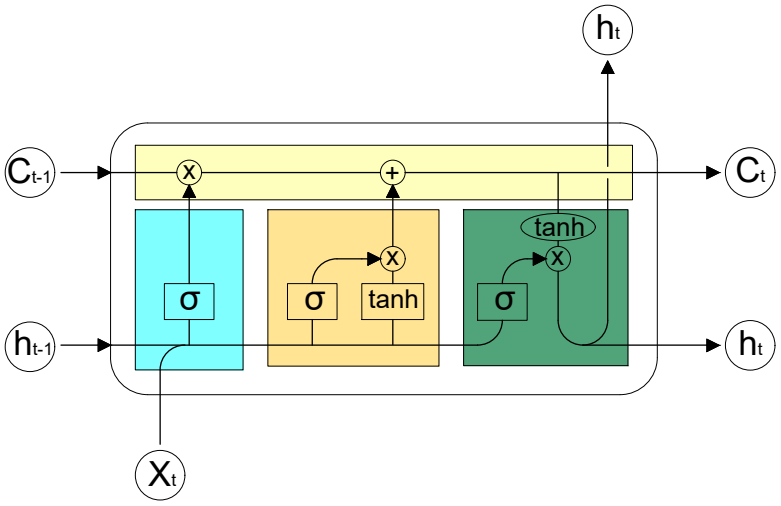

Figure S13: LSTM memory cell

The current data input and hidden state vectors are represented by X_t and h_t respectively. The current LSTM cell input vector length is a fraction of the sample length as specified in the model input shape definition. The cell state, also known as the constant error carousel (CEC), is represented by the yellow box. The automatic modifications to the cell state control the information contained in the output vectors. The cell state and hidden state vectors of the previous step are represented by C_{t-1} and h_{t-1} respectively.

The forget gate layer, represented by the blue shaded area, decides what information to keep from the previous cell state. This is done by multiplying each value of the cell state by the value returned by the *sigmoid* function of the updated weights and biases as follows:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

The update layer, represented by the orange shaded area, combines a list of values to update (i_t) with the vector of new cell state values ($\widetilde{C}_t$) determined using the *tanh* function of the updated weights and biases

as follows:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = tanh(Wc * [h_{t-1}, x_t] + b_c)$$

The output layer, represented by the green shaded area, combines a list of values to update (o_t) with the *tanh* function of the vector of the new cell state values ($\widetilde{C}_t$) as follows:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

The vector of new cell state values (C_t) is determined as follows:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

Thus, the memory cell updates after a defined number of timesteps and outputs a vector of scaled predictions.

3 Results

The reference calorific value reductions were compared to the predicted calorific value reductions. The predictors used in mock experiment number 3 are the same as those used in the reference experiment. Consequently, the calorific value reduction should be the same. As shown in figure S14 the linear relationship between the calorific value reductions in the reference data set and the mock data set demonstrates the capacity of the equation used to generate mock data to reproduce the response data acquired during the the reference experiment.

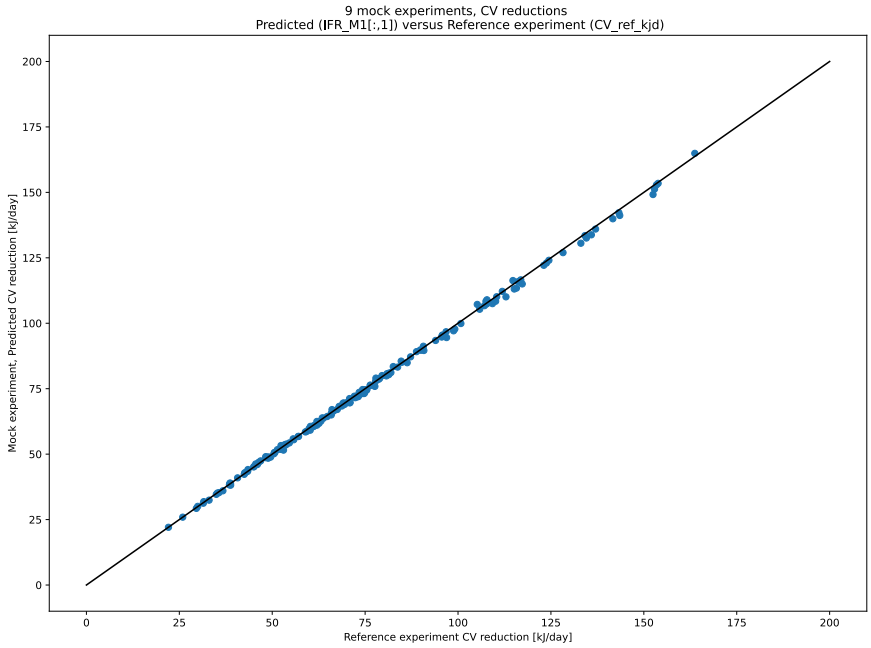

Figure S14: Calorific value reduction: reference data set versus mock experiment E3

As shown in figure S15 the rank of the experiments in terms of CV reduction varies with the flow rate. The predictors used in experiment E3 (ESD=36, MAT=PUF, HDR=3.6) yield the highest CV reduction when the flow rate is high. The predictors used in experiment E1 (ESD=4, MAT=PVC, HDR=0.5) yield the highest CV reduction when the flow rate is low. This implies that set of predictors can be selected according to the flow rate range.

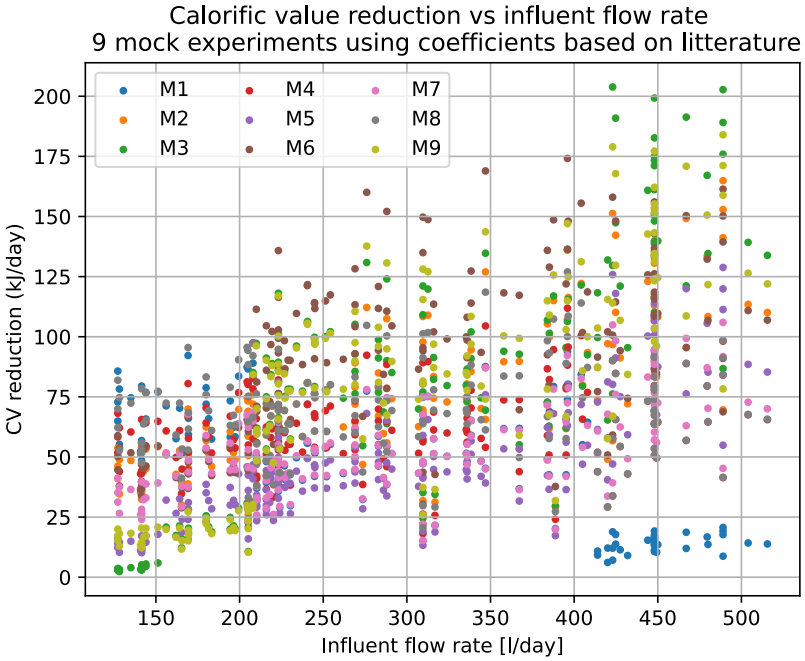

Figure S15: Calorific value reduction versus influent flow

Polynomial

As shown in figure S16 (unshuffled, true time series) the ploynomial model does not accurately predict extreme values during the entire experiment.

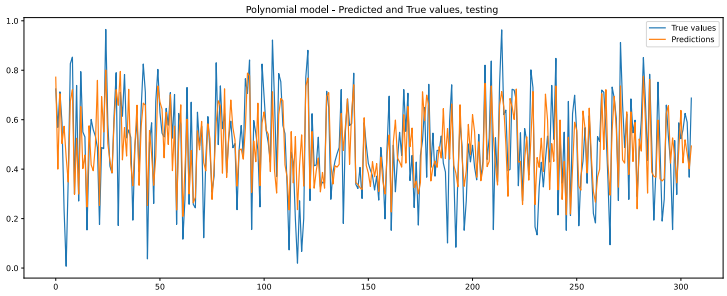

Figure S16

As shown in figure S17 (shuffled time series) the polynomial model does not accurately predict extreme values during the entire experiment.

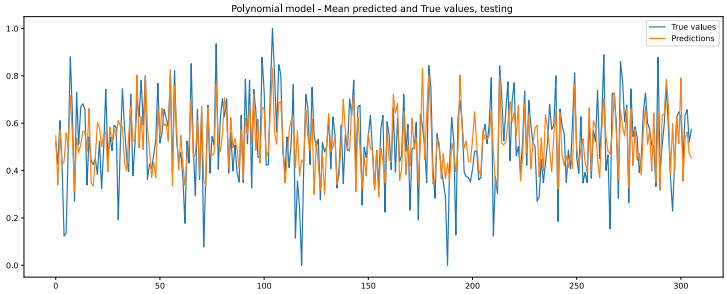

Figure S17

Comparaison of the polynomial, MPL and LSTM models

Figure S18 shows plots of predicted versus true values obtained from the polynomial, MLP, and the LSTM models using not-shuffled, true time series data.

Figure S19 shows plots of predicted versus true values obtained from the polynomial, MLP, and the LSTM models using pre-shuffled data.

Figure S20 compares the results of analysis of the prediction accuracy of multiple runs of the polynomial model and of multiple runs of the MLP and LSTM models with random weight initialisation. The high slope and the narrow distribution of the slope and the RMSE demonstrate the accuracy of MLP model. The highway LSTM has a higher slope and RMSE and wider distribution of both slope and RMSE than the 2 other models. The polynomial model has a much lower slope and a higher RMSE than the MLP model.

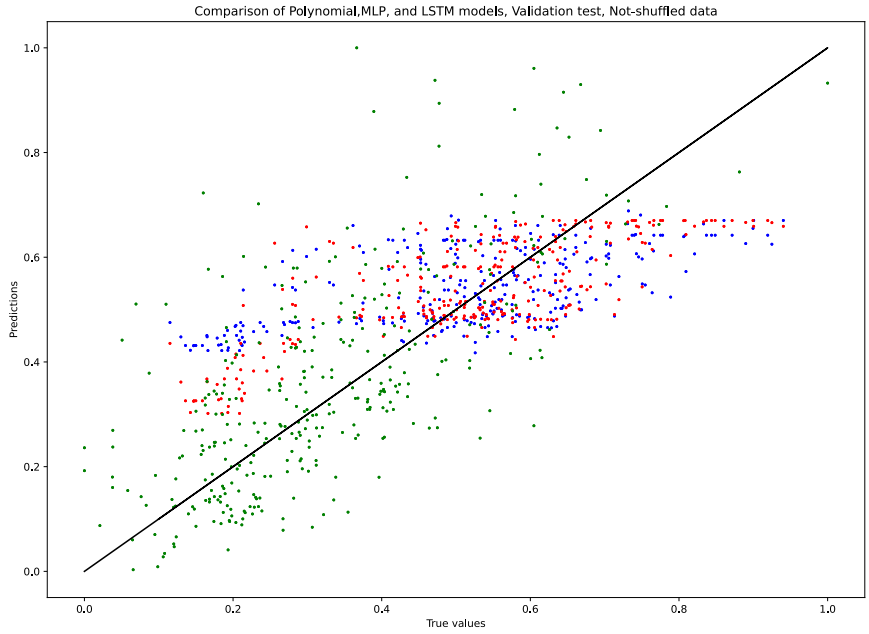

Figure S18: **True versus predicted values (unshuffled, true times series data) Polynomial model: blue; MLP model: red; LSTM model: green**

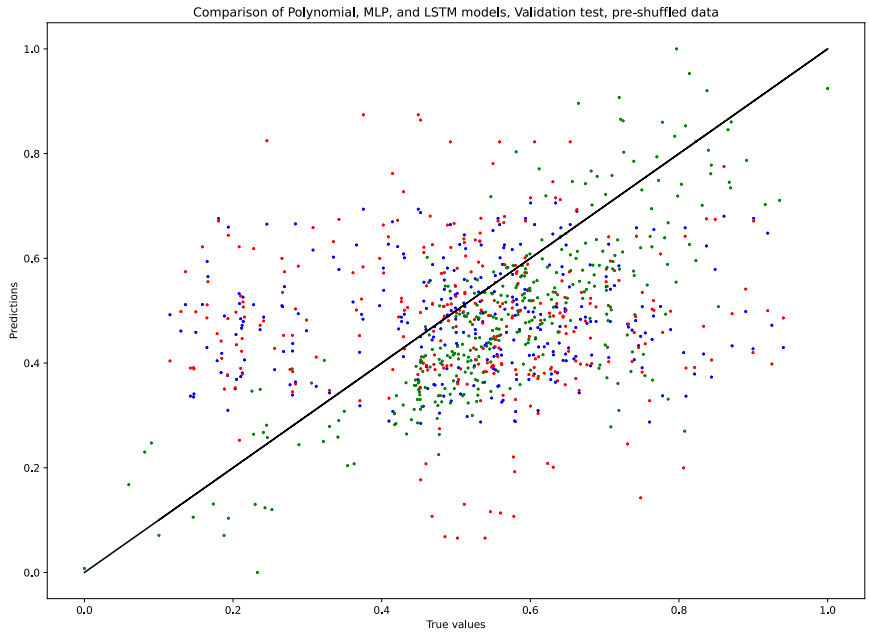

Figure S19: **True versus predicted values (pre-shuffled data) Polynomial model:blue; MLP model: red; LSTM model: green**

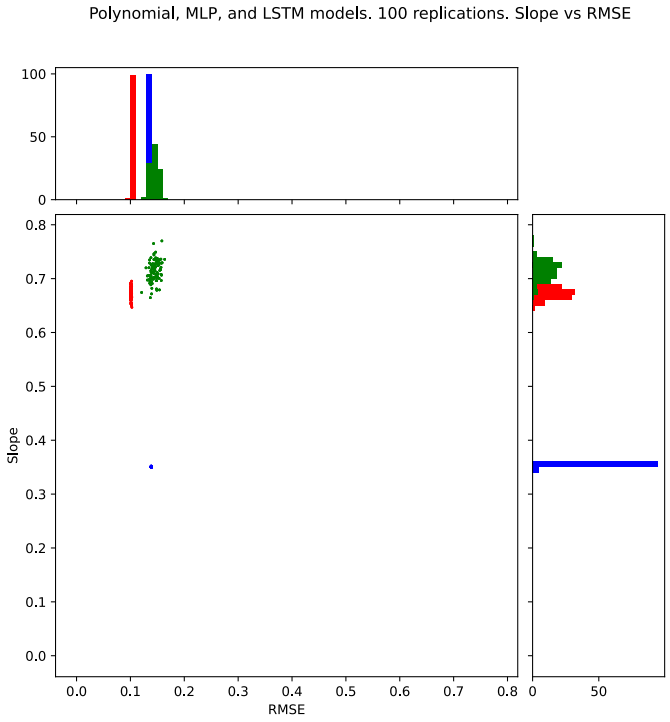

Figure S20: **RMSE and Regression line slope of the polynomial model and multiple runs of the MLP and LSTM models** . A slope equal to 1 corresponds to a perfect match between predicted and true values. The histograms show the corresponding frequency distributions.
Polynomial model: blue; MLP model: red; LSTM model: green

Simulations

Figure S21 shows examples of the low (127 - 321 l/day) and high (322 - 516 l/day) range influent flow regimes used during simulation.

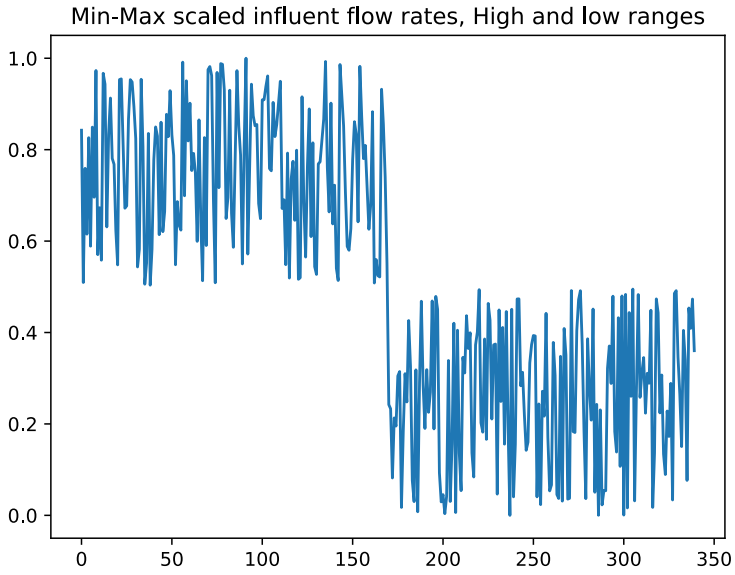

Figure S21: **Randomly generated influent flow rates.** Left: High range (322 - 516 l/day); Right: Low range (127 - 321 l/day).

References

[1] P. Araujo et al. "Multilayer perceptron neural network for flow prediction". In: *J. Environ. Monit.* 13.1 (2011), pp. 35–41. DOI: 10.1039/C0EM00478B.

[2] S Di Berardino, S Costa, and A Converti. "Semi-continuous anaerobic digestion of a food industry wastewater in an anaerobic filter". In: *Bioresource Technology* 71.3 (2000), pp. 261–266. DOI: https://doi.org/10.1016/S0960-8524(99)00080-2.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[4] D.J. Elliot Lynn C. Smith and A. James. "Mixing in upflow anaerobic filters and its influence on performance and scale-up". In: *Water Research* 30.12 (1996), pp. 3061–3073. DOI: 10.1016/S0043-1354(96)00169-8.

[5] C. Olah. *Understanding LSTMs*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs.

[6] Jean Philippe Steyer Renaud Escudié Thierry Conte and Jean Philippe Delgenès. "Hydrodynamic and biokinetic models of an anaerobicfixed-bed reactor". In: *Process biochemistry* 40.7 (40 2005), pp. 2311–2323. DOI: doi:10.1016/j.procbio.2004.09.004.

[7] Skipper Seabold and Josef Perktold. “statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*. 2010.

[8] Dhandapani Thirumurthi. “Effects of mixing velocity on anaerobic fixed film reactors”. In: *Water Research* 22.4 (1988), pp. 517–523. DOI: 10.1016/0043-1354(88)90049-8.