Stochastic Environmental Research and Risk Assessment (2023) 37:4911–4919
https://doi.org/10.1007/s00477-023-02551-9

ORIGINAL PAPER

# Prediction of pore-scale clogging using artificial intelligence algorithms

Chao Lei[1] · Mandana Samari-Kermani[2] · Hamed Aslannejad[3] · Alraune Zech[2]

Accepted: 20 August 2023 / Published online: 5 September 2023

**Abstract**
We use five established, but conceptually different artificial intelligence algorithms for analysing clogging and quantifying colloid transport at pore scale: artificial neural networks, decision tree, random forest, linear regression, and support vector regression. We test how these algorithm can predict clogging by interpolating physics based simulation data. Our training and test data set is based on results from Lattice Boltzmann simulations reproducing the physics of colloid transport through a typical pore throat present in glass beads or medium sized sand. We perform hyperparameter optimization through cross validation for all algorithms. The tree based methods have the highest Nash–Sutcliffe efficiencies among all tested algorithms with values mostly above 0.9 for the independent test data. The event of clogging can be predicted even with 100% accuracy. Our results indicate a non-linear, rather categorial nature of the (simulation) data. This is in contrast to the typical use of neural network algorithms for simulation data while tree based methods are often applied to observational data. We partly link this to the small size of our dataset. Our application of artificial intelligence in porous media research shows that time-consuming Lattice Boltzmann simulations can be easily supplemented and extended at small computational costs while predictability of clogging and quantitative effects of process specific parameters on colloidal transport are given with high reliability.

**Keywords** Clogging · Porous media · Artificial intelligence algorithms · Pore scale · Colloid transport

## 1 Introduction

Clogging of pumping wells is a phenomenon typically observed in civil engineering, such as for drinking water supply, groundwater remediation, or artificial recharge systems. Van Beek et al. (2009) estimated the cost to tackle the problem of clogging to around 5000 euro per year per well (in the Netherlands). Thus, improving the understanding of clogging mechanisms and predicting clogging can help reducing the probability of occurrence and eventually save costs by an increase of efficiency. From the pore scale viewpoint, clogging is typically caused by an accumulation of colloids at the interface between the aquifer material and gravel pack of the pumping well. Samari-Kermani et al. (2020, 2021) performed physics-based pore scale simulations of colloid transport studying the impact of hydro-dynamic forces, gravity, electrostatic forces and van der Waals forces making use of the Lattice Boltzmann Method (LBM).

Artificial intelligence (AI) algorithms offer an alternative to quickly study the impact of parameters by relating input to output of existing (simulation) data. AI can effectively substitute for physics-based models and greatly reduce the computational time necessary for data interpolation and outcome prediction.

The application of AI and machine learning became more popular in porous media research (Tahmasebi et al.

✉ Alraune Zech
a.zech@uu.nl

Chao Lei
chao.lei@ufz.de

Mandana Samari-Kermani
m.samarikermani@uu.nl

Hamed Aslannejad
h.aslannejad@uu.nl

[1] Helmholtz Centre for Environmental Research-UFZ, Permoserstr. 15, 04318 Leipzig, Germany

[2] Department Earth Science, Utrecht University, Princetonlaan 8a, 3584CB Utrecht, Netherlands

[3] Copernicus Institute of Sustainable Development, Utrecht University, Princetonlaan 8a, 3584CB Utrecht, Netherlands

2020) and in hydrology (Lange and Sippel 2020) in recent years. A typical application is the prediction of porosity and/or permeability, e.g. from image analysis (Wu et al. 2018; Rabbani and Babaei 2019; Tembely et al. 2020), observation of soil properties (Jorda et al. 2015; Gupta et al. 2021; Araya and Ghezzehei 2019) or direct numerical simulation and experiments (Jiang et al. 2021; Erofeev et al. 2019; Tian et al. 2021). Babakhani et al. (2017) and Goldberg et al. (2015) are examples for applications of AI on prediction and sensitivity of colloid and nanopartilce transport in porous media. The choice of AI algorithm differs between applications. While studies with soil property data showed good results with tree based algorithm (linked to the nature of the categorical data), most studies using simulation data make use of Neural Network algorithms. The application of multiple AI algorithms is rarely done which leaves the studies with the assumption that the selected algorithm was well suited for the data.

The results of Samari-Kermani et al. (2020, 2021) provide an opportunity to study the application of AI to clogging at the pore scale. We can analyse parameter impact quickly and interpolate results to predict clogging for parameter combinations not tested in physical simulation due to the computational burden. We test multiple established, but conceptually different AI algorithms to identify which method is best suited for this particular kind of data. We develop a workflow for identifying the relationship between four process parameters and four quantitative output values as well as the general event of pore clogging.

## 2 Materials and methods

### 2.1 Data

We use numerically generated clogging data from Samari-Kermani et al. (2020, 2021). They performed 162 colloid transport simulations in a constricted tube with the Lattice Boltzmann method (LBM) on a high resolution numerical grid. The domain has a sinusoidal pore shape of 200 μm length and 50 μm height which reduces to 20 μm at the throat. It represents a typical pore throat present in glass beads or medium-sand packs.

The LBM is well suited to simulate the physics of fluid flow as well as collision and streaming of particles in an irregular geometry. This advantage, however, comes at a substantial cost of high computational effort and long simulation run times. Individual runs took between one and 20 days (on a regular PC, single core), depending on particle size, velocity and clogging behaviour.

Clogging behaviour is modelled at the pore scale by allowing collision and streaming of particles. All relevant physical processes are represented, such as hydro-dynamic forces, gravity, buoyancy, van der Waals forces, and electrostatic forces. This complete formulation also accounts for the effects of inter-particle forces and pore structure changes due to colloid retention. Input parameters are: (i) particle diameter (*D*), specifying the size of the transported colloids; (ii) mean flow velocity (*U*) which is the average velocity at steady state while no particles are inside the pore; (iii) ionic strength (*IS*) being the concentration of ions in the solution; and (iv) zeta potential (*Z*) of the particles and pore surface which specifies the electrostatic potential of the electrical double layer enclosed with particles in solution.

Samari-Kermani et al. (2020, 2021) tested three values for each parameter (Table 1) in all combinations to obtain systematic results. The values for the zeta potential refer to both, the particles and the pore surface having the same absolute value. Particles are always negatively charged while the surface charge determined two conditions: (i) favorable, where the charge of the surface is positive, thus opposite to that of particles (81 samples); and (ii) unfavorable, where charges of surface and particles are alike (81 samples). In case of favorable conditions, particles were easily deposited due to attractive van der Waals forces and electric double layer interactions. In contrast, under unfavorable conditions particles were able to roll over the grain surface reducing the chance of clogging.

Simulations were first analysed on the event of clogging, i.e. a total blockage of the flow through the pore throat by agglomerates of particles. Results are quantified through four output values: (i) The average coordination number (*CN*) is the mean of all particles coordination numbers. Each particle's coordination number identifies the number of other particles it is connected to (e.g. zero for single particles or three if a particle is in contact to three others within an aggregate). (ii) The surface coverage (*SC*) is the ratio of grain surface area occupied by attached particles to the total grain surface area. (iii) The hydraulic conductivity (*HC*) relates mean velocity to the pressure gradient in the pore before injecting any colloids. Values are non-dimensionalised by the highest HC related to each of the three fluid velocities. (iv) The void fraction (*VF*) is the ratio of

**Table 1** Input parameter values used by Samari-Kermani et al. (2020)

| | Value 1 | Value 2 | Value 3 |
|---|---|---|---|
| particle size *D* (um) | 3 | 5 | 10 |
| mean fluid velocity *U* (m/day) | 1 | 5 | 10 |
| ionic strength *IS* (M) | 0.001 | 0.05 | 0.3 |
| zeta potential *Z* (mv) | ±17.5 | ±45.56 | ±60 |

pore volume not occupied by the particles, to total pore volume.

The quantitative output values are linked to the event of clogging. For instance, a very low conductivity and high coordination number are indicators for clogging as both suggest large agglomerates formed by the particles which block the pore throat. Links between input and output parameters in the context of the physical processes are further discussed in Sect. 4.

We apply AI to two data sets based on the results of the 162 simulations of Samari-Kermani et al. (2020, 2021): *Data Set 1* relates all variations of the four input parameters (Table 1) to each of the four output values, individually. *Data Set 2* has the same four input parameters but only one output value for each simulation which reflects the event of clogging. It is specified by a binary output value of 1 for clogging or $-1$ for non-clogging. Both data sets are subdivided into favorable and unfavorable conditions, resulting in two times 81 samples for both data sets. We consider the numerical LBM results as ground truth to be tested against AI model output.

## 2.2 Metrics and data normalization

Where required by the AI, we normalized input data of the coordination number (all other input parameters were already between 0 and 1). Values are processed via $x_N = \frac{x_i - x_{min}}{x_{max} - x_{min}} \in [0, 1]$, where $x_{max}$ and $x_{min}$ refer to the maximal and minimal occurring values.

Our standard performance measure is the Nash–Sutcliffe model efficiency (NSE):

$$NSE(\vec{z}, \vec{y}) = 1 - \frac{\sum_{i=1}^{n} (z_i - y_i)^2}{\sum_{i=1}^{n} (z_i - \bar{z})^2}, \tag{1}$$

where $\bar{z}$ is the average of $\vec{z}$, which we consider as sample value vector (ground truth) while $\vec{y}$ is the model output vector.

The NSE relates the variability explained by the model $\vec{y}$ to the total variability in the sample values $\vec{z}$. A value close to 1 indicates a useful model while a value close to or below zero indicates that the model is not well suited. Note that the NSE reflects the coefficient of determination ($R^2$) for statistical models, i.e. for the model performance during the training phase.

## 2.3 AI Algorithms

We applied the five AI algorithms: (i) artificial neural networks (ANN); (ii) decision tree (DT); (iii) random forest (RF); (iv) linear regression (LR); and (v) support vector regression (SVR). Each algorithm has a set of internal parameters and hyperparameters, depending on its specific structure. While internal parameters are trained during the learning process, hyperparameters are specified by the user before training and affect the performance of AI models (Wu et al. 2019). We shortly outline the applied algoritms. For detailed information, the reader is referred to e.g. Hastie et al. (2009), Tahmasebi et al. (2020) or Russell and Norvig (2020). We make use of the algorithms implementation in the Python package `scikit-learn` (Pedregosa et al. 2011) following their specification and standard choices of hyperparameters.

### 2.3.1 Artificial neural network (ANN)

Artificial Neural Networks is a modeling technique inspired by the working mechanism of the brain. ANN is structured in layers each having a certain number of neurons. Neurons receive signals from neurons in other layers, process them, and forward them to neurons in the next layer. ANN has one input layer, an output layer and a certain number of hidden layers in between. The number of neurons in the input and output layer are determined by the number of input and output values, respectively. The number of hidden layers and the number of neurons in the hidden layers are hyperparameters. Given the limited number of available data, we apply one hidden layer and only tune the number of neurons $N_{\mathrm{ANN}}$ during hyperparameter testing.

Information propagates from the input layer through the hidden layer(s) to the output layer. Neurons process information through an activation function, which is also subject of choice by the user. We used the common sigmoid function. Other settings can be found in the accompanying python-scripts (Zech and Lei 2023).

### 2.3.2 Decision tree (DT)

Decision tree makes use of a tree-like structure of decisions with nodes, branches and leaves. The decisions leading to various tree branches refer to input parameters, while leaves represent target output values. The data set is split up into subsets relating the different input parameters to conclusions about output values. Decisions are controlled by conditional statements and regression to find the optimal splitting point.

Key parameters of the algorithm (we control here) are the maximum tree depth $D_{\mathrm{DT}}$ which limits the number of branching and the minimum sample split number *mss* which limits further branching of nodes, having *mss* or fewer samples. The choice of the two hyperparameters is particularly important to avoid overfitting by having deep trees that are likely to master too many details of the training dataset.

### 2.3.3 Random forest (RF)

Random Forest is an ensemble algorithm that combines a certain number $N_{RF}$ of randomized decision trees (Breiman 2001). The training data set for each decision tree is selected through the bagging procedure by resampling the original training data set with random replacements. Thus, some input samples may be used many times, while others are ignored. The final results follow from averaging the prediction of each tree. We test model performance for various values of the hyperparameter tree estimator $N_{RF}$ where the default value is 100 (Pedregosa et al. 2011).

### 2.3.4 Linear regression (LR)

Linear regression is a classical method to construct a predictive model based on fitting of input to known output values. The linear model function reads $y_{LR} = \vec{\omega} \cdot \vec{x} + b$, where $\vec{x}$ is the vector of input values, $\vec{\omega}$ is the parameter vector and $b$ is the disturbance term. We apply LR with Ridge regression for fitting: $\min_{\vec{\omega},b}\left(\mathrm{MSE}\left(\vec{y}_{LR}, \vec{y}_T\right) + \alpha||\vec{\omega}||_2^2\right)$ where $||\vec{\omega}||_2^2$ is the $L_2$ norm and $\alpha$ is a hyperparameter in the regularization term to avoid over-fitting.

### 2.3.5 Support vector regression (SVR)

Support vector regression is trained by finding a hyperplane that separates the training data and assigns new data points to a class based on their position within the grid of a linear problem (Rosenbaum et al. 2013). To optimize prediction accuracy, the algorithm fits the error within a certain margin, either by a hard or soft margin that allows some miss-classification. We tune the optimization parameter $C$, also called penalty. It compromises between correct classification of a sample against the maximization of the decision function's margin. For larger values of $C$, smaller margins will be preferred if this leads to better predictions. A lower $C$ promotes a larger margin at the cost of the training efficiency.

SVR is adapted to non-linear problems by introducing kernel functions which map data into higher dimensional space. We tested SVR with linear, polynomial and radial basis function kernel. We focus here on SVR with radial basis (or Gaussian) function. This gives another hyperparameter $\gamma > 0$ which influences the shape of the Gaussian kernel function.

## 2.4 Algorithm application

All algorithm training and testing is implemented in Python using the `scikit-learn` package (Pedregosa et al. 2011). Scripts are public available on *Github* (Zech and Lei 2023).

The workflow is: (i) split the entire data set into the training and test data set; (ii) identify optimal hyperparameter(s) for each AI algorithm for the training data set; (iii) evaluate model performance on the test data set.

The data is split using a 90–10 ratio. Thus, the training data set consists of 73 samples while the test data sets consist of 8 samples for each of the two conditions (favorable and unfavorable). The initial split of the data set was done randomly, but then kept the same for all algorithm application.

Table 2 shows the hyperparameters we tune and the range of tested values. We identify optimal hyperparameters through a combination of cross validation and performance on the entire training data set. Details on the procedure and selection of hyperparameters is provided in the *SI*.

After hyperparameter identification, all algorithms are trained (again) on the entire training data set with the optimal hyperparameters to specify the algorithms' internal parameters. Then, the trained models are applied to the test data set to which they have not been exposed before. Model performances are ranked based on the NSE (Sect. 2.2). The run time of all algorithms for training and testing is in the order of minutes. Hyperparameter testing took at times a few hours due to the iterations.

For the regression task on *Dataset 1*, we run the training and testing of all AI models separately for each of the four output values coordination number, hydraulic conductivity, surface coverage and void fraction for both conditions—favorable and unfavorable. Thus, we have eight cases of

**Table 2** Hyperparameters and values tested for artificial neural network (ANN), decision tree (DT), random forest (RF), linear regression (LR), and support vector regression (SVR)

| Algorithms | Hyperparameter | Values |
|---|---|---|
| ANN | $N_{ANN}$—Neurons in hidden layer | $2, \ldots, 200$ |
| DT | $D_{DT}$—maximal depth | $2, \ldots, 19$ |
| | *mss*—min. sample split | $2, \ldots, 9$ |
| RF | $N_{RF}$—number of tree estimator | 100, 200, 300 |
| LR | $\alpha$—regularization parameter | $10^{-4}, 10^{-3}, \ldots, 10, 100$ |
| SVR | $C$—penalty | $10^{-3}, 10^{-2}, \ldots, 10, 100$ |
| | $\gamma$ – shape | $10^{-3}, 10^{-2}, \ldots, 10, 100$ |

output values which all have an own set of optimal hyperparameters for each algorithm. Motivated by the algorithm performances for *Dataset 1*, we only apply ANN, DT and RF to the classification task for *Dataset 2*.

## 3 Results

### 3.1 Training and hyperparameter identification

Figure 1 shows the performance of ANN as function of the number of neurons and cross validation iterations for each of the four output values. It shows a clear trend: with increasing number of neurons, the NSE converges except for the surface coverage under favorable conditions. The optimal hyperparameter $N_{\text{ANN}}$ is consequently chosen in the range of 150 or higher, except for SC where the performance peak is between 50 and 80 neurons. This peak in $N_{\text{ANN}}$ can be the result of overfitting. Selected optimal values are listed in Table S1 (*SI*). Figure 1 further shows that results do not significantly differ with the number of iterations tested.

The model performance during cross validation is partly poor, particularly for favorable conditions, visible in low and/or negative NSE values in Fig. 1. We link this to the small training data set which is further reduced during cross validation. Nonetheless, the performances for the entire training data set with the optimal hyperparameter are good. NSE values are between 0.65 and 0.98 with the lowest value for the void fraction under favorable conditions which is in line with the trend seen in Fig. 1.

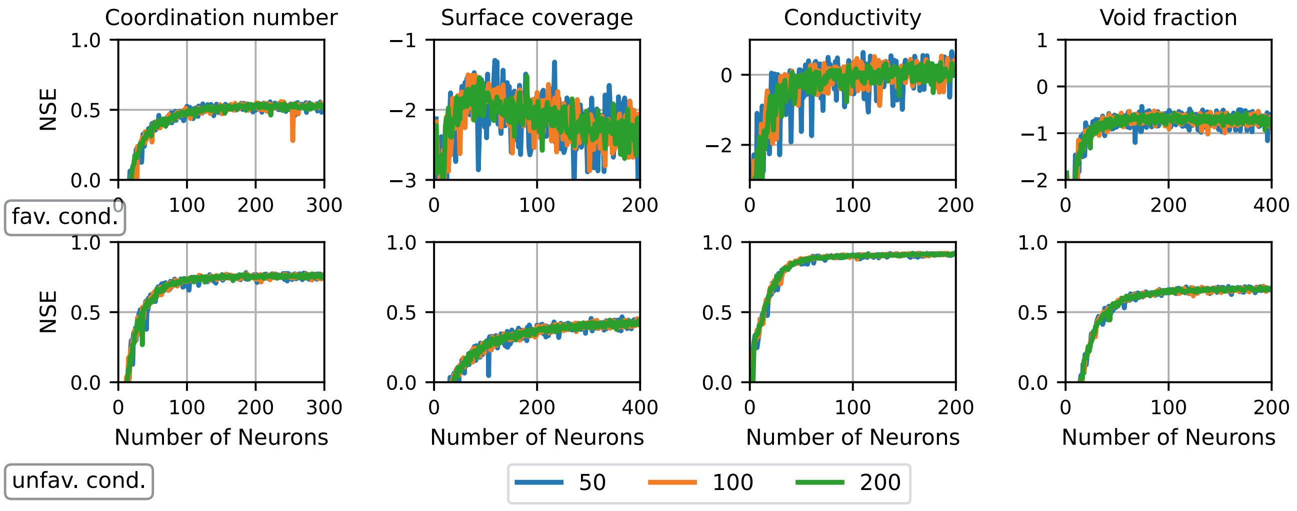

**Fig. 1** Performance of hyperparameter testing for ANN as function of number of neurons and test iterations (colored lines) based on NSE evaluated for the coordination number, surface coverage, conductivity and void fraction (columns) for favorable (upper row) and unfavorable (lower row) conditions

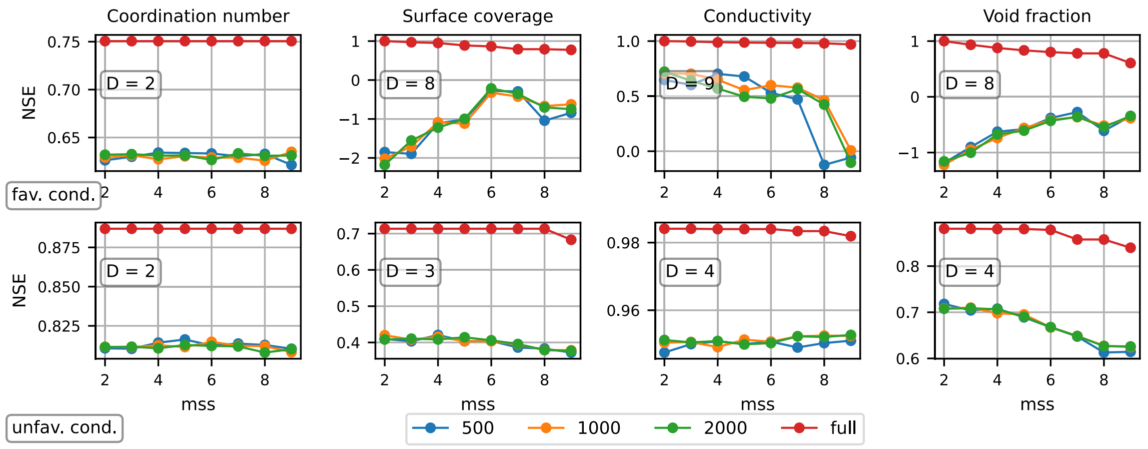

**Fig. 2** Performance of hyperparameter testing for DT as function of minimum sample split *mss* and test iterations (colored lines), including full training data set evaluated for the coordination number, surface coverage, conductivity and void fraction (columns) for favorable (upper row) and unfavorable (lower row) conditions. The tree depth *D* is kept at the optimal value

Cross validation for the two hyperparameters of DT, maximum depth $D$ and minimum sample split *mss*, provided ranges of good results. Values were selected based on the performance on the entire training data set. Details are provided in Figure S2 and Table S2 (*SI*).

Figure 2 displays the performance for various values of *mss* (with fixed optimal $D_{\mathrm{DT}}$) for the entire training data set and for cross-validation. The high NSEs for most parameter combinations and particularly for the full training data set (often close to one) indicate that the specific choice of hyperparameters is of minor relevance for DT.

For RF, the default number of tree estimator $N_{\mathrm{RF}} = 100$ is the best choice. The cross validation with $N_{\mathrm{RF}} = 100, 200$ and 300 did not show significant differences for all four output values (Figure S3, *SI*). The same holds when evaluating the hyperparameter performance for the entire training data set. In general, the model performance was very good with NSE values between 0.936 and 0.996 for all eight cases.

Hyperparameter optimization on $\alpha$ for the LR Ridge algorithm provided ranges of similarly performing values. From that range, we decided for the $\alpha$ value which showed the highest score for the entire training data set. We see that the performance of LR is already worse (compared to the other algorithms) for the training data set with NSE values between 0.01 and 0.62 (Table S4, *SI*).

SVR showed best performances for the combination of $\gamma = 1$ and $C = 10$ or $C = 100$ with a few minor deviations (Figure S5 and Table S5, *SI*). NSE values for the entire training data set range between 0.1 and 0.94.

## 3.2 Algorithms' performance analysis

Figure 3 summarizes the performance of all algorithms on *Dataset 1* in terms of NSE for the test data using the optimal hyperparameters. Most NSE values are above zero, many even close to one, indicating a good predictability of output values by the AI algorithms.

The algorithms performing best are DT and RF. They outperform all other algorithms, except for surface coverage under unfavorable conditions where ANN is the best model. While RF has the highest score in most cases, DT has the highest NSE in average. NSE values are for most cases very close to one. We relate the mediocre results of ANN to the small training data set (less than 100 samples), as ANN is typically best suited for very large data sets (Alwosheel et al. 2018). LR and SVR are the algorithms with poor performances.

Figure 3 furthermore shows that almost all output values can be predicted with decision tree and partly random forest at a very high precision level, i.e. NSE value very close to one. The only exceptions are the coordination number under favorable conditions ($\mathrm{NSE}_{\mathrm{DT}} = 0.8$) and the surface coverage under unfavorable conditions ($\mathrm{NSE}_{\mathrm{DT}} = 0.69$), indicating a complexity of input parameter to output value relation which cannot be captured easily by neither of the algorithms.

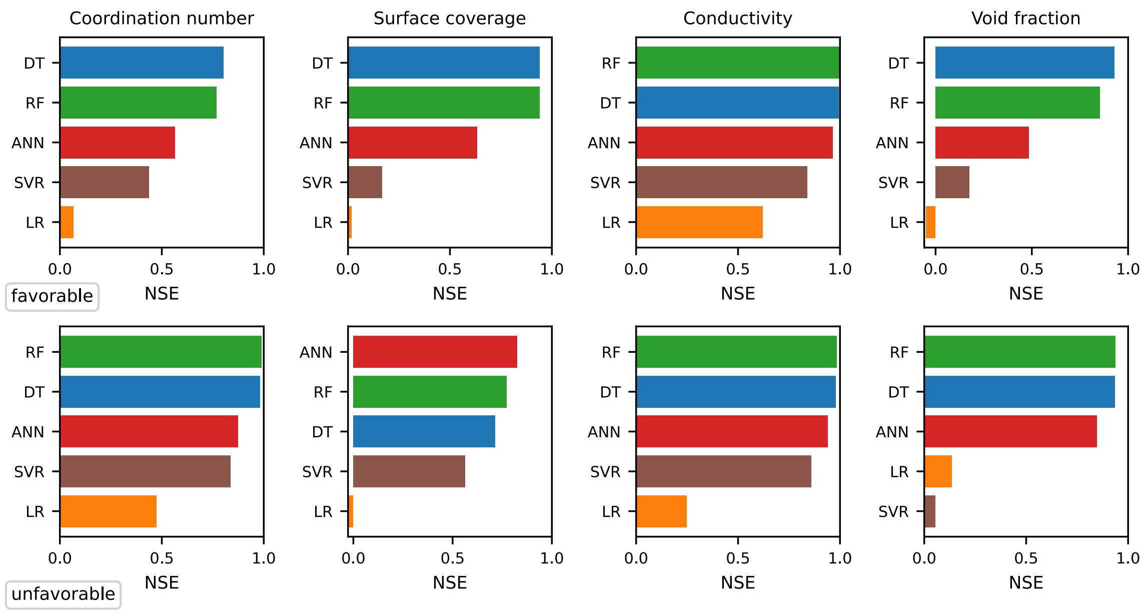

**Fig. 3** Model performances (NSE) on test data set using optimal hyperparameters for all four output parameters (columns) under favorable (upper row) and unfavorable (lower row) conditions. Algorithms are color coded: red—artificial neural network (ANN), blue—decision tree (DT), green random forest (RF), orange—linear regression (LR), brown - support vector regression (SVR). Values can be found in the *Supporting Information*

### 3.3 Predicting clogging

We trained ANN, DT and RF (based on their good performance for *Dataset 1*) for predicting the event of clogging (*Dataset 2*). The identified hyperparameter during training are 'small' being in line with the relative simple classification task: ANN uses 5 and 4 neurons in a single hidden layer under favorable and unfavorable condition, respectively; DT was trained to an optimal minimum sample split of $mms = 2$ (reasonable for a two value classification) and an optimal maximum depth $D_{\mathrm{DT}}$ of 5 and 3 for favorable and unfavorable conditions, respectively. For RF we made use of the standard input parameters. Hyperparameter performance is summarized in Fig. S7 (*SI*). All three algorithms were capable of perfectly predicting the clogging for the test data set for both conditions; i.e. with a NSE of one.

## 4 Discussion

From the performance of the different algorithms, we can infer information on the process-based nature of the data. The poor performance of LR and SVR show that the physical relationship between the input parameter and output variables for the clogging phenomena is neither linear nor can it be described with the Gaussian kernel. Methods such as ANN, DT, RF that accommodate non-linear structures are more suitable for the described problem.

The performance of AI algorithms also depends on the relation of (training) data to the domain of interpolation. For predicting clogging, the latter is small (actually bimodal), explaining the good performance of all algorithms despite the small amount of input training data of 73 samples. The parameter space for the output parameters CN, SC, HC and VF is larger given the continuous range of potential values. Again, the poor performance of LR for most cases (Fig. 3) shows that the relation between inputs and outputs is not linear.

The non-linear relation between input and output can be linked to the physics of clogging. An increase of the ionic strength leads to the formation of bigger agglomerates, represented by a higher coordination number. At the same time, an increase of the flow velocity counteracts on that as higher hydrodynamic forces lead to detachment and split off of agglomerates. This is also supported by correlation coefficients between input and output parameters provided in Figure S8 (*SI*).

All algorithms had most problems predicting surface coverage under unfavorable conditions (Fig. 3). Here, the presence of a secondary energy minimum, led to rolling of colloids on the grain surface rather than attachment. Consequently, the surface coverage is high while not resulting in a clogged pore which counteracts the usual pattern of input to output relation.

Comparing our work to other studies of AI application to pore-scale related process data shows surprising results with regard to the type of suited algorithms. While neural network algorithms are typically used in studies based on simulation results, such as Babakhani et al. (2017), Wu et al. (2018), Rabbani and Babaei (2019), Tembely et al. (2020) and Jiang et al. (2021), we see that ANN does not perform best. Instead, the tree like algorithm are better suited for our data, which we partly relate to the small amount of training data which can be better handled by DT and RF. At the same time, the very good performance of DT and RF indicates that the relation between input and output data is of categorial nature. Araya and Ghezzehei (2019) found a similar trend of tree-based AI algorithms performing best for predicting saturated hydraulic conductivity.

## 5 Summary and conclusion

We applied five machine learning algorithms (Artifical Neural Networks, Decision Tree, Random Forest, Linear Regression and Support Vector Regression) for regression and classification tasks associated with the clogging phenomenon in porous media. As ground truth, we used results from Lattice Boltzmann simulations mimicking the physics of colloid transport at pore scale (Samari-Kermani et al. 2020, 2021). The process input parameters particle size, mean flow velocity, ionic strength and zeta potential were related to quantitative output values of coordination number, surface coverage, hydraulic conductivity, and void fraction as well as the general event of pore clogging.

In contrast to highly time-consuming Lattice Boltzmann simulations (days to weeks), the application of AI (order of minutes) allows us to quickly predict clogging through data interpolation. Furthermore, the quantitative effect of process specific parameters on colloidal transport can be extended from a few values to a continuous parameter range at small computational costs. However, AI is not suited for extrapolation, i.e. predicting clogging outside of the parameter range tested with LBM simulations.

The small training and test data set also causes some limitations. The capability of algorithms to learn and predict is restricted which can be resolved by a broader range of tested input parameters (in the simulations). However, as shown, some AI algorithms are able to handle even this small amount of training data for providing reliable interpolation results with little effort on hyperparameter tuning. From our results we conclude:

- Clogging and colloid transport quantifiers can be well interpolated with artificial intelligence algorithms when trained with a relatively small data set from physical simulations.
- Decision tree and random forest are best suited as they can be trained already on small data sets. Furthermore, their algorithm structure fits to the physical non-linear relation between input values and simulation results.
- Combining AI with a few (time-consuming) physics-based simulations (e.g. from Lattice Boltzmann) is a quick and computationally cost-effective way to extend data sets on predicting the quantitative effect of parameters.

As application of AI in porous media research, our work is an example for developing (simple) model emulators based on process-based models. On the one hand, our results supplements and extends the work of Samari-Kermani et al. (2020, 2021) providing additional data on pore scale clogging. On the other hand, the procedure can be applied to other data sets (subject to similar physics) giving indication on the choice of algorithms and hyperparameter tuning.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s00477-023-02551-9.

**Acknowledgements** The data used as well as python scripts for reproducing the Figures are provided in an open-source repository https://github.com/AlrauneZ/Clogging_AI. We thank three anonymous reviewers for their helpful comments.

**Author contributions** C.L. setup the data analysis with support of H.A. and A.Z. A.Z and C.L. wrote the main manuscript text, prepared all figures and the associated python scripts. M.S. provided the data and supported analysis of results. All authors reviewed the manuscript and contributed to the revision process.

**Funding** The authors have not disclosed any funding.

## Declarations

**Conflict of interest** The authors declare no competing interests.

## References

Alwosheel A, van Cranenburgh S, Chorus CG (2018) Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. J Choice Model 28:167–182. https://doi.org/10.1016/j.jocm.2018.07.002

Araya SN, Ghezzehei TA (2019) Using machine learning for prediction of saturated hydraulic conductivity and its sensitivity to soil structural perturbations. Water Resour Res 55(7):5715–5737. https://doi.org/10.1029/2018WR024357

Babakhani P, Bridge J, Doong R et al (2017) Parameterization and prediction of nanoparticle transport in porous media: a reanalysis using artificial neural network. Water Resour Res 53(6):4564–4585. https://doi.org/10.1002/2016WR020358

Breiman L (2001) Random Forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324

Erofeev A, Orlov D, Ryzhov A et al (2019) Prediction of porosity and permeability alteration based on machine learning algorithms. Transp Porous Media 128(2):677–700. https://doi.org/10.1007/s11242-019-01265-3

Goldberg E, Scheringer M, Bucheli TD et al (2015) Prediction of nanoparticle transport behavior from physicochemical properties: machine learning provides insights to guide the next generation of transport models. Environ Sci Nano 2(4):352–360. https://doi.org/10.1039/C5EN00050E

Gupta S, Lehmann P, Bonetti S et al (2021) Global prediction of soil saturated hydraulic conductivity using random forest in a covariate-based geotransfer function (CoGTF) framework. J Adv Model Earth Syst 13(4):e2020MS002,242. https://doi.org/10.1029/2020MS002242

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer, New York

Jiang F, Yang J, Boek E et al (2021) Investigation of viscous coupling effects in three-phase flow by lattice-Boltzmann direct simulation and machine learning technique. Adv Water Resour 147(103):797. https://doi.org/10.1016/j.advwatres.2020.103797

Jorda H, Bechtold M, Jarvis N et al (2015) Using boosted regression trees to explore key factors controlling saturated and near-saturated hydraulic conductivity. Eur J Soil Sci 66(4):744–756. https://doi.org/10.1111/ejss.12249

Lange H, Sippel S (2020) Machine learning applications in hydrology. In: Forest–water interactions. Springer Nature, Berlin pp 233–257. https://doi.org/10.1007/978-3-030-26086-6_10

Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Rabbani A, Babaei M (2019) Hybrid pore-network and lattice-Boltzmann permeability modelling accelerated by machine learning. Adv Water Resour 126:116–128. https://doi.org/10.1016/j.advwatres.2019.02.012

Rosenbaum L, Dörr A, Bauer MR et al (2013) Inferring multi-target QSAR models with taxonomy-based multi-task learning. J Cheminformatics 5:33. https://doi.org/10.1186/1758-2946-5-33

Russell S, Norvig P (2020) Artificial intelligence: a modern approach, 4th edn. Pearson, Hoboken

Samari-Kermani M, Jafari S, Rahnama M et al (2020) Direct pore scale numerical simulation of colloid transport and retention. Part I: fluid flow velocity, colloid size, and pore structure effects. Adv Water Resour 144:103694. https://doi.org/10.1016/j.advwatres.2020.103694

Samari-Kermani M, Jafari S, Rahnama M et al (2021) Ionic strength and zeta potential effects on colloid transport and retention processes. Colloids Interface Sci Commun 42(100):389. https://doi.org/10.1016/j.colcom.2021.100389

Tahmasebi P, Kamrava S, Bai T et al (2020) Machine learning in geo- and environmental sciences: from small to large scale. Adv Water Resour 142(103):619. https://doi.org/10.1016/j.advwatres.2020.103619

Tembely M, AlSumaiti AM, Alameri W (2020) A deep learning perspective on predicting permeability in porous media from network modeling to direct simulation. Comput Geosci 24(4):1541–1556. https://doi.org/10.1007/s10596-020-09963-4

Tian J, Qi C, Sun Y et al (2021) Permeability prediction of porous media using a combination of computational fluid dynamics and hybrid machine learning methods. Eng Comput 37(4):3455–3471. https://doi.org/10.1007/s00366-020-01012-z

van Beek K, Breedveld R, Stuyfzand P (2009) Preventing two types of well clogging. J Am Water Work Assoc 101(4):125–134. https://doi.org/10.1002/j.1551-8833.2009.tb09880.x

Wu J, Yin X, Xiao H (2018) Seeing permeability from images: fast prediction with convolutional neural networks. Sci Bull 63(18):1215–1222. https://doi.org/10.1016/j.scib.2018.08.006

Wu J, Chen XY, Zhang H et al (2019) Hyperparameter optimization for machine learning models based on Bayesian optimization. J Electron Sci Technol 17(1):26–40. https://doi.org/10.11989/JEST.1674-862X.80904120

Zech A, Lei C (2023) AlrauneZ/Clogging_AI: Version 1.0. https://doi.org/10.5281/zenodo.8123319, https://github.com/AlrauneZ/Clogging_AI