Supporting Information

Machine learning based prediction of gas hydrate dynamics: comparing with fundamental model against experimental data

Gauri Shankar Patel and Amiya K Jana*

Energy and Process Engineering Laboratory, Department of Chemical Engineering, Indian Institute of Technology Kharagpur, India – 721302

S1. Machine Learning (ML)

Learning, in a broad sense, involves gaining new or altering current attitudes, values, knowledge, abilities, or tastes. Machine learning (ML) is creating computer algorithms that acquire knowledge from examples rather than relying on explicit programming to accomplish a specific task.[1] The interdisciplinary nature of machine learning spans various research domains, strengthening its presence and significance. It encompasses an array of algorithms capable of conducting nonparametric, multivariate, and nonlinear classification or regression tasks. [2] Over the last 20 years, machine learning has evolved significantly from an experimental concept to a widely used practical technology. In artificial intelligence (AI), ML is the preferred approach for creating functional software in natural language processing, speech recognition, computer vision, and robotics control. The historical development of ML and AI is closely linked, with significant milestones marking their progress. Alan Turing's Turing Test in 1950 and Marvin Minsky's creation of the first neural network, SNARC, in 1951 are the foundational developments. Arthur Samuel's checkers program in 1952 have demonstrated early ML capabilities.[3] A broad spectrum of ML algorithms exists to address the multitude of data characteristics and problem scenarios encountered in various machine learning challenges.

S1.1. Machine learning paradigms

Machine learning methods can be classified as influenced by the algorithm's training method and the output types. As illustrated in Figure S1, this includes unsupervised, supervised,

*Corresponding author. Fax: +91 3222 282250.
E-mail address: akjana@che.iitkgp.ac.in (A. K. Jana).

reinforcement, evolutionary, ensemble, semi-supervised, neural network, instance-based, multi-task, and hybrid learning.[4,5] The different learning methods are described in the following sections.

S1.1.1. Supervised Learning

Supervised learning is a method in which a machine is trained to predict outputs by providing a set of example input-output pairs. The objective is to formulate a model that deduces the underlying rules linking inputs to their results, enabling the prediction of outputs for new, unobserved inputs. Supervised learning is utilized to make predictions using past data. The tasks within supervised learning are typically divided into two types: classification, which involves identifying category labels, and regression, which focuses on predicting continuous numerical values. [6]

S1.1.2. Unsupervised Learning

Unsupervised learning is so named because it does not rely on predefined labels or guidance, and there is no "teacher" with the correct answers. Instead, algorithms independently sift through data to uncover meaningful patterns and structures. This self-guided learning leverages the inherent relationships and linkages within the data. Unlike supervised learning, unsupervised learning does not employ labelled training datasets. It encompasses two main categories: clustering, which groups similar data points, and association, which identifies rules that pinpoint relationships between variables.

S1.1.3. Reinforcement Learning

Reinforcement learning operates on a trial-and-error basis. In this approach, an algorithm receives feedback through rewards or penalties, indicating its actions' accuracy rather than explicit correct answers. This method involves a process of exploration, where the algorithm iteratively tests different actions to determine the most effective strategy. Often described as learning from a 'Critic,' the algorithm adjusts its actions based on the feedback received without offering direct solutions or recommendations for the problem. Another important concept in reinforcement learning is exploration-exploitation trade-off. The agent must balance between exploiting the actions that are known to yield high rewards and exploring new actions to potentially discover better strategies. Despite its potential, reinforcement learning also faces challenges such as sample inefficiency, credit assignment, and exploration in high-dimensional state spaces. [5]

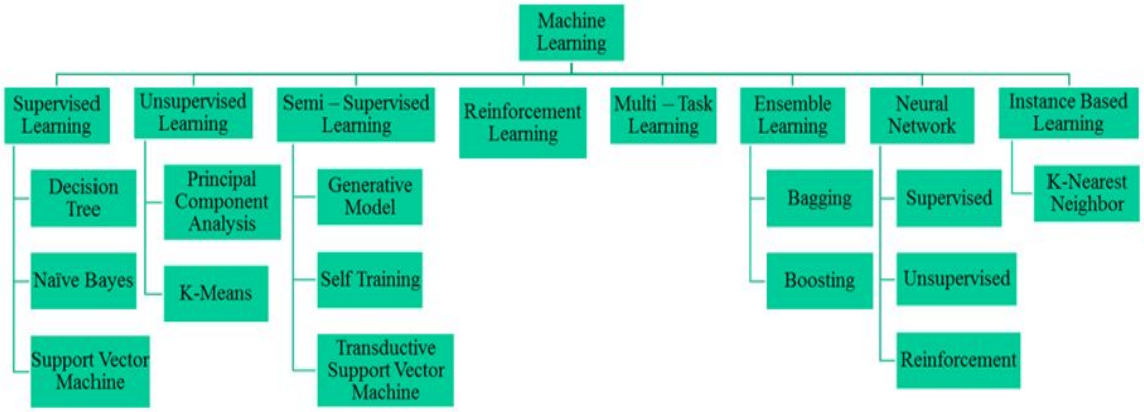


Figure S1. Machine learning methods.

S1.1.4. Instance-Based Learning

In contrast to other machine learning techniques that start with a well-defined target function based on training data, this method begins without specifying any target function. It operates by storing training examples and deferring the generalization process until it needs to classify a new instance. This approach is often called a 'lazy learner' because it does not learn until necessary. Lazy learning methods accumulate a database of training examples, and upon receiving new input data, they compare it against the stored instances using a similarity metric to identify the closest match for making predictions. Lazy learners tailor the target function individually and locally for each new instance needing classification rather than applying a global estimation across all instances. This results in quicker training times but slower prediction times. Algorithms like K-Means, hierarchical clustering are well-known examples of instance-based learning methods.

S1.2. Machine learning algorithms

Although numerous ML algorithms are mentioned in the above section, some commonly used machine learning algorithms are described below.

S1.2.1. Random Forest (RF)

The Random forest (RF) algorithm is a sophisticated approach to supervised machine learning that builds on the concept of classification and regression trees (CART). Unlike the traditional decision tree method that creates just one CART, RF constructs many trees, each based on a random subset of predictors from the pool of features. This strategy effectively addresses the

primary weakness of decision trees: their tendency to fit too closely to the training data, a problem known as overfitting.

In the instance of regression tasks, RF shown in Figure S2 takes the average of the predictions from each tree to arrive at a final prediction value. It is also proficient in equalizing errors within the dataset. However, it is essential to note that the Random Forest algorithm has limitations, particularly when it comes to extrapolating predictions beyond the scope of the training data's range [7]. Consider the training set as $D = \{(x_1.y_1).(x_2.y_2)....(x_m, y_m)\}$ where D_t represents the training data for tree h_t and H^{oob} represents the out-of-bag prediction outcome for sample *x*, thus

$$H^{oob}(x) = \arg\max \sum_{t=1}^{T} (h_t(x) = y) \tag{1}$$

The generalization error for the out-of-bag (OOB) data can be expressed as:

$$\varepsilon^{oob}(x) = \frac{1}{|D|} \sum_{(x,y \in D)} I\left(H^{oob}(x) \neq y\right) \tag{2}$$

The parameter *K* governs the randomness operation of the RF and is typically defined as. [8] To determine the significance of each feature X_i, the variable is randomly permuted. The feature significance is measured as follows:

$$I(X_i) = \frac{1}{B} \sum_{t}^{B} \tilde{O}OBerr_{t^i} - OOBerr_t \tag{3}$$

Here, X_i indicates the permuted i^{th} factor in the factor vector *X*, *B* the number of trees in the Random Forest, $\tilde{O}OBerr_{t^i}$ the model estimation error of the disturbed out-of-bag (OOB) sample consisting of the permuted feature X_i for tree *t*, and $OOBerr_t$ the intact OOB data sample comprising the permuted variable. [9]

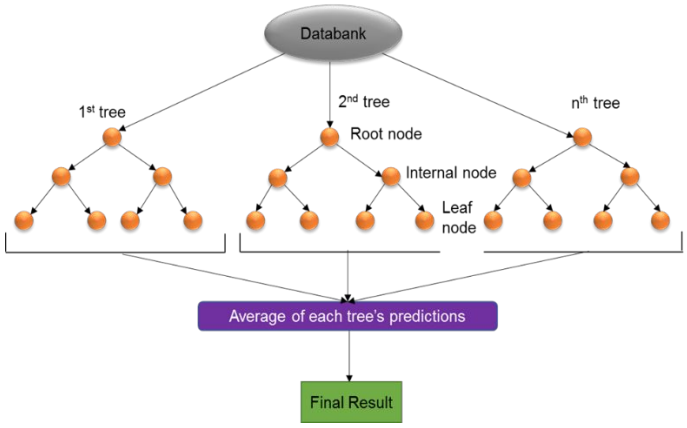


Figure S2. Random forest algorithm.

S1.2.2. Decision Tree (DT)

Decision tree (DT) is a flexible ML algorithm applicable to categorizing data and predicting continuous outcomes. DT (Figure S3) is structured similarly to a tree, with each internal node testing a specific feature, the branches denoting the test results, and the terminal nodes reflecting the outcome or forecast. The algorithm develops the tree through a recursive division of the data into smaller groups, guided by the principle of optimizing information gain or reducing inconsistency at every division point. This method is instrumental in a variety of predictive modeling scenarios.

DT algorithm begins with a single node L_0 which serves as the root. As the learning process unfolds, each newly created node L_q processes a specific subset S_q of the training dataset S. If all elements within the subset S_q belong to the same class, the node is labelled as a leaf, and no further splitting occurs. However, if the elements in S_q represent different classes, the algorithm selects the best attribute for splitting based on a split measure function. Among the available attributes at the current node, the chosen attribute a_i partitions the set of attribute values A^i into two distinct subsets: A_L^i and $A_R^i (A^i = A_L^i \cup A_R^i)$. The choice of A_L^i spontaneously fixes the complementary subset A_R^i so the partition is signified only by A_L^i. The set of all potential partitions of set A^i is represented by V_i. These subsets A_L^i and A_R^i further split the dataset S_q into two disjoint subsets: right $R_q\left(A_L^i\right)$ and left $L_q\left(A_L^i\right)$:

$$L_q\left(A_L^i\right) = \left\{s_j \in S_q \mid \upsilon_j^i \in A_L^i\right\} \tag{4}$$

$$R_q\left(A_L^i\right) = \left\{s_j \in S_q \mid \upsilon_j^i \in A_R^i\right\} \tag{5}$$

The sets $L_q\left(A_L^i\right)$ and $R_q\left(A_L^i\right)$ are determined by the chosen attribute and its partitioned values. Let $p_{L,q}\left(A_L^i\right), p_{R,q}\left(A_L^i\right)$ represents the fraction of data elements from S_q that belong to the subset $R_q\left(A_L^i\right), L_q\left(A_L^i\right)$. Since the fractions $p_{L,q}\left(A_L^i\right), p_{R,q}\left(A_L^i\right)$ are interrelated:

$$p_{R,q}\left(A_L^i\right)=1-p_{L,q}\left(A_L^i\right) \tag{6}$$

Only one of these parameters needs to be considered, for example $p_{L,q}\left(A_L^i\right)$. The portion of elements from $L_q\left(A_L^i\right)$ and $R_q\left(A_L^i\right)$ belonging to class k is denoted by $p_{k,q}$. The impurity measure used in the DT algorithm is the Gini index. For any subset S_q of the training dataset, it is calculated by the formula:[10]

$$Gini\left(S_q\right)=1-\sum_{k=1}^{K}\left(p_{k,q}\right)^2 \tag{7}$$

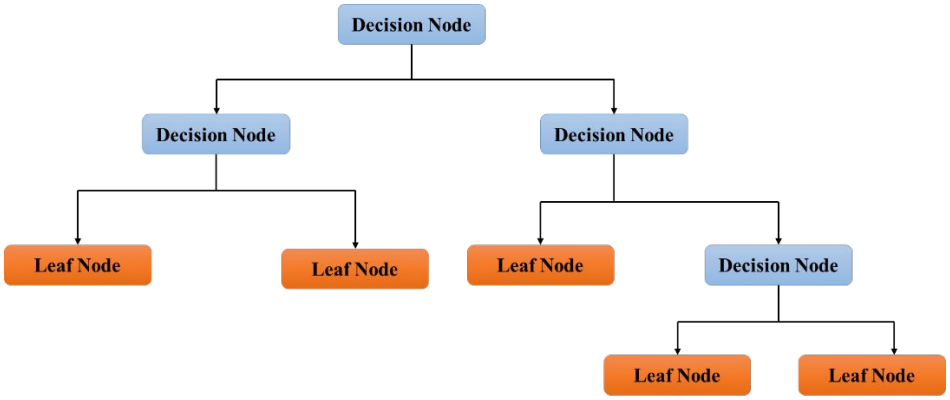


Figure S3. Decision tree algorithm.

Selecting a decision tree over the neural network is motivated by the complex nature of the problem, which entails optimizing six parameters for gas hydrate thermo-kinetics. This model offers a clear and interpretable structure, facilitating the understanding of the key factors influencing the thermo-kinetics of gas hydrates. Machine learning algorithms based on decision-making can effectively handle missing data and are less susceptible to overfitting, making them well-suited for real-world datasets encountered in gas hydrates. Decision trees are faster to train and they require less computational power than neural networks, which are resource-intensive and complex to optimize. Machine learning techniques typically have fewer hyperparameters and are easier to optimize compared to neural networks, which have complex architectures and numerous tuning parameters.

S1.2.3. Gradient Boosting (GB)

Gradient boosting (GB) regression is a method that refines an assembly of simple predictive models, often decision trees, through iteration. Diverging from the random forest (RF) approach, which fits numerous trees to the entire dataset at once, GB builds one tree at a time. Each new tree is trained to accurate the errors made by the preceding ones, focusing on the residuals left behind. The LSBoost algorithm is employed as a GB variant to develop the predictive model in this context. LSBoost enhances the standard gradient boosting method by introducing additional randomization and an innovative regularization feature, that has been proven in various studies to deliver superior results.[11] Given a training dataset $D = \{x_i, y_i\}^N$, the objective of gradient boosting is to approximate $\hat{F}(x)$ a function that maps instances x to their corresponding output values y. This is achieved by minimizing the expected value of a specified loss function $L\left(y, F\left(x\right)\right)$. Gradient boosting constructs an additive approximation of $F^*(x)$ by combining functions in the form of a weighted sum:[12]

$$F_m\left(x\right) = F_{m-1}\left(x\right) + \rho_m h_m\left(x\right) \tag{8}$$

Here, ρ_m is the weight of the m^{th} function, $h_m\left(x\right)$.

The approximation is built iteratively. Initially, a constant approximation of $F^*(x)$ is computed from:

$$F_0\left(x\right) = \arg\min_{\alpha} \sum_{i=1}^{N} L\left(y_i, \alpha\right) \tag{9}$$

Subsequent models are aimed at minimizing

$$\left(\rho_m, h_m, \left(x\right)\right) = \arg\min_{\rho, h} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(x_i\right) + \rho h\left(x_i\right)\right) \tag{10}$$

However, rather than directly solving the optimization problem, each h_m can be viewed as a greedy step in a gradient descent optimization for F^*. For this purpose, each model h_m is trained on a new dataset $D = \left\{x_i, r_{mi}\right\}_{i=1}^{N}$ where the pseudo-residuals r_{mi} is obtained from:

$$r_{mi} = \left[\frac{\partial L\left(y_i, F\left(x\right)\right)}{\partial F\left(x\right)}\right]_{F(x)=F_{m-1}(x)} \tag{11}$$

S1.2.4. Extra Trees (ET)

The extremely randomized trees, called extra trees (ET), is an ensemble learning process based on decision or regression trees. This approach (Figure S4) constructs numerous trees using the entire set of training data. Extra trees stands out from other tree-based ensemble methods by its unique strategy of randomly choosing split points for nodes and using the complete dataset for tree growth rather than a bootstrapped sample. This method builds an ensemble of fully grown trees following a conventional top-down approach without pruning.[13] ET algorithm begins at the root node by selecting a split rule based on a random subset of features and a partially random cut point. This process iterates in each child node until a leaf node is reached. The algorithm relies on three essential parameters: *M,* the number of decision trees in the ensemble, *K* the number of features randomly selected, and n_{min} minimum number of instances required to split a node.

In a formal manner, given a training dataset $X = \{x_1, x_2, ..., x_N\}$ where each sample $x_i = \{f_1, f_2, ..., f_D\}$ is a *D*-dimensional vector with f_j as the feature and $j \in \{1, 2, ..., D\}$ the Extra-Trees algorithm generates *M* independent decision trees. At each decision tree node S_p represents the subset of the training dataset *X.* Then, at each node *p* the Extra-Trees algorithm selects the best split based on S_p and a random subgroup of features.[14]

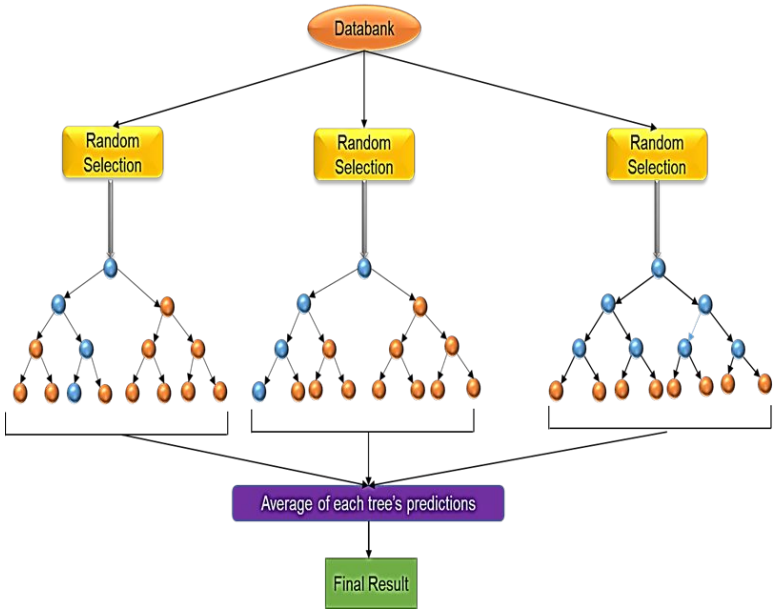


Figure S4. Extra trees algorithm.

Table S1. Comparing four different ML algorithms.

Property	Decision Tree	Random Forest	Gradient Boosting	Extra Tree

Description	Simple, interpretable model with a tree-like structure	Ensemble method using multiple decision trees with voting/averaging	Sequential ensemble method focusing on correcting errors	Ensemble of decision trees with random splits
Training Process	Greedy algorithm to split nodes	Builds multiple trees using bootstrapped data samples	Builds trees sequentially, each correcting the predecessor	Builds multiple trees with randomly selected split points
Complexity	Low	Moderate to High	High	Moderate
Overfitting	High likelihood if not pruned	Reduced compared to a single tree	Can overfit if not properly tuned	Reduced compared to a single tree
Interpretability	High	Lower than a single decision tree	Lower than Random Forest	Lower than a single decision tree
Prediction Speed	Fast	Slower than a single tree	Slower due to sequential nature	Fast
Handling of Overfitting	Pruning techniques can be used	Averaging/voting reduces overfitting	Regularization techniques (e.g., learning rate, max depth)	Random splits reduce variance
Scalability	Moderate	High	Moderate to high	High
Sensitivity to Outliers	High	Lower due to averaging	Sensitive but can be controlled with robust loss functions	Lower due to randomness
Handling of Missing Data	Requires imputation or missing value handling	Can handle missing values using surrogate splits	Can handle missing values using surrogate splits	Can handle missing values using surrogate splits
Memory Usage	Low	High	High	Moderate to high

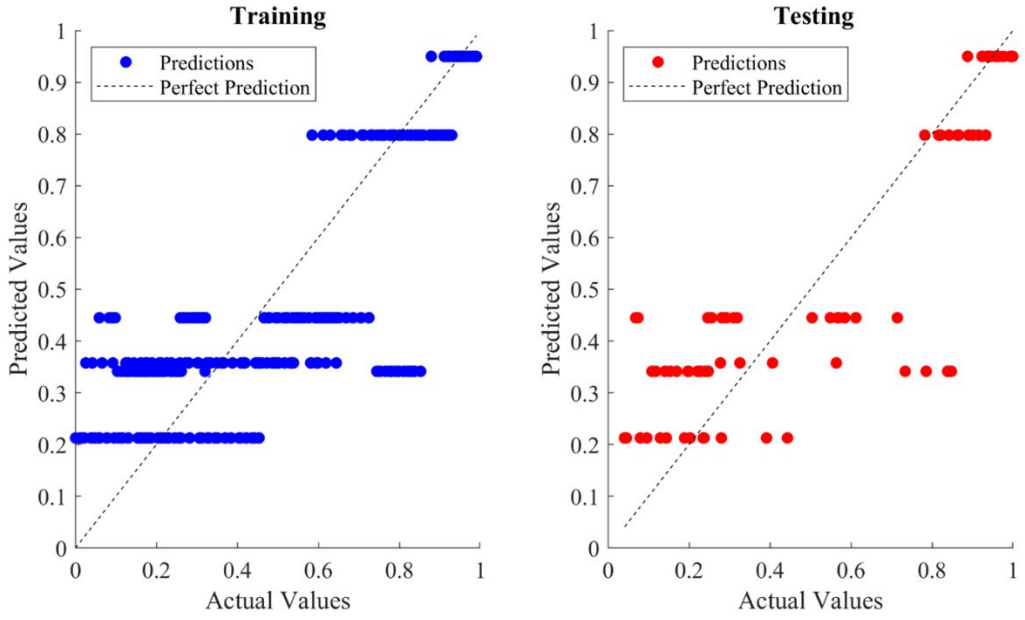
Bias-Variance Trade off	High bias, low variance	Lower bias, higher variance than a single tree	Low bias, high variance but can be tuned	Higher bias, lower variance due to random splits

Table S2. Advantages and disadvantages of four different ML algorithms.

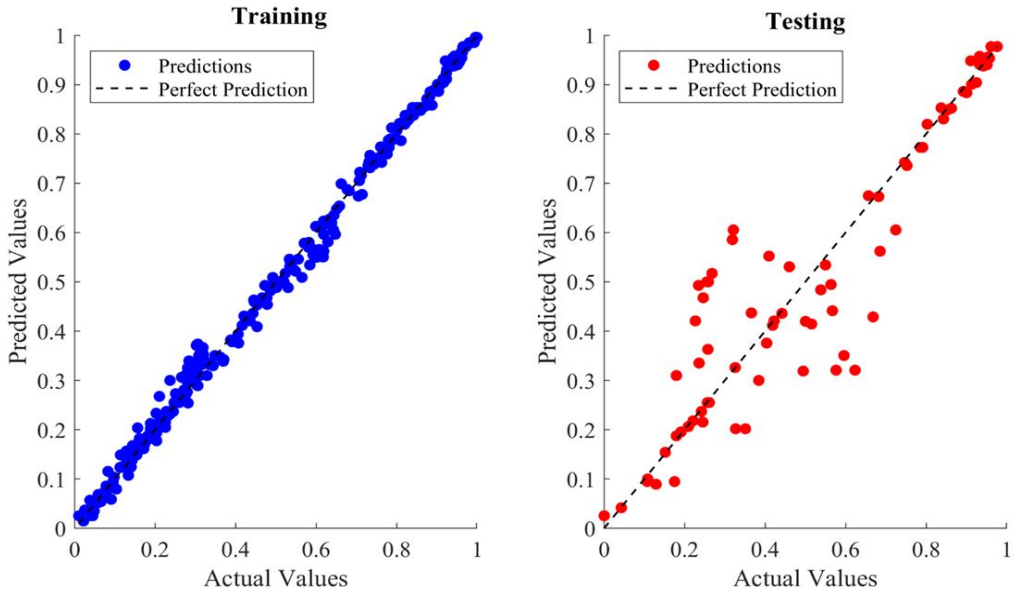
Model	Advantages	Disadvantages
Decision Tree	• Simple and easy to interpret. • Requires little data pre-processing. • Handles both numerical and categorical data. • Fast prediction speed.	• Prone to overfitting. • Sensitive to small variations in the data. • Can create biased trees if some classes dominate. • Low scalability for large datasets
Random Forest	• Reduces overfitting compared to a single decision tree. • Handles large datasets and high-dimensional spaces well. • Provides feature importance. • Robust to noise and outliers.	• Computationally intensive, especially with a large number of trees. • Less interpretable than a single decision tree. • May require more memory and computational resources. • Slower prediction speed due to ensemble nature
Gradient Boosting	• High predictive accuracy. • Can handle various types of data and complex relationships. • Effective for both regression and classification problems. • Can handle missing values using surrogate splits.	• Susceptible to overfitting if not carefully tuned. • Requires more training time compared to Random Forest. • Sensitive to outliers and noisy data. • High complexity and requires careful tuning of hyperparameters.
Extra Tree	• Faster training compared to Random Forest. • Reduces variance, leading to potentially better generalization. • Handles large datasets efficiently.	• Higher bias compared to Random Forest due to randomness in splits. • Less stable predictions due to high randomness.

	• Provides feature importance.	• May require more trees to achieve the same performance as Random Forest. • Less interpretable due to ensemble nature.

S2. Regression plots of training and testing data (Set 1, CO_2 hydrates based data) for ML algorithms

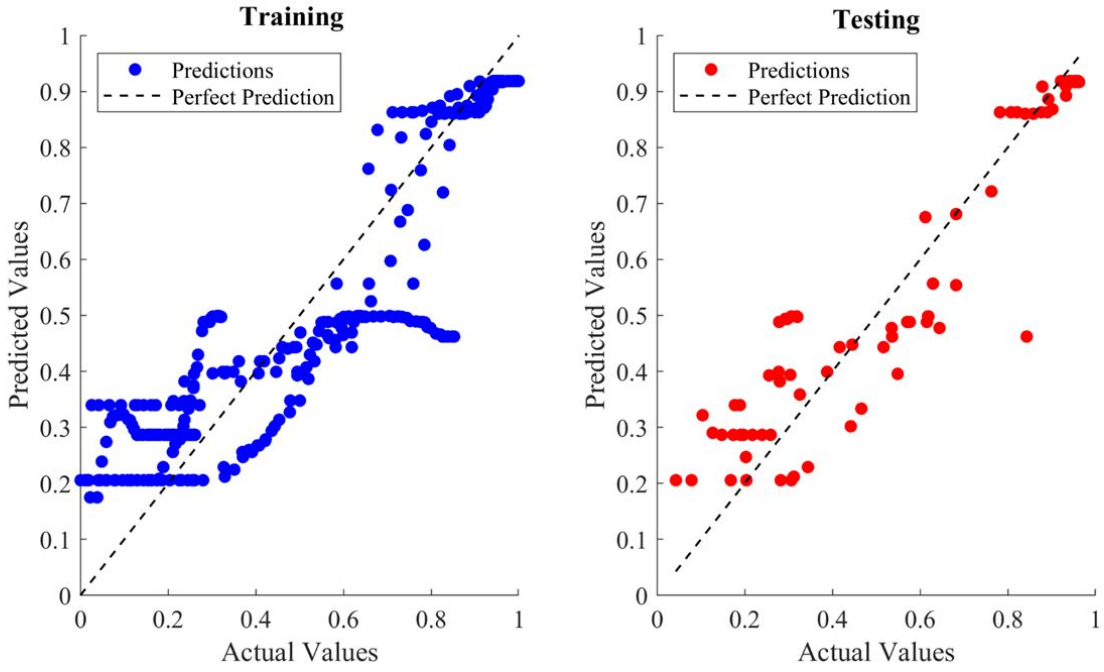


(a)

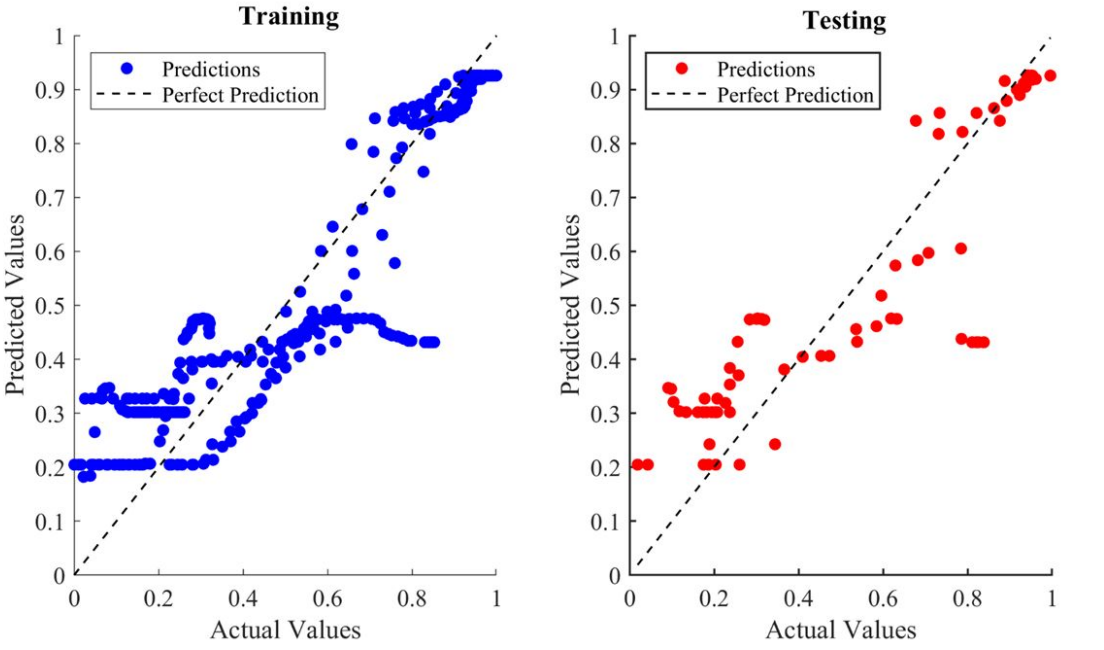


(b)

Figure S5. Regression plot for training and testing data: (a) decision tree (DT), and (b) gradient boosting (GB) regression.



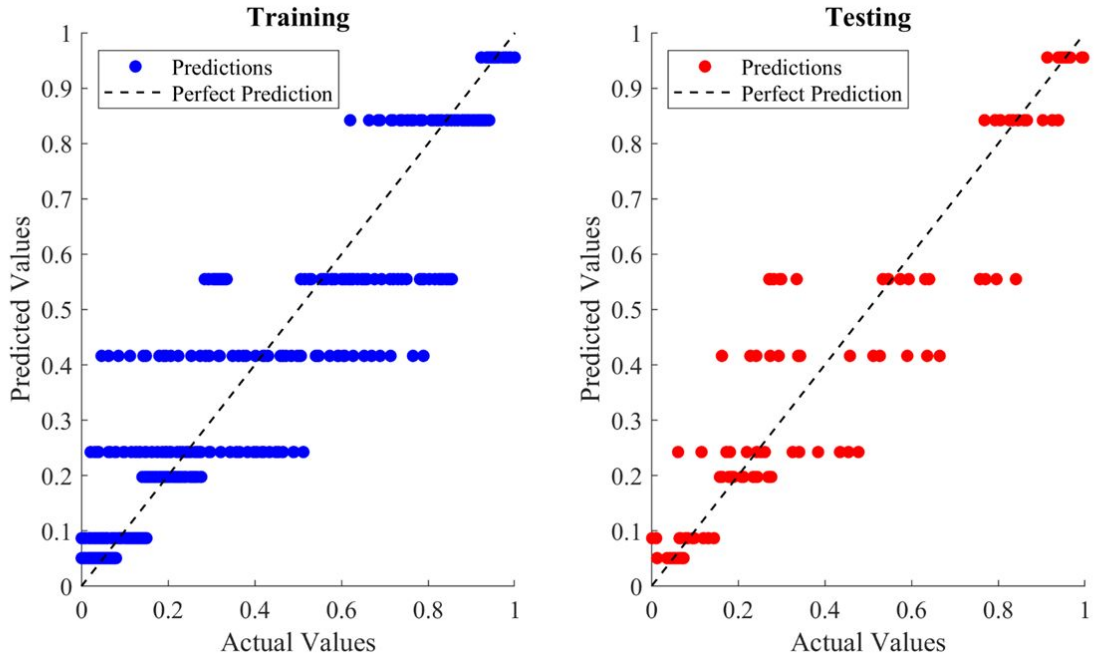
(a)



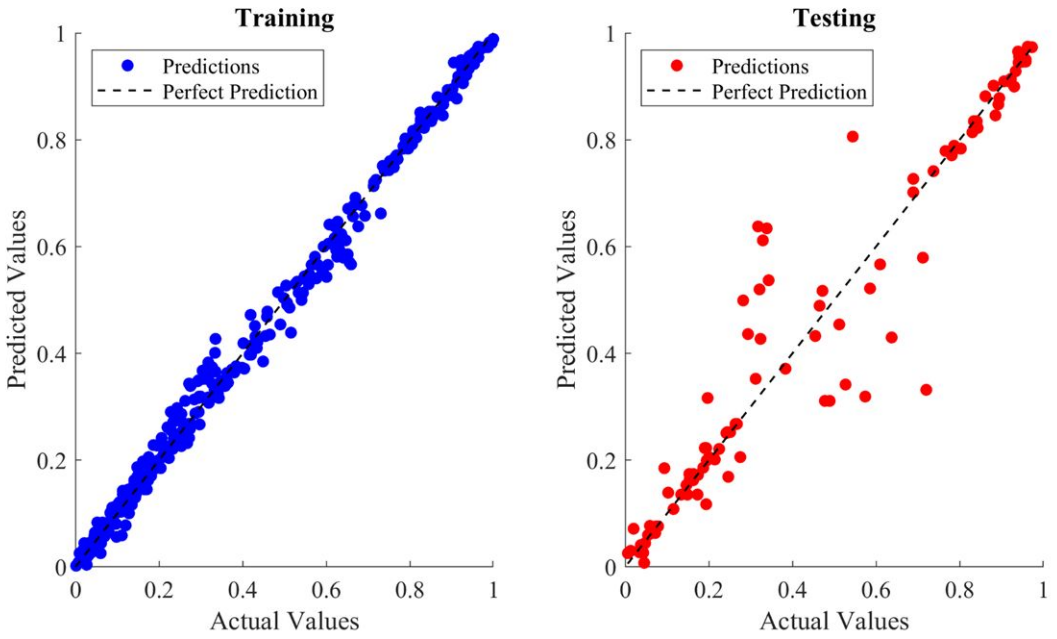
(b)

Figure S6. Regression plot for training and testing data: (a) random forest (RF), and (b) extra trees (ET).

S3. Regression plot of training and testing data (Set 2, CO_2 + CH_4 hydrates based data) for ML algorithms

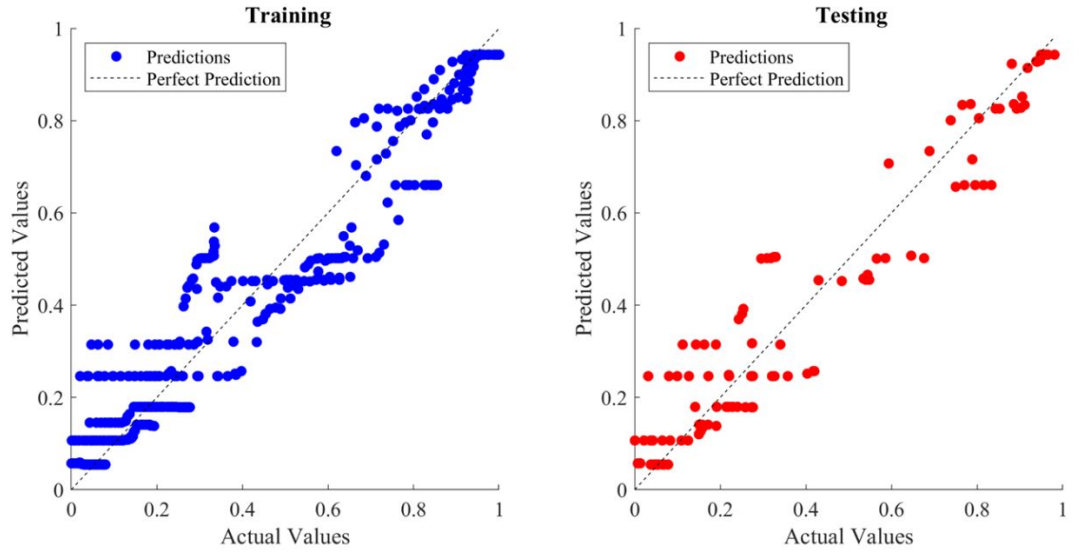


(a)

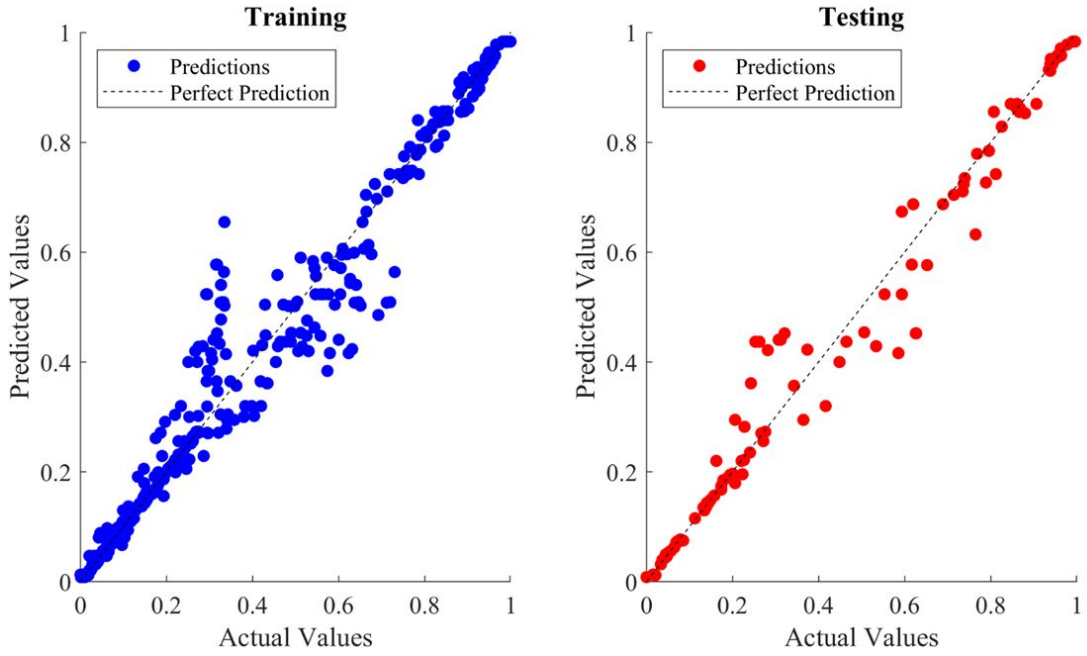


(b)

Figure S7. Regression plot for training and testing data: (a) decision tree (DT), and (b) gradient boosting (GB) regression.



(a)



(b)

Figure S8. Regression plot for training and testing data: (a) random forest (RF), and (b) extra trees (ET).

S4. Sensitivity analysis for optimization of parameters

The six different parameters as well as the hyperparameters of ML model are optimized using sensitivity analysis. The analysis for different ML models for CO_2 gas hydrates dataset (Set 1) is attached below.

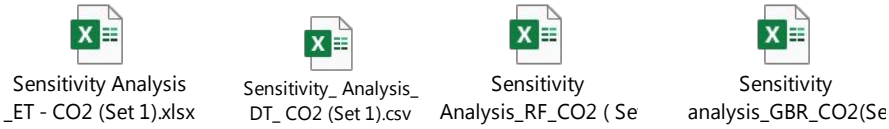


Table S3. Results of K - Fold Cross – validation for CO_2 (Set 1) dataset.

K Fold	Decision Tree			Random Forest			Gradient Boosting			Extra Tree		
	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)
1	0.0128	0.113	86.93	0.0219	0.148	78.06	0.0019	0.038	98.05	0.0152	0.123	84.50
2	0.0139	0.117	86.10	0.0196	0.139	79.95	0.0022	0.041	97.72	0.0154	0.124	84.51
3	0.0129	0.113	87.06	0.0178	0.133	82.01	0.0023	0.043	97.54	0.0156	0.123	84.32
4	0.0141	0.118	86.07	0.0185	0.136	80.97	0.0024	0.043	97.56	0.0157	0.125	83.86
5	0.0138	0.112	85.58	0.0175	0.132	82.57	0.0022	0.042	97.77	0.0154	0.124	84.27

Table S4. Results of K - Fold Cross – validation for CO_2-CH_4 (Set 2) dataset.

K Fold	Decision Tree			Random Forest			Gradient Boosting			Extra Tree		
	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)	MSE	RMSE	R^2(%)
1	0.006	0.076	93.99	0.0112	0.105	92.12	0.0016	0.038	98.44	0.0111	0.105	89.80
2	0.006	0.076	94.09	0.0110	0.104	91.04	0.0018	0.038	98.31	0.0126	0.112	88.93
3	0.009	0.094	91.09	0.0220	0.148	92.32	0.0018	0.038	98.39	0.0122	0.110	88.92
4	0.009	0.094	91.24	0.0154	0.124	91.56	0.0019	0.040	98.21	0.0112	0.105	89.50
5	0.010	0.096	91.01	0.0164	0.128	91.10	0.0018	0.039	98.32	0.0113	0.106	89.48

S5. Regression plot for ML algorithms (with optimized parameters) developed on CO_2 hydrates based data (Set 1)

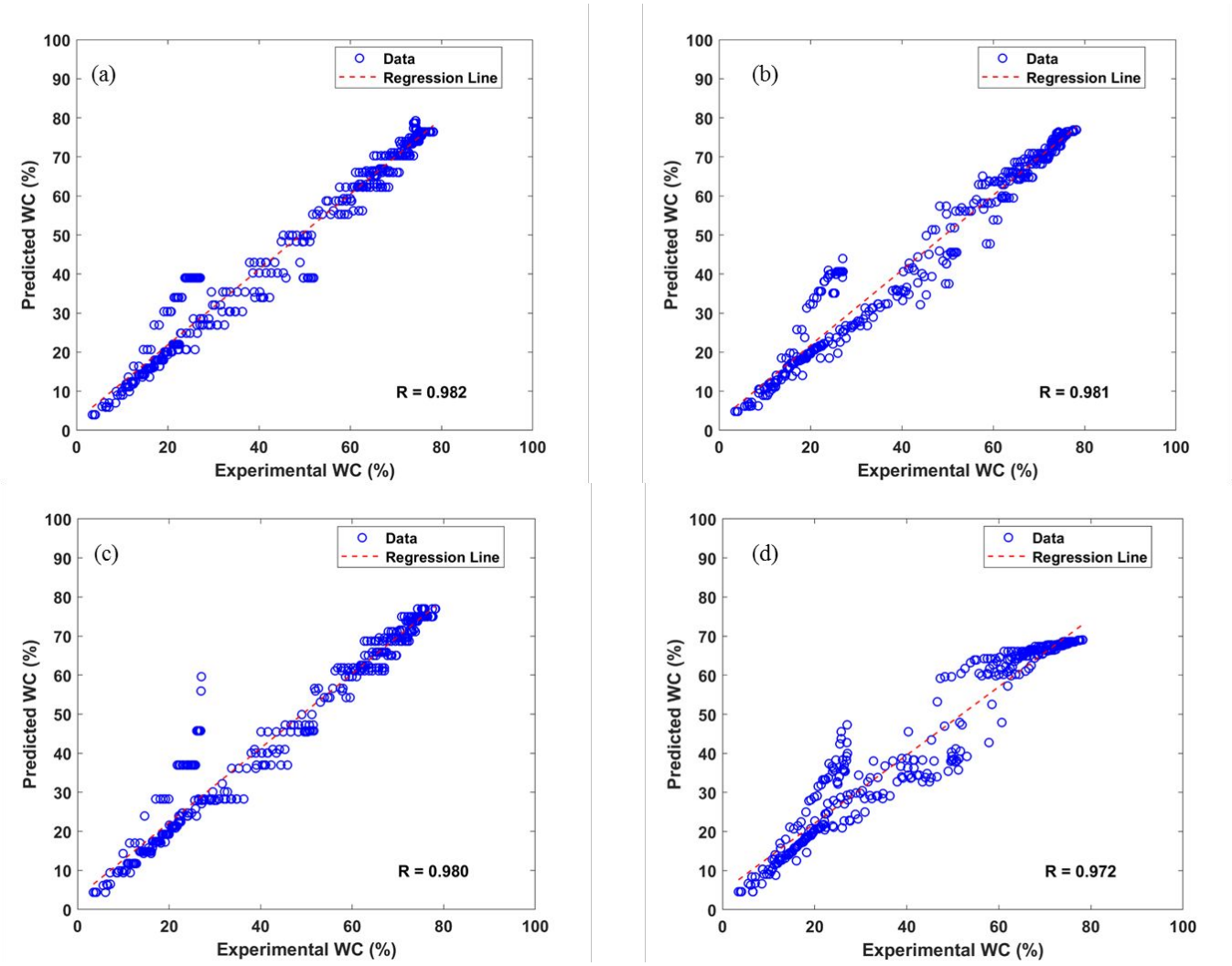


Figure S9. Regression plot of experimental and predicted water conversion percentage during gas hydrate formation for ML algorithms (Set 1): (a) decision tree (DT), (b) gradient boosting (GB), (c) random forest (RF), and (d) extra trees (ET).

S6. Regression plot for ML algorithms (with optimized parameters) developed on CO_2+CH_4 hydrates based data (Set 2)

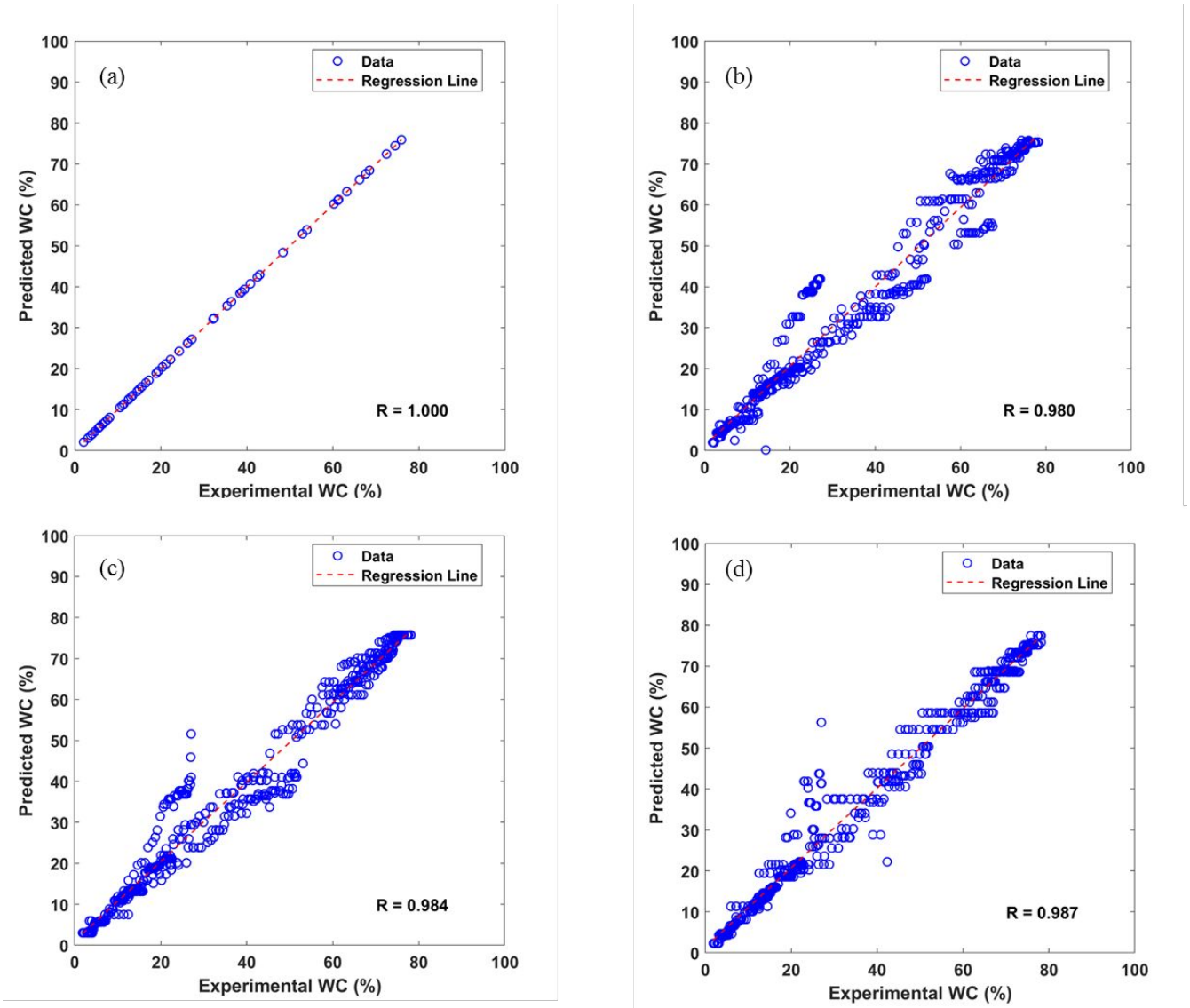


Figure S10. Regression plot of experimental and predicted water conversion percentage during gas hydrate formation for ML algorithms (Set 2): (a) decision tree (DT), (b) gradient boosting (GB), (c) random forest (RF), and (d) extra trees (ET).

References

(1) Burzykowski, T.; Rousseau, A.-J.; Geubbelmans, M.; Valkenborg, D. Introduction to Machine Learning. *Am. J. Orthod. Dentofacial Orthop.* **2023**, *163* (5), 732–734.

(2) El Bouchefry, K.; De Souza, R. S. Learning in Big Data: Introduction to Machine Learning. In *Knowledge Discovery in Big Data from Astronomy and Earth Observation*; Elsevier, 2020; pp 225–249.

(3) El Bouchefry, K.; De Souza, R. S. Learning in Big Data: Introduction to Machine Learning. In *Knowledge Discovery in Big Data from Astronomy and Earth Observation*; Elsevier, 2020; pp 225–249.

(4) Fong, A. C. M. Welcome Message from the Editor-in-Chief. *J. Adv. Inf. Technol.* **2010**, *1* (1), 1–1.

(5) Alzubi, J.; Nayyar, A.; Kumar, A. Machine Learning from Theory to Algorithms: An Overview. *J. Phys. Conf. Ser.* **2018**, *1142*, 012012.

(6) Badillo, S.; Banfai, B.; Birzele, F.; Davydov, I. I.; Hutchinson, L.; Kam-Thong, T.; Siebourg-Polster, J.; Steiert, B.; Zhang, J. D. An Introduction to Machine Learning. *Clin. Pharmacol. Ther.* **2020**, *107* (4), 871–885.

(7) Xu, H.; Jiao, Z.; Zhang, Z.; Huffman, M.; Wang, Q. Prediction of Methane Hydrate Formation Conditions in Salt Water Using Machine Learning Algorithms. *Comput. Chem. Eng.* **2021**, *151*, 107358.

(8) Breiman, L. Random Forest. *Mach. Learn.* **2001**, *45* (1), 5–32.

(9) Amiri-Ramsheh, B.; Safaei-Farouji, M.; Larestani, A.; Zabihi, R.; Hemmati-Sarapardeh, A. Modeling of Wax Disappearance Temperature (WDT) Using Soft Computing Approaches: Tree-Based Models and Hybrid Models. *J. Pet. Sci. Eng.* **2022**, *208*, 109774.

(10) Rutkowski, L.; Jaworski, M.; Pietruczuk, L.; Duda, P. The CART Decision Tree for Mining Data Streams. *Inf. Sci.* **2014**, *266*, 1–15.

(11) Jiao, Z.; Hu, P.; Xu, H.; Wang, Q. Machine Learning and Deep Learning in Chemical Health and Safety: A Systematic Review of Techniques and Applications. *ACS Chem. Health Saf.* **2020**, *27* (6), 316–334.

(12) Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A Comparative Analysis of Gradient Boosting Algorithms. *Artif. Intell. Rev.* **2021**, *54* (3), 1937–1967.

(13) Yarveicy, H.; Ghiasi, M. M. Modeling of Gas Hydrate Phase Equilibria: Extremely Randomized Trees and LSSVM Approaches. *J. Mol. Liq.* **2017**, *243*, 533–541.

(14) Camana Acosta, M. R.; Ahmed, S.; Garcia, C. E.; Koo, I. Extremely Randomized Trees-Based Scheme for Stealthy Cyber-Attack Detection in Smart Grid Networks. *IEEE Access* **2020**, *8*, 19921–19933.

.....