Advancing Machine Learning in Industry 4.0: Benchmark Framework for Rare-event Prediction in Chemical Processes

Vikram Sudarshan and Warren D. Seider*
Department of Chemical and Biomolecular Engineering
University of Pennsylvania
Philadelphia, PA 19104-6393

ABSTRACT

Previously, using forward-flux sampling (FFS) and machine learning (ML), we developed multivariate alarm systems to counter rare un-postulated abnormal events. Our alarm systems utilized ML-based predictive models to quantify committer probabilities as functions of key process variables (e.g., temperature, concentrations, and the like), with these data obtained in FFS simulations. Herein, we introduce a comprehensive benchmark framework for rare-event prediction, comparing ML algorithms of varying complexity, including Linear Support-Vector Regressor and k-Nearest Neighbors, to more sophisticated algorithms, such as Random Forests, XGBoost, LightGBM, CatBoost, Dense Neural Networks, and TabNet. This evaluation uses comprehensive performance metrics: *RMSE,* model training, testing, hyperparameter tuning and deployment times, and number and efficiency of alarms. These balance model accuracy, computational efficiency, and alarm-system efficiency, identifying optimal ML strategies for predicting abnormal rare events, enabling operators to obtain safer and more reliable plant operations.

Keywords: Benchmark Framework, Machine Learning, Rare Abnormal Events, Alarm-system Efficiency

*Corresponding author
Email address: seider@seas.upenn.edu (W.D. Seider)

1. INTRODUCTION

1.1. Progress of Industrial Revolution and Challenge of Rare, Un-postulated Abnormal Events

Over the past few centuries, several industrial revolutions have transformed the chemical and manufacturing industries. These began with the first industrial revolution in the mid-1700s, focused on mechanization through water and steam engines, and railroads (Crafts, 2011; Mohajan, 2019); then moved to the second industrial revolution in the mid-to-late 1800s, focused on electrification, ramping-up manufacturing and improving efficiency by introducing assembly lines (Mokyr and Strotz, 1998); then advanced to the third industrial revolution in the late-1900s, introducing automation technologies (e.g., distributed control systems; , DCS), computers and electronics (Mohajan, 2021; Naboni and Paoletti, 2015); and then proceeded to the current Industry 4.0 vision of digitalization, consisting of path-breaking technologies, such as the internet-of-things (IoT) (Belli et al., 2019; Soori et al., 2023), artificial intelligence and machine learning (AI/ML) (Bécue et al., 2021; Candanedo et al., 2018; Dingli et al., 2021), cybersecurity and cyber-physical systems (Culot et al., 2019; Ervural and Ervural, 2018; Hashimoto et al., 2013), and, big-data analytics and cloud computing (Gokalp et al., 2016; Kim, 2017). Numerous perspectives are anticipating Industry 5.0, with foci on customization and sustainability, consisting of technologies such as human-computer interaction, collaborative robotics, and, augmented reality and mixed reality (AR/MR) (Barata and Kayser, 2023; Demir et al., 2019; Ghobakhloo et al., 2023; Raja Santhi and Muthuswamy, 2023).

Remarkably, despite these breakthroughs, the chemical manufacturing industries struggle to prevent safety accidents (e.g., thermal runaways, release of flammables, and chemical spillage) and reliability failure events (e.g., poor product quality and related financial losses). The former have resulted in numerous fatalities, including: the Pemberton Mill accident in 1860, the Grover shoe factory disaster in 1905, the Flixborough disaster in 1974 (Hailwood, 2016), the Bhopal gas tragedy in 1984 (Broughton, 2005; Gupta, 2002; Sriramachari, 2004), the Chernobyl disaster in 1986 (Saenko et al., 2011), the BP Texas City refinery

explosion in 2005 (Holmstrom et al., 2006), the Deepwater Horizon oil spill in 2010 (Beyer et al., 2016), and the Fukushima disaster in 2011 (Labib and Harris, 2015). Often, such catastrophic accidents are triggered by rare, *un-postulated* abnormal events unidentified during prior HAZOP studies, and unknown at the time of occurrence. From the perspective of chemical process safety, rare events are defined as "low-frequency high-consequence" events (Aven, 2020). Additionally, there are very few occurrence data, making it challenging to predict their likelihood using data-driven quantitative techniques. While extensive near-miss data often help to prevent accidents, more accurate estimates are needed. Moreover, routine alarm management systems, created using HAZOP studies, are often unable to identify such abnormal rare events; e.g., the root-cause of the BP Texas City refinery explosion was not identified during HAZOP studies (U.S. Chemical Safety and Hazard Investigation Board, 2007). While automated Safety Instrumented Systems (SIS) are usually successful in preventing accidents through interlock activation, they contribute to plant reliability issues (causing shutdowns, maintenance, and start-up), resulting in production-time and financial losses. **Given these numerous challenges, there is a strong motivation to develop enhanced multivariate alarm systems for identifying and handling these rare un-postulated abnormal events more-efficiently – enabling operators to improve plant safety and reliability.**

1.2. Artificial Intelligence and Machine Learning (AI/ML) for Quantitative Analyses of Rare Events

AI/ML is one of the cornerstones of Industry 4.0 vision for improved automation through digital transformation. Over the past decade, there has been an exponential rise in AI/ML research across several scientific domains, including chemical engineering applications: drug discovery (Lavecchia, 2015; Vamathevan et al., 2019), catalysis (Kitchin, 2018; Toyao et al., 2020), materials science (Morgan and Jacobs, 2020; Wei et al., 2019), computational fluid dynamics (Hanna et al., 2020; Kochkov et al., 2021), molecular dynamics (Gastegger et al., 2017; Wang et al., 2020), process monitoring and fault detection (Arunthavanathan et al., 2022; Harkat et al., 2020), to name a few. With respect to quantitative estimation of rare-events for chemical process safety, AI/ML-based techniques have been developed; a parametric

reduced-order modeling approach was developed to estimate and analyze the consequence of rare abnormal events, using the k-Nearest Neighbors ML algorithm, and demonstrated on a cardon dioxide release study (Kumari et al., 2021). Additionally, optimal ML algorithms were applied to predict and analyze the root-causes of occupational safety events (Sarkar et al., 2019). In related work, three categories of classification ML algorithms, including wide, deep, and wide and deep, were introduced and analyzed using accident data for severity predictions (Tamascelli et al., 2022). Moreover, a novel anomaly detection-based classification algorithm was developed using real-time data from industrial processes (Quatrini et al., 2020).

Despite significant advances, the utilization of ML algorithms for prediction of rare abnormal events presents significant concerns. From amongst a vast choice of ML techniques, it is crucial to select an algorithm most relevant to the target application. Additionally, most ML algorithms developed for rare events are purely data-driven, based on data from process historians, accident data, or alarm databases. And, due to the scarcity of data for truly rare events, data quality is a concern, given that ML model performance relies heavily on such data (Budach et al., 2022; Jain et al., 2020). **Given this lack of occurrence data, it is important to integrate AI/ML-based techniques with efficient simulation-based techniques (e.g., path-sampling), capable of identifying and generating pathways for rare un-postulated abnormal events.**

1.3. Benchmark Analyses of ML Algorithms

With the challenge in selecting relevant algorithms, benchmark analyses of ML algorithms are ubiquitous across several scientific domains having access to open-source databases. A large-scale benchmark framework, MoleculeNet, was developed for benchmarking ML algorithms for molecular datasets, including data for over 700,000 compounds (Wu et al., 2018). Similar analyses and comparisons among several ML algorithms have been conducted for traffic-sign recognition (Stallkamp et al., 2012), healthcare datasets (Purushotham et al., 2018), federated learning (He et al., 2020), scientific machine learning (Thiyagalingam et al., 2022), detection of software defects (Aleem et al., 2015), time-series forecasting

(Pfisterer et al., 2021; Xie and Wang, 2020), and cancer research (Feltes et al., 2019), to name a few. Such rigorous benchmark analyses have also been extended specifically for tabular data –; the most common data format utilized across several scientific domains (Shwartz-Ziv and Armon, 2022). Many of these studies report that for supervised learning tasks (regression and classification) using tabular data, gradient-boosting frameworks (e.g., XGBoost, CatBoost, LightGBM) outperform more-complex neural network-based, deep-learning architectures, achieving comparable or superior accuracies at lower computational costs (Borisov et al., 2022; Grinsztajn et al., 2022; Shwartz-Ziv and Armon, 2022; Uddin and Lu, 2024). More-recently, in related research, a comprehensive survey was conducted for predicting rare-events – considering data, preprocessing, algorithmic techniques, and evaluations (Shyalika et al., 2023). While most studies include several datasets and algorithms, it is very challenging to extend these for data concerning safety and reliability of chemical processes – due to lack of occurrence data accompanying such rare-events. Often, datasets pertaining to rare-events are highly imbalanced; with the number of instances indicative of rare-events significantly less than both normal and near-miss instances. **Additionally, apart from model accuracies/errors and computational costs, it is also crucial to analyze the impact of ML algorithms on alarm-system efficiency; e.g., the number and efficiency of alarms annunciated in identifying abnormal behavior accurately – a missing component in existing benchmark studies.**

1.4. Prior Research: Developing Improved, Multivariate Alarm Systems Using Forward-Flux Sampling and Machine Learning

Given the limitations of HAZOP-based alarm management systems in identifying and mitigating rare un-postulated events, in previous research, we developed improved, novel, multivariate alarm systems using *forward-flux sampling (FFS)* and *machine learning* – based on random statistical "noise"-induced perturbations in one or more process variables that ultimately result in rare un-postulated abnormal shifts from normal to undesirable (*unsafe* or *unreliable*) regions (Sudarshan et al., 2023, 2021). Our alarm systems utilize ML-based algorithms that predict the *committer probability* as a function of the process variables. Then, to enhance the quality and efficiency of the alarm systems, we developed an integrated

framework for alarm rationalization and dynamic risk analyses (Sudarshan et al., 2024a). First, our techniques were demonstrated successfully for a relatively simple exothermic CSTR process model. Then, we improved our methods for more-complex polymerization CSTRs, resulting in *dynamic, bidirectional* multivariate alarm systems based on real-time predictions of committer probabilities, using more-advanced nonparametric ML algorithms. This addressed the *decision-science* component of risk assessment and machine learning; given predictions by the ML algorithms, determining the actionable strategies for reducing the real-time committer probabilities (Sudarshan et al., 2024b).

1.5. Benchmark Analyses of ML Algorithms for Rare Abnormal Events

Herein, we introduce a comprehensive framework for benchmark analyses, comparing several ML algorithms, of varying complexities, for un-postulated rare-event predictions of chemical process models. We begin with Linear Support-Vector Regressor (Linear SVR), k-Nearest Neighbors (kNNs), and move to more-complex algorithms, including: gradient-boosted decision trees (XGBoost, LightGBM, CatBoost) and deep-learning approaches (dense neural networks and TabNet). Two chemical process models are considered; a PI-controlled exothermic CSTR, and a PID-controlled polystyrene CSTR, using five tabular datasets for committer probability-process variables data generated using the branched-growth variant of FFS (BGFFS). In our evaluation, several metrics are considered: *RMSE*; clock-times recorded for training, testing, hyperparameter tuning and model deployment; and factors affecting alarm systems, *number* and *efficiency* of multivariate alarms activated in real-time based on the predictions provided by each ML algorithm. By considering diverse evaluation metrics (with alarm efficiency being novel when benchmarking ML algorithms), we seek to identify optimal ML strategies to predict and handle rare un-postulated abnormal events, thereby, improving overall safety and reliability.

2. MATERIALS AND METHODS

2.1. Overview of Key Steps

Figure 1 provides an overview of the steps and methods utilized in this paper, with the steps described in subsequent sections.

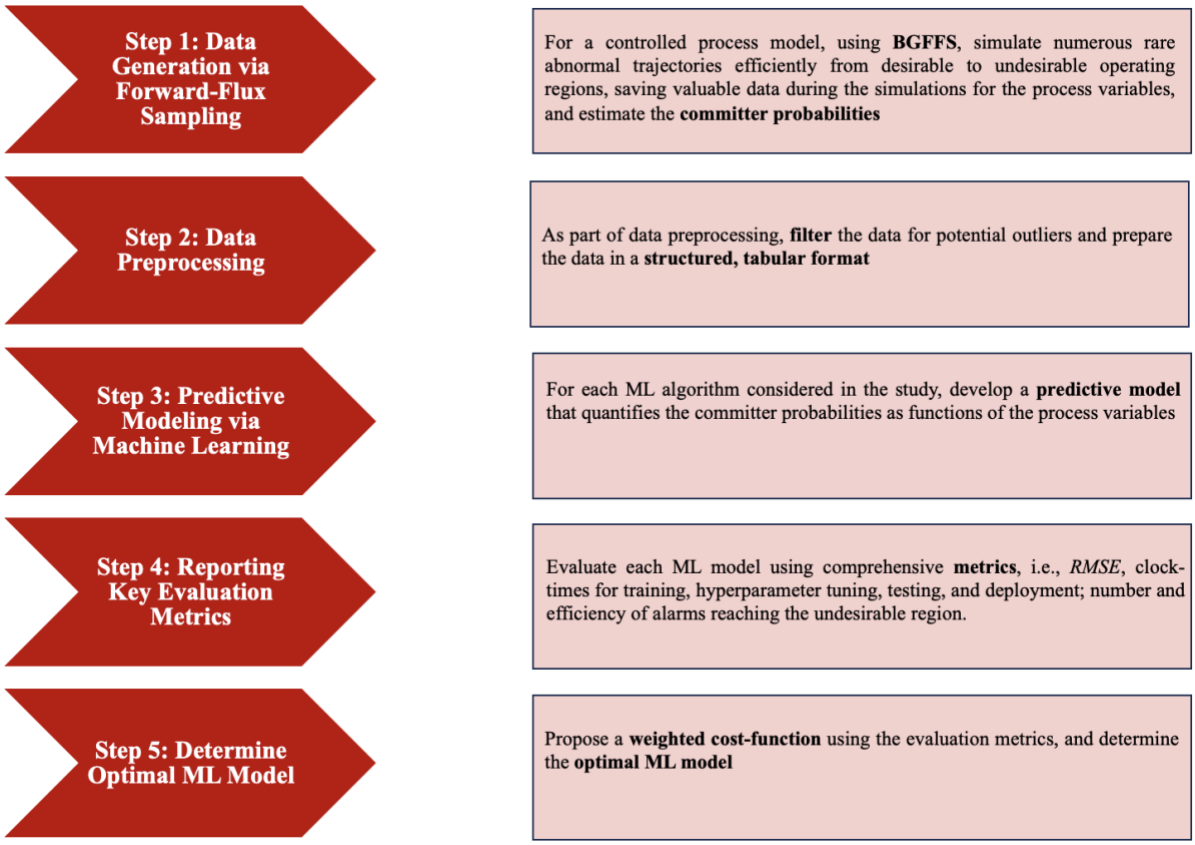


Figure 1. Overview of Key Steps

2.2. Step 1: Data Generation via Forward-flux Sampling

Path-sampling algorithms are Markov-Chain Monte-Carlo (MCMC)-based techniques, utilized routinely in molecular dynamics (MD) to analyze and simulate rare events from chemical reactants to products, including: crystal nucleation of hard spheres (Filion et al., 2010) and sodium chloride (Jiang et al., 2018), stochastic nonequilibrium systems (Allen et al., 2006), methane hydrate nucleation (Arjun and Bolhuis,

2023; Bi and Li, 2014), and the like. In MD, a rare event is an event whose initiation time (time taken to initiate the rare event) is multiple orders of magnitude greater than its duration (Borrero and Escobedo, 2007; Hartmann et al., 2014), In previous research, as a first application of path-sampling algorithms for analyzing rare events for chemical process safety, Moskowitz et al. (2018) introduced transition-path sampling (TPS) [developed originally for MD by Bolhuis et al. (2002), Dellago et al. (2002, 1998)], demonstrated on an exothermic CSTR and an air separation unit (ASU). To overcome the computational limitations of TPS, Sudarshan et al. (2021) introduced forward-flux sampling (FFS) [developed originally for MD by Allen et al. (2006)]. Note that FFS is from the same family of path-sampling algorithms as TPS, simulating rare un-postulated trajectories more-efficiently in a forward, piecewise manner, with the direct variant, DFFS, introduced initially. More recently, the branched-growth variant of FFS (BGFFS) was utilized, generating trajectories more-suitable for committer analyses [conducted previously in MD by Borrero and Escobedo (2007); Peters and Trout (2006)], resulting in improved, multivariate alarm systems for a P-only controlled exothermic CSTR (Sudarshan et al., 2024a, 2023) and a PID-controlled polystyrene CSTR (Sudarshan et al., 2024b). The steps involved in the BGFFS algorithm, shown schematically in Figure 2, include:

i) Define the initial desirable basin A and terminal undesirable basin B.

ii) Pick a suitable order parameter variable, λ; typically, this is a process variable that has a strong influence on the process dynamics, capturing process deviations more-rapidly than other variables, and is not perturbed significantly using statistical noise; e.g., the reactor temperature.

iii) Based on the chosen λ, divide the space between the two basins into finite interfaces: $\lambda_0, \lambda_1 \ldots \lambda_n$, where λ_0 and λ_n represent the bounds for basins A and B. Note that *n* is the number of interfaces.

iv) Simulate a long initial trajectory that generates finite crossings across λ_0; if required, repeat this step for multiple trajectories to generate sufficient crossing points, with valuable data for all process variables being saved at every crossing point.

v) Compute the initial rate of transition across λ_0, r_0, as the total crossings divided by the total time spent in basin A by all the initial trajectories.

vi) Select a crossing point from among the saved crossings across λ_0 and simulate m_0 trajectories from that point, each of which continues until λ_1 is crossed. Save the variables at all such crossing points.

vii) Simulate m_1 trajectories from every crossing point across λ_1 that generate crossing points across λ_2. Save the variables at all such crossing points.

viii) Iterate step vii) for all subsequent interfaces till λ_n; stated differently, simulate m_i trajectories from all crossing points at λ_i that continue until λ_{i+1} is reached; save the variables at all such crossing points at λ_{i+1}; $\forall\ i = 2, 3 \ldots n-1$.

ix) Compute the overall transition probability of reaching basin B from basin A:

$$p_{\mathrm{A}\to\mathrm{B}} = \frac{N(\lambda_n|\lambda_0)}{\prod_{i=0}^{n-1} m_i} \tag{1}$$

where $N(\lambda_n|\lambda_0)$ is the number of branches that reach basin B (from λ_{n-1}) and $\prod_{i=0}^{n-1} m_i$ are the total possible number of branches.

x) Compute the overall rate of transition, $r_{\mathrm{A}\to\mathrm{B}}$, as the product of r_0 and $p_{\mathrm{A}\to\mathrm{B}}(\lambda_n|\lambda_0)$.

xi) Repeat steps iv) – x) for other crossing points at λ_0 and compute the average overall probability and rate of transition: $\bar{p}_{\mathrm{A}\to\mathrm{B}}$ and $\bar{r}_{\mathrm{A}\to\mathrm{B}}$.

Note that every crossing point generated during the BGFFS algorithm, with variables **x**, has an associated committer probability, $p_{\mathrm{B}}(\mathbf{x})$, defined as the probability of a trajectory fired from that point reaches or "commits" to the terminal basin B. The committer probabilities are computed recursively as (Borrero and Escobedo, 2007):

$$p_j{}^i(\lambda_{i+1}|\lambda_i) = \frac{N_j{}^i}{m_i} \tag{2}$$

$$p_{Bj}{}^{i} = \; p_j{}^{i}(\lambda_{i+1}|\lambda_i) \times \frac{\sum_{k=1}^{N_j{}^{i}} p_{Bk}{}^{i+1}}{N_j{}^{i}} = \frac{\sum_{k=1}^{N_j{}^{i}} p_{Bk}{}^{i+1}}{m_i}; \quad i = n-1, n-2, \ldots 0 \tag{3}$$

where $p_j{}^{i}$ is the probability for a trajectory initiated from a point j at λ_i to reach the next interface, λ_{i+1}; $N_j{}^{i}$ is the number of successful trajectories reaching λ_{i+1} from that point; m_i is the total number of trajectories initiated from that point; and $p_{Bj}{}^{i}$ is the committer probability for that point. For example calculations, please refer to Sudarshan et al. (2024b, 2023). Additionally, note that depending on the process parameter selected, initially as the *response-action variable* (a variable that is varied in real-time in response to alarms), the BGFFS algorithm and p_B calculations are repeated for multiple discrete values of the response-action variable. Hence, this makes the response-action variable *discrete-valued*, whereas, the other variables saved during the BGFFS algorithm and the estimated p_B are *continuous-valued*. Additionally, note that BGFFS enables us to simulate datasets that represent a wide and sufficient volume of rare-event pathways, enabling ML algorithms to train effectively despite potential shifts in distributions.

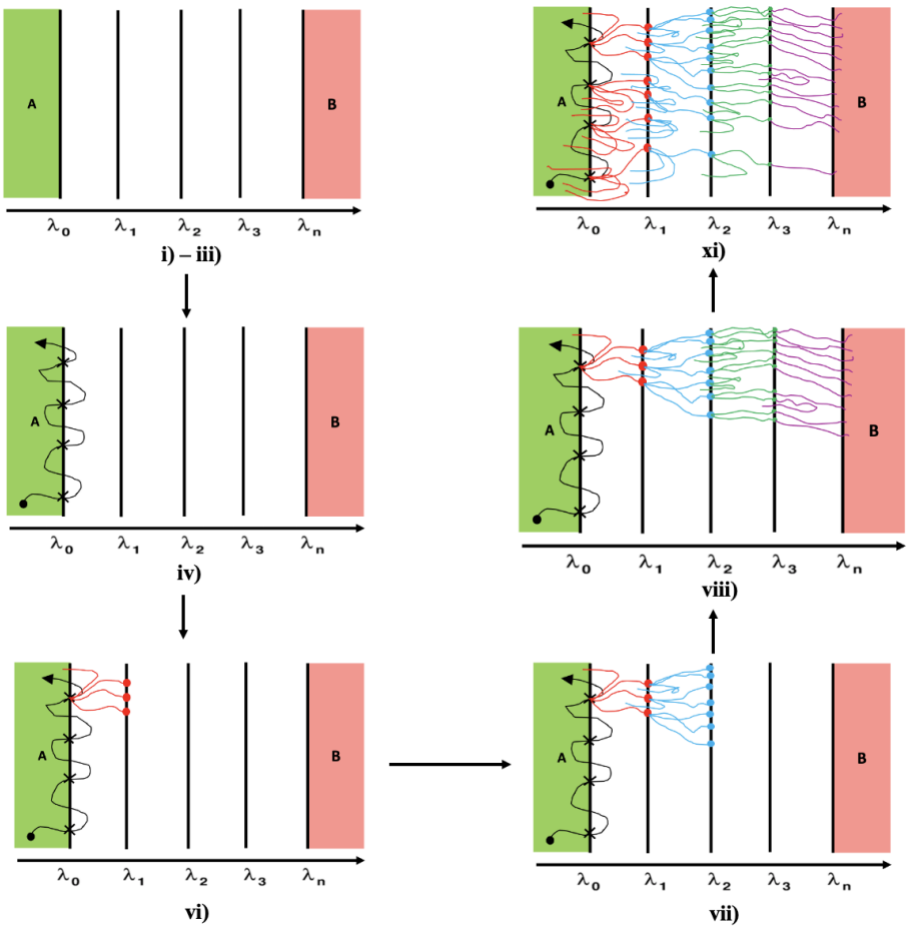


Figure 2. Schematic showing key steps for simulating abnormal trajectories using BGFFS algorithm (refer to the points in Section 2.2.

2.3. Step 2: Data Preprocessing

As part of preprocessing, the p_B – process variables data generated during the BGFFS algorithm are filtered to remove outliers and structured in a clean, tabular format, with the process variables (e.g., temperature, concentration, and the like) as the input variables, and p_B as the dependent variable. Note that during BGFFS, due to statistical noise-induced random perturbations, a wide distribution of p_B is obtained for crossing points across each order parameter interface, λ_i. Hence, to improve the predictions provided by

the ML models, it is important to incorporate data filtering for these p_B. Herein, simple filtering techniques are utilized to retain the p_B centered around its mean; stated differently, only those data are retained that satisfy:

$$\overline{p_{B,i}} - c_i\sigma_i \leq p_{B,i} \leq \overline{p_{B,i}} + c_i\sigma_i \quad (4)$$

where $\overline{p_{B,i}}$ and σ_i are the mean and standard deviation of the p_B for crossing points generated across λ_i; and c_i is a filter factor, determined experimentally, such that neither too many nor too few data are filtered. Hence, post preprocessing, the data are organized in a clean, tabular format, as shown in Table 1. Note that additional preprocessing steps may be required depending on each ML algorithm.

Table 1. Schematic for Tabular Data in our Analyses

$\boldsymbol{p_B}$	$\boldsymbol{X_1}$	$\boldsymbol{X_2}$	$\boldsymbol{X_3}$	$\boldsymbol{X_4}$
$p_{B,1}$	$X_{1,1}$	$X_{2,1}$	$X_{3,1}$	$X_{4,1}$
$p_{B,2}$	$X_{1,2}$	$X_{2,2}$	$X_{3,2}$	$X_{4,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$p_{B,N_{\text{samples}}}$	$X_{1,N_{\text{samples}}}$	$X_{2,N_{\text{samples}}}$	$X_{3,N_{\text{samples}}}$	$X_{4,N_{\text{samples}}}$

2.4. Step 3: Predictive Modeling via Machine Learning

Post data generation and preprocessing, using supervised machine learning, models are developed that predict p_B for given process variables; stated differently, a *regression* problem is solved, given that p_B is continuous-valued. For each ML algorithm considered in this benchmark study, model development involves three steps:

I) Data Splitting: The preprocessed tabular data are divided into *training* and *testing* data, using *randomized* 70%-30% splits, as done routinely in practice (Bichri et al., 2024; Kahloot and Ekler, 2021; Vrigazova, 2021).

II) Hyperparameter Optimization with Cross-Validation: Typically, ML models consist of two entities: hyperparameters to be optimized before training; and model training parameters learned during training. The predictive performance of ML models is extremely sensitive to the choice of hyperparameters – hence, these need to be optimized carefully. There are several open-source software packages available for hyperparameter optimization, including: Hyperopt (Bergstra et al., 2015), Optuna (Akiba et al., 2019), Ray tune (Liaw et al., 2018), Optunity (Claesen et al., 2014), and the like. Herein, the Optuna framework is chosen, utilizing a Bayesian optimization technique called a tree-structured parzen estimator, TPE (Bergstra et al., 2011; Watanabe, 2023), to determine the optimum set of hyperparameters. Additionally, in detailed benchmark studies comparing various optimization techniques and open-source frameworks, Optuna-TPE provided the most favorable performance and computation times (Motz et al., 2022; Shekhar et al., 2022). Typically, the hyperparameter optimization process is carried out with k-folds cross validation:

a) Divide training data into k sets ("folds") randomly. Herein, $k = 3$.

b) Sample a combination of hyperparameters.

c) Set $i = 1$.

d) Place set i aside, and train the model using the remaining k - 1 sets. (When $k = 3$, these are sets 2 and 3.)

e) Evaluate the performance of the trained model using set i as the *validation* set and compute the *validation score* (e.g., RMSE – root-mean-squared-error).

f) When $i < k$, set $i = i + 1$. Return to d).

g) When $i = k$, compute the average validation score.

h) Repeat steps b) – g).

i) Return the combination of hyperparameters that resulted in the *maximum*/*minimum* average evaluation score, depending on the chosen metric (e.g., return the combination that resulted in the *minimum* average RMSE).

III) Model training with the Optimum Hyperparameters: Post optimization, the ML model, with its optimum hyperparameters, is trained using the entire training data.

Note that eight ML algorithms of varying complexities are considered in this benchmark study. These are described briefly:

1) Linear SVR (Linear Support-Vector Regressor): An extension of the popular support-vector machines (SVM) algorithm developed originally for classification problems [when the dependent variable is discrete-valued or categorical – e.g., "high", "medium", and "low"] (Cortes and Vapnik, 1995), linear SVR is a *parametric* ML algorithm that involves training a linear model, referred to as *hyperplane*, with a loss function that minimizes prediction error, also maintaining a tolerance margin, *tube* (Drucker et al., 1996). The parametric model for linear SVR is:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{5}$$

where **x** is the vector of input features; **w** is the vector of coefficients (*weights*), and *b* is the *bias* term, for each sample. Next, the objective function to be minimized during training, and constraints are:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{i=1}^{N_{\text{samples}}} (\xi_i + \xi_i^*) \tag{6}$$

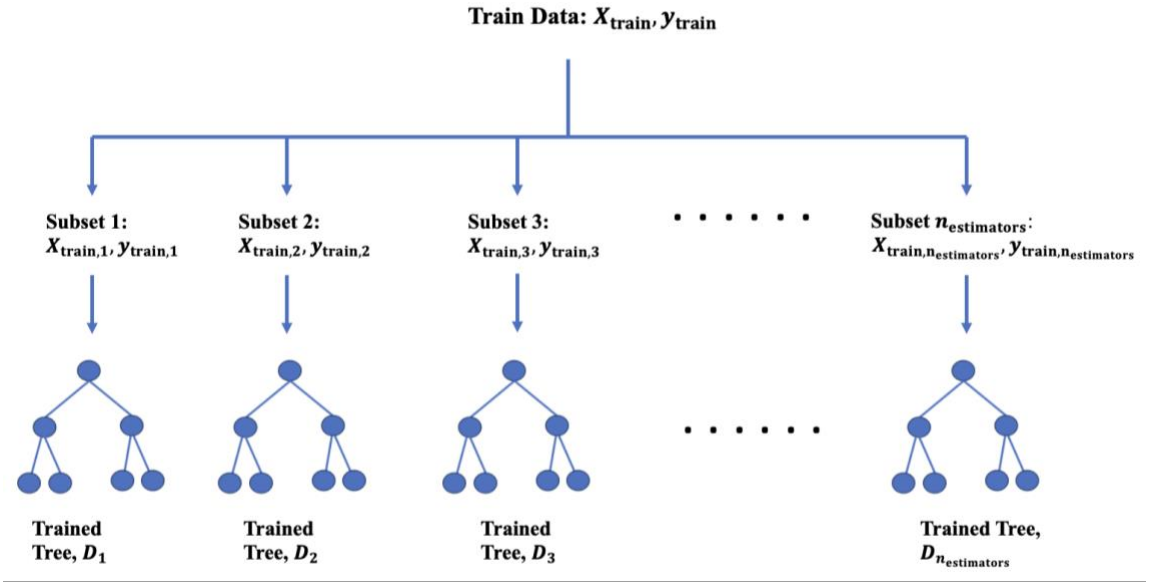
$$\begin{gathered} y_i - (\mathbf{w}^T\mathbf{x_i} + b) \leq \varepsilon + \xi_i \\ (\mathbf{w}^T\mathbf{x_i} + b) - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{gathered} \tag{7}$$

where C is the regularization term for mitigating overfitting; N_{samples} is the number of samples; ξ_i and ξ_i^* are *slack variables* (these define the penalty given to samples that violate the tolerance margin in the objective function); ε defines the tolerance margin. Note that $\mathbf{w}$, b, ξ_i, and ξ_i^* are parameters learned during training, while C and ε are hyperparameters. For more details, please refer to Drucker et al. (1996). Additionally, note that Linear SVR requires additional preprocessing steps; categorical and discrete-valued input variables need to be transformed into integers. Additionally, all input variables need to be scaled appropriately.

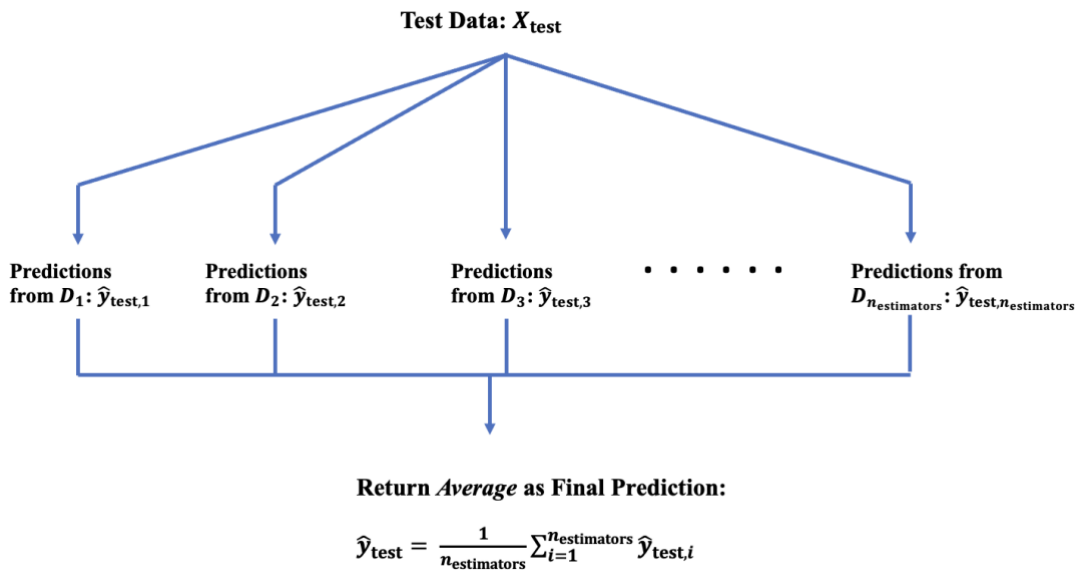
2) kNN (k-Nearest Neighbors): kNN is a *nonparametric* supervised learning algorithm that estimates the relationship between the input features and output using the concept of *nearest neighbors*. Unlike most algorithms, kNN does not involve a training phase – for regression tasks, kNN estimates the output for an unknown sample (from the test data) by computing the average output of the k known samples (from the training data) nearest to it. kNN requires two key hyperparameters to be specified: the number of nearest-neighbors, k; and a distance metric for estimating the nearest samples – possible choices include: Euclidean, Manhattan, and Minkowski metrics (Danielsson, 1980; Li et al., 2011; Suwanda et al., 2020). Similar to Linear SVR, kNN requires categorical and discrete-valued data to be transformed into integers, as well as appropriate scaling of input variables, given that kNN involves distance calculations that are sensitive to scaling.

3) RF (Random Forests): RF is a *nonparametric* learning algorithm (Breiman, 2001) that involves training an *ensemble* of decision trees using *bagging* (bootstrap aggregating); several decision trees (*weak learners*) are trained in parallel *independently* using different subsets of data that are sampled randomly (shown schematically in Figure 3a). Then, when predicting using test data, during regression tasks, the predicted

output is computed as the *average* of the predictions provided by the trained trees (shown schematically in Figure 3b). Please refer to the Appendix, Section A.4, for a simple example of a decision tree. Additionally, RF consists of several training parameters; e.g., optimal feature (input variable) for splitting, optimal split threshold for that feature, and the like. Key hyperparameters include: *n_estimators* (number of trees), *max_depth* (maximum depth for each tree), *min_samples_split* (minimum number of samples required to split), and the like. Please refer to Breiman (2001) for more details regarding the training parameters, hyperparameters, and implementation of RF. Additionally, note that RF requires categorical and discrete-valued input variables to be transformed into integers, but does not require input features to be scaled.



(a) X_{train}: Input variables for train data; y_{train}: Dependent variable for train data



(b) X_{test}: Input variables for test data; $\hat{y}_{test,i}$: Prediction for trained decision tree *i,* D_i, given X_{test}

Figure 3. Schematic implementing key steps in RF for regression tasks, showing the: (a) Training phase; (b) Testing phase

4) XGBoost (eXtreme Gradient Boosting): XGBoost is a nonparametric ensemble learning algorithm that belongs to the gradient-boosting family; several decision trees are trained *sequentially*, wherein, each newly-trained tree attempts to improve the predictions made by the previous trees (hence, the term "boosting"). Additionally, the "eXtreme" component refers to additional regularization terms in the objective function to prevent overfitting; the "Gradient" term implies that new trees are trained using the gradients (first-order derivatives) and hessians (second-order derivatives) w.r.t the errors between the previous tree and the data. Since its development by Chen and Guestrin (2016), XGBoost has become a popular algorithm for tabular datasets, for regression and classification problems across several fields (Cerna et al., 2020; Li et al., 2019; Ma et al., 2021; Ogunleye and Wang, 2020), including classification of rare events (Ashraf et al., 2023; Wang et al., 2023). Note that in prior research, Sudarshan et al. (2024b)

developed improved dynamic bidirectional multivariate alarm systems for handling rare un-postulated abnormal events using XGBoost predictive models. The training parameters for XGBoost are similar to those for RF and decision trees (e.g., optimal feature for splitting, optimal split threshold for the feature, and the like). XGBoost also consists of several hyperparameters: *n_estimators*, *max_depth*, *eta* (learning rate that scales the contribution of each tree), *subsample* (specifies fraction of data used in training each tree – helps minimize overfitting by introducing randomness), *reg_alpha* (parameter for L1-norm regularization), *reg_lambda* (parameter for L2-norm regularization), and the like. Note that some of these hyperparameters were optimized using Optuna appropriately by Sudarshan et al. (2024b) while developing XGBoost models. For more details on the algorithm and hyperparameters, please refer to Chen and Guestrin (2016) and xgboost developers (2023).

5) LightGBM (Light Gradient-Boosting Machines): Like XGBoost, LightGBM is a ML algorithm by Ke et al. (2017) that belongs to the gradient-boosting family. The key difference between XGBoost and LightGBM is: trees in XGBoost follow a *level-wise* growth strategy, in which, two resulting nodes at each level are split simultaneously; whereas trees in LightGBM follow a *leaf-wise* growth strategy; only one of the nodes, chosen optimally, is split further at each level (Liang et al., 2020), potentially reducing model development times – these are shown schematically in Figure 4. Note that the training parameters and hyperparameters for LightGBM are similar to those for XGBoost. For more details, please refer to Ke et al. (2017).

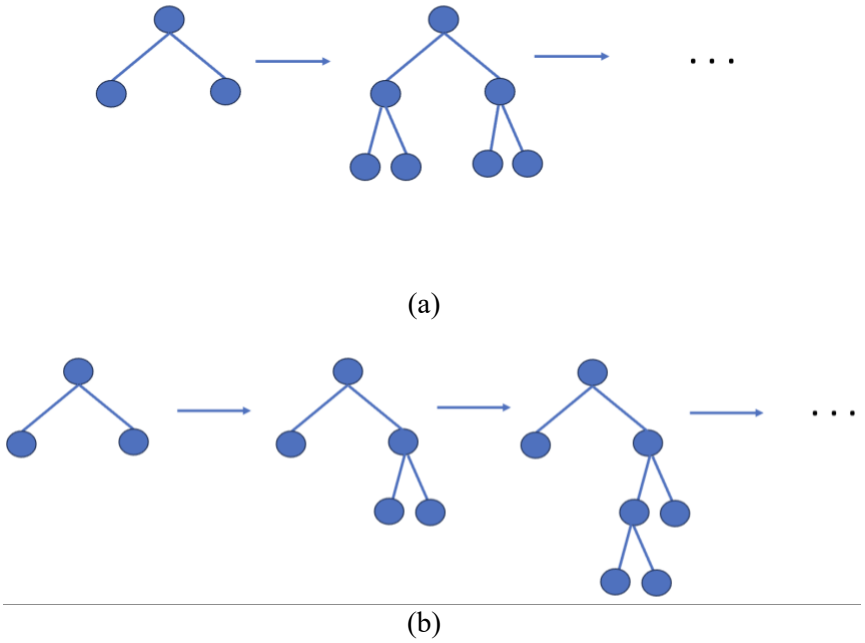


Figure 4. (a) Level-wise growth strategy followed by XGBoost; (b) Leaf-wise growth strategy followed by LightGBM

6) CatBoost (Categorical Boosting): CatBoost is another ML algorithm belonging to the gradient-boosting family, developed by Prokhorenkova et al. (2018). Compared to XGBoost and LightGBM, CatBoost implements two improvements to mitigate overfitting: i) Ability to handle data with categorical input features more-efficiently by calculating *ordered target statistics*; ii) Implementing *ordered boosting* – as per Prokhorenkova et al. (2018), gradient-boosting frameworks developed previously suffer from *prediction shift*; typically, each tree in the ensemble is trained using the entire training data, leading potentially to overfitting. In ordered boosting, each tree is trained using *random permutation sets* of the training data, using only the data before each example in the permutation set – ensuring improved robustness of models in the face of unseen data. Several articles have compared the performance of XGBoost, LightGBM, and CatBoost across different applications for: home-credit dataset (Daoud, 2019), insurance claims (So, 2024), medicare fraud detection (Hancock and Khoshgoftaar, 2020), to name a few.

Note that the training parameters and hyperparameters for CatBoost are similar to XGBoost and LightGBM. For more details, please refer to Prokhorenkova et al. (2018).

7) DNN (Dense Neural Network): Deep learning models based on Artificial Neural Networks (ANNs) have become popular across several fields, most notably for computer-vision applications such as image recognition (Traore et al., 2018) and video encoding (Ma et al., 2020); natural-language processing (NLP), such as dialogue summarization (Chen et al., 2021); machine translation (Singh et al., 2017); sentiment analysis (dos Santos and Gatti, 2014), and the like. The basic building blocks of ANNs are referred to as *perceptrons*, developed by Rosenblatt (1958). A single perceptron model consists of: i) A Linear model, $f(\mathbf{x})$, comprising of weights $\mathbf{w}$, and bias b (shown previously in Eq. (5)), where $\mathbf{x}$ is the vector of input features, and; ii) An activation function, $g(f(\mathbf{x}))$, applied to the output of the linear model. Choices for activation function include: rectified linear unit (*ReLU*) (Nair and Hinton, 2010), hyperbolic tangent (*tanh*), sigmoid (suitable for binary classification tasks), and the like. DNNs, also referred to as fully-connected neural networks (FCNNs), consist of multiple *layers* of several perceptrons, with the DNN referred to as a *deep* DNN when there are two or more *hidden layers* (the layers between the input and the output). DNN also require a *loss function* to be specified (for regression tasks, this is typically the mean-squared-error, *MSE*), with the weights and biases optimized during training through *back-propagation* over several training *epochs*. Additionally, several optimization routines exist for DNN training: gradient-descent, stochastic gradient-descent (Ruder, 2017), Adam (Adaptive Moment Estimation) (Kingma and Ba, 2017), and the like. For detailed explanations, please refer to Zou et al. (2009) and Hassoun (1995). Additionally, similar to Linear SVR and kNN, DNNs require categorical and discrete-valued input variables to be transformed into integers, as well as appropriate scaling of input variables.

8) TabNet (Tabular Networks): Given that gradient-boosting frameworks regularly outperform DNNs on tabular datasets, as observed over several benchmark studies (Borisov et al., 2022; Grinsztajn et al., 2022;

Shwartz-Ziv and Armon, 2022; Uddin and Lu, 2024), efforts have been made to develop novel neural network-based architectures, specifically for tabular data. One, developed by Arik and Pfister (2020), is TabNet. Compared to the classical DNN architecture that consists primarily of fully-connected layers, TabNet utilizes an *attention mechanism* (an *attentive transformer* layer that assigns weights to different input features, with important features weighted more heavily) to select the input features most influential for predictions. Note that this feature-selection capability is inspired from tree-based, gradient-boosting models, wherein, each tree splits data related to the most important features. Additionally, while both DNNs and TabNet can utilize *embedding* layers for processing categorical input features, TabNet is designed specifically to handle tabular data with mixed input feature types more efficiently. Note that the choice of preprocessing strategy required depends on the nature of the data and task being addressed. Recently, several research articles have considered TabNet for applications such as: rainfall forecasting, (Yan et al., 2021), electric load forecasting (Borghini and Giannetti, 2021), diabetes classification (Joseph et al., 2022), insurance claims prediction (McDonnell et al., 2023), to name a few. For more details regarding the architecture, parameters, and hyperparameters, please refer to Arik and Pfister (2020).

2.5. Step 4: Reporting Key Evaluation Metrics

After developing each dataset, as discussed in Section 3, all ML models are evaluated comprehensively and holistically across three key domains:

A) **Model Accuracy on Test Data:** The accuracy of the trained ML model on the test data is evaluated using:

$$RMSE\ (\text{metric}_1) = \text{Root Mean Squared Error} = \sqrt{\frac{\sum_{i=1}^{N_{\text{samples}}} \left(p_{\text{B,test}}(i) - \hat{p}_{\text{B, test}}(i)\right)^2}{N_{\text{samples}}}} \qquad (8)$$

where $p_{\mathrm{B,test}}$ is the committer probability for the test data, and $\hat{p}_{\mathrm{B,\,test}}$ is the committer probability using the trained ML model.

B) **Computational Efficiency:** Four clock-times are recorded to evaluate computational costs:

i) t_{hyper} (metric_2): Time recorded for hyperparameter optimization (Step II, Section 2.4).

ii) $t_{\mathrm{train}}(\mathrm{metric}_3)$: Time recorded for model training (Step III, Section 2.4).

iii) t_{test} (metric_4): Time recorded for model testing; stated differently, time taken to generate committer probability predictions for the test data.

iv) t_{deploy} (metric_5) : Time recorded for model deployment; stated differently, time taken by each ML model to generate new committer probability predictions on-line for a new dynamic simulation (the number of dynamic simulations, $N_{\mathrm{sim}} = 1$), t_{sim}, time for a *simulation*, and model *call frequency*, $call_{\mathrm{freq}}$ (frequency an ML model is called to generate new predictions on-line); e.g., $call_{\mathrm{freq}} = 30$ indicates the ML model is called once every 30 time-steps on-line to generate fresh predictions. Note that small $call_{\mathrm{freq}}$ leads to more-frequent on-line predictions of p_{B} at excessive computational costs.

C) **Alarm-system Efficiency:** To evaluate the impact of each ML model on efficiency of alarm systems, we first need to define a specific alarm system. For all datasets utilized in this benchmark study, a 2-level alarm system is assumed (number of alarm levels, $n_{\mathrm{levels}} = 2$) based on p_{B} limits (an alarm at level k is activated when the real-time p_{B} predicted by a ML model crosses the p_{B} limit defined at that level): for this benchmark study, these limits were set at $p_{\mathrm{B},1} = 0.2$, and, $p_{\mathrm{B},2} = 0.5$; where $p_{\mathrm{B},1}$ and $p_{\mathrm{B},2}$ are the p_{B} limits defined for levels 1 and 2. Then, the theoretical performance of each alarm level is computed using $p_{k,\mathrm{theoretical}}$ – defined as the *theoretical* probability with which, alarms, active at the current level k, reach the next level. Table 2 shows the 2-level alarms specified and the associated $p_{k,\mathrm{theoretical}}$ values. For instance, $p_{1,\mathrm{theoretical}} = p_{\mathrm{B},1}/p_{\mathrm{B},2} = 0.2/0.5 = 0.4$. And, $p_{2,\mathrm{theoretical}} = p_{\mathrm{B},2}/1.0 = 0.5$, given that level 2 is the last alarm-level before the undesirable region is reached (where $p_{\mathrm{B}} = 1.0$).

Table 2. 2-level Alarms Specified and Associated $p_{k,\text{theoretical}}$

Level No. (k)	$\boldsymbol{p_{\mathrm{B},k}}$	$\boldsymbol{p_{k,\text{theoretical}}}$
1	0.2	0.4
2	0.5	0.5

Next, for each ML algorithm, the performance of the alarm system is measured using $p_{k,\text{measured}}$ – defined as the probability for alarms active at the current level *k* of reaching the next level *measured* over several dynamic simulations, using real-time ML model predictions on-line, computed as:

$$p_{k,\text{measured}} = \frac{n_{\text{alarms},k\to k+1}}{n_{\text{alarms},k}} \tag{9}$$

where $n_{\text{alarms},k}$ is the number of alarms active at level *k*; $n_{\text{alarms},k\to k+1}$ is the number of alarms at level *k* that are active when the process reaches level *k*+1. Given $p_{k,\text{theoretical}}$ and $p_{k,\text{measured}}$, two alarm metrics are proposed to evaluate the alarm-system efficiency: *absolute probability difference* (Δp), and *total alarms*:

$$\Delta p\ (metric_6) = \sum_{k=1}^{n_{\text{levels}}} k\left|p_{k,\text{theoretical}} - p_{k,\text{measured}}\right| \tag{10}$$

$$Total\ Alarms\ (metric_7) = n_{\text{alarms}} = \sum_{k=1}^{n_{\text{levels}}} n_{\text{alarms},k} \tag{11}$$

Eq. (10) accounts for both *false positive rates* (fewer than expected alarms at the current level remain active when the process reaches the next level) and *false negative rates* (more than expected alarms at the current level remain active when the process reaches the next level). Additionally, as per Eq. (10), differences in higher-level alarms are penalized more heavily. Δp and *Total Alarms* are recorded over several dynamic simulations (e.g., $N_{\text{sim}} \sim 50$), with simulation time, t_{sim}, and model call-frequency, $call_{\text{freq}}$. While recording

these metrics, response actions are not included, given that $p_{k,\text{theoretical}}$ estimates do not account for changes in process dynamics when response actions are activated. Additionally, to ensure consistency, while recording t_{deploy}, Δp, and *Total Alarms*, for each dynamic simulation, a random seed number is utilized across all models (e.g., 50 random seed numbers are utilized for $N_{\text{sim}} = 50$). Note that using a random seed number ensures that the same sequence of statistical noise samples is generated for a dynamic simulation – enabling consistent comparison among ML algorithms.

2.6. Step 5: Determine Optimal ML Model

Given the evaluation metrics defined in Section 2.5, for our benchmark analyses, a weighted cost function to be minimized is proposed:

$$Cost = \sum_{i}^{n_{\text{metrics}}} (a_i)(\text{metric}_{i,\text{scaled}}) \tag{12}$$

$$[a_1, a_2, a_3, a_4, a_5, a_6, a_7] = [0.125, 0.05, 0.05, 0.05, 0.125, 0.3, 0.3] \tag{13}$$

where n_{metrics} is the total number of evaluation metrics; $n_{\text{metrics}} = 7$; a_i is the weighting coefficient for metric_i. To ensure consistency in scaling, each evaluation metric in Eq. (12) is scaled by its *maximum value* obtained across all ML models. As per Eq. (13), the coefficients are weighed such that primary importance is allocated to alarm-system efficiency (Δp and *Total Alarms*) – $a_6 = 0.3$; $a_7 = 0.3$; followed by model accuracy ($a_1 = 0.125$) and model deployment ($a_5 = 0.125$), with clock-times concerning hyperparameter optimization ($a_2 = 0.05$), model training ($a_3 = 0.05$), and model testing ($a_4 = 0.05$) weighed least. **For each dataset, the *optimal ML model* has the lowest *Cost.***

3. RESULTS AND DISCUSSIONS

This section provides results for the benchmark analysis obtained for each of the five datasets. Note that each ML algorithm, for each of its evaluation metrics and *Cost* estimate, lower values indicate better performance.

3.1. Dataset I (PI-Controlled Exothermic CSTR)

For the exothermic CSTR with abnormal transitions towards the unreliable region (see the process model summarized in Appendix A.1), the discrete values considered for the response-action variable, residence time, τ (Sudarshan et al., 2023), are:

$$\tau \in \{0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59\} \tag{14}$$

Note that all other process parameters remain constant (see Table A.1 in the Appendix). For recording t_{deploy} ($metric_5$), $N_{\text{sim}} = 1$; $t_{\text{sim}} = 30{,}000$ mins; $call_{\text{freq}} = 200$; and $\tau = 0.53$ min. Additionally, for recording metrics concerning alarm-system efficiency, $N_{\text{sim}} = 50$; $t_{\text{sim}} = 30{,}000$ mins; $call_{\text{freq}} = 200$; and $\tau = 0.53$ min. Figure 5 shows p_{B} as function of temperature for Dataset I with the colorbar varying from light to intense for the residence time, τ. It is observed that p_{B} for reaching the unreliable region increases as temperature and residence time decrease.

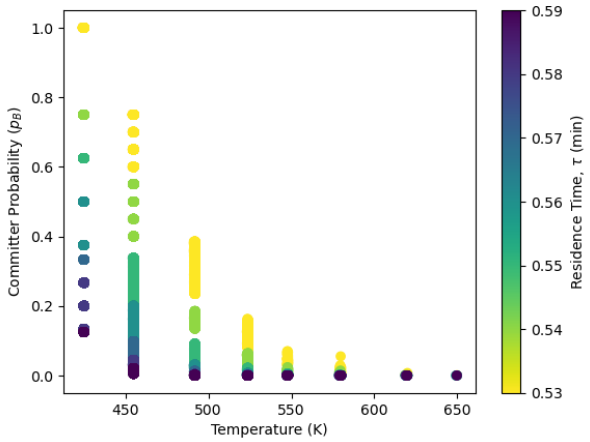
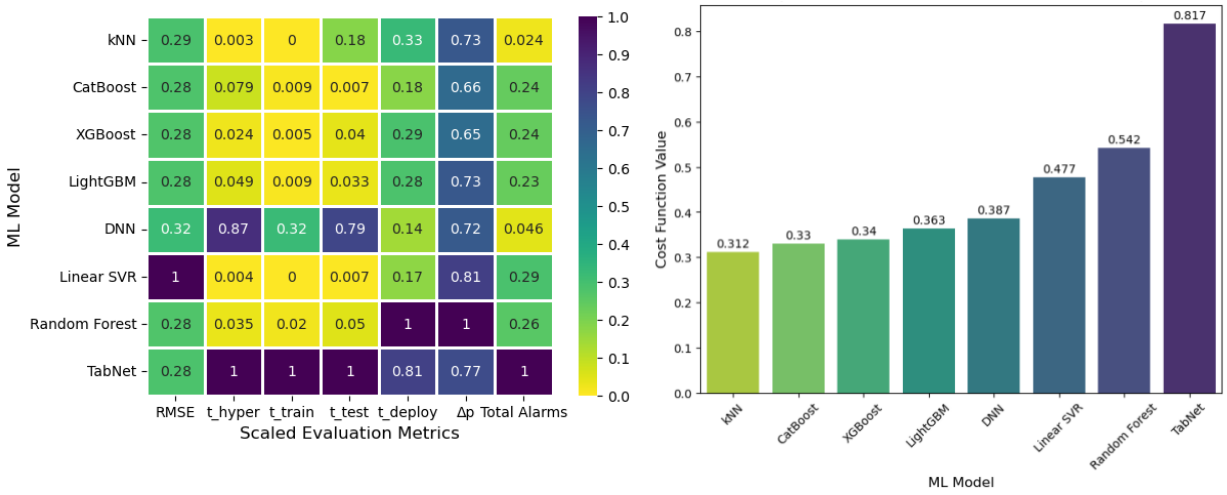


Figure 5. p_B as function of temperature and residence time for Dataset I

Figure 6a shows heatmaps for scaled evaluation metrics recorded for dataset I for the eight ML algorithms. For each metric, lighter colors indicate better performance. Most algorithms show comparable performance for model accuracy ($RMSE_{scaled}$ ~ 0.28-0.29), with Linear SVR having the worst model accuracy, and DNN having slightly-higher *RMSE*. Note that high *RMSE* indicates less-accurate on-line p_B predictions, potentially contributing to increased false and missed alarms. With a relatively-simple development process, Linear SVR compensates slightly with higher computational efficiency, followed by kNN and gradient-boosting frameworks, with DNN having low costs for deployment despite higher costs for training, hyperparameter optimization, and testing. For alarm-system efficiency, kNN and DNN perform significantly better – note that higher Δp and *total alarms* indicate increased false alarms and missed alarms, potentially resulting in alarm flooding (with the number of alarms significantly greater than operators can handle, leading to operator distraction and missed alarms). Despite much promise for tabular datasets, TabNet underperforms significantly in most evaluation metrics. Additionally, despite lower model development costs (t_{hyper} and t_{train}), RF records high costs for model deployment (t_{deploy}) – potentially resulting in a *lag* between on-line process variable measurements and p_B predictions. Please refer to Table A.4 in the Appendix for the hyperparameters optimized for each ML algorithm.

Figure 6b shows the *Cost* computed for each ML algorithm. Given the weights defined in Eq. (13), for Dataset I, kNN is observed to have the lowest cost, and is ranked as the most-optimal ML algorithm, followed closely by CatBoost and XGBoost, with TabNet ranking last.



(a) (b)

Figure 6. For Dataset 1, (a) Heatmap showing the scaled evaluation metrics; (b) *Cost* computed for all ML models

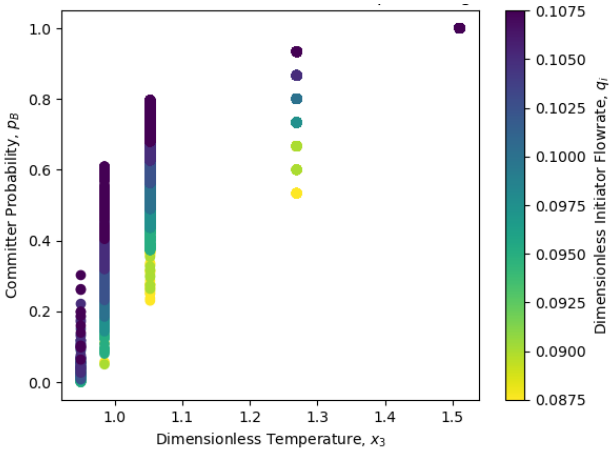
3.2. Datasets II – V (PID-Controlled Polystyrene CSTR)

For the PID-controlled polystyrene CSTR (for the process model summarized in Appendix A.2), Table 3 shows the specifications, including the response-action variable, terminal region, and discrete values considered for the response-action variables. For each dataset, all other process parameters remain constant (see Table A.2). Note that for datasets II and III, $q_m = 0.4$; and, for datasets IV and V, $q_i = 0.1$. For recording t_{deploy} ($metric_5$), $N_{sim} = 1$; $t_{sim} = 150$; $call_{freq} = 30$; $q_i = 0.1$, and $q_m = 0.3775$. Additionally, for metrics concerning alarm efficiency, $N_{sim} = 50$; $t_{sim} = 150$; $call_{freq} = 30$, $q_i = 0.1$, and $q_m = 0.4$. Figure 7 a-d shows p_B as function of dimensionless temperature for datasets II-V with the colorbar varying from light to intense for the response-action variable. For datasets II and IV, it is observed that p_B increases as dimensionless temperature and response-action variables increase; whereas, for datasets III and V, while p_B decreases with

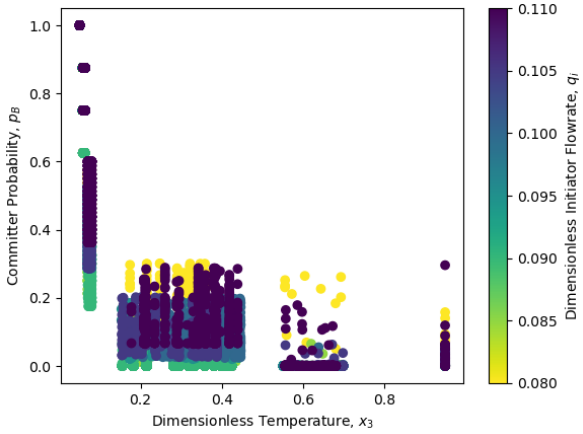
increase in dimensionless temperature, the trend with respect to response-action variables is not very clear (Sudarshan et al., 2024b).

Table 3. Specifications for Datasets II – V for PID-controlled Polystyrene CSTR

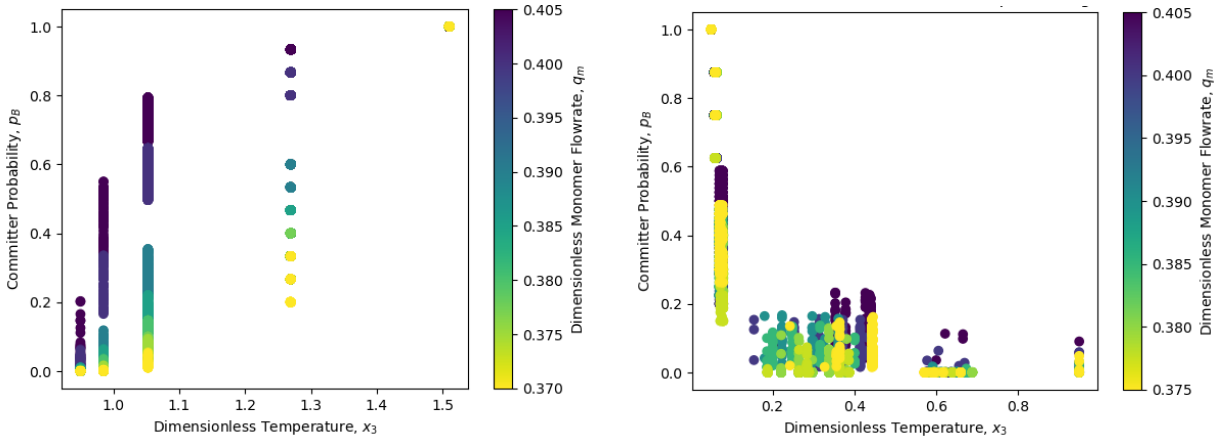
Dataset	**Response-action Variable**	**Terminal Region**	**Discrete Values Considered for Response-action Variable**
II	q_i	Unsafe	[0.0875, 0.09, 0.095, 0.0975, 0.1, 0.1025, 0.105, 0.1075]
III	q_i	Unreliable	[0.08, 0.085, 0.09, 0.095, 0.1, 0.105, 0.11]
IV	q_m	Unsafe	[0.37, 0.375, 0.3775, 0.38, 0.385, 0.39, 0.4, 0.405]
V	q_m	Unreliable	[0.375, 0.3775, 0.38, 0.385, 0.39, 0.4, 0.405]



(a)



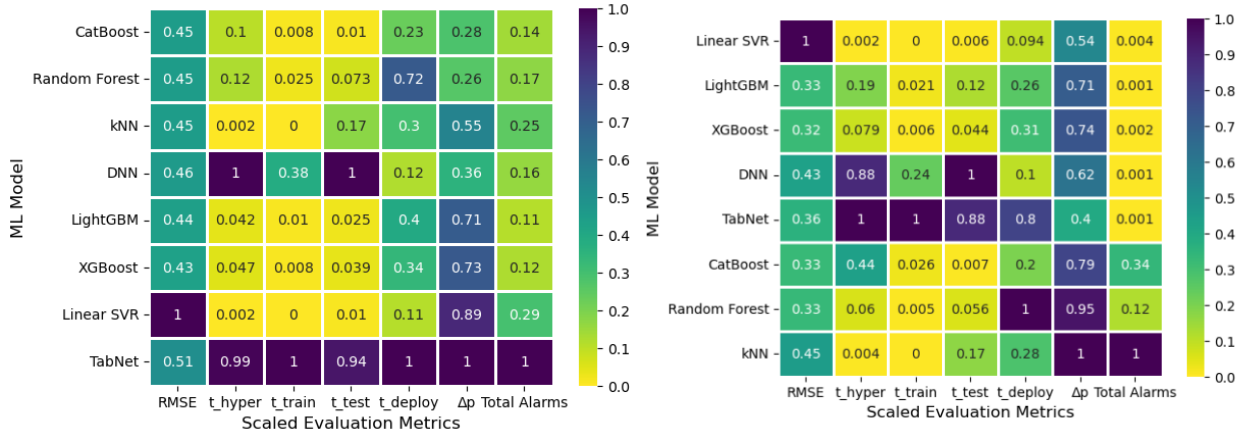
(b)



(c) (d)

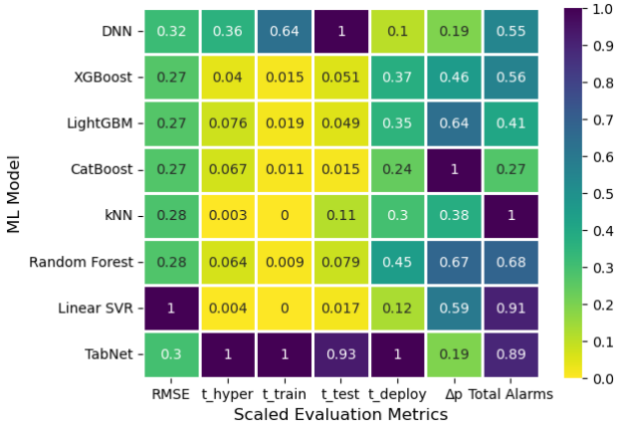
Figure 7. p_B as function of dimensionless temperature for Datasets: (a) II; (b) III; (c) IV; (d) V

Figure 8 a-d show heatmaps for scaled evaluation metrics for datasets II – V. For model accuracy, gradient-boosting algorithms achieve stronger predictive performance, with Linear SVR scoring the lowest *RMSE*, given its low-complexity. More-complex DNN and TabNet algorithms do not justify their *RMSE* scores. For computational efficiency, the less-complex algorithms, Linear SVR and kNN offer fast computational times, followed by the gradient boosting algorithms, with DNN and TabNet having lowest computational efficiency (except fast model deployment for DNN, consistent with that observed for dataset I). Additionally, RF records high costs for deployment, despite relatively-lower costs for model development, as observed for dataset I. For alarm efficiency, performance varies across the datasets, with best being CatBoost and RF for dataset II; Linear SVR and LightGBM for dataset III; DNN, XGBoost, and LightGBM for dataset IV; CatBoost and LightGBM for dataset V. Note – TabNet performs poorly uniformly.

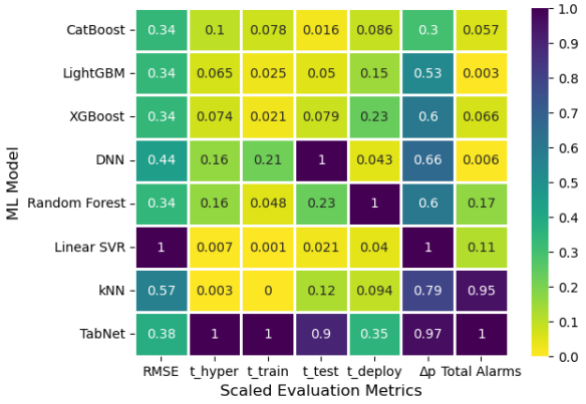


(a)

(b)



(c)



(d)

Figure 8. Heatmaps showing scaled evaluation metrics for datasets (a) II; (b) III; (c) IV; (d) V

For datasets II-V, Figure 9 a-d show the *Cost* for all ML algorithms, given the weights defined in Eq. (13). CatBoost ranks as the most-optimal ML model for datasets II and V, with DNN the most-optimal for dataset IV. Note that despite poor model accuracy, Linear SVR compensates by having improved

computational and alarm efficiencies, thereby, unexpectedly ranking as the most-optimal ML algorithm for dataset III. With the exception of dataset III, TabNet is the least-optimal model.

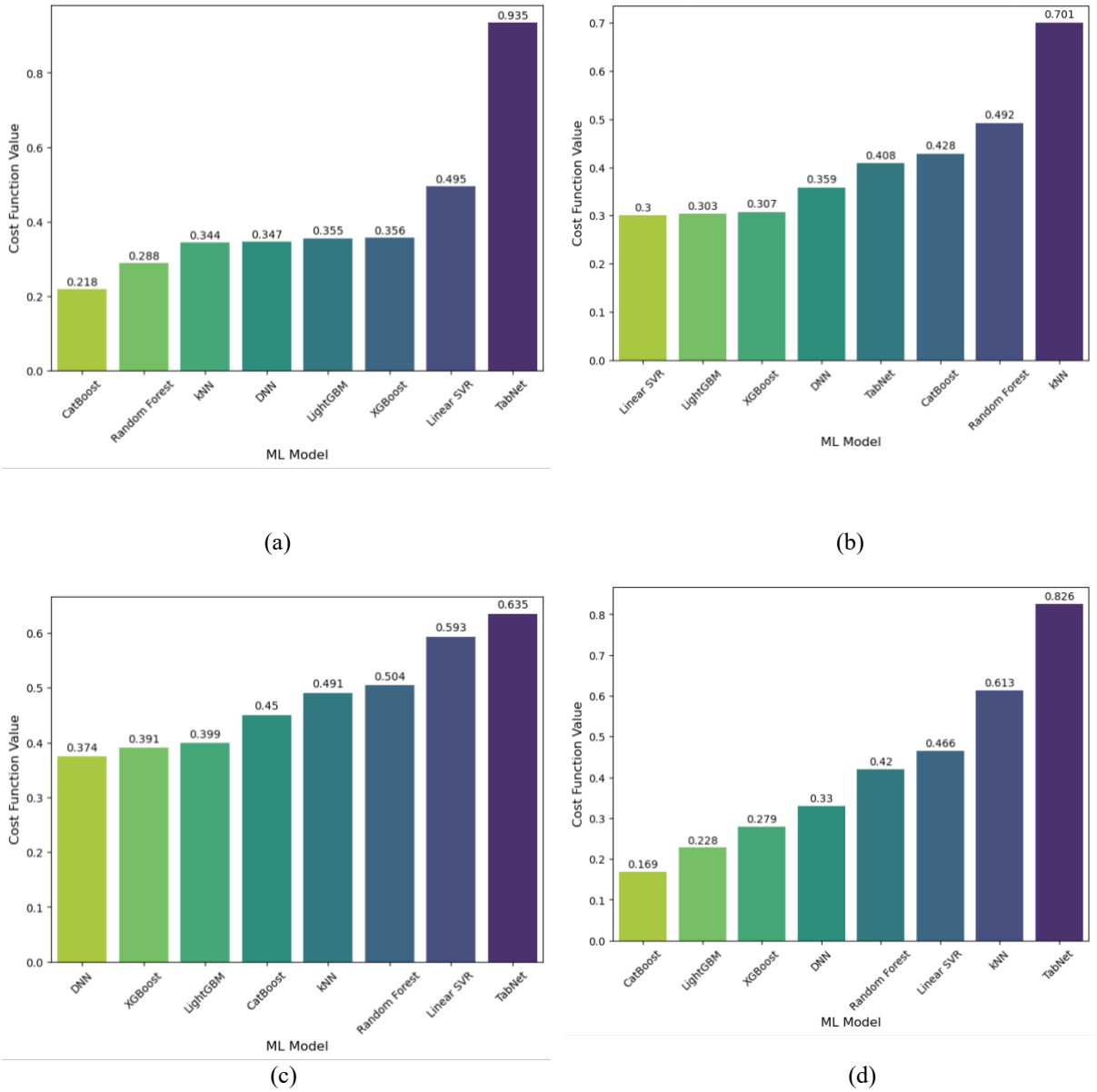


Figure 9. *Cost* computed for all ML models for datasets: (a) II; (b) III; (c) IV; (d) V

3.3. Average Rankings Across All Datasets for Weights in Eq. (13)

For all ML models, Figure 10 shows the *average (mean) rankings* across all five datasets, using the weighting coefficients in Eq. (13). Clearly, the gradient-boosting frameworks achieve favorable rankings, with CatBoost achieving the highest ranking, followed by LightGBM and XGBoost, with RF, Linear SVR and TabNet recording the lowest rankings.

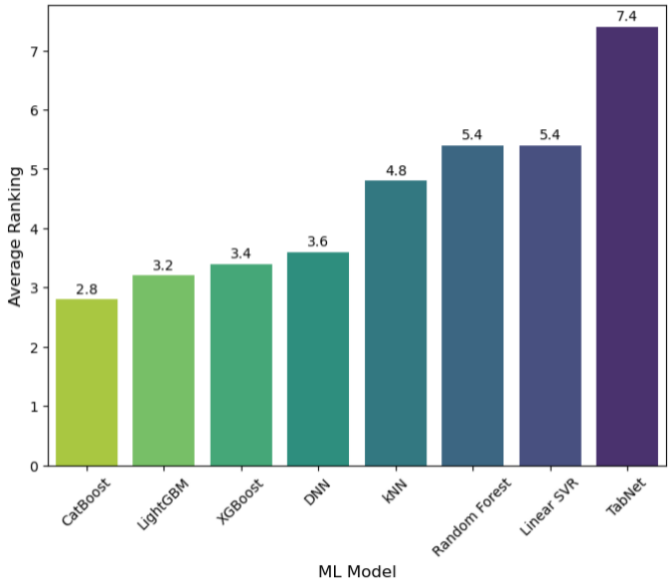


Figure 10. Average local rankings computed for models across all datasets for weights in Eq. (13)

3.4. Double-Averaged Global Ranking Across All Datasets and 500 Weight Combinations

Note that the *Cost* results in Sections 3.1-3.3 (Figures 6b, 9, and 10) are for the single combination of weighting coefficients in Eq. (13). For more-comprehensive analyses, several combinations of weighting coefficients are justified. Herein, 500 combinations are considered, with each combination sampled randomly from uniform distributions, given upper and lower bounds:

$$
\begin{aligned}
0.1 \leq a_1 \leq 0.2 \\
0.05 \leq a_2 \leq 0.1 \\
0.05 \leq a_3 \leq 0.1 \\
0.05 \leq a_4 \leq 0.1 \\
0.1 \leq a_5 \leq 0.2 \\
0.3 \leq a_6 \leq 4 \\
0.3 \leq a_7 \leq 4
\end{aligned}
\tag{15}
$$

Note that while the choice of distribution is arbitrary, with the normal distribution and others possible, these weighting coefficient bounds emphasize alarm-system efficiency over other metrics.

As shown in Figure 11, the use of these randomly-sampled weighting coefficients provides a similar ranking. CatBoost achieves the highest global ranking, followed by XGBoost, LightGBM, and DNN, with TabNet achieving the lowest ranking. The increased ranking of XGBoost may warrant further consideration.

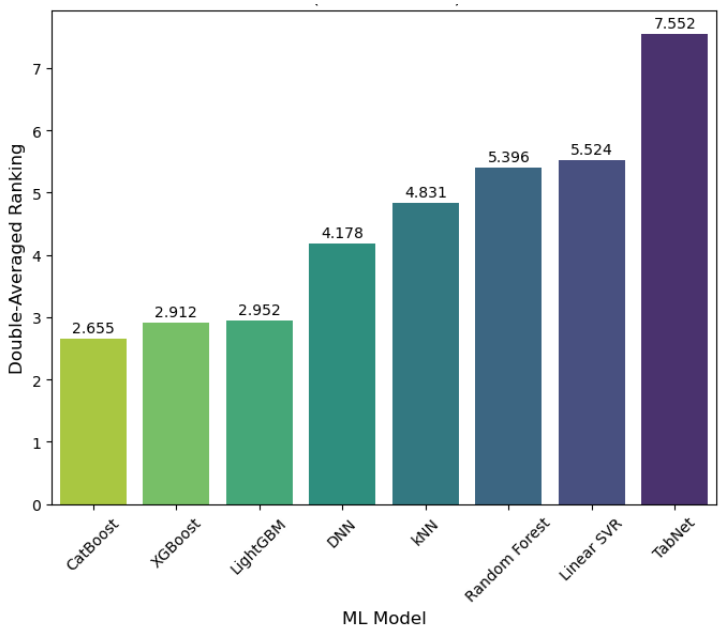


Figure 11. Double-averaged, global ranking for 500 combinations of weighting coefficients across all datasets

4. CONCLUSIONS

In previous research, using path-sampling (BGFFS) and non-parametric machine learning, dynamic, bidirectional multivariate alarm systems were developed for rare un-postulated abnormal movements away from normal operating regions, demonstrated successfully on a PID-controlled polymerization CSTR. However, in predictive modeling, only one ML algorithm was explored: XGBoost.

Herein, a comprehensive framework is developed for benchmark analyses to explore optimal ML algorithms of varying complexities, for enhancing predictions of rare abnormal events using chemical process models. For evaluation, several metrics are considered, permitting balances between model accuracy, and computational and alarm-system efficiencies, with more preference given to alarm-system efficiencies. For the weighting coefficients considered in Eq. (13), the gradient-boosting frameworks: XGBoost, LightGBM, and CatBoost, outperform other algorithms, achieving strong predictive performance at low computational costs, also providing relatively favorable metrics for alarm-system efficiency. DNN and TabNet require more computational resources that are not justified by their model accuracy, although DNN offers fast deployment across all datasets. Despite much promise for tabular datasets, TabNet consistently performs poorly across all datasets. Additionally, Linear SVR and kNN compensate for lower model accuracies by having low computational costs, but, along with RF, are outperformed consistently by the gradient-boosting frameworks when all metrics in the *Cost* are considered. Moreover, based on the global rankings recorded in Figure 11 that consider 500 randomly-sampled combinations of weighting coefficients, CatBoost is the most-optimal algorithm across all datasets and evaluation metrics, followed by XGBoost, LightGBM, and DNN. Note that increased *RMSE*, Δp and *total alarms* may contribute potentially to increased false alarm and missed alarm rates. Additionally, higher model deployment times may result in a lag between real-time process variable measurements and p_B predictions.

To our knowledge, this manuscript is the first ML benchmark analysis that evaluates algorithms for predicting rare events in chemical process safety. Additionally, while most benchmark frameworks place

emphasis on model accuracy and computational costs alone, our work is the first that attempts to evaluate comprehensively and holistically ML algorithms of varying complexity by also considering alarm-system metrics (i.e., the number and efficiency of alarms activated). Hence, such comprehensive benchmark frameworks will aid the operator in selecting the most-optimal ML algorithm for process monitoring and predictive maintenance against rare abnormal events, improving their effectiveness in ensuring safety and reliability.

5. FUTURE RESEARCH

Note that despite encouraging findings herein, a few limitations should be addressed in future research. These are discussed briefly:

1) *Hybrid Models Capturing Physics and Plant Data*: Herein, all datasets generated using BGFFS are based on first-principles models (material and energy-balance ODEs, reaction kinetics, and the like), with assumptions simplifying the process models (Sudarshan et al., 2024b, 2021). In future research, hybrid computational models (e.g., physics-informed neural networks, PINNs) involving underlying physics, coupled with plant data from sensors, alarm databases, and the like, should be developed, with benchmark analyses extended to such models.

2) *Considering Several Alarm-system Combinations and Improved Alarm Rationalization*: The analyses presented herein utilize a 2-level alarm system (see Table 2). For more-comprehensive analyses, it is important to consider several alarm-system combinations, but this would require significant computational costs. Hence, to address this, in future work, more-rigorous sensitivity analyses should be conducted to measure the sensitivity of alarm-system combinations to the optimal ML algorithm. Additionally, Δp (see Eq. (10)) should be utilized to develop more-intelligent, automated/semi-automated, alarm rationalization strategies, a significant improvement compared to the framework developed in our prior research (Sudarshan et al., 2024a).

3) *Accounting for Source-to-source Variability*: Herein, the BGFFS algorithm simulates numerous rare-event pathways efficiently by introducing random perturbations in inlet feed concentrations – the only source of variability considered. To further account for source-to-source variability, future research should extend our simulations to include variations and quantify uncertainty in crucial process parameters that affect operational behavior of chemical processes.

4) *Model Interpretability as a Benchmark Metric*: Model interpretability is increasingly important in the explainable AI paradigm (Linardatos et al., 2021). As complex ML models are integrated increasingly into automation and decision-making in chemical industries, especially in safety-critical environments, the need to ensure they are transparent and interpretable is becoming more important. This is crucial for increasing trust in model predictions and facilitating informed decision-making by industrial practitioners. Although explainable AI frameworks such as LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016) and SHAP (Shapley Additive explanations) (Lundberg and Lee, 2017) offer valuable insights into feature importance and local model behavior (stated differently, the influence of each input feature on model predictions), there remains a lack of standardized quantitative metrics for comparing interpretability across different models. Future research is needed to develop such metrics to be included in benchmark analyses.

5. ACKNOWLEDGEMENTS

The NSF CBET funding for our grant, 2220276, is greatly appreciated. Thanks are extended to Amish J. Patel, Ulku G. Oktem, and Jeffrey E. Arbogast, who provided advice throughout this research.

APPENDIX

A.1. Proportional-Integral (PI)-Controlled Exothermic CSTR

Previously, novel, multivariate alarm systems were developed using path-sampling and predictive modeling for a relatively simple controlled exothermic CSTR, followed by the alarm rationalization-DRAn integrated framework for further enhancement (Sudarshan et al., 2024a, 2023). In this Appendix A.1, we provide a brief summary of the process model. Figure A.1a shows a schematic of the model for a Proportional-Integral (PI)-controlled exothermic CSTR, with first-order kinetics, $A \rightarrow P$. The assumptions for this *ideal* process model include: i) Constant residence time; ii) Incompressible flow; iii) Complete back-mixing. The model controls the reactor temperature, T, by manipulating the coolant flow-rate, F_C. Additionally, Figure A.1b shows the steady-state behavior for the model, with multiple steady-states observed for residence time; $\tau \in [0.47, 0.56]$ min. Two stable steady-states are observed: at the high conversion-high temperature basin 'A', and low-conversion, low-temperature basin 'B'. In between these two, a wide unstable "cliff" exists, such that when the process operates near this cliff, with sufficient input perturbation, the process shifts rapidly to either of the two stable regions (Moskowitz et al., 2018; Sudarshan et al., 2021).

The governing equations for the PI-controlled process are:

$$V\frac{dC_{\mathrm{A}}}{dt} = \frac{V}{\tau}(C_{\mathrm{Af}} - C_{\mathrm{A}} + \boldsymbol{\eta}) - Vk_0 \exp\left(-\frac{E}{RT}\right)C_{\mathrm{A}} \tag{A.1}$$

$$\rho V c_{\mathrm{p}}\frac{dT}{dt} = \frac{\rho V c_{\mathrm{p}}}{\tau}(T_{\mathrm{f}} - T) - V\Delta H k_0 \exp\left(-\frac{E}{RT}\right)C_{\mathrm{A}} + UA(T_{\mathrm{c}} - T) \tag{A.2}$$

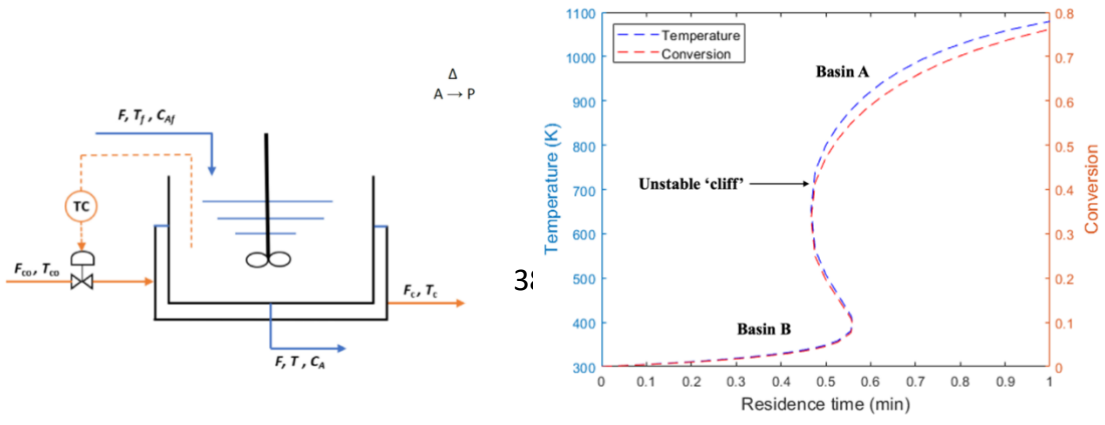
$$\frac{dT_{\mathrm{C}}}{dt} = \frac{F_{\mathrm{C}}}{V_{\mathrm{j}}}(T_{\mathrm{C0}} - T_{\mathrm{C}}) - \frac{UA}{\rho_{\mathrm{w}} V_{\mathrm{j}} c_{\mathrm{pw}}}(T_{\mathrm{C}} - T) \tag{A.3}$$

$$F_{\mathrm{C}} = F_{\mathrm{C0}} + K_{\mathrm{C}}\left(T - T_{\mathrm{SP}} - \frac{e_{\mathrm{I}}}{\tau_{\mathrm{I}}}\right); 30 \leq F_{\mathrm{C}} \leq 70 \tag{A.4}$$

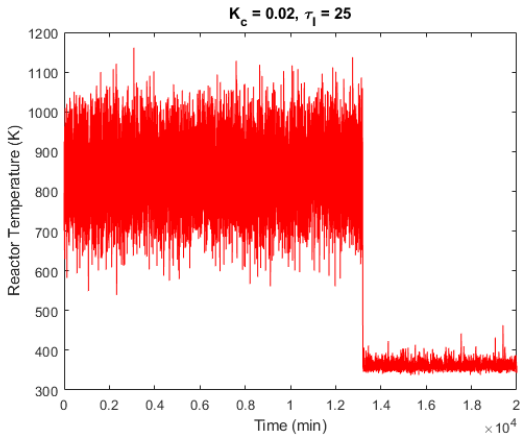
$$\frac{de_{\mathrm{I}}}{dt} = T_{\mathrm{SP}} - T \tag{A.5}$$

$$C_{\mathrm{A0}} = 1.2\,\mathrm{kmol/m^3}\,;\; T_0 = 700\;K \tag{A.6}$$

where C_{A} is the concentration of reactant A; T is the reactor temperature; C_{Af} and T_{f} are the concentration and temperature for the reactant feed stream; F_{C0} is the cooling water flow-rate at steady state, K_{C} is the controller gain, e_{I} is the integral error and τ_{I} is the integral time constant; T_{C0} is the inlet temperature of the cooling water, C_{A0} is the initial value for the concentration, T_0 is the initial value for the temperature, T_{SP} is the set-point temperature for the controller, V_{reactor} is the volume of the reactor, U is the overall heat-transfer coefficient, A is the heat-transfer area, ΔH is the heat of reaction, ρ is the feed density, c_{p} is the heat capacity of the feed stream, V_{j} is the volume of the cooling-water jacket, ρ_{w} is the density of the cooling water, and c_{pw} is the specific-heat capacity of the cooling water (refer to Table A.1). To induce un-postulated abnormal transitions from the desirable basin "A" to the undesirable and unreliable basin "B", statistical "noise"-induced perturbations, η, are utilized. Note that η is sampled randomly at every integration time-step from a normal distribution; $\eta \sim \mathcal{N}(\mu,\ {\sigma_\eta}^2)$, with mean: $\mu = 0$, and variance: ${\sigma_\eta}^2 = 0.02$. Figure A.1c shows a dynamic brute-force simulation of the process under *noisy* operation, showing the un-postulated abnormal transition from basins A to B. Dataset I contains process variable data for several such trajectories simulated efficiently using BGFFS, followed by calculations of the committer probabilities and preprocessing (Section 2.3).



(a) (b)



(c)

Figure A.1. (a) Schematic of the PI-controlled exothermic CSTR; (b) Process under steady-state operation; (c) Process showing the "noise"-induced un-postulated abnormal transition.

Table A.1. Process Constants and Parameters for PI-Controlled Exothermic CSTR

Parameter	**Value**	**Unit**
A	30	m^2
C_{Af}	2	$kmol/m^3$
$c_p = c_{pw}$	4	kJ/(kg-K)
E	1.50E+04	kJ/kmol
F_{C0}	50	m^3/min
k_0	17.038	1/min
R	8.314	kJ/(kmol-K)
T_{C0}	300	K
T_f	300	K
T_{SP}	800	K
U	100	$kJ/(min\text{-}K\text{-}m^2)$
$V_{reactor} = V_j$	10	m^3
ΔH	-2.20E+06	kJ/kmol
$\rho = \rho_w$	1,000	kg/m^3
K_C	0.02	$m^3/(min\text{-}K)$
τ_I	25	min

A.2. Proportional-Integral-Derivative (PID)-Controlled Polymerization CSTR

Given the limitations of the alarm systems developed for the controlled exothermic CSTR, in recent research, improved, dynamic, bidirectional multivariate alarm systems were developed for a PID-controlled polystyrene CSTR, capable of handling un-postulated abnormal shifts to multiple undesirable regions: unsafe and unreliable (Sudarshan et al., 2024b). In this Appendix A.2, we provide a brief summary of the process model. Figure A.2a shows a schematic of the model. The governing equations for the dimensionless PID-controlled polystyrene CSTR are:

$$\frac{dx_1}{d\tau} = q_{\mathrm{i}} x_{1\mathrm{f}} - (q_{\mathrm{i}} + q_{\mathrm{m}} + q_{\mathrm{s}}) x_1 - \phi_{\mathrm{d}} \kappa_{\mathrm{d}}(x_3) x_1 \tag{A.7}$$

$$\frac{dx_2}{d\tau} = q_{\mathrm{m}}(x_{2\mathrm{f}} + \boldsymbol{\eta}) - (q_{\mathrm{i}} + q_{\mathrm{m}} + q_{\mathrm{s}}) x_2 - \phi_{\mathrm{p}} \kappa_{\mathrm{p}}(x_3) x_2 x_5; \ \eta \sim \mathcal{N}\left(0, {\sigma_\eta}^2\right) \tag{A.8}$$

$$\frac{dx_3}{d\tau} = (q_{\mathrm{i}} + q_{\mathrm{m}} + q_{\mathrm{s}})(x_{3\mathrm{f}} - x_3) + \beta \phi_{\mathrm{p}} \kappa_{\mathrm{p}}(x_3) x_2 x_5 - \delta(x_3 - x_4) \tag{A.9}$$

$$\frac{dx_4}{d\tau} = \delta_1 [q_{\mathrm{c}}(x_{4\mathrm{f}} - x_4) + \delta \delta_2 (x_3 - x_4)] \tag{A.10}$$

$$x_5 = \sqrt{\frac{2f \phi_{\mathrm{d}} \kappa_{\mathrm{d}}(x_3) x_1}{\phi_{\mathrm{t}} \kappa_{\mathrm{t}}(x_3)}} \tag{A.11}$$

$$q_{\mathrm{c}} = q_{\mathrm{c},0} - K_{\mathrm{c}} \left[(x_{3,\mathrm{sp}} - x_3) + \frac{1}{\tau_{\mathrm{I}}} \int_0^t (x_{3,\mathrm{sp}} - x_3) dt' + \tau_{\mathrm{D}} \frac{d(x_{3,\mathrm{sp}} - x_3)}{dt} \right] \tag{A.12}$$

$$0 \leq q_{\mathrm{c}} \leq 5 \tag{A.13}$$

$$x_{1,0} = 0.0041; \ x_{2,0} = 0.2156; \ x_{3,0} = 0.951; x_{4,0} = -1.1191; \ q_{c,0} = 1.5 \tag{A.14}$$

$$\kappa_{\mathrm{d}}(x_3) = \exp\left(\frac{\gamma_{\mathrm{d}} x_3}{1 + \frac{x_3}{\gamma_{\mathrm{p}}}} \right) \tag{A.15}$$

$$\kappa_{\mathrm{t}}(x_3) = \exp\left(\frac{\gamma_{\mathrm{t}} x_3}{1 + \frac{x_3}{\gamma_{\mathrm{p}}}}\right) \tag{A.16}$$

$$\kappa_{\mathrm{p}}(x_3) = \exp\left(\frac{x_3}{1 + \frac{x_3}{\gamma_{\mathrm{p}}}}\right) \tag{A.17}$$

where x_1, x_2, x_3, x_4, x_5 are the dimensionless initiator concentration, monomer concentration, reactor temperature, coolant temperature, and concentration of growing polymer; $q_{\mathrm{i}}, q_{\mathrm{m}}, q_{\mathrm{s}}, q_{\mathrm{c}}$ are the dimensionless flow rates for initiator, monomer, solvent and coolant streams; $\phi_{\mathrm{d}}, \phi_{\mathrm{p}}, \phi_{\mathrm{t}}$ are Damkohler numbers for initiator decomposition, propagation, and termination; $\gamma_{\mathrm{d}}, \gamma_{\mathrm{p}}, \gamma_{\mathrm{t}}$ are dimensionless activation energies for initiator decomposition, propagation, and termination; β is the dimensionless heat of reaction, δ is the dimensionless heat-transfer coefficient, δ_1 is the dimensionless reactor volume, δ_2 is the dimensionless specific heat, f is the initiator efficiency; $x_{1\mathrm{f}}$, $x_{2\mathrm{f}}$, $x_{3\mathrm{f}}$, $x_{4\mathrm{f}}$ are the dimensionless initiator feed concentration, monomer feed concentration, reactor feed temperature, and coolant feed temperature; $x_{1,0}$, $x_{2,0}$, $x_{3,0}$, $x_{4,0}$, $q_{\mathrm{c},0}$ represent initial values. Similar to the PI-controlled exothermic CSTR, statistical "noise", η, is sampled at every integration time-step, from a normal distribution, $\mathcal{N}(\mu, {\sigma_\eta}^2)$, with mean: $\mu = 0$; and variance: ${\sigma_\eta}^2 = 0.0014$. Note that η is added to the dimensionless monomer concentration, $x_{2\mathrm{f}}$, in Eqn. (A.8). Figure A.2b shows the process under steady-state operation, with the process constants and parameters in Table A.2; the intermediate, *unstable* region is the desirable region, with possible un-postulated abnormal shifts to both *unsafe* and *unreliable* regions, which are stable. Figure A.2 c-d shows the process under *noisy* operation, with un-postulated abnormal transitions observed towards both undesirable regions. In Figure A.2c, the process moves from $x_3 = 0.95$ in the unstable region to a brief visit to the *unreliable* region before settling in the unstable region. In Figure A.2d, it moves first to the *unsafe* region before returning to the unstable region. For more details regarding the process model and these *noisy* trajectories, please refer to and Russo and Bequette (1998) and Sudarshan et al. (2024b).

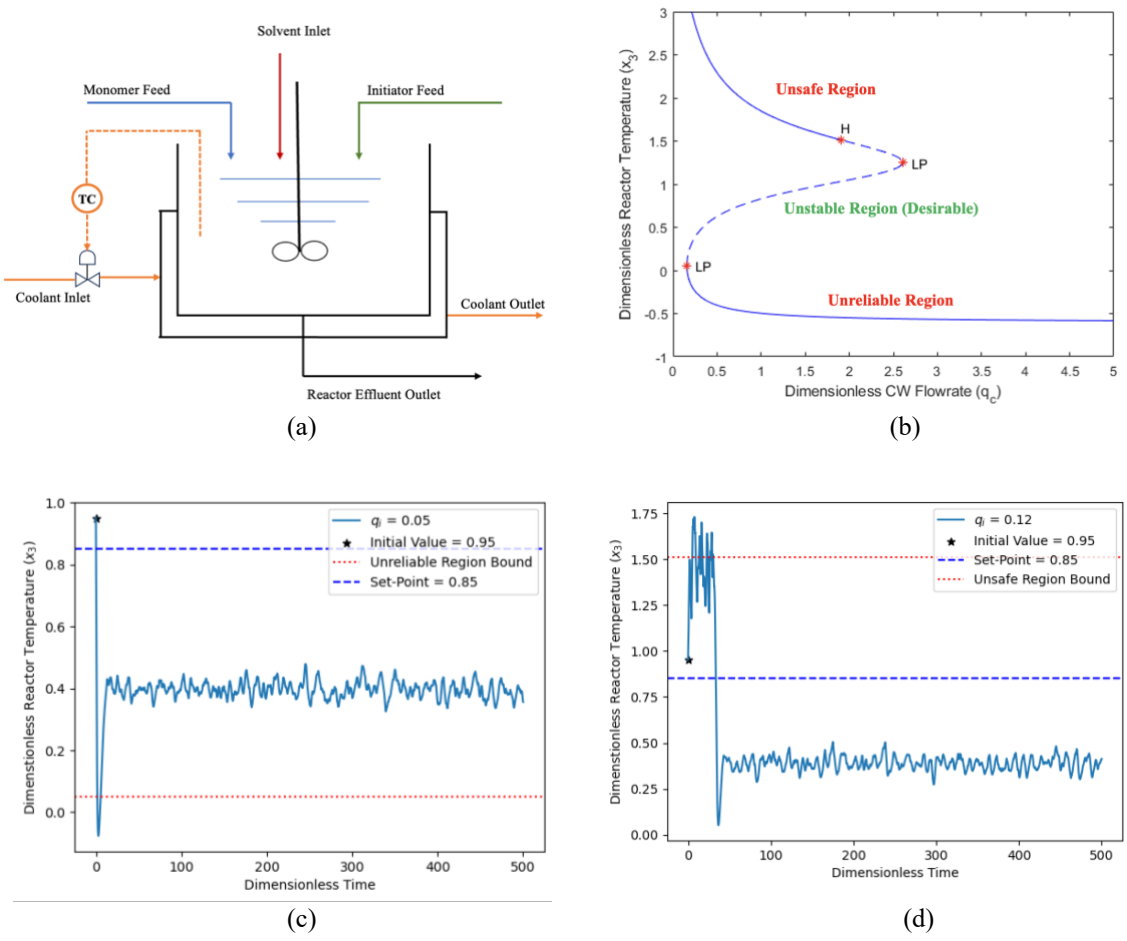


Figure A.2. (a) Schematic of PID-controlled polystyrene CSTR; (b) Process under steady-state operation, showing the desirable unstable and undesirable stable region. Note the key bifurcation points: limit points, LP, and Hopf bifurcation points, H; (c) Process under dynamic operation, showing the un-postulated transition towards the unreliable region; (d) Process under dynamic operation, showing the un-postulated transition towards the unsafe region.

Table A.2. Process Constants and Parameters for PID-controlled Polystyrene CSTR

Parameter	Value
q_{i}	0.1
q_{m}	0.4
q_{s}	0.48571
ϕ_{d}	0.01688
ϕ_{p}	2.1956×10^{7}
ϕ_{t}	9.6583×10^{12}
$x_{1\mathrm{f}}$	0.06769
$x_{2\mathrm{f}}$	1.0
$x_{3\mathrm{f}}$	0.0
$x_{4\mathrm{f}}$	-1.5
δ	0.74074
δ_1	0.90569
δ_2	0.37256
β	13.17936
f	0.6
$x_{3,\mathrm{sp}}$	0.85
K_{c}	50
τ_{D}	0.9
τ_{I}	5

A.3. Computational Specifications

Note that all analyses and results presented herein were conducted on a Desktop computer, having specifications:

i) Operating System (OS): Linux via WSL (Windows Subsystem for Linux) 2.

ii) CPU: 12$^{\mathrm{th}}$ - generation Intel i7-12700K with 12 cores (8 performance + 4 efficiency), 32 GB DDR5 RAM.

iii) GPU: NVIDIA RTX 3060 Ti, 8 GB RAM.

The Python programming language (version 3.9) is utilized, leveraging several powerful open-source software packages, including: NumPy (Harris et al., 2020), Pandas (McKinney, 2010) , SciPy (Virtanen et al., 2020), Matplotlib (Hunter, 2007), Scikit-Learn (Pedregosa et al., 2011) XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2018), Numba (Lam et al., 2015), Optuna (Akiba et al., 2019), to name a few. For developing DNNs, PyTorch (Paszke et al., 2019) is utilized; with PyTorch-TabNet (Arik and Pfister, 2020; Dreamquark, 2019) utilized to develop TabNet models. Also, for GPU-acceleration during model development, the NVIDIA Compute Unified Device Architecture (CUDA) toolkit (NVIDIA et al., 2022) is utilized. Note that while XGBoost, LightGBM, CatBoost, PyTorch and PyTorch-TabNet have native support for CUDA-based GPU-acceleration, the Scikit-Learn-based implementation of Linear SVR, kNNs, and RF does not support GPU acceleration natively – to address this, *RAPIDS AI*: an open-source suite of software packages developed for GPU acceleration (RAPIDS Development Team, 2023), is utilized, with cuML and cuDF packages providing GPU acceleration for Linear SVR, kNNs, and RF.

A.4. Simple Example of a Decision Tree

Figure A.3 shows a schematic for a decision tree involving an example dataset created for demonstration purposes only, containing just 10 samples ($N_{samples}$ = 10), to predict the committer probability, p_B as a function of temperature, T. For this dataset, Table A.3 shows p_B and T for each of the samples. At each iteration, the optimum split threshold for T is computed using the variance reduction method described in (Breiman et al., 2017). On this basis, the data is split further, with the splitting process terminated when insufficient data remain, after which the average p_B is returned. For instance, at the first split, the optimum split threshold for T is computed as T = 480 K – then, the data are divided into two sets: 5 samples for which, T <= 480 K, and 5 samples for which, T > 480 K. The splitting process continues for both sets until the number of samples remaining in a set is <=N_{min} (N_{min} is the threshold for the number of samples in a set

to stop splitting; e.g., in Figure A.3, $N_{min} = 3$) thereby, returning the average p_B for that set. To check for consistency with the data in Table A.3, note the average p_B values returned at the end of the decision tree in Figure A.3 – for instance, there are 2 samples for which, $T <= 360$ ($T = \{300, 340\}$; $p_B = \{0.1, 0.2\}$), with the average p_B for these samples = 0.15, consistent with Figure A.3. Similarly, there are 3 samples for which, $360 < T <= 480$ ($T = \{380, 420, 460\}$; $p_B = \{0.3, 0.4, 0.5\}$), with their average $p_B = 0.4$.

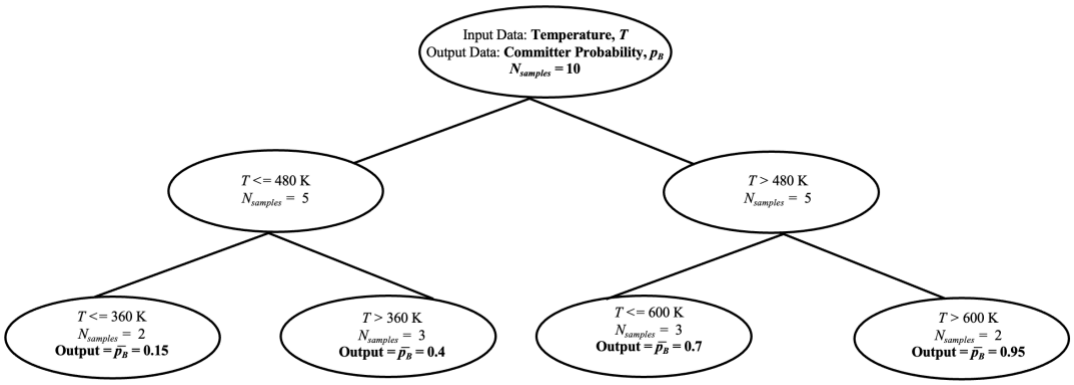


Figure A.3. Schematic of a decision tree, a flowchart-like model that helps make decisions by answering a series of questions based on the input variables (e.g., temperature) of the data, ultimately leading to a decision or prediction.

Table A.3. Example Dataset for the Decision Tree Trained in Figure A.3

Temperature, *T* (K)	**Committer Probability, p_B**
300	0.1
340	0.2
380	0.3
420	0.4
460	0.5
500	0.6
540	0.7
580	0.8
620	0.9
660	1.0

A.5. Table of Hyperparameters Optimized for Each ML Algorithm

Table A.4 shows the hyperparameters optimized for each model, as part of ML model development described in Section 2.4, along with their references.

Table A.4. Hyperparameters Optimized for Each ML Algorithm in Section 2.4

ML Algorithm	Hyperparameters Optimized	References
Linear SVR	*C*, *epsilon*, *loss*	Pedregosa et al. (2011); RAPIDS Development Team (2023)
kNNs	*n_neighbors*, *metric*	Pedregosa et al. (2011); RAPIDS Development Team (2023)
RF	*n_estimators*, *max_depth*, *min_samples_split*, *min_samples_leaf*	Pedregosa et al. (2011); RAPIDS Development Team (2023)
XGBoost	*eta*, *subsample*, *n_estimators*, *reg_alpha*, *reg_lambda*, *max_depth*, *early_stopping_rounds*	Chen and Guestrin (2016)
LightGBM	*learning_rate*, *subsample*, *reg_alpha*, *reg_lambda*, *max_depth*, *num_boosting_rounds*, *early_stopping_rounds*	Ke et al. (2017)
CatBoost	*eta*, *n_estimators*, *reg_lambda*, *max_depth*, *early_stopping_rounds*	Prokhorenkova et al. (2018)
DNN	*embedding_size*, *learning_rate*, *num_layers*, *neurons_per_layer*, *epochs*, *patience*	Paszke et al. (2019)
TabNet	*cat_emb_dim*, *max_epochs*, *patience*	Arik and Pfister (2020); Dreamquark (2019)

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. https://doi.org/10.48550/arXiv.1907.10902

Aleem, S., Capretz, L.F., Ahmed, F., 2015. Benchmarking Machine Learning Technologies for Software Defect Detection. https://doi.org/10.48550/arXiv.1506.07563

Allen, R.J., Frenkel, D., ten Wolde, P.R., 2006. Simulating rare events in equilibrium or nonequilibrium stochastic systems. J. Chem. Phys. 124, 024102. https://doi.org/10.1063/1.2140273

Arik, S.O., Pfister, T., 2020. TabNet: Attentive Interpretable Tabular Learning. https://doi.org/10.48550/arXiv.1908.07442

Arjun, A., Bolhuis, P.G., 2023. Homogeneous nucleation of crystalline methane hydrate in molecular dynamics transition paths sampled under realistic conditions. J. Chem. Phys. 158, 044504. https://doi.org/10.1063/5.0124852

Arunthavanathan, R., Ahmed, S., Khan, F., Imtiaz, S., 2022. Machine Learning for Process Fault Detection and Diagnosis, in: Machine Learning in Chemical Safety and Health. John Wiley & Sons, Ltd, pp. 113–137. https://doi.org/10.1002/9781119817512.ch6

Ashraf, M.T., Dey, K., Mishra, S., 2023. Identification of high-risk roadway segments for wrong-way driving crash using rare event modeling and data augmentation techniques. Accid. Anal. Prev. 181, 106933. https://doi.org/10.1016/j.aap.2022.106933

Aven, T., 2020. Chapter Six - Rare event risk assessments, in: Khan, F.I., Amyotte, P.R. (Eds.), Methods in Chemical Process Safety, Advanced Methods of Risk Assessment and Management. Elsevier, pp. 205–237. https://doi.org/10.1016/bs.mcps.2020.02.003

Barata, J., Kayser, I., 2023. Industry 5.0 – Past, Present, and Near Future. Procedia Comput. Sci., CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2022 219, 778–788. https://doi.org/10.1016/j.procs.2023.01.351

Bécue, A., Praça, I., Gama, J., 2021. Artificial intelligence, cyber-threats and Industry 4.0: challenges and opportunities. Artif. Intell. Rev. 54, 3849–3886. https://doi.org/10.1007/s10462-020-09942-2

Belli, L., Davoli, L., Medioli, A., Marchini, P.L., Ferrari, G., 2019. Toward Industry 4.0 With IoT: Optimizing Business Processes in an Evolving Manufacturing Factory. Front. ICT 6.

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for Hyper-Parameter Optimization, in: Advances in Neural Information Processing Systems. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D., 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. Comput. Sci. Discov. 8, 014008. https://doi.org/10.1088/1749-4699/8/1/014008

Beyer, J., Trannum, H.C., Bakke, T., Hodson, P.V., Collier, T.K., 2016. Environmental effects of the Deepwater Horizon oil spill: A review. Mar. Pollut. Bull. 110, 28–51. https://doi.org/10.1016/j.marpolbul.2016.06.027

Bi, Y., Li, T., 2014. Probing Methane Hydrate Nucleation through the Forward Flux Sampling Method. J. Phys. Chem. B 118, 13324–13332. https://doi.org/10.1021/jp503000u

Bichri, H., Chergui, A., Hain, M., 2024. Investigating the Impact of Train / Test Split Ratio on the Performance of Pre-Trained Models with Custom Datasets. Int. J. Adv. Comput. Sci. Appl. 15. https://doi.org/10.14569/IJACSA.2024.0150235

Bolhuis, P.G., Chandler, D., Dellago, C., Geissler, P.L., 2002. Transition Path Sampling : Throwing Ropes Over Rough Mountain Passes, in the Dark. Annu. Rev. Phys. Chem. 53, 291–318. https://doi.org/10.1146/annurev.physchem.53.082301.113146

Borghini, E., Giannetti, C., 2021. Short Term Load Forecasting Using TabNet: A Comparative Study with Traditional State-of-the-Art Regression Models. Eng. Proc. 5, 6. https://doi.org/10.3390/engproc2021005006
Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G., 2022. Deep Neural Networks and Tabular Data: A Survey. IEEE Trans. Neural Netw. Learn. Syst. 1–21. https://doi.org/10.1109/TNNLS.2022.3229161
Borrero, E.E., Escobedo, F.A., 2007. Reaction coordinates and transition pathways of rare events via forward flux sampling. J. Chem. Phys. 127, 164101. https://doi.org/10.1063/1.2776270
Breiman, L., 2001. Random Forests. Mach. Learn. 45, 5–32. https://doi.org/10.1023/A:1010933404324
Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J., 2017. Chapter 8: Regression Trees, in: Classification and Regression Trees. Chapman and Hall/CRC.
Broughton, E., 2005. The Bhopal disaster and its aftermath: a review. Environ. Health 4, 6. https://doi.org/10.1186/1476-069X-4-6
Budach, L., Feuerpfeil, M., Ihde, N., Nathansen, A., Noack, N., Patzlaff, H., Naumann, F., Harmouch, H., 2022. The Effects of Data Quality on Machine Learning Performance. https://doi.org/10.48550/arXiv.2207.14529
Candanedo, I.S., Nieves, E.H., González, S.R., Martín, M.T.S., Briones, A.G., 2018. Machine Learning Predictive Model for Industry 4.0, in: Uden, L., Hadzima, B., Ting, I.-H. (Eds.), Knowledge Management in Organizations, Communications in Computer and Information Science. Springer International Publishing, Cham, pp. 501–510. https://doi.org/10.1007/978-3-319-95204-8_42
Cerna, S., Guyeux, C., Arcolezi, H.H., Couturier, R., Royer, G., 2020. A Comparison of LSTM and XGBoost for Predicting Firemen Interventions, in: Rocha, Á., Adeli, H., Reis, L.P., Costanzo, S., Orovic, I., Moreira, F. (Eds.), Trends and Innovations in Information Systems and Technologies, Advances in Intelligent Systems and Computing. Springer International Publishing, Cham, pp. 424–434. https://doi.org/10.1007/978-3-030-45691-7_39
Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. https://doi.org/10.1145/2939672.2939785
Chen, Y., Liu, Y., Chen, L., Zhang, Y., 2021. DialogSum: A Real-Life Scenario Dialogue Summarization Dataset. https://doi.org/10.48550/arXiv.2105.06762
Claesen, M., Simm, J., Popovic, D., Moreau, Y., De Moor, B., 2014. Easy Hyperparameter Search Using Optunity. https://doi.org/10.48550/arXiv.1412.1114
Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20, 273–297. https://doi.org/10.1007/BF00994018
Crafts, N., 2011. Explaining the first Industrial Revolution: two views. Eur. Rev. Econ. Hist. 15, 153–168. https://doi.org/10.1017/S1361491610000201
Culot, G., Fattori, F., Podrecca, M., Sartor, M., 2019. Addressing Industry 4.0 Cybersecurity Challenges. IEEE Eng. Manag. Rev. 47, 79–86. https://doi.org/10.1109/EMR.2019.2927559
Danielsson, P.-E., 1980. Euclidean distance mapping. Comput. Graph. Image Process. 14, 227–248. https://doi.org/10.1016/0146-664X(80)90054-4
Daoud, E.A., 2019. Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset. Int. J. Comput. Inf. Eng. 13, 6–10.
Dellago, C., Bolhuis, P.G., Csajka, F.S., Chandler, D., 1998. Transition path sampling and the calculation of rate constants. J. Chem. Phys. 108, 1964–1977. https://doi.org/10.1063/1.475562
Dellago, C., Bolhuis, P.G., Geissler, P.L., 2002. Transition Path Sampling, in: Prigogine, I., Rice, S.A. (Eds.), Advances in Chemical Physics. Wiley, pp. 1–78. https://doi.org/10.1002/0471231509.ch1
Demir, K.A., Döven, G., Sezen, B., 2019. Industry 5.0 and Human-Robot Co-working. Procedia Comput. Sci., 3rd World Conference on Technology, Innovation and Entrepreneurship "Industry 4.0 Focused Innovation, Technology, Entrepreneurship and Manufacture" June 21-23, 2019 158, 688–695. https://doi.org/10.1016/j.procs.2019.09.104

Dingli, A., Haddod, F., Klüver, C. (Eds.), 2021. Artificial Intelligence in Industry 4.0: A Collection of Innovative Research Case-studies that are Reworking the Way We Look at Industry 4.0 Thanks to Artificial Intelligence, Studies in Computational Intelligence. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-61045-6

dos Santos, C., Gatti, M., 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts, in: Tsujii, J., Hajic, J. (Eds.), Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. Presented at the COLING 2014, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pp. 69–78.

Dreamquark, 2019. pytorch_tabnet documentation.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V., 1996. Support Vector Regression Machines, in: Advances in Neural Information Processing Systems. MIT Press.

Ervural, B.C., Ervural, B., 2018. Overview of Cyber Security in the Industry 4.0 Era, in: Ustundag, A., Cevikcan, E. (Eds.), Industry 4.0: Managing The Digital Transformation, Springer Series in Advanced Manufacturing. Springer International Publishing, Cham, pp. 267–284. https://doi.org/10.1007/978-3-319-57870-5_16

European Commission, 2020. Industry 5.0 [WWW Document]. URL https://research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/industry-50_en (accessed 6.18.24).

Feltes, B.C., Chandelier, E.B., Grisci, B.I., Dorn, M., 2019. CuMiDa: An Extensively Curated Microarray Database for Benchmarking and Testing of Machine Learning Approaches in Cancer Research. J. Comput. Biol. 26, 376–386. https://doi.org/10.1089/cmb.2018.0238

Filion, L., Hermes, M., Ni, R., Dijkstra, M., 2010. Crystal nucleation of hard spheres using molecular dynamics, umbrella sampling, and forward flux sampling: A comparison of simulation techniques. J. Chem. Phys. 133, 244115. https://doi.org/10.1063/1.3506838

Gastegger, M., Behler, J., Marquetand, P., 2017. Machine learning molecular dynamics for the simulation of infrared spectra. Chem. Sci. 8, 6924–6935. https://doi.org/10.1039/C7SC02267K

Ghobakhloo, M., Iranmanesh, M., Tseng, M.-L., Grybauskas, A., Stefanini, A., Amran, A., 2023. Behind the definition of Industry 5.0: a systematic review of technologies, principles, components, and values. J. Ind. Prod. Eng. 40, 432–447. https://doi.org/10.1080/21681015.2023.2216701

Gokalp, M.O., Kayabay, K., Akyol, M.A., Eren, P.E., Koçyiğit, A., 2016. Big Data for Industry 4.0: A Conceptual Framework, in: 2016 International Conference on Computational Science and Computational Intelligence (CSCI). Presented at the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 431–434. https://doi.org/10.1109/CSCI.2016.0088

Grinsztajn, L., Oyallon, E., Varoquaux, G., 2022. Why do tree-based models still outperform deep learning on typical tabular data? Adv. Neural Inf. Process. Syst. 35, 507–520.

Gupta, J.P., 2002. The Bhopal gas tragedy: could it have happened in a developed country? J. Loss Prev. Process Ind. 15, 1–4. https://doi.org/10.1016/S0950-4230(01)00025-0

Hailwood, M., 2016. Learning from Accidents – Reporting is not Enough. Chem. Eng. Trans. 48, 709–714. https://doi.org/10.3303/CET1648119

Hancock, J., Khoshgoftaar, T.M., 2020. Performance of CatBoost and XGBoost in Medicare Fraud Detection, in: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). Presented at the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 572–579. https://doi.org/10.1109/ICMLA51294.2020.00095

Hanna, B.N., Dinh, N.T., Youngblood, R.W., Bolotnov, I.A., 2020. Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD). Prog. Nucl. Energy 118, 103140. https://doi.org/10.1016/j.pnucene.2019.103140

Harkat, M.-F., Kouadri, A., Fezai, R., Mansouri, M., Nounou, H., Nounou, M., 2020. Machine Learning-Based Reduced Kernel PCA Model for Nonlinear Chemical Process Monitoring. J. Control Autom. Electr. Syst. 31, 1196–1209. https://doi.org/10.1007/s40313-020-00604-w

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hartmann, C., Banisch, R., Sarich, M., Badowski, T., Schütte, C., 2014. Characterization of Rare Events in Molecular Dynamics. Entropy 16, 350–376. https://doi.org/10.3390/e16010350

Hashimoto, Y., Toyoshima, T., Yogo, S., Koike, M., Hamaguchi, T., Jing, S., Koshijima, I., 2013. Safety securing approach against cyber-attacks for process control system. Comput. Chem. Eng., PSE-2012. 57, 181–186. https://doi.org/10.1016/j.compchemeng.2013.04.019

Hassoun, M.H., 1995. Fundamentals of Artificial Neural Networks. MIT Press.

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Zhu, X., Wang, J., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., Avestimehr, S., 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. https://doi.org/10.48550/arXiv.2007.13518

Holmstrom, D., Altamirano, F., Banks, J., Joseph, G., Kaszniak, M., Mackenzie, C., Shroff, R., Cohen, H., Wallace, S., 2006. CSB investigation of the explosions and fire at the BP texas city refinery on March 23, 2005. Process Saf. Prog. 25, 345–349. https://doi.org/10.1002/prs.10158

Hunter, J.D., 2007. Matplotlib: A 2D Graphics Environment. Comput. Sci. Eng. 9, 90–95. https://doi.org/10.1109/MCSE.2007.55

Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., Munigala, V., 2020. Overview and Importance of Data Quality for Machine Learning Tasks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20. Association for Computing Machinery, New York, NY, USA, pp. 3561–3562. https://doi.org/10.1145/3394486.3406477

Jiang, H., Haji-Akbari, A., Debenedetti, P.G., Panagiotopoulos, A.Z., 2018. Forward flux sampling calculation of homogeneous nucleation rates from aqueous NaCl solutions. J. Chem. Phys. 148, 044505. https://doi.org/10.1063/1.5016554

Joseph, L.P., Joseph, E.A., Prasad, R., 2022. Explainable diabetes classification using hybrid Bayesian-optimized TabNet architecture. Comput. Biol. Med. 151, 106178. https://doi.org/10.1016/j.compbiomed.2022.106178

Kahloot, K.M., Ekler, P., 2021. Algorithmic Splitting: A Method for Dataset Preparation. IEEE Access 9, 125229–125237. https://doi.org/10.1109/ACCESS.2021.3110745

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.-Y., 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.

Kim, J.H., 2017. A Review of Cyber-Physical System Research Relevant to the Emerging IT Trends: Industry 4.0, IoT, Big Data, and Cloud Computing. J. Ind. Integr. Manag. 02, 1750011. https://doi.org/10.1142/S2424862217500117

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980

Kitchin, J.R., 2018. Machine learning in catalysis. Nat. Catal. 1, 230–232. https://doi.org/10.1038/s41929-018-0056-y

Kochkov, D., Smith, J.A., Alieva, A., Wang, Q., Brenner, M.P., Hoyer, S., 2021. Machine learning–accelerated computational fluid dynamics. Proc. Natl. Acad. Sci. 118, e2101784118. https://doi.org/10.1073/pnas.2101784118

Kumari, P., Bhadriraju, B., Wang, Q., Kwon, J.S.-I., 2021. Development of parametric reduced-order model for consequence estimation of rare events. Chem. Eng. Res. Des. 169, 142–152. https://doi.org/10.1016/j.cherd.2021.02.006
Labib, A., Harris, M.J., 2015. Learning how to learn from failures: The Fukushima nuclear disaster. Eng. Fail. Anal. 47, 117–128. https://doi.org/10.1016/j.engfailanal.2014.10.002
Lam, S.K., Pitrou, A., Seibert, S., 2015. Numba: a LLVM-based Python JIT compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15. Association for Computing Machinery, New York, NY, USA, pp. 1–6. https://doi.org/10.1145/2833157.2833162
Lavecchia, A., 2015. Machine-learning approaches in drug discovery: methods and applications. Drug Discov. Today 20, 318–331. https://doi.org/10.1016/j.drudis.2014.10.012
Li, W., Yin, Y., Quan, X., Zhang, H., 2019. Gene Expression Value Prediction Based on XGBoost Algorithm. Front. Genet. 10. https://doi.org/10.3389/fgene.2019.01077
Li, Z., Ding, Q., Zhang, W., 2011. A Comparative Study of Different Distances for Similarity Estimation, in: Chen, R. (Ed.), Intelligent Computing and Information Science, Communications in Computer and Information Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 483–488. https://doi.org/10.1007/978-3-642-18129-0_75
Liang, W., Luo, S., Zhao, G., Wu, H., 2020. Predicting Hard Rock Pillar Stability Using GBDT, XGBoost, and LightGBM Algorithms. Mathematics 8, 765. https://doi.org/10.3390/math8050765
Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I., 2018. Tune: A Research Platform for Distributed Model Selection and Training. https://doi.org/10.48550/arXiv.1807.05118
Linardatos, P., Papastefanopoulos, V., Kotsiantis, S., 2021. Explainable AI: A Review of Machine Learning Interpretability Methods. Entropy 23, 18. https://doi.org/10.3390/e23010018
Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, pp. 4768–4777.
Ma, M., Zhao, G., He, B., Li, Q., Dong, H., Wang, S., Wang, Z., 2021. XGBoost-based method for flash flood risk assessment. J. Hydrol. 598, 126382. https://doi.org/10.1016/j.jhydrol.2021.126382
Ma, S., Zhang, X., Jia, C., Zhao, Z., Wang, Shiqi, Wang, Shanshe, 2020. Image and Video Compression With Neural Networks: A Review. IEEE Trans. Circuits Syst. Video Technol. 30, 1683–1698. https://doi.org/10.1109/TCSVT.2019.2910119
McDonnell, K., Murphy, F., Sheehan, B., Masello, L., Castignani, G., 2023. Deep learning in insurance: Accuracy and model interpretability using TabNet. Expert Syst. Appl. 217, 119543. https://doi.org/10.1016/j.eswa.2023.119543
McKinney, W., 2010. Data Structures for Statistical Computing in Python. Proc. 9th Python Sci. Conf. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a
Mohajan, H., 2021. Third Industrial Revolution Brings Global Development. J. Soc. Sci. Humanit. 7, 239–251.
Mohajan, H., 2019. The First Industrial Revolution: Creation of a New Global Human Era. J. Soc. Sci. Humanit. 5, 377–387.
Mokyr, J., Strotz, R.H., 1998. The second industrial revolution, 1870-1914. Storia Dell'economia Mond. 21945.
Morgan, D., Jacobs, R., 2020. Opportunities and Challenges for Machine Learning in Materials Science. Annu. Rev. Mater. Res. 50, 71–103. https://doi.org/10.1146/annurev-matsci-070218-010015
Moskowitz, I.H., Seider, W.D., Patel, A.J., Arbogast, J.E., Oktem, U.G., 2018. Understanding rare safety and reliability events using transition path sampling. Comput. Chem. Eng. 108, 74–88. https://doi.org/10.1016/j.compchemeng.2017.06.016
Motz, M., Krauß, J., Schmitt, R.H., 2022. Benchmarking of hyperparameter optimization techniques for machine learning applications in production. Adv. Ind. Manuf. Eng. 5, 100099. https://doi.org/10.1016/j.aime.2022.100099

Naboni, R., Paoletti, I., 2015. The Third Industrial Revolution, in: Naboni, R., Paoletti, I. (Eds.), Advanced Customization in Architectural Design and Construction. Springer International Publishing, Cham, pp. 7–27. https://doi.org/10.1007/978-3-319-04423-1_2
Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 807–814.
NVIDIA, Vingelmann, P., Fitzek, F.H.P., 2022. NVIDIA CUDA Toolkit 11.8.90.
Ogunleye, A., Wang, Q.-G., 2020. XGBoost Model for Chronic Kidney Disease Diagnosis. IEEE/ACM Trans. Comput. Biol. Bioinform. 17, 2131–2140. https://doi.org/10.1109/TCBB.2019.2911071
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/arXiv.1912.01703
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Peters, B., Trout, B.L., 2006. Obtaining reaction coordinates by likelihood maximization. J. Chem. Phys. 125, 054108. https://doi.org/10.1063/1.2234477
Pfisterer, F., Beggel, L., Sun, X., Scheipl, F., Bischl, B., 2021. Benchmarking time series classification -- Functional data vs machine learning approaches. https://doi.org/10.48550/arXiv.1911.07511
Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. CatBoost: unbiased boosting with categorical features, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.
Purushotham, S., Meng, C., Che, Z., Liu, Y., 2018. Benchmarking deep learning models on large healthcare datasets. J. Biomed. Inform. 83, 112–134. https://doi.org/10.1016/j.jbi.2018.04.007
Quatrini, E., Costantino, F., Di Gravio, G., Patriarca, R., 2020. Machine learning for anomaly detection and process phase classification to improve safety and maintenance activities. J. Manuf. Syst. 56, 117–132. https://doi.org/10.1016/j.jmsy.2020.05.013
Raja Santhi, A., Muthuswamy, P., 2023. Industry 5.0 or industry 4.0S? Introduction to industry 4.0 and a peek into the prospective industry 5.0 technologies. Int. J. Interact. Des. Manuf. IJIDeM 17, 947–979. https://doi.org/10.1007/s12008-023-01217-8
RAPIDS Development Team, 2023. RAPIDS: Libraries for End to End GPU Data Science.
Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16. Association for Computing Machinery, New York, NY, USA, pp. 1135–1144. https://doi.org/10.1145/2939672.2939778
Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. 65, 386–408. https://doi.org/10.1037/h0042519
Ruder, S., 2017. An overview of gradient descent optimization algorithms. https://doi.org/10.48550/arXiv.1609.04747
Russo, L.P., Bequette, B.W., 1998. Operability of chemical reactors: multiplicity behavior of a jacketed styrene polymerization reactor. Chem. Eng. Sci. 53, 27–45. https://doi.org/10.1016/S0009-2509(97)00281-9
Saenko, V., Ivanov, V., Tsyb, A., Bogdanova, T., Tronko, M., Demidchik, Yu., Yamashita, S., 2011. The Chernobyl Accident and its Consequences. Clin. Oncol., The Radiobiological Consequences of the Chernobyl Accident 25 Years On - April 2011 23, 234–243. https://doi.org/10.1016/j.clon.2011.01.502
Sarkar, S., Vinay, S., Raj, R., Maiti, J., Mitra, P., 2019. Application of optimized machine learning techniques for prediction of occupational accidents. Comput. Oper. Res. 106, 210–224. https://doi.org/10.1016/j.cor.2018.02.021

Shekhar, S., Bansode, A., Salim, A., 2022. A Comparative study of Hyper-Parameter Optimization Tools. https://doi.org/10.48550/arXiv.2201.06433
Shwartz-Ziv, R., Armon, A., 2022. Tabular data: Deep learning is not all you need. Inf. Fusion 81, 84–90. https://doi.org/10.1016/j.inffus.2021.11.011
Shyalika, C., Wickramarachchi, R., Sheth, A., 2023. A Comprehensive Survey on Rare Event Prediction. https://doi.org/10.48550/arXiv.2309.11356
Singh, S.P., Kumar, A., Darbari, H., Singh, L., Rastogi, A., Jain, S., 2017. Machine translation using deep learning: An overview, in: 2017 International Conference on Computer, Communications and Electronics (Comptelix). Presented at the 2017 International Conference on Computer, Communications and Electronics (Comptelix), pp. 162–167. https://doi.org/10.1109/COMPTELIX.2017.8003957
So, B., 2024. Enhanced gradient boosting for zero-inflated insurance claims and comparative analysis of CatBoost, XGBoost, and LightGBM. Scand. Actuar. J. 0, 1–23. https://doi.org/10.1080/03461238.2024.2365390
Soori, M., Arezoo, B., Dastres, R., 2023. Internet of things for smart factories in industry 4.0, a review. Internet Things Cyber-Phys. Syst. 3, 192–204. https://doi.org/10.1016/j.iotcps.2023.04.006
Sriramachari, S., 2004. The Bhopal gas tragedy: An environmental disaster. Curr. Sci. 86, 905–920.
Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C., 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Netw., Selected Papers from IJCNN 2011 32, 323–332. https://doi.org/10.1016/j.neunet.2012.02.016
Sudarshan, V., Seider, W.D., Patel, A.J., Arbogast, J.E., 2021. Understanding rare safety and reliability events using forward-flux sampling. Comput. Chem. Eng. 153, 107387. https://doi.org/10.1016/j.compchemeng.2021.107387
Sudarshan, V., Seider, W.D., Patel, A.J., Oktem, U.G., Arbogast, J.E., 2024a. Alarm rationalization and dynamic risk analyses for rare abnormal events. Comput. Chem. Eng. 184, 108633. https://doi.org/10.1016/j.compchemeng.2024.108633
Sudarshan, V., Seider, W.D., Patel, A.J., Oktem, U.G., Arbogast, J.E., 2024b. Path-Sampling and Machine learning for rare abnormal Events: Application to polymerization CSTRs. Chem. Eng. Sci. 120513. https://doi.org/10.1016/j.ces.2024.120513
Sudarshan, V., Seider, W.D., Patel, A.J., Oktem, U.G., Arbogast, J.E., 2023. Multivariate alarm systems to recognize rare unpostulated abnormal events. AIChE J. 70, e18284. https://doi.org/10.1002/aic.18284
Suwanda, R., Syahputra, Z., Zamzami, E.M., 2020. Analysis of Euclidean Distance and Manhattan Distance in the K-Means Algorithm for Variations Number of Centroid K. J. Phys. Conf. Ser. 1566, 012058. https://doi.org/10.1088/1742-6596/1566/1/012058
Tamascelli, N., Solini, R., Paltrinieri, N., Cozzani, V., 2022. Learning from major accidents: A machine learning approach. Comput. Chem. Eng. 162, 107786. https://doi.org/10.1016/j.compchemeng.2022.107786
Thiyagalingam, J., Shankar, M., Fox, G., Hey, T., 2022. Scientific machine learning benchmarks. Nat. Rev. Phys. 4, 413–420. https://doi.org/10.1038/s42254-022-00441-7
Toyao, T., Maeno, Z., Takakusagi, S., Kamachi, T., Takigawa, I., Shimizu, K., 2020. Machine Learning for Catalysis Informatics: Recent Applications and Prospects. ACS Catal. 10, 2260–2297. https://doi.org/10.1021/acscatal.9b04186
Traore, B.B., Kamsu-Foguem, B., Tangara, F., 2018. Deep convolution neural network for image recognition. Ecol. Inform. 48, 257–268. https://doi.org/10.1016/j.ecoinf.2018.10.002
Uddin, S., Lu, H., 2024. Confirming the statistically significant superiority of tree-based machine learning algorithms over their counterparts for tabular data. PLOS ONE 19, e0301541. https://doi.org/10.1371/journal.pone.0301541
U.S. Chemical Safety and Hazard Investigation Board, 2007. BP America (Texas City) Refinery Explosion Investigation Report.

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., Zhao, S., 2019. Applications of machine learning in drug discovery and development. Nat. Rev. Drug Discov. 18, 463–477. https://doi.org/10.1038/s41573-019-0024-5

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vrigazova, B., 2021. The Proportion for Splitting Data into Training and Test Set for the Bootstrap in Classification Problems. Bus. Syst. Res. Int. J. Soc. Adv. Innov. Res. Econ. 12, 228–242.

Wang, T., Bian, Y., Zhang, Y., Hou, X., 2023. Classification of earthquakes, explosions and mining-induced earthquakes based on XGBoost algorithm. Comput. Geosci. 170, 105242. https://doi.org/10.1016/j.cageo.2022.105242

Wang, Y., Lamim Ribeiro, J.M., Tiwary, P., 2020. Machine learning approaches for analyzing and enhancing molecular dynamics simulations. Curr. Opin. Struct. Biol., Theory and Simulation Macromolecular Assemblies 61, 139–145. https://doi.org/10.1016/j.sbi.2019.12.016

Watanabe, S., 2023. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. https://doi.org/10.48550/arXiv.2304.11127

Wei, J., Chu, X., Sun, X.-Y., Xu, K., Deng, H.-X., Chen, J., Wei, Z., Lei, M., 2019. Machine learning in materials science. InfoMat 1, 338–358. https://doi.org/10.1002/inf2.12028

Wu, Z., Ramsundar, B., N. Feinberg, E., Gomes, J., Geniesse, C., S. Pappu, A., Leswing, K., Pande, V., 2018. MoleculeNet: a benchmark for molecular machine learning. Chem. Sci. 9, 513–530. https://doi.org/10.1039/C7SC02664A

xgboost developers, 2023. xgboost Release 1.7.6.

Xie, J., Wang, Q., 2020. Benchmarking Machine Learning Algorithms on Blood Glucose Prediction for Type I Diabetes in Comparison With Classical Time-Series Models. IEEE Trans. Biomed. Eng. 67, 3101–3124. https://doi.org/10.1109/TBME.2020.2975959

Yan, J., Xu, T., Yu, Y., Xu, H., 2021. Rainfall Forecast Model Based on the TabNet Model. Water 13, 1272. https://doi.org/10.3390/w13091272

Zou, J., Han, Y., So, S.-S., 2009. Overview of Artificial Neural Networks, in: Livingstone, D.J. (Ed.), Artificial Neural Networks: Methods and Applications. Humana Press, Totowa, NJ, pp. 14–22. https://doi.org/10.1007/978-1-60327-101-1_2