Acta Geotechnica
https://doi.org/10.1007/s11440-023-02117-7

RESEARCH PAPER

Machine learning with monotonic constraint for geotechnical engineering applications: an example of slope stability prediction

Te Pei[1] · Tong Qiu[2]

Received: 28 November 2022 / Accepted: 24 October 2023

Abstract
Machine learning (ML) algorithms have been widely applied to analyze geotechnical engineering problems due to recent advances in data science. However, flexible ML models trained with limited data can exhibit unexpected behaviors, leading to low interpretability and physical inconsistency, thus, reducing the reliability and robustness of ML models for risk forecasting and engineering applications. As input features for geotechnical engineering applications often represent physical parameters following intrinsic and often monotonic relationships, incorporating monotonicity into ML models can help ensure the physical realism of model outputs. In this study, monotonicity was introduced as a soft constraint into artificial neural network (ANN) models, and their results were compared with several benchmark ML models. During the training process, data augmentation and point-wise gradient were used to evaluate the monotonicity of model predictions, and monotonicity violations were minimized through a modified loss function. A compilation of slope stability case histories from the literature was used for model development, benchmarking their performance, and evaluating the effects of monotonicity constraints. Cross-validation procedures were used for all model performance evaluations to reduce bias in sample selections. Results showed that unconstrained ML models produced predictions that violate monotonicity in many parts of the input space. However, by adding monotonicity constraints into ANN models, monotonicity violations were effectively reduced while maintaining relatively high performance, thus providing a more robust and interpretable prediction. Using slope stability prediction as a proxy, the methods developed in this study to incorporate monotonicity constraints into ML models can be applied to many geotechnical engineering applications. The proposed approach enhances the reliability and interpretability of ML models, resulting in more accurate and consistent outcomes for real-world applications.

Keywords Interpretability · Machine learning · Monotonicity · Slope stability

1 Introduction

Slope stability analysis is an important component of landslide risk assessment. As communities expand into hilly terrains to accommodate continued population and economic growth, an increasing number of infrastructure systems become vulnerable to geohazards, including slope failures. Furthermore, climate change is expected to cause more frequent extreme rainfalls and wildfires, which may result in a higher occurrence of landslides and related damages [12]. Therefore, predicting slope stability with high levels of accuracy and efficiency is essential.

Slope stability evaluation has been traditionally accomplished using physics-based approaches, which depend on physical laws from soil mechanics to determine slope stability conditions. The factor of safety based on limit equilibrium methods (LEMs) (e.g., [2, 29, 40]) or numerical methods such as finite element methods (FEMs) (e.g., [14]) has been widely used to quantify slope stability. However, due to the complex nature of hillslope subsurface

✉ Tong Qiu
tuq1@psu.edu

Te Pei
tpei@ccny.cuny.edu

[1] Department of Civil Engineering, The City University of New York (City College), New York, NY 10031, USA

[2] Department of Civil and Environmental Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Published online: 28 November 2023

conditions, successful determination of slope stability using physics-based methods requires extensive site investigation and analysis, limiting the scale and efficiency of slope stability analysis as well as its applicability to risk management over a large region. On the other hand, data-driven methods, which use statistical and machine learning (ML) methods to develop functional relationships between input and output, have been widely used by researchers to predict slope stability (e.g., [9, 18, 24, 26, 28, 32, 34, 42, 43, 51, 53]). However, these methods were mainly developed and validated on small datasets as slope stability case histories are relatively scarce. Based on the literature review by Mahmoodzadeh et al. [34] and Lin et al. [28], only 169 samples were used on average for the development of data-driven models in slope stability studies (see Fig. 1). Hence, these studies do not belong in the big data category in the context of ML applications. In Fig. 1, two studies with dataset sizes between 601 and 700 samples used simulated datasets generated by geotechnical engineering software (e.g., [16]); these datasets were synthetic and different from actual slope case histories.

Although ML models are powerful in extracting features from data, they usually need a sufficient amount of data to achieve high performance. Flexible ML models with large hypothesis space may have unexpected behavior in parts of the input space not covered by training data, reducing the reliability and interpretability of ML models for risk forecasting and engineering applications [50]. Therefore, ML models should be constrained to improve their generalization capability. Figure 2 illustrates the model response for a flexible model and a monotonically constrained model for predicting the same dataset. As shown in Fig. 2, the monotonically constrained model can better represent the general trend for the dataset. The monotonic relationships

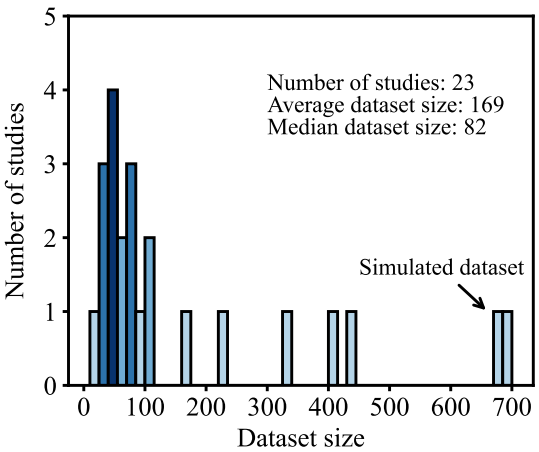


Fig. 1 Histogram showing dataset size used in previous studies for slope stability prediction

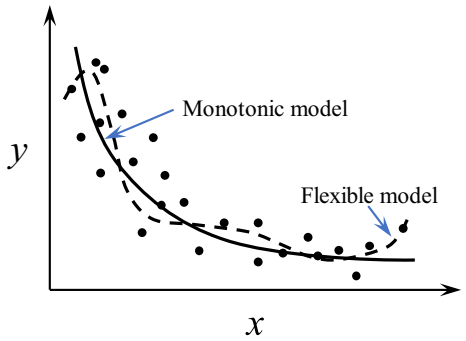


Fig. 2 Illustration of ML model predictions based on a flexible model vs. a monotonic model

between input and output variables are common prior knowledge in ML applications [20]. The input parameters for slope stability predictions often represent physical parameters, such as soil strength parameters, hydrological parameters, and slope geometries, and usually contribute monotonically to slope stability conditions based on principles of soil mechanics (e.g., [31]). For example, with everything else being equal, the factor of safety is expected to increase monotonically as soil strength parameters increase and decrease as the slope angle increases. However, ML models trained purely on data may not reflect these physical relationships resulting in low interpretability and poor generalization capability. Moreover, standard ML models often have difficulties capturing monotonic relationships directly from data, even if the dataset is monotonic [15]. For example, Li and Wang [24] developed an artificial neural network (ANN) model and a support vector machine (SVM) model based on slope case histories to predict slope stability conditions and factors of safety. They conducted a sensitivity analysis and compared model predictions with empirical equations. Results showed that the trained ANN model's predictions exhibited non-monotonic behavior with respect to input features and were contradictory to our physical understanding of slope failure (e.g., the predicted slope stability decreases as soil cohesion increases); the SVM model, on the other hand, has better generalization capability than the ANN model. Zhou et al. [53] conducted a sensitivity analysis of their trained ML models for slope stability predictions, which also exhibited strong non-monotonic relationships with respect to changes in input variables, limiting the interpretability and generalization capability of their models.

Monotonicity is desirable for ML models to ensure the physical consistency of model outputs. Recent research has focused on adding monotonicity or interaction constraints into ML models to address the fairness, interpretability, and generalization issues in various scientific domains [27]. For example, Daw et al. [10] embedded a monotonicity

constraint into the loss function to model lake temperatures at variable depths. Their monotonicity constraint was designed to ensure the model prediction reflects a monotonic relationship between water depth and density of water (i.e., denser water should be at a deeper depth) and provide physically consistent results. In addition, Stanley et al. [45, 47] applied monotonic constraints to their ML models for landslide susceptibility mapping, in which a direction was assigned to model response for each input variable using prior knowledge based on their contribution to landslide risk (e.g., non-decreasing monotonicity between soil moisture and landslide risk). Bandai and Ghezzehei [4] developed a physics-informed neural network (PINN) for the inverse solution of the Richardson-Richards equation and the estimations of water retention curves and hydraulic conductivity functions. The weight parameters of their PINN model were constrained to be non-negative such that the network can produce positive monotonic relationships for water retention curves and hydraulic conductivity functions with respect to matric potential. Monotonicity constraints have been incorporated into conventional ML algorithms and statistical models (e.g., [3, 5–8, 15, 25, 46]). In addition, with recent advances in deep learning, the current research interest has focused on incorporating monotonicity constraints efficiently into neural networks, which can be achieved by either using specifically designed architecture (e.g., [1, 36, 52]) or altering the learning process through regularization (e.g., [17, 27]).

Although embedding monotonicity constraints into ML models has been a trending topic in the data science community, there is a lack of monotonicity and generalization capability of existing ML models for geotechnical engineering applications (e.g., slope stability predictions). The primary purpose of this study is to improve the robustness and interpretability of ML models in the geotechnical engineering domain by embedding monotonicity constraints guided by prior knowledge, using slope stability predictions as an example. Two tasks were accomplished in this study: (1) the monotonicity consistency of commonly used ML models was evaluated, and (2) monotonicity constraints were incorporated into ML models, and their effects on model performance were evaluated. In the following sections, a database used to develop and validate the ML models is first described, followed by descriptions of ML models, the framework for evaluating monotonicity inconsistency, implementation of monotonic constraints, and the design of experiments. Lastly, the performance of these models was compared and evaluated, and the effects of monotonicity constraints were demonstrated.

2 Input parameters for slope stability prediction

The stability of a slope is a combined effect of gravity, soil properties, hydrologic conditions, and slope geometries. In geotechnical engineering, six parameters are commonly used to represent these effects (e.g., [31, 44]); they are soil unit weight (γ), cohesion (c), internal friction angle (ϕ), slope angle (β), slope height (H), and pore pressure ratio (r_u). Values of these parameters have been well-documented in slope failure case histories and have been widely used for developing data-driven models for slope stability predictions (e.g., [28, 53]). Among these parameters, the soil strength parameters c and ϕ are stabilizing factors, and γ, β, H, and r_u are often considered destabilizing factors based on typical circular failure mechanisms for soil slopes (e.g., [30, 31]).

It should be noted that when analyzing slope stability problems, the complex soil profile and the resultant complex slip surface are important considerations but not incorporated as input parameters in this study. The intent of this study is to demonstrate the advantages of incorporating monotonicity constraints in ML models through a classical geotechnical engineering application that has been widely studied using traditional ML methods; the selection of these six input parameters is more suitable for slopes with a relatively uniform soil profile for simplicity.

2.1 Database description

In the present study, a slope case history database compiled by Hoang and Pham [18] was used to develop and evaluate the performance of ML models and the effects of monotonicity constraints. The database contains 168 slope case histories from previously published literature for circular failure slopes. The database reports values of six influencing factors related to slope stability (i.e., γ, c, ϕ, β, H, and r_u) and slope stability conditions (i.e., stable/failure) for each record in the database. Most of these slopes are soil or highly weathered rock slopes, except several slopes with large cohesion values, which are likely to be rock slopes. For the present study, three samples in the dataset with $c > 200$ kPa were removed and 165 samples remained, including 83 positive samples (i.e., stable slopes) and 82 negative samples (failed slopes). Basic descriptive statistical analysis for the reduced database is summarized in Table 1. Figure 3 presents box plots showing the distribution of each influencing factor. The box corresponds to the likely range of variation (i.e., between the first and third quartiles). The lines extending from the bottom and top of each box mark the minimum and maximum values, respectively, within a statistically acceptable range. Any

Table 1 Descriptive statistics of the database used in the present study

	γ (kN/m^3)	c (kPa)	ϕ (deg)	β (deg)	H (m)	r_u (-)
Mean	21.67	29.89	28.57	35.81	103.83	0.22
Std	4.13	33.22	10.63	10.07	134.19	0.16
Min	12.00	0.00	0.00	16.00	3.60	0.00
25th	18.84	10.05	24.50	30.00	12.00	0.00
50th	20.96	19.96	30.15	35.00	50.00	0.25
75th	25.00	37.50	36.00	45.00	115.00	0.30
Max	31.30	150.05	45.00	59.00	511.00	0.50

value outside this range, called an outlier, is displayed as an individual point. Figure 3 shows that the values of each influencing factor cover a wide range of variations, and distributions of these influencing factors are not uniform. Therefore, data standardization was used for scaling the database to ensure model training results. Figure 4 presents the correlation matrix plot for the samples in the dataset, which visualizes pairwise relationships between input attributes. In Fig. 4, the central diagonal plots are the univariate histograms for each feature. The plots below the main-diagonal show scatter plots with fitted regression lines for any pairs of features in the dataset; the numbers above the central diagonal are the correlation coefficient between features. As shown in Fig. 4, no strong correlation was found between input features except a moderate correlation between γ and H (i.e., $r = 0.64$) and between ϕ and β (i.e., $r = 0.60$).

3 Methodologies

3.1 Definition of individual monotonicity

The objective of an ML model is to learn a mathematical model f that maps from an input space X to an output space Y using samples $(x, y) \in (X \times Y)$. Following the formulation proposed by Liu et al. [27], who assumed that the input x is partitioned into $x=[x_\alpha, x_{\neg\alpha}]$ where α is an input feature of interest and $\neg\alpha$ is its complement (i.e., all other features), x_α and $x_{\neg\alpha}$ are the corresponding sub-vectors of x. The model f is monotonically increasing with respect to feature α if $f(x_\alpha, x_{\neg\alpha}) \geq f(x'_\alpha, x_{\neg\alpha})$ and monotonically decreasing with respect to feature α if $f(x_\alpha, x_{\neg\alpha}) \leq f(x'_\alpha, x_{\neg\alpha})$, for all $x_\alpha \geq x'_\alpha$, where x_α and x'_α are sub-vectors of feature α, and $x_\alpha \geq x'_\alpha$ denotes the inequality for all the elements (i.e., $x_i \geq x'_i$ for all $i \in \alpha$). Model predictions that disobey monotonic relationships for each of these two categories can be considered a violation of the corresponding monotonicity and f is said to be individually monotonic on x if no monotonic adversarial examples can be found.

In the present study, increasing monotonicity was enforced and evaluated for soil strength parameters c and ϕ, and soil unit weight γ (i.e., the slope becomes more stable as these parameters increase), whereas decreasing monotonicity was enforced and evaluated for slope geometry parameters β and H, and hydrological condition parameter r_u (i.e., the slope becomes less stable as these parameters increase). Based on this setup, monotonicity constraints were applied to all features. It should be noted

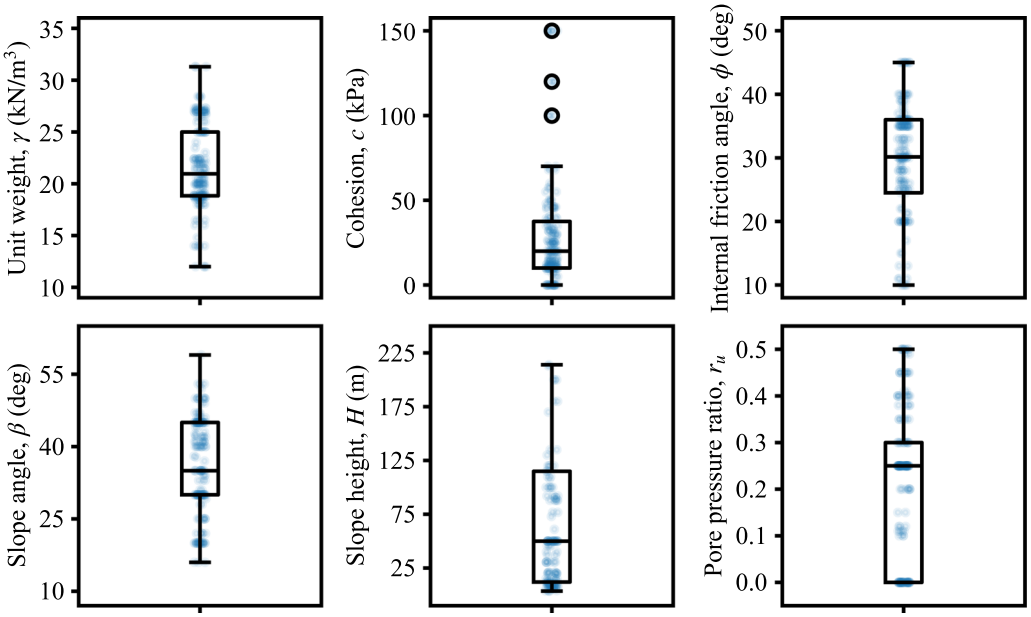


Fig. 3 Box plots showing the distribution of each influencing factor in the database

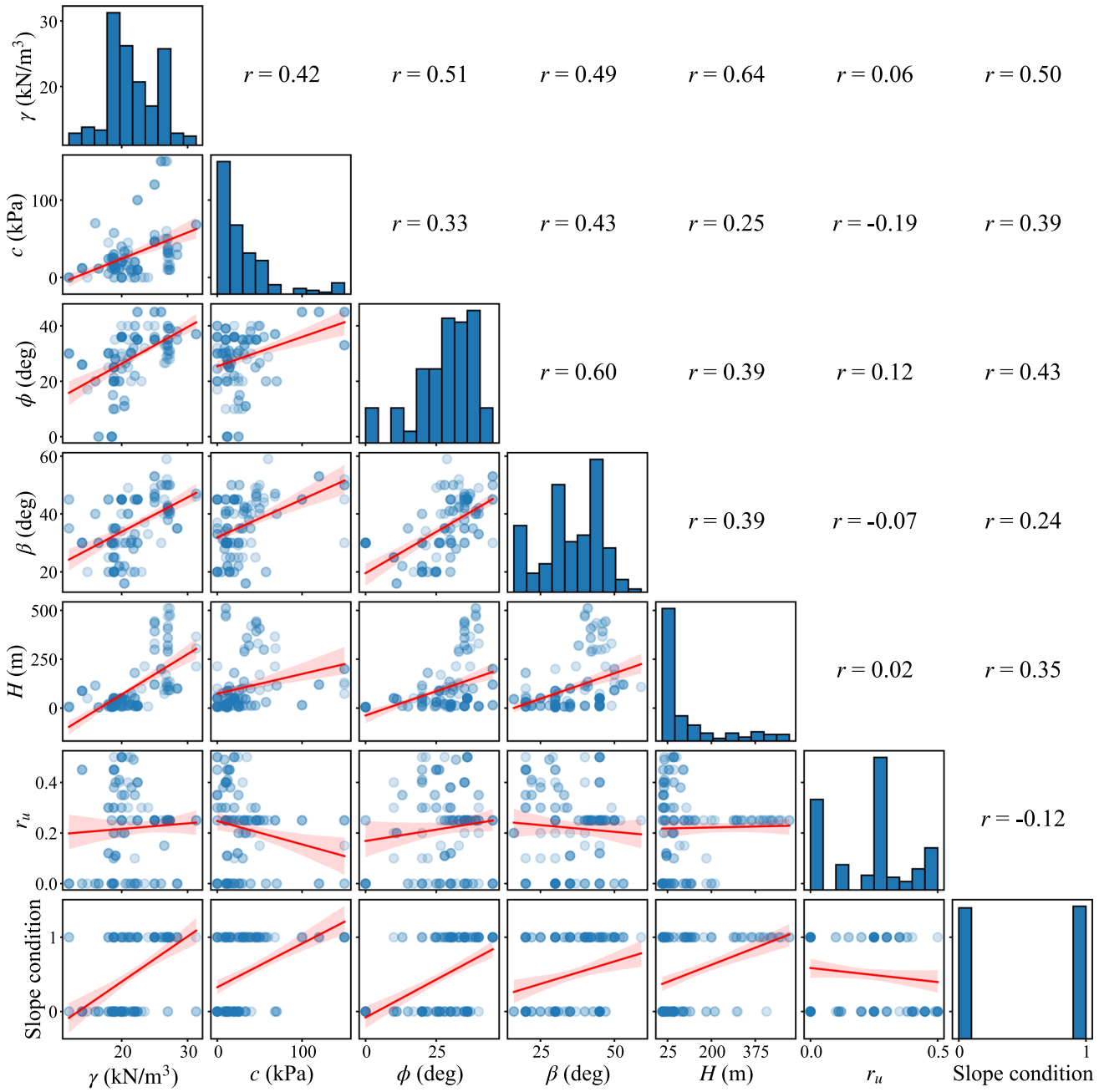


Fig. 4 Pairwise relationships between variables in the dataset

that γ is usually considered a destabilizing factor for slope stability based on some equilibrium analysis (e.g., [31]), assuming all of the parameters are independent. However, due to correlations between γ and other parameters in nature, a slope with higher soil unit weight may be more stable, which has been noted in studies based on slope stability case histories (e.g., [32]). For example, a higher γ typically indicates a denser arrangement of soil particles and/or a larger proportion of coarse-grained particles, which could contribute to higher soil strength and a more stable slope. Thus, this study considers γ as a stabilizing factor to represent this aspect of geotechnical domain knowledge that is not included in typical physics-based models for slope stability analysis. It should be noted that the method proposed in this study is designed to apply monotonicity constraints to any chosen subset of input features. In practice, it is recommended to use expert domain knowledge to identify and select the features with the highest degree of confidence; by enforcing monotonicity constraints on these specific features, model performance can be optimized, thereby enhancing accuracy

and ensuring consistency with established domain knowledge.

3.2 Proposed formulations for incorporating monotonicity constraints

In practice, monotonicity violations are challenging to inspect across the entire input space unless the model is constructed to be monotonic [27]. For the present study, the monotonicity inconsistency (MI) of each feature was examined through an augmented dataset where inputs were conditioned to be monotonic with respect to the corresponding feature. For the input dataset $x = [x_\alpha, x_{\neg\alpha}]$, the augmented dataset $\hat{x}$ for verifying monotonicity on feature α was constructed as $\hat{x} = [\hat{x}_\alpha, \hat{x}_{\neg\alpha}]$ such that $\hat{x}_\alpha$ contains all the elements from x_α sorted in ascending order (i.e., $\hat{x}_\alpha[i] < \hat{x}_\alpha[i+1]$) and $\hat{x}_{\neg\alpha} = \overline{x}_{\neg\alpha}$. Then the difference in model predictions can be computed for any adjacent sample pairs as:

$$\Delta f(\hat{x})[i] = f(\hat{x}[i+1]) - f(\hat{x}[i]) \tag{1}$$

the MI for feature α can then be calculated as the average of all the violations of monotonicity using the equation below:

$$\mathrm{MI}_\alpha = \begin{cases} \dfrac{1}{n-1}\sum_{i=1}^{n-1} \max(0, -\Delta f(\hat{x})[i]), & \text{for increasing monotonicity} \\ \dfrac{1}{n-1}\sum_{i=1}^{n-1} \max(0, \Delta f(\hat{x})[i]), & \text{for decreasing monotonicity} \end{cases} \tag{2}$$

where n is the number of samples. Based on Eq. (2), the monotonicity inconsistency for all the input features can be calculated as:

$$\mathrm{MI} = \sum_{\alpha=1}^{N} \mathrm{MI}_\alpha \tag{3}$$

where N is the total number of features. It should be noted that the monotonicity inconsistency evaluated by Eqs. (1–3) can be applied to a broad range of models, as monotonicity was evaluated based on an augmented dataset that was conditioned to be monotonic. Detailed implementation of MI can be found in Algorithm 1.

Algorithm 1 Monotonicity inconsistency inspected by data augmentation

Input:
Training data: $x = [x_\alpha, x_{\neg\alpha}]$, where x_α is the corresponding sub-vector for feature α with n elements, $x_{\neg\alpha}$ is its complement
Output:
Monotonicity inconsistency (MI)
1: **for** $\alpha = 1, 2, \ldots, N$ **do**
2: Construct augmented data $\hat{x} = [\hat{x}_\alpha, \hat{x}_{\neg\alpha}]$, where $\hat{x}_{\neg\alpha} = \overline{x}_{\neg\alpha}$ is the mean of the feature subset $x_{\neg\alpha}$
3: Sort $\hat{x}_\alpha$ in ascending order such that $\hat{x}_\alpha[i+1] > \hat{x}_\alpha[i]$
4: Compute forward difference $\Delta f(\hat{x})[i] = f(\hat{x}[i+1]) - f(\hat{x}[i])$
5: **if** enforcing increasing monotonicity for α **then**
6: Inspect increasing monotonicity: $\mathrm{MI}_\alpha = \frac{1}{n-1}\sum_{i=1}^{n-1} \max(0, -\Delta f(\hat{x})[i])$
7: **else** [enforcing decreasing monotonicity for α]
8: Inspect decreasing monotonicity: $\mathrm{MI}_\alpha = \frac{1}{n-1}\sum_{i=1}^{n-1} \max(0, \Delta f(\hat{x})[i])$
9: **end for**
10: **return** output $\mathrm{MI} = \sum_{\alpha=1}^{N} \mathrm{MI}_\alpha$

Conventional ML algorithms such as XGBoost [6] and gradient boost machine (GBM) [35] that enforce monotonicity at the algorithm level or ANNs that enforce monotonicity through specifically designed architecture are often restrictive and complex [27]. Evidence has shown that these methods typically do not generalize well (e.g., [3, 23]). The present study enforced monotonicity through data-based regularizations by modifying the loss function used in the training process. This approach is straightforward and can be easily applied to off-the-shelf models. A generic form of the loss function L with monotonicity constraint can be expressed as:

$$L = L_{data} + \lambda_r R + \lambda_{mono} L_{mono} \tag{4}$$

where L_{data} measures the supervised error between the model prediction and the ground truth, R is the model complexity loss, and L_{mono} is the monotonicity loss which measures the consistency of model predictions. Parameters λ_r, and λ_{mono} are hyper-parameters that determine the weight of each term in the loss function. Note that the first two terms on the right-hand side of Eq. (4) are standard losses for ML models. This study applied the proposed methods to slope stability predictions, which can be considered a binary classification problem, and the following binary cross-entropy loss is typically used:

$$L_{data} = \frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}) \tag{5}$$

The monotonicity of model predictions with respect to input features can be enforced by evaluating the point-wise gradients [17], which are partial derivatives computed at input data points. The monotonicity loss $L_{mono,\alpha}$ for feature α can be calculated as the summation of all the violations of monotonicity using the equation below:

$$L_{mono,\alpha} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \max(0, -\frac{\partial f(x[i])}{\partial x_\alpha[i]}), & \text{for increasing monotonicity} \\ \frac{1}{n}\sum_{i=1}^{n} \max(0, \frac{\partial f(x[i])}{\partial x_\alpha[i]}), & \text{for decreasing monotonicity} \end{cases} \tag{6}$$

Based on Eq. (6), the monotonicity loss for all the input features can be calculated as:

$$L_{mono} = \sum_{\alpha=1}^{N} L_{mono,\alpha} \tag{7}$$

It should be noted that training data distributions may affect the performance of the proposed monotonicity constraints if L_{mono} is only evaluated at training sample locations, as the effect of L_{mono} is localized around data points where monotonicity was measured. As the calculation of L_{mono} does not require labels, it can be used to evaluate monotonicity at any arbitrary location in the input space. In the present study, data augmentation was used to generate simulated data that covers the input space based on the uniform distribution (i.e., $\hat{x} \sim Uni(x)$) to improve the robustness of the proposed monotonicity constraints; as such, the proposed method pays equal attention and enforces monotonicity uniformly across the entire input space. Detailed implementation of L_{mono} can be found in Algorithm 2. The present study implemented the proposed framework for incorporating monotonicity constraints in the ANN model, which offers flexibility in designing and optimizing custom loss functions with the backpropagation algorithm [38].

Algorithm 2 Monotonicity loss using point-wise gradients

Input:

Training data: $x = [x_\alpha, x_{\neg\alpha}]$, where x_α is the corresponding sub-vector for feature α with n elements, $x_{\neg\alpha}$ is its complement

Output:

Monotonicity loss (L_{mono})

if data augmentation is **true**

  Construct augmented data and $\hat{x} \sim Uni(x)$

else

  No data augmentation and $\hat{x} = x$

for $\alpha = 1, 2, \ldots, N$ **do**

  if enforcing increasing monotonicity for α **then**

    Inspect increasing monotonicity $L_{mono,\alpha} = \frac{1}{n}\sum_{i=1}^{n} \max(0, -\frac{\partial f(\hat{x}[i])}{\partial \hat{x}_\alpha[i]})$

  else [enforcing decreasing monotonicity for α]

    Inspect decreasing monotonicity $L_{mono,\alpha} = \frac{1}{n}\sum_{i=1}^{n} \max(0, \frac{\partial f(\hat{x}[i])}{\partial \hat{x}_\alpha[i]})$

end for

return output $L_{mono} = \sum_{\alpha=1}^{N} L_{mono,\alpha}$

3.3 Description of ML models

ANN is a type of ML method and the heart of deep learning algorithms; its structure is inspired by the biological neural networks that constitute animal brains. A typical ANN comprises node layers, such as an input layer, multiple hidden layers, and an output layer. Each node (neuron) in ANN layers contains an associated weight and specific threshold. When activated, it passes processed data to the next layer. ANN develops functional relationships between inputs and outputs by adjusting connections between layers and node weights during the training process. The weights of hidden layers in ANN models are usually initialized based on random values of certain statistical distributions [50]. However, input data with limited observations is often insufficient to steer the model toward an acceptable performance. Similar to the concept of transfer learning commonly used in computer vision [48], one can use domain knowledge to inform the initialization of model parameters. This way, the pre-trained model learns feature representations for the desired task and can be fine-tuned using actual observations. For example, Read et al. [39] and Jia et al. [19] used pre-trained models trained on simulated data from physics-based models to enhance the accuracy and generalizability of their deep learning (DL) models for lake temperature prediction. Ma et al. [33] transferred weights trained on a dense USA dataset to data-sparse regions like Chile and China. In this study, random samples of the six influencing factors were first generated based on uniform distributions with the same range of values as those shown in Table 1, and values of the factor of safety (*FS*) for these simulated samples were calculated based on the stability chart for uniform slopes by Michalowski [31]. Figure 5 presents the data augmentation procedure used in this study. ANN models were trained to predict *FS* using these simulated samples during model pre-training (see Fig. 7). Subsequently, for the fine-tuning of these models, the output layer for the ANN models was modified by adding a sigmoid activation function for the classification task, and the pre-trained weights were used to initialize training for the ANN models to predict slope stability conditions using case histories. Figure 6 presents the workflow for developing the ANN models in this study. The same model structure and model validation procedure were used to evaluate the model performance for preparing the pre-trained model. In addition to ANN, the present study considered four commonly used conventional ML models for slope stability classifications, including logistic regression (LR), SVM, random forest (RF), and GBM. These models have been successfully used in various

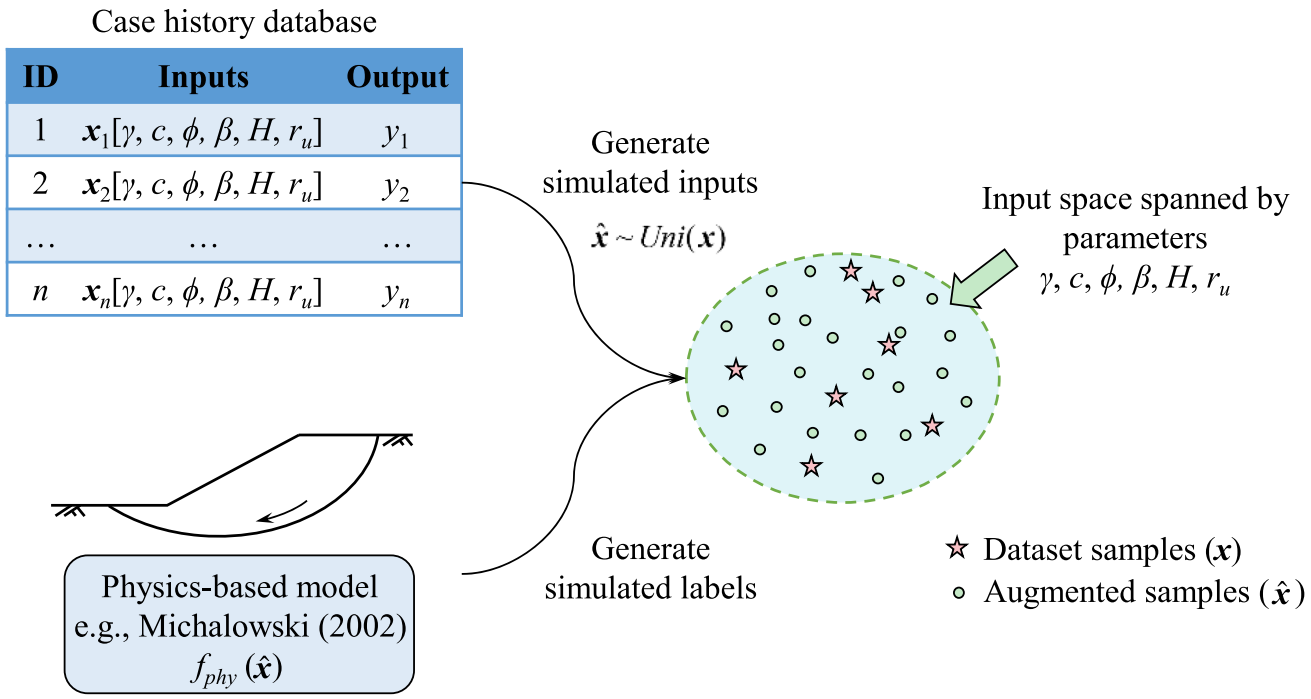
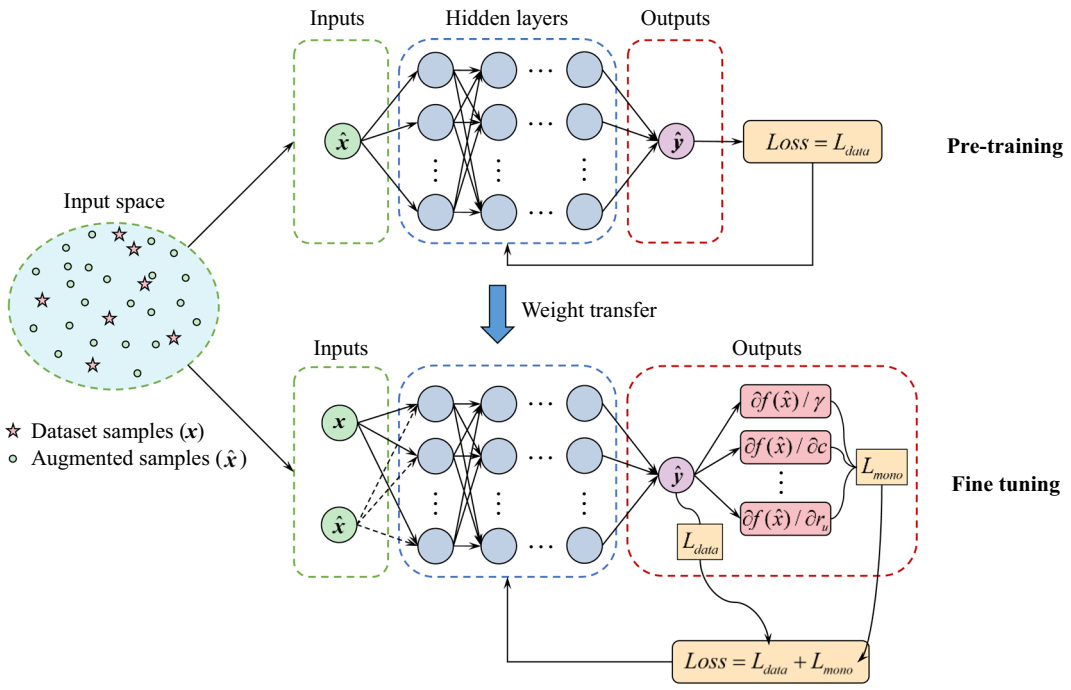


Fig. 5 Illustration of data augmentation procedure



09

Fig. 6 Schematic illustration of model training procedure, including both data augmentation and model pre-training

engineering applications, including slope stability predictions (e.g., [53]). A detailed introduction to these models can be found in Kuhn and Johnson [21].

3.4 Evaluations metrics

The performance of classification models is typically evaluated based on the confusion matrix and receiver operating characteristic (ROC) curve. The confusion matrix reports the four possible outcomes of model

predictions: (1) true positive (TP), which represents the number of correctly predicted positive samples; (2) true negative (TN), which represents the number of correctly predicted negative samples; (3) false positive (FP), which represents the number of incorrectly predicted positive class; and (4) false negative (FN), which represents the number of incorrectly predicted negative class. Six performance indicators for classification tasks can be calculated using these four parameters: accuracy, precision, recall, F_1, true positive rate (TPR), and false positive rate (FPR).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \tag{8}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{11}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{13}$$

Note that the F_1 score provides an aggregate measure of the model performance score by combining precision and recall into a single metric. The ROC curve is a 2D plot of FPR vs. TPR for all classification thresholds. The area under the ROC curve (AUC) can be calculated based on the ROC curve, providing an aggregate measure of model performance. AUC is often used as a single-value evaluation for classification models as it measures the model's capability to distinguish two classes. A no-skill model (i.e., similar to random guessing) will have an AUC score of 0.5, whereas a perfect model will have an AUC score of 1.0. In addition to the evaluation metrics mentioned previously, the present study used MI as an additional score to measure the monotonicity of model predictions. In summary, six evaluation metrics were considered in the present study for a comprehensive model performance evaluation, including accuracy, precision, recall, F_1, AUC, and MI.

3.5 Dataset partition and model development

The model development framework in the present study involves two components: model selection and model performance evaluation. For model selection, each baseline model used in the present study contains hyperparameters; these hyperparameters need to be carefully tuned to ensure model performance. For model evaluations, the performance of ML models needs to be evaluated on new datasets to test their generalization performance. The k-fold cross-validation technique [41] was used for dataset partition in the present study for both model selection and model performance evaluation. In cross-validation, the dataset is divided into k folds and the ML model is trained using k-1 folds and validated using the remaining one fold. This process repeats k times to allow each fold to be served as a validation fold, and the final model performance is the average model performance for each validation fold. For the present study, $k = 5$ was used (i.e., five-fold cross-validation) for the consideration of computation efficiency and bias. The cross-validation procedure measures the generalization performance of a model fitted to all available data [49]. In the present study, the hyperparameters of each model were tuned to minimize generalization performance on each cross-validation. The generalization performance evaluated for those optimal hyperparameter values can then be used to obtain the best model. This approach is also known as flat cross-validation and has been widely used by researchers to select both the algorithms and the hyperparameters (e.g., [13, 22]); this approach is computationally inexpensive and performs similarly to some more comprehensive cross-validation techniques, such as nested cross-validation [49]. In the present study, PyTorch [38] was used to develop the ANN model, and the rest of the models were developed using Scikit-learn [37]. In addition, the cross-validation procedure for all the models was repeated five times (resulting in 25 candidate models in total) to account for variations in random seeds and ensure the repeatability of the results.

4 Results and discussion

4.1 Performance of ML models

The AUC score measures the model's capability in discriminating positive (stable) and negative (failure) classes; its value is not affected by the classification threshold. In the present study, the hyperparameters for each ML model were tuned to achieve the highest validation AUC score based on the five-fold cross-validation procedure with five repetitions; the validation performance of 25 candidate models was reported in this study to represent model performance. The optimum hyperparameters for LR, SVM, RF, GBM, and ANN can be found in Table 2.

The output of ML models is the probability of possible outcomes for each sample. A default threshold of 0.5 was used to split the model predictions into two categories for binary classification; subsequently, the classification scores were calculated according to these predicted binary outcomes. Table 3 presents the classification performance on the validation dataset based on the five-fold cross-

validation procedure with five repetitions. As shown in Table 3, the trained ML models achieved an average AUC score of 0.936, an accuracy of 0.885, and a F_1 score of 0.883. These values are relatively high and suggest that the trained ML models can effectively differentiate slope stability conditions (i.e., stable vs. failure). In addition, it can be noted from Table 3 that the performance varies among these ML models. For example, the LR model exhibited the worst performance due to its simplicity, whereas ML models with ensemble techniques such as RF and GBM achieved the highest classification performance with average AUC scores above 0.96. The SVM and ANN models achieved intermediate performance with average AUC scores of 0.925 and 0.947, respectively. In addition, Table 3 also shows that the five ML models had an average MI score of 0.025, indicating monotonicity inconsistency in the model predictions. Based on the results in Table 3, it can be concluded that ML models can achieve high levels of classification performance; however, the trained models may produce predictions that violate monotonicity relationships, which can be attributed to the randomness in developing ML models and in the training dataset.

4.2 Effect of model pre-training

In this study, a uniform distribution is employed to generate 1000 simulated samples that span the entire input space formed by the range of values as those shown in Table 1 for the pre-training of the ANN model. This approach ensures an equitable representation of all regions within the input space, thereby enabling the model to allocate equal attention and consideration to all parts of the input space during the learning process. The *FS* values for these simulated samples were then calculated based on the stability chart for uniform slopes by Michalowski [31]; subsequently, ANN models were trained to predict *FS* using these simulated samples. Figure 7 presents the model performance in predicting the simulated dataset based on the cross-validation procedure. As shown in Fig. 7, the trained ANN model achieved an average validation R^2 of 0.92, suggesting an excellent performance. The trained model from the third fold was randomly chosen as a pre-trained model for developing classification models to predict slope stability based on the case histories using the model development procedure described in Sect. 3.3; the weights in the pre-trained model were fine-tuned by learning from case histories. Figure 8 compares the performance between ANN models with and without pre-training vs. training epoch based on the cross-validation procedure. In Fig. 8, the solid line represents the mean value, and the shaded area represents standard deviations. As shown in Fig. 8a, compared with ANN models without pre-training, ANN models with pre-training had higher AUC scores with faster convergence and a smaller range of variations during the training process. This performance improvement can be attributed to the fact that the pre-trained model contains feature representations for the desired task (i.e., pre-trained to predict the stability chart as shown in Fig. 7); thus, the subsequent training using actual

Table 2 Summary of hyperparameters tunned based on the cross-validation procedure

Model	Parameters
LR	Slover: LBFGS; penalty: L2; C: 1.0
SVM	Kernel: RBF; C:1000; gamma: Scale
RF	Criterion: Gini; n_estimators: 100; minimum_samples_split: 2; minimum_samples_leaf:1
GBM	Loss: Deviance; learning rate: 1.0; n_estimators: 100; minimum_samples_split: 2; minimum_samples_leaf: 1, maximum_depth = 5
ANN	Number of hidden layers: 3; neurons per layer: 16; activation function: tanh; optimizer: Adam; learning rate: 0.005; epoch:114

Table 3 Summary of model performance based on the cross-validation procedure

Model	Accuracy	Precision	Recall	F_1	AUC	MI
LR	0.795	0.815	0.774	0.788	0.856	0.001
SVM	0.892	0.911	0.879	0.891	0.925	0.028
RF	0.909	0.923	0.899	0.908	0.965	0.020
GBM	0.915	0.932	0.901	0.914	0.964	0.052
ANN	0.882	0.888	0.887	0.883	0.947	0.025
Avg	0.879	0.894	0.868	0.877	0.931	0.025

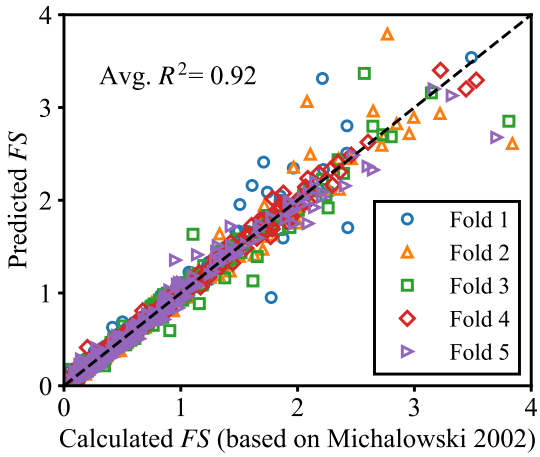


Fig. 7 ANN model performance in predicting simulated dataset based on cross-validation procedure

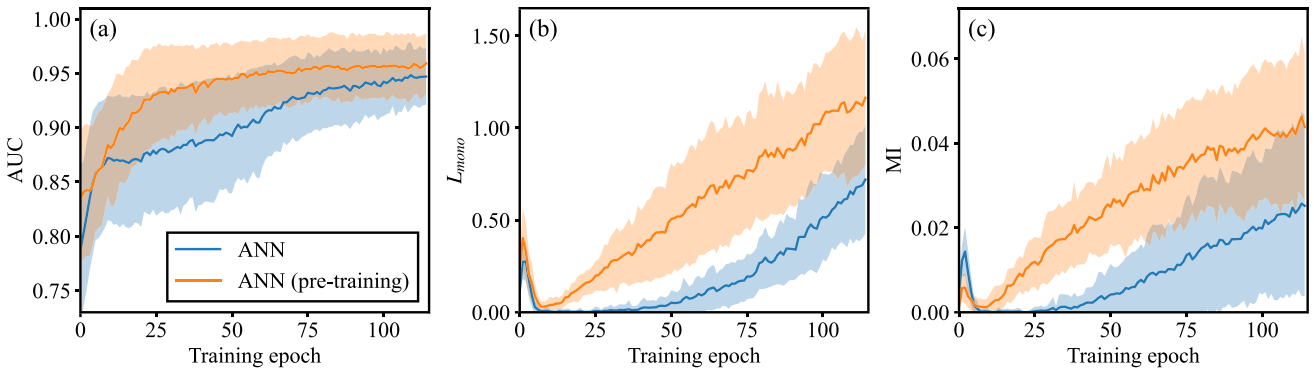


Fig. 8 ANN model performance with and without pre-training vs. training epoch: **a** AUC; **b** L_{mono}; and **c** MI

Table 4 Summary of ANN model performance with and without pre-training

Model	Accuracy	Precision	Recall	F_1	AUC	MI
ANN	0.882	0.888	0.887	0.883	0.947	0.025
ANN (pre-training)	0.916	0.937	0.899	0.915	0.959	0.044
Relative diff	+ 3.85%	+ 5.60%	+ 1.39%	+ 3.62%	+ 1.27%	+ 73.75%

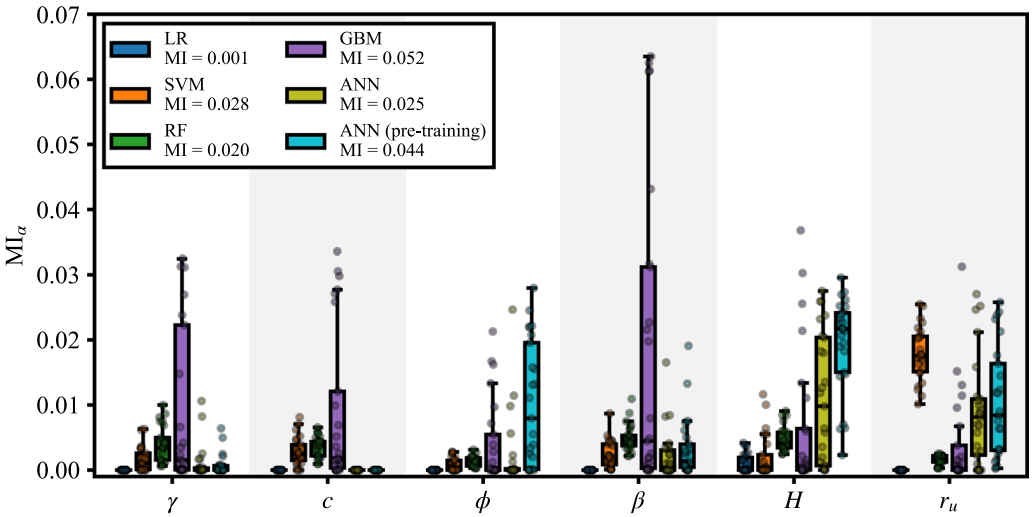


Fig. 9 Summary of MI_α for trained ML models based on cross-validation procedure

observations can achieve global minimum easier than ANN models initialized using randomly generated weights. Figures. 8b and c present monotonicity inconsistency measured by L_{mono} and MI, respectively; these two figures show that monotonicity inconsistency generally increases with training epochs, and ANN models with pre-training exhibited higher monotonicity inconsistency. This can be attributed to the fact that the pre-trained model was well-trained by a large number of simulated data; thus, the subsequent fine-tuning tended to adjust excessively to the data from case histories and learn complex field situations. Table 4 summarizes the performance of ANN models with and without pre-training. As shown in Table 4, ANN models with pre-training performed better in all the classification metrics; however, they have significantly higher monotonicity inconsistency than ANN models without pre-training. Results in Fig. 8 and Table 4 suggest that ANN models may benefit from pre-trained weights and gain improvements in certain data science evaluation criteria; however, one should pay attention when adopting pre-

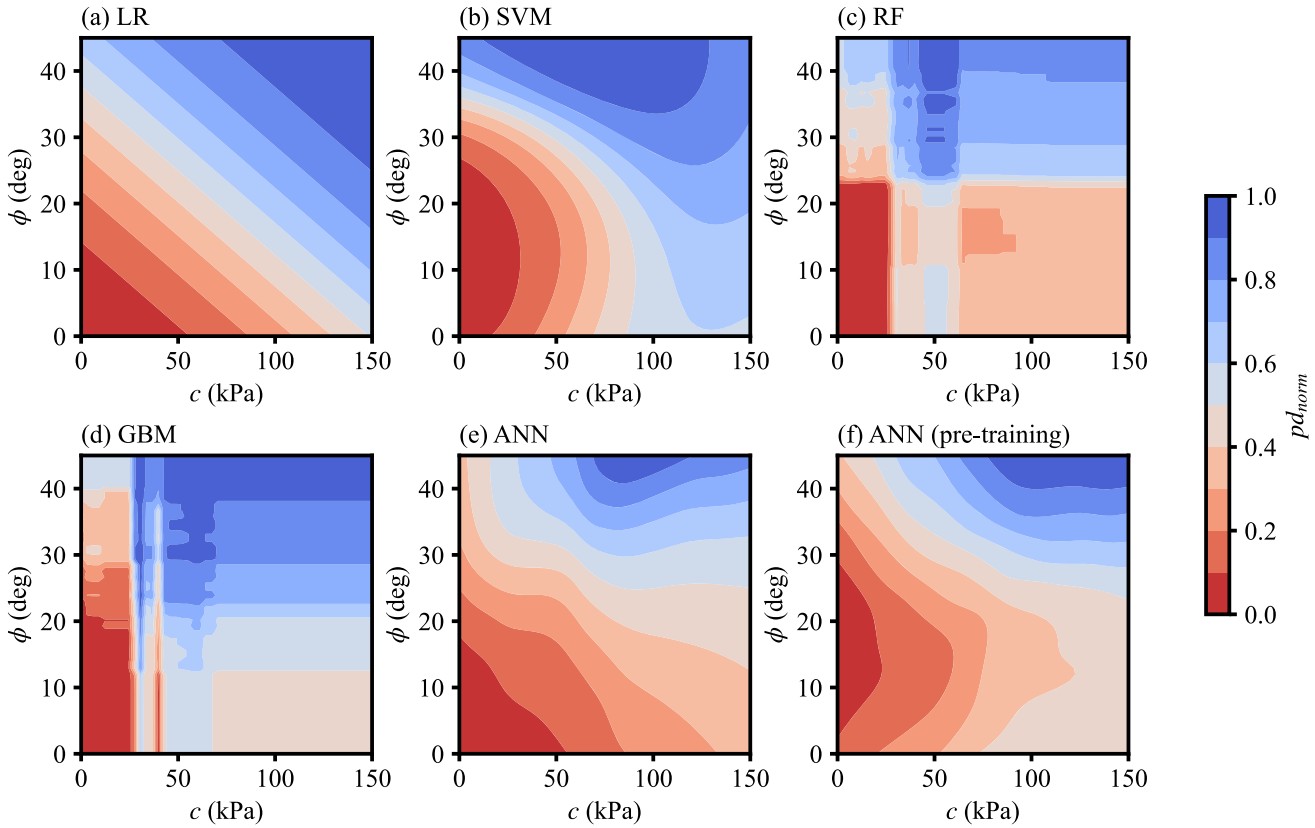


Fig. 10 Partial dependence plots based on trained ML models showing the effect of c and ϕ for predicting slope stability conditions: **a** LR; **b** SVM; **c** RF; **d** GBM; **e** ANN; and **f** ANN (pre-training)

trained weights and examine the model's physics consistency using domain knowledge (Table 4).

4.3 Monotonicity evaluation for ML models

The monotonicity of model predictions with respect to each input feature was evaluated using MI_α based on an augmented dataset that was conditioned to be monotonic. Figure 9 presents boxplots showing monotonicity inconsistency measured by MI_α for all the trained ML models based on the cross-validation procedure with five repetitions. As shown in Fig. 9, positive values of MI_α can be generally observed for all the trained ML models, indicating that model predictions disobey the pre-set monotonicity criteria. It can be noted that monotonicity inconsistencies are mainly observed in features β, H, and r_u. Among all the ML algorithms, the GBM model exhibited the worst monotonicity consistency as measured by MI_α. In contrast, the LR model exhibited the best monotonicity consistency, which can be expected as the LR model is unconditionally monotonic at the algorithm level. However, positive values of MI_α can still be observed for LR models due to sampling randomness in the training datasets.

In addition to MI_α, the partial dependence (pd) plots [11] were used to evaluate the relationship between the model predictions and each input feature, marginalizing over the values of all the other complement features. Computationally, the values of pd for feature α at x_α can be calculated as the average in model predictions as:

$$pd(x_\alpha) = \frac{1}{n}\sum_{i=1}^{n} f(x_\alpha, x_{\neg\alpha}^i) \tag{14}$$

In the present study, the pd value for each x_α was calculated using 25 randomly generated samples of $x_{\neg\alpha}$ within the input space. The partial dependence plots can then be obtained by calculating pd at multiple values of x_α. In addition, as each ML model may have a different range of response with respect to input features, the calculated values of pd were normalized to the range between zero and one using the equation below to facilitate comparison:

$$pd_{norm} = \frac{pd - \min(pd)}{\max(pd) - \min(pd)} \tag{15}$$

Figure 10 presents the partial dependence plot based on trained ML models showing the effect of c and ϕ for predicting slope stability conditions, each subplot in Fig. 10 is generated using one trained model randomly

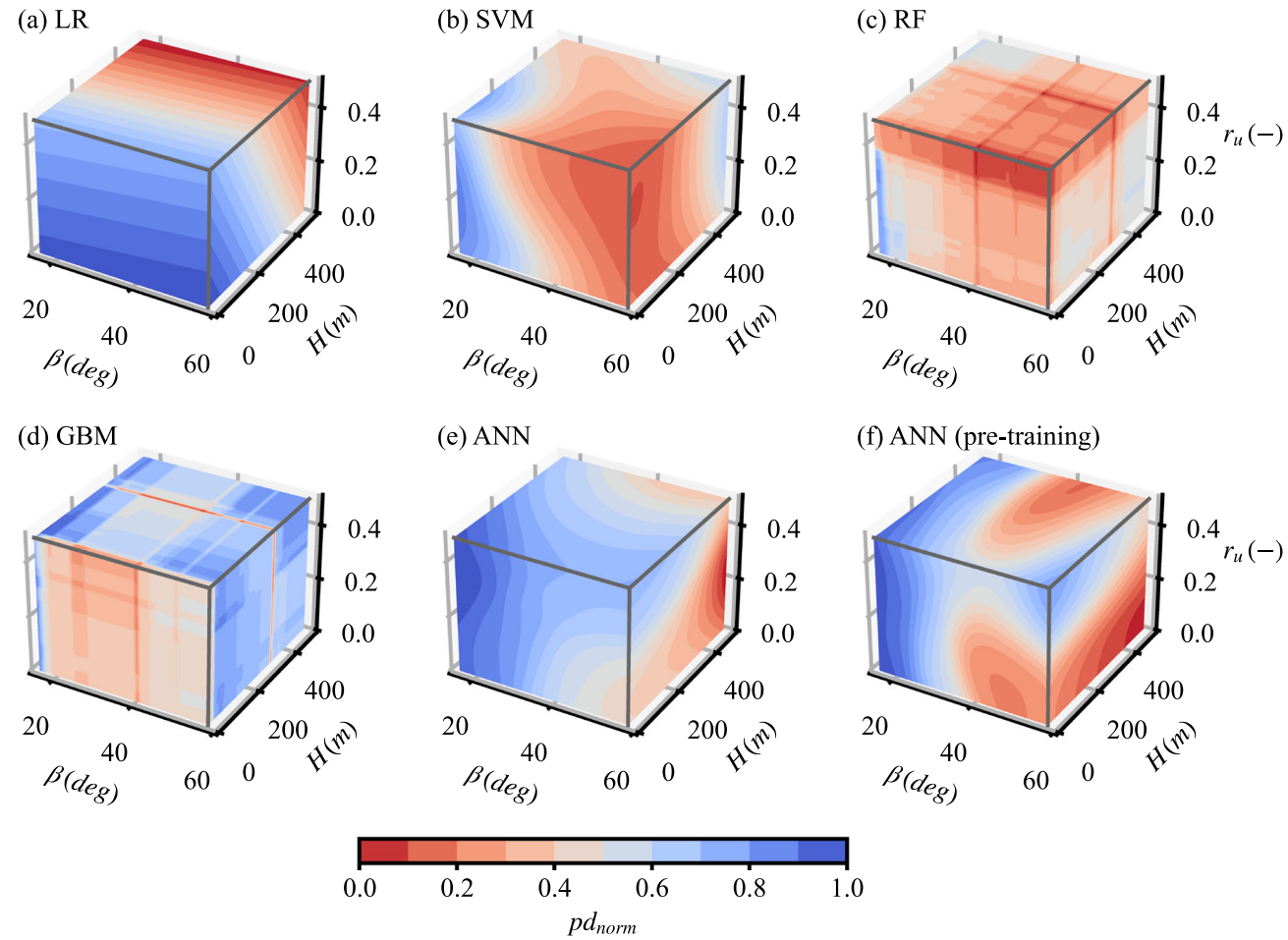


Fig. 11 Partial dependence plots based on trained ML models showing the effect of β, H, and r_u for predicting slope stability conditions: **a** LR; **b** SVM; **c** RF; **d** GBM; **e** ANN; and **f** ANN (pre-training)

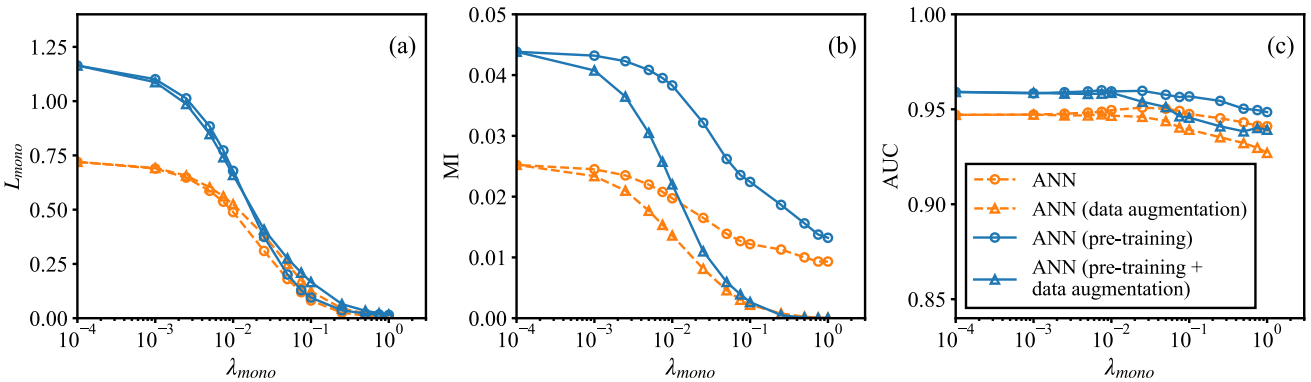


Fig. 12 Effect of λ_{mono} on the performance of ANN models based on cross-validation procedure: **a** L_{mono}; **b** MI; and **c** AUC

chosen from the candidate models trained using the cross-validation procedure. A higher value of pd_{norm} indicates that the model predicts the slope to be more stable, and a lower pd_{norm} value indicates that the model predicts the slope to be less stable. As shown in Fig. 10, all ML models generally predicted a higher value of pd_{norm} with an increase in soil strength parameters c and ϕ; however, monotonicity inconsistencies can be observed for all ML models except the LR model, as indicated by the color contours. Figure 11. presents the corresponding partial dependence plot based on trained ML models showing the effect of β, H, and r_u for predicting slope stability

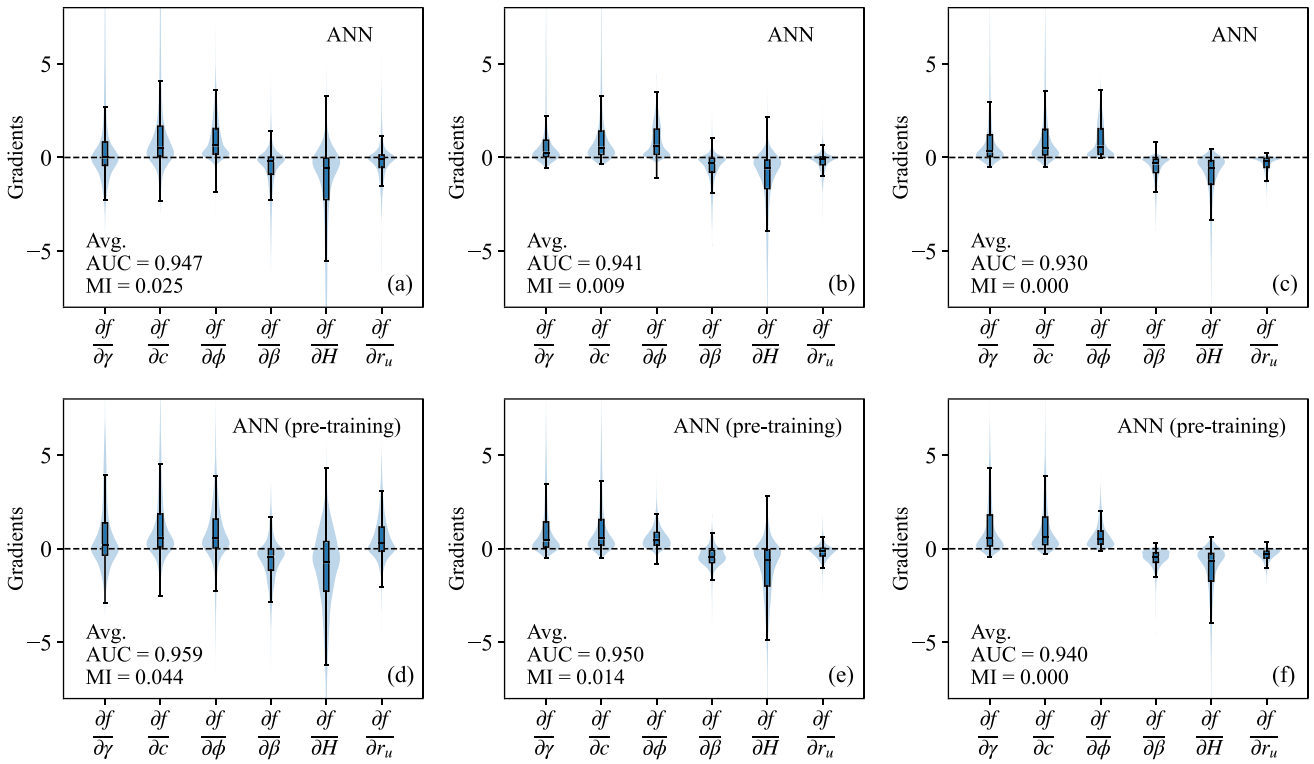


Fig. 13 Comparison of gradients between ANN models with and without monotonicity constraint ($\lambda_{mono} = 0.75$): **a** ANN models without constraint; **b** ANN models with L_{mono}; **c** ANN models with L_{mono} and data augmentation; **d** ANN (pre-training) models without constraint; **e** ANN (pre-training) models with L_{mono}; and **f** ANN (pre-training) models with L_{mono} and data augmentation

conditions. As shown in Fig. 11, all trained ML models have difficulties producing the correct monotonicity relationship, even the LR model predicts the slope stability condition increases with increasing β value. Moreover, the model response with respect to β, H, and r_u for all of the other ML models except the LR model is more complex and less interpretable, which is also evident from box plots in Fig. 9 for features β, H, and r_u. Based on the results shown in Figs. 9, 10 and 11, it can be concluded that ML models trained purely on data may not reflect the correct monotonicity relationships and produce results contradictory to geotechnical domain knowledge, which can be a particular concern for future slope stability predictions using these trained ML models. Figures 9, 10 and 11 also suggest that model evaluation criteria in data science may be inadequate and judgment of model performance using domain knowledge is needed.

4.4 Effect of monotonicity constraints

In the present study, monotonicity was enforced through the L_{mono} term in the loss function during the training process. Values of L_{mono} were computed in two ways: (1) at training data points or (2) at augmented data points randomly sampled from the input space based on uniform distribution. Figure 12 presents the effect of λ_{mono} on model performance by averaging results from all trained models from the cross-validation procedure with five repetitions. Note that $\lambda_{mono} = 0$ corresponds to models without constraint and a greater λ_{mono} value corresponds to a stronger regularization from the monotonicity loss term (i.e., L_{mono}). Figure 12 shows that in all cases as λ_{mono} increases, monotonicity inconsistency measured by L_{mono} and MI significantly decreases and a slight drop in classification performance (i.e., AUC score) can be observed. This indicates that the proposed monotonicity constraint can steer the model prediction to comply with the pre-set monotonicity relationship. The decrease in classification performance is expected as the L_{mono} term reduces the model's flexibility, and the dataset may not reflect the monotonicity relationship due to randomness. In addition, by comparing Figs. 12a and b, the effect of L_{mono} with data augmentations can not be easily observed by L_{mono} computed on validation datasets; however, their effect on enforcing monotonicity over the input space can be observed through MI, which also used data augmentation. The monotonicity enforced through L_{mono} with data augmentations can reduce MI values much more rapidly to zero than those without data augmentations. This indicates that data augmentations can produce a more robust

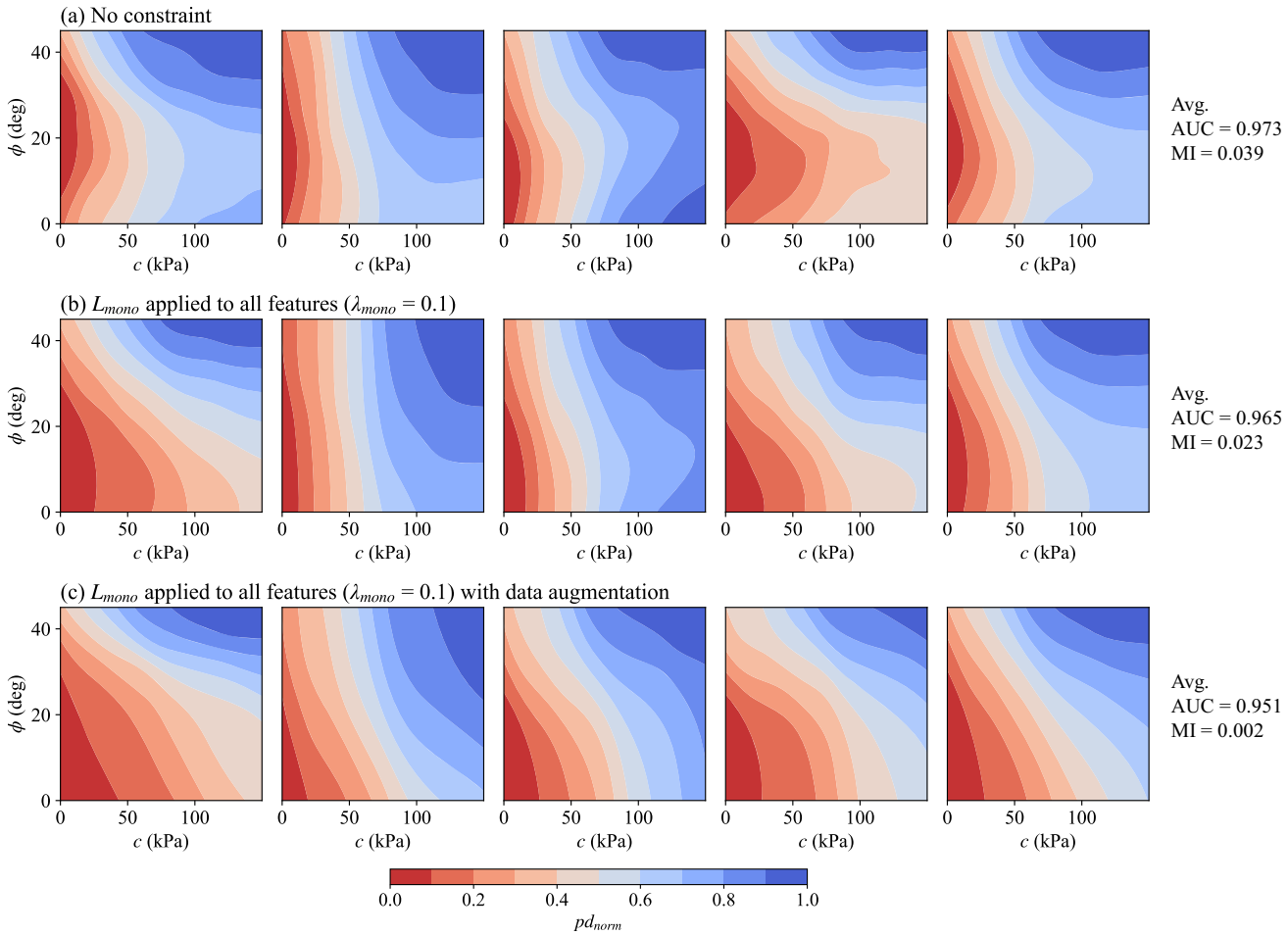


Fig. 14 Partial dependence plots (pd_{norm}) showing the effect of c and ϕ for predicting slope stability conditions for ANN models (with pre-training): **a** no constraint; **b** L_{mono} applied to all features; and **c** L_{mono} applied to all features with data augmentation

regularization of monotonicity over a much larger region in the input space. Moreover, it can be noted that enforcing monotonicity with data augmentation resulted in a more significant drop in classification performance at the same λ_{mono} value than those without data augmentation (see Fig. 12c). However, these models still perform adequately well with much more generalizable responses in terms of monotonicity consistency that aligns with domain knowledge.

Figure 13 presents the boxplots with kernel density estimates showing gradients of ANN model outputs with respect to each input feature; these gradients are computed using all trained models from the cross-validation procedure with five repetitions. For each model, a simulated dataset of 250 samples was uniformly sampled within the input space, and the gradients for each model were computed using this simulated dataset (i.e., each boxplot contains 6,250 data points). As shown in Figs. 13a and d, gradients for ANN models without constraints exhibited non-monotonic behavior as values of gradients for each feature contain different signs. It can be noted in Figs. 13b and e that monotonicity consistency can be generally improved by enforcing monotonicity through L_{mono} at training data points. However, this approach does not perform well for features that exhibit strong non-monotonicity (e.g.,, H). As shown in Figs. 13c and f, enforcing monotonicity through L_{mono} with data augmentation can eliminate the majority of monotonicity inconsistencies over the entire input space.

Figure 14 presents the partial dependence plot showing the effect of c and ϕ for predicting slope stability for ANN models without monotonicity constraints, with L_{mono} applied to all features, and with L_{mono} applied to all features with data augmentation. Each row in Fig. 14 contains partial dependence plots based on trained models from one cross-validation. By comparing Figs. 14a through c, monotonicity consistency in model predictions is improved due to the L_{mono} term in the loss function, and this effect can be further improved by computing L_{mono} with data augmentation. Figure 15 presents the corresponding plots showing the effect of β, H, and r_u for predicting slope stability for ANN models; a similar trend can be observed.

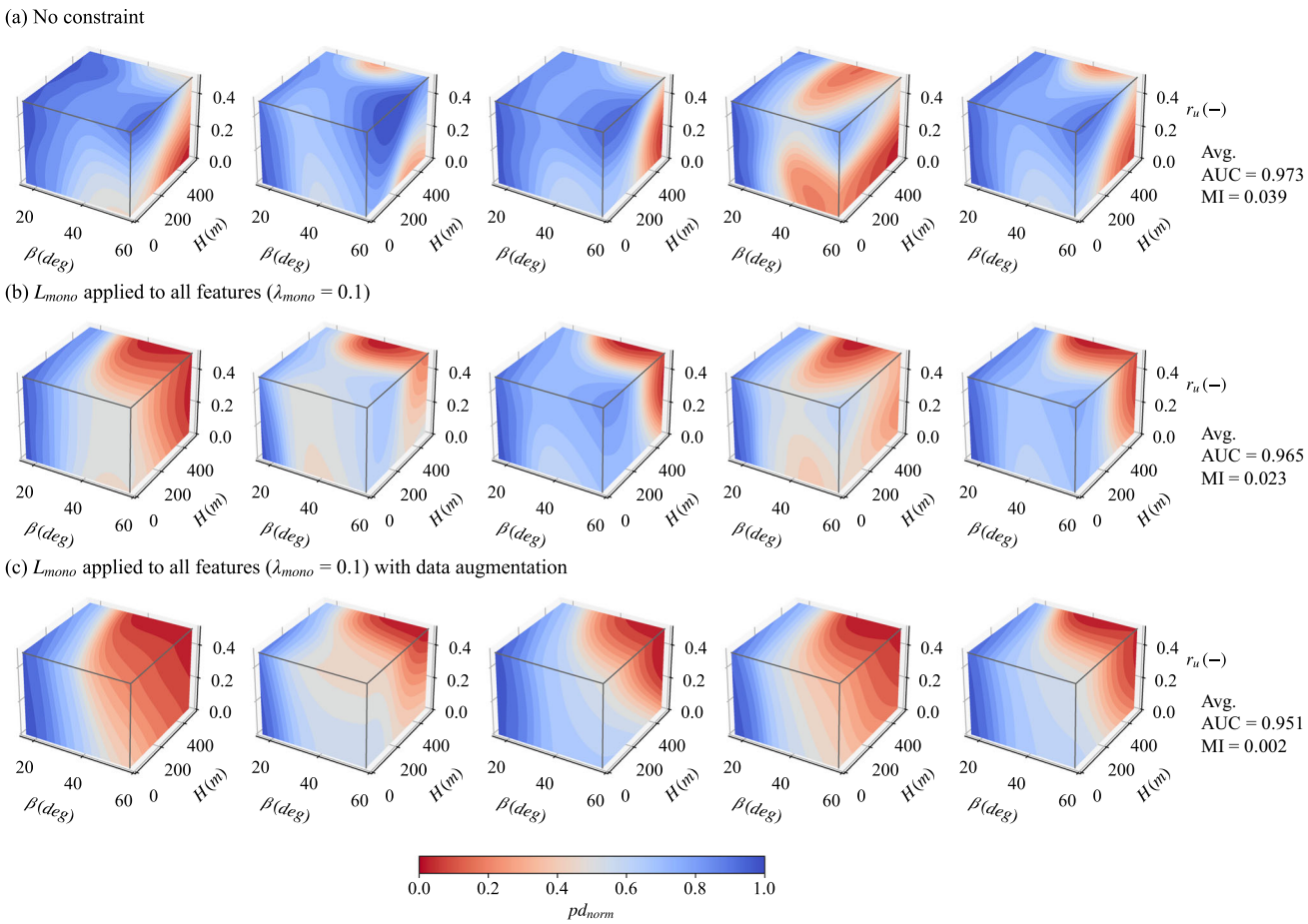


Fig. 15 Partial dependence plots (pd_{norm}) showing the effect of β, H, and r_u for predicting slope stability conditions for ANN models (with pre-training): **a** no constraint; **b** L_{mono} applied to all features; and **c** L_{mono} applied to all features with data augmentation

Based on Figs. 14 and 15, it can be concluded that the trained models can better capture monotonic relationships between input futures and slope stability conditions by enforcing monotonicity and producing predictions that are significantly more interpretable and align with our physical understanding of slope stability.

5 Discussion

Monotonicity constraints simplify the relationship between input and output variables, making it easier to predict how the model will behave when faced with new or unseen data (i.e., better extrapolation and generalization capability), especially when these relationships are based on known physical laws or established domain knowledge. In addition, monotonic relationships are often easier to interpret and understand than non-monotonic ones. When dealing with complex geotechnical problems, having an interpretable model can help engineers and decision-makers understand the factors influencing the predictions, gain insights into the underlying processes, and gain their confidence in the developed model. Therefore, monotonicity is essential in ML applications in geotechnical engineering as it helps address issues related to physical consistency, safety and reliability, interpretability, and regulatory compliance. Besides predicting slope stability, the proposed methods in this study can help facilitate the application of ML in other predictive tasks in geotechnical engineering, such as bearing capacity prediction, settlement prediction, and liquefaction potential assessment. It can also help ML applications in regional landslide susceptibility mapping.

This study demonstrated the value of enforcing monotonicity for ML models to produce predictions aligned with our geotechnical domain knowledge. However, the proposed method has several limitations that need to be addressed in future research. For example, the proposed method only applied a single weight term to control the degree of regularization for the monotonicity loss term. Ideally, different weights should be applied to each feature as they contribute differently to monotonicity

inconsistency, which can be formulated into an optimization problem. In addition, the proposed monotonicity constraint was enforced through the loss function, which can not guarantee unconditional monotonicity. The performance of the proposed monotonicity constraint should be further evaluated with different datasets. Moreover, as geotechnical engineering problems often exhibit complex behavior, values of additional constraints should be evaluated, such as interaction and convexity/concavity constraints.

6 Conclusions

With recent advances in data science, ML models have been adopted in many applications. However, the applicability of ML models for modeling scientific problems involving complex physics is still limited as ML models purely rely on features from data for model development. ML models trained with pure data, especially with limited data, can often produce counter-intuitive predictions, raising challenges in applying data science models in geotechnical engineering applications. This study presents methods for evaluating and enforcing the monotonicity of model predictions which help ML models produce predictions that align with domain knowledge. The slope stability predictions were used as an example problem to demonstrate the effectiveness of the proposed methods. The following conclusions can be drawn from the results of this study.

(1) Commonly used ML models without constraints can achieve high performance based on certain data science evaluation matrices. However, the trained models may produce predictions that violate geotechnical domain knowledge.
(2) Pre-trained weights based on simulated data can be beneficial in improving ML model performance; however, the model with pre-trained weights exhibited more significant monotonicity inconsistency.
(3) The proposed monotonicity constraint can effectively steer the model prediction to comply with preset monotonicity relationships. Moreover, the effectiveness of the proposed monotonicity constraint can be enhanced with data augmentation.

Acknowledgements This research was partially supported by Google AI Impacts Challenge Grant 1904-57775. The second authors' effort was also supported by the U.S. National Science Foundation under Award No. ICER-2022444. These supports are gratefully acknowledged. The authors would also like to acknowledge the assistance of Zhanzhao Li (Pennsylvania State University) for his insightful input during the planning and development of this work.

Data availability Some data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request. These data include the results used to generate all figures and tables.

References

1. Archer NP, Wang S (1993) Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. Decis Sci 24:60–75. https://doi.org/10.1111/j.1540-5915.1993.tb00462.x
2. Bishop AW (1955) The use of the slip circle in the stability analysis of slopes. Géotechnique 5:7–17. https://doi.org/10.1680/geot.1955.5.1.7
3. Bartley C, Liu W, Reynolds M (2019) Enhanced random forest algorithms for partially monotone ordinal classification. Proc Conf AAAI Artif Intell 33:3224–3231. https://doi.org/10.1609/aaai.v33i01.33013224
4. Bandai T, Ghezzehei TA (2021) Physics-informed neural networks with monotonicity constraints for Richardson-Richards equation: estimation of constitutive relationships and soil water flux density from volumetric water content measurements. Water Resour Res. https://doi.org/10.1029/2020wr027642
5. Chen CC, Li ST (2014) Credit rating with a monotonicity-constrained support vector machine model. Expert Syst Appl 41:7235–7247. https://doi.org/10.1016/j.eswa.2014.05.035
6. 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. ACM, New York, NY, USA
7. Canini K, Cotter A, Gupta MR, Fard MM, Pfeifer J (2016) Fast and flexible monotonic functions with ensembles of lattices. In: Proceedings of the 30th international conference on neural information processing systems 29:6835–6845. https://doi.org/10.5555/3157382.3157425
8. Doumpos M, Zopounidis C (2009) Monotonic support vector machines for credit risk rating. New Math Nat Comput 05:557–570. https://doi.org/10.1142/s1793005709001520
9. Das SK, Biswal RK, Sivakugan N, Das B (2011) Classification of slopes and prediction of factor of safety using differential evolution neural networks. Environ Earth Sci 64:201–210. https://doi.org/10.1007/s12665-010-0839-1
10. Daw A, Karpatne A, Watkins W et al (2017) Physics-guided neural networks (PGNN): an application in lake temperature modeling. arXiv [cs.LG]. https://doi.org/10.48550/ARXIV.1710.11431
11. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232. https://doi.org/10.1214/aos/1013203451
12. Fischer EM, Knutti R (2015) Anthropogenic contribution to global occurrence of heavy-precipitation and high-temperature extremes. Nat Clim Chang 5:560–564. https://doi.org/10.1038/nclimate2617
13. Feurer M, Klein A, Eggensperger K et al (2019) Auto-sklearn: efficient and robust automated machine learning. In: Automated machine learning. Springer, Cham, pp 113–134
14. Griffiths DV, Lane PA (1999) Slope stability analysis by finite elements. Géotechnique 49:387–403. https://doi.org/10.1680/geot.1999.49.3.387
15. González S, Herrera F, García S (2015) Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. New Gener Comput 33:367–388. https://doi.org/10.1007/s00354-015-0402-4

16. Gordan B, Jahed Armaghani D, Hajihassani M, Monjezi M (2016) Prediction of seismic slope stability through combination of particle swarm optimization and neural network. Eng Comput 32:85–97. https://doi.org/10.1007/s00366-015-0400-7
17. Gupta A, Shukla N, Marla L et al (2019) How to incorporate monotonicity in deep networks while preserving flexibility? arXiv [cs.LG]. https://doi.org/10.48550/ARXIV.1909.10662
18. Hoang N-D, Pham A-D (2016) Hybrid artificial intelligence approach based on metaheuristic and machine learning for slope stability assessment: a multinational data analysis. Expert Syst Appl 46:60–68. https://doi.org/10.1016/j.eswa.2015.10.020
19. Jia X, Willard J, Karpatne A, Read JS, Zwart JA, Steinbach M, Kumar V (2021) Physics-guided machine learning for scientific discovery: an application in simulating lake temperature profiles. ACM IMS Trans Data Sci 2:1–26. https://doi.org/10.1145/3447814
20. Kotlowski W, Slowinski R (2013) On nonparametric ordinal classification with monotonicity constraints. IEEE Trans Knowl Data Eng 25:2576–2589. https://doi.org/10.1109/tkde.2012.204
21. Kuhn M, Johnson K (2013) Applied predictive modeling, 1st edn. Springer, New York
22. Kotthoff L, Thornton C, Hoos HH, Hutter F, Leyton-Brown K (2019) Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA. In: Automated machine learning. Springer, Cham, pp 81–95. https://doi.org/10.1007/978-3-030-05318-5_4
23. Kokel H, Odom P, Yang S, Natarajan S (2020) A unified framework for knowledge intensive gradient boosting: leveraging human experts for noisy sparse domains. Proc Conf AAAI Artif Intell 34:4460–4468. https://doi.org/10.1609/aaai.v34i04.5873
24. Li J, Wang F (2010) Study on the forecasting models of slope stability under data mining. In: Earth and Space 2010. American Society of Civil Engineers, Reston, VA. https://doi.org/10.1061/41096(366)77
25. Li S-T, Chen C-C (2015) A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. IEEE Trans Fuzzy Syst 23:1713–1727. https://doi.org/10.1109/tfuzz.2014.2374214
26. Lin Y, Zhou K, Li J (2018) Prediction of slope stability using four supervised learning methods. IEEE Access 6:31169–31179. https://doi.org/10.1109/access.2018.2843787
27. Liu X, Han X, Zhang N, Liu Q (2020) Certified monotonic neural networks. arXiv [cs.LG]. https://doi.org/10.48550/arXiv.2011.10219
28. Lin S, Zheng H, Han B, Li Y, Han C, Li W (2022) Comparative performance of eight ensemble learning approaches for the development of models of slope stability prediction. Acta Geotech 17:1477–1502. https://doi.org/10.1007/s11440-021-01440-1
29. Morgenstern NR, Price VE (1965) The analysis of the stability of general slip surfaces. Géotechnique 15:79–93. https://doi.org/10.1680/geot.1965.15.1.79
30. Michalowski RL (1995) Slope stability analysis: a kinematical approach. Géotechnique 45:283–293. https://doi.org/10.1680/geot.1995.45.2.283
31. Michalowski RL (2002) Stability charts for uniform slopes. J Geotech Geoenviron Eng 128:351–355. https://doi.org/10.1061/(asce)1090-0241(2002)128:4(351)
32. Manouchehrian A, Gholamnejad J, Sharifzadeh M (2014) Development of a model for analysis of slope stability for circular mode failure using genetic algorithm. Environ Earth Sci 71:1267–1277. https://doi.org/10.1007/s12665-013-2531-8
33. Ma K, Feng D, Lawson K, Tsai W-P, Liang C, Huang X, Sharma A, Shen C (2021) Transferring hydrologic data across continents—leveraging data-rich regions to improve hydrologic prediction in data-sparse regions. Water Resour Res. https://doi.org/10.1029/2020wr028600
34. Mahmoodzadeh A, Mohammadi M, Farid Hama Ali H et al (2022) Prediction of safety factors for slope stability: comparison of machine learning techniques. Nat Hazards 111:1771–1799. https://doi.org/10.1007/s11069-021-05115-8
35. Natekin A, Knoll A (2013) Gradient boosting machines, a tutorial. Front Neurorobot. https://doi.org/10.3389/fnbot.2013.00021
36. Nguyen A-P, Martínez MR (2019) MonoNet: Toward interpretable models by learning monotonic features. arXiv [cs.LG]. https://doi.org/10.48550/arXiv.1909.13611
37. 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 JMLR 12(85):2825–2830
38. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf 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. In: Proceedings of the 33rd international conference on neural information processing systems, vol 32, pp 8026–8037. https://doi.org/10.5555/3454287.3455008
39. Read JS, Jia X, Willard J et al (2019) Process-guided deep learning predictions of lake water temperature. Water Resour Res 55:9173–9190. https://doi.org/10.1029/2019wr024922
40. Spencer E (1967) A method of analysis of the stability of embankments assuming parallel inter-slice forces. Géotechnique 17:11–26. https://doi.org/10.1680/geot.1967.17.1.11
41. Stone M (1974) Cross-validatory choice and assessment of statistical predictions. J R Stat Soc 36:111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x
42. Sah NK, Sheorey PR, Upadhyaya LN (1994) Maximum likelihood estimation of slope stability. Int J Rock Mech Min Sci Geomech Abstr 31:47–53. https://doi.org/10.1016/0148-9062(94)92314-0
43. Sakellariou MG, Ferentinou MD (2005) A study of slope stability prediction using neural networks. Geotech Geol Eng 23:419–445. https://doi.org/10.1007/s10706-004-8680-5
44. Sun J, Zhao Z (2013) Stability charts for homogenous soil slopes. J Geotech Geoenviron Eng 139:2212–2218. https://doi.org/10.1061/(asce)gt.1943-5606.0000938
45. Stanley TA, Kirschbaum DB, Sobieszczyk S, Jasinski MF, Borak JS, Slaughter SL (2020) Building a landslide hazard indicator with machine learning and land surface models. Environ Model Softw 129:104692. https://doi.org/10.1016/j.envsoft.2020.104692
46. Sharma A, Wehrheim H (2020) Testing monotonicity of machine learning models. arXiv [cs.LG]. https://doi.org/10.48550/arXiv.2002.12278
47. Stanley TA, Kirschbaum DB, Benz G, Emberson RA, Amatya PM, Medwedeff W, Clark MK (2021) Data-driven landslide nowcasting at the global scale. Front Earth Sci. https://doi.org/10.3389/feart.2021.640043
48. Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang J (2016) Convolutional Neural Networks for medical image analysis: Full training or fine tuning? IEEE Trans Med Imaging 35:1299–1312. https://doi.org/10.1109/tmi.2016.2535302
49. Wainer J, Cawley G (2021) Nested cross-validation when selecting classifiers is overzealous for most practical applications. Expert Syst Appl 182:115222. https://doi.org/10.1016/j.eswa.2021.115222
50. Willard J, Jia X, Xu S, Steinbach M, Kumar V (2023) Integrating scientific knowledge with machine learning for engineering and environmental systems. ACM Comput Surv 55:1–37. https://doi.org/10.1145/3514228

51. Yang CX, Tham LG, Feng XT, Wang YJ, Lee PKK (2004) Two-stepped evolutionary algorithm and its application to stability analysis of slopes. J Comput Civ Eng 18:145–153. https://doi.org/10.1061/(asce)0887-3801(2004)18:2(145)
52. You S, Ding D, Canini K, Pfeifer J, Gupta MR (2017) Deep lattice networks and partial monotonic functions. In: Proceedings of the 31st international conference on neural information processing systems, pp 2985–2993. https://doi.org/10.5555/3294996.3295058
53. Zhou J, Li E, Yang S, Wang M, Shi X, Yao S, Mitri HS (2019) Slope stability prediction for circular mode failure using gradient boosting machine approach based on an updated database of case histories. Saf Sci 118:505–518. https://doi.org/10.1016/j.ssci.2019.05.046