Scientific Research Publishing

International Journal of Geosciences, 2025, 16(10), 738-761
https://www.scirp.org/journal/ijg
ISSN Online: 2156-8367
ISSN Print: 2156-8359

Performance Evaluation of Soft Computing Techniques in Blast-Induced Predictions: The Case of Backbreak

Festus Kunkyin-Saadaari[1,2*], Charles Asiedu[1,2], Ephraim Atta-Duncan[1], Victor Kwaku Agadzie[1,2], Mary Adu-Gyamfi[1]

[1]Department of Mining Engineering, Faculty of Mining and Minerals Technology, University of Mines and Technology (UMaT), Tarkwa, Ghana
[2]Department of Mining Systems and Rock Engineering, Division of Geo-Industrial Sciences Technology, Festari Institute of Professional Studies, Tarkwa, Ghana
Email: *fsaadaari@umat.edu.gh, ingasiedu@gmail.com, ephraimduncan68@gmail.com, vkagadzie@gmail.com, maryadugyamfiofficial@gmail.com

How to cite this paper: Kunkyin-Saadaari, F., Asiedu, C., Atta-Duncan, E., Agadzie, V.K. and Adu-Gyamfi, M. (2025) Performance Evaluation of Soft Computing Techniques in Blast-Induced Predictions: The Case of Backbreak. *International Journal of Geosciences*, **16**, 738-761.
https://doi.org/10.4236/ijg.2025.1610037

Received: August 23, 2025
Accepted: October 26, 2025
Published: October 29, 2025

Abstract

The accurate prediction of backbreak, a crucial parameter in mining operations, has a significant influence on safety and operational efficiency. The occurrence of this phenomenon is detrimental to the safety, capital and human resources of a mine. This framework applies machine learning algorithms to predict backbreak. An enhanced precision will be explored specifically employing gradient boosting decision trees (GBDT), light gradient boosting machines (LightGBM), backpropagation neural network (BPNN), Graph Neural Networks (GNNs) and Kolmogorov-Arnold Network (KAN) algorithm. The study utilises a comprehensive dataset from the Goldfields Ghana Limited, Damang Mine comprising geomechanical, drilling, and blasting parameters (burden, spacing, stemming height, geometric stiffness, and powder factor) as well as backbreak data. The potential of each algorithm to learn the intricate relationships between the input features and backbreak values is investigated. To quantitatively assess the predictive performance of the models, the evaluation metrics, coefficient of determination (R^2), mean absolute error (MAE), and mean square error (MSE) are employed. The results revealed that GBDT and BPNN algorithms exhibited robust predictive capabilities, capturing the complex non-linear patterns in the dataset, achieving higher R^2 values (97% and 92% respectively) and lower MAE scores (0.2603 and 0.456, respectively) and MSE scores (0.1456 and 0.3798, respectively). The LightGBM and KAN models also showed their predictive potential and captured the complex non-linear patterns in the dataset but not as efficiently as GBDT and BPNN. GNN

showed the least performance in backbreak prediction. The findings highlighted the potential of the GBDT model to enhance backbreak prediction accuracy, thereby aiding in safer and more efficient excavation practices.

Keywords

Backbreak Prediction, Machine Learning, Open-Pit Mining, Gradient Boosting, Neural Networks, Blast Design Parameters

1. Introduction

Backbreak is defined as the undesirable fracturing of rock behind the final wall of an open-pit mine due to blasting. This phenomenon can lead to instability of mine walls, equipment damage, poor fragmentation, and increased mining costs [1], as resources must be spent to treat and correct its occurrence and effects. To prevent backbreak occurrence, various factors must be considered, including the blasting pattern's geometrical properties, explosives' qualities, and the geomechanical characteristics of the rock mass [2]. In a blast pattern, backbreak is characterized by fractured rocks extending past the rear row of holes, with its quantification typically involving measuring the distance between the final excavation line and the point where the rock breaks [3].

The evolution of backbreak prediction methods in mining operations has undergone a significant transformation over the past decades. According to [1], blast design in the past relied on empirical modelling approaches that sought to achieve optimal outcomes such as appropriate fragmentation, minimized backbreak, optimal muck pile configuration, and reduced boulder formation. Further stating that, the empirical methods lacked a direct mechanism for backbreak prediction and typically considered only a limited subset of the relevant parameters that influence blasting performance. The early 2000s saw advancement through numerical modeling approaches, with finite element analysis demonstrating improved accuracy against field data, though these methods often required extensive computational resources [4]. [5] Kuz-Ram model, while primarily focused on fragmentation, provided valuable insights into backbreak mechanisms.

The transition to computational methods marked a significant shift in prediction approaches. The Distinct Element Method (DEM) offered detailed insights into rock mass behaviour and fracture propagation, though its computational demands often made it impractical for real-time applications [6]. [7] then introduced neural networks to evaluate blasting pattern parameters' effects on backbreak, demonstrating the potential of computational methods in improving prediction accuracy. This was followed by [1] application of fuzzy set theory, which provided a more nuanced approach to handling the uncertainties inherent in backbreak prediction.

Machine learning applications in mining have expanded considerably,

show+casing the versatility and effectiveness of these techniques across different operational aspects. [8] demonstrated the successful application of machine learning in predicting slope deformation in open-pit mines, while [9] extended its use to predict haul truck fuel consumption. The implementation of deep learning algorithms for dust concentration monitoring by [10] and rockburst prediction by [11] further exemplifies the growing sophistication of machine learning applications in mining operations. These developments have established a strong foundation for the application of advanced algorithms in solving complex mining challenges.

Recent developments in machine learning techniques have introduced more sophisticated approaches to backbreak prediction. The emergence of ensemble learning methods, particularly gradient boosting algorithms, has opened new possibilities for improving prediction accuracy. [12] development of LightGBM as an efficient Gradient Boosting Decision Tree framework, coupled with [13] comparative analysis of GBDT, XGBoost, and LightGBM algorithms in rock stability prediction, has demonstrated the potential of these methods in mining applications. The advancement in neural network architectures, as reviewed by [14], has further enhanced the capability to capture complex non-linear relationships in mining parameters. These developments, combined with improvements in computational efficiency and data processing capabilities, have set the stage for more accurate and reliable backbreak prediction methods.

The accurate prediction of backbreak in open-pit mining operations remains a significant challenge that has not been adequately resolved by existing methods. While previous studies have explored various approaches-from [7] evaluation of blasting parameters using neural networks to [15] RES-based model for risk assessment, none have comprehensively addressed the complex non-linear relationships between blasting pattern geometry, explosive properties, and geomechanical characteristics. Although machine learning has shown promise in mining applications, as demonstrated by [8] in slope deformation prediction and [10] in dust concentration monitoring, its full potential for backbreak prediction remains unexplored. The continued occurrence of unwanted rock fracturing beyond the final pit wall, resulting in safety hazards, equipment damage, poor fragmentation, and increased mining costs [1], underscores the limitations of current prediction methods.

This research specifically addresses the gap in applying advanced machine learning techniques to predict backbreak more accurately. The study has three primary objectives: (1) to develop and implement a comparative analysis of GBDT, LightGBM, BPNN, GNN and KAN algorithms for backbreak prediction; (2) to evaluate the predictive performance of these algorithms using real-world data from Goldfields Ghana Limited, Damang Mine; and (3) to determine the most effective algorithm combination for improving backbreak prediction accuracy in open-pit mining operations. While [3] and [16] have explored individual machine learning approaches for backbreak prediction, there has not been a comprehen-

sive comparative analysis of these specific algorithms in this context. Despite the proven efficiency of LightGBM in complex datasets [12] and the established capabilities of GBDT and BPNN in mining applications [13] [14], their combined potential for backbreak prediction remains unexplored.

The remainder of this paper is structured as follows: Section 2 describes the study area and presents the methodology, including detailed descriptions of the GBDT, LightGBM, BPNN, GNN and KAN algorithms and their implementation. Section 3 presents the results and discusses the comparative performance of the algorithms using evaluation metrics such as coefficient of determination (R^2), mean absolute error (MAE), and mean square error (MSE). Finally, Section 4 concludes the study by summarizing the key findings and their implications for improving backbreak prediction in open-pit mining operations. This systematic approach represents a novel contribution to the field by combining advanced ML techniques in a way that has not been previously attempted for backbreak prediction, with the potential to significantly improve safety and operational efficiency in open-pit mining operations.

2. Materials and Methods Used

2.1. Study Area Description

The Damang Gold Mine is in southwest of Ghana, West Africa approximately 300 km west of Accra (5˚11'N, 1˚57'W), and about 30 km north of Tarkwa. The deposit is found within the Tarkwaian sediments of the Ashanti Belt. The Ashanti Belt strikes northeast with a broadly synclinal structure and comprises lower Proterozoic sediments, as well as volcanic rocks and the Birimian System metasediments. The Tarkwaian sedimentary sequence within the Ashanti Belt exhibits lower-intensity metamorphism compared to other rock formations in the region. It is composed of immature sedimentary units and coarse-grains that include several series, such as the Huni series, the Banket series, the Damang phyllite series, and the Kawere series.

The mine mineralisation occurs within the Tarkwaian's paleoplacer deposits. These deposits are composed of distinct tabular units of quartz pebble conglomerates, commonly referred to as reefs. These reefs are interspersed between argillaceous sandstone blocks and quartzites. The mineralisation in the Damang deposit is likely associated with the concentration of gold within these paleoplacer deposits. Hydrothermal, oxide and Witwatersrand-style paleoplacer mineralisation are exploited at the mine. The hydrothermal deposits occur within the pyrite and pyrrhotite alteration selvedge and are found next to the echelon quartz veins. Accessory vein minerals such as ilmenite, carbonate, tourmaline and apatite are also linked to the gold. Conventional open-pit methods are employed, which utilise standard truck-shovel techniques to mine the ore. Loading and hauling activities are undertaken using excavators in backhoe configurations. To support drill-and-blast and haulage operations, vehicle, road, and bench maintenance, dust, and erosion control are undertaken. For optimal fragmentation and blast control,

both conventional (Nonel) and electronic detonators are employed in rock blasting. The location of the study area is depicted in Figure 1.

Figure 1. Location of study area (source: designed by the authors).

2.2. Datasets

A dataset comprising the parameters: spacing (S), burden (B), geometric stiffness (K), stemming height (T), backbreak (BB) and powder factor (P) was used. A total of 60 data points were obtained from the Goldfields Ghana Limited, Damang Mine. Table 1 represents the statistical description of the dataset employed.

Table 1. Statistical description of dataset.

Parameter	Unit	Maximum	Minimum	Mean	Standard Deviation
Burden (B)	m	6.50	0.73	3.11	0.75
Spacing (S)	m	6.00	0.57	3.82	0.73
Stemming (T)	m	4.50	0.73	2.93	0.64
Powder Factor (P)	kg/m^3	1.64	0.15	0.57	0.25
Geometric Stiffness (K)	-	8.65	0.44	4.69	1.72
Backbreak (BB)	m	10.00	1.00	3.88	2.11

2.3. Data Preparation

The train_test_split was used on the dataset to separate into training and testing sets for all algorithms. This is important to assess how effectively the model generalizes to new, unseen, or future data. The dataset was split into 80:20, where 80% was for training with the corresponding 20% for testing. 12 out of 60 data points were for testing, and 48 out of 60 data points were for training. The training set (comprising input parameters and target variable) is used in a process of adjusting

the model's internal parameters to make it capable of learning patterns and relationships from the data after which the testing set is employed to assess the trained model's performance in real-world scenarios.

2.4. Method Used

This prediction task employs machine learning algorithms for regression modelling using GBDT, LightGBM, BPNN, GNN and KAN. These algorithms were selected for their proven capabilities in handling non-linear relationships and complex datasets in mining applications [17]. Using a standard 80:20 training-testing split ratio, all three algorithms process the same input parameters (burden, spacing, stemming height, powder factor, and geometric stiffness) to predict back-break values.

2.4.1. GBDT

Theory

GBDT ensures that the errors produced in the past are fixed by adding the prior under-fitted forecasts to the ensemble one after the other. In the gradient boosting process, models are iteratively built, with the errors and losses (loss function) from previous models being calculated and used as target variables to improve predictions on subsequent models and combining the predictions from the multiple models to minimise errors [13]. This process repeats until a stopping condition is satisfied or a maximum number of models is reached [18]. The gradient descent method is utilised in this algorithm to minimise the loss function. According to [19] for the initial model prediction Equation (1). **Figure 2** illustrates the implementation of the GBDT algorithm.

$$F_{0(x)} = \arg\min \gamma \sum_{i=1}^{n} nL\left(y_i, \gamma\right) \tag{1}$$

where; L is the loss function, γ is the predicted value, y_i is the observed value, n is the number of observations, $\arg\min \gamma$ denotes finding the value of γ that minimizes the loss function

The loss function is calculated by Equation (2),

$$L = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \gamma\right)^2 \tag{2}$$

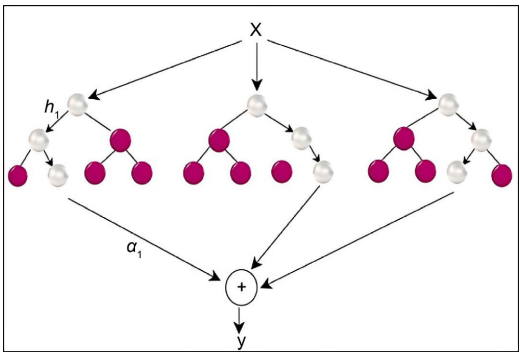

Figure 2. Architecture of GBDT.

Implementation

The dataset (n = 60) was initially sectioned using stratified partitioning into training (n_train = 48, 80%) and test (n_test = 12, 20%) sets depicted in Figure 3. To enhance model accuracy and ensure robust performance assessment, particularly given the limited dataset size, k-fold cross-validation with k = 5 was implemented on the training data. This process systematically divided the training dataset into five equally sized folds. The GBDT model underwent five iterations of training and evaluation, with each iteration utilizing a distinct fold as the validation set while the remaining four folds served as training data. This approach allowed for a more reliable estimation of the model's generalization performance and demonstrated its stability across different subsets of the training data.

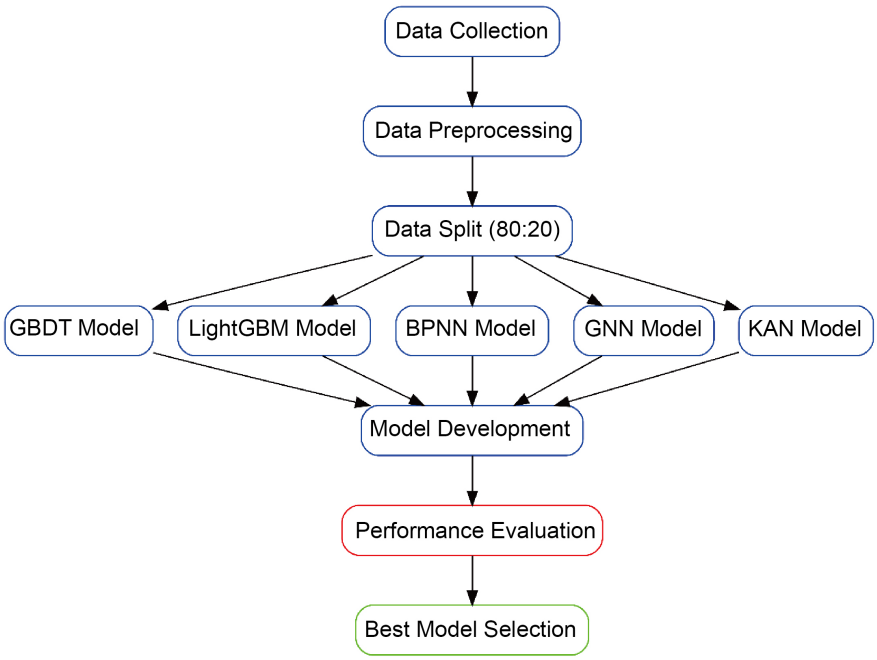

Figure 3. Model implementation process.

Through systematic hyperparameter optimization, it was determined that 300 estimators yielded optimal model performance. The model's configuration was further refined through additional hyperparameters, notably the maximum depth and learning rate. An exhaustive grid search strategy, implemented via GridSearchCV, was employed to identify the optimal hyperparameter combination. This methodological approach systematically explored the hyperparameter space, evaluating various permutations to determine the configuration that maximized the predictive model's performance. A summary of the optimal tuned hyperparameters is presented in Table 2.

2.4.2. LightGBM

Theory

According to [20], LightGBM is a robust gradient boosting framework that lev-

erages decision trees to enhance model efficiency and minimize memory consumption. It introduces two innovative techniques known as Gradient-based One Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [12] [21]. The GOSS technique gives more weight to instances with larger gradients, indicating that they are under-trained. These instances contribute more to the calculation of information gain. GOSS selectively keeps instances with large gradients (above a threshold or within the top percentiles) and only randomly discards instances with small gradients [22]. This improves the accuracy of information gain estimation compared to uniform random sampling, especially when the range of information gain values is large. EFB bundles exclusive features (mutually exclusive features that have no simultaneous non-zero values) into a single feature. This significantly reduces the complexity of histogram building from being proportional to the number of data points multiplied by the number of features to being proportional to the number of data points multiplied by the number of bundles. Since the number of bundles is much smaller than the number of features, this approach improves the training speed of the framework without sacrificing accuracy. The architecture of LightGBM is depicted in **Figure 4**.

Parameters that can be tuned in LightGBM are max depth (maximum limit of tree depth is effective in controlling overfitting), categorical feature (feature to be used for model training), bagging fraction (proportion of data selected for each iteration.) num iterations (total count of iterations to be executed), num leaves (total count of leaves in a tree), max bin (maximum count of bins for grouping feature values), min data in a bin (least amount of data contained in a bin), task (operation to be performed, either training or prediction) and feature fraction (proportion of features selected for each iteration). Overfitting, which occurs when models excessively capture details and noise from the training data, resulting to poor performance on new data, should be watched for in decision trees. The model accuracy is enhanced and the risk of overfitting is reduced through *K*-fold cross-validation technique [23].

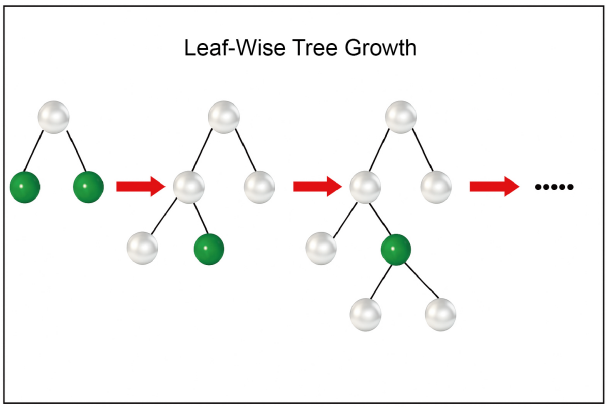

Figure 4. Architecture of LightGBM.

Implementation

The experimental methodology employed a systematic data partitioning ap-

proach, where the dataset (n = 60) was stratified and randomly split into training (n_train = 48, 80%) and test (n_test = 12, 20%) sets. For hyperparameter optimization, an exhaustive grid search strategy was implemented using GridSearchCV, coupled with a k-fold cross-validation mechanism (k = 5) to ensure robust model selection and mitigate overfitting risks. The grid search systematically explored the defined hyperparameter space to identify the optimal configuration that maximized model performance. The k-fold cross-validation procedure partitioned the training data into five stratified subsets, enabling comprehensive model evaluation across multiple data combinations while maintaining the statistical distribution of the target variable. This rigorous validation framework ensured the selected hyperparameters generalized well across different data subsets, while the held-out test set provided an unbiased assessment of the final model's performance. Table 2 presents the optimal hyperparameters for the model.

Table 2. Hyperparameter tuning for algorithms.

Algorithm	Hyperparameter	Optimal Tuned Values
GBDT	n_estimators	300
	learning_rate	0.01
	max_depth	5
LightGBM	n_estimators	100, 200, 300, 400, 500
	learning_rate	0.1, 0.05, 0.01, 0.001
	max_depth	3, 4, 5
BPNN	hidden_layer_sizes	64
	activation	Relu
	solver	adam
	max_iter	1000
	random_state	42
GNN	hidden_channels	224
	dropout_rate	0.3
	learning_rate	0.00425
	epochs	500
KAN	width	[5, 5]
	grid	3
	k (spline order)	2
	lambda	0.001

2.4.3. BPNN

Theory

BPNN, originally discovered in 1969, has been the most widely used neural network because of its ability to learn complex relations [24]. BPNN enables the computation of the loss function gradient concerning all network weights [14]. The

BPNN learning process involves forward propagation and backward propagation. During forward propagation, input signals enter the neural network from the input layer through hidden layers to the output layer [25]. The network's weights and biases remain unchanged, and the current layer's neuron outputs affect only the following layer. If the desired output isn't reached, the process can switch to backpropagation to correct errors [26]. With backpropagation, the error signal is the difference between the actual and desired network outputs and sent from the output layer back through the layers to the input layer [25] [27]. The network's weights are remodelled based on this error feedback, continuously adjusting them to bring the actual output closer to the expected output [25]. The weights of the network are then updated according to a gradient descent rule [14]. The architecture of BPNN is shown in **Figure 5**.

Implementation

Feature preprocessing was implemented through the MinMaxScaler class from scikit-learn, which normalized input parameters to a uniform range, thereby preventing the dominance of features with larger magnitudes in the learning process. The dataset ($n = 60$) underwent stratified partitioning into training (n_train = 48, 80%) and test (n_test = 12, 20%) sets. To ensure robust model evaluation and hyperparameter tuning, especially given the limited dataset size, k-fold cross-validation with $k = 5$ was applied to the training data. This involved iteratively training and validating the model across five distinct subsets of the training data.

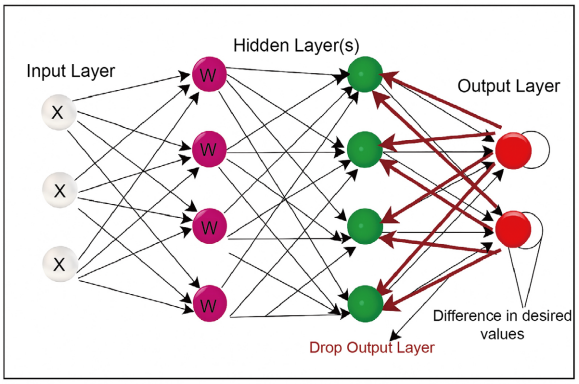

Figure 5. BPNN Architecture.

The neural network architecture was configured with specific hyperparameters: ReLU activation function, hidden layer dimensions of 64 neurons, and a maximum iteration limit of 1000 epochs (representing complete forward and backward passes of the entire dataset). A random state of 42 was established to ensure experimental reproducibility, while the network's weight optimization employed the Adam (Adaptive Moment Estimation) optimizer. The Adam optimizer is known for its adaptive learning rate mechanism, achieved by tracking exponentially decaying averages of previous gradients and their squared values. This allows it to adjust the learning rate dynamically, improving convergence rates in various optimization problems [28]. The ReLU activation function is mathematically as

shown in Equation (3):

$$h = \max(0, a) \tag{3}$$

where a represents any real number, such that the function returns 0 for inputs less than or equal to 0, and returns a otherwise. A summary of the BPNN optimally tuned hyperparameter configuration is summarized in **Table 2**.

2.4.4. GNN

Theory

Graph Neural Networks (GNNs) extend traditional neural networks to operate on graph-structured data, making them effective for capturing complex relationships between data points. GNNs represent data as nodes interconnected by edges, where each node contains feature information, and the edges define the relationships between nodes [29].

The Graph Convolutional Network (GCN), first proposed by [30], is a specific variant of GNN that generalizes convolutional operations to graph-structured data. In GCNs, each node's representation is updated by aggregating information from its neighbours. The basic operation in a GCN layer is shown in Equation (4):

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{4}$$

where $H^{(l)}$ represents the node features at layer l, $\tilde{A} = A + I$ is the adjacency matrix with added self-connections, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $W^{(l)}$ is the trainable weight matrix for layer l, and σ is a non-linear activation function [31].

For regression tasks like backbreak prediction, GNNs offer advantages by considering the relationships between different blasting scenarios rather than treating each instance independently. This approach enables the model to leverage similarities and patterns across different blast design parameters to improve prediction accuracy [32].

Implementation

Feature normalization was implemented through MinMaxScaler to ensure uniform feature scaling, preventing features with larger magnitudes from dominating the learning process. The dataset (n = 60) underwent stratified partitioning into training (n_train = 48, 80%) and test (n_test = 12, 20%) sets depicted in **Figure 3**. To ensure robust model evaluation and hyperparameter optimization, particularly given the limited dataset size, k-fold cross-validation with k = 5 was applied to the training data. This involved iteratively training and validating the model across five distinct subsets of the training data.

A k-nearest neighbors (kNN) graph was constructed to establish the topological structure of the data. The choice of a neighborhood size of k=10 for the kNN graph was determined through empirical evaluation during the hyperparameter tuning process, where various values of k were tested to identify the configuration that best captured the local data manifold while balancing computational efficiency

and predictive performance. This selection aimed to ensure sufficient connectivity within the graph to propagate information effectively without introducing excessive noise from distant neighbors.

A three-layer Graph Convolutional Network (GCN) architecture was implemented using the PyTorch Geometric framework. The network architecture incorporated dropout regularization to prevent overfitting. The model configuration featured the following components:

i. Input layer: Accepting normalized blasting parameters,

ii. Three graph convolutional layers with ReLU activations,

iii. Two fully connected layers for the final prediction,

iv. Dropout regularization between layers.

Hyperparameter tuning for the GNN model, including the learning rate, number of hidden units in each layer, and dropout rates, was conducted using an exhaustive grid search strategy. This process systematically explored a predefined range of values for each hyperparameter, with the optimal combination selected based on the model's performance during the 5-fold cross-validation. The training was conducted using the Adam optimizer, known for its adaptive learning rate mechanism, which dynamically adjusts learning rates to improve convergence. The mean squared error (MSE) loss function was employed for optimization, with model checkpoints saved periodically during training to capture the best performing model throughout the training epochs.

2.4.5. KAN

Theory

The Kolmogorov-Arnold Network (KAN) is a neural network architecture inspired by the Kolmogorov-Arnold representation theorem, which states that any multivariate continuous function can be represented as a composition of continuous functions of a single variable and addition operations [33]. Unlike traditional neural networks that use fixed activation functions, KANs employ learnable univariate activation functions represented as splines, providing greater flexibility in function approximation. KAN architecture is presented in **Figure 6**.

The KAN's mathematical foundation stems from Kolmogorov's superposition theorem, which stipulates that any continuous function $f:[0,1]^n \to R$ is expressed in Equation (5)

$$f\left(x_1, x_2, \cdots, x_n\right) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}\left(x_p\right) \right) \tag{5}$$

where Φ_q and $\varphi_{\{q,p\}}$ are continuous univariate functions.

KANs implement this theorem by representing each univariate function as a spline with learnable parameters. The spline function of order k with grid size g has a form shown in Equation (6)

$$s(x) = \sum_{i=0}^{g} c_i B_i^k\left(x\right) \tag{6}$$

where B_i^k are basis splines and c_i are learnable coefficients [34]. This approach provides KANs with high expressivity while maintaining interpretability, as the

network can be decomposed into simpler univariate functions.

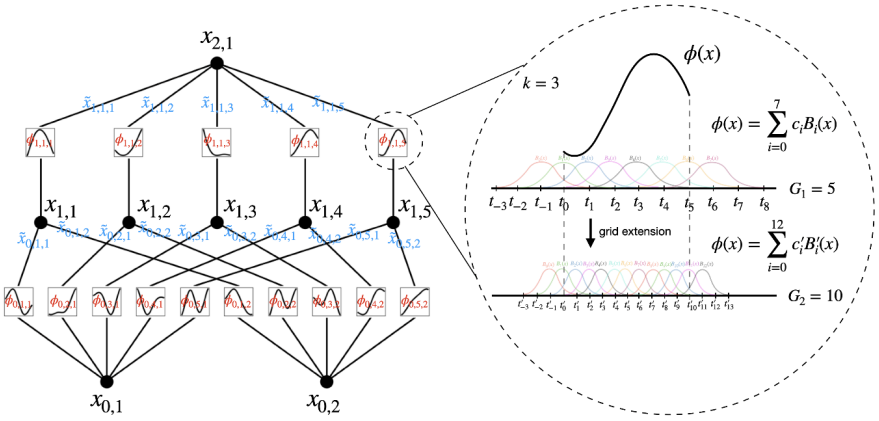

Figure 6. KAN Architecture [35].

Implementation

The KAN implementation used a systematic approach for feature preprocessing, model optimization, and evaluation. The dataset ($n = 60$) underwent feature standardization using StandardScaler to normalize all input parameters to a consistent scale. The dataset was then stratified and randomly split into training (n_train = 48, 80%) and test (n_test = 12, 20%) sets.

To ensure robust model evaluation and hyperparameter tuning, particularly given the limited dataset size, k-fold cross-validation with $k = 5$ was applied to the training data (the 48 data points). This involved iteratively training and validating the model across five distinct subsets of the training data, providing a more reliable estimate of the model's generalization performance and assessing its stability.

The architecture featured an input layer dimensioned according to the feature count, followed by a hidden layer of neurons, and an output layer with a single neuron. The model was configured with a specific grid size and spline order, which provided a balanced trade-off between model complexity and generalization capability. A regularization parameter was applied to prevent overfitting. The training was conducted using the Adam optimizer. The mean squared error loss function was employed for optimization, with model checkpoints saved periodically during training. The final model was then validated with the held-out test set (the remaining 12 datasets) to provide an unbiased assessment of its performance.

2.4.6. *K*-Fold Cross-Validation

K-fold cross-validation (CV) is a widely adopted and robust technique for evaluating machine learning models, particularly when aiming to ensure generalization capability and prevent overfitting [36]. Unlike a single train-test split, which can yield biased performance estimates, k-fold CV partitions the dataset into k equal subsets. The model is then iteratively trained on $k - 1$ folds and validated on the remaining fold, repeating this process k times so that each data point serves as

both a training and validation example [36].

This method provides a more reliable estimate of a model's generalization error by averaging performance metrics across all *k* iterations [37]. Furthermore, it offers crucial insights into model stability; consistent performance across folds indicates robustness and reduced sensitivity to specific data compositions [38]. Consequently, *k*-fold cross-validation is an essential tool for developing reliable and generalizable predictive models. The performance metrics (e.g., R^2, MAE) are computed for each iteration, and the final reported performance is typically the average of these *k* measurements [38].

2.4.7. Hyperparameter Tuning-GridSearchCV

Theory

Hyperparameters govern model capacity and regularisation but are not learned directly from the data; their values must be specified a priori and strongly influence generalisation performance [38]. Grid search with cross-validation remains a widely adopted, reproducible strategy for hyperparameter selection because it exhaustively evaluates a discrete, user-defined search space under a consistent validation protocol [36] [39]. Formally, let $\mathcal{H} = h$ denote the Cartesian product of candidate values for each hyperparameter, $\{V_1, \cdots, V_J\}$. For a *K*-fold scheme, the cross-validated empirical risk for configuration *h* is represented in Equation (7).

$$\hat{R}_{CV}(h) = \frac{1}{K}\sum_{k=1}^{K} L_{val}^{(k)}\left(f_h^{(-k)}\right) \tag{7}$$

where $f_h^{(-k)}$ is the model trained on the *K*-1 non-held-out folds and $L_{val}^{(k)}(\cdot)$ is the loss evaluated on the *k*-th validation fold. The selected configuration minimises this estimate expressed in Equation (8)

$$h^* = \arg\arg \hat{R}_{CV}(h) \tag{8}$$

Because the grid is discrete, its size grows multiplicatively with the number of tuned hyperparameters, this is expressed in Equation (9).

$$|H| = \prod_{j=1}^{J} |V_j| \tag{9}$$

so practitioners often balance coverage and computational cost by focusing on the most influential knobs (e.g., learning rate, tree depth, number of estimators for boosting; hidden width and regularisation for neural networks). Compared with alternatives such as random search [40] or Bayesian optimisation [41], grid search is simple, deterministic, and easy to reproduce; however, it can be sample-inefficient in high-dimensional spaces. Cross-validation provides an approximately unbiased estimate of out-of-sample error when folds are i.i.d., but model selection on the same resamples can introduce selection bias; a strictly unbiased assessment may use a held-out test set or nested CV [42].

Several best practices from the literature improve the robustness of GridSearchCV in small-n regression: (i) construct pipelines that apply all preprocessing (e.g., scaling, encoding) within each fold to prevent leakage; (ii) stratify folds by target quantiles

when feasible to stabilise error estimates in heteroscedastic settings; (iii) use evaluation metrics aligned with the study objective (e.g., MAE/MSE/R^2 for this work); and (iv) fix random seeds and shuffle splits to enhance reproducibility [36] [39]. For boosted trees, literature commonly reports strong sensitivity to learning rate and depth lower learning rates with more estimators tend to generalise better, provided early-stopping or adequate regularisation is used [38] [43]. For neural networks, grid sizes are often kept modest and centred on architecture (width, depth) and optimiser settings because training is comparatively expensive [44].

GridSearchCV in the Context of This Study

Consistent with prior work, this study employs scikit-learn's GridSearchCV to systematise hyperparameter selection for tabular learners. For GBDT and LightGBM, grids spanned the principal capacity and optimisation controls (e.g., n_estimators, learning_rate, max_depth), evaluated under $K = 10$ folds, with performance aggregated via the same loss functions later used for testing (MAE, MSE, and R^2). The resulting configurations (see **Table 2**) reflect the literature's guidance on conservative learning rates with sufficient estimators for boosted ensembles. For neural models (BPNN, GNN, KAN), hyperparameters were tuned using compact, problem-informed grids around architecture and regularisation settings while preserving the same K-fold protocol to ensure comparability across learners. All preprocessing (e.g., scaling via MinMaxScaler/StandardScaler) was embedded in fold-wise pipelines to eliminate information leakage, and the final generalisation estimates were computed on the held-out test partition to guard against selection bias.

2.5. Evaluation Metrics Considered

Machine learning prediction models' efficacy and performance can be assessed using evaluation metrics. Mean absolute error (MAE), coefficient of determination (R^2), and mean squared error (MSE) are the three metrics used in this study to assess how well the models performed. R^2 evaluates the extent to which the observed results align with the predicted values from the regression line. This assessment provides insights into the degree to which the independent variable accounts for the variability in the model. Essentially, it gauges the effectiveness of the model in explaining the dataset. Mathematically R^2 can be expressed as shown in Equation (10) [45]

$$R^2 = \frac{SSR}{SST} = \frac{\sum\left(\hat{y}_i - \underline{y}\right)^2}{\sum\left(y_i - \underline{y}\right)^2} \tag{10}$$

where SSR is Sum Square of Residuals (squared difference between the predicted ($\hat{y}_i$) and the average value (y_i)). SST is Sum Square of Total (squared difference between the actual (y) and average value (y_i)). The MAE is a measure of the prediction error of a model; it is measured by taking the average of the absolute difference between actual values and the predictions. For this, the number of datapoints (n), actual output value (y) and predicted output value ($\hat{y}$) are considered Equation (11) [46].

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{11}$$

MSE is a key metric in statistical analysis and machine learning, used to measure the variation between actual and expected results. The average of the squared deviations between these values is used to compute it, which serves to magnify larger errors, making it particularly useful for models that aim to minimise prediction errors. The mathematical expression for MSE is the expectation of the squared deviation of an estimator from the true parameter value, encompassing both the variance and the bias squared of the estimator [47]. Mathematically, MSE can be expressed as shown in Equation (12)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{12}$$

where y is the true value, $\hat{y}$ is the predicted value and n is the number of datapoints.

3. Results and Discussion

Model performance

The integration of k-fold cross-validation into the model development process significantly enhanced the robustness and reliability of the performance evaluation for the five machine learning models: Gradient Boosting Decision Trees (GBDT), Light Gradient Boosting Machine (LightGBM), Back Propagation Neural Network (BPNN), Graph Neural Network (GNN), and Kolmogorov-Arnold Network (KAN). This rigorous validation approach, particularly crucial given the dataset's size, provides a more accurate representation of each model's generalization capabilities. The comparative analysis, encompassing both cross-validation (CV) and independent test set results, revealed distinct patterns in their predictive efficacy for backbreak estimation. The evaluation metrics; Coefficient of Determination (R^2), Mean Absolute Error (MAE), and Mean Squared Error (MSE) were employed to quantitatively assess the models' performance.

Cross-Validation Performance

Table 3 presents the detailed cross-validation performance results for each model. These metrics, averaged across five folds, offer a more stable and less biased estimate of the models' true performance. GBDT demonstrated a robust performance with an R^2 of 0.6438 ± 0.2196, MAE of 0.5812 ± 0.1999, and MSE of 1.0482 ± 0.5176. While these values might appear lower than typical R^2 values for highly accurate models, the standard deviations indicate a degree of variability across the folds, which is expected in cross-validation and provides a more realistic view of performance. KAN exhibited the highest R^2 in cross-validation at 0.7753 ± 0.1901, coupled with a MAE of 0.5498 ± 0.1266 and MSE of 0.6178 ± 0.3083, suggesting its strong and consistent performance across different data subsets. BPNN showed an R^2 of 0.3455 ± 0.4226, MAE of 1.0498 ± 0.4577, and MSE of 2.0794 ± 1.0477. LightGBM and GNN, however, displayed less favorable performance in the cross-validation phase, with negative R^2 values, indicating that

their predictions were worse than simply predicting the mean of the target variable. Specifically, LightGBM had an R^2 of −0.2728 ± 0.2903, MAE of 1.7055 ± 0.2245, and MSE of 4.2453 ± 1.1320, while GNN showed an R^2 of −0.2917 ± 0.1771, MAE of 1.7302 ± 0.2405, and MSE of 4.4082 ± 1.2973. The negative R^2 values for LightGBM and GNN in CV suggest potential issues with model stability or suitability for this specific dataset under cross-validation conditions.

Table 3. Model CV performance results.

ML Algorithm	Evaluation Metrics		
	R^2	MAE	MSE
GBDT	0.6438 Â ± 0.2196	0.5812 Â ± 0.1999	1.0482 Â ± 0.5176
LightGBM	−0.2728 Â ± 0.2903	1.7055 Â ± 0.2245	4.2453 Â ± 1.1320
BPNN	0.3455 Â ± 0.4226	1.0498 Â ± 0.4577	2.0794 Â ± 1.0477
GNN	−0.2917 Â ± 0.1771	1.7302 Â ± 0.2405	4.4082 Â ± 1.2973
KAN	0.7753 Â ± 0.1901	0.5498 Â ± 0.1266	0.6178 Â ± 0.3083

Test Set Performance

Table 4 summarizes the performance of the models on the independent test set, providing an unbiased assessment of their generalization capabilities on unseen data. In this crucial evaluation, GBDT emerged as the top-performing model, achieving an impressive R^2 of 0.9728, a MAE of 0.2603, and an MSE of 0.1456. This indicates that GBDT can explain approximately 97.28% of the variance in backbreak predictions on new data, with a low average absolute error and mean squared error. BPNN also demonstrated strong performance on the test set, with an R^2 of 0.9289, MAE of 0.456, and MSE of 0.3798, making it the second-best performer. KAN, despite its strong CV performance, showed a slightly lower R^2 of 0.8719 on the test set, with a MAE of 0.6107 and MSE of 0.6842. LightGBM improved significantly from its CV performance, achieving an R^2 of 0.7276, MAE of 0.9778, and MSE of 1.4553. GNN, however, continued to struggle, exhibiting a negative R^2 of −0.0184, MAE of 1.9642, and MSE of 5.4418, reinforcing the observation that it is not well-suited for this specific prediction task.

Table 4. Model test performance results.

ML Algorithm	Evaluation Metrics		
	R^2	MAE	MSE
GBDT	0.9728	0.2603	0.1456
LightGBM	0.7276	0.9778	1.4553
BPNN	0.9289	0.456	0.3798
GNN	−0.0184	1.9642	5.4418
KAN	0.8719	0.6107	0.6842

Comparison of CV and Test Results

Figure 7 visually represents the model performance comparison of test and CV results. The results presented in Table 3 and Table 4 align with the general trends observed in Figure 7, particularly highlighting the superior performance of GBDT and BPNN. The figure illustrates the R^2 values, MAE, and MSE for both CV and test sets, allowing for a direct visual comparison. The significant improvement in R^2 for GBDT from CV (0.6438) to test (0.9728) underscores the effectiveness of the k-fold cross-validation in hyperparameter tuning and model selection, leading to a highly generalized model. Similarly, BPNN also shows strong and consistent performance across both CV and test sets. KAN, while performing exceptionally well in CV, shows a slight drop in R^2 on the test set, which is still a commendable performance.

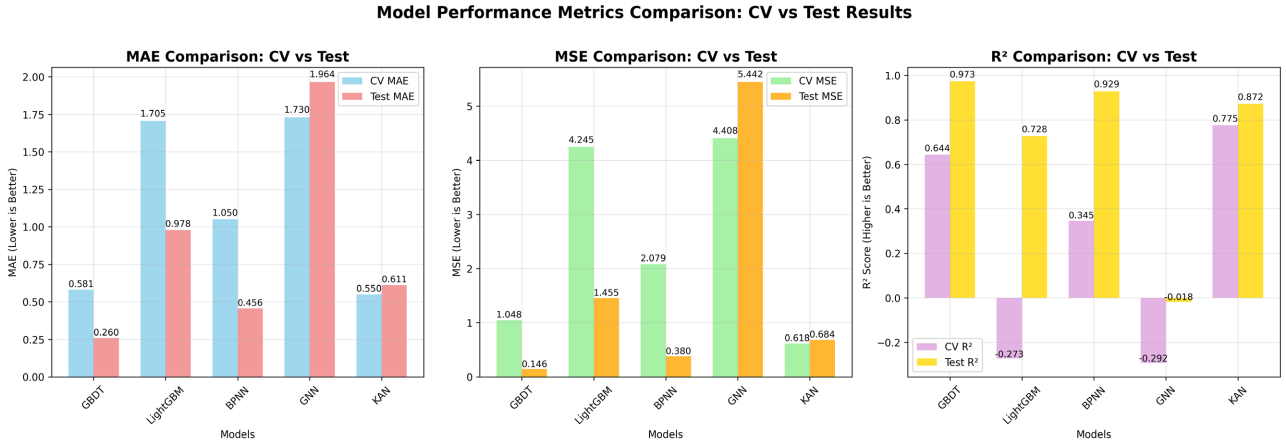

Figure 7. Model Performance Comparison of Test and CV results.

The negative R^2 values for LightGBM and GNN in CV, and GNN in the test set, are critical observations. A negative R^2 indicates that the model performs worse than a simple horizontal line (mean of the observed data), suggesting that these models are not suitable for predicting backbreak with this dataset and methodology. This could be due to various factors, including the complexity of the models relative to the dataset size, or their inability to capture the underlying relationships effectively.

Analysis of the Best Performing Model: GBDT

The Gradient Boosting Decision Trees (GBDT) model consistently demonstrated superior performance, particularly on the independent test set, making it the best-performing model for backbreak prediction in this study. Its high R^2 value of 0.9728 signifies that the model effectively captures the complex, non-linear relationships between the input blasting parameters and backbreak. The low MAE (0.2603) and MSE (0.1456) further confirm its accuracy, indicating minimal deviation between predicted and actual backbreak values. The strength of GBDT lies in its ensemble learning approach, where it sequentially builds decision trees, with each new tree correcting the errors of the previous ones. This iterative refinement

process, coupled with its ability to handle various data types and capture intricate interactions between features, contributes to its robust predictive power.

The implementation of k-fold cross-validation during the training phase further optimized GBDT's hyperparameters, ensuring its generalization capability and preventing overfitting. The model's ability to accurately predict backbreak has significant practical implications for mining operations, enabling more precise blast design, improved safety, and enhanced operational efficiency.

The sensitivity analysis also conducted for the GBDT model in Figure 8 provided crucial insights into the relative importance of input parameters in predicting backbreak. Powder factor emerged as the most influential parameter, followed by burden and stemming height, indicating their critical role in determining backbreak extent. Geometric stiffness demonstrated moderate influence, while stemming showed relatively lower effects. This hierarchical understanding of parameter importance translates directly to practical mining operations.

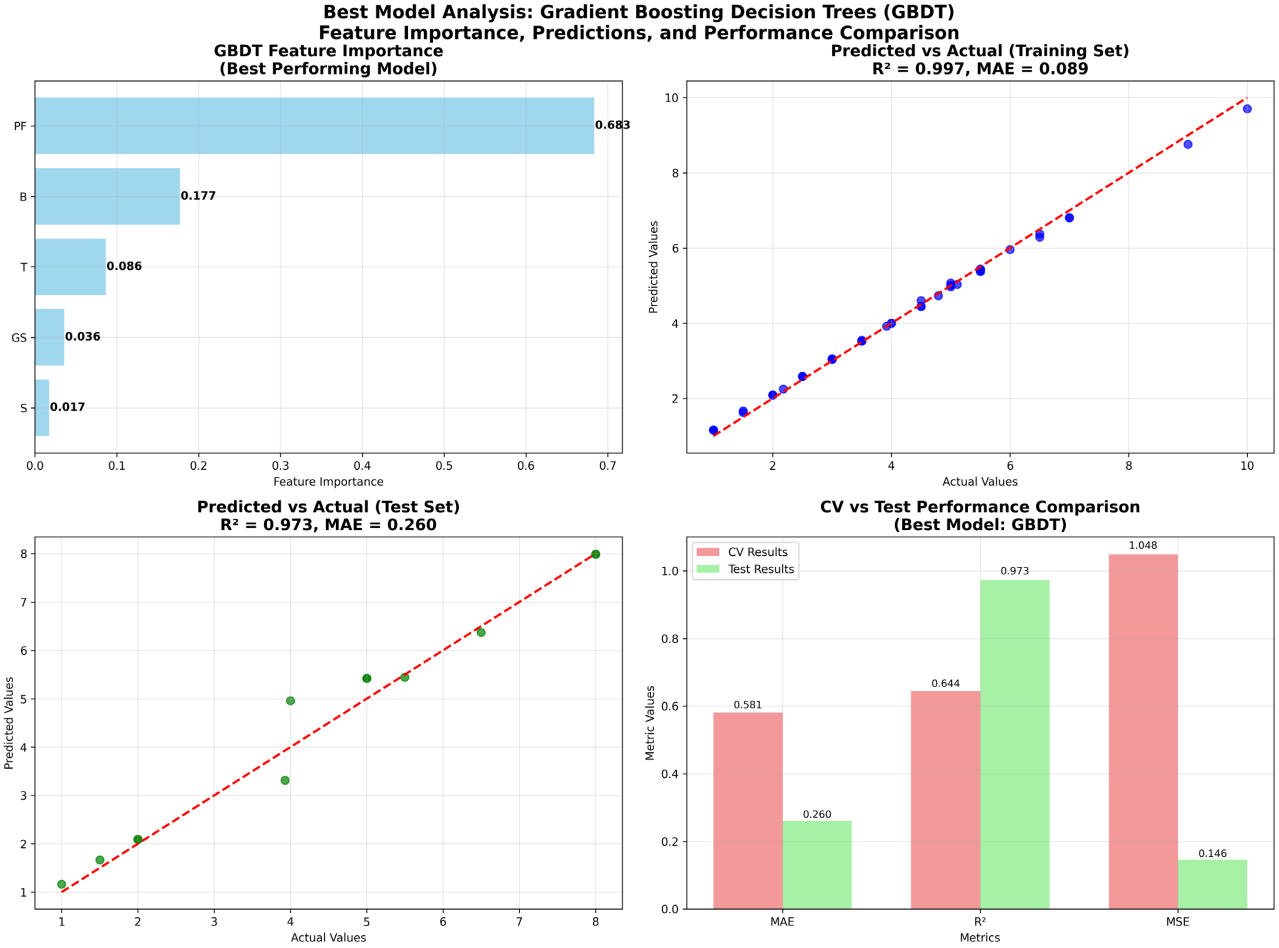

Figure 8. GBDT model performance analysis.

Implications to Research and Practice

Despite the inherent limitations of a relatively small dataset (n = 60) and data

from a single mine site, this study offers significant insights into machine learning applications for backbreak prediction. The integration of *k*-fold cross-validation provides a more robust assessment of model generalization, particularly highlighting the Gradient Boosting Decision Trees (GBDT) model's exceptional performance on the independent test set (R^2 = 0.9728). This high accuracy suggests GBDT's strong potential for practical application in optimizing blast designs, leading to improved safety, reduced operational costs, and enhanced efficiency in open-pit mining. The consistent performance of BPNN and the promising, albeit slightly lower, performance of KAN also indicate their utility. While the results are preliminary and necessitate further validation with larger, more diverse datasets, they underscore the transformative potential of advanced machine learning techniques in addressing complex mining challenges and guiding more precise and safer blasting operations.

4. Conclusions

This study rigorously evaluated the effectiveness of five machine learning algorithms Gradient Boosting Decision Trees (GBDT), Light Gradient Boosting Machine (LightGBM), Back Propagation Neural Network (BPNN), Graph Neural Network (GNN), and Kolmogorov-Arnold Network (KAN) in predicting blast-induced backbreak in open-pit mining operations at Goldfields Ghana Limited, Damang Mine. By integrating *k*-fold cross-validation and assessing performance on an independent test set, the analysis provided a comprehensive understanding of each model's generalization capabilities.

The GBDT model consistently demonstrated superior performance, particularly on the independent test set, achieving an impressive R^2 of 0.9728, a Mean Absolute Error (MAE) of 0.2603, and a Mean Squared Error (MSE) of 0.1456. This highlights its robust predictive power and ability to capture complex non-linear relationships in backbreak prediction. The BPNN model also exhibited strong performance on the test set (R^2 = 0.9289, MAE = 0.456, MSE = 0.3798), positioning it as a highly effective alternative. KAN, while showing excellent cross-validation performance (R^2 = 0.7753 ± 0.1901), maintained a commendable R^2 of 0.8719 on the test set. In contrast, LightGBM showed moderate performance on the test set (R^2 = 0.7276), while GNN consistently struggled, exhibiting negative R^2 values in both cross-validation and test sets, indicating its unsuitability for this specific task.

The findings underscore the significant potential of advanced machine learning approaches, particularly GBDT and BPNN, for enhancing backbreak prediction accuracy. While these promising results are based on a limited dataset and require further validation with larger, more diverse datasets from multiple mine sites, they establish a strong foundation for the application of machine learning in optimizing blast design and improving safety and efficiency in open-pit mining. Future research should focus on expanding data collection, incorporating additional geological and operational parameters, and exploring hybrid algorithms or ensem-

ble methods that leverage the strengths of top-performing models like GBDT and BPNN. Further investigation into interpretable models like KAN could also provide valuable insights into the underlying physical relationships. Ultimately, integrating these predictive models with real-time monitoring systems holds the key to achieving more robust and adaptable backbreak prediction solutions.

5. Limitations

This study provides a comprehensive analysis of various machine learning algorithms for backbreak prediction; however, it is important to acknowledge certain limitations, primarily concerning the scope of input variables. While our dataset from Goldfields Ghana Limited, Damang Mine, included critical parameters such as burden, spacing, stemming height, powder factor, and geometric stiffness, other potentially influential variables were not incorporated. Specifically, factors such as bench height, rock density, and explosive type were omitted from the analysis. This exclusion was primarily due to the inconsistent availability and recording of these specific data points within the operational dataset provided.

It is recognized that these omitted variables can significantly influence blast-induced backbreak. For instance, bench height plays a crucial role in determining the burden and spacing design, and variations in bench geometry can alter stress distribution within the rock mass, potentially affecting the extent and pattern of backbreak [48]. Similarly, rock density is a fundamental geomechanical property that dictates the rock's response to explosive energy; variations in density can lead to differential fragmentation and backbreak. Furthermore, the type of explosive used, including its energy content, detonation velocity, and coupling, directly impacts the energy transfer to the rock and the resulting blast performance, including backbreak [49]. Future research could benefit from incorporating these variables to develop more robust and universally applicable backbreak prediction models, provided that comprehensive and consistent data collection for these parameters becomes feasible.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Monjezi, M., Rezaei, M. and Yazdian, A. (2010) Prediction of Backbreak in Open-Pit Blasting Using Fuzzy Set Theory. *Expert Systems with Applications*, **37**, 2637-2643. https://doi.org/10.1016/j.eswa.2009.08.014

[2] Shamsoddin Saeed, M., Jalalifar, H., Shamsoddini, H. and Darbor, M. (2022) A Comparative Study on the Application of Regression-PSO and ANN Methods to Predict Backbreak in Open-Pit Mines. *Analytical and Numerical Methods in Mining Engineering*, **11**, 55-66.

[3] Ghasemi, E., Amnieh, H.B. and Bagherpour, R. (2016) Assessment of Backbreak Due to Blasting Operation in Open Pit Mines: A Case Study. *Environmental Earth Sciences*, **75**, Article No. 552. https://doi.org/10.1007/s12665-016-5354-6

[4] Jing, L. (2003) A Review of Techniques, Advances and Outstanding Issues in Numerical Modelling for Rock Mechanics and Rock Engineering. *International Journal of Rock Mechanics and Mining Sciences*, **40**, 283-353. https://doi.org/10.1016/s1365-1609(03)00013-3

[5] Cunningham, C.V.B. (2005) The Kuz-Ram Fragmentation Model—20 Years on. *Brighton Conference Proceedings*, *European Federation of Explosives Engineers*, Brighton, 13-16 September 2005, 201-210.

[6] Morris, J.P., Rubin, M.B., Block, G.I. and Bonner, M.P. (2006) Simulations of Fracture and Fragmentation of Geologic Materials Using Combined FEM/DEM Analysis. *International Journal of Impact Engineering*, **33**, 463-473. https://doi.org/10.1016/j.ijimpeng.2006.09.006

[7] Monjezi, M. and Dehghani, H. (2008) Evaluation of Effect of Blasting Pattern Parameters on Back Break Using Neural Networks. *International Journal of Rock Mechanics and Mining Sciences*, **45**, 1446-1453. https://doi.org/10.1016/j.ijrmms.2008.02.007

[8] Du, S., Feng, G., Wang, J., Feng, S., Malekian, R. and Li, Z. (2019) A New Machine-Learning Prediction Model for Slope Deformation of an Open-Pit Mine: An Evaluation of Field Data. *Energies*, **12**, Article No. 1288. https://doi.org/10.3390/en12071288

[9] Alamdari, S., Basiri, M., Mousavi, A. and Soofastaei, A. (2022) Application of Machine Learning Techniques to Predict Haul Truck Fuel Consumption in Open-Pit Mines. *Journal of Mining and Environment*, **13**, 69-85.

[10] Li, L., Zhang, R., Sun, J., He, Q., Kong, L. and Liu, X. (2021) Monitoring and Prediction of Dust Concentration in an Open-Pit Mine Using a Deep-Learning Algorithm. *Journal of Environmental Health Science and Engineering*, **19**, 401-414. https://doi.org/10.1007/s40201-021-00613-0

[11] Xue, Y., Bai, C., Qiu, D., Kong, F. and Li, Z. (2020) Predicting Rockburst with Database Using Particle Swarm Optimization and Extreme Learning Machine. *Tunnelling and Underground Space Technology*, **98**, Article ID: 103287. https://doi.org/10.1016/j.tust.2020.103287

[12] Ke, G., *et al.* (2017) LightGBM: A highly efficient Gradient Boosting Decision Tree. *Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 3146-3154.

[13] Liang, W., Luo, S., Zhao, G. and Wu, H. (2020) Predicting Hard Rock Pillar Stability Using GBDT, XGBoost, and LightGBM Algorithms. *Mathematics*, **8**, Article No. 765. https://doi.org/10.3390/math8050765

[14] Li, M. (2024) Comprehensive Review of Backpropagation Neural Networks. *Academic Journal of Science and Technology*, **9**, 150-154. https://doi.org/10.54097/51y16r47

[15] Faramarzi, F., Ebrahimi Farsangi, M.A. and Mansouri, H. (2012) A Res-Based Model for Risk Assessment and Prediction of Backbreak in Bench Blasting. *Rock Mechanics and Rock Engineering*, **46**, 877-887. https://doi.org/10.1007/s00603-012-0298-y

[16] Khandelwal, M. and Monjezi, M. (2012) Prediction of Backbreak in Open-Pit Blasting Operations Using the Machine Learning Method. *Rock Mechanics and Rock Engineering*, **46**, 389-396. https://doi.org/10.1007/s00603-012-0269-3

[17] Russell, R. (2018) Machine Learning: Step-by-Step Guide to Implement Machine Learning Algorithms with Python.

[18] Tatsat, H., Puri, S. and Lookabaugh, B. (2020) From Building Trading Strategies to Robo-Advisors Using Python.

[19] Saied, M., Guirguis, S. and Madbouly, M. (2023) A Comparative Study of Using Boosting-Based Machine Learning Algorithms for IoT Network Intrusion Detection. *International Journal of Computational Intelligence Systems*, **16**, Article No. 177. https://doi.org/10.1007/s44196-023-00355-x

[20] Alcolea, A. and Resano, J. (2021) FPGA Accelerator for Gradient Boosting Decision Trees. *Electronics*, **10**, Article No. 314. https://doi.org/10.3390/electronics10030314

[21] Sai, M.J., Chettri, P., Panigrahi, R., Garg, A., Bhoi, A.K. and Barsocchi, P. (2023) An Ensemble of Light Gradient Boosting Machine and Adaptive Boosting for Prediction of Type-2 Diabetes. *International Journal of Computational Intelligence Systems*, **16**, Article No. 14. https://doi.org/10.1007/s44196-023-00184-y

[22] Chen, T., Xu, J., Ying, H., Chen, X., Feng, R., Fang, X., *et al.* (2019) Prediction of Extubation Failure for Intensive Care Unit Patients Using Light Gradient Boosting Machine. *IEEE Access*, **7**, 150960-150968. https://doi.org/10.1109/access.2019.2946980

[23] Soper, D.S. (2021) Greed Is Good: Rapid Hyperparameter Optimization and Model Selection Using Greedy K-Fold Cross Validation. *Electronics*, **10**, Article No. 1973. https://doi.org/10.3390/electronics10161973

[24] Hecht-Nielsen (1989) Theory of the Backpropagation Neural Network. *International Joint Conference on Neural Networks*, New York, 18-22 June 1989, 593-605. https://doi.org/10.1109/ijcnn.1989.118638

[25] Li, J., Cheng, J., Shi, J. and Huang, F. (2012) Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. In: Jin, D. and Lin, S., Eds., *Advances in Computer Science and Information Engineering*, Springer, 553-558. https://doi.org/10.1007/978-3-642-30223-7_87

[26] Suliman, A. and Zhang, Y. (2015) A Review on Back-Propagation Neural Networks in the Application of Remote Sensing Image Classification. *Journal of Earth Science and Engineering*, **5**, 1. https://doi.org/10.17265/2159-581x/2015.01.004

[27] Asaad, R.R. and Ali, R.I. (2019) Back Propagation Neural Network (BPNN) and Sigmoid Activation Function in Multi-Layer Networks. *Academic Journal of Nawroz University*, **8**, 216-221. https://doi.org/10.25007/ajnu.v8n4a464

[28] Dereich, S. and Jentzen, A. (2024) Convergence Rates for the Adam Optimizer.

[29] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., *et al.* (2020) Graph Neural Networks: A Review of Methods and Applications. *AI Open*, **1**, 57-81. https://doi.org/10.1016/j.aiopen.2021.01.001

[30] Kipf, T.N. and Welling, M. (2017) Semi-Supervised Classification with Graph Convolutional Networks.

[31] Hamilton, W.L. (2020) Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **14**, 1-159.

[32] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S. (2019) A Comprehensive Survey on Graph Neural Networks. *Journal of Latex Class Files*, **20**, 1-22.

[33] Kilani, B.H. (2024) Kolmogorov-Arnold Networks: Key Developments and Uses.

[34] Karlik, B. and Olgac, A.V. (2011) Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. *International Journal of Artificial Intelligence and Expert Systems*, **1**, 111-122.

[35] Liu, Z., *et al.* (2025) KAN: Kolmogorov-Arnold Networks.

[36] Kohavi, R. (1995) A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the* 14*th International Joint Conference on*

Artificial Intelligence, Montreal, 20-25 August 1995, 1137-1145.

[37] Stone, M. (1974) Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society Series B*: *Statistical Methodology*, **36**, 111-133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

[38] Hastie, T., Tibshirani, R. and Friedman, J. (2009) The Elements of Statistical Learning Data Mining, Inference, and Prediction. 2nd Edition, Springer. https://doi.org/10.1007/978-0-387-84858-7

[39] Pedregosa, F., *et al.* (2011) Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830.

[40] Bergstra, J. and Bengio, Y. (2012) Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, **13**, 281-305.

[41] Snoek, J., Larochelle, H. and Adams, R.P. (2012) Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, Vol. 5, 2951-2959.

[42] Cawley, G.C. and Talbot, N.L.C. (2010) On Over-Fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, **11**, 2079-2107.

[43] Friedman, J.H. (2001) Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, **29**, 1189-1232. https://doi.org/10.1214/aos/1013203451

[44] Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. The MIT Press. http://www.deeplearningbook.org/

[45] Hahn, G.J. (1973) The Coefficient of Determination Exposed! *Chemical Technology*, **3**, 609.

[46] Robeson, S.M. and Willmott, C.J. (2023) Decomposition of the Mean Absolute Error (MAE) into Systematic and Unsystematic Components. *PLOS ONE*, **18**, e0279774. https://doi.org/10.1371/journal.pone.0279774

[47] Schluchter, M.D. (2014) Mean Square Error. Wiley.

[48] Prasad, S., Choudhary, B.S. and Mishra, A.K. (2017) Effect of Blast Design Parameters on Blast Induced Rock Fragmentation Size—A Case Study. *The International Conference on Deep Excavation*, *Energy Resources and Production*, IIT Kharagpur, 24-26 January 2017, 1-7.

[49] Dotto, M.S. and Pourrahimian, Y. (2024) The Influence of Explosive and Rock Mass Properties on Blast Damage in a Single-Hole Blasting. *Mining*, **4**, 168-188. https://doi.org/10.3390/mining4010011