Forecasting high energy seismic bump using K-nearest neighbor, Logistic Regression and Decision Tree

Lokesh Aravind Santhavadhana  
*Faculty of Engineering Environmentand Computing*   
*Coventry University* Coventry, England  
santhavadl@uni.coventry.ac.uk

*Abstract*— This study addresses the critical task of forecasting high-energy seismic bumps in underground mines. The paper evaluates the effectiveness of three machine learning techniques: K-nearest neighbor, Logistic Regression, and Decision Tree. The challenge lies in the unbalanced distribution of positive and negative examples, with a scarcity of instances indicating hazardous conditions. By analyzing recorded tremor energies and seism acoustic activity, the objective is to accurately predict increased seismic activity, enabling timely implementation of risk mitigation measures. The findings contribute to the field by assessing the performance of different machine learning algorithms in handling unbalanced data and improving the prediction of high-energy seismic bumps, thus facilitating proactive hazard management, and enhancing safety measures in mining operations.

Keywords— Seismic hazard, mining hazards, underground mines, machine learning, high-energy seismic bumps, K-nearest neighbor, Logistic Regression, Decision Tree, risk mitigation, proactive hazard management.

# Introduction (*forecasting seismic bump*)

Forecasting seismic bumps in mining operations is a crucial task for ensuring the safety of underground mines. Seismic bumps, similar to earthquakes, pose significant risks to the stability of rock masses and the well-being of workers. These mining-induced seismic events, also known as rock bursts, are characterized by the sudden release of accumulated energy within the rock mass, resulting in ground vibrations and potential rock failures.

Seismic hazard prediction in mines is a challenging endeavour, comparable to earthquake prediction in seismology Both seismic events and earthquakes share similar characteristics, such as the occurrence of energy release, the propagation of seismic waves, and the potential for damaging effects [3]. However, predicting seismic hazards in mining settings presents additional complexities due to the dynamic and ever-changing nature of underground conditions.

The detection and prediction of seismic hazards in mining operations have traditionally relied on statistical techniques, which often fall short in accurately forecasting these events. The inherent complexity of seismic processes and the vast disparity between low-energy and high-energy events make it challenging to develop precise and reliable prediction models based solely on statistical approaches[1].

To overcome these limitations, the integration of machine learning methods has shown promise in improving the accuracy of seismic hazard forecasting. By leveraging advanced seismic and seismoacoustic monitoring systems, machine learning algorithms can uncover patterns and relationships within the data to enhance the understanding of rock mass behaviour and develop more effective prediction methods.

In this study, we focus on the task of forecasting high-energy seismic bumps using machine learning techniques, specifically K-nearest neighbour, Logistic Regression, and Decision Tree algorithms [2]. We aim to address the challenge of unbalanced data distribution, where the number of positive examples indicating hazardous conditions is limited compared to the number of negative examples.

The successful prediction of increased seismic activity can enable mining authorities to implement timely risk mitigation measures, such as distressing shooting or evacuating workers from areas at risk of rockbursts. Thus, accurate and reliable forecasting of seismic bumps holds significant practical importance in ensuring the safety of mining operations.

# The data set

First, confirm that you have the correct template for your paper size. The dataset used in this context is sourced from the UCI Machine Learning Repository, a widely recognized and trusted platform for accessing and sharing machine learning datasets. Mining activities are often associated with various dangers known as mining hazards. One significant hazard is seismic activity, which frequently occurs in underground mines. Seismic hazards, similar to earthquakes, are challenging to detect and predict. While advanced monitoring systems have improved our understanding of rock mass processes and the ability to predict seismic hazards, the current methods are far from perfect. The complexity of seismic processes, along with the disproportionate occurrence of low-energy versus high-energy events, makes traditional statistical techniques insufficient for accurate hazard prediction. Therefore, there is a need to explore new opportunities, including the application of machine learning methods, to enhance hazard prediction in mining.

In seismic hazard assessment, data clustering techniques can be utilized, and artificial neural networks have been employed for predicting seismic tremors. However, the results obtained from these methods often yield binary outcomes, classifying events as either "hazardous" or "non-hazardous." The imbalanced distribution of positive (hazardous) and negative (non-hazardous) examples poses a significant challenge in seismic hazard prediction. The existing methods still struggle to achieve satisfactory sensitivity and specificity in their predictions. Factors influencing seismic hazards, such as the occurrence of tremors with high energy levels (> 10^4J), have been proposed in studies. The main objective of seismic hazard prediction methods is to forecast increased seismic activity, which can potentially lead to rockbursts. Such predictions aim to provide timely information about the likelihood of hazardous situations, enabling appropriate measures to be taken, such as risk reduction strategies or evacuating workers from threatened areas.

The presented dataset used for seismic hazard prediction is characterized by an imbalanced distribution of positive and negative examples. Among the attributes in the dataset are evaluations of seismic hazard assessment obtained through seismic and seismoacoustic methods, information about the shift type (coal-getting or preparation), seismic energy recorded in the previous shift, the number of seismic bumps within different energy ranges, and other related parameters. The decision attribute in the dataset indicates whether a high-energy seismic bump occurred in the next shift, defining the state as either "hazardous" or "non-hazardous”. And By enhancing our ability to predict hazardous situations, mining operations can implement appropriate measures to reduce risks and ensure the safety of workers.

1. Attribute table

|  |  |
| --- | --- |
| **Attribute** | **Type** |
| seismic | Categorical |
| seismoacoustic | Categorical |
| shift | Categorical |
| genergy | Numeric |
| gpuls | Numeric |
| gdenergy | Numeric |
| gdpuls | Numeric |
| ghazard | Categorical |
| nbumps | Numeric |
| nbumps2 | Numeric |
| nbumps3 | Numeric |
| nbumps4 | Numeric |
| nbumps5 | Numeric |
| nbumps6 | Numeric |
| nbumps7 | Numeric |
| nbumps89 | Numeric |
| energy | Numeric |
| maxenergy | Numeric |
| class | Categorical |

# Data Preparation

The data is loaded from the 'seismic\_bumps.arff' file using the loadarff function from the arff module. The loaded data is converted into a DataFrame called 'seismic\_bumps' using the pd.DataFrame() function from the pandas library. Categorical variables are handled using one-hot encoding. The pd.get\_dummies() function is used to convert categorical variables into binary dummy variables. The features and the target variable are split into separate variables. The features are stored in the variable 'X', and the target variable is stored in the variable 'y'. Then the feature names are stored in the 'feature\_names' variable, excluding the 'class' columnanywhere in the paper. Do not number text heads-the template will do that for you.

## Categorical Variable Encoding

One-hot encoding is performed on the categorical variables using the pd.get\_dummies() function. It converts each categorical variable into binary dummy variables, representing the presence or absence of each category.The resulting dummy variables are added to the 'seismic\_bumps' DataFrame.The original categorical variables ('seismic', 'seismoacoustic', 'shift', 'ghazard') are dropped from the DataFrame using the drop() function.The target variable ('class') is mapped from binary strings (b'1' and b'0') to integers (1 and 0) using the map() function.

1. categorical encoding

|  |  |  |
| --- | --- | --- |
| class |  | class |
| b'0' | 0 |
| b'0' | 0 |
| b'1' | 1 |
| b'0' | 0 |
| b'1' |  | 1 |
| b'1' |  | 1 |

## Feature importance

* Feature importance is a technique used to determine the relevance or contribution of each feature in a machine learning model. It helps in understanding which features have the most significant impact on the model's predictions. In the provided code, feature importance is calculated using a decision tree classifier.
* The decision tree classifier assigns an importance value to each feature based on how much it contributes to reducing the impurity in the dataset. The higher the importance value, the more influential the feature is in making predictions.

# MACHINE LEARNING CLASSIFICATION TECHNIQUES

## K-NearestNeighbor(K-NN)

K-Nearest Neighbor is a popular classification algorithm that assigns labels to data points based on the majority label of their k nearest neighbors. By calculating distances between the data point and its neighbors, K-NN determines the most common class among them. This non-parametric algorithm doesn't assume any underlying data distribution and is easy to implement. However, its accuracy depends on the choice of k and the availability of a substantial training dataset.

## Logistic Regression

Logistic Regression is a widely used binary classification algorithm that models the relationship between independent variables and a binary outcome. By employing a logistic or sigmoid function, it estimates the probability of the outcome belonging to a particular class. Logistic Regression is interpretable, accommodates both categorical and continuous variables, and offers insights into feature importance. Nonetheless, it assumes a linear relationship between predictors and the log-odds of the outcome.

## Decision Tree

Decision Tree is an intuitive and versatile algorithm that uses a hierarchical structure to make decisions based on input features. It partitions the feature space by values and creates decision rules at internal nodes. Through recursive splitting, it constructs a tree-like structure. Decision Trees are applicable to both classification and regression tasks, handle various types of variables, and capture complex feature interactions. However, they may overfit and be sensitive to minor changes in training data. Techniques like pruning can mitigate these issues.

# experimental setup

The code begins by loading the "seismic\_bumps" dataset from an ARFF file using the loadarff function from the arff module. The data is then converted into a Pandas DataFrame called seismic\_bumps for further processing. The categorical variables in the dataset are handled by one-hot encoding them using the pd.get\_dummies function. The categorical columns ('seismic', 'seismoacoustic', 'shift', and 'ghazard') are dropped from the DataFrame since they have been one-hot encoded. The target variable 'class' is mapped from binary bytes (b'1', b'0') to integers (1, 0).

The next step involves splitting the dataset into training and testing datasets using the train\_test\_split function from the sklearn.model\_selection module. The training dataset comprises 70% of the data, while the testing dataset comprises the remaining 30%.

After the data is split, a logistic regression model is trained on the training dataset. This is done using the LogisticRegression class from the sklearn.linear\_model module. The model is fitted to the training data using the fit method. The performance of the model is evaluated by computing the accuracy on both the training and testing datasets using the score method. Additionally, predictions and prediction probabilities are obtained using the predict and predict\_proba methods, respectively. The classification report and confusion matrix are printed for the training set to assess the model's performance.

The next step involves hyperparameter tuning for logistic regression using grid search. This is done using the GridSearchCV class from the sklearn.model\_selection module. Grid search allows us to explore different combinations of hyperparameters and find the best ones for the model. The parameter grid specifies the hyperparameter values to search over. The best hyperparameters are determined using the best\_params\_ attribute of the grid search object. A new logistic regression model is then created with the best hyperparameters and trained on the training data. The performance of the tuned model is evaluated on the testing dataset by computing the accuracy and printing the classification report.

Moving on, a decision tree classifier is trained using the DecisionTreeClassifier class from the sklearn.tree module. The model is fitted to the training data using the fit method. The decision tree is visualized using the plot\_tree function from the sklearn.tree module. Feature importances are computed using the feature\_importances\_ attribute of the trained model. The classification report and confusion matrix are printed for the testing set to evaluate the performance of the decision tree classifier.

## Feature importance

* Feature importance is a technique used to assign a score or weight to each feature based on its contribution to the prediction. It helps identify the most influential features in a dataset. In the provided code, the feature importance is calculated using the feature\_importances\_ attribute of the trained decision tree model (seismic\_bumps\_DecTree). The resulting importance scores are stored in the importances variable.
* Feature ranking involves sorting the features based on their importance scores. In the code, the importances array is sorted in descending order using the argsort() function, and the indices of the sorted array are stored in the indices variable. This ranking provides insight into the most important features according to the decision tree model.

A picture containing text, screenshot, plot, diagram

Description automatically generated

1. Feature Importance in Decision Tree

The code then prints the feature ranking in descending order, displaying the feature number, name, and its importance score using a loop over the number of features in the dataset.

Additionally, a bar plot is generated to visualize the feature importances. The plot represents the importance scores on the y-axis and the corresponding feature names on the x-axis. The features are presented in descending order of importance.

This analysis aids in understanding which features significantly impact the model's predictions and can be valuable for tasks such as feature selection or feature engineering.

Hyperparameter tuning for the decision tree classifier is performed using grid search, similar to the previous step. The GridSearchCV class is used to explore different combinations of hyperparameters and find the best ones for the decision tree classifier. The best hyperparameters are determined using the best\_params\_ attribute of the grid search object. A new decision tree classifier is then created with the best hyperparameters and trained on the training data. The performance of the tuned model is evaluated on the testing dataset by printing the classification report and confusion matrix.

Finally, a k-Nearest Neighbors (k-NN) classifier is trained using the KNeighborsClassifier class from the sklearn.neighbors module. The model is fitted to the training data using the fit method. The classification report and confusion matrix are printed for the testing set to assess the performance of the k-NN classifier.

# experimental result

## Logistic Regression

The logistic regression model is trained on the training set using default hyperparameters. Accuracy is calculated for both the training and testing sets [4]. The training set is used to generate a classification report and confusion matrix. A grid search is performed to identify optimal hyperparameters for the logistic regression model using cross-validation. The model is then retrained using the tuned hyperparameters, and its performance is evaluated on the testing set. The evaluation includes a classification report, confusion matrix, cross-validation scores, and ROC AUC score. Additionally, an ROC curve is plotted.

A picture containing text, screenshot, square, rectangle

Description automatically generated

1. Confusion Matrix For Logistic RegressionA picture containing text, line, diagram, plot

   Description automatically generated
2. ROC Curve AUR for Logistic Regression

## Decision Tree

The next model utilizes a decision tree classifier. It is trained on the training set with a maximum depth of 10 and entropy as the criterion. The resulting decision tree is visualized, and feature importances are calculated and presented graphically. The trained model is then evaluated on the testing set, generating a classification report and confusion matrix [5]. A grid search is conducted to tune the decision tree's hyperparameters through cross-validation. The best hyperparameters are reported, and the model is retrained accordingly. The performance of the tuned model is assessed on the testing set, including a classification report and confusion matrix. Furthermore, the confusion matrix of the tuned model is plotted.

A picture containing screenshot, rectangle, design

Description automatically generated

1. Decision Tree

A picture containing text, screenshot, square, rectangle

Description automatically generated

1. Confusion Matrix For Decision Tree

A picture containing line, plot, diagram, parallel

Description automatically generated

1. ROC Curve AUR for Decision Tree

## k-Nearest Neighbors (k-NN)

The k-NN classifier is trained on the training set using default hyperparameters. Accuracy is computed for the training set. The model is subsequently evaluated on the testing set, producing a classification report and confusion matrix. A grid search is performed to determine the optimal hyperparameters for the k-NN classifier via cross-validation [6]. The best hyperparameters are reported, and a new instance of the k-NN classifier is created with the tuned hyperparameters. The model is retrained using the tuned hyperparameters and evaluated on the testing set. The evaluation includes a classification report, cross-validation scores, ROC AUC score, and an ROC curve.

A picture containing text, screenshot, square, rectangle

Description automatically generated

1. Confusion Matrix For k-Nearest Neighbors (k-NN)

A picture containing text, line, plot, diagram

Description automatically generated

1. ROC Curve AUR for k-Nearest Neighbors (k-NN)

# conclusion and discussion

the logistic regression model achieved an accuracy of 0.89 on the testing set. After hyperparameter tuning, the accuracy improved to 0.92. Cross-validation scores indicate consistent performance, with an average accuracy of 0.93. However, the ROC AUC score of 0.498 suggests that the model's ability to differentiate between positive and negative instances is relatively weak.

The decision tree model achieved an accuracy of 0.91 on the testing set, which remained unchanged after hyperparameter tuning. Cross-validation scores also indicate consistent performance, with an average accuracy of 0.92. The decision tree model demonstrated better discriminatory ability with an ROC AUC score of 0.67.

1. Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Accuracy after tuned | Cross validation score | ROC AUC Tuned |
| Logistic Regression | 0.89 | 0.92 | 0.93 | 0.498 |
| Decision tree | 0.91 | 0.92 | 0.92 | 0.67 |
| k nearest neighbor | 0.93 | 0.93 | 0.93 | 0.61 |

Among the three models, the k-NN classifier performed the best, achieving an accuracy of 0.93 on both the testing set and after hyperparameter tuning. Cross-validation scores also indicate stable performance, with an average accuracy of 0.93. However, the ROC AUC score of 0.61 suggests that the model's ability to discriminate between positive and negative instances is moderate.

# appendix

Original dataset and python programming files are

available at this link: <https://github.com/lokesharavinds/7072CEM_Machine_Learning>

##### References

1. C. Wang, Evolution, Monitoring and Predicting Models of Rockburst. Springer, 2017.
2. Hans-Günther Hahnekamp, A Simple Seismo-acoustic Monitoring System for Use in Rockburst-prone Zones. Mining Research and Development Establishment, 1982.
3. K. Vandette, “Earthquake similarities complicate the development of early warning systems,” Earth.com, Apr. 09, 2019. https://www.earth.com/news/earthquake-similarities-warning-systems/ (accessed Jun. 14, 2023).
4. “sklearn.linear\_model.LogisticRegression — scikit-learn 0.21.2 documentation,” Scikit-learn.org, 2014. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html
5. scikit learn, “sklearn.tree.DecisionTreeClassifier — scikit-learn 0.22.1 documentation,” Scikit-learn.org, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
6. scikit-learn developers, “sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation,” Scikit-learn.org, 2019. https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html