Tunnelling and Underground Space Technology 152 (2024) 105843

Contents lists available at ScienceDirect

Tunnelling and Underground Space Technology incorporating Trenchless Technology Research

journal homepage: www.elsevier.com/locate/tust

International tunnelling association

Predicting rock type from MWD tunnel data using a reproducible ML-modelling process

Tom F. Hansen [a,b,*], Zhongqiang Liu [a], Jim Torresen [b]

[a] Norwegian Geotechnical Institute, Sandakerveien 140, 0484 Oslo, Norway
[b] University of Oslo, Informatics Institute, Blindern, 0316 Oslo, Norway

ARTICLE INFO

Keywords:
Machine learning
MWD
Tunnelling
Rock type
Reproducibility

ABSTRACT

Despite the increasing global usage of Measure While Drilling (MWD) data in tunnel projects, the application of machine learning (ML) techniques for real-time rock type prediction still needs to be explored. This paper introduces a novel ML approach to predict rock types in advance of the tunnel face using MWD data. Drawing on a diverse dataset of 4986 samples from 15 Norwegian tunnels, this study employed a pipeline including the LightGBM machine learning algorithm to forecast rock types 3–6 m ahead of excavation, achieving a balanced accuracy of 0.96 for six primary rock types using all 48 MWD-features. A more challenging label configuration with ten classes of near similar rock types, aimed to test the outer boundaries of model performance, achieved a score of 0.87. Performance in geologic transition zones is compared to regular zones. Notably, this capability facilitates proactive logistics for excavated rock material reuse and rock engineering strategies. Data leakage and reproducibility challenges in ML-based research are addressed in a step-by-step approach, and comparisons are drawn between digital and conventional scientific experimentation.

1. Introduction

Measure While Drilling (MWD) data, collected automatically by sensors on drilling machines in drill and blast tunnelling projects globally, monitors variations in the rock mass, providing a digital signature of the rockmass for all drilled holes ahead of the tunnel face (Van Eldert et al., 2017). This dataset, alongside face-seismics (Dickmann and Hecht-Méndez, 2022), offers unique spatial detail ahead of the face. Traditionally, MWD data has been used to visualise geological changes, aiding engineers in addressing geological challenges ahead. However, the data's complexity and noisy pattern often make it difficult for humans to interpret and classify the data into concrete measures, such as identifying lithology. Moreover, engineers must interpret this data swiftly to maximise the planning horizon and decision-making options as opportunities to revise operations diminish after blasting.

Currently, tunnelling lacks reliable and efficient methods for accurately predicting rock type in advance, constraining the planning of rock support, excavation design, and the logistical management of excavated rock reuse. Liu and Gan compiled a review of techniques for geological prediction in tunnelling using underground data, not including MWD data (Liu and Gan, 2023). They highlighted the significant limitations of current methods, such as geophysical prospecting and core drilling, which notably impede the excavation process. An industry-applicable, efficient prediction solution that minimally disrupts excavation remains elusive.

A few studies have leveraged MWD data to predict rock types in a quarry, surface mining, and offshore drilling. In Table 1 we have summarised their findings.

Existing research shows promising predictive performance but is limited by a narrow range of rock types, small MWD datasets, and a focus on single holes. Some studies did not detail their machine learning (ML) modelling methods. Given a recent Princeton study (Kapoor and Narayanan, 2023) highlighting data leakage and methodological errors in ML-based science, these omissions introduce uncertainty to the results. Analyses of full-face infrastructure tunnelling with large, diverse datasets are yet to be conducted.

Our hypothesis is that a machine learning (ML) model can be trained to accurately predict rock types in full-face drill and blast tunnel excavation ahead of the face using MWD data from all blasting holes in one round. Our dataset comprises 4986 samples, including around 500,000 drillholes from 15 Norwegian road and railway tunnels, covering ten different rock types. This dataset provides sensor data for a complete

* Corresponding author.
E-mail address: tom.frode.hansen@ngi.no (T.F. Hansen).

https://doi.org/10.1016/j.tust.2024.105843
Received 21 December 2023; Received in revised form 16 April 2024; Accepted 19 May 2024
Available online 10 June 2024

Table 1
Studies leveraging MWD-data and machine learning to predict rock type.

Objective	Best algorithm	Description
Classifying four rock types in an open pit iron mine	LogiBoost	Kadkhodaie-Ilkhchi et al. (2010) (Kadkhodaie-Ilkhchi et al., 2010) applied MWD data to classify four rock types (iron ore zones A and B, banded iron formation, shale) in 28 down-hole drillings, each 12 m deep, at an Australian open-pit iron mine. They utilised 12 MWD parameters, including bit pressure, rotation pressure, and pull-down rate, to identify rock types at each measurement point within the drillholes. Although the exact measurement resolution was not detailed, it is inferred from plots to range between 2–5 cm. The study compared the accuracy of models using Multi-Layer Perceptron (MLP), Logiboost, and a fuzzy logic system, finding Logiboost to be the most effective with a 78.4 % accuracy rate. The research highlighted Logiboost's superiority over the computationally intensive MLP, which is susceptible to overfitting, and the less accurate, subjective fuzzy logic system.
Coal seam detection in open pit mine	Neural network. Multi-layer perceptron (MLP)	Leung and Scheding (2015) (Leung and Scheding, 2015) employed MWD data for automated coal seam detection within single drillholes at an Australian open pit coal mine, analysing data from 35 down-hole drillings. The study did not specify the measurement resolution. They derived a metric, Modulated Specific Energy (SEM), from four MWD parameters: penetration rate, rotary speed, weight on bit, and torque, to identify coal seams. Utilising a neural network, they achieved a 92.2 % accuracy in the binary classification task, with a precision of 72.6 % and a recall of 81 %.
Classifying three rock types in a marble quarry	Hidden Markow Model.	Vezhapparambu et al. (2018) (Vezhapparambu et al., 2018) utilised MWD data to categorise rock types in three single down-hole drillings, ranging from 14 to 18 m, in a Norwegian marble quarry. They employed penetration rate, rotation pressure, and dampening pressure as MWD features to train a Hidden Markov Model (HMM), leveraging the sequential nature of MWD data to identify three rock types: pure marble, intrusions, and fractured marble. An optical televiewer camera was used for detailed lithology labelling. The sensor's sampling resolution was 2 cm. Although standard machine learning metrics were not reported, plots showing the classified rock types along the drill holes suggest a reasonable accuracy.
Binary classifying shale-rich from other rock types in oil/gas exploration	Gradient Boosting	Klyuchnikov et al. (2019) (Klyuchnikov, 2019) analysed MWD data from 27 oil wells at the Novoportovskoye field in Russia, with a 10 cm resolution. They investigated parameters including weight on bit, torque, penetration rate, standpipe pressure, and hook load, along with metrics such as adjusted penetration rate and specific drilling energy. Additionally, they assessed statistical metrics across the boreholes. A baseline model reached 86.5 % accuracy. They then developed a binary machine learning model to differentiate shale-rich from other rock types, testing logistic regression, gradient boosting, feedforward, and LSTM neural networks. Gradient boosting was the most effective, with 91.1 % accuracy.
Binary classifying iron ore units in open pit iron mine	Gaussian process model	Silversides et al. (2022) (Silversides and Melkumyan, 2022) applied MWD data from an Australian open-pit iron mine to binary classify two iron ore units using a Gaussian process model. They utilised penetration rate, force on the bit, and torque, measured every 10 cm. Based on the accepted level of uncertainty, the results varied from 81.4 % to 86.8 %. The study did not specify the number of drillholes analysed.

tunnel round, typically 5.5 m in length. The aim is to train an ML model to predict rock type ahead of the tunnel face using a geologically diverse and large dataset to address noise and calibration issues from sensor data across over 20 different tunnel rigs. We reduce dimensionality and extract information from approximately 25,000 MWD values per blasting round, using six statistical metrics for each of the eight MWD parameters, resulting in a 48-value feature vector for each rock type logged. We employ complex tree-based models, which generally perform best on tabular data and compare them with various models to understand the relationship between MWD data and rock type. In our search for the best-performing models, we experiment with different feature sets, optimise hyperparameters using Bayesian optimisation, and eliminate outliers with advanced techniques.

Recent research has identified data leakage, low reproducibility, and overly optimistic ML models due to methodological errors as significant challenges in ML-based science (Kapoor and Narayanan, 2022). The concept of "responsible AI engineering," as defined by Schneiderman (Shneiderman, 2021) and further explored by Soklaski et al. (Soklaski et al., 2022), is an extension of this discussion. Considering these insights, additional efforts have been made in this study to adhere to a rigorous scientific ML experimentation process for this supervised prediction task using conventional ML algorithms on tabular data.

The following sections describe the dataset and our methodology, detailing a modelling pipeline and a scientifically grounded ML experimentation process. We then present our findings and discuss the implications of our results. Lastly, we summarise the study and provide avenues for future research.

1.1. Dataset

The dataset, detailed with preprocessing in a separate paper (Hansen et al., 2024), comprises MWD data (features) and rock type classifications for 4986 blasting rounds across 15 geologically diversified hard rock tunnels from four Norwegian infrastructure projects: UDK, UNB, RV4, and E39. It includes major rock types such as Precambrian Gneisses, Permian Basalt and Granite, Permian Rhomb porphyry, and Cambro-Silurian shales, limestone, and claystone. Table 2 lists the MWD parameters and their abbreviations. The MWD data, measured at approximately 2 cm intervals in all blast drillholes, is split into 3,739 training and 1,247 test samples, with a split ratio of 0.25, following Hastie et al. (Hastie et al., 2009).

1.2. Preprocessing single holes

Preprocessing of MWD-data varies somewhat across studies, primarily for drill rod coupling effects and energy loss in long holes (over 5–6 m). Our dataset consists solely of short blasting holes drilled with a single rod, simplifying the preprocessing required. The processing steps employed largely align with the processes for blasting (single rod) holes, as described in Eldert et al. (van Eldert et al., 2020). The primary operations were to remove the first 0.5 m of data to exclude non-representative collaring information and eliminate compromised sensor data. Subsequently, we processed the parameters for each hole in two ways, resulting in two sets of parameters (labelled Norm and RMS) through normalisation or root-mean-square filtering.

Table 2
Features of the dataset used for this study. Normalised and RMS-filtered MWD-parameters are given by the abbreviated names.

Original parameter name and unit	Abbreviation for normalised/ filtered form in study	Description
Penetration rate (m/min)	PenetrNorm	Normalised penetration.
Penetration rate (m/min)	PenetrRMS	RMS filtered penetration
Rotation pressure (bar)	RotaPressNorm	Normalised rotation pressure
Rotation pressure (bar)	RotaPressRMS	RMS filtered rotation pressure
Feeder pressure (bar)	FeedPressNorm	Normalised feeder pressure
Hammer pressure (bar)	HammerPressNorm	Normalised hammer pressure
Flushing water flow (l/min)	WaterflowNorm	Normalised waterflow
Flushing water flow (l/min)	WaterFlowRMS	RMS filtered waterflow

- Normalise all values for each drillhole to exclude the effects of drill hole depth. The mean and standard deviation are calculated from the MWD-values for the single drillhole in the normalising process. The values are then used to normalise each value, z, using Eq. (1). z is one MWD value for a single hole, $\bar{\mu}$ is the mean, and σ is the standard deviation. Normalised parameters are calculated for penetration, rotation pressure, feeder pressure, hammer pressure and water flow. Normalising, return values which are centred around origo, with both negative and positive values $\rightarrow x \in R$
- Run a root-mean-square (RMS) filtering process for values from the parameters penetration, rotation pressure, and flushing water flow. An RMS process removes noise and smooths the parameter values. The RMS values for all the values of one drillhole are calculated by applying a moving RMS filter on all values $\{z_i | i \in 1, 2, 3 \cdots, n\}$ as described in Eq. (2). n is the number of values in the drillhole, w is the sliding window size, $z_j =$ is the j-th element of the sequence in the sliding window starting at position i. For each position, i, in the sequence, it takes the mean of the squared values in the window starting at i and ending at i + w-1, and then takes the overall mean of these windowed means. RMS-filtering returns only positive values $\rightarrow x \in R_{\geq 0}$

$$z_{norm} = \frac{z - \bar{\mu}}{\sigma} \tag{1}$$

$$\mathrm{RMS}(z, w) = \sqrt{\frac{1}{n - w + 1} \sum_{i=1}^{n-w+1} \left(\frac{1}{w} \sum_{j=i}^{i+w-1} z_j^2 \right)} \tag{2}$$

This process resulted in the following eight MWD parameters (abbreviated form): PenetrNorm, PenetrRMS, RotaPressNorm, RotaPressRMS, FeedPressNorm, HammerPressNorm, WaterFlowNorm, WaterFlowRMS. The parameters can be divided into dependent and independent (Feeder pressure and Hammer Pressure). Earlier research (Van Eldert et al., 2017) indicates that independent parameters are influenced only by the operator and the drilling rig's control system, not by the rock mass, but later studies have reached other conclusions (Navarro et al., 2018).

1.3. Processing and dimension reduction of all preprocessed and cleaned single holes in a blasting round

For each tunnel, parameter values were merged according to the profile numbers of a blasting round (vertical sections, as shown in Fig. 1) and further subdivided into shorter sections for rounds containing more than one rock type. We computed mean, median, standard deviation, variance, skewness, and kurtosis for each MWD parameter across all drillholes in a round, generating 48 MWD feature values (six statistical metrics across eight MWD parameters). For instance, a standard road tunnel with an 80 m^2 profile, 100 drillholes for full face excavation, a blasting length of 5 m, and 50 sensor values per meter (2 cm resolution) yields a raw data set of 25,000 values per blasting round. Fig. 1 illustrates this process. The computed statistical values were labelled with suffixes Mean, Median, Std, Variance, Skew, and Kurtosis. Exemplified For PenetrNorm, we derived the following metrics: PenetrNormMean, PenetrNormMedian, PenetrNormVariance, PenetrNormStd, PenetrNormSkew, PenetrNormKurtosis. The datasets from all 15 tunnels were then combined into a single dataset. As (Hansen et al., 2024), this combined dataset is well-aligned, with consistent data distributions. We suggest that using statistical metrics to condense the MWD data helps mitigate issues with calibration, erroneous sensor data, and missing holes. Further machine learning (ML) preprocessing steps, including outlier removal and data scaling, before ML algorithm training, are described in section 2.1

Fig. 2 displays the sample count for each label/rock type in the dataset, with the left figure condensing labels into six main rock types from the ten detailed in the right plot. This dual presentation aims to explore two scenarios. For planning purposes, such as reusing excavated rock or assessing stability issues linked to specific rock types, predicting main rock types (as shown in the left plot) is deemed sufficient. The right plot, however, details a more complex label setup by dividing Gneiss, limestone, and shale into their specific tunnel labels, thus presenting a more challenging scenario for training and predicting with ML models due to minor lithological differences, for example, between Granitic Gneiss and Amphibolitic Gneiss, and the expected subtle differences in MWD feature data signatures. Using our study's approach, this setup tests the limits and performance of ML models in predicting rock types from MWD data. Nonetheless, predicting main rock types is usually adequate for actual tunnel operations, making it more pertinent to focus on communicating performance for these types from this study. The performance differences are detailed in the results chapter.

Optimal ML model construction could benefit from using data from individual drillholes for detailed geological information, enabling high-resolution rock type prediction for subsequent blasting. However, our study did not have detailed labels for each hole. We relied on standard tunnelling datasets, mapping lithology from tunnel faces or contours rather than individual holes. Creating a big dataset from single-hole data is resource-intensive, requiring borehole camera inspections for geological mapping. Although successful rock type prediction from single-hole data has been reported in smaller projects (ref. literature review in the introduction section), scaling this approach to larger datasets across multiple tunnels introduces calibration and data quality challenges. Additionally, the cost of assembling comprehensive datasets and the challenge of filtering corrupt or noisy data from single-hole datasets may not justify the potential benefits, possibly leading to inferior results compared to our study's approach.

2. Methodology for experimentation

Data leakage and overly optimistic ML models due to methodological errors and low reproducibility have lately been identified as critical challenges to address in the research of ML applications (Kapoor and Narayanan, 2022) and science in general (Announcement: Reducing our irreproducibility, 2013). By following a carefully crafted modelling process described in Section 2.1 (visualised in Fig. 3) and supported by digital tools to ensure a robust scientific experimentation process in Section 2.2, we attempt to address these challenges. To the authors' knowledge, such a concise and detailed guide to applied ML prototyping for tabular datasets in the context of ML-based science does not exist, and we hope it will inspire other researchers.

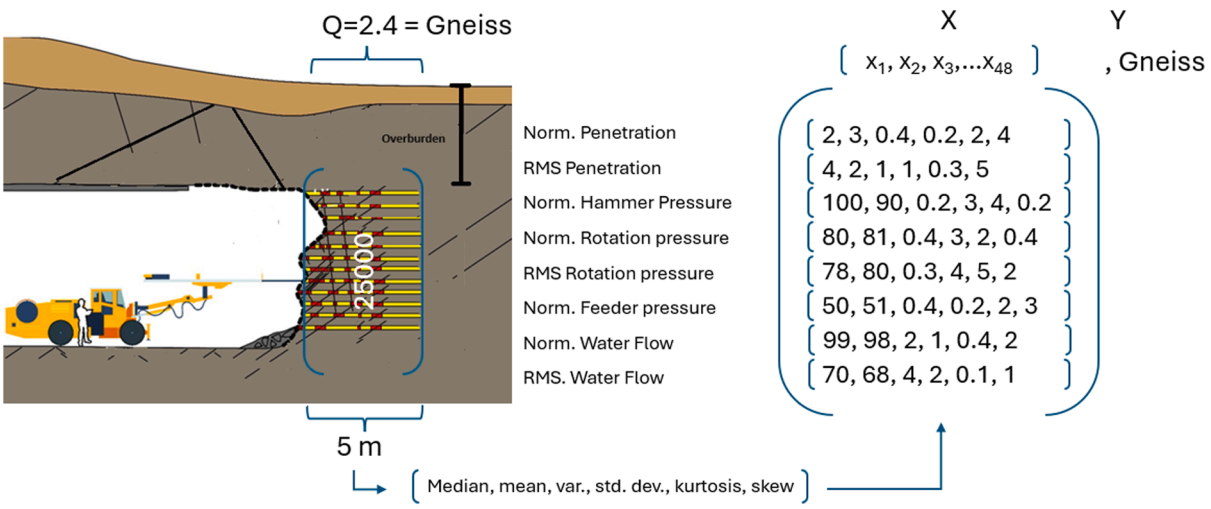


Fig. 1. Illustrating the collection of MWD values for each tunnel blasting round.

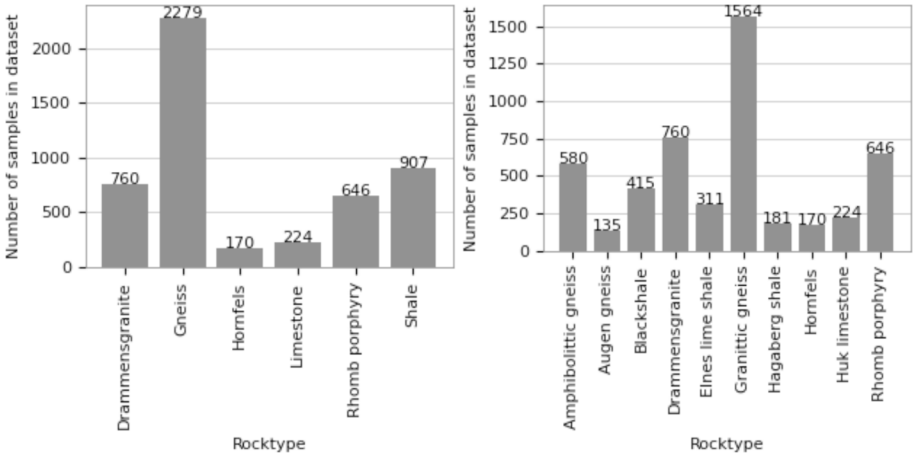


Fig. 2. The number of samples with rock type labels. Higher level grouping to the left.

The process is outlined in a bulleted list in Appendix A. It applies to datasets of up to medium size, with a maximum of around 500,000 samples and 1,000 features. As such, data can be efficiently managed and processed on a single computer, eliminating the need for distributed computational solutions. Our classification model and ML process have been implemented using Python and made openly available as code in the Github repo (see Supplementary material).

2.1. A modelling pipeline preventing data leakage and methodological errors

Our process, initiated after cleaning, NA-infill, and exploratory data analysis (EDA), draws inspiration from Scikit-learn's cross-validation schemes (3.1. Cross-validation: evaluating estimator performance, 2023), best practices in scientific computing (Wilson et al., 2017); (Wilson, et al., 2014) and general ML guidelines (Hastie et al., 2009) (Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2023). Several steps in the procedure, illustrated in Fig. 3, employ popular tools and libraries, proven effective with our data, easing the selection process for applied ML researchers. On a higher level, the process starts with processing and quality controlling the dataset, ending in a split in train and test sets. Then, it continues with feature selection, feature engineering, performance exploration, pipeline, and hyper-parameter optimisation, which should only be done using the training dataset. The process ends with the final evaluation using the untouched testset. It's worth noting that several steps can be performed manually, such as outlier removal based on boxplot inspection, and we advise validating the automated solutions presented. In line with Wilson et al.'s recommendation (Wilson et al., 2017), we prioritised established, well-maintained software libraries, such as Scikit-learn's balanced accuracy metric calculation and data scaling functions (Pedregosa, 2011), over hand-crafted and error-prone solutions.

Our first step, as depicted in step 2 of Fig. 3, is duplicate examination, quality control, and outlier removal. This step is crucial to avoid overlap between training and testing sets and to check that each feature and label conforms to value limits and restricted datatypes using the Pandera package (Bantilan, 2020). For outlier removal, we first removed outliers for single features using Median Absolute Deviation (MAD), which does not assume a normal distribution of the features. Multivariate outlier removal was then done using the Isolation Forest algorithm, both implemented in Python Outlier Detection (PYOD) (Zhao, 2019).

Continuing down the training dataset path, we carried out feature selection from the 48 MWD-features. Feature selection simplifies the model, speeds up the model training and often enhances performance by eliminating noisy or highly correlated features, thereby reducing over-fitting. We employed the techniques implemented in the Featurewiz library (AutoViML/featurewiz, 2023), which initially automate the removal of correlated features using the Sulov method, followed by recursive XGBoost for feature reduction. Featurewiz can also perform automated feature engineering before feature selection, generating

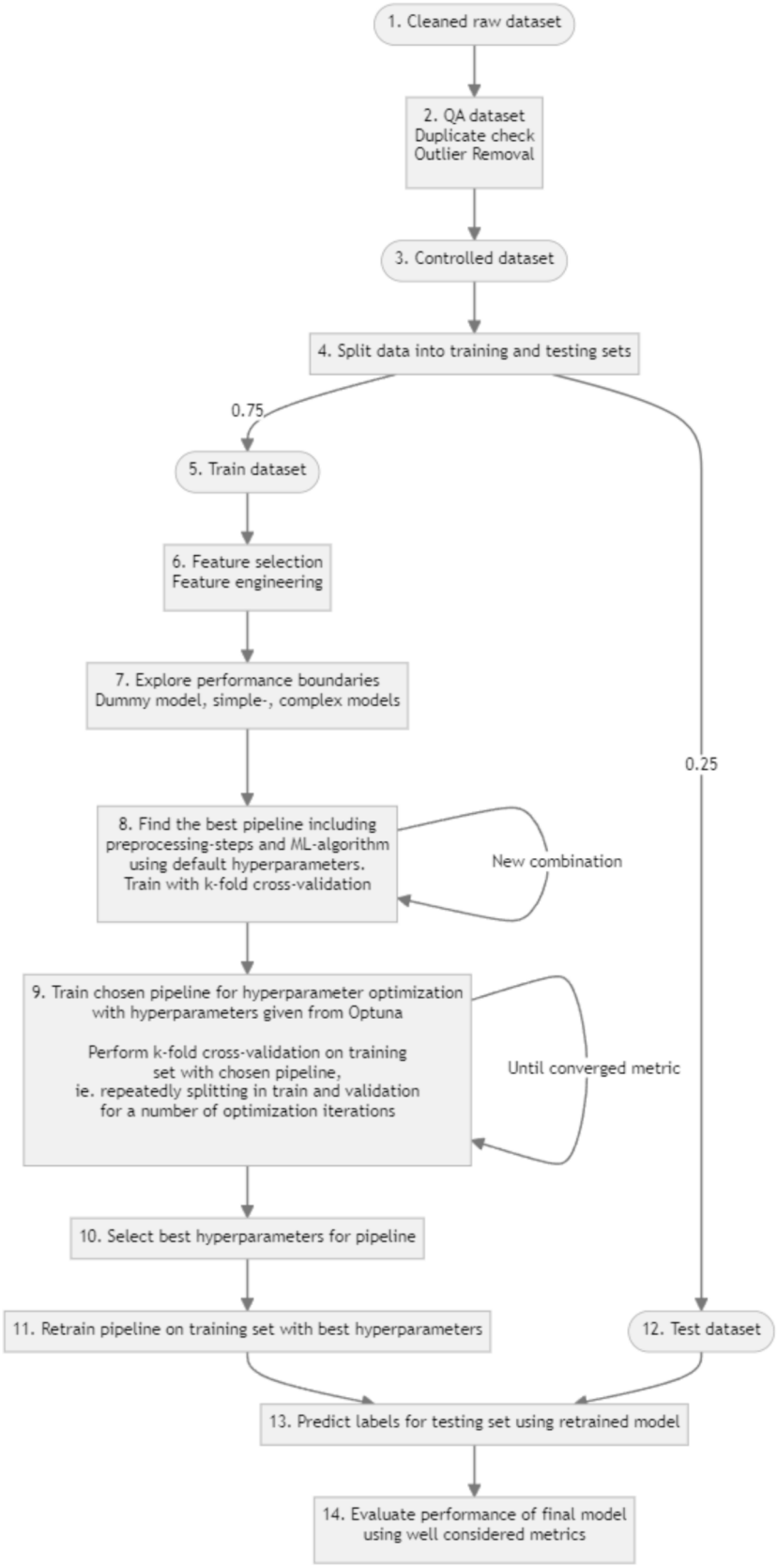


Fig. 3. Flowchart for ML prototyping, addressing data leakage and methodological errors highlighted in recent literature (Klyuchnikov, 2019). Rounded boxes describe the resulting dataset. Rectangular boxes are processes.

hundreds of artificial features, typically through interaction, grouping, or transforming existing ones.

From the feature selection process in step 6, we ended the process with six sets of features to compare in model evaluation:

- An automatically selected set using Featurewiz.
- A set determined by domain knowledge, mainly removing the features for standard deviation and mean, with near similar properties to median and variance.
- A maximum 48 feature dataset, using all features.
- A set with independent MWD-parameters removed (Feeder pressure, Hammer pressure).
- A set with only mean value for all eight MWD-parameters to inspect a minimum set.
- A set with features based only on penetration rate, often found to be the most important feature for rockmass prediction in literature (Van Eldert et al., 2017); (Isheyskiy and Sanchidrián, 2020).

Preprocessing data presents a risk of data leakage from the training to the testing set, mainly through passing information such as the scaler's mean value. To mitigate this, Scikit-learns pipeline functionality structures preprocessing and splitting steps, ensuring transformative steps are only based on the training set. In Fig. 4 (a subgraph of step 8 in Fig. 3), we have specified a generic pipeline, including the typical process steps, before running hyperparameter optimisation. The pipeline should apply to most tabular ML projects but with different content from project to project. Apart from the ML-algorithm, not all steps will be included in every project. This pipeline is used to optimise the model performance on the training set and for the final evaluation on the test set.

Scaling is often critical and is the most common numeric transformation for many ML algorithms' performance, but notably not for tree-based ones, as the data scale doesn't influence the feature-splitting process (Wilson, et al., 2014). Other common transformation steps include one-hot-encoding of categorical values and processing features with a Quantile transformer in mapping to a normal distribution. We have included feature selection and feature engineering in step 6 and this pipeline. In this step, we tested the six sets passed on from the operations in step 6. If necessary, we included feature reduction in the pipeline to reduce the number of features, normally by using Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA). For our project, we are interested in detecting MWD-features of particular importance for the performance and will be reluctant to include feature reduction, which will reduce the interpretability.

Balancing labelled data before fitting various classifiers is vital for predicting less frequent classes. Unbalanced datasets are common, as depicted by the "Gneiss" label in Fig. 2. Balancing a dataset depends on the specific application and nature of the data. In some cases, balancing might introduce more problems than it solves, such as distorting the data's underlying distribution or providing a misleading evaluation of a model's performance (Krawczyk, 2016). For this study, balancing was necessary, achieved through random undersampling of the labels to a certain level, then oversampling with SMOTE (Chawla et al., 2002) to that level (both implemented in the imblearn library)—a two-step approach generally recommended (Chawla et al., 2002). A hyperparameter optimisation process, see step 9 in Fig. 3 (described in a later paragraph), aimed to determine the optimal sampling level for this problem. To find the best pipeline combination for our project, we investigated several alternatives in each step and also not to include the step, e.g., a pipeline without feature reduction.

The resulting pipeline can be viewed as an extended ML algorithm and is passed on to training and testing in the same way as an isolated ML algorithm. In detecting the best pipeline, each pipeline combination was trained with default parameters for each step (e.g., StandardScaler, SMOTE, and RandomForest). In evaluating the performance for each combination, we subsequently and randomly divided the training set into training and validation subsets, using 5-fold cross-validation. We used the StratifiedKFold splitter to preserve the same ratio of class balance in train and validation, thus getting a more reliable result and a robust model. Cross-validation improves the result's reliability and sensitivity to splitting by averaging the performance over five different training and testing splits, thus checking the stability of the dataset.

In the preceding stages, default algorithm parameters were utilised. At step 9, the three highest-performing machine learning (ML) algorithms underwent hyperparameter optimisation. Multiple ML algorithms were selected to ensure the optimisation process did not alter their performance ranking. In this study, these three pipelines demonstrated comparable scores. However, this could vary in other projects, with one or several pipelines excelling, contingent on their performance. Optimisation was done separately for each algorithm using the Optuna library (Akiba et al., 2019). Optuna uses the tree-structured Parzen Estimator (TPE) to explore the parameter space intelligently. We defined a range of values for each hyperparameter and an objective function to evaluate model performance. The TPE sampler proposes new parameters based on a probabilistic model, aiming for better results. Optimisation continues until the chosen performance metric converges, ensuring efficient and effective tuning. In our project, approximately 100 trials were needed for each algorithm to converge to a stable value for the balanced accuracy metric. The combined train/validation dataset was used in a 5-fold cross-validation setup for each model fit during optimisation. To expedite the trial process, thus running more trials, we parallelised Optuna tuning using the Parallel class in Joblib (Joblib: running Python functions as pipeline jobs, 2023), with a shared updated database file to store the results of all previous trials.

Finally, each pipeline's best-performing hyperparameters (step 10) were used to train the model on the full training set and tested on the reserved test set. Steps 9–14 were then repeated for the three best-performing pipelines to ascertain whether hyperparameter tuning affected the performance order. The best performer for steps 8, 9, and 14 was chosen based on the balanced accuracy metric implemented in Scikit-learn (Pedregosa, 2011). Balanced accuracy is a metric calculated as the average of recall obtained on each class, thereby treating all classes equally regardless of their frequency in the data (see Eq. (3). Trained models skewed to accurately predict majority classes will thus be penalised. Recall is a metric measuring the proportion of actual positive cases the model correctly identified. We chose balanced accuracy as the primary guiding metric since the dataset is unbalanced and multiclass, and the cost of false positives and false negatives is similar for all classes, i.e., the classes are equally important. For comparison, we also provide standard accuracy, which has an intuitive understanding but risks being disproportionally influenced by the majority classes.

To increase the robustness in ranging the models, we calculated the AUC-ROC score. The "ROC curve" is a graphical representation of a classifier's performance, plotting the true positive rate (sensitivity, see Eq. (4) against the false positive rate (1-specificity, see Eq. (5) at various classification thresholds. The area under the ROC curve (AUC-ROC) is a standard metric to assess the classifier's overall performance, with higher values indicating better discriminatory power. AUC-ROC is not sensitive to class imbalance, agnostic to the classification threshold level, and is well-established in the literature for comparing different

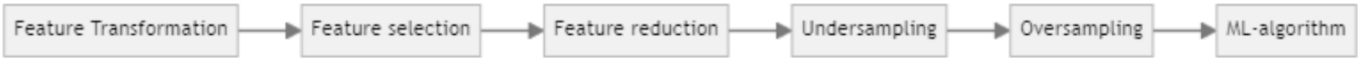


Fig. 4. Scikit-learn-based ML pipeline (utilised in step 8 in Fig. 3).

classifiers (Fawcett, 2006). In conclusion, we consider these three metrics to cover the essential aspects of model evaluation in this study. In addition, we evaluate the single classes individually by visualising a confusion matrix that presents class recall values on the diagonal. For targeted error identification at a class level, we considered precision, recall and F1.

$$\text{Balanced Acc.} = \frac{1}{\text{N (number of classes)}} \sum_{i=1}^{N} \frac{T_{P_i}(\textit{Truepos.forclass}(i)}{T_{P_i} + F_{N_i}(\textit{Falseneg.forclass}(i)} \text{ (Recall)} \tag{3}$$

$$\text{TPR} = \frac{\text{TP (True Positives)}}{\text{TP (True Positives)} + \text{FN (False Negatives)}} \tag{4}$$

$$\text{FPR} = \frac{\text{FP (False Positives)}}{\text{FP (False Positives)} + \text{TN (True Negatives)}} \tag{5}$$

2.2. Ensuring a proper scientific ML experimentation process

While Chapter 3.1 discussed the specifics of the machine learning (ML) process, this chapter considers the broader scientific experimental process that runs concurrently, like a physical experiment, e.g., triaxial testing of rock cores. Crucially, ML-based research should adhere to the scientific method, where a key aspect is the ability to reproduce the results from your experiments (Gauch, 2003). This involves carefully planning and describing the experimental setup and its requirements, allowing other researchers or reviewers to reproduce the results. This builds trust in your findings and facilitates systematically recording all experimental details. The model must be plausible and trustworthy. In Table 3, we have organised the experimental processes and categorised them into objectives. The tools described are those used in our study; however, alternative options are available.

3. Results and Discussion

3.1. Outlier removal and feature selection

Applying the Isolation Forest algorithm for outlier removal resulted in the exclusion of 91 samples from our dataset, utilising an outlier probability confidence of 0.5, informed by domain knowledge and dataset size. The distribution of values was compared pre- and post-removal, alongside the performance of models trained at varying

Table 3
Experimental processes to ensure reproducible ML-based science.

Objectives	Description of process
Code quality	This study aims to verify the hypothesis that rock types can be predicted using a trained machine-learning model applied to a labelled MWD dataset. The code serves as the detailed blueprint for this experiment; therefore, it must be understandable, clean, and well-structured. Code is read more often than it is written. We endeavoured to follow the main principles outlined by Wilson et al. (Wilson et al., 2017); (Wilson, et al., 2014) and Martin (Martin, 2009). We used meaningful variable names, modularised the code, used type annotations in Python to clarify the format of inputs and outputs, and provided documentation for each function. We used the industry standard auto-formatter, Black (Black 23.7.0 documentation, 2023) to increase the code's readability and recognizability. We also set up test functions to detect errors, thus ensuring the quality of our experimentation and illustrating how a function operates.
Version controlling code and dataset	We organised a well-structured project and regularly committed the code using the version control system git to a private GitHub repository (accessible to reviewers), which will be made public upon the paper's acceptance. The dataset (model ready csv-files) was version-tracked using the Data Version Control (DVC) system (Barrak et al., 2021) and quality-controlled while input-reading through Pandera (Bantilan, 2020) and shared on the scientific platform Zenodo (European Organization For Nuclear Research and OpenAIRE, 2013).
Controlling programming environment	We leveraged Poetry (Poetry - Python dependency management and packaging made easy, 2023), an environment and package handling system, to manage dependencies. Poetry automatically generates a lock file describing all packages and their corresponding versions. The Python version used in this project was specified in a.*python-version* file and managed using the Pyenv tool (pyenv/pyenv: Simple Python version management, 2023), simplifying the process of downloading and switching between Python versions. Notable differences exist between Python versions, such as Python 3.9 (introduction of match-case statements), Python 3.10 (advanced type annotations), and Python 3.11 (performance enhancements), underscoring the importance of this process.
Configuration of parameters	Configuration values were controlled and type-parsed with Pydantic (and enhanced with tab completion) (Barrak et al., 2021). Experiment results were thoroughly organised and saved for all experiments using the Hydra (Yadan, 2019) and MLflow (Chen, 2020) systems. Hydra helps avoid the pitfall of embedding "magic numbers" within the code and enables swift experimentation prototyping from the terminal.
Experiment tracking	MLflow gives you an overview and a log of all experiments with results in a clean web-based view. Each model step (finding pipeline, hyperparameter optimisation, evaluation, final training) was grouped in experiments in Mlflow for easy comparison of experiment performance.
Orchestrating experiments	In our research, we integrated Hydra with GNU Make (GNU Project, 2023), a widely used automation tool, to execute scientific experiments efficiently. Hydra manages and organises diverse experiment configurations, enabling flexible and scalable setups. GNU Make, encapsulated in a Makefile, orchestrates these experiments, ensuring reproducibility and efficiency. This methodology not only streamlines the experiment process but also facilitates ease of replication for other researchers, embodying the principles of open and reproducible science.
Control randomness	Seed values were established to control the randomness in data splitting and algorithms. Unless clearly stated, seeding has been used in all experiments to be able to compare results. However, seeding was regularly turned off to explore the spread of results.
Control operating system and hardware	Employing all the processes mentioned above enables the reproducibility of our research results in nearly all instances, provided the same dataset is used. However, an exception could be made when the hardware controlling software, such as CPU and GPU drivers, used to run the computations varies. This is particularly important when executing hyperparameter optimisation for demanding algorithms, like tree-boosting algorithms or neural networks. Another factor to consider is the operating system used and computational speed. Over time, the same code might exhibit slight differences in behaviour when executed on different systems, like Windows or Linux, or even when run on different versions of the same operating system. We created a Docker (Docker: Accelerated Container Application Development, 2023) image to address this issue and ran the optimisation and training within a Docker container. This approach fixed the software and hardware settings, ensuring that our experimental run is reproducible now and in the foreseeable future.
Run more experiments	Running the training using a Docker container simplifies the process of running the experiment distributed on a High-Performance Computer (HPC). It eliminates the need to clone the code and set up the environment on the remote HPC machine. Running experiments fast makes it possible to investigate more model adjustments, often leading to better performance. By leveraging containerisation, we can ensure a consistent environment across different machines, thus bolstering the reproducibility and robustness of our research results.

Table 4
Performance of models trained and evaluated using different outlier removal techniques using the LightGBM algorithm. Sample numbers are numbers after outlier removal. The original dataset had 4986 samples.

Outlier removal technique	Num. samples removed	Balanced accuracy
Multivariate outlier removal with a threshold of 0.5	91	0.963
Univariate outlier removal with hard numbers	378	0.951
Univariate outlier removal with MAD of 3.5	1713	0.949
Univariate and multivariate outlier removal	1785	0.947
No outlier removal	0	0.923

confidence levels. Lower confidence levels excessively eliminated samples from the lower rock type classes, while higher levels scarcely removed samples despite conspicuous outliers in some feature distributions. Table 4 lists the performance of models trained with various outlier removal strategies, showing the superior performance of only utilising multivariate removal. Univariate outlier removal using median absolute deviation (MAD) and defining hard-coded boundaries based on the inspection of boxplots were trialled. However, it was observed that outliers identified for a specific feature did not consistently correspond with the same sample across other features. This could result in the unwarranted removal of samples based on an insignificant feature. Consequently, we opted for multivariate feature removal due to the intricate interrelationships between features and labels.

Feature selection, automated through Featurewiz, reduced the feature set from 48 to 18, applying a correlation threshold between features of 0.6, as determined through experimentation (see Appendix D for a list of features). At least two parameters were selected from each MWD parameter, primarily an average metric (mean or median) and a distribution shape metric (variance, std., skewness, kurtosis). This selection presumably then majorly describes the distribution of the MWD-values involved, resulting in a sufficiently detailed MWD signature to predict the rock type, with only a 1 % drop in performance (see Table 5). Additional attempts were made to increase performance by generating new features from existing features in simple interaction multiplications, but these efforts did not improve our results. Table 5 compares the performance of a LightGBM model trained on different feature sets. For the four first feature combinations in both datasets, the difference is 1 %, which can be regarded as marginal, supported by similar AUC-ROC scores. Performance drops significantly when using fewer features, such as only mean features or those solely based on penetration.

Implications of these differences are significant in practical applications. For instance, using a test set of 1253 samples, the difference in accuracy (0.95 vs. 0.94) results in 63 and 76 false predictions, respectively. The model's utility is particularly notable in predicting the correct rock type in transition zones between geological features. In an experiment detailed in Chapter 4.3, we marked all samples with a different kind of rock in one of the following two samples as transition zones. This led to 670 transition zones in the training set and 234 in the testing set. Given that most false predictions occur in these transition zones (as detailed in Chapter 4.3), the 13 additional false predictions are likely concentrated among these 234 samples.

For the performance summarised in Chapter 4.2, we used a feature set with all features. However, considering the low performance difference, one could argue for using a feature set based solely on dependent MWD features, which is more commonly supported in the literature (Van Eldert et al., 2017). This approach would streamline the model and potentially make it more applicable and more straightforward to interpret within the context of existing research.

3.2. Model pipeline performance

Three superior pipelines emerged from the pipeline optimisation process. All of them utilised tree-based models –CatBoost (Prokhorenkova et al., 2023), LightGBM (Ke, et al., 2017), and XGBoost (Chen and Guestrin, 2016). Each pipeline consisted of the same elements: no PCA/LDA feature reduction, no data scaling, random undersampling to the number of samples in the second most prevalent class, and subsequent oversampling to this level using SMOTE. Hyperparameter optimisation (converging at approximately 100 trials) reordered the models' performance ranking, shifting LightGBM and CatBoost. The process boosted the models' performance, measured in balanced accuracy, from 0.940 to 0.949, 0.943 to 0.948, and 0.935 to 0.947. See detailed performance in Table 6. Fig. 5 depicts the optimisation trial results for the LightGBM pipeline, indicating optimal value attainment after roughly 40 trials. Appendix B presents a parallel coordinate plot summarising the distinct algorithm parameter combinations that led to the best performance. Appendix C lists the optimised parameters. The plot shows a distinct band of parameter values, which led to the best performance.

The final entries in Table 6 compare the performance of evaluating models trained on the entire training dataset instead of a smaller train set when training on splits using cross-validation in the optimisation process. LightGBM stands out with a balanced accuracy of 0.96. However, other tree-based models also show comparable and similar AUC-ROC scores, indicating robust performance across this model category. For broader context, Table 7 includes performances from various other models. A complex and computationally intensive Neural Network model achieves 0.946, demonstrating high effectiveness albeit with greater resource demands. The Logistic Regression model, known for its

Table 5
Performance metrics on the test set for the LightGBM model were trained on different feature sets for the six-class and the ten-class datasets.

Featureset	Num features	Accuracy	Balanced accuracy	AUC-ROC
Six main rock types:				
all features	48	0.970	0.950	0.997
domain features	32	0.970	0.948	0.997
automated features	18	0.960	0.940	0.997
all dependent features	36	0.962	0.939	0.997
only mean	8	0.940	0.917	0.993
only penetration	12	0.876	0.850	0.977
Ten more similar rocktypes:				
all features	48	0.870	0.852	0.987
all dependent features	36	0.862	0.850	0.985
domain features	32	0.860	0.837	0.985
automated features	18	0.860	0.837	0.986
only mean	8	0.821	0.802	0.980
only penetration	12	0.677	0.694	0.950

Table 6
Performance metrics for best-performing algorithms. Values from models trained on the dataset with six main rocktypes.

Metrics	Cat Boost	LightGBM	XGBoost
Cross-validation using train dataset - pipeline detection in step 8 in Fig. 3			
Balanced accuracy	94.3	94.0	93.5
Accuracy	96.4	96.3	94.3
AUC-ROC	0.996	0.995	0.978
Cross-validation using train dataset - hyperparameter optimisation in step 9			
Balanced accuracy	94.8	94.9	94.7
Accuracy	96.8	96.9	95.8
AUC-ROC	0.996	0.997	0.981
On testing data – step 14			
Balanced accuracy	95.6	96.0	95.4
Accuracy	96.3	96.5	96.3
AUC-ROC	0.998	0.998	0.997

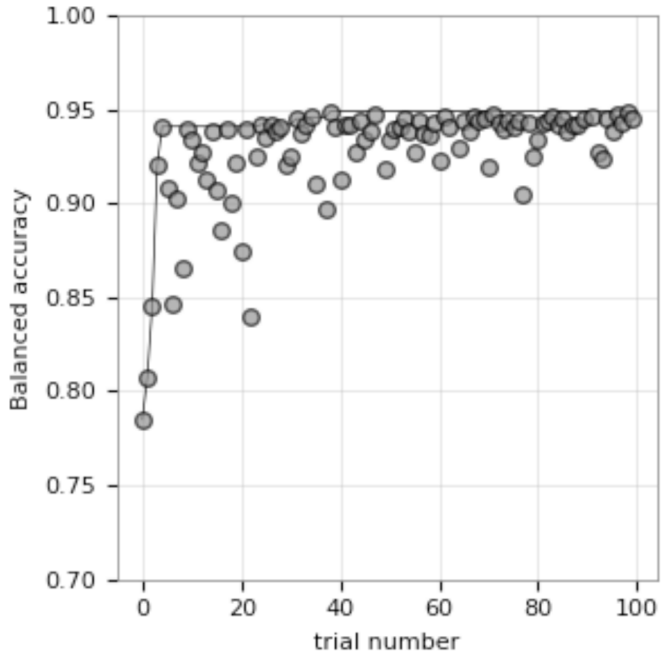


Fig. 5. History of optimisation trials for optimising a pipeline with the LightGBM algorithm.

interpretability, attains a score of 0.92, making it a viable option when model clarity is prioritised. In contrast, while interpretable, a Decision Tree model scores only 0.826, indicating less effectiveness in this context. Additionally, a dummy model predicting only the majority class was included to establish the baseline for minimal model utility. This comparative analysis helps understand the trade-offs between model complexity, interpretability, and performance in the context of this study.

Table 8 presents the final performance metrics evaluated on the test set using the ten-class dataset. Consistent with the six-class dataset, the LightGBM model exhibits the highest performance, with the other two tree-based models close in performance. Additionally, the table includes the Extra Trees model, a more constrained variant of the Random Forest model, which is often effective with unbalanced datasets. The Extra Trees model stands out as a particularly viable option when computational efficiency is a key consideration, balancing performance and resource demands.

The confusion matrix in Fig. 6 reveals variations in predictive performance among different rock types. The diagonal of the confusion matrix represents the normalised values of correctly predicted instances for each class, known as the recall values. Recall measures the model's ability to correctly identify all relevant instances of a given class. High prediction scores are evident for specific types such as Rhomb Porphyry, Hornfels, and Drammensgranite. Despite their similarities, the model effectively distinguishes between three varieties of Gneisses and shales. However, some prediction leakage occurs within these more similar rock types but not between distinct rock groups. This absence of cross-group prediction errors is encouraging, indicating the model's effectiveness in classifying rock types based on the MWD signature data collected.

Fig. 8 displays a confusion matrix for six primary rock types, illustrating enhanced performance with broader label categorisation. Here, various limestones, shales, and Gneisses are consolidated into groupings. The right side of the plot indicates the sample count per rock type, confirming that low sample sizes do not skew the improved results. This visualisation also highlights the minimal number of samples incorrectly classified into other categories, demonstrating the model's accuracy in broader categorisations.

The MWD signatures of more similar rock types, like Blackshale and Hagaberg shale, are expected to be more alike than those of broader categories, such as Shale and Limestone, making them harder to differentiate when training a machine learning (ML) model, leading to increased prediction errors. This could account for the lower performance observed for similar rocks like Amphibolitic Gneiss, Augen Gneiss, Hagaberg Shale, and Blackshale in Fig. 6, as opposed to more easily separable MWD signatures of broader rock types in Fig. 8. To explore this further, we compared the cumulative probability distributions (CDF) of the MWD feature PenetrNormMedian across detailed and combined datasets for each rock type. Fig. 7 shows that the distributions for similar rock types, notably Blackshale and Hagaberg shale, overlap more than those for broader categories, indicating the explanation as mentioned above. This analysis focuses on a single feature, yet the overall impact of multiple features is likely more significant.

3.3. Factors in the dataset affecting model performance

This section explores the impact of dataset characteristics, besides feature selection, on the performance of our predictive model. We specifically focus on data section length (how many values are utilised in building the MWD signature) and geologic transition zones.

3.3.1. Data section length

In addition to inherent model errors, labelling inaccuracies for non-sedimentary rocks may contribute to errors. Labels and MWD-data sectioning in these tunnels are aligned with blasting rounds, potentially including samples with mixed rock types, particularly in

Table 7
Performance metrics on the test set for selected algorithms are trained on the dataset with six main rock types.

Algorithm	Accuracy	Balanced accuracy	AUC-ROC
MLPClassifier	0.961	0.946	0.995
ExtraTreesClassifier	0.941	0.934	0.996
KNeighborsClassifier	0.945	0.929	0.971
LogisticRegression	0.927	0.921	0.992
DecisionTreeClassifier	0.837	0.826	0.969
DummyClassifier	0.493	0.167	0.500

Table 8
Performance metrics on the test set for the best-performing algorithms are trained on the dataset with ten more similar rock types.

Algorithm	Accuracy	Balanced accuracy	AUC-ROC
LGBMClassifier	0.877	0.872	0.989
ExtraTreesClassifier	0.835	0.870	0.990
CatBoostClassifier	0.860	0.862	0.989
XGBoost	0.858	0.861	0.989

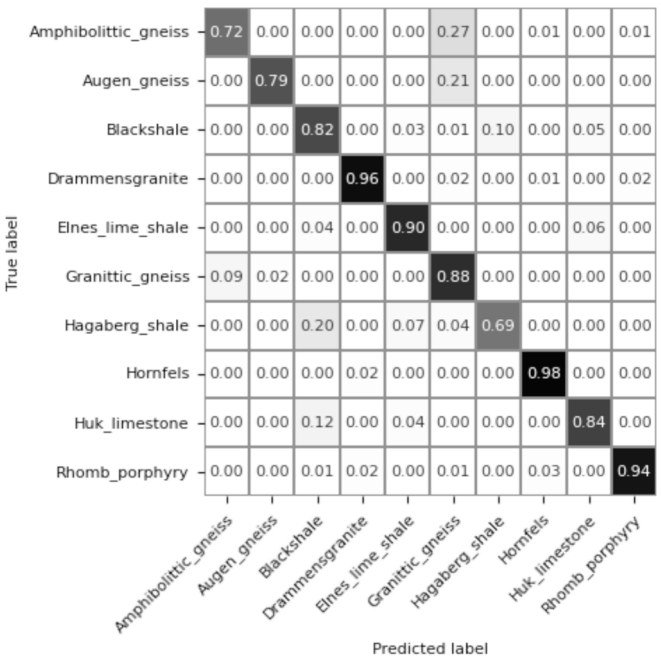


Fig. 6. Confusion matrix for best ML pipeline with the LightGBM classifier, ten-class version.

geological transition zones. These zones often have blurred rather than sharp, vertical boundaries, complicating label accuracy. Conversely, sedimentary rock labels in this dataset are derived from detailed core logging at the tunnel face, likely resulting in more accurate labels. However, the one-meter-long sample lengths introduce uncertainty, increasing the chance of contaminating MWD data with data from different rock types. Additionally, in sedimentary rocks, where labels are based on core logging and possibly adjusted post-blasting, vertical variations in rock types across the tunnel profile might be more prevalent due to horizontal benching. These factors collectively influence our predictive model's uncertainty and potential errors for different rock types.

Fig. 9 illustrates the results of an experiment designed to assess the impact of sample length on predictive performance. This experiment addresses data "pollution" and investigates how the number of parameter values per sample and the influence of MWD data in zones between consecutive blasting rounds affect model accuracy. A key point to consider is that an excavated tunnel face is typically irregular, leading to zones with fewer data points, particularly when 0.5 m of data at the start of the hole were removed in preprocessing. This removal could also reflect the drilling process's collaring phase. The Gran tunnel, characterised by its shorter one-meter sample lengths, also contains a higher frequency of geological transition zones. These zones present greater predictive challenges compared to more uniform sections, potentially impacting the model's overall performance.

From our evaluated test set, we calculated balanced accuracy for various sample lengths. To assess the influence of sample size on our results, we plotted the number of samples at each length. We observed that data points with fewer samples are less reliable. In multiple iterations of the experiment, results were variable for sample counts below 50, rendering the accuracy for lengths of 3, 4, 5.5, 7, and 8 m more uncertain. Sample lengths of 5 and 6 m yielded nearly identical results, with a slight decrease in performance for 1-meter samples. This decrease for 1-meter samples, compared to 5 and 6 m, could be attributed to factors discussed earlier, although the difference is not substantial. Our findings suggest that reducing the number of parameter values in the MWD signature does not significantly impact results at this level. However, determining the lower boundary for further reductions, such as to the minimum level of one drillhole, remains an area for future research.

3.3.2. *Geologic transition zones*

In an experiment examining the prevalence of false predictions in specific samples or zones, we focused on the performance of samples in geological transition zones within the ten-class dataset. Initially, we assessed samples from regular zones, observing a performance increase to 0.884 (and 0.968 for a six-class dataset) compared to 0.852 (0.95 for six-class) for the entire test set (refer to Table 8 for metrics). When analysing samples exclusively in transition zones, performance dropped to 0.73 (0.897 for six-class), significantly lower than the complete test set's performance. This confirms that transition zones are a significant source of false predictions. However, labelling these predictions as entirely incorrect is arguable. In transition zones, where multiple rock types may be present in a single MWD signature, a more accurate label could be a combination, such as 'mix_limestone_shale'. Examining the confidence scores for these predictions might reveal instances where the model detected multiple rock types.

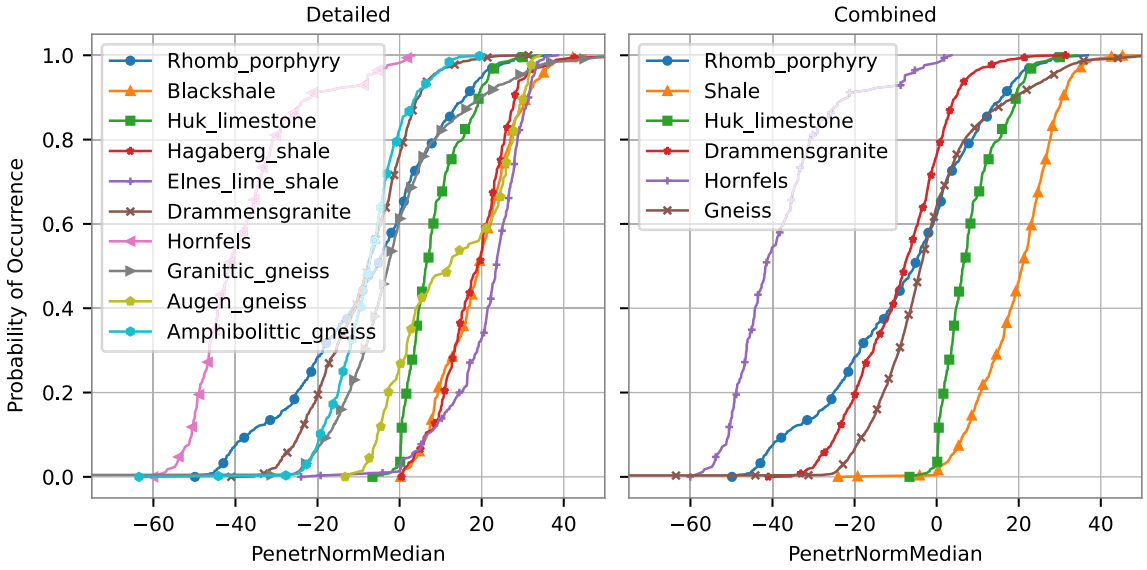


Fig. 7. Comparison between cumulative probability distributions (CDF) for the detailed dataset and the more combined dataset.

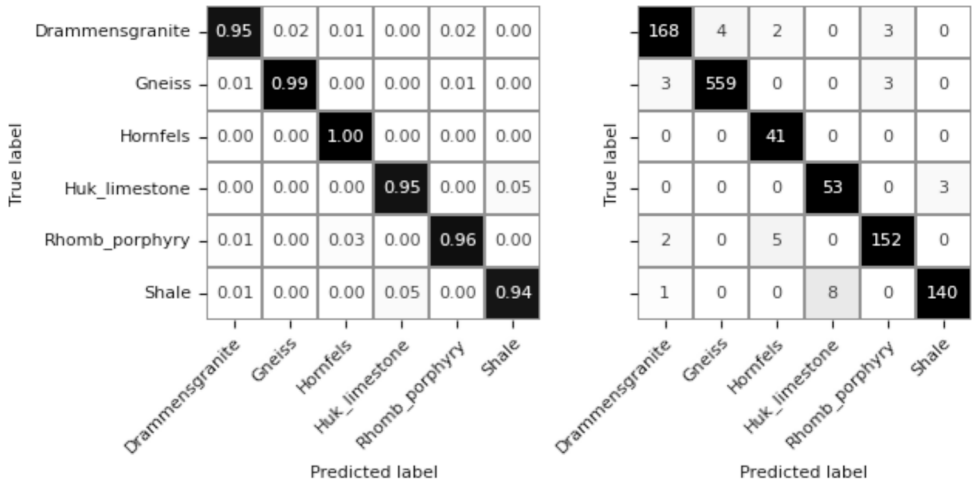


Fig. 8. Confusion matrix from the best-performing ML pipeline with the Light GBM classifier, 6-class version. Left: normalised version. Right: number of samples in each class.

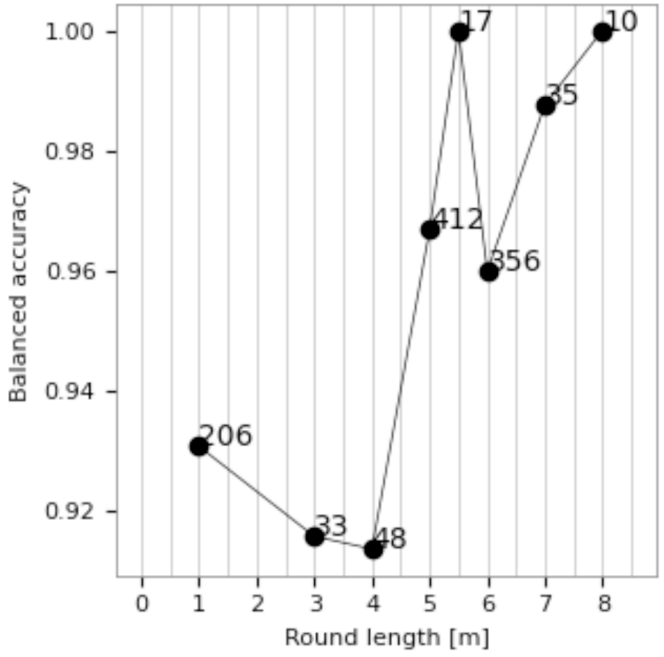


Fig. 9. Visualising balanced accuracy vs. MWD signature length for the six-class dataset. The number of samples for each length in the testset is above the point.

Another aspect to consider in transition zones is the potential variation in rock mechanical properties within a single rock type. Phenomena such as the 'baking' effect from intrusive rocks or weathering in contact zones could influence the MWD signature, adding complexity to the prediction process in these areas.

Compared to transition zones, the impact on performance in regular zones is asymmetrical, primarily due to the higher sample count in regular zones. Fig. 10 presents a confusion matrix for these zones, highlighting distinct patterns. Notably, Hornfels and Rhomb Porphyry show no errors, with Augen Gneiss also demonstrating low error rates. This indicates that, albeit low, errors for these distinct rock types predominantly occur in regular zones (as per Fig. 6). Therefore, when these rock types are predicted in transition zones, the predictions are mostly accurate, signifying high precision. The precision score in machine learning is a metric that quantifies the accuracy of the positive predictions.

The confusion matrix in Fig. 10 confirms that errors are more common among similar Gneisses and sedimentary rocks, with this trend now more pronounced. The higher error rate in transition zones for these rock types suggests that the model's difficulty distinguishing between closely related or overlapping geological characteristics is exacerbated in areas with more complex geological compositions, possibly due to more noise in the MWD-data in these zones.

Fig. 11 compares the model's performance within the six-class dataset, specifically contrasting regular zones (right plot) against transition zones (left plot). The model exhibits near-perfect accuracy in regular zones, while performance notably drops for some rock types in transition zones, particularly for limestone. The performance in the left plot underscores a significant correlation between the MWD signature and the specific rock type.

The primary factor diminishing performance appears to be the presence of multiple rock types within a single MWD signature. This phenomenon is especially evident in geological transition zones, where

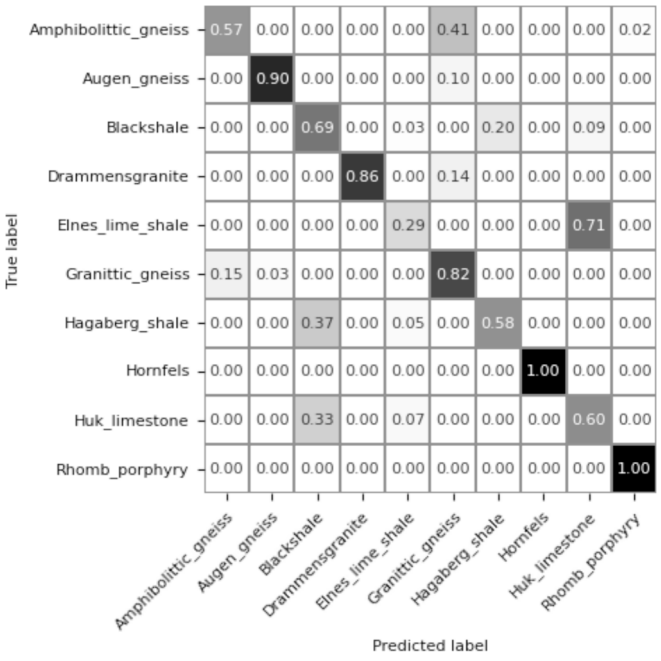


Fig. 10. A confusion matrix showing performance only for prediction in transition zones.

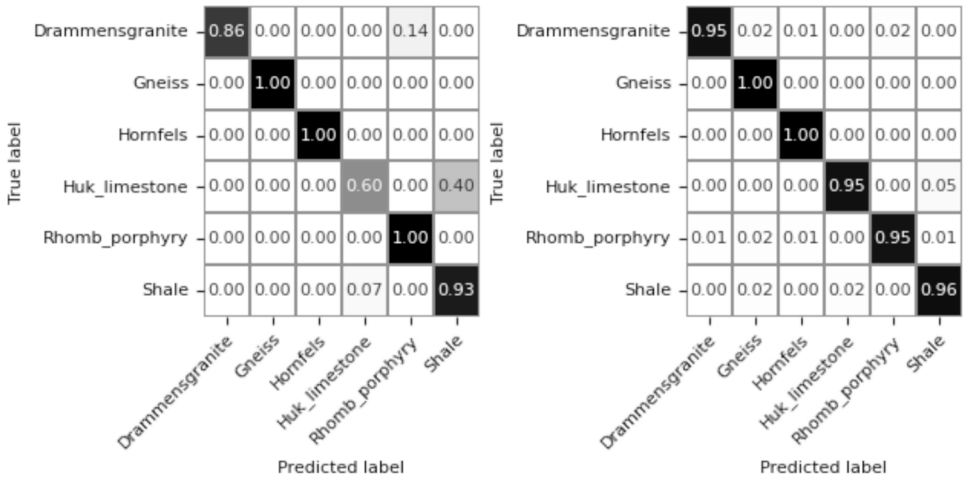


Fig. 11. The confusion matrix for the six-class dataset was evaluated only in regular zones (right) and exclusively in geologic transition zones (left).

the blending of different rock types in close proximity leads to complex MWD signatures. Such scenarios challenge the model's ability to accurately identify a singular rock type, reflecting the intricacies of geological variation and its impact on predictive modelling.

3.4. Shortcomings of the study

Some potential shortcomings of the study include:

- Limited representation of rock types across projects: The dataset features only one rock type, Rhomb porphyry, represented in two projects. This means it was logged by different geologists and likely involved data from different MWD sensors. If there are notable variations in MWD data distributions for the same rock type across projects, calibration discrepancies could adversely affect model performance. Fig. 12, however, shows that distributions for parameters like normalised penetration and feeder pressure are consistent across different tunnels and projects, and the model's performance in detecting Rhomb porphyry was convincing at 0.94. Despite this, calibration remains a potential issue for identifying other rock types, suggesting that future dataset expansions should include thorough checks of rock type distributions.
- Impact of independent parameters on model performance: The importance of independent parameters, such as Feeder Pressure and Hammer Pressure, is another consideration. Our analysis also included models based solely on dependent parameters (see performance in Table 5), resulting in a slight decrease in performance (from 0.95 to 0.94 for the six-class dataset and from 0.852 to 0.85 for the ten-class dataset). This marginal drop suggests that while independent parameters play a role, the model performs well with dependent parameters.
- In geological environments with rapid changes, the method faces challenges detecting anything beyond vertical boundaries between rock types. In our study, such areas often result in uncertain predictions, indicated by probabilities being split across two or more rock types. To address this, one approach could be to train on MWD signatures of entire blasting rounds for sections with a specific label of only one rock type, and then predictions could be made on smaller, more segmented sections of MWD signatures. Alternatively, an effort could be made to split the MWD signatures in line with the intricate geological variations observed. This would require a more detailed understanding and representation of the geological shifts within the dataset, potentially improving the model's accuracy in areas with rapidly changing geological features.

4. Conclusion and outlook

Our study demonstrates the ability to predict rock types from MWD data gathered as 48 ingested statistical values from all MWD-values in every blasting hole in an excavation round. Using MWD data from 3-6 m long blasting drill holes, we forecast rock type before blasting, aiding real-time logistical and rock engineering decisions. Using a LightGBM algorithm model pipeline trained on a dataset from 15 hard rock tunnels, we achieved a balanced accuracy of 0.87 for a complex ten-class dataset with similar rock types and 0.96 for the six primary classes. Training models on two label configurations aimed to assess model performance

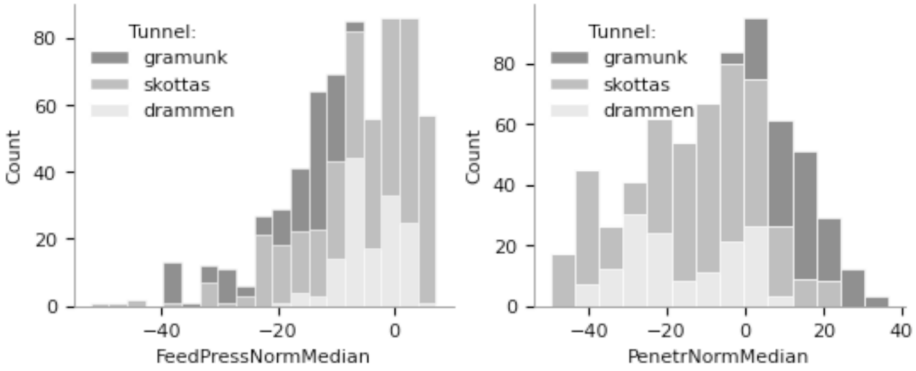


Fig. 12. Stacked histograms for the two MWD-parameters FeederPressNormMedian and PenetrNormMedian for the rock type Rhombporphyry in different tunnel projects (Skottås and Gråmunk are in the same project).

limits with the ten-class dataset, whereas the six-class dataset performance is more aligned with real-world tunnelling decision support needs, where detailed differentiation is not crucial. A detailed label configuration can be applied for more detailed needs, such as identifying Gneisses with sulphidic minerals, albeit with reduced performance for similar rock types. This presents a trade-off between label detail and model performance. Introducing uncertainty and confidence metrics could mitigate lower performance, though this was not explored further. Expanding the models to include more rock types could maintain the balance between detailed and higher-performance models.

Our closer examination of errors revealed that significant inaccuracies for sedimentary rocks and similar types occur in geological transition zones. In contrast, fewer errors for distinct types like Rhomb Porphyry, Hornfels, and Augen Gneiss arise in regular zones. These errors may stem from uncertainty in human-labelled data, including mislabelled sections and sections containing multiple rock types, affecting performance in transition zones. Our analysis also indicates that reducing the number of values in the MWD signature by segmenting the MWD data into shorter sections (down to one meter) does not notably affect the results at this stage. Identifying the minimal data requirement, potentially down to a single drill hole's level, remains a topic for future investigation. Further research should also focus on geological transition zones to enhance prediction accuracy in these zones.

Considering the challenge of data leakage and low reproducibility in ML-based research, our study presents a structured process addressing these concerns, drawing parallels between digital and physical scientific experimentation. To foster community-driven research, our code and methodologies are shared transparently on GitHub, detailing every configuration for researchers aiming for reproducibility with similar datasets.

It's essential to augment the dataset considerably with MWD data from several tunnel drilling rigs and more rock types to advance to a production-grade model capable of detecting a wide range of rock types. One approach could be to comprehensively capture geological regions, ensuring every construction project within a region is represented. Alternatively, the model can be fine-tuned with samples from the initial rounds of a newly encountered rock type. This ability needs further investigation, especially in the number of necessary samples.

A similar model can be constructed using data from long exploratory holes to broaden the planning horizon. Many projects employ a strategy of drilling a few holes of 24 + meter length, spaced out across the face, and repeated every 15–20 m to ensure overlap. It's a proposition for future research to determine if data from a limited number of holes (as opposed to 100–150 for blasting) can still deliver reliable models on big, diversified datasets. Dealing with data from long holes is also more challenging regarding preprocessing. Moreover, future studies should explore rock type predictions on an individual hole basis in tunnelling using big and diversified datasets. If successful, capturing complicated geologic transition zones and detailing the predicted rock type volumes will be possible.

Granular forecasts and extended planning horizons can improve the model procedure in this study, already demonstrating a potential to transform excavation planning from reactive to proactive, offering increased safety due to the detection of challenging rock transition zones, cost savings and reductions in CO_2 emissions, thanks to minimised rock mass transportation and increased reuse of resources.''

5. Supplementary Material

The Python code for the presented ML pipeline is available under the following Github Repository: https://github.com/tfha/ML-MWD-prediction-tabular-rocktype

The dataset used in the study is available at Zenodo for research upon request: https://doi.org/10.5281/zenodo.10358374

The principles and pipeline in the experimentation process for machine learning model development are presented in an online public presentation: https://prezi.com/view/chJ8Djt4GKjeAdFiGvAl/.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used GPT-4 from OpenAI in order to improve the readability and language of some paragraphs in the text. After using this tool/service, the authors reviewed and edited the content as needed and takes full responsibility for the content of the publication.

CRediT authorship contribution statement

Tom F. Hansen: Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Zhongqiang Liu:** Writing – review & editing, Conceptualization. **Jim Torresen:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tom Frode Hansen reports equipment, drugs, or supplies was provided by Bever Control AS. Tom Frode Hansen reports a relationship with Bever Control AS that includes: non-financial support. The assistance mentioned in point 1 (reported work) is the dataset used in the study. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper..

Data availability

Data will be made available on request.

Acknowledgement

The authors gratefully acknowledge Thorvald B. Wetlesen, Ivar Oppen, and Christian H. Svendsen from the tunnel software/hardware company Bever Control, which has helped in developing ideas, review, and facilitation of data from the clients Bane NOR, Statens Vegvesen, Nye Veier, and the contractor AF-Gruppen. Georg H. Erharter's review of design ideas and validity is also highly appreciated.

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

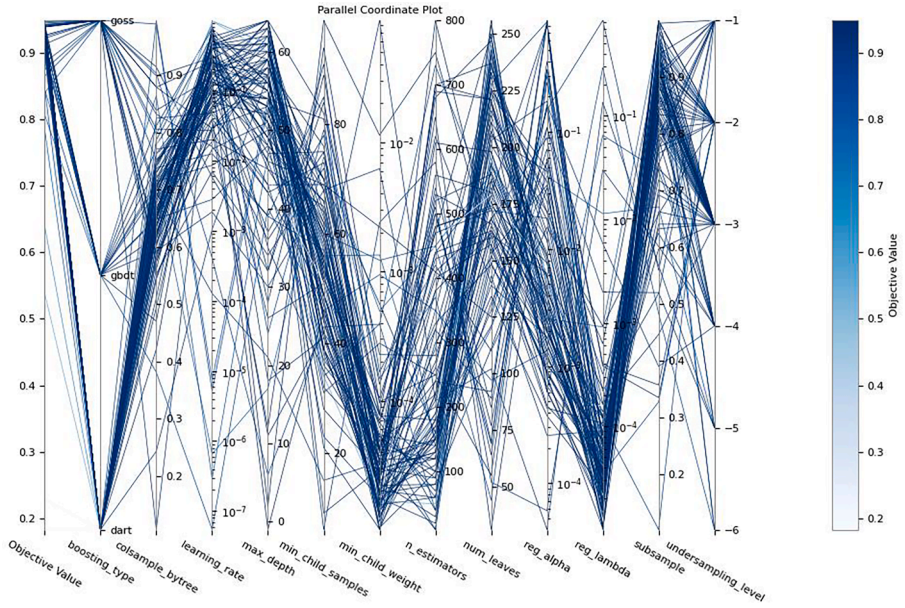
Appendix A. A reliable ML experimentation process

From sections 2.1 and 2.2, we summarise a proposed process for reproducible science-based ML experimentation, including suggested Python libraries:

1. Outlier removal. Libraries: PYOD (for isolation forest and median absolute deviation)
2. Data Partitioning and duplicate checking. Libraries: Scikit-learn, Pandas
3. Split the data into separate training and test datasets. Library: Scikit-learn

4. Feature selection. Libraries: Featurewiz
5. Data Preprocessing in the pipeline: Normalize or scale the data using various scalers within a pipeline object. When working with unbalanced datasets, label classes are balanced through undersampling and oversampling techniques. Libraries: Scikit-learn and imbalanced-learn (for oversampling and undersampling).
6. Machine learning pipelines: Create machine learning pipelines that include preprocessing steps and a range of classifiers (linear, non-linear, and tree-based algorithms). Libraries: Scikit-learn, LightGBM, XGboost.
7. Pipeline selection. Select the best pipeline (trained with default parameters using cross-validation). Library: Scikit-learn
8. Hyperparameter Optimization: Optimize the top-performing pipelines' hyperparameters using a Bayesian approach with a Tree-structured Parzen Estimator (TPE) sampler. Use cross-validation in performance assessment. Libraries/tools: Optuna, Joblib (parallel processing), Scikit-learn.
9. Model Evaluation: Evaluate the models based on well-considered metrics (e.g., balanced accuracy). The best model is then selected to predict the test set. Library: Scikit-learn.
10. Reproducibility. Ensure reproducibility of scientific experiments:
    a. Experiment level (Python libraries). Scikit-learn (control randomness by setting seed), Hydra (for configuration control), MLflow (for experiment tracking and model registry), DVC (for dataset versioning),
    b. Software level. Pyenv (Python version control), Poetry (locking versions and handling package dependencies), GNU-make (for running experiments), Docker (for containerised execution of experiments on a remote machine), GitHub (sharing code for study), Zenodo (sharing dataset).

Appendix B. Parallel coordinate plot of hyperparameter optimisation trials for the pipeline with the LightGBM algorithm



Appendix C. Hyperparameters for the optimised LightGBM model

Algorithm parameters

boosting_type: dart.
colsample_bytree: 0.6563099142197473.
learning_rate: 0.32031365887407864.
max_depth: 49.
min_child_samples: 49.
min_child_weight: 2.9301413598309467e-05.
n_estimators: 130.
num_leaves: 236.
reg_alpha: 0.020733894378166445.
reg_lambda: 2.584873506220451e-05.
subsample: 0.7813241713152921.

Pipeline parameters

undersampling_level: −2 # undersampling to the number of samples in the second most prevalent class.
undersampling algorithm: SMOTE.
scaling: None.
PCA: None.

Appendix D. Listing features from automated feature selection using Featurewiz

'FeedPressNormMedian', 'FeedPressNormVariance', 'HammerPressNormKurtosis', 'HammerPressNormMedian', 'HammerPressNormStandardDeviation', 'PenetrNormMean', 'PenetrNormStandardDeviation', 'PenetrRMSKurtosis', 'PenetrRMSMean', 'RotaPressNormMedian', 'RotaPressNormStandardDeviation', 'RotaPressRMSKurtosis', 'RotaPressRMSMean', 'WaterFlowNormMedian', 'WaterFlowNormSkewness', 'WaterFlowRMSMedian', 'WaterFlowRMSSkewness', 'WaterFlowRMSStandardDeviation'.

References

"3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.2.1 documentation." Accessed: Feb. 09, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.

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

Announcement: Reducing our irreproducibility. Nature, vol. 496, no. 7446, pp. 398–398, Apr. 2013, doi: 10.1038/496398a.

"AutoViML/featurewiz: Use advanced feature engineering strategies and select best features from your data set with a single line of code." Accessed: Jul. 15, 2023. [Online]. Available: https://github.com/AutoViML/featurewiz.

Bantilan, N. pandera: Statistical Data Validation of Pandas Dataframes, 2020, Accessed: Aug. 07, 2023. [Online]. Available: https://en.wikipedia.org/wiki/List_of_the_.

Barrak, A., Eghan, E. and Adams, B., "On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects," 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 2021, pp. 422-433, doi: 10.1109/SANER50967.2021.00046. keywords: {Couplings;Measurement;Pipelines;Tools;Software;Data models;Complexity theory;DVC;ML Pipeline;Co-evolution;ML versioning}.

Black 23.7.0 documentation. Accessed: Aug. 10, 2023. [Online]. Available: https://black.readthedocs.io/en/stable/.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357. https://doi.org/10.1613/jair.953.

Chen, A., et al., 2020. Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle. In: Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning, in DEEM'20. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3399579.3399867.

Chen T., Guestrin C. 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, pp. 785–794. doi: 10.1145/2939672.2939785.

Dickmann, T., Hecht-Méndez, J. 2022. Correlating rock support and ground treatment means with in-tunnel seismic data. In: ITA-AITES World Tunnel Congress, WTC2022, 2022. [Online]. Available: https://www.researchgate.net/publication/364316692.

Docker: Accelerated Container Application Development. Accessed: Aug. 07, 2023. [Online]. Available: https://www.docker.com/.

European Organization For Nuclear Research and OpenAIRE, "Zenodo." CERN, 2013. doi: 10.25495/7GXK-RD71.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit Lett 27 (8), 861–874. https://doi.org/10.1016/J.PATREC.2005.10.010.

Gauch, H.G., 2003. Scientific method in practice. Cambridge University Press.

GNU Project, "GNU Make." 2023. Accessed: Dec. 06, 2023. [Online]. Available: https://www.gnu.org/distros/distros.html.

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition [Book]. Accessed: Jul. 12, 2023. [Online]. Available: https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/.

Hansen, T.F., Liu, Z., Torressen, J. 2024. Building and analysing a labelled measure while drilling dataset from 15 hard rock tunnels in Norway. Available at SSRN: doi: https://doi.org/10.2139/ssrn.4729646.

Hastie, T., Tibshirani, R., Friedman, J. 2009. The elements of statistical learning, vol. 33.

Isheyskiy, V., Sanchidrián, J.A., 2020. Prospects of applying MWD technology for quality management of drilling and blasting operations at mining enterprises. Minerals 10 (10), 1–17. https://doi.org/10.3390/min10100925.

"Joblib: running Python functions as pipeline jobs — joblib 1.3.1 documentation." Accessed: Jul. 25, 2023. [Online]. Available: https://joblib.readthedocs.io/en/latest/index.html.

Kadkhodaie-Ilkhchi, A., Monteiro, S.T., Ramos, F., Hatherly, P., 2010. Rock recognition from MWD Data: A comparative study of boosting, neural networks, and fuzzy logic. IEEE Geosci. Remote Sens. Lett. 7 (4), 680–684. https://doi.org/10.1109/LGRS.2010.2046312.

Kapoor, S., Narayanan, A. 2022. Leakage and the Reproducibility Crisis in ML-based Science. no. Ml, 2022, [Online]. Available: http://arxiv.org/abs/2207.07048.

Kapoor, S., Narayanan, A., 2023. Leakage and the reproducibility crisis in machine-learning-based science. Patterns 4 (9), 100804. https://doi.org/10.1016/j.patter.2023.100804.

Ke G. 2017. et al., LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Klyuchnikov, N., et al., 2019. Data-driven model for the identification of the rock type at a drilling bit. J. Pet. Sci. Eng. 178 (March), 506–516. https://doi.org/10.1016/j.petrol.2019.03.041.

Krawczyk, B., 2016. Learning from imbalanced data: open challenges and future directions. Progr. Artificial Intellig. 5 (4), 221–232. https://doi.org/10.1007/S13748-016-0094-0/TABLES/1.

Leung, R., Scheding, S., 2015. Automated coal seam detection using a modulated specific energy measure in a monitor-while-drilling context. Int. J. Rock Mech. Min. Sci. 75, 196–209. https://doi.org/10.1016/j.ijrmms.2014.10.012.

Liu, M., Gan, Q., 2023. Applied research of comprehensive advance geological prediction in Daluoshan water diversion tunnel. Sci. Rep. 13 (1), Dec. https://doi.org/10.1038/s41598-023-36090-8.

Martin, R.C., 2009. Clean code: a handbook of agile software craftsmanship. Pearson Education.

Navarro, J., Sanchidrian, J.A., Segarra, P., Castedo, R., Paredes, C., Lopez, L.M., 2018. On the mutual relations of drill monitoring variables and the drill control system in tunneling operations. Tunn. Undergr. Space Technol. 72 (October 2017), 294–304. https://doi.org/10.1016/j.tust.2017.10.011.

Pedregosa, F., et al., 2011. Scikit-learn: Machine Learning in {P}ython. J. Mach. Learn. Res. 12, 2825–2830.

"Poetry - Python dependency management and packaging made easy." Accessed: Aug. 07, 2023. [Online]. Available: https://python-poetry.org/.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A. 2023. CatBoost: unbiased boosting with categorical features. Accessed: Aug. 07, 2023. [Online]. Available: https://github.com/catboost/catboost.

pyenv/pyenv: Simple Python version management. Accessed: Aug. 07, 2023. [Online]. Available: https://github.com/pyenv/pyenv.

Shneiderman, B., Aug. 2021. Responsible AI. Commun ACM 64 (8), 32–35. https://doi.org/10.1145/3445973.

Silversides, K.L., Melkumyan, A., 2022. Machine learning for classification of stratified geology from MWD data. Ore Geol. Rev. 142, 104737 https://doi.org/10.1016/j.oregeorev.2022.104737.

Soklaski, R., Goodwin, J., Brown, O., Yee, M., Matterer, J. Tools and Practices for Responsible AI Engineering. 2022, [Online]. Available: http://arxiv.org/abs/2201.05647.

Van Eldert, J., Schunnesson, H., Johansson, D. 2017. The History and future of rock mass characterisation by drilling in drifting from sledgehammer to PC-tablet. In: Conference: 26th International symposium on mine planning & equipment selection, no. July 2019.

van Eldert, J., Schunnesson, H., Saiang, D., Funehag, J. 2020. Improved filtering and normalizing of Measurement-While-Drilling (MWD) data in tunnel excavation. Tunnel. Undergr. Space Technol., vol. 103, no. March, p. 103467. doi: 10.1016/j.tust.2020.103467.

Vezhapparambu, V.S., Eidsvik, J., Ellefmo, S.L. 2018. Rock classification using multivariate analysis of measurement while drilling data: Towards a better sampling strategy. Minerals, vol. 8, no. 9, 2018, doi: 10.3390/min8090384.

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., Teal, T.K., 2017. Good enough practices in scientific computing. PLoS Comput. Biol. 13 (6), e1005510.

Wilson G., et al. 2014. Best practices for scientific computing. PLoS Biol., vol. 12, no. 1, 2014, doi: 10.1371/journal.pbio.1001745.

Yadan, O. 2019. Hydra - A framework for elegantly configuring complex applications. [Online]. Available: https://github.com/facebookresearch/hydra.

Zhao, Y., Nasrullah, Z., Li, Z. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. J. Mach. Learn. Res., vol. 20, pp. 1–7, 2019, Accessed: Jul. 12, 2023. [Online]. Available: https://pyod.readthedocs.io.