**Brief description of the methods used**

**(1) Decision Tree**

Decision Tree (DT) focuses on inferring a set of classification rules represented in the form of a DT from a set of unordered instances (Safavian & Landgrebe, 1991). It employs a top-down recursive approach, branching downward based on different attributes. The splitting process continues until a stopping criterion is met, such as reaching a maximum depth or no further improvement in impurity. Conclusions are reached at the leaf nodes of the DT. Thus, from the root node to the leaf node, there corresponds to a logical rule, and the entire tree represents a set of expression rules. When DT is used for multi-class classification of pyrite, the leaves represent the multiple pyrite classes of training LA-ICP-MS analyses, and branches represent features. One of the major advantages of the DT algorithm is that it does not require the user to have extensive background knowledge during the learning process. As long as the training instances can be expressed in terms of attributes and conclusions, this algorithm can be used for learning.

Classification and Regression Trees (CART) is a DT algorithm that can handle both categorical and continuous input variables. Due to its good fitting ability, we adopted this algorithm in this study. CART is used for binary splitting, which involves splitting the data into two groups based on the value of a single input variable. DT evaluates different features and selects the one that best separates the data based on certain criteria, such as the Gini impurity index:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

where *C* is the total number of pyrite classes, and probability of selecting an LA-ICP-MS analysis with pyrite class 𝑖. The Gini impurity of a pure node (the same class) is equal to zero.

**(2) Random Forest**

Random Forest (RF) is an ensemble learning algorithm that combines multiple decision trees to make predictions (Breiman, 2001). It improves prediction accuracy without significantly increasing computational complexity. RF is insensitive to multicollinearity, and its results are robust to missing and imbalanced data. It can effectively predict the impact of numerous explanatory variables.

RF can explain the effects of several independent variables (X1, X2,..., X*k*) on the dependent variable *Y*. If there are *n* observations for the dependent variable *Y*, and *k* variables are related to it, RF randomly selects *n* observations from the original data during the construction of classification trees. Some observations are selected multiple times, while others may not be selected at all. This is done through the method of Bootstrap resampling. Additionally, RF randomly selects a subset of variables from the *k* independent variables to determine the nodes of the classification tree. As a result, each constructed tree may differ. During prediction, each decision tree in the RF independently classifies the input data. In the case of multi-class classification, different voting strategies can be used. One common approach is the "majority voting" method, where each tree's prediction is considered as a vote for a particular class. The class with the most votes is assigned as the final prediction.

In our case, employing RF, several CART-type decision trees are created using randomly selected subsets of training LA-ICP-MS analyses and features. Each tree in the forest makes a prediction based on the values of the features and provides a pyrite class label. The final prediction of the RF is the dominant class.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

where 𝑓 = *mode*, *n* number of tress DT in RF. The 𝑚𝑜𝑑𝑒 function returns the most frequently occurring predicted class across all trees.

Theoretically, the RF algorithm has several advantages over a single decision tree in that it can handle high-dimensional, less balanced datasets with many attributes and can reduce overfitting.

**(3) Naive Bayes**

Naive Bayes (NB) is a probabilistic classifier based on Bayes' theorem with the assumption of feature independence (Rish, 2001). In multi-class classification, NB calculates the posterior probability of each class given the input features and assigns the class with the highest probability as the prediction. The algorithm uses the joint probability of the features and the class labels along with prior probabilities to compute the posterior probabilities.

Let denote a set of n LA-ICP-MS analyses, where is the *l*th LA-ICP-MS analysis, *l* = 1,2,…,*n*. Let denote the feature vector of LA-ICP-MS analysis concerning the feature set , where denotes the *k*th feature (geochemical element) in the set *T*, *l* = 1,2,…,*n*, *k* = 1,2,…,*e*. Let denote a set of *m* pyrite classes, where is the *i*th pyrite class of the LA-ICP-MS analyses, *i* = 1,2,…,*m*. The probability of LA-ICP-MS analysis belonging to the pyrite class can be estimated by

|  |  |  |
| --- | --- | --- |
|  | , *l* = 1,2,…,*n*, *i* = 1,2,…,*m* | (3) |

where is the conditional probability of the geochemical element occurring in an LA-ICP-MS analysis that belongs to the pyrite class and is the prior probability of an LA-ICP-MS analysis that belongs to the pyrite class and can be estimated according to the training LA-ICP-MS analyses.

Then, the pyrite class of a test LA-ICP-MS analysis is determined according to the highest probability among the probabilities of the LA-ICP-MS analysis belonging to the different pyrite classes, i.e.,

|  |  |  |
| --- | --- | --- |
|  | *l* = 1,2,…,*n* | (4) |

**(4) Logistic Regression**

Logistic Regression (LR) is a linear classification algorithm that models the relationship between the input features and the probability of belonging to a specific class (Dreiseitl & Ohno-Machado, 2002). In multi-class tasks, the Softmax function is used to calculate the probabilities for each class. The Softmax function converts the linear combination of input features and weights into a probability distribution over the classes. The predicted class is the one with the highest probability. The formula for the Softmax function is:

|  |  |  |
| --- | --- | --- |
|  | , *k* = 1,2,…,*K* | (5) |

where is a score of pyrite class *k* for input the LA-ICP-MS analysis *x*. The score can be defined as a linear function.

During training, the parameters of the model (i.e., the weights and biases of the linear functions) are learned using an optimization algorithm that minimizes a loss function. The most commonly used loss function for softmax regression is cross-entropy loss. The cross-entropy loss for a single example (𝑥,*y*) is defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

where is the indicator function, taking the value 1 if the true label is 𝑘 and 0 otherwise. The class with the highest probability is chosen as the output for the prediction:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

where is the predicted pyrite class label.

**(5) K-Nearest Neighbor**

K-Nearest Neighbor (KNN) is known for its simplicity and intuitiveness (Cunningham & Delany, 2021). KNN is a non-parametric algorithm that classifies data points based on the class labels of their nearest neighbors in the feature space. It can handle non-linear decision boundaries and adapt to the complexity of the dataset. In multi-class classification, KNN finds the K nearest neighbors of a data point based on a distance metric (e.g., Euclidean distance) and assigns the class label based on the majority vote of those neighbors. The algorithm does not involve explicit formulae but relies on the proximity of data points in the feature space.

In this study, to determine the pyrite class of a test LA-ICP-MS analysis , *K* nearest neighbors are first selected from the training LA-ICP-MS analyses, which are denoted as *td*1, *td*2, *tdK*. Let denote a set of *m* pyrite classes, where is the *i*th pyrite class of the LA-ICP-MS analyses, *i* = 1,2,…,*m*. According to the similarities of between the *K* nearest neighbors and the different pyrite classes of the *K* nearest neighbors, a KNN score of LA-ICP-MS analysis concerning pyrite class can be calculated by

|  |  |  |
| --- | --- | --- |
|  |  | (8) |
|  |  | (9) |

where is the similarity between LA-ICP-MS analysis and training LA-ICP-MS analyses ; is the indicator of the training LA-ICP-MS analyses with respect to pyrite class . According to the obtained KNN scores, the pyrite class of the test LA-ICP-MS analysis is determined, i.e.,

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

**(6) Support Vector Machines**

Support Vector Machines (SVM) is widely used for both binary and multi-class classification tasks (Cortes & Vapnik, 1995). SVM aims to find an optimal hyperplane that separates data points of different classes in a high-dimensional feature space. In multi-class tasks, SVM uses one-vs-one or one-vs-rest strategies. In one-vs-one, SVM trains a binary classifier for each pair of classes and selects the class that wins the most pairwise comparisons. In one-vs-rest, SVM trains a binary classifier for each class against the rest and assigns the data point to the class with the highest confidence score. The decision boundary is determined by the support vectors, which are the data points closest to the hyperplane. SVM can handle non-linear classification tasks by mapping the input data into a higher-dimensional feature space using kernel functions. Kernel functions capture complex relationships between the data points in the original feature space and allow SVM to find non-linear decision boundaries in the transformed space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. After several rounds of hyperparameters optimization, the Model finally selects "rbf" as the kernel function.

Let and denote two different pyrite classes of training LA-ICP-MS analyses. Let = +1 if LA-ICP-MS analysis belongs to , and let = -1 if LA-ICP-MS analysis belongs to . Let be a vector representing the hyperplane that separates the feature vectors of training LA-ICP-MS analyses belonging to the two pyrite classes and . The solution of the optimization problem, i.e., , can be represented by

|  |  |  |
| --- | --- | --- |
|  | , | (11) |

where is obtained by solving a dual optimization problem; is the number of feature vectors of training LA-ICP-MS analyses, i.e., = + , and are the numbers of training LA-ICP-MS analyses belonging to the pyrite classes and , respectively; is a support vector of the hyperplane if and only if the is greater than zero. It is necessary to point out that SVM cannot be directly used to train a multi-class pyrite classifier since SVM is a binary classifier. Thus, to train the multi-class pyrite classifiers based on the SVM, one-vs-one or one-vs-rest strategies need to be used.

**(7) Multilayer Perceptron**

Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected neurons, including an input layer, one or more hidden layers, and an output layer (Pal & Mitra, 1992). In multi-class classification, the amount of neurons in the output layer of MLP is equal to the number of classes to be classified, while the number of neurons in the input layer is associated with the number of features. The network learns by adjusting the weights through a process called backpropagation, where the errors between predicted and actual outputs are propagated backward to update the weights. When performing forward propagation, the network calculates the output of each layer based on an activation function from the previous layer as well as corresponding weight and bias values, as shown in Eq. 12.

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

where denotes the output matrix, is weight matrix and denotes bias vector.

Since the output of an MLP could be any value, an activation function is used to normalize the output. The activation function can transform the output of each layer to a certain range as shown in Eq. 13.

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

where denotes the activated output matrix.

MLP uses Relu as the activation function for the hidden layer and Softmax activation function in the output layer to compute the probabilities for each class. Relu, as defined in Eq. 14, is an activation function that only transforms values less than zero to 0. The Softmax activation function, as defined in Eq. 15, is usually used for multi-classification, which can improve the defects of the sigmoid function for multi-classification and ensure that the probability sum of the output layer is equal to 1. It can help determine the most probable prediction.

|  |  |  |
| --- | --- | --- |
|  |  | (14) |
|  |  | (15) |

where *J* is the number of pyrite classes, denotes the *i*th output value.

The loss function, as defined in Eq. 16, is used to calculate the error between the predicted value and the actual value and then use back-propagation to adjust the weights *w* and bias *b*.

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

where *m* is the number of samples, is the predicted value, and *y* is the exact value.