Here, we only briefly derive the theoretical analysis of the XGBoost model, and the theoretical analysis of other algorithms refers to the corresponding papers. XGBoost, developed by Chen and Guestrin (2016), is an improved gradient boosting decision tree algorithm.

Given that our zircon trace element dataset, DS has n samples and m geochemical indicators (called features in machine learning)

.

A tree ensemble model predicts the geological temperature (it refers to the dependent variable in machine learning) such  using the geochemical indicators (it refers to the independent variable in machine learning) and K additive functions:



where K denotes the number of trees in the model,  denotes the k-th tree with leaf scores and F is the space of regression trees.

To solve the above equation, we minimize the loss function and regularized objective:



where  is a loss function, which is used to measure the difference between the predicted geological temperature (prediction ) and the actual temperature (target ).

In the above formula, Ω is used to penalize the complexity of the model, so as to avoid model overfitting. And it is calculated using:

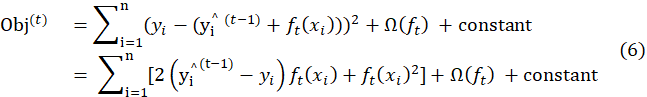


where  and  are penalty parameters, T represents the number of leaves in the tree, and w represents the weight of each leaf.  represents the L2 norm of the leaf weight w.

Boosting is applied to the decision tree to minimize the objective function. Boosting works by adding a new function f to the model as it continues to train. So, in the t-th iteration, a new function (tree) is added as follows:



Using the mean square error (MSE) as the loss function, the objective function of the t-th iteration is rewritten as:



By performing a second-order Taylor expansion, we can obtain Eq. (7):



where  and  are shown in Eqs. (8) and (9), respectively:

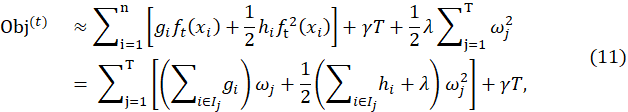




Removing the constant term, we can rewrite Eq. (7) as Eq. (10), which is the simplified objective function at step (10):

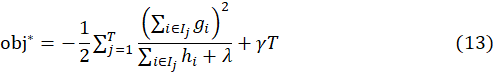


Substituting Eq. (2) and Eq. (4) into Eq. (10), we deduce Eq. (11):

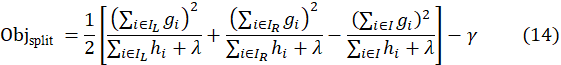


where  is the instance set of leaf j. For a given structure q(x) , the optimal weight 𝜔∗ j of leaf j can be computed by Eq. (12), and the corresponding optimal value obj\* is calculated by Eq. (13):





Since, in practice, it is difficult to calculate this value for all possible tree structures, instead, the following formula is used:



where  and  are the instance sets of the left and right nodes after splitting, respectively; I =  ⋃.