ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2023, Vol. 62, No. 4, pp. 619–626. © Pleiades Publishing, Ltd., 2023.
ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2023. © Pleiades Publishing, Ltd., 2023.
Russian Text © The Author(s), 2023, published in Izvestiya Akademii Nauk, Teoriya i Sistemy Upravleniya, 2023, No. 4, pp. 17–24.

New Two-Level Machine Learning Method for Evaluating the Real Characteristics of Objects

A. A. Dokukin[a],* and O. V. Sen'ko[a],**

[a] *Federal Research Center "Computer Science and Control," Russian Academy of Sciences, Moscow, 119333 Russia*

** e-mail: dalex@ccas.ru*

*** e-mail: senkoov@mail.ru*

Received February 14, 2023; revised March 1, 2023; accepted April 3, 2023

Abstract—A new two-level ensemble regression method, as well as its modifications and application in applied problems, are considered. The key feature of the method is its focus on constructing an ensemble of predictors that approximate the target variable well and, at the same time, consist of algorithms that, if possible, differ from each other in terms of the calculated predictions. The ensemble with the indicated properties at the first stage is constructed through the optimization of a special functional, whose choice is theoretically substantiated in this study. At the second stage, a collective solution is calculated based on the forecasts formed by this ensemble. In addition, some heuristic modifications are described that have a positive effect on the quality of the forecast in applied problems. The effectiveness of the method is confirmed by the results obtained for specific applied problems.

DOI: 10.1134/S1064230723040020

INTRODUCTION

Ensemble methods have a very long history and are an important part of machine learning technologies used in solving problems of learning by precedents: classification or prediction of numerical variables [2, 3]. The ensemble method is usually understood as a method that calculates solutions in two stages: by separate ensemble algorithms and by a collective algorithm.

Among ensemble technologies, the method of random forests [4] and the gradient boosting method [5] are most widely used. These technologies have been successfully used earlier, including for the tasks of evaluating real characteristics from indicative descriptions of objects.

The task of learning from precedents in the above sense is defined as follows. We consider a set of objects that make it possible to measure or calculate n of their numerical characteristics (features) $X_1,\dots,X_n$. We assume that some of these objects also measure the target feature Y, which we will call precedents. A training sample is a set of precedents $S = \{(y_1,x_1),\dots,(y_m,x_m)\}$, where y_i, $i=\overline{1,m}$, are the values of the target variable Y for the ith object and x_i is the vector of the feature description of the ith object, $x_i = (X_1^i,\dots,X_n^i)$. It is required to build an algorithm A to determine (predict) the value Y for other objects: $A(x) = y$. In the case when Y takes on categorical values, the task can be called a classification task, and in the case when Y is a continuous numerical characteristic, it is called a regression.

In the random forest method [4], trees are generated independently using a combination of bagging and the random subspace method [6, 7]. In other words, every new tree T_k, added to the ensemble at the step k, is based on the sample S_k^b, which is a sample with a return from the projection of the original training sample $S_k^b \subset S^b = \{(y_1,x_1^b),\dots,(y_m,x_m^b)\}$, where x_i^b is the vector projection x_i to the feature subspace $X^b \subset \{X_1,\dots,X_n\}$. The tree generation process stops when the total number of trees in the ensemble reaches a predetermined number. In the random forest method, simple collective solutions are used: when solving classification problems, an object belongs to the class to which it was assigned by most of the trees in the ensemble; when solving regression problems, the collective forecast is calculated as the average forecast over the ensemble.

In methods based on gradient boosting, each new tree T_k, added to the ensemble at step k, performs one step of gradient descent. We assume the function $l(y_j, A(x_j))$ describes the loss that occurs when used as the forecast Y at the point x_j of value $A(x_j)$, for example, the quadratic error $l(y_j, A(x_j)) = (y_j - A(x_j))^2$. For training, gradient l is calculated by $(A(x_1),\dots,A(x_m))$ and $\nabla l = \{r_1,\dots,r_m\}$, where $r_j = \partial l(y_j, A(x_j))/\partial A(x_j)$, $j = \overline{1,m}$. Obviously, for the quadratic error $r_j = 2(A(x_k) - y_j)$. We assume that A_k is an ensemble built by the kth step:

$$A_1 = T_0,$$

$$A_k = T_0 - \sum_{i=1}^{k-1}\varepsilon_i T_i, \quad k > 1,$$

where T_0 is some initial approximation, for example, identical zero. If now T_k is trained on the sample $\{r_1^k,\dots,r_m^k\}$, thereby approximating the gradient of the loss function at $A = A_k$ and ε_k is the gradient descent step at step k, then the ensemble constructed in this way will optimize the given functional increasingly accurately with each step, and, therefore, approximate the target variable.

The method of random regression forests and the gradient boosting method are widely used to solve various applied problems and do so highly efficiently in many cases. Moreover, neither of them is undisputedly better than the other; in practice, some tasks are performed better by one or the other method, sometimes by a significant margin. At the same time, it can be assumed that ensemble solutions can be highly efficiently solved by other methods, apart from the two mentioned above. There is even an intention to combine these approaches, which is realized to some extent in the proposed method.

1. OPTIMIZED FUNCTIONALITY

In the random forest method, ensembles are generated randomly, which obviously does not ensure their optimality. At the same time, gradient boosting is not a full ensemble method—it is undoubtedly the sum of algorithms, but each one separately approximates the derivatives of the loss function rather than the target variable. Our method proposes to combine the best of the two approaches. We will build an ensemble of various elements, each of which is random to a certain extent, and approximates the target variable. In this case, the elements themselves will be built by optimizing a special functional, to the derivation of which we turn.

We consider an iterative procedure for constructing an ensemble. We denote the arbitrary algorithm by $B_i(x)$: as a tree, for example, and a linear combination of trees obtained at the ith step. By step k the ensemble will consist of all the algorithms built in the previous steps, and the ensemble's solution $A_k(x)$ is the average of these algorithms:

$$A_1(x) = O(x),$$

$$A_k(x) = \frac{1}{k-1}\sum_{i=1}^{k-1}B_i(x), \quad k > 1,$$

where $O(x)$ is identical zero.

We are interested in the mean square of the deviation:

$$L(S, A_k) = \frac{1}{m}\sum_{j=1}^{m} l\left(y_j, A_k(x_j)\right) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - A_k(x_j)\right)^2.$$

Note that this functional can be expressed in terms of the mean square of the deviation over all objects and algorithms,

$$\frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - B_i(x_j))^2,$$

for which we will consider the latter:

$$\frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - B_i(x_j))^2 = \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - A_k(x_j) + A_k(x_j) - B_i(x_j))^2$$

$$= \frac{k}{km}\sum_{j=1}^{m}(y_j - A_k(x_j))^2 + \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(A_k(x_j) - B_i(x_j))^2$$

$$- \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - A_k(x_j))(A_k(x_j) - B_i(x_j)) = \frac{1}{m}\sum_{j=1}^{m}(y_j - A_k(x_j))^2$$

$$+ \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(A_k(x_j) - B_i(x_j))^2 - \frac{1}{m}\sum_{j=1}^{m}(y_j - A_k(x_j))\frac{(A_k(x_j) - B_k(x_j))}{k}.$$

From this it follows that the mean squared error for the algorithm A_k is calculated by the formula

$$\frac{1}{m}\sum_{j=1}^{m}(y_j - A_k(x_j))^2 = \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - B_i(x_j))^2 - \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(A_k(x_j) - B_i(x_j))^2$$

$$+ \frac{1}{m}\sum_{j=1}^{m}(y_j - A_k(x_j))\frac{(A_k(x_j) - B_k(x_j))}{k}.$$

The two terms $L(S, A_k)$ in the resulting formula have a simple interpretation. The first one describes the deviation of forecasts from the true values Y, and the second one is the average variance of predictions for objects from S. The last term is more difficult to interpret, but several observations can be made. First, it tends to zero as the size of the ensemble k increases. Second, its appearance is caused by the desire to simplify the algorithm and determine A_k as the sum of the terms of the previous steps without including the desired algorithm; otherwise this term is strictly equal to zero. Finally, in the future, the method will require several more nonstrict transitions and additional heuristics, so we will not include the last term in the functional:

$$L(S, A_k) = \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - B_i(x_j))^2 - \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(A_k(x_j) - B_i(x_j))^2. \tag{1.1}$$

Thus, the optimality of the ensemble can be achieved due to two factors: a good approximation of the relationship of Y with variables $X_1, \ldots, X_n$ on the training sample and a strong variance in the forecasts of the training objects.

These considerations allow us to choose the functional for optimizing the new ensemble term B_k, $B_k = \arg\min_B Q(B, A_k)$. Rigorous calculation of quality gain $L(S, A_k) - L(S, A_{k-1})$ according to (1.1) leads to a significant complication of the algorithm:

$$L(S, A_k) - L(S, A_{k-1}) = \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(y_j - B_i(x_j))^2 - \frac{1}{(k-1)m}\sum_{j=1}^{m}\sum_{i=1}^{k-1}(y_j - B_i(x_j))^2$$

$$- \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}(A_k(x_j) - B_i(x_j))^2 + \frac{1}{(k-1)m}\sum_{j=1}^{m}\sum_{i=1}^{k-1}(A_{k-1}(x_j) - B_i(x_j))^2$$

$$= \frac{1}{km}\sum_{j=1}^{m}(y_j - B_k(x_j))^2 + \frac{1}{k(k-1)m}\sum_{j=1}^{m}\sum_{i=1}^{k-1}(y_j - B_i(x_j))^2$$

$$- \frac{1}{km}\sum_{j=1}^{m}(A_k(x_j) - B_k(x_j))^2 - \frac{1}{k(k-1)m}\sum_{j=1}^{m}\sum_{i=1}^{k-1}(A_{k-1}(x_j) - B_i(x_j))^2.$$

The contribution of one element to this functional decreases with an increase in the size of the ensemble; hence, it is more logical to use $k(L(S, A_k) - L(S, A_{k-1}))$, and this difference, in turn, with increasing k approaches

$$Q(B, A_k) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - B(x_j)\right)^2 - \frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}\left(A_k(x_j) - B(x_j)\right)^2,$$

which is much more convenient. Nevertheless, $Q(B, A_k)$ does not reach the minimum $B(x_1),\ldots,B(x_m)$ and requires the introduction of parameter μ, where $\mu \in [0,1)$. Further constructions are based on the assumption that the optimality of the ensemble in the sense of ensuring the maximum generalizing ability with its help will be achieved for corrective trees, under which the functional $Q(B, A_k, \mu)$ is minimal:

$$Q(B, A_k, \mu) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - B(x_j)\right)^2 - \mu\frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}\left(A_k(x_j) - B(x_j)\right)^2, \tag{1.2}$$

i.e., $B_k = \arg\min_B Q(B, A_k, \mu)$.

Further, functional (1.2) allows us to apply the gradient boosting procedure directly to construct the elements of the ensemble. This approach was implemented in [8] and showed good results in practical problems. However, it also has some shortcomings. First of all, it increases the complexity of the training. In addition, it becomes necessary to select the gradient boosting step.

The direct construction of a tree optimizing functional (1.2) does not have these shortcomings. The gradient boosting analogy remains, but B reaches the minimum of the functional in one step, which reduces its complexity and does not require the selection of a parameter. However, this method is related to some technical difficulties and will be the goal of further research. To test the concept itself, we consider another approach implemented using the standard scikit-learn methods [9]. We assume that the minimum of Q is reached at the point $B^*(x_1),\ldots,B^*(x_m)$. The regression tree T_k, trained on the sample $\{(B^*(x_1), x_1),\ldots,(B^*(x_m), x_m)\}$, can be used as B_k. This approach is similar in complexity and the number of parameters to the previous one; however, in this paper we will sacrifice simplicity to achieve additional diversity in the ensemble using additional techniques such as bagging and the random subspace method. We describe this approach in more detail.

We assume that we have obtained ensemble $B_1,\ldots,B_{k-1}$ for the first $k-1$ steps. We generate bootstrap replication S_k^b of samples S in the projection onto a random subspace, as described in the introduction. We build a regression tree T_k by replication S_k^b. The algorithm B_k is sought as the sum $B_k = T_k + t_k$, where t_k is a correction tree that is built based on the minimization condition $Q(T_k + t_k)$.

Let us rewrite functional (1.2) for this procedure:

$$Q(t, \mu) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - T_k(x_j) - t(x_j)\right)^2 - \mu\frac{1}{km}\sum_{j=1}^{m}\sum_{i=1}^{k}\left(A_k(x_j) - T_k(x_j) - t(x_j)\right)^2. \tag{1.3}$$

At the first step, a real vector $t^* = t_1^*,\ldots,t_m^*$, whose components are the optimal prediction biases computed by the tree T_k, i.e., $t^* = \arg\min Q(t, \mu)$, is sought. The minimum value of the functional $Q(t, \mu)$ according to t_j is achieved at $\partial Q(t, \mu)/\partial t(x_j) = 0$ or when

$$t(x_j) = t_j^* = \frac{k}{k-\mu}y_j - T_k(x_j) - \frac{\mu}{k-\mu}A_k(x_j), \quad j = \overline{1,m}. \tag{1.4}$$

At the same time, the calculation of the optimal biases by formula (1.4) can lead to a decrease in the accuracy of the generated algorithm B_k towards T_k subject to the following sets of inequalities:

$$B_k(x_k) = T_k(x_j) + t_k(x_j) < T_k(x_j) < y_j \tag{1.5}$$

or

$$B_k(x_k) = T_k(x_j) + t_k(x_j) > T_k(x_j) > y_j. \tag{1.6}$$

Such a decrease in accuracy can be avoided if, when one of the sets of inequalities (1.5) or (1.6) is satisfied, we equate t_j^* to zero.

Thus, a recursive procedure for constructing an ensemble of algorithms is described. Starting with an empty ensemble and identically zero as the collective solution, we fill it with the sums of pairs of trees, where the first one approximates the target variable directly and the second one approximates the correction that minimizes the functional $Q(t, \mu)$ (1.3). The construction of the ensemble is completed if k reaches a user-defined threshold N. The ensembles that are constructed using this procedure will be called *decorrelated* below.

2. COLLECTIVE SOLUTION AND ADDITIONAL HEURISTICS

Together with the ensemble generation method, the procedure that calculates the collective solution plays an important role in making the ensemble algorithm efficient. Although the ensemble was built based on considerations of minimizing the error of the averaged response of the algorithms, the use of such a collective decision scheme turned out to be insufficiently effective. In [8], several other methods were considered, in addition to the average over the decorrelated ensemble, in particular, stacking [10], i.e., application of predictions calculated by individual trees of the ensemble as features for second-level algorithms that calculate the output collective solution. The experiments presented in [8] show that higher accuracy is achieved with random regression forest stacking than with simple averaging. For this reason, several variants of the collective solution are implemented in the method: averaging and stacking with the gradient boosting method and random forest, as well as their convex combination.

The most accurate prediction is usually achieved with large ensemble sizes. However, an excessively large number of signs leads to an increase in instability and a decrease in the accuracy of the forecast. Various approaches can be used to reduce the feature space. The proposed method uses two techniques: decimation and reoptimization. In the first case, after calculating the complete decorrelated ensemble, all regression trees included in it are ranked by the value of the functional $Q(B,\mu)$, where for an arbitrary element B_j, the ensemble is assumed to be equal to $\{B_i \in A_k \,|\, i \neq j\}$. Further, all insufficiently effective terms are excluded from the ensemble in such a way as to leave the predetermined N' elements. The experiments on real problems did not reveal a noticeable increase in efficiency from using this procedure. Perhaps this is not the case with a larger number of initial terms; however, in this case the computational complexity of the procedure increases significantly. Reoptimization leaves the number of terms, but tries to further spread them out. We consider the ensemble $\{B_1,\ldots,B_N\}\backslash B_j$ and apply the procedure for constructing a new algorithm B to this ensemble, and then replace B_j in the resulting algorithm, etc., for all terms in turn. The effect of this procedure was also not spectacular; however, these two procedures were left in the method for further research.

Somewhat better efficiency was shown by the use of a variant of the method of extreme grouping (EG) of the parameters to reduce the dimension of the feature space [11]. At the initial stage, the features are randomly divided into a certain predetermined number of groups, for each of which the corresponding group factor is calculated as the average of all the features included in the group. In this case, some of the features are multiplied by -1 to ensure the same direction of the links with the target variable. The procedure used is similar to the standard k-average clustering method and consists of transferring each of the features to the group for which the modulus of the correlation coefficient between the feature and the corresponding group factor is maximum. Next, the group factors are recalculated. The process ends if there is no need to transfer features in any of the steps. In the implemented method, EG can be used both for the initial features, which is especially important, for example, for chemical problems with a high correlation of the features, and for the generated space.

This study does not use stacking or the preliminary clustering of features, since they are general procedures applicable to any method.

Thus, the properties of the final algorithm are determined by the following set of parameters: N is the number of elements of the ensemble (in the experiments $N = 200$ was fixed); N' is the number of ensemble elements after sifting (in the experiments, $N' = N$ was fixed); μ is the effect of the decorrelating component of the functional (in the experiments, $\mu = 0.5$); ν is the proportion of features used to build T_k; κ is the contribution parameter t_k: $B_k = T_k + \kappa t_k$ (in the experiments $\kappa = 1$ was fixed); G is the number of EG groups in the ensemble; and corrector = average|forest|boosting is the type of corrective procedure. The parameters of the final corrective procedure are set separately. For boosting and bagging, the default parameters are taken and no parameters are needed for averaging.

As is usually the case, there is no single set of hyperparameters that is effective for all applications. An example of combinations that allow us to achieve advantages over the pure gradient boosting and random forest methods for predicting the properties of chemical elements is presented in [12]. The next chapter describes the application of the developed method to several other problems.

3. APPLICATION PROBLEMS

The following problems were used to assess the quality of the studied algorithm and the range of optimal hyperparameters.

Houses: 'House Prices' sample: Advanced Regression Techniques'[1], containing data for estimating real estate prices on various grounds, from the topography of the site and the length of the adjacent street to the type and quality of the finish of the building and its proximity to the railroad. There are 1460 objects in the problem, described by 79 features, most of which are categorical. To apply the method to these features, the following coding procedure was applied: for the training subsample, the feature values were replaced by the average target indicator for all objects with the same feature value; in the test subsample, the value from the training subsample was used; if the test subsample included a value unknown in the training subsample, it was replaced by the average value of the target feature over the entire sample; and gaps in the quantitative features were replaced by zeros, since their nature is the area of the missing element.

Prices and costs are another selection of real estate[2] from the UCI repository [13]. The sample contains 372 objects described by 107 numerical features, which predict the price of real estate and the cost of its construction. These two tasks were considered separately.

Systolic is the task of estimating systolic pressure using an electrocardiogram (ECG) signal [14]. The sample contains 836 objects described by 160 spectral indices.

The developed method is implemented using the standard methods from the scikit-learn library [9] and the novelty is reduced to the use of a modified target variable. The tree T_k is sought using a BaggingRegressor with the number of elements equal to one and *max_features* $= \nu$. The correction t_k is implemented using a DecisionTreeRegressor. For the experiments, some of the hyperparameters were fixed and several runs were made with different values of μ, ν, G, and the corrector. They are presented in this order in the description of the results.

The reference methods were also taken from scikit-learn: BoostingRegressor as an implementation of gradient boosting and RandomForestRegressor as a random forest. All methods were run with an ensemble size of 200 and the rest of the default settings. We consciously abandoned attempts to compare quality with other researchers and conducted our own tests for several reasons, mainly in order to compare the methods in relatively the same conditions and identify the effect of the decorrelation technique. Thus, the Houses task requires feature preprocessing, and the results are improved by the leaders of the Kaggle competition, including by improving this processing. The same is true for the Costs and Prices problems in many respects: the most recent available study of these data is also largely focused on the specifics of the sample [15], while we study the general approach. The Systolic problem in this form has not been studied by anyone.

The results and launch parameters are presented in Table 1. To assess the quality of the prediction, two characteristics were used: the coefficient of determination (R-square, R^2) and the mean absolute error (MAE). The tasks were solved in the sliding control mode with division into 10 parts. In addition, the results were averaged over 10 runs to reflect their robustness; therefore, the table shows the mean and standard deviation (stdev) of the indicators.

The first thing we should pay attention to is the different ratios of the quality of the reference methods themselves. It can be seen that in problems with real estate, boosting shows much better quality than random forest. At the same time, boosting lags far behind in the problem of pressure estimation. In turn, the proposed method always demonstrates superiority over the outsider. At the same time, in the Prices and Costs tasks, it only managed to catch up with the leading method. In the Houses problem, the leadership of the proposed method is clear, as it leads both in terms of R^2 and the mean absolute error. The best result of the method was demonstrated in the problem of estimating systolic pressure, in which the leader of the pair of reference methods—this time the random forest—managed to significantly outperform in terms of both criteria.

As for the choice of hyperparameters, their nonuniversality was once again confirmed. In some cases, the advantage was given by a high ν, and in other cases, on the contrary. In problems with real estate, random forest turned out to be a good corrector, and in pressure prediction, simple averaging turned out to be a good corrector.

[1] https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data.
[2] https://archive.ics.uci.edu/ml/datasets/Residential+Building+Data+Set.

Table 1. Experimental results

Task	Model	R^2		MAE	
		mean	stdev	mean	stdev
Houses	Boosting	0.882	0.006	17642	352.37
	Forest	0.861	0.0051	17923	112.72
	0.5, 0.4, 40	0.881	0.0134	16645	211.1
	0.5, 0.4, 40 boost	0.879	0.01	17232	239.12
	0.5, 0.4, 40, average	**0.888**	**0.0035**	**16577**	**94.9**
Costs	Boosting	0.96	0.0027	**16.6**	0.342
	Forest	0.95	0.0035	20.19	0.372
	0.7, 40 boost	0.946	0.0056	20.59	0.659
	0.7, 40, forest	0.932	0.0042	19.41	0.662
	0.7, 40, average	0.933	0.0057	19.35	0.494
	1, 40, forest	0.96	.0036	18.21	0.558
	1, 80, forest	**0.961**	**0.0015**	17.09	**0.26**
	1, 120, forest	0.96	0.0032	18.07	0.409
Prices	Boosting	**0.977**	**0.0017**	**81.76**	**3.014**
	Forest	0.961	0.003	121.21	3.073
	0.7, 40 boost	0.929	0.0063	202.99	7.278
	0.7, 40, forest	0.941	0.0056	184.44	9.078
	0.7, 40, average	0.938	0.0064	191.83	8.133
	1, 40, forest	0.971	0.0027	103.53	3.15
	1, 80, forest	0.972	0.0021	102.17	3.41
	1, 120, forest	0.97	0.0033	104.39	3.5
Systolic	Boosting	0.432	0.0134	6.02	0.055
	Forest	0.464	0.0073	5.81	0.043
	0.3, 80 boost	0.475	0.0077	5.75	0.044
	0.3, 80	0.482	0.0084	5.79	0.05
	0.3, 80, average	**0.486**	**0.0064**	**5.75**	**0.033**

CONCLUSIONS

A new ensemble regression method is presented. An additional substantiation of the approach is given, which refines the conclusion drawn in previous works. The results of testing the method and evaluating the possibility of its use to solve applied problems in business and medicine are also considered. The results of the experiments confirm the promise of the proposed approach.

Further research is expected to focus on increasing the speed of the method, which can be achieved, first, by directly constructing the elements B_k as trees with a special quality functional, and second, by optimizing the construction of these trees, including using the cython library.

FUNDING

This work was supported by the state task, project 0063-2019-0003 with the help of the infrastructure of the Informatika Center for Collective Use of the Federal Research Center “Computer Science and Control,” Russian Academy of Sciences [1].

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Regulations on the "Informatics" common use center. http://www.frccsc.ru/ckp. Accessed February 14, 2023.
2. Z. H. Zhou, *Ensemble Methods: Foundations and Algorithms* (Chapman and Hall/CRC, New York, 2012).
3. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction* (Springer, New York, 2009).
4. L. Breiman, "Random forests," Mach. Learning **45** (1), 5–32 (2001).
5. R. E. Schapire and Y. Freund, *Foundations and Algorithms* (MIT Press, Cambridge, Mass., 2012).
6. T. K. Ho, "The random subspace method for constructing decision forests," IEEE Trans. Pattern Anal. Mach. Intell. **20** (8), 832–844 (1998).
7. N. Garcia-Pedrajas and D. Ortiz-Boyer, "Boosting random subspace method," Neuron Networks **21** (9), 1344–1362 (2008).
8. Yu. I. Zhuravlev, O. V. Senko, A. A. Dokukin, N. N. Kiselyova, and I. A. Saenko, "Two-level regression method using ensembles of trees with optimal divergence," Dokl. Math. **103**, 1–4 (2021).
9. F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in Python," Mac. Learn. Res. **12**, 2825–2830 (2011).
10. D. H. Wolpert, "Stacked generalization," Neuron Networks **5** (2), 241–259 (1992).
11. E. M. Braverman and I. B. Muchnik, *Structural Methods for Processing Empirical Data* (Nauka, Moscow, 1983) [in Russian].
12. O. V. Senko, A. A. Dokukin, N. N. Kiselyova, V. A. Dudarev, and Yu. O. Kuznetsova, "New two-level ensemble method and its application to chemical compounds properties prediction," Lobachevskii J. Math. **44** (1), 188–197 (2023).
13. M. H. Rafiei and H. Adeli, "A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units," J. Constr. Eng. Manage. **142** (2) (2015).
14. O. V. Sen'ko, V. Ya. Chuchupal, and A. A. Dokukin, "Noninvasive blood pressure assessment with CardioQvark," Mat. Biol. Bioinf. **2** (12), 536–546 (2017).
15. F. Mostofi, V. Toğan, and H. B. Basağa, "Real-estate price prediction with deep neural network and principal component analysis," Organ. Technol. Manage. Constr. **14** (1), 2741–2759 (2022).