UDC 519.254:519.237.8

O. Matsuga, B. Nashylnyk
Oles Honchar Dnipro National University

DECISION TREE ENSEMBLE CONSTRUCTING ALGORITHMS AND THEIR COMPARATIVE ANALYSIS

Ансамблі дерев рішень вважають одними з найбільш ефективних засобів класифікації. У літературі запропоновано значну кількість алгоритмів для їх побудови, проте на практиці актуальним лишається питання вибору алгоритму, який забезпечує найвищу точність класифікації. У зв'язку з цим у роботі було поставлено задачу порівняти точність класифікації ансамблями дерев рішень, побудованими найбільш популярними та потужними алгоритмами.

Для виконання цієї задачі було створено програмне забезпечення, в якому реалізовано наступні алгоритми: випадковий ліс, стекінг з використанням генетичного алгоритму, алгоритми побудови ансамблів надзвичайно випадкових дерев та досконалих випадкових дерев, а також використано open-source реалізації випадкового лісу з пакету Accord та градієнтного бустінгу з пакету XGBoost. Програмне забезпечення розроблене на мові C# у середовищі Microsoft Visual Studio у вигляді настільного додатку. За його допомогою проведено обчислювальні експерименти на 10 реальних наборах даних різних обсягів та розмірностей, взятих з UCI Machine Learning Repository.

Суть експерименту на кожному наборі даних полягала в наступному. Набір даних випадковим чином було розбито на 2 частини: початкову та тестову. Навчальну використано для побудови ансамблю з 50 дерев, а тестову – для оцінювання точності класифікації. Випадкове розбиття набору й оцінку точності класифікації було виконано 30 разів, після чого обчислено середнє значення точності класифікації та його середньоквадратичне відхилення. Тоді було проведено порівняння середніх за допомогою двохвибіркового *t* тесту на рівні значущості 0,05.

За результатами обчислювальних експериментів найкращий результат показав алгоритм побудови ансамблю надзвичайно випадкових дерев: побудований ансамбль мав найвищу точність класифікації на 7 наборах даних і на більшості наборів переміг або не програв іншим ансамблям. Трохи гірші результати показали випадковий ліс та градієнтний бустинг з пакету XGBoost, які виявилися кращими на 5 та 4 наборах відповідно та не часто поступалися за точністю класифікації іншим алгоритмам. Стекінг з використанням генетичних алгоритмів, ансамбль досконалих випадкових дерев та випадковий ліс із пакету Accord показали найнижчу точність класифікацію на більшості наборів.

Ключові слова: *класифікація, дерево рішень, ансамбль дерев рішень, точність класифікації, обчислювальний експеримент.*

Проведён сравнительный анализ шести алгоритмов построения ансамблей деревьев решений. В качестве метрики выбрано точность классификации новых наблюдений. Сравнение проведено на основе результатов вычислительных экспериментов на реальных наборах данных. Согласно результатов экспериментов наивысшую точность классификации обеспечил алгоритм построения чрезвычайно случайных деревьев.

Ключевые слова: *классификация, дерево решений, ансамбль деревьев решений, точность классификации, вычислительный эксперимент.*

A comparative analysis of six algorithms that construct a decision tree ensemble was carried out. The classification accuracy for new data was chosen as a metric. The comparison was based on the computational experiments results with real datasets. The Extra-Trees algorithm provided the best result according to the experiments results.

Keywords: *classification, decision tree, decision tree ensemble, classification accuracy, computational experiment.*

Introduction and problem definition. The classification task is one of the most important data mining tasks. It arises in such areas as commerce, marketing, medicine, biology and many others. The classification task concerns constructing a classifier for the prediction to which of a set of categories (classes) a new observation belongs. A training dataset containing observations whose category membership is known must be necessarily given in order to construct the classifier.

Decision tree ensemble is one of the most popular tools for solving the classification task. Many algorithms for constructing such ensembles have been developed. But the problem of choosing the algorithm with the best accuracy is very topical and important in practice. In this connection the work is aimed at the comparison of the new observations classification accuracy provided by decision tree ensembles constructed by different algorithms.

Analysis of recent researches and publications. Decision tree ensemble is a special case of a classifier ensemble in which decision trees are used as basic classifiers. The main idea of the classifier ensemble is to combine a set of basic classifiers in order to make more accurate predictions than any single classifier is able to [1]. It is known that ensembles can improve the accuracy, stability and robustness of basic classifiers. But the following properties should be taken into account while constructing ensemble: the correlation and diversity in basic classifiers, the number of classifiers in the ensemble, the method of combination of basic classifiers predictions and the portion of the original dataset, used to train a certain classifier, is also used to train other classifiers [1].

Some techniques have been developed to construct classifier ensembles including decision tree ensembles: bagging, boosting, stacking [1].

Bagging (bootstrap aggregating) was proposed by Leo Breiman in 1994 [2]. It reduces the variance of classifiers and helps avoid overfitting. In bagging each classifier (typically decision tree) in the ensemble is trained on a random sample with replacement obtained from the training dataset (bootstrap sample). The bootstrap sample has the same cardinality as the original training set, but many of the original examples may be repeated in the bootstrap sample while others may be left out. The predictions of the trained classifiers are combined by majority voting. The Random Forest is an extension over bagging with decision trees as basic classifiers [3]. It differs from the bagging in one way: it uses a modified tree learning algorithm in which a random subset of features (instead of all features) is used while splitting each tree node. This random selection of features allows to reduce the correlation in trees in the ensemble and thereby leads to better ensemble performance.

As an extension of the Random Forest idea the Extra-Trees algorithm that constructs an extremely randomized tree ensemble [4] and the algorithm that constructs a perfect random tree ensemble [5] can be considered. Both of them combine decision trees in ensemble by majority voting and build each tree on the entire dataset but randomly.

Boosting is another ensemble technique that sequentially constructs “weak” classifiers (typically decision trees) each of which tries to correct the errors of the previous ones. Then the “weak” classifiers are combined in a “strong” one by weighted voting [1]. This technique reduces the bias of classifiers. One of well-known boosting algorithms is AdaBoost (Adaptive Boosting) developed by Robert Schapire and Yoav Freund in 1996. Each “weak” classifier is trained on a random sample obtained from the training dataset using examples weights. These weights change in each iteration of the algorithm: they increase for examples that are misclassified and decrease for examples that are correctly classified. Thus, the subsequent classifier is trained to correctly predict the examples misclassified by the previous ones. A more effective extension over boosting is gradient boosting that was proposed in 1999 but was published officially later [6]. Its idea is that boosting can be interpreted as an optimization algorithm on a suitable cost function. In [6] it is introduced as an iterative functional gradient descent algorithm. The gradient boosting is implemented in

XGBoost [7], LightGBM and CatBoost libraries which have helped win various machine learning competitions many a time. Their effectiveness can be explained by the following key properties: work speed, possibility of paralleling calculations and the video card power use, scalability for big data and various optimizations of the original algorithm.

Stacking (1992) is a technique of combining classifiers, that introduces a meta-learner concept. The stacking algorithm involves the following steps [1]: 1) split the training set into two disjoint sets; 2) train several basic classifiers on the first part; 3) test the basic classifiers on the second part; 4) use the predictions from the third step as the inputs and the correct responses as the outputs, train a higher level classifier. In [6] authors proposed a modified stacking algorithm that uses a genetic algorithm (GA) to create an ensemble and improve the performance of the standard stacking algorithm [8].

Based on the carried out survey the most popular, widely used and well-functioning (according to other experimental results) decision tree ensemble constructing algorithms were chosen for comparison:

– Random Forest, based on bagging [3];

– Extra-Trees algorithm [4];

– algorithm that constructs a perfect random tree ensemble [5]

– gradient boosting and its implementation from XGBoost library [6, 7];

– GA-based stacking ensemble algorithm [8].

Task definition. The task is to carry out computational experiments to compare the chosen decision tree ensemble constructing algorithms with regard to the classification accuracy.

Main material. To perform the computational experiments own software was developed in the Visual Studio on the C# programming language. The software is a desktop application which allows to construct ensembles using the Random Forest, Extra-Trees algorithm, algorithm that constructs a perfect random trees ensemble, GA-based stacking ensemble algorithm and also using open-source implementations of the Random Forest (Accord.NET Framework [9]) and gradient boosting (XGBoost library [7]). It has user friendly graphical interface and allows a user to

– load samples of any size and dimension,

– construct ensembles of as many trees as a user wishes,

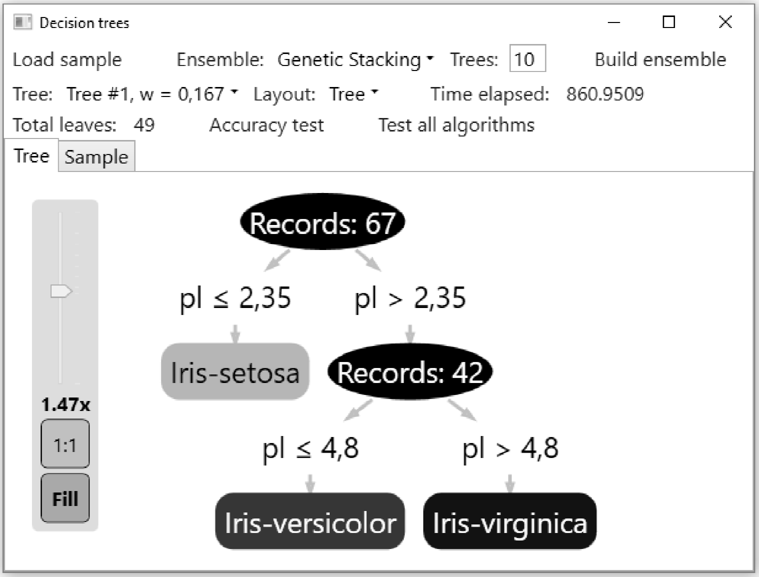
– view constructed trees,

– analyze ensemble constructing time and the total number of leaves,

– estimate the classification accuracy using a test set or cross-validation.

The software also allows comparing the mean classification accuracy over randomly partitions of a dataset for different algorithms using the two-sample *t* test [10].

The main program window is shown in Figure 1.



Ensemble: Genetic Stacking
Trees: 10
Tree: Tree #1, w = 0,167
Layout: Tree
Time elapsed: 860.9509
Total leaves: 49

Figure 1 – The main program window

In order to compare algorithms implemented in the software, we carried out computational experiments on 10 real datasets with different sizes and dimensions using the created software. All the datasets were taken from the UCI Machine Learning Repository [11].

The experiment for each dataset and algorithm was conducted in this way. The dataset was randomly split into two sets: training and test. An ensemble of 50 trees was learned on the training set using the algorithm and the classification accuracy was estimated on the test set. To reduce the variability arising due to random splitting and randomness in the algorithm this process was repeated 30 times and the results were averaged over the splits. The mean classification accuracy over 30 splits and standard deviation of the mean were calculated.

Then a statistical test to compare the mean classification accuracy of the different algorithms was performed. For this purpose, the two-sample two-sided *t* test with a 95% confidence level was used.

The mean classification accuracy obtained for each algorithm and dataset is given in Table 1. The algorithms in the table are denoted as follows: RF is the implementation of the Random Forest, ARF is the Random Forest implementation from the Accord.NET Framework, ET is the Extra-Trees algorithm, GS is the GA-based stacking ensemble algorithm, PERT is the algorithm that constructs a perfect random tree ensemble, XGB is the gradient boosting implementation from the XGBoost library. The datasets characteristics are denoted as follows: *n* – training set size; *N* – full dataset size; *k* – number of attributes (discrete and continuous); *c* – number of classes. The maximum mean classification accuracy and the accuracies that equal it according to the two-sample *t* test are marked bold in Table 1.

Table 1

Mean classification accuracy ± standard deviation of mean

№	*n*/*N*/*k*/*c*	RF	ARF	ET	GS	PERT	XGB
1	200/351/34/2	**0,934 ± 0,015**	**0,932 ± 0,018**	**0,926 ± 0,02**	0,913 ± 0,022	0,909 ± 0,019	0,924 ± 0,021
2	1000/15000/8/2	0,972 ± 0,002	0,968 ± 0,003	**0,973 ± 0,002**	0,965 ± 0,003	0,963 ± 0,006	0,97 ± 0,002
3	1000/40212/26/2	0,891 ± 0,003	0,886 ± 0,005	**0,893 ± 0,002**	0,887 ± 0,004	0,883 ± 0,001	**0,892 ± 0,003**
4	300/569/30/2	0,957 ± 0,012	0,941 ± 0,015	**0,965 ± 0,01**	0,945 ± 0,012	0,952 ± 0,011	0,956 ± 0,012
5	100/178/13/3	**0,971 ± 0,015**	0,926 ± 0,028	**0,971 ± 0,017**	0,938 ± 0,027	**0,976 ± 0,016**	0,963 ± 0,021
6	500/1728/6/4	0,882 ± 0,014	0,939 ± 0,011	0,944 ± 0,01	0,87 ± 0,013	0,867 ± 0,015	**0,956 ± 0,011**
7	100/150/4/3	**0,949 ± 0,028**	**0,946 ± 0,028**	**0,951 ± 0,029**	**0,953 ± 0,024**	**0,955 ± 0,022**	**0,946 ± 0,025**
8	200/303/13/5	**0,574 ± 0,041**	0,532 ± 0,04	0,54 ± 0,035	**0,572 ± 0,038**	**0,574 ± 0,046**	0,553 ± 0,04

9	100/121/ 49/2	**0,725 ± 0,073**	0,656 ± 0,114	0,679 ± 0,098	**0,705 ± 0,069**	0,686 ± 0,097	**0,74 ± 0,081**
10	500/1372/ 4/2	0,986 ± 0,005	0,971 ± 0,01	**0,995 ± 0,004**	0,978 ± 0,007	**0,996 ± 0,003**	0,987 ± 0,006

Table 2 reports on the "win/draw/loss" statuses of all pairs of algorithms by the results of the *t* test performing. In each pair the algorithm had the "win" status if its mean classification accuracy was greater than that of another algorithm, the "loss" status if it was less and "draw" status otherwise.

Table 2

Amount of the "wins/draws/losses" statuses of the algorithm in the row versus the algorithm in the column according to the t test

	RF	ARF	ET	GS	PERT	XGB
RF	–	7/2/1	2/3/5	7/3/0	4/5/1	3/6/1
ARF	1/2/7	–	0/5/5	3/4/3	4/2/4	0/2/8
ET	5/3/2	5/5/0	–	7/2/1	5/4/1	3/5/2
GS	0/3/7	3/4/3	1/2/7	–	1/6/3	0/4/6
PERT	1/5/4	4/2/4	1/4/5	3/6/1	–	2/3/5
XGB	1/6/3	8/2/0	2/5/3	6/4/0	5/3/2	–

Considering the computational experiments results the Extra-Trees algorithm showed the best performance: the ensemble constructed using it had the maximum classification accuracy on 7 datasets and won or at least did not lose with respect to the ensembles constructed using other algorithms on most datasets. The Random Forest and the gradient boosting from the XGBoost library provided the maximum classification accuracy on 5 and 4 datasets respectively and very rarely lost with respect to other algorithms (Table 2). The algorithm that constructs a perfect random tree ensemble, the GA-based stacking ensemble algorithm and the Random Forest from the Accord.NET Framework often lost.

Conclusions. In the paper the comparative analysis of six decision tree ensemble constructing algorithms with regard to the classification accuracy was carried out. The computational experiments were performed on 10 real datasets with different sizes and dimensions from the UCI Machine

Learning Repository. The experiments were performed with the software developed in the Visual Studio on the C# programming language. According to the experiments results the Extra-Trees algorithm showed the best performance (the maximum classification accuracy on 7 datasets out of 10). The Random Forest and the gradient boosting from the XGBoost library also provided good classification accuracy with respect to other algorithms. The worst results were obtained using the algorithm that constructs a perfect random tree ensemble, the GA-based stacking ensemble algorithm and the Random Forest from the Accord.NET Framework.

Further researches can be directed to discover the relationship between the number of trees in the ensemble and the classification accuracy.

References

1. Rokach L., Maimon D. Data mining with decision trees. Theory and applications. 2nd ed. Singapore: World Scientific Publishing Company. 2015. 305 p.

2. Breiman L. Bagging Predictors. *Machine Learning*. 1996. Vol. 24. P. 123-140.

3. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45. P. 5-32.

4. Geurts P., Ernst D., Wehenkel L. Extremely randomized trees. *Machine Learning*. 2006. Vol. 63. P. 3-42.

5. Cutler A., Zhao G. PERT – Perfect Random Tree Ensembles. *Computing Science and Statistics*. 2001. Vol. 33. P. 490-497.

6. Friedman J.H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 2001. Vol. 29, № 5. P. 1189-1232.

7. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. P. 785-794.

8. Sikora R., Al-laymoun O. A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms. *Journal of International Technology and Information Management*. 2014. Vol. 23, Issue 1, Article 1.

9. Accord.NET Machine Learning Framework. URL: http://accord-framework.net/index.html (дата звернення: 09.09.18).

10. Бабак В.П., Білецький А.Я., Приставка О.П., Приставка П.О. Статистична обробка даних. К.: МІВВЦ, 2001. 388 с.

11. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml/ (дата звернення: 09.09.18).

Надійшла до редколегії 10.09.2018