ML-based Approach for Credit Risk Assessment Using Parallel Calculations

Lesia Hentosh[1], Yevhen Tsikalo[2], Natalya Kustra[3] and Hakan Kutucu[4]

[1]*Department of Artificial intelligence, Lviv Polytechnic National University, Lviv, 79013, Ukraine; lesia.i.mochurad@lpnu.ua*
[2]*Department of Accounting and Audit, Ivan Franko National University of Lviv, Lviv, 79008, Ukraine; yevhen.tsikalo*@lnu.edu.ua
[3]Department of Publishing Information Technologies*, Lviv Polytechnic National University, Lviv, 79013, Ukraine; Nataliia.O.Kustra*@lpnu.ua
[4]*Department of Software Engineering, Karabuk University, Karabuk, 78050, Turkey; hakankutucu@karabuk.edu.tr*

Abstract

In banks and other credit organizations, the task of credit scoring often arises when making decisions on granting loans. The last one consists of making a reasoned decision based on information about the applicant, whether she should be granted a loan, and, if so, on what terms. This paper proposes the application of parallel calculations of the Random forest algorithm when solving the credit scoring task. This approach made it possible to reduce the time of model training and dataset processing significantly. Expectedly, when applying less data, the resulting acceleration and efficiency worsen. Using only 2500 entries, the execution time of the sequential algorithm is less than the parallel algorithm. The developed software was tested on three different processors: 4-core, 8-core, and 12-core, to evaluate the parallelization quality of data pre-processing. The classification algorithm is computationally complex and time-consuming, so we obtained practically the same acceleration for processing 5000 and 10000 records. With this amount of data, the 12-core processor gave the biggest gain in time when working with 12 threads. As a result, it is possible to have an acceleration of more than 6. This efficiency indicator of the proposed parallel algorithm can be significantly improved by varying the number of threads and considering the current trends in developing the multi-core architecture of computing systems. Also, using data without pre-processing, the following evaluation metrics were obtained: AUC=0.9 and Precision=0.845, and using data after pre-processing, these metrics were: AUC=0.86, Precision=0.89.

Keywords

Credit scoring, Random forest, classification task, parallel algorithm, acceleration.

CITRisk'2022: 3nd International Workshop on Computational & Information Technologies for Risk-Informed Systems, January 12, 2023, Neubiberg, Germany
EMAIL: *lesia.i.mochurad@lpnu.ua* (LHentosh), *Nataliia.O.Kustra@lpnu.ua* (N.Kustra), *yevhen.tsikalo@lnu.edu.ua* (Ye.Tsikalo), *hakankutucu@karabuk.edu.tr* (H.Kutucu)
ORCID: 0000-0002-4957-1512 (L.Hentosh), 0000-0002-3562-2032 (N.Kustra), 0000-0001-8051-9299 (Ye.Tsikalo), 0000-0001-7144-7246 (H.Kutucu)

1. Introduction

To date, a significant problem for many banking institutions is the non-repayment of loans at 60-65% of borrowers [1, 2]. As a result, the number of credit risks also increases risks to income and capital due to the inability of the party who has assumed obligations to fulfill the terms of any financial agreement with the bank or otherwise fulfill the obligations assumed. There are many methods for credit risk assessment: credit assessment methods, decision tree methods, rating methods, Monte Carlo method, scoring, taxonomic analysis, stress testing, etc. [3, 4]. The last is widely used in Ukraine.

In this work, we will look at the credit scoring method. In turn, we understand credit scoring as a classifier problem that determines whether a loan should be granted to a borrower based on the machine learning method [5]. Banking systems already use their own scoring systems, and they analyze the risks for their credit portfolio based on them. Many experts use neural networks and support vector machines to create such a system. The solution proposed in this work is based on the Random forest algorithm. This method is based on the construction of a large number (ensemble) of decision trees, each of which is built on a sample obtained from the original training sample using bootstrap (sampling with return) [1, 6]. Objects are classified by "voting." Each tree assigns a classified object to one of the classes and the class for which the most trees voted wins. Each decision tree does not give very high classification accuracy, but the result is true due to large number of trees.

A common problem of classification algorithms, including the Random forest algorithm, is wasting time due to processing large volumes of data. The solution proposed in this work is the use of parallel computing. This solution will be used for data pre-processing and training the model itself. Parallel computing makes programs run faster as more processes or threads are used [7]. Various studies have already demonstrated the implementation of parallel calculations for the Random Forest algorithm [8].

The purpose of this work is the parallelization of the Random Forest algorithm for the analysis of risks in lending and the evaluation of the obtained efficiency coefficients.

With the development of technology, the problem of qualitative and effective classification arises since data volume grows yearly. Current datasets can contain millions, if not billions, of records. Moreover, these data have a lot of noise, and some attributes do not carry any useful information about the object at all.

So, for high-quality training of models, the data should first be processed so that they carry more content specifically for our task and then highlight their main features. A Random forest was presented as a classifier in [9]. The authors outlines an approach to improving credit score modeling using Random forests and compares Random forests with logistic regression. However, they did not analyze the problems of using this classifier to analyze large volumes of data.

Paper [10] proposes a credit scoring method that uses information from decision trees to improve the performance of logistic regression. However, this classifier gained the most popularity primarily due to its application in tracking [11].

In our work, we use this classifier for recognition/classification tasks. The advantages of randomized trees are that they are much faster to train and test than traditional classifiers (such as SVM), they reduce the variance, and they increase the accuracy of the model by averaging the previously retrained data compared to conventional decision trees.

Paper [12] develops a decision tree ensemble model using the differential sampling rate, Synthetic Minority Oversampling Technique, and AdaBoost which is a prediction framework integrating supply chain information to predict enterprise credit risk. The advantage of the

classification approach considered in this work over the method in [12] is that the model's performance primarily depends on the model itself and the amount of data. Since we are working with a limited dataset, we will use a Random forest instead of a fully connected network to maximize the performance and training speed of the model. However, the most crucial advantage of the approach proposed in the work is that the data can be pre-processed with the help of the VGG-16 neural network. It makes it possible to significantly speed up the training and resistance to overfitting the classifier. So, in the final result, we will get a classifier with a high learning speed, which can be used, for example, on modern tensor cores of the Nvidia video card for processing large datasets (more than 10^{10} records) and a relatively high resistance to outliers and retraining.

2. Proposed methodology

2.1. Statement of the task and description of the dataset

As input, the pre-trained model should classify credit risks with a value from a binary set. The dataset [13] consists of 600 records with 10 features about customers of a German bank (see Figure 1). There are 2 types of data: object, int64. Attributes: Age, Job, Credit Amount, Duration – int64; Sex, Housing, Savings, Checking accounts, Purpose, Risk – object.

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	NaN	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad
5	35	male	1	free	NaN	NaN	9055	36	education	good
6	53	male	2	own	quite rich	NaN	2835	24	furniture/equipment	good
7	35	male	3	rent	little	moderate	6948	36	car	good
8	61	male	1	own	rich	NaN	3059	12	radio/TV	good
9	28	male	3	own	little	moderate	5234	30	car	bad

Figure 1: Initial dataset before pre-processing

2.2. Data pre-processing

In order to improve the quality of predictions, the data can be processed in a certain way so that the necessary features are highlighted. Let us start by considering the features of this dataset and the quantitative distribution of good and bad borrowers (see Figure 2).

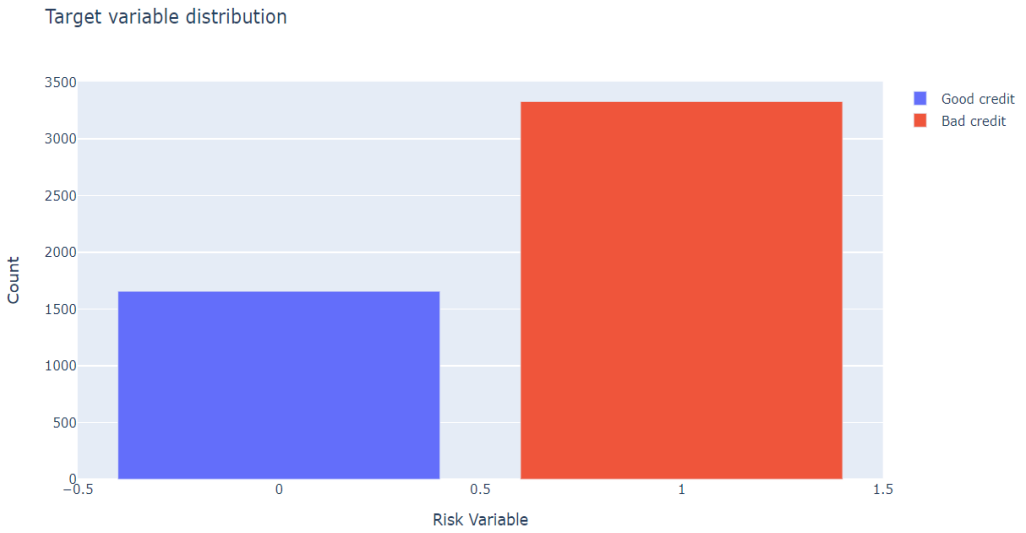

Figure 2: Quantitative ratio of good/bad bank customers

Figure 3 shows how the age distribution of borrowers affects credit repayment/non-repayment. A descriptive statistic of age distribution is a measure of central tendency; the distribution deviates from normal in these cases.

Figure 3: Distribution of borrowers by age

If we check the normality of this distribution on the QQ-Plot, we get Figure 4. We can also see deviations in the 2nd and 4th quartiles.

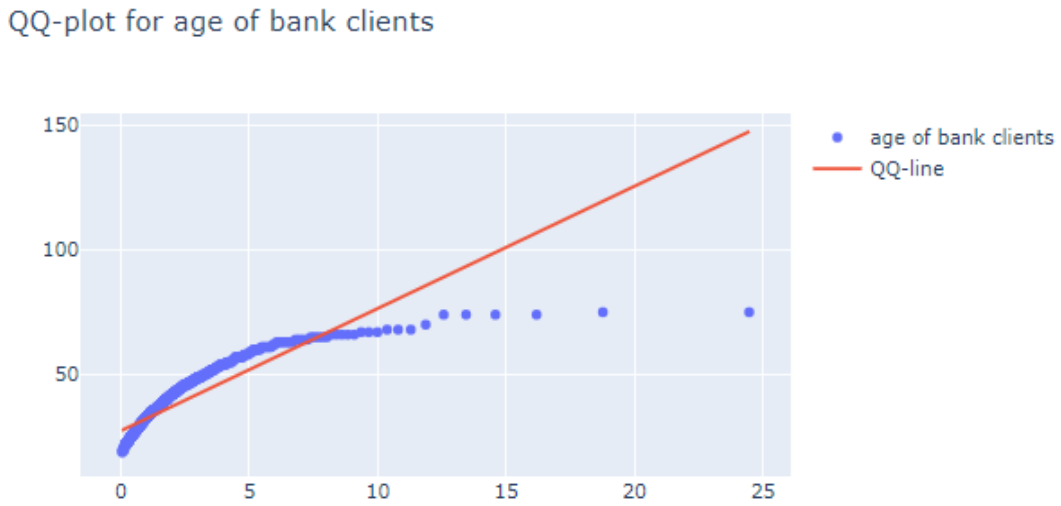

Figure 4: Deviation of the distribution of borrowers by age from normal

We will visualize the credit amount distribution based on the housing availability at the bank's client. The visualization of distributions in Figure 5 shows that borrowers borrow more significant amounts if they do not have their own homes.

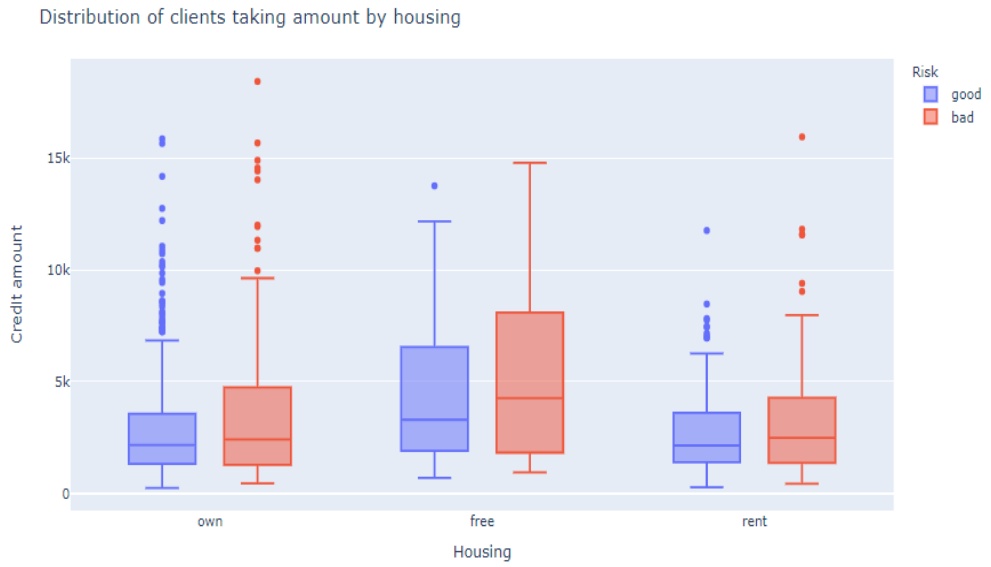

Figure 5: Distribution of loan amounts based on the availability of housing for an honest/dishonest client

The next process that the data goes through before classification is the selection of features of objects when passing through the convolutional, activation, and pooling layers of the pre-trained VGG-16 neural network. These features are transferred to the input of the classifier. Let us clarify

that the characteristics of the object and the corresponding label – class – "risky loan", and "risk-free loan" are submitted to the input of the Random forest classifier.

2.3. Consistent and proposed parallel algorithm with computational complexity analysis

Consistent algorithm Random forest [14].

Background: training set $S := (x_1, y_1), \dots, (x_n, y_n)$, attributes F and the number of trees in the forest B

function RandomForest (S, F)
1 $H \leftarrow \emptyset$
2 for $i \in 1, \dots, B$ do
3 $S^{(i)} \leftarrow A$ subset of attribute objects from S
4 $h_i \leftarrow$ RandomizedTreeLearn $(S^{(i)}, F)$
5 $H \leftarrow H \cup \{h_i\}$
6 end for
7 return H
8 end function

function RandomizedTreeLearn (S, F)
1 for each tree node:
2 $f \leftarrow$ a very small subset of the attributes of F
3 Division by the best function in f (gini,entropy)
4 returning the learned tree
5 end for
6 end function

Parallel algorithm Random forest

omp_set_num_threads(NUM_THREADS);
1 function RandomForest (S, F)
2 $H \leftarrow \emptyset$
3 #pragma omp parallel private $(i,\ s_i,\ h_i)$ shared(H)
4 {
5 #pragma omp for
6 for $i \in 1, \dots, B$ do
7 $S^{(i)} \leftarrow A$ subset of objects from S
8 $h_i \leftarrow$ RandomizedTreeLearn $(S^{(i)},\ F)$
9 $H \leftarrow H \cup \{h_i\}$
10 end for
11 }
12 return H
13 end function

For classification tasks, it is advisable to establish $f = \sqrt{d}$. We usually take d – to be the number of functions for regression problems. It is recommended to build each tree until all its leaves will contain only n_min=1 examples of classifying and n_min=5 examples for regression.

Analysis of the complexity of sequential and parallel algorithms:

Sequential: Complexity_sequential = $O(T * n^2 * \sqrt{p})$

Parallel: Complexity_parallel = $O\left(\frac{T*n^2*\sqrt{p}}{N_{threads}}\right)$, where:

1) T is the number of trees to be built;
2) $\sqrt{p}$ isthe number of features that are taken into account at each node of the tree;
3) n^2 is the number of tree nodes*the number of partitions of the value of the variable;
4) $N_{threads}$ isthe number of threads allocated for building trees.

3. Research results

These are the results of the implementation of the proposed parallel algorithm. These results were obtainedon 3 different processors: 4-core, 8-core, and 12-core. Python language and Joblib library [15] were chosen for algorithmimplementation. Joblib is optimized for the fast and reliable use of big data. By default, Joblib uses the 'loky' internal module [16] to run separate Python worker processes to execute tasks concurrently on different CPUs. it is possible to unlock the Python Global Interpreter Lock (GIL) for most of your computations. In that case, this technology allows the use of multithreading, which can significantly increase execution speed. Tables 1-3 present the execution time of the proposed parallel algorithm on a 4, 8, and 12-core processor.

Table 1

Execution time of the parallel algorithm on a 4-core processor, ms

Number of records, n	The number of threads			
	1	2	4	8
2500	663.35	424.5	301.0	308.2
5000	1326.7	849.1	602.0	616.3
10000	2669.4	1714.1	1220.0	1248.6

Table 2

Execution time of the parallel algorithm on an 8-core processor, ms

Number of records, n	The number of threads			
	1	2	4	8
2500	572.6	350.8	241.7	190.1
5000	1123.2	679.5	461.4	358.2
10000	2230.4	1343.0	906.8	700.3

Table 3

Execution time of the parallel algorithm on a 12-core processor, ms

Number of records, n	The number of threads			
	1	3	6	12
2500	591.5	217.7	127.3	92.6
5000	1180.2	432.4	251.7	182.1
10000	2365.9	869.1	508.0	369.7

It can be seen from Tables 1-3 that the best result can be obtained by the parallel algorithm in the case when the number of threads is equal to the number of cores of the multiprocessor computer system, which indicates the reliability of the obtained results. Furthermore, even with12 threads on a 12-core processor, it was possible to reduce the calculation time by more than 6 times (see Table 4).

Table 4

Indicators of acceleration and efficiency of the software implementation of the parallel algorithm on a 12-core processor

Number of records, n	3 threads		6 threads		12 threads	
	Acceleration	Efficiency	Acceleration	Efficiency	Acceleration	Efficiency
2500	2.71	0.53	4.64	0.77	6.38	0.90
5000	2.73	0.54	4.68	0.78	6.48	0.91
10000	2.72	0.53	4.65	0.77	6.39	0.90

There are various ways to improve efficiency [17]. As seen from Table 4, with an increase in the number of threads, the efficiency approaches unity, which also indicates the optimization of the algorithm by taking into account such properties as multithreading and multi-core of modern personal computers.

Next, in Figures 6-7, we will present data classification results using the trained Random Forest model.

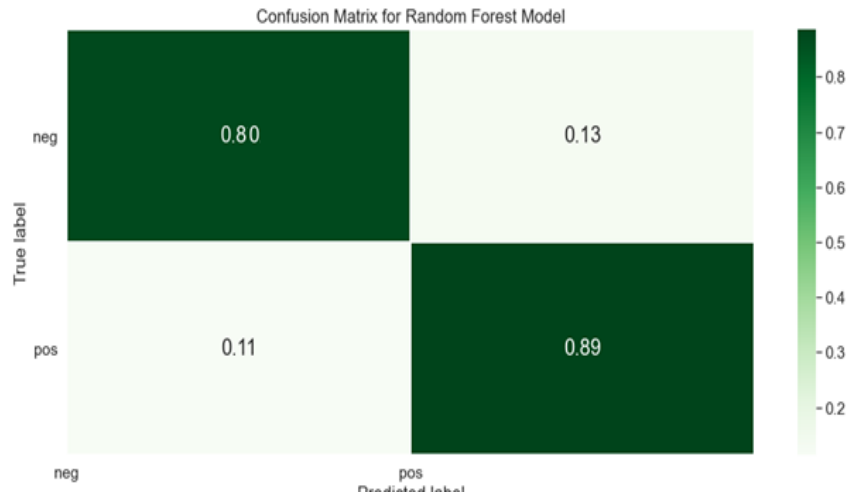

Figure 6: Error matrix for data without pre-processing

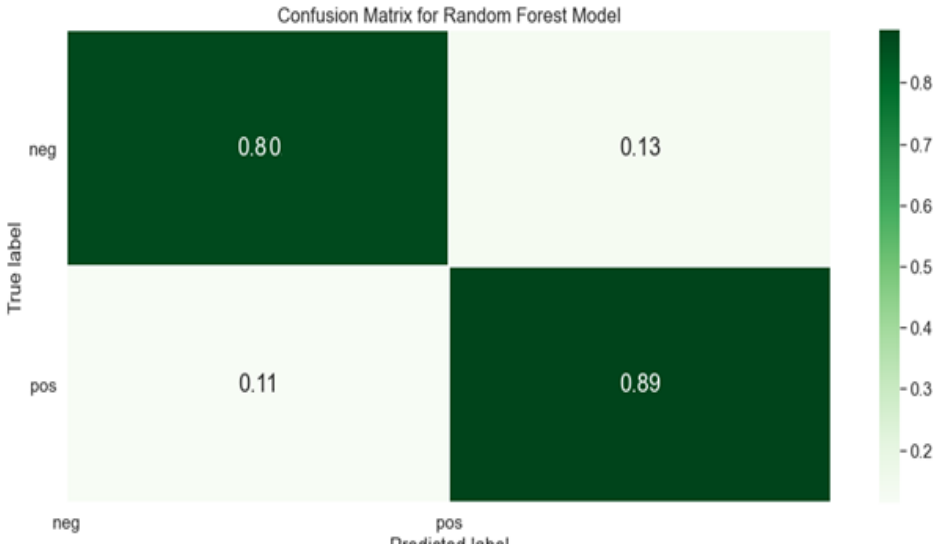

Figure 7: Error matrix for data after pre-processing

To see more clearly (see Figure 8) how the parallel classifier works, let us have an example: A 24-year-old skilled worker with his own home and an average checking account, who wants to buy a radio/TV, the probability of his good faith is $\Pr(Y = 1|X; W) = 0.824$. Another example is a 42-year-old skilled worker without housing with a small checking account who wants to buy a car, the likelihood of a credit $\Pr(Y = 1|X; W) = 0.45$.

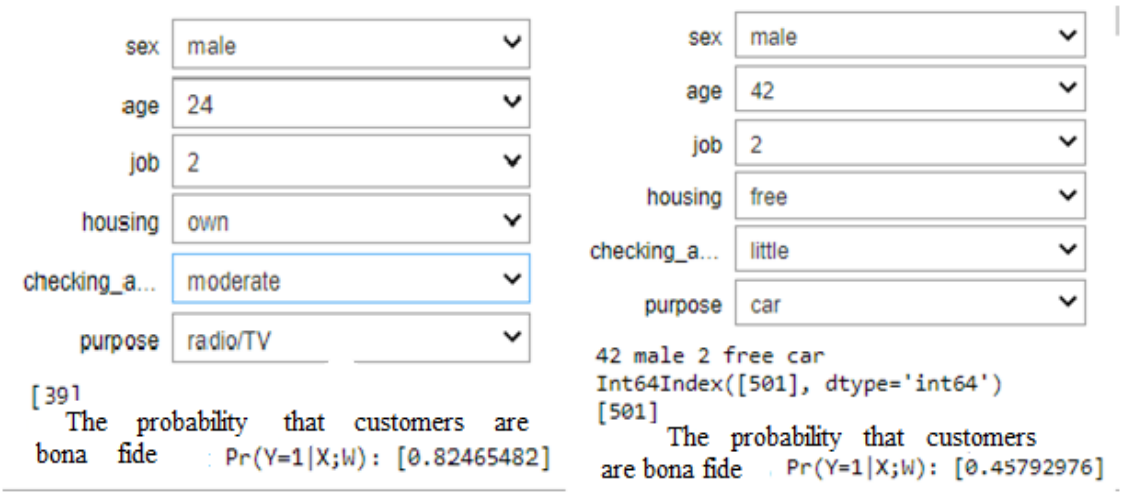

Figure 8: Test examples of the classifier program

Let us introduce a few more indicators that will allow us to evaluate our model better (see Figure 9).

Precision is the ratio $\text{precision} = \text{tp} / (\text{tp} + \text{fp})$, where tp is the number of true positive elements, and fp is the number of false positives. Precision is intuitively the ability of a classifier not to flag a sample as positive if it is negative.

The recall is the ratio $\text{recall} = \text{tp} / (\text{tp} + \text{fn})$, where tp is the number of true positives and fn is the number of false negatives. The recall is intuitively the ability of the classifier to find all positive samples.

The F-beta score can be interpreted as the weighted harmonic value of precision and recall when the F-beta score reaches its best value at 1 and worst at 0. The F-beta score considers recall more than precision, the beta coefficient. beta = 1.0 means that recall and precision are equally important.

Support is the number of cases of each class in y_test.

	precision	recall	f1-score	support
0	0.70	0.63	0.67	63
1	0.63	0.70	0.67	57
avg / total	0.67	0.67	0.67	120

	precision	recall	f1-score	support
0	0.78	0.75	0.76	63
1	0.73	0.77	0.75	57
avg / total	0.76	0.76	0.76	120

Figure 9: Performance indicators of the model

The time of parallel training on data after pre-processing on a 4-core processor is presented in Figure 10. Table 5 shows the acceleration of the model training result using a parallel algorithm on a 4-core processor.

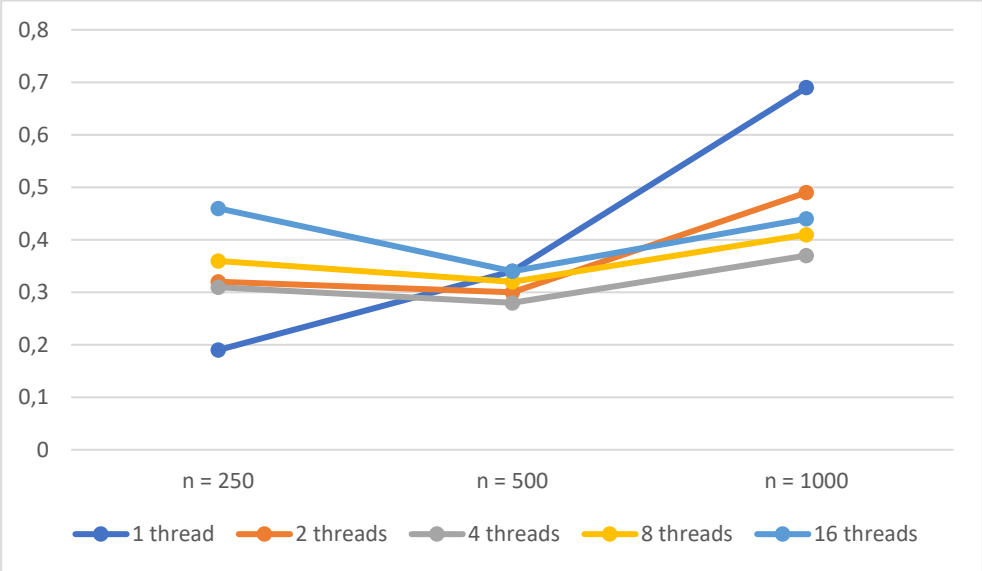

Figure 10: Parallel training on data after pre-processing, sec

Table 5

Acceleration of parallel training on data after pre-processing on a 4-core processor

Number of records, n	2 threads	4 threads	8 threads	16 threads
2500	0.76	0.81	0.76	0.68
5000	1.39	1.58	1.48	1.39
10000	1.57	1.89	1.68	1.57

So, parallelization of Random forest algorithm training on a 4-core processor showed good results for a large sample of data. We obtained high acceleration with 2 and 4 threads, and predictably the acceleration was degraded with 8 and 16 threads.

4. Conclusion

The work demonstrated the use of parallel computing in typical machine learning algorithms, such as Random forest. This approach made it possible to reduce the time of model training and dataset processing significantly. From the numerical results, it can be seen that the increase in productivity strongly depends on the architecture of the specific computer on which the code is executed. However, this research is very relevant based on current trends in developing the multi-core architecture of computer systems. It is possible to get an acceleration of more than 6 times.

The effectiveness of the Random forest classification algorithm in the credit scoring problem was also demonstrated. The numerical values of the area under the AUC - ROC curve for data before pre-processing was 0.80, and for data after pre-processing was 0.9.

The algorithm for classifying features of objects proposed in the work made it possible to significantly improve the accuracy of calculations compared to the work [9].

Using the data without pre-processing, we obtained the following indicators:

- $TN = 0.80$;
- $TP = 0.89$;
- $AUC = 0.9$;
- $Precision = 0.845$.

Using data after pre-processing, these indicators were:

- $TP = 0.94$;
- $TN = 0.84$;
- $AUC = 0.86$;
- $Precision = 0.89$.

Therefore, it can be concluded that the proposed algorithms have successfully coped with the task. The work also demonstrates how one of the essential programming languages in machine learning, Python, can be combined with parallel computing in the Joblib library for significant optimization of sequential algorithms.

References

[1] O.Artemieva, A Bestiuk, The influence of the credit risks on the banking system of Ukraine in conditions of transformation processes, Scientific Bulletin of KSU, Series «Economic Sciences», 37, 2020. DOI:10.32999/ksu2307-8030/2020-37-11

[2] R.Flunger, A.Mladenow, C.Strauss, Game Analytics—Business Impact, Methods and Tools, Studies in Systems, Decision and Control, 377, 2022, pp. 601–617

[3] Qiang Cai and Qian Qian, Summary of Credit Risk Assessment Methods, Advances in Intelligent Systems Research, 148, 2018, pp. 89-93

[4] S.Moradi, F.Mokhatab Rafiei, A dynamic credit risk assessment model with data mining techniques: evidence from Iranian banks. Financ Innov 5, 15, 2019. DOI:10.1186/s40854-019-0121-9

[5] M.Sipper, J.H.Moore, Conservation machine learning: a case study of random forests, Sci Rep 11, 3629, 2021. DOI:10.1038/s41598-021-83247-4

[6] R.Tkachenko, Z.Duriagina, I.Lemishka, I.Izonin, Trostianchyn Development of machine learning method of titanium alloy properties identification in additive technologies, Eastern-European Journal of Enterprise Technologies, 3(12 (93)), 2018

[7] L.Mochurad, A.Ilkiv, A novel method of medical classification using parallelization algorithms. International scientific journal «Computer systems and imformation technologies», 1, 2022. DOI:10.31891/CSIT-2022-1-3

[8] N.Azizah, L.S.Riza, Y.Wihardi, Implementation of random forest algorithm with parallel computing in R. Journal of Physics: Conference Series, Volume 1280, Issue 2, 2019. DOI:10.1088/1742-6596/1280/2/022028

[9] Sh.Dhruv, Improving the Art, Craft and Science of Economic Credit Risk Scorecards Using Random Forests: Why Credit Scorers and Economists Should Use Random Forests (June 9, 2011). DOI:10.2139/ssrn.1861535

[10] E.Dumitrescu, S.Hué, Ch.Hurlin, S.Tokpavi, Machine Learning or Econometrics for Credit Scoring: Let's Get the Best of Both Worlds, 2020. URL: https://hal.archives-ouvertes.fr/hal-02507499v2
[11] D.David, Random Forest Classifier Tutorial: How to Use Tree-Based Algorithms for Machine Learning. AUGUST 6, 2020. URL: https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/
[12] X.Hu, T.Zhou, Y.Zhang, Enterprise credit risk prediction using supply chain information: A decision tree ensemble model based on the differential sampling rate, Synthetic Minority Oversampling Technique and AdaBoost, Expert Systems, 39, 6. DOI:10.1111/exsy.12953
[13] Analysis of German Credit Data. URL: https://online.stat.psu.edu/stat508/resource/analysis/gcd
[14] F.Neaz, H.Tawqir, A.Marium, T.Granthi, SH.Dev Sana, Personnel security system of nuclear power plants using machine learning for psychological, behavioral and social media activity analysis, Bachelor of science in computer science and engineering, 2018, 43 p.
[15] X.Li, Yu Wang, S.Basu, K.Kumbier, B.Yu, A Debiased MDI Feature Importance Measure for Random Forests, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada. DOI:10.48550/arXiv.1906.10845
[16] A.Malakhov, D.Liu, A.Gorshkov, T.Wilmarth,. Composable Multi-Threading and Multi-Processing for Numeric Libraries, 2018, pp.18-24. DOI:10.25080/Majora-4af1f417-003
[17] Embarrassingly parallel for loops. URL: https://joblib.readthedocs.io/en/latest/parallel.html
[18] N.Boyko, K.Shakhovska, L.Mochurad, J.Campos, Information system of catering selection by using clustering analysis, CEUR Workshop Proceedings, 2533, 2019, pp. 94–106