UDC 519.6
https://doi.org/10.31891/CSIT-2022-1-3

LESIA MOCHURAD, ANDRII ILKIV
Lviv Polytechnic National University

A NOVEL METHOD OF MEDICAL CLASSIFICATION USING PARALLELIZATION ALGORITHMS

Methods of machine learning in the medical field are the subject of significant ongoing research, which mainly focuses on modeling certain human actions, thought processes or disease recognition. Other applications include biomedical systems, which include genetics and DNA analysis. The purpose of this paper is the implementation of machine learning methods – Random Forest and Decision Tree, further parallelization of these algorithms to achieve greater accuracy of classification and reduce the time of training of these classifiers in the field of medical data processing, determining the presence of human cardiovascular disease. The paper conducts research using machine learning methods for data processing in medicine in order to improve the accuracy and execution time using parallelization algorithms. Classification is an important tool in today's world, where big data is used to make various decisions in government, economics, medicine, and so on. Researchers have access to vast amounts of data, and classification is one of the tools that helps them understand data and find certain patterns in it. The paper used a dataset consisting of records of 70000 patients and containing 12 attributes. Analysis and preliminary data preparation were performed. The Random Forest algorithm is parallelized using the sklearn library functional. The time required to train the model was reduced by 4.4 times when using 8 parallel streams, compared with sequential training. This algorithm is also parallelized based on CUDA. As a result, the time required to train the model was reduced by 83.4 times when using this technology on the GPU. The paper calculates the acceleration and efficiency coefficients, as well as provides a detailed comparison with a sequential algorithm.

Key words: *machine learning method, random forest algorithm, CUDA technology, acceleration, efficiency factor.*

ЛЕСЯ МОЧУРАД, АНДРІЙ ІЛЬКІВ
Національний університет «Львівська політехніка»

НОВІТНІЙ МЕТОД МЕДИЧНОЇ КЛАСИФІКАЦІЇ З ВИКОРИСТАННЯМ АЛГОРИТМІВ ПАРАЛЕЛІЗАЦІЇ

Методи машинного навчання в медичній галузі є предметом значних постійних досліджень, які в основному концентруються на моделюванні деяких людських вчинків, процесів мислення або розпізнаванні захворювань. Інші галузі застосування – це біомедичні системи, які включають генетику та аналіз ДНК. У роботі проведено дослідження з використанням методів машинного навчання для обробки даних в медицині з метою покращенням точності та часу виконання за допомогою алгоритмів розпаралелення. Класифікація є важливим інструментом у сучасному світі, де робота з великими даними використовується для прийняття різного роду рішень в уряді, економіці, медицині, тощо. Одним із методів навчання для класифікації є випадковий ліс. Використання останнього може призвести до значного покращення точності прогнозування, тобто, кращої здатності прогнозувати нові випадки даних. Відсутність необхідності надавати конкретний алгоритм ідентифікації хвороби представляє велику перевагу перед застосуванням методів машинного навчання. Дослідники мають доступ до величезних обсягів даних, і класифікація є одним із інструментів, який допомагає їм зрозуміти дані та знайти певні закономірності у них. У роботі використано датасет, який складається з записів про 70000 пацієнтів й містить 12 атрибутів. Проведено аналіз та попередню підготовку даних. Здійснено паралелізацію алгоритму Випадковий ліс з використанням функціоналу бібліотеки sklearn. При цьому час, необхідний для тренування моделі зменшився в 4.4 рази, при використанні 8 паралельних потоків, в порівнянні з послідовним тренуванням. Також проведено розпаралелення даного алгоритму на основі CUDA. В результаті час, необхідний для тренування моделі зменшився в 83.4 рази, при використанні цієї технології на GPU. У роботі здійснено обрахунок коефіцієнтів прискорення та ефективності, а також наведено детальне порівняння з послідовним алгоритмом.

Ключові слова: *метод машинного навчання, алгоритм випадковий ліс, технологія CUDA, прискорення, коефіцієнт ефективності.*

Introduction

Artificial intelligence (AI) – based medical technologies are fast evolving into good solutions for clinical practice. Deep learning algorithms can deal with the increased amount of data provided by smartphones and other mobile monitoring sensors in various fields of medicine [1-3]. Currently, only very specific conditions in clinical practice benefit from the use of AI, such as heart disease, atrial fibrillation, seizures, epilepsy and hypoglycemia, or diagnosis of the disease based on histopathological examination or medical imaging. The introduction of such medicine provides greater autonomy and more personalized treatment.

Medical technology is widely used to access a range of tools that enable healthcare professionals to provide patients and society with a better quality of life through early diagnosis, reduction of complications, optimization of treatment, and reduction of hospital stays. Although before the mobile phones became widely use, prosthetics, stents, implants were largely known as traditional medical devices. The advent of smartphones, portable devices, sensors and communication systems revolutionized medicine with its ability to implement AI in very small accessories. AI has revolutionized medical technology and can solve complex problems in a variety of areas with vast amounts of data.

Machine learning methods for classification, regression and other tasks that operate by constructing many decision trees during learning and deriving a class (classification) or average prediction (regression) of individual trees called Random Forest [4]. Using this approach can significantly improve the accuracy of forecasting and result in better ability to predict new data cases.

However, the development of AI systems for medicine is not a trivial task. Main obstacles are collecting and organizing the data that will be used for training of classifier. This becomes a major problem, especially when the system requires large data sets over a long period of time, which in most cases are not available due to the lack of an efficient recording system. For example, to solve the problem of diagnosing heart disease, it is necessary that the accuracy for healthy patients is as high as possible, because incorrect classification in this category can lead to a healthy patient for no reason to switch to treatment.

Heart disease has been serious problem in both urban and rural areas in most countries. In 2020, Heart disease been the leading cause of death in the United States, England and Canada, accounting for 25.4 % of all deaths in the United States. A similar situation is observed in other countries [5]. In the case of cardiovascular disease, it is very important to make correct diagnosis at an early stage. It has been observed that in many cases, incorrect diagnosis leads to a violation of the patient's health.

Classification of medical data is considered to be a difficult task in the field of medical informatics. In this paper, an approach to the classification of medical data is developed and implemented. Random forest classifier is used to construct the predictor. This machine learning algorithm is expected to be effective for other diseases with similar datasets.

The purpose of this paper is the implementation of machine learning methods – Random Forest and Decision Tree, further parallelization of these algorithms to achieve greater accuracy of classification and reduce the time of training of these classifiers in the field of medical data processing, determining the presence of human cardiovascular disease. The purpose of the classification of cardiovascular disease is to classify the presence or absence of this disease on the basis of a set of data, which includes diagnostic measurements of humans.

The object of research is parallelization of machine learning methods – Random Forest and Decision Tree.

The subject of research is the process of accelerating the implementation of machine learning methods Random Forest and Decision Tree to increase the accuracy of the results obtained from these algorithms.

Related works

Practical implementation of machine learning tools will allow doctors to focus only on providing care to patients who have been suspected of exacerbating a disease. Especially during a pandemic, this method of detecting the disease will save the lives of many people by helping them before the disease worsens.

As the issues of this topic are relevant nowadays, an analysis of related works with similar research in this subject area was conducted. This provides an opportunity to get acquainted with existing approaches to predicting the presence or absence of disease, to study approaches to achieve the desired results using and accelerating machine learning methods – Random Forest and Decision Tree.

The relevance of using the Random Forest algorithm in Big Data processing is described in [6]. Focusing on classification problems, this paper proposes a selective review of available proposals that deal with scaling random forests to Big Data problems.

Bin Dai, Rung-Ching Chen, Shun-Zhi Zhu, Wei-Wei Zhang conducted research on the detection of breast cancer using the machine learning algorithm Random Forest [7]. The study was presented at the International Symposium on December 6-8, 2018. A group of scientists from different universities demonstrated the results of using this algorithm and the results achieved. Using the method of teaching several classifiers called ensemble training, it was possible to achieve high accuracy of disease prediction. This has great practical application as a method of providing an auxiliary medical diagnosis.

Leonard Lindroth in his work [8] conducted research on the parallelization of the machine learning algorithm Random Forest. The study identified the optimal parameters for obtaining the most effective acceleration. Studies have shown that openMP parallelization using the master thread for result combining and worker threads for mathematical computing have accelerated the learning process of the Random Forest method, which allows you to work more efficiently with large amounts of data and tune hyperparameters in less time, which allows algorithm to show greater accuracy of training in no time.

N. Azizah in his work [9] compared machine learning algorithms and concluded that the Random Forest method showed the best results in reducing learning time after its parallelization. The main problem shown in the study is the time needed to build models of trees that will form the Random Forest algorithm. By parallelizing the process of building trees, we load each computational unit more, which leads to increase of efficiency and acceleration.

In [10] the author conducted a study on the use of machine learning Decision Tree for analysis in the medical field. He used the Decision Tree algorithm to deal with complex medical problems that were difficult to overcome. The paper showed that the Decision Tree method is a tool that helps to make decisions about the treatment of patients in difficult situations. By analyzing information with a similar medical history in other patients, the trained decision tree algorithm can classify the cause of the disease, which helps to quickly find a way to treat it.

To sum up, predicting the course of the disease using data of similar cases of such disease is possible and relevant today. The parallelization of machine learning methods, especially Random Forest and Decision Tree, is important to optimize the learning of classifiers on large data sets, which will help to increase the accuracy of predictions faster. A common problem that often occurs when implementing random forest is long processing time because it uses a lot of data and build many tree models to form random trees because it uses single processor. This research proposes random forest method with parallel computing and implemented on CUDA.

As in the related works the parallelization of data of algorithms of machine learning at processing of medical data was not used, it confirms urgency of our work.

Methodology

Decision trees are a kind of non-parametric models that can be used for both classification and regression. This means that decision trees are flexible models that do not increase their number of parameters with the addition of functions, and they can display a categorical forecast (for example, whether a plant is a certain species or not) or a numerical forecast (for example, house price).

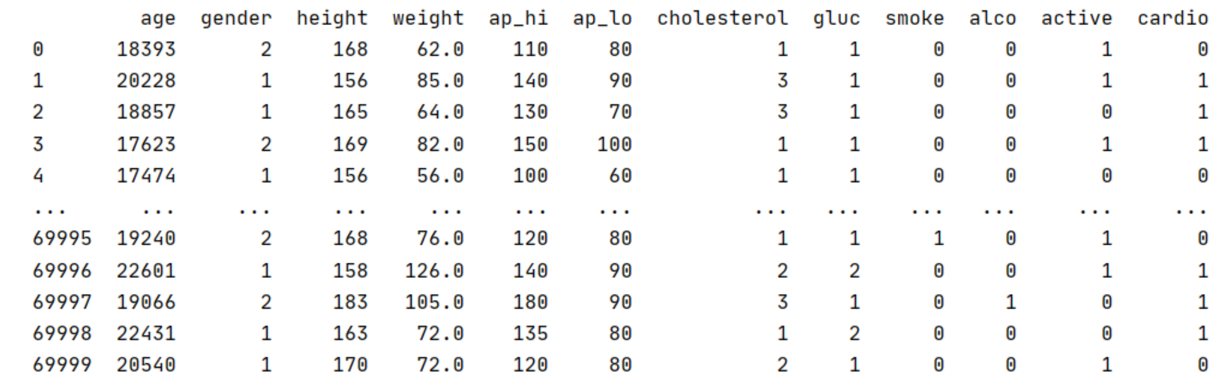
Decision trees are constructed using two types of elements: nodes and branches. At each node, one of the features of our data is evaluated to share observations in the learning process or to make the data point follow a certain path during forecasting [11].

Solution trees are built by recursively evaluating different characteristics and using the function that classifies the data at each node.

Low correlation between models is a key factor. Just as low-correlated investments (such as stocks and bonds) combine to form an aggregate that exceeds the sum of their parts, uncorrelated models can produce aggregate forecasts that are more accurate than any individual [12]. The reason for this wonderful effect is that the trees protect each other from their individual mistakes (if they are not all constantly wrong in one direction). While some trees may be wrong, many others will be true because as a group of trees, they will move in the right direction. To get the most from Random Forest Classifier:

1. Getting better accuracy with each learning step to be sure that classifier is not making random guesses.
2. Predictions made by each tree should have low correlation.

In this paper "Cardiovascular Disease dataset" was used. It was downloaded for Kaggle [13]. The purpose of the classification is to predict whether there is an exacerbation of cardiovascular disease in a patient with certain indicators obtained during the examination. The dataset consists of records of 70000 patients and contains 12 features.



```
         age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  cardio
0      18393       2     168    62.0    110     80            1     1      0     0       1       0
1      20228       1     156    85.0    140     90            3     1      0     0       1       1
2      18857       1     165    64.0    130     70            3     1      0     0       0       1
3      17623       2     169    82.0    150    100            1     1      0     0       1       1
4      17474       1     156    56.0    100     60            1     1      0     0       0       0
...      ...     ...     ...     ...    ...    ...          ...   ...    ...   ...     ...     ...
69995  19240       2     168    76.0    120     80            1     1      1     0       1       0
69996  22601       1     158   126.0    140     90            2     2      0     0       1       1
69997  19066       2     183   105.0    180     90            3     1      0     1       0       1
69998  22431       1     163    72.0    135     80            1     2      0     0       0       1
69999  20540       1     170    72.0    120     80            2     1      0     0       1       0
```

Fig. 1. Structure of dataset

There are 12 features in dataset displayed on Fig. 1:

Demographic Features:

- Gender (categorical variable);
- Age (continuous variable);
- Height (continuous variable);
- Weight (continuous variable).

Behavioral Features:

- Smoke – indicates if patient smokes (categorical variable);
- Alco – indicates if patient drinks alcohol (categorical variable);
- Active – indicates if patient leads an active lifestyle (categorical variable).

Medical Features:

- ap_hi – upper blood pressure (continuous variable);
- ap_lo – lower blood pressure (continuous variable);
- cholesterol – cholesterol content in the patient's blood (categorical variable);
- gluc – the content of glucose in the patient's blood (categorical variable).

Target variable:

- cardio – the patient has a cardiovascular risk.

The analysis and preliminary preparation of data was carried out, which provided the basis for the transition to the experimental part of the work.

Software implementation of these models was performed using a high-level Python programming language. The sklearn library was used to construct a classification Random Forest, as well as to parallelize it on different processor threads.

Computational complexity of the algorithm

Computational complexity of Random Forest classifier learning process is: $O(n * log(n) * d * k)$, n – number of rows in dataset; k – number of trees; d – size of training dataset.

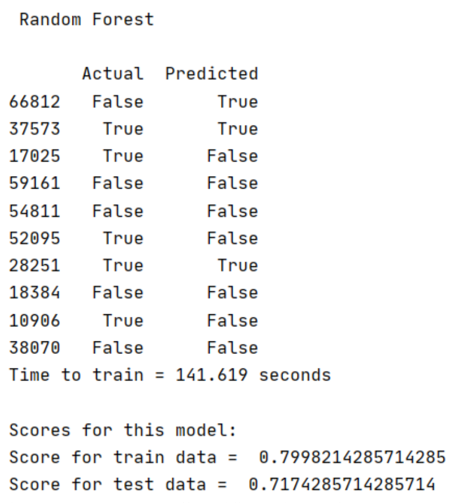
Acceleration: $S_p(n) = \frac{T_1(n)}{T_p(n)} = \frac{O(n*log(n)*d*k)}{O(\frac{n}{p}*log(\frac{n}{p})*d*k)}$, p – number of threads; $T_1(n)$ – execution time of sequential algorithm; $T_p(n)$ – execution time of parallel algorithm by p threads.

Effectiveness : $E_p(n) = \frac{S_p(n)}{p} = \frac{O(n*log(n)*d*k)}{p*O(\frac{n}{p}*log(\frac{n}{p})*d*k)}$.

Implementation

Dataset will be splitted into train dataset (80%) and test one (20%).

To implement the classification, we will use RandomForestClassifier from the sklearn library. We will use 1000 trees for training.



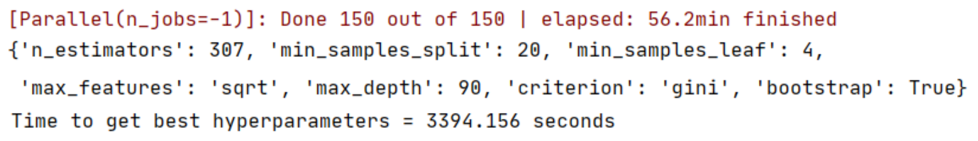
```
 Random Forest

       Actual  Predicted
66812   False       True
37573    True       True
17025    True      False
59161   False      False
54811   False      False
52095    True      False
28251    True       True
18384   False      False
10906    True      False
38070   False      False
Time to train = 141.619 seconds

Scores for this model:
Score for train data =  0.7998214285714285
Score for test data =  0.7174285714285714
```

Fig. 2. Random Forest Classifier with default parameters

The accuracy is low so we need to improve this model (see Fig. 2). Hyperparameter Tuning will be used in order to find best parameters. Function RandomizedSearchCV from sklearn library will be used to perform this.

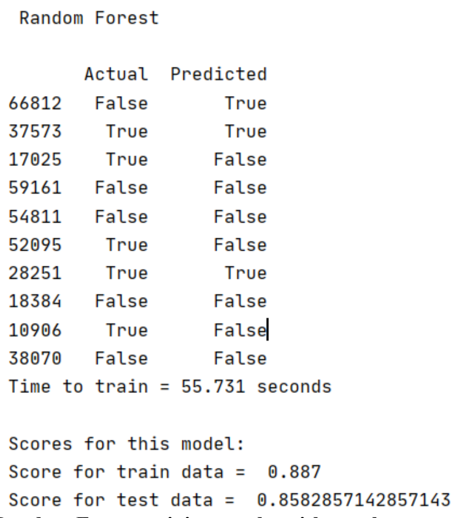


```
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed: 56.2min finished
{'n_estimators': 307, 'min_samples_split': 20, 'min_samples_leaf': 4,
 'max_features': 'sqrt', 'max_depth': 90, 'criterion': 'gini', 'bootstrap': True}
Time to get best hyperparameters = 3394.156 seconds
```

Fig. 3. Hyperparameters tuned by RandomizedSearchCV

Using new hyperparameters (Fig. 3) we can run train model one more train and compare results.

The accuracy of the model has significantly increased (for the test sample by 14.1%), and now it shows a fairly good result of 85.8%(see Fig. 4). However, the training still takes too much time, because of the significant amount of training data: 56 thousand records, the time required to train a given model is: 55.7 seconds.



```
 Random Forest

       Actual  Predicted
66812   False       True
37573    True       True
17025    True      False
59161   False      False
54811   False      False
52095    True      False
28251    True       True
18384   False      False
10906    True      False
38070   False      False
Time to train = 55.731 seconds

Scores for this model:
Score for train data =  0.887
Score for test data =  0.8582857142857143
```

Fig. 4. Random Forest training results with new hyperparameters

Random Forest Parallelization using sklearn library

As we know, the Python programming language runs on a single CPU, so it is impossible to directly parallelize any program. The Random Forest algorithm can be parallelized in Python by creating a forest with a large number of trees using each thread. This approach called the reinforcement classification algorithm [9].

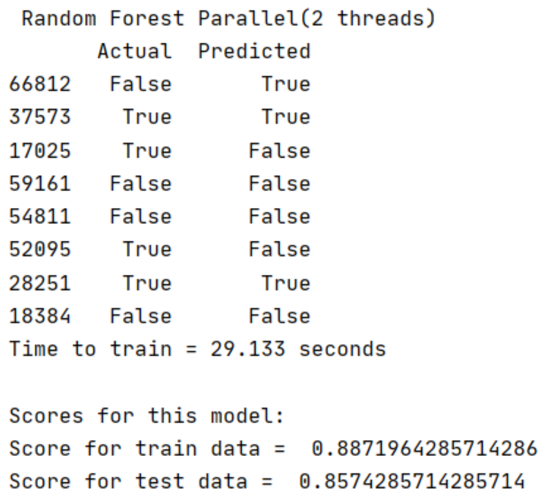
If we have a large number of trees in our forest, it makes its structure better and is considered the best forest.

The Random Forest algorithm that performs the classification task follows next scenario: the more trees, the better the accuracy and therefore the better the model. In a Random Forest, we do not use the same approach as in the decision tree algorithm, as we do not use entropy and information accumulation. Instead, in a Random Forest, we split a training dataset into smaller particles that consist of random elements from a training dataset.

Parallel computing basically involves using two or more cores (or threads) at the same time to solve a problem. The main goal here is to split the task into smaller subtasks and complete them simultaneously [9].

To perform this task, we will use the functionality of the sklearn library, which allows us to run the Python interpreter on several different threads at the same time, which we will use.

The Random Forest algorithm is based on splitting the main dataset into n-number of particles and constructing a classification tree for each of them, then combining the results. So, we will train several trees at the same time (depending on the number of available threads) on different threads.

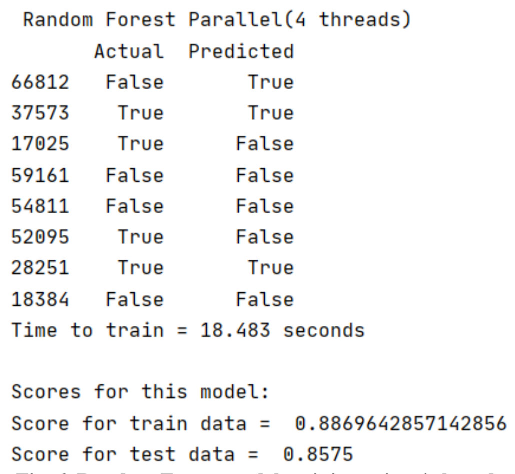


```
 Random Forest Parallel(2 threads)
       Actual  Predicted
66812   False       True
37573    True       True
17025    True      False
59161   False      False
54811   False      False
52095    True      False
28251    True       True
18384   False      False
Time to train = 29.133 seconds

Scores for this model:
Score for train data =  0.8871964285714286
Score for test data =  0.8574285714285714
```

Fig. 5. Random Forest model training using 2 threads

In the Fig. 5 improved results are displayed. We can see a significant improvement, in terms of speeding up the training process of our model – 1.9 times in comparison to sequential training, using 2 parallel threads.

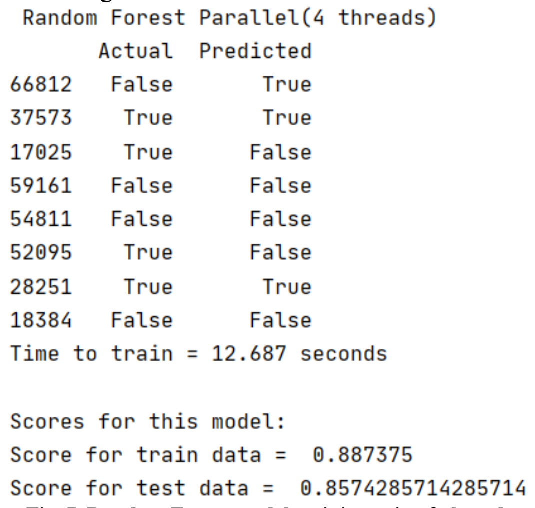


```
 Random Forest Parallel(4 threads)
       Actual  Predicted
66812   False       True
37573    True       True
17025    True      False
59161   False      False
54811   False      False
52095    True      False
28251    True       True
18384   False      False
Time to train = 18.483 seconds

Scores for this model:
Score for train data =  0.8869642857142856
Score for test data =  0.8575
```

Fig. 6. Random Forest model training using 4 threads

In Fig. 6 we can see that the time required to train the model is reduced by 3 times when using 4 parallel threads in comparison to sequential training.



```
Random Forest Parallel(4 threads)
      Actual  Predicted
66812  False       True
37573   True       True
17025   True      False
59161  False      False
54811  False      False
52095   True      False
28251   True       True
18384  False      False
Time to train = 12.687 seconds

Scores for this model:
Score for train data =  0.887375
Score for test data =  0.8574285714285714
```

Fig. 7. Random Forest model training using 8 threads

In Fig. 7 we can see that the time required to train the model decreased by 4.4 times when using 8 parallel threads in comparison to sequential training.

Random Forest algorithm parallelization using CUDA

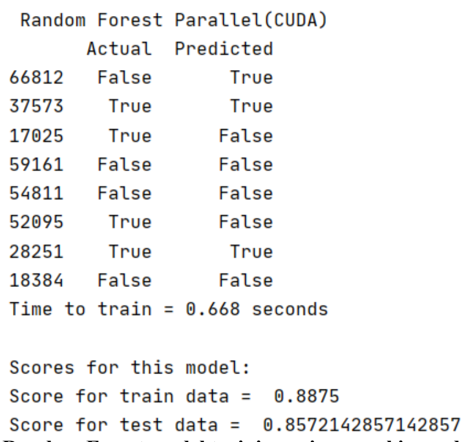
The RF model, which is implemented using the CUDA environment [14], contains 2 high-performance separate algorithms that select values for the training of each feature and the association of nodes. Quantiles and min / max histograms are used for this purpose.

In both cases, the values to be divided are chosen for each of the features. These algorithms use the GPU acceleration approach, which significantly reduces the number of operations required to perform data sharing calculations.

Min / max histograms are built for each feature of the tree node. The range of values of each feature is divided equally. The end of each interval is considered as a potential value for separation. In this approach, the division values for each attribute are calculated in each node, adapted to the data intervals in each node of the tree.

The quantile separation algorithm calculates in advance the potential values for the separation for each feature at the root of the tree. Each column with the feature is sorted in ascending order and divided equally by the nodes of the dataset. The end of each interval is considered as a potential value for a split.

As we can see in Fig. 8, that the time required to train the model using GPU has decreased by 83.4 times in comparison to sequential training. However, it should be noted that this is only the time of training of Random Forest model, not taking into account the time spent on splitting the dataset into smaller particles and pre-processing them to upload to the GPU. Although, if you take into account the time required to upload data to GPU, the time required to train the model is reduced by approximately 23 times in comparison to sequential training.



```
 Random Forest Parallel(CUDA)
       Actual  Predicted
66812   False       True
37573    True       True
17025    True      False
59161   False      False
54811   False      False
52095    True      False
28251    True       True
18384   False      False
Time to train = 0.668 seconds

Scores for this model:
Score for train data =  0.8875
Score for test data =  0.8572142857142857
```

Fig. 8. Random Forest model training using graphic card (GPU)

Experiments

In the paper we compared different approaches to build a model for classifying the presence of exacerbations of cardiovascular disease in patients with different indicators of diagnostic examination. In Table 1 we displayed comparison of the execution times of sequential and parallel implementations of the program on the CPU (using the sklearn library) and the GPU (CUDA), as well as analyzed them.

Table 1

Execution time of sequential and parallel Random Forest algorithm, s

Sequential Execution	**Parallel Execution**			
	2 threads	**4 threads**	**8 threads**	**GPU**
55.731	29.133	18.483	12.687	0.668

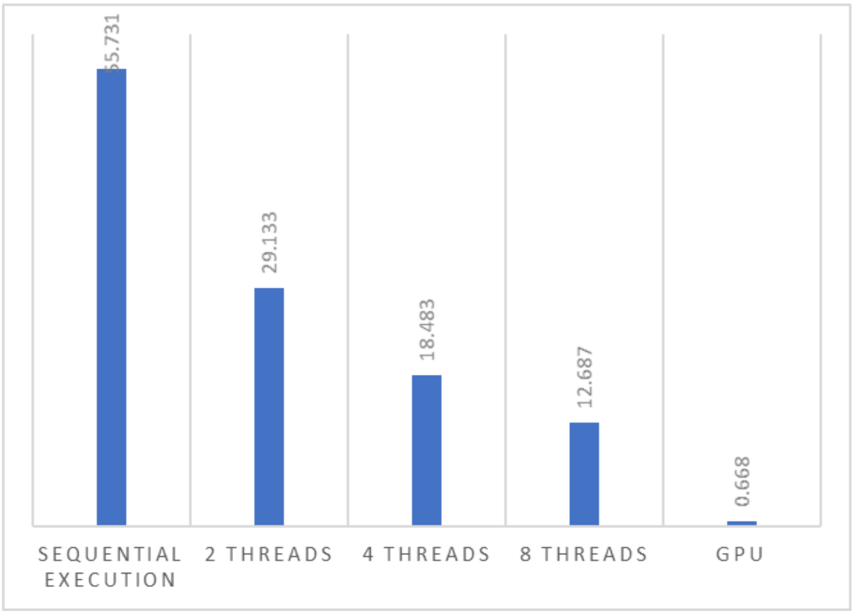


Fig. 9 Comparison of model training time using CPU and GPU

In Fig. 9 we can see that parallel training of Random Forest model is far more effective than sequential one. We can also see that the training time of the model using maximum possible number of parallel threads is 4.4 times less than the training time for sequential processing, but in comparison training using GPU is even more effective. This indicates the high computing power of the graphic cards and the advantages of conducting parallel computing using GPU.

Now let's calculate the experimental rates of acceleration and efficiency of parallel algorithms for different number of threads, if parallel calculations are performed on the processor, as well as the rates of acceleration of parallel algorithms for the GPU.

Table 2

Acceleration rates for Random Forest parallel algorithm

Number of threads			GPU
2	4	8	
1.913	3.015	4.393	83.43

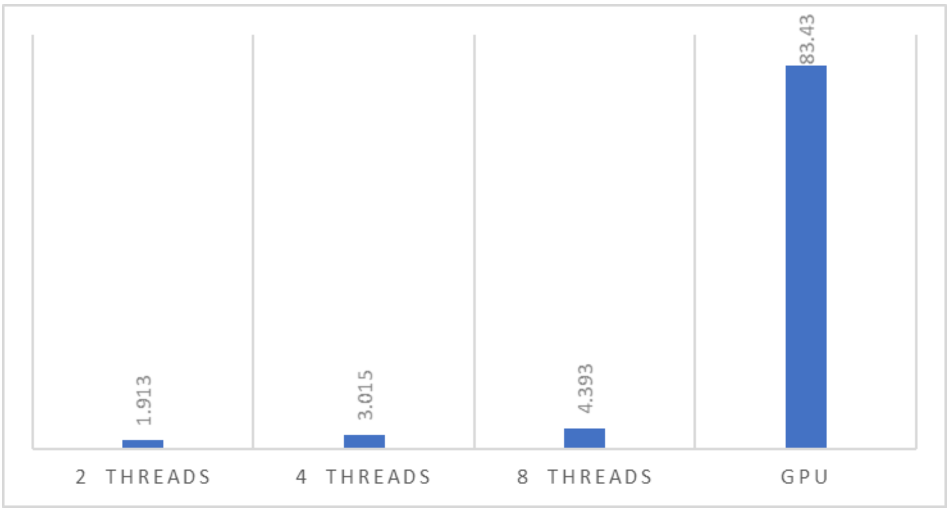


Fig. 10. Diagram of acceleration rates for different number of threads (on the CPU), as well as when running on the GPU

As the number of threads increases, the acceleration rate increase too. However, in comparison to the parallel computing using GPU, the acceleration we get when running algorithm using the CPU is much smaller, which again indicates the high power of computing with a graphic card.

Table 3

Effectiveness rates for Random Forest parallel algorithm

Number of threads		
2	4	8
0.957	1.508	0.549

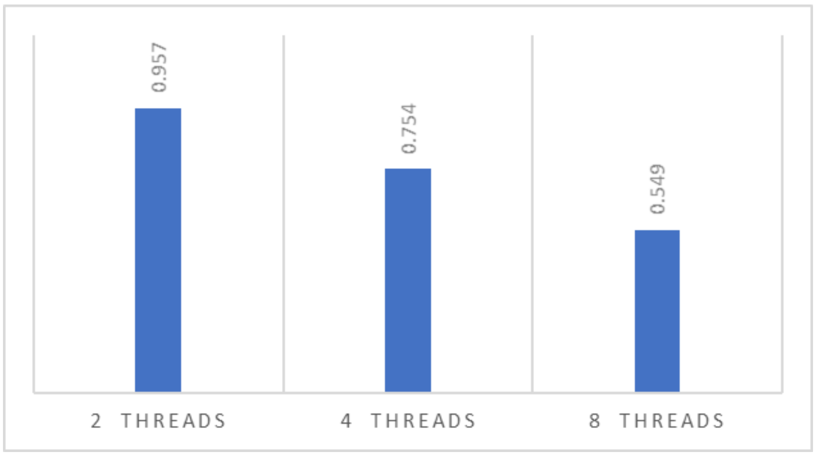


Fig. 11. Chart of effectiveness rates for different number of threads (per CPU)

In comparison to acceleration, efficiency does not increase with increasing number of threads, but rather decreases. The reason is increasing the number of parallel threads that we use for computing increases the load on the processor, which causes less computational efficiency.

Conclusions

Modern information technologies are increasingly used in the field of health care. Due to this, medicine today acquires completely new features. In many medical studies, it is simply impossible to do without a computer and special software for it. This process is accompanied by significant changes in medical theory and practice related to adjustments both at the stage of training of medical workers and for medical practice.

The methods of data analysis for the prediction of cardiovascular diseases are mentioned in this paper. The Cardiovascular Disease dataset was used for the study, which contains data on 70,000 patients with 12 features. Medical data processing was performed using the RF machine learning algorithm, namely for classification according to the set parameters of human cardiovascular disease. Because it was necessary to work with a large amount of data, the study was close to real.

The parallelization of this machine learning algorithm was also carried out, the obtained results showed that high rates of acceleration and efficiency were achieved. Performing parallelization using CPU and GPU, it was possible to compare the results of the time of calculation and processing of medical data. However, when comparing our result with results [8], it is clear that the time required to train the model was reduced by 83.4 times when using technology CUDA on the GPU.

Due to the advantage of the GPU in the number of processor cores, it was possible to train the classifier of the RF algorithm 20 times faster than when training using CPU on 8 threads, which was the maximum available number in our study. The RF algorithm is the optimal classifier for optimization by parallelization and allowed to achieve good results shown in this paper.

References

1. Mochurad L., Hladun Ya. Modeling of Psychomotor Reactions of a Person Based on Modification of the Tapping Test. *International Journal of Computing*, 20(2), 190-200, 2021. doi: 10.47839/ijc.20.2.2166.
2. Wang H, Pujos-Guillot E, Comte B, de Miranda JL, Spiwok V, Chorbev I, Castiglione F, Tieri P, Watterson S, McAllister R, de Melo Malaquias T, Zanin M, Rai TS, Zheng H. Deep learning in systems medicine. *Brief Bioinform.* 2021 Mar 22;22(2):1543-1559. doi: 10.1093/bib/bbaa237. PMID: 33197934; PMCID: PMC8382976.
3. Decuyper, M., Maebe, J., Van Holen, R. *et al.* Artificial intelligence with deep learning in nuclear medicine and radiology. *EJNMMI Phys* 8, 81 (2021). https://doi.org/10.1186/s40658-021-00426-y
4. Ender Konukoglu, Ben Glocker. Chapter 19 - Random Forests in medical image computing, Handbook of Medical Image Computing and Computer Assisted Intervention, Academic Press, 2020, Pages 457-480, ISBN 9780128161760, https://doi.org/10.1016/B978-0-12-816176-0.00024-7.
5. Ahmad F.B., Anderson R.N. The Leading Causes of Death in the US for 2020. *JAMA.* 2021;325(18):1829–1830. doi:10.1001/jama.2021.5469.
6. Robin Genuer, Jean-Michel Poggi, Christine Tuleau-Malot, Nathalie Villa-Vialaneix. Random Forests for Big Data, *Big Data Research*, Volume 9, 2017, pp. 28-46, doi: 10.1016/j.bdr.2017.07.003.
7. B. Dai, R. -C. Chen, S. -Z. Zhu and W. -W. Zhang. Using Random Forest Algorithm for Breast Cancer Diagnosis, *2018 International Symposium on Computer, Consumer and Control (IS3C)*, 2018, pp. 449-452, doi: 10.1109/IS3C.2018.00119.
8. Leonard Lindroth. Parallelization of Random Forest. *Master of Science in Engineering: Computer Security*, 2021, 31 p.
9. N. Azizah, L. S. Riza, Y. Wihardi. Implementation of Random Forest algorithm with parallel computing, *Journal of Physics: Conference Series*, Volume 1280, 2019, 6 p., doi:10.1088/1742-6596/1280/2/022028.
10. Bae, Jong-Myon. The clinical decision analysis using decision tree. *Epidemiology and health* vol. 36 e2014025. 30 Oct. 2014, doi:10.4178/epih/e2014025.
11. Vishal Mehta. Accelerating Random Forests Up to 45x Using cuML. 2021. Âåá-ñàéò URL: https://developer.nvidia.com/blog/accelerating-random-forests-up-to-45x-using-cuml/
12. Cassidy A.P. and Deviney F.A. Calculating feature importance in data streams with concept drift using Online Random Forest, *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 23-28, doi: 10.1109/BigData.2014.7004352
13. Data sampling. URL: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset
14. Mochurad L.I. Optimization of numerical solution of model problems on the basis of parallel calculations. Chapter 1: monograph. Lviv: PE "BONA Publishing House". (2021) 208 p.

Lesia Mochurad **Леся Мочурад**	PhD, Associate Professor of Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine, e-mail: lesia.i.mochurad@lpnu.ua. https://orcid.org/0000-0002-4957-1512 Scopus Author ID: 57210286606, ResearcherID: AAZ-9150-2020 https://scholar.google.com/citations?user=jq_lWBkAAAAJ&hl=ru&oi=sra	кандидат технічних наук, доцент, доцент кафедри систем штучного інтелекту національного університету "Львівська політехніка", м. Львів, Україна
Andrii Ilkiv **Андрій Ільків**	student of Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine, e-mail: andrii.ilkiv.knm.2018@lpnu.ua. https://orcid.org/0000-0001-6438-0784	студент кафедри систем штучного інтелекту національного університету "Львівська політехніка", м. Львів, Україна.