The Random Forest Algorithm as an Element of Statistical Learning for Disease Prediction

Nataliya Boyko[1], Roman Omeliukh[1] and Nataliia Duliaba[1]

[1]*Lviv Polytechnic National University, Profesorska Street 1, Lviv, 79013, Ukraine*

Abstract

Every day new calls appear before physicians, such as appearance of new viruses and illnesses. Many researches are conducted for the invention of new and improvement of present methods of diagnostics, but one of basic questions is remained by early diagnostics of heart troubles. In fact with every year all more young patients call to the doctors, therefore loading on the system of health protection in earnest increases and not all get medicare in time. Many cases of more heavy flow of illnesses can be prevented if diagnostics was conducted quicker and more precisely. It was therefore decided to devote this research of method of prognostication such as Ramdom Forest. As by means of the software based on this method of prognostication, we will be able to do predictions about cardiac ailments more precisely and quicker, that can considerably help to shorten a death rate. Thus, this research will allow the creation of software that can predict heart disease based on information entered by the patient and allow self-diagnosis.

Keywords

Mean squared error, the result of the model prediction, residual sum of squares, decision tree, True positive, True negative, False positive, False-negative.

1. Introduction

Healthcare has always been a major area of machine learning research. Our learning algorithms can help doctors make diagnoses, analyze X-rays. The cost of error is very high, therefore research in this area is always relevant and necessary [6, 17].

This paper will conduct a theoretical analysis of the method of random forest and all its components as well, namely the decision tree, bootstrapping and bagging. Here will also be a theoretical comparison with another ensemble method of the machine training - boosting trees.

Random forest is an ensemble method, it unites many weak models to create the one powerful model. The random forest has shown itself well at work with tabular data, it is resistant to rewriting [1, 9].

CITRisk'2022: 3rd International Workshop on Computational & Information Technologies for Risk-Informed Systems, January 12, 2023, Neubiberg, Germany
EMAIL: nataliya.i.boyko@lpnu.ua (N.Boyko); roman.omeliukh.knm.2019@lpnu.ua (R.Omeliukh); nataliia.i.duliaba@lpnu.ua (N.Duliaba)
ORCID: 0000-0002-6962-9363 (N.Boyko); 0000-0001-6859-2792 (R.Omeliukh); 0000-0002-7420-962X (N.Duliaba)

In the practical part of this work the analysis of a dataset from patients in the hospital, and the goal will be to predict based on the provided data or the patient has a cardiovascular disease. Such a model can be very useful in hospitals as an additional diagnosis check before appointment of treatment [3, 10].

2. Review of literature sources

The FP-Growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree. This tree structure will maintain the relationship between a set of elements. The database is fragmented with a recurring component. This fragmented part is called "pattern fragment". Sets of elements of these fragmentary patterns are analyzed. Thus, this method reduces the search for frequent item sets comparatively.

In the book, Trevor Hastie, Robert Tibshirani, and Jerome Friedman "The Elements of Statistical Learning Data Mining, Inference, and Prediction" [1] said that "random forests are a significant modification of the bagging method, which creates a large number of trees and then averages them. In many tasks, random forests are very similar to the boosting method and are easier to train and configure. As a result, random forests are popular and implemented in various tasks" [12, 18]

In Leo Braiman's book "Random forests" [2] said – "random forests are an effective tool in prediction. Thanks to the law of large numbers, they do not overflow. Entering the right type of randomness makes them accurate classifiers and regressors. In addition, the structure in terms of the strength of individual predictors and their correlation gives an idea of the ability of the random forest to predict" [7, 14]

In the work of Leo Breiman and Adele Cutler, "Random forests" [3], the following features of the random forest algorithm are described: "the best in terms of accuracy among modern algorithms; works effectively on large databases; can handle thousands of input variables without deleting them; provides estimates of which variables are important for classifications; generates an internal unbiased estimate of the generalization error as the forest grows; has an efficient method for estimating missing data and maintains accuracy when large portions of data are missing; has methods for equalizing errors in unbalanced class datasets; prototypes are calculated that provide information about the relationship between variables and classification; calculates closeness between pairs of cases, which can be used in clustering, finding outliers, or (by scaling) provide interesting insights into the data; offers an experimental method for detecting variable interactions" [4, 13]

In the book S. Revathi, A. Malathi, "Optimization of KDD Cup 99 Dataset for Intrusion Detection Using Hybrid Swarm Intelligence with Random Forest Classifier", the advantages of random forest are called the following: "overcoming the problem of overfitting; in training data, the algorithm is less sensitive to outliers; parameters can be easily set and therefore eliminated the need to prune trees of varying importance and accuracy is automatically generated" [5, 16].

So, in various sources, the random forest method is called entirely accurate, able to process a variety of extensive data with a lost part, such that it stands out among others. Due to these and other advantages, the field of application of random forests is wide [7].

3. Materials and Methods

3.1. Decision trees

Decision trees can be applied to regression and classification problems. The decision tree model is a combination of logic rules "if then else" in the Tree data structure. The inner tops of the tree are marked with thermonyms, and the ribs are marked with test scales. The weight of the test example is compared with these test weights while the prediction. Leaves in trees denote the values of the classes to be assumed [8, 12].

The advantage of the decision tree is the ability to visualize the results and interpretation of the obtained data while obtaining the prediction.

There are two types of decision trees:

- classification trees, when the predicted result is a class the data belongs to.
- regression trees, when the predicted result can be considered as real number (for example, air temperature, house price).

Algorithms for designing decision trees consist of stages "construction" or "tree building" and "tree pruning". The questions of a choice of criterion of splitting and a stop of training (if provided by the algorithm) are solved during the tree creation. The question of cutting off some of its branches is solved during the stage of tree reduction [11].

The process of creating a tree is from top to bottom, being descending. During the process, the algorithm must find a splitting criterion, sometimes also called a splitting criterion, to split the set into subsets that would be associated with this test node. Each test node must be marked with a specific attribute. There is a rule for selecting an attribute: it must split the input data set so that the objects of the subsets obtained as a result of this splitting, were members of the same class or were as close as possible to such a splitting [2, 14].

There are different criteria for splitting. The most famous are the measure of entropy and the Gini index. If a given set T, including examples of n classes, the Gini index is determined (Formula 1):

$$gini(T) = 1 - \sum_{j=1}^{n} \quad p_j^2, \ p_j = \frac{N_j}{N} \text{ - part of class j at node T.} \tag{1}$$

where *T* is the current node, pj is the probability of class *j* in the node *T*, *n* is the number of classes, *N* is the number of objects in the node [16].

The measure of entropy in the construction of decision trees is a measure of the diversity of classes in node. As a result of the spliting, nodes with a smaller variety of states of the original variable. Therefore, the entropy falls, and the quantity internal information in the node grows. Formally, the entropy of a certain node *T* of the decision tree is determined by (Formula 2):

$$info(T) = -\sum_{j=1}^{n} \quad p_j \cdot log(p_j). \tag{2}$$

The entropy of the whole splitting is the sum of the entropies of all the nodes multiplied by the fraction records of each node in the total number of records: (Formula 3):

$$Info(S) = \sum_{j=1}^{n} \quad . \frac{N_j}{N} \cdot Info(T_j). \tag{3}$$

To select a split attribute, use the criterion is called information gain or entropy decrease (Formula 4).

$$Gain(S) = info(T) - info_s(T). \tag{4}$$

As the best attribute for use in the splitting S is selected the one that provides the largest increase in Gain (S) information [2, 15].

In the process of building the tree the special procedures that allow you to create optimal trees are used. So, the size of the tree does not become excessively large.

Usually, in order to avoid overfitting, a so-called trimming is used (pruning), that is, they allow the tree to grow further, and then cut off the extra branches. Pruning a tree at some node is that subtree rooted at that node is removed from the tree, and a label is placed in the node with the same result as in the node this node more. Pruning branches or replacing some branches with a subtree should be carried out where this procedure does not increase the error. Process passes from bottom to top, that is, ascending.

3.2. Ensemble methods

Decision trees are a simple model, so it may not be enough to solve complex problems on large datasets. Therefore, in practice, ensemble methods based on decision trees, such as random forests and boosting systems, are more commonly used.

For example, in a classification problem, an algorithm is considered weak if its error in the training sample is less than 50% but greater than 0%. In the case of binary classification (when there are only two classes), we can say that the classifier is weak if it is not much better than a simple "guess". If the classifier error in the training sample can be reduced to a value arbitrarily close to 0% in polynomial time, then the classifier is called strong [3, 17].

The general idea of ensemble methods is that with a few weak classifiers you can create a rule that can increase the accuracy of the prediction, and thus create one strong meta-classifier.

For example, if there are 5 binary classifiers, each of which has a prediction accuracy of 60%, then combining them into an ensemble, where the decisive prediction of the object class will be decided by a majority vote of these five classifiers, you can increase the accuracy of the meta-classifier to 68%, as shown (Formula 5):

$$r = \sum_j^n \quad C_n^j \cdot p^j \cdot q^{(n-j)} \text{ , where } p = 0.6, q = 0.4, n = 5 \tag{5}$$

$$r = \sum_{j=3}^{5} \quad C_5^j \cdot 0.6^j \cdot 0.4^{(5-j)} = 0.68$$

Simple majority voting (for classifiers) and averaging the results of individual trees (for regression models) is not the only way to combine the predictions of the basic models in ensemble methods. The weighted voting method is also often used, where the results of the models are given more weight in the voting with greater accuracy [3, 18].

3.3. Bootstrap

Statistical bootstrap is a method of determining probability distribution statistics based on multiple generations of samples by the Monte Carlo method based on an existing sample [4, 5]].

The bootstrap method itself is as follows. Suppose there is a sample Z of size N - a set of independent equally distributed random variables. Evenly take from the sample N objects with return. This means that we will select an arbitrary sample object each time (it is assumed that each object has the same probability of being selected from the sample), and each time we select from all source objects.

Because objects are taken from a return sample, objects can be repeated in the resulting sample. Repeating the procedure of creating new subsamples, M subsamples will be generated: Z_1 , Z_2 , ..., Z_m. These are multiplied subsamples are not independent, but as the size of the original sample increases, the effect of the dependence may weaken, and therefore the values of the statistics of these samples can be treated as independent random variables [1, 7].

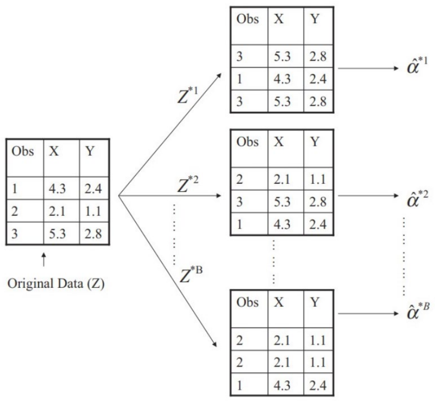


Figure 1: Graphic illustration of a bootstrap on a small dataset

Fig. 1 shows an example of using a bootstrap on a simple dataset Z, with three lines (n = 3). In order to create bootstrap samples, n observations are randomly selected from the original sample. These samples have replicates (subsample Z^1contains the third observation twice). Thus, B subsamples were created, each of which has statistics α^i [1, 6].

3.4. Bagging

Decision trees usually have low bias but are very large dispersion (variance), that is, they suffer from sensitivity to small changes in training data. That is, if the training sample is divided into two parts and build decision trees on each of them, then the results of these trees on one test sample can be very different.

Bootstrap proved to be very useful in forming an ensemble based on decision trees, as it helps to combat the problem of high variability of the model. When bagging, trees are not pruned, so each of them individually has a high variability, but averaging their results, the variability decreases.

By averaging several observations, it is possible to reduce the estimation of the variance of the data, similarly having a large number of subsamples of the general population, training the model on each of these subsamples and averaging predictions, you can reduce the variability of the meta-model [1, 9].

In practice, it is often not possible to create several full-fledged independent training samples, so bootstrap is used. Having generated M different training bootstrap samples, training the model on each of them, we get M models (Formula 6):

$$\widehat{f^1}(x), \widehat{f^2}(x), \dots, \widehat{f^M}(x). \tag{6}$$

Begging (bootstrap aggregation) - it is a procedure of taking average collective forecast from models based on bootstrap samples (Fortmula 7):

$$\widehat{f_{bag}}(x) = \frac{1}{M}\sum_{m=1}^{B} \quad \widehat{f^{m}}(x). \tag{7}$$

Bagging can be applied to a wide range of machine learning models, but it has proven to be particularly useful for decision trees. This is because bagging avoids the high correlation between decision trees that occurs when training them using the same data [5, 12].

To apply the bagging procedure to regression decision trees, M trees are trained using M bootstrap samples, and the prediction results of these trees are aggregated (usually averaged). There are several possible approaches to solving the problem of classification, but the simplest is majority voting. For this test observation, we can record the class predicted by each of the trees and take the majority of votes: the general forecast is the class that is most common among the forecasts [1, 18].

A feature of bagging is that it allows you to get a validation score models without using cross-validation or a validation set. Because the bagging involves training models on bootstrap samples, then a particular model uses for training only a part of the unique values in the original dotoset. Thus, those data that did not fall into the bootstrap sample can be used to validate this model. This approach is called error estimation [6].

3.5. Random forest

Random forest is a further improvement of decision tree bagging that is to eliminate the correlation between trees. As with bagging, we train M decision trees on bootstrap samples. But when building trees, each partition, a random subsample of all predictors is selected and the partition it is allowed to make only one of the changes from this subsample. In other words, when constructing a random forest, on each partition, the tree is not allowed touse most of the available predictors.

Most often, in practice, the number of predictors that will be considered at each partition m is defined as the rounded up square root of the number of all predictors p (Formula 8):

$$m \approx \sqrt{p}. \tag{8}$$

Choosing a small value of m when constructing a random forest will usually be useful in the presence of a large number of correlated predictors. Of course, if the random forest is built using $m=p$, then the whole procedure comes down to simple running [1, 9].

This decorrelation approach is based on the assumption that if a particular dataset has one very strong predictor, then most trees will use it in the first partition. In this case, most trees will look similar to each other, and therefore will be very highly correlated. Averaging the predictions of many correlated trees will not reduce the overall variability of the model, which means that bagging will not give a significant decrease in virulence compared to a simple deep decision tree.

In the random forest method, due to the use of a predictor subsample, on average $(p - m)/p$ partitions in trees will not be able to use a strong predictor, and therefore weaker predictors will be more likely to be used on the first partitions.

The algorithm for building a random forest, which consists of N trees, is as follows:

1. For each tree to generate a bootstrap sample X_n;
2. Construct each tree Tn on the sample X_n:
3. At each partition of the tree, according to a given criterion (measure of entropy, Gini index) is selected the best predictor, from the available random subsets of predictors.

Continue to build a tree until its leaves are no more than specified the number of minimum objects or until the specified value is reached maximum depth of the tree.

The result will be a model (Formula 9):

$$\widehat{f_{rand}}(x) = \frac{1}{M}\sum_{m=1}^{B} \quad \widehat{f^{m}}(x)\ . \tag{9}$$

That is, the result in the case of classification is chosen by voting, and in case of regression by averaging.

3.6. Assessing the importance of predictors

Random forest can greatly improve the accuracy of predictions compared to the decision tree, but the disadvantage of random forest is that the resulting model is more difficult to interpret [10, 13].

In the case of the decision tree, it is very easy to interpret the results, because the tree can be visualized and see which predictors were used for breaking down faster, and thus understanding the importance of each predictor in a given problem.

The average decrease in accuracy given by the predictor is determined during the out-of-bag error calculation phase. The more the accuracy of predictions decreases due to the exclusion (or permutation) of one variable, the more important this predictor is, and therefore, predictors with a large mean decrease in accuracy are more important for this task. The mean reduction in Gini uncertainty, or RSS (Residual Sum of Squares) error in regression problems, is a measure of how each predictor contributes to the homogeneity of nodes and leaves in the final random forest model. Each time a single variable is used to partition a node, the Gini uncertainty, or RSS, for the child nodes is calculated and compared with the output factor node [4, 12].

Changes in the value of the separation criterion are summed for each variable and normalized at the end of the calculation. The larger the number obtained, the a more important predictor for this task.

3.7. Variation and displacement of the random forest

With the help of bagging and decorrelation, the random forest helps to reduce the variation of the model, but randomization when creating a boost-sample and when choosing predictors for partitioning in the decision tree increases the bias of the model.

Variation of a random forest is given by the Formulas 10-12:

$$Var\ f\ (x) = \rho(x) \cdot \sigma^2(x). \tag{10}$$

where $\rho(x)$ - correlation between any two trees (Formula 11).

$$\rho(x) = corr[Pred(x, T_1(Z_1)), Pred(x, T_2(Z_2))]. \tag{11}$$

where $T_1(Z_1)$,$T_2(Z_2)$ - random couple of trees trained on the bootstrap samples Z_1, Z_2.

$$\sigma^2(x) = Var(x, T(X)) \ - \ selective\ variance\ of\ any\ random\ trees. \tag{12}$$

If we consider the variance of one tree $\sigma^2(x)$, it practically does not change from the sample of predictors for separation for each tree separately, but for the ensemble the sample of predictors plays a big role, because ρ(x) changes from this sample, and therefore the variance for the tree is much higher than for the ensemble [1, 13].

The displacement of a random forest is usually greater (in absolute terms) than the displacement of an unpruned tree, because bootstrap and predictor subsamples are randomized. Therefore, the improvement in the accuracy of predictions, obtained by random forest, is solely the result of reducing the variance.

3.8. Comparative characteristics

This section will provide a theoretical comparative characterization of the random forest with another ensemble method - boosting trees.

Boosting (boosting) is a procedure of consistent construction compositions of machine learning algorithms, where each following algorithm seeks to compensate for the shortcomings of the composition of all previous algorithms. Boosting is a greedy algorithm for constructing algorithms [3].

The differences between boosting decision trees and random forest are given in Table 1.

Table 1
Comparative characteristics with boosting trees

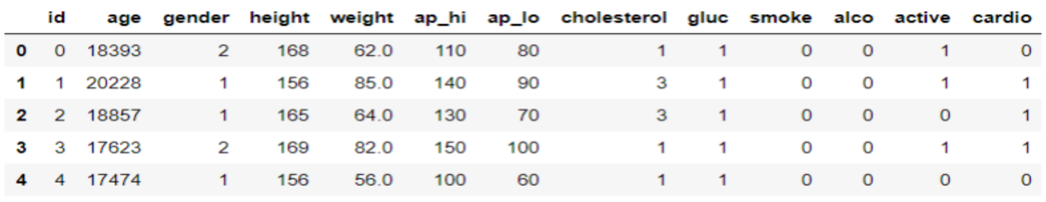
Random forest	Boosting trees
An ensemble of strong models (deep decision trees), each of which has high variation and low displacement.	An ensemble of weak models (shallow decision trees), each of which has low variation and high displacement.
Reduces prediction error by reducing variance of the models.	Reduces prediction error by reducing bias of the models.
Uses bootstrap sampling for training individual trees, randomly selects possible predictors to break the tree, which minimizes correlation between different trees.	Each tree increases its weight classes that were harder to predict (where the bigger mistake) and on these weighted data trains the next tree, thus compensating for those errors made by the previous tree
Training can take place in parallel, as separate trees independent of each other.	Training should take place consistently since, a separate tree tries to compensate for mistakes previous.
The result is selected majority voting / averaging	The result is chosen weighted by voting / weighted amount.

4. Experiments

4.1. Analysis and preparation of the dataset Decision trees

For the practical part of this work, it was decided to investigate the detset, which contains data on patients and whether they were found to have cardiovascular disease. That is, the problem of classification will be solved (Fig. 2).

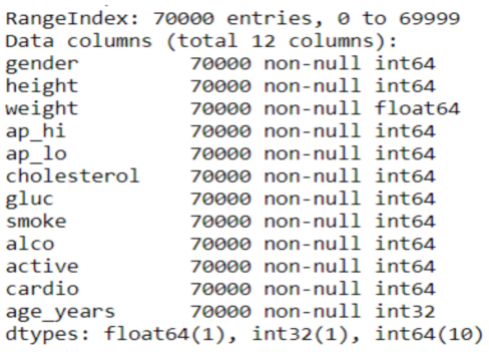
This dataset consists of 70,000 patient records, which include: age, height, weight, sex, blood pressure, cholesterol, and blood glucose, whether the patient smokes and drinks alcohol, and whether he is physically active.



	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

Figure 2: View of the first records in the dataset

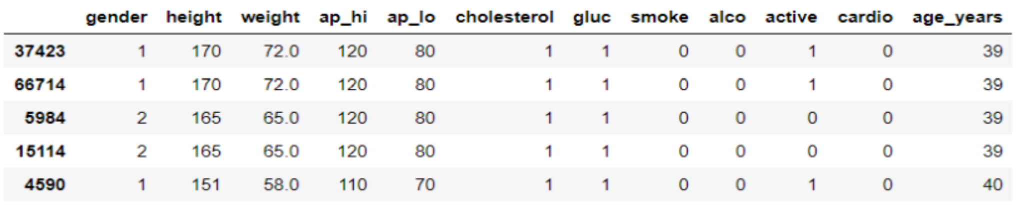
Let's look at the general information on columns in a dataset (Fig. 3.). You can see that the dataset consists of numeric data, but some of them correspond to categorical variables. You can also see that the dataset does not contain missing values.



```
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
gender         70000 non-null int64
height         70000 non-null int64
weight         70000 non-null float64
ap_hi          70000 non-null int64
ap_lo          70000 non-null int64
cholesterol    70000 non-null int64
gluc           70000 non-null int64
smoke          70000 non-null int64
alco           70000 non-null int64
active         70000 non-null int64
cardio         70000 non-null int64
age_years      70000 non-null int32
dtypes: float64(1), int32(1), int64(10)
```

Figure 2: Information about columns in the dataset

After checking the dataset for the absence of duplicate lines. In Fig. 3 examples of duplicate rows in the dataset are displayed. Because complete duplicates do not contain useful information, these rows can be deleted.



	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_years
37423	1	170	72.0	120	80	1	1	0	0	1	0	39
66714	1	170	72.0	120	80	1	1	0	0	1	0	39
5984	2	165	65.0	120	80	1	1	0	0	0	0	39
15114	2	165	65.0	120	80	1	1	0	0	0	0	39
4590	1	151	58.0	110	70	1	1	0	0	1	0	40

Figure 3: Example of duplicate data in a dataset

We will check the relationships between the target variable and other variables. In Fig. 4 shows the number of patients in whom (yellow) and did not detect (green) vascular diseases relative to their age years). You can see that the attitude of patients to healthy patients increases with age.

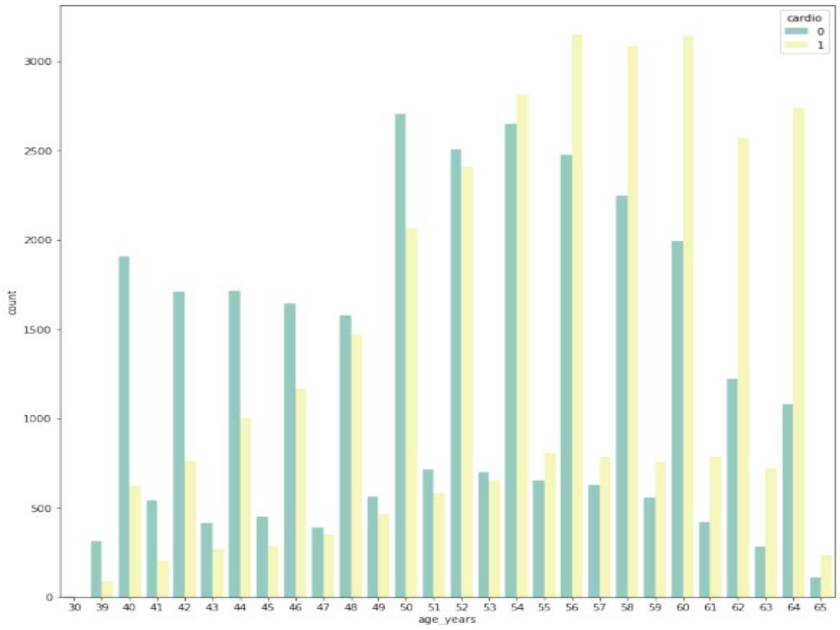


Figure 4: Dependence of sulfur-vascular diseases on age

Let's check if there are linear dependencies between the variables. For this build a correlation matrix that will contain the correlation value for all pairs variables. If there is a high correlation between the variable and the target variable, you can will find a linear relationship between these variables. If non-target variables are have high correlation values, which means that they carry a very similar information, and therefore, one of these variables can be removed, and thus reduce model complexity. Fig. 5 shows the correlation matrix for our dataset. You can see that there is no direct linear relationship between variables. However, it can be noted that the largest correlation values relative to the target variable (cardio) have columns that correspond to blood pressure (ap_hi, ap_lo).

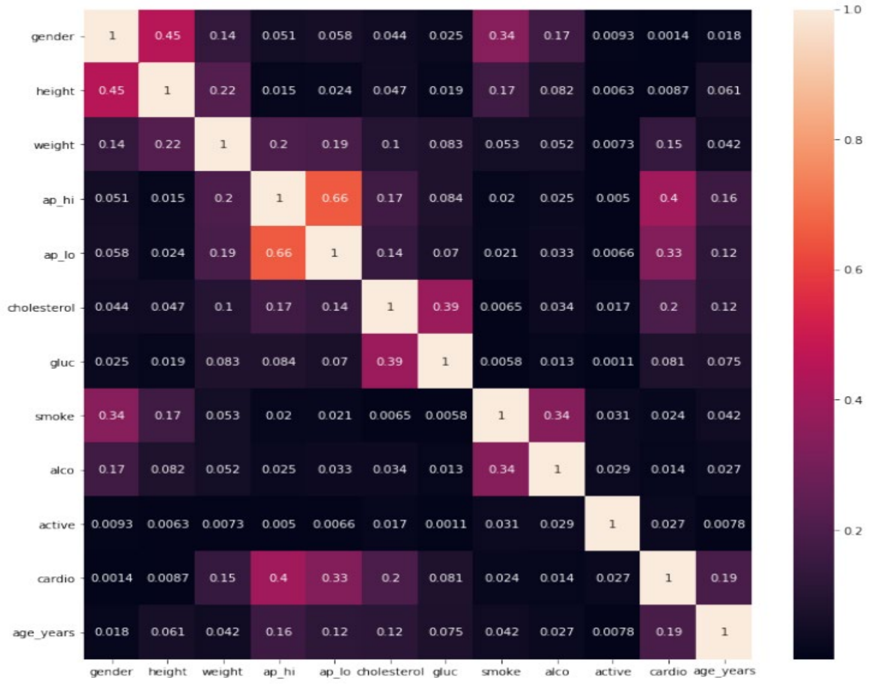
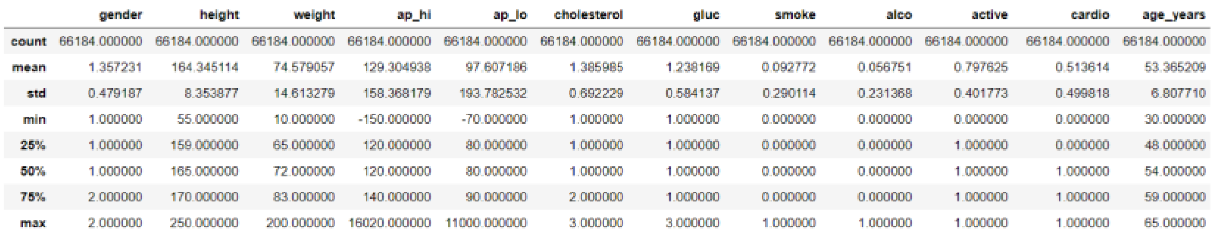


Figure 5: Correlation matrix

We will display statistical data about the columns in the dataset. Fig.6 shows the mean, standard deviation, minimum and maximum values and quantiles for each column of the dataset. From these data it can be seen that the columns ap_hi and ap_lo have extreme points, because the maximum value is very different, which led to the fact that the mean value and the median are also markedly different.



	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_years
count	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000	66184.000000
mean	1.357231	164.345114	74.579057	129.304938	97.607186	1.385985	1.238169	0.092772	0.056751	0.797625	0.513614	53.365209
std	0.479187	8.353877	14.613279	158.368179	193.782532	0.692229	0.584137	0.290114	0.231368	0.401773	0.499818	6.807710
min	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	30.000000
25%	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000	48.000000
50%	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	54.000000
75%	2.000000	170.000000	83.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000	1.000000	1.000000	59.000000
max	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000	65.000000

Figure 6: Statistical measures of dataset columns

Let's construct box diagrams for these columns to visually see the data distribution (Fig. 7). You can see that these columns have extreme data, which are indicated on the graph by points that extend beyond the quantile interval. Delete the lines that contain the extreme points.

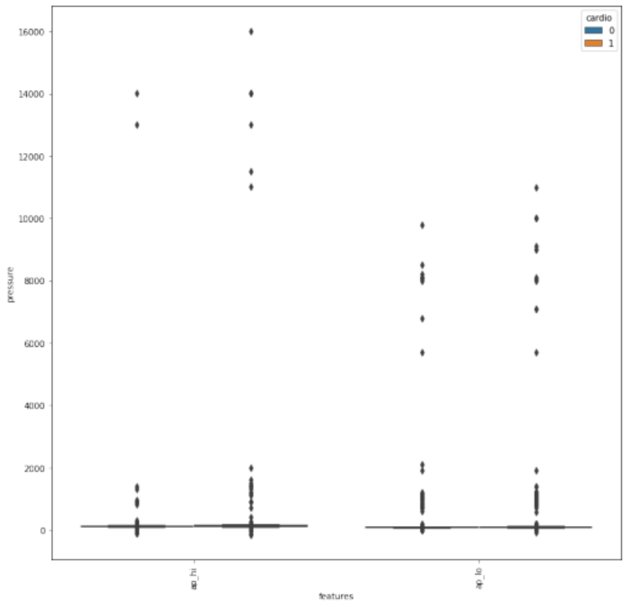
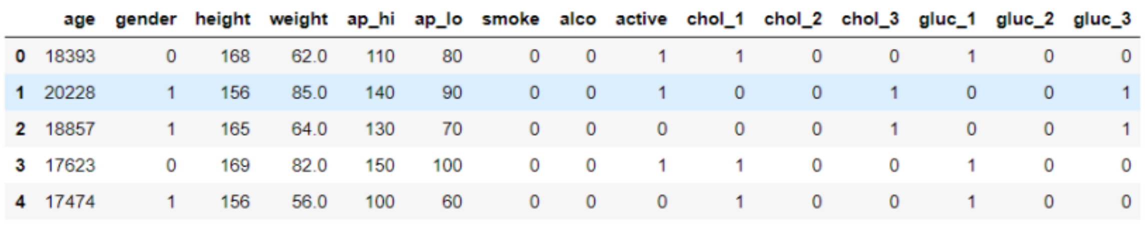


Figure 7: Box charts

We encode categorical columns with one hot encoding. There are 2 such columns in the dataset: "cholesterol" and "gluc". Each of them has three unique values, so after unary coding, each of them will turn into three columns with Boolean values.

After the stages of data analysis and purification, our dataset will look as follows (Fig. 8):



	age	gender	height	weight	ap_hi	ap_lo	smoke	alco	active	chol_1	chol_2	chol_3	gluc_1	gluc_2	gluc_3
0	18393	0	168	62.0	110	80	0	0	1	1	0	0	1	0	0
1	20228	1	156	85.0	140	90	0	0	1	0	0	1	0	0	1
2	18857	1	165	64.0	130	70	0	0	0	0	0	1	0	0	1
3	17623	0	169	82.0	150	100	0	0	1	1	0	0	1	0	0
4	17474	1	156	56.0	100	60	0	0	0	1	0	0	1	0	0

Figure 8: Dataset after data cleaning

4.2. Training a Random Forest Model

Before training, the dataset should be divided into a training set and a test set. Let's do it with a ratio of 80% - training, 20% - test.

A class was chosen to train the random forest model RandomForestClassifier from the scikit-learn library.

In order to improve the accuracy of the model, the search for optimal Grid Search hyperparameters was used. It sorts combinations from the given hyperparameters and chooses the best combination.

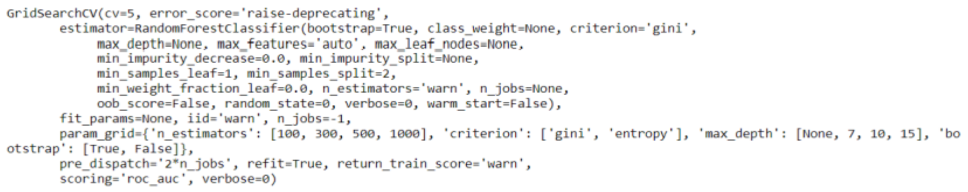
Hyperparameters that were used in the search:

- n_estimators - the number of trees used to build a random forests;
- criterion - the criterion of partitioning a tree;
- max_depth - the maximum depth that the tree can reach, after which construction stops.

As a result, after searching for hyperparameters, the model will be returned, which showed the best predictions on a given metric.

Since the problem of binary classification is solved, the most the objective metric will be roc-auc (area under the error curve) because it is does not depend on the distribution of classes.

Fig. 9 shows the parameters of the model that proved to be the best. You can see that the number of trees was chosen as large as possible, namely 1000, and the maximum depth of the tree is 10.



```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [100, 300, 500, 1000], 'criterion': ['gini', 'entropy'], 'max_depth': [None, 7, 10, 15], 'bo
otstrap': [True, False]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=0)
```

Figure 9: Model parameters

5. Results

In this section, we derive model evaluation indicators.

Fig. 10 show the error matrices for the training and test set, respectively. You can see that the relative distribution of errors is preserved from the training dataset to the test, and therefore the model is good generalized. Also, the model makes fewer False Negatives errors (when the correct prediction is 1 and the model gives 0), which is very important on medical datasets, because it is worse to predict that a person is not sick when in fact, he is sick.

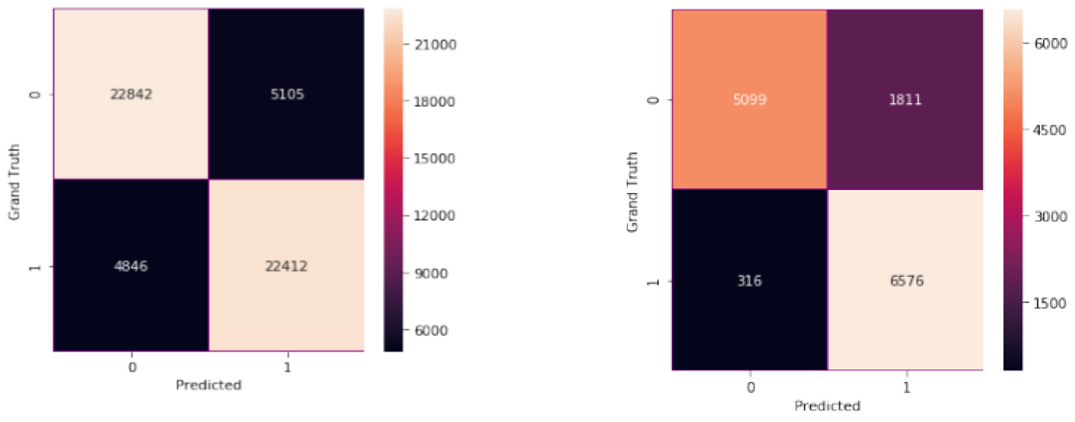
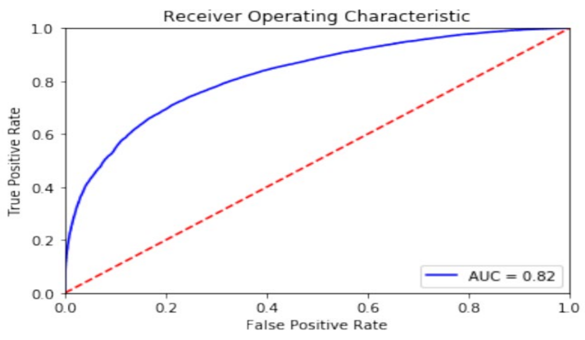
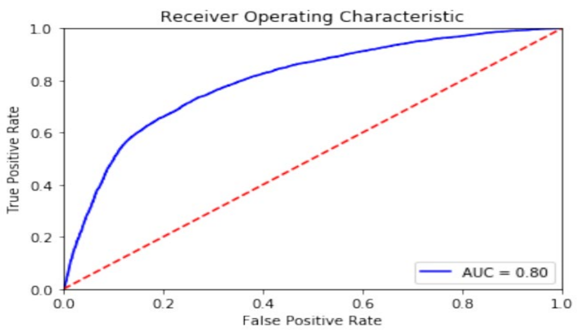


Figure 10: a) Matrix of errors on the training set; b) Error matrix on the test set

Fig. 11. show the error curves (ROC), and calculated the area under them (roc-auc) for the training and test sets, respectively. This metric shows the dependence of the completeness (recall) of predictions, that is, the fate of objects class 1 of all class 1 objects that were correctly excused, to the fate of the objects class 0 that were predicted incorrectly This metric has possible values [0; 1]. As you can see, the metric values are close to the training and test datasets, so the model is well generalized.



a)



b)

Figure 11: a) Roc-auc on the training dataset; b) Roc-auc on the test dataset

6. Conclusions

This paper provides a theoretical analysis of machine learning methods, namely decision trees, a general analysis of ensemble methods, and a detailed analysis of batching and random forests.

The peculiarities of the random forest method, such as the use of bags and subsamples of predictors when the tree is broken, are analyzed to reduce the variability of the model. Theoretical comparative characterization of random forests and selected trees were also carried out.

In practice, the problem of diagnosing cardiovascular diseases was solved using the random forest method. The data set was analyzed (checked the absence and repeated values, linear dependencies, and extreme values), and data was prepared for training and model training. Model results were also analyzed using various metrics (error matrix, roc-AUC).

References

[1] A.Kumar, The Ultimate Guide to AdaBoost Algorithm, What is AdaBoost Algorithm? Artificial Intelligence, 2020. URL: https://www.mygreatlearning.com/blog/adaboost-algorithm/

[2] Y.Kyrychenko, Features of development a system of number plate localization based on the viola-jones method using adaboost? Paradigm of knowledge, Vol 2, No 28, 2018. URL: https://www.naukajournal.org/index.php/Paradigm/article/view/1512/1595

[3] K.V.Murygin, Features of the implementation of the AdaBoost algorithm for detecting objects in images, Shtuchniy telecom, 2009, no 3, pp. 573-581

[4] J.Brownlee, Boosting and AdaBoost for Machine Learning. Machine Learning Algorithms, April 25, 2016. URL: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/
[5] M.I.Jordan1, T.M.Mitchell, Machine learning: Trends, perspectives, and prospects, Artificial intelligence, 2015, 349(5)
[6] J.Friedman, Another approach to polychotomous classification, Machine Learning, 2004
[7] J.Friedman, T.Hastie, R.Tibshirani, Additive logistic regression: a statistical view of boosting. Annals of Statistics, 2000, no 28, pp.337–407
[8] T.Hastie, R.Tibshirani, J.Friedman, The Elements of Statistical Learning. SpringerVerlag, New York, 2001
[9] Y.Lee, Y.Lin, G.Wahba, Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data, Journal of the American Statistical Association, 2004, no 99, pp. 67–81
[10] P.Buhlmann, B.Yu, Boosting with the l2 loss: regression and classification, Journal of the American Statistical Association, 2003, Vol. 98(462)
[11] N.Boyko, N.Tkachuk, Processing of Medical Different Types of Data Using Hadoop and Java MapReduce. In: The 3rd International Conference on Informatics & Data-Driven Medicine (IDDM 2020), Växjö, Sweden, November 19-21, 2020, pp. 405-414
[12] J.Zhu, H.Zou, T.Hastie, Multi-class AdaBoost, Statistics and its Interface, 2006, 2(3). DOI: 10.4310/SII.2009.v2.n3.a8/
[13] J.H.Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, 2001, 29(5), pp. 1189-1232. DOI: 10.2307/2699986
[14] L.Breiman, Prediction Games and Arcing Algorithms, Neural Computation, 1999, 11(7), pp.1493-517. DOI:10.1162/089976699300016106
[15] P.Zhang, G.Zhang, Q.Pan, An ensemble learning method for classification of multiple-label data, Journal of Computational Information Systems, 2015, 11(12), pp.4539-4546. DOI: 10.12733/jcis14783
[16] T.Basyuk, A.Vasilyuk, V.Lytvyn, Mathematical model of semantic search and search optimization, In: Proceedings of the 3rd International conference on computational linguistics and intelligent systems, COLINS-2019, Kharkiv, Ukraine, 2362, 2019, pp.96–105
[17] D.Alonso, F.Merjildo, L.L.Ling, Enhancing the Performance of AdaBoost Algorithms by Introducing a Frequency Counting Factor for Weight Distribution Updating, CIARP 2012, LNCS 7441, 2012, pp.527–534
[18] N.Boyko, K.Kmetyk-Podubinska, I.Andrusiak, Application of Ensemble Methods of Strengthening in Search of Legal Information, Lecture Notes on Data Engineering and Communications Technologies, 77, 2022, pp.188–200