Comparative analysis of machine learning methods to assess the quality of IT services

Maksim A. Bolshakov[a], Igor A. Molodkin[a] and Sergei V. Pugachev[a]

[a] *Saint Petersburg Railway Transport University of Emperor Alexander I, 9 Moskovsky Ave., Saint-Petersburg, 190031, Russia*

Abstract

The article considers the issue of choosing a machine learning method for solving the applied problem of assessing the current state of the quality of IT services. As a method of choice, a comparative analysis of the generally accepted methods of machine learning was carried out using a set of criteria that made it possible to evaluate their effectiveness and efficiency. F-measure is considered as the main criteria, as a generalizing concept of the completeness and accuracy of classification, for each class of states separately and the duration of the training and prediction procedure. All operations were carried out on the same dataset, namely, on the data of the centralized monitoring and management system for the IT infrastructure of Russian Railways in terms of the ETRAN IT service. Due to their heterogeneity and taking into account the practice of applying the Barlow hypothesis, the initial data went through preliminary processing, the algorithm of which is also described in detail in the article

Keywords

machine learning, Barlow hypothesis, F-measure, dataset, data heterogeneity.

1. Introduction

The requirement for software developers and device manufacturers in terms of mandatory monitoring of their performance is an established and mandatory norm. As a result, most of the IT services provided by operating organizations can be assessed not only by the final failure state, but also by the current local characteristics of their work.

Consider the currently operating centralized system for monitoring and managing the IT infrastructure of Russian Railways, which is operated by the Main Computer Center. The huge array of accumulated and constantly updated data of the specified monitoring system allows us to assume the success of using machine learning methods to solve the problem of assessing the quality of IT services by determining the current state of the specified infrastructure and the service applications implemented on it.

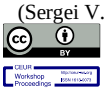
It is possible to assess the state of the IT infrastructure at a certain point in time by predicting its final performance based on the current data of the monitoring system. To do this, it is necessary to solve the problem of classifying the final state of the IT infrastructure, that is, to determine the class label (the type of the final state of the incident / regular operation) based on the current values of the monitoring system metrics. At the same time, for the choice of the implementation method, the number of available classes is not so important - the binary classification is a special case of the multiclass classification, and is not a decisive characteristic when choosing a teaching method. Typically, the nature of the input data, namely the types and formats, have a significant impact on the choice of machine learning methods. At the same time, data preprocessing algorithms do not depend on the chosen training method itself and must ensure the correct use of the initial data as training and test samples for all further methods of solving the problem. [1]

2. Primary data processing

Primary data is understood as the whole set of the characteristics of the IT infrastructure involved in the

Models and Methods for Researching Information Systems in Transport, Dec. 11-12, St. Petersburg, Russia

EMAIL: bolshakovm@yandex.ru (Maksim A. Bolshakov); molodkin@pgups.ru (Igor A. Molodkin); nki.pugachev@yandex.ru (Sergei V. Pugachev)



operability of the specified system, taken by the current monitoring and management system, and the set of service characteristics that make it possible to define this automated system as an IT service. The quality of the obtained primary data, namely their heterogeneity, imply additional processing regardless of the choice of a specific machine learning method. For the correct formation of the training sample, the most suitable programming language is Python version 3.7.4 and the following imported libraries: Pandas and Numpy. The specified libraries must be installed on the script execution server as Site-package libraries. Initially, the data received from the monitoring system is presented as a csv file with a separator | and a set of columns in the following order:mon_obj
metric_name
metric_value
value_time
isIncident
These columns contain the following information:
mon_obj — monitoring object name
metric_name — metric's name
metric_value — metric value at the time of data collection
value_time — date / time when data was collected
isIncident — critical state indicator (at the first stage, a binary classification is highlighted: 1 - critical state, 0 - normal operation).
After that via variable small_columns_list = ['mon_obj','metric_name','metric_value', 'value_time','isIncident'] and using pandas library new DataFrame is formed, which is suitable for usage.
d = pd.read_csv('data/data.csv', sep='|', encoding='utf-8', skipinitialspace=True, skiprows=1, names=small_columns_list, low_memory=False)
Next, you need to define a new composite_metric_name column in DataFrame d, which will have a composite name from the values of the columns with the data about the object of monitoring and the name of the metric for each current:
row.d['composite_metric_name']=['mon_obj']+'_'+d['metric_name]
Then a new variable column_names is declared, consisting of the unique values of the newly generated column
'composite_metric_name'column_names=d['composite_metric_name'].unique()
These steps are necessary to unambiguously compare the metric and its values at any given time. Further, after transposing the DataFrame into a more convenient form and removing unnecessary columns, it will be possible to remove static data as a method to reduce the dimensionality and optimize the use of computing resources without losing data quality:
defdelete_cols_w_static_value():
column_names_with_static_value = []
forcol_name in column_names:
if (merged_df[col_name].nunique() == 1):
column_names_with_static_value.append(col_name)
if 'isIncident' in column_names_with_static_value:
column_names_with_static_value.remove('isIncident')
merged_df.drop(column_names_with_static_value, axis = 1, inplace = True)
After reducing the dimension in this way, it is necessary to recode the text values, that is, replace the existing text values of the metrics with numeric ones by entering new arguments of the objective function. For each current text value, a separate column is created in the current dataset and a value of 1 or 0 is determined, thus obtaining the correct data for analysis and comparability.
def find_strings_column():
df_cols = merged_df.columns
ret_col_names = [];
for col in df_cols:
if (merged_df[col].dtypes == 'object'):
ret_col_names.append(col)
returnret_col_names
The variable string_columns stored list of names of the columns that contain the string values. Next step is to form a dictionary of string values. Values in columns should then be replaced by indexes of elements from the dictionary.
def make_dict_with_strings_column():
ret = {}
forsc in strings_column:
ret[sc] = list(merged_df[sc].unique())
returnret
By using the Save function of the NumPy library, this dictionary is saved to a file.
np.save('data/dict_of_string_values.npy', dict_of_string_values)
And further, using the function
def modify_value_string_column():
for c in dict_of_string_values.keys():
merged_df[c] = merged_df[c].map(lambda x: dict_of_string_values[c].index(x)), all text values are changed to numeric values.
At the end of the primary data processing, the data frame values are standardized so that the variance is unitary, and the average value of the series is 0 - this can be done using the built-in Standardscaler tool or through a self-written function:
def normalize_df():
cols = list(merged_df.columns)
for cl in cols:
x_min = min(list(merged_df[cl]))
x_max = max(list(merged_df[cl]))
merged_df[cl]= merged_df[cl].map(lambda x: ((x - x_min)/(x_max - x_min)))

For the convenience, the final result should be saved to a file.
merged_df.to_csv(path_or_buf='data/normalised_df.csv', sep='|',index=False)
As a result of the actions performed, a dataset was obtained that is suitable for applying various machine learning models with the following characteristics:
• The number of records in the training set is 10815 (9323 class 0, 1492 class 1);
• The number of records in the test set is 3605 (3297 of class 0, 308 of class 1).

3. Criteria for comparing results

When solving any supervised learning problem, there is no most suitable machine learning algorithm - there is such a thing as the "No Free Lunch" theorem, the essence of which is the impossibility to unambiguously approve the best machine learning method for a specific task in advance. The applicability of this theorem extends, among other things, to the well-studied area of problems of binary classification, therefore, it is imperative to consider a set of methods and, based on the results of practical tests, evaluate the effectiveness and applicability of each of them. [2]
The effectiveness and efficiency will be primarily assessed by the most common and user-understandable criterion Accuracy.
Accuracy is a measure that indicates the proportion of correct decisions of the classifier:

$$Accuracy = \frac{P}{N},$$

where P is the number of states correctly identified by the system, and N is the total number of states in the test sample.
However, for the problem being solved, this criterion is not enough, since in its calculation it assigns the same weight to all final states (classes), which may be incorrect in the considered case of non-uniform distribution of time moments over the final states. Thus, for a more correct comparison and taking into account the fact that in the example under consideration the share of normal states is much greater than the share of critical states, the assessment of the classifiers should be based, among other things, according to the following criteria: Precision (accuracy - in a calculation other than Accuracy), and Recall (completeness).
These criteria are calculated separately for each summary class, where Precison is the proportion of situations that really belong to this class relative to all situations that the system has assigned to this class. System completeness (Recall) is the proportion of situations found by the classifier that belong to a class relative to all situations of this class in the test sample. More clearly, these criteria can be presented through the contingency (contingency) table, also built separately for each final class [3]

Class (1/0)		Training Sample	
		Positive	Negative
Result	Positive	TP	FP
	Negative	FN	TN

These results are used directly in the calculation of the criteria for the classifier as follows:

$$Pr\,e\,cision = \frac{TP}{TP+FP};$$

$$Recall = \frac{TP}{TP + FN}$$

As can be seen from the calculation algorithm, these criteria provide a more complete understanding of the quality of the classifier's work. It would be logical to say that the higher the accuracy and completeness, the better, but in reality, maximum accuracy and completeness are not achievable at the same time and it is necessary to find a balance between these characteristics. To do this, you need to use the F measure, which is the following calculated value:

$$F = (\beta^2 + 1) \cdot \frac{Pr\,ecision \cdot Re\,call}{\beta^2 \cdot Pr\,ecision + Re\,call},$$

where 0<β<1, if the greater weight for the selection of the classifier has accuracy and β> 1 with the preference for completeness. In the case β = 1, a balanced F-measure is obtained, denoted as:

$$F_1 = 2 \cdot \frac{Pr\,ecision \cdot Re\,call}{Pr\,ecision + Re\,call}.$$

In addition, for all classifiers, we will apply an estimate of the duration of the execution of Wall time operations, by which we will count the time from the moment the model is created and the beginning of training until the moment the values of the performance assessment metrics are issued based on the comparison of the predicted results with the test sample.
All work is carried out on a computer complex with the following characteristics:
Intel Core i9-9980XE 3.00 Ghz, 128 Mb, 4xNVIDIA RTX 2080Ti 11 Gb, 1 TB PCIe SSD.

4. K-nearest neighbors (knn) algorithm

The specified algorithm works as follows - let a training sample of pairs "object (state characteristic) - response (state class)" be given:

$$X^m = \{(x_1, y_1), \dots (x_m, y_m)\},$$

then the distance function P (x, x ') is given on the set of objects. This function should be a reasonably adequate model of object similarity. The larger the value of this function, the less similar two objects x, x 'are [4]. For an arbitrary object μ, we arrange the objects of the training sample in the order of increasing distances to μ:

$$P(\mu_1; p(\mu_i, x_{1;\mu}) < p(\mu_i, x_{2;\mu}) \dots < p(\mu_i, x_{m;\mu}),$$

where some x (1; μ) denotes the object of the training sample that is the i-th neighbor of the object μ. We introduce a similar notation for the answer to the i-th neighbor y (i; μ). Thus, an arbitrary object μ generates a new renumbering of the sample. In its most general form, the nearest neighbors algorithm is:

$$a(\mu) = argmax_{y \in Y} \sum_{i=1}^{m} [y(x_{i;\mu}) = y] \cdot w(i, \mu),$$

where w (i, μ) is a given weight function that estimates the degree of importance of the i-th neighbor for classifying the object μ, while this function cannot be negative and does not increase with respect to i.
For the k nearest neighbors method:

$$(i; \eta) = [i \le k]$$

The qualitative characteristics of using the KNeighborsClassifier algorithm with the n_neighboors = 10 parameter are as follows:

Accuracy 0.949

	Precision	Recall	F1 score
1	0.63	0.97	0.77
0	1.0	0.95	0.97

Wall time: 1 min 15 s

It is clearly seen that the accuracy assessment in the form of Accuracy is not enough for a comparative analysis - with a value of this indicator of 0.949 - almost 95% accuracy - we see that the accuracy of determining the final state 1 (inoperability) through the Precision characteristic is only 63%. As a result of the work, one should record the F-Measure value for each class and the total duration of the classifier's work.

5. Logistic regression

To predict the probability of occurrence of a certain event by the values of the set of signs, a dependent variable Y is introduced, which takes values 0 or 1 and a set of independent variables x1, ... xn, based on the values of which it is required to calculate the probability of accepting a particular value of the dependent variable. [5]
Let the objects be specified by numerical features:

$$f_j: X \to R, j = 1 \dots n$$

and the space of feature descriptions in this case $X=R_n$. Let Y in this case be a set of class labels and a training set of object-response pairs

$$X_m = \{(x_1, y_1), \dots, (x_m, y_m)\}.$$

Consider the case of two classes: Y={-1,+1}. In logistic regression, a linear classification algorithm aX→Y of the form:

$$a(x, w) = sign\left(\sum_{j=1}^{n} w_j f_i(x) - w_0\right) = sign\langle x, w\rangle,$$

where w_j – scale of j-th feature, w_0 − decision threshold $w=(w_0,\dots,w_n)$ − scales vector, ⟨x,w⟩ − dot product of the feature description of an object by a vector of weights. It is assumed that the zero feature is artificially introduced: $f_0(x) = -1$.
The task of training a linear classifier is to adjust the weight vector w based on the sample X_m. In logistic regression, for this, the problem of minimizing empirical risk is solved with a loss function of the following form:

$$Q(w) = \sum_{i=1}^{m} ln(1 + exp(-y_i < x_i, w >)) \to min\, w$$

After the solution w is found, it becomes possible not only to perform classification for an arbitrary object x, but also to estimate the posterior probabilities of its belonging to the existing classes:

$$P\{y|x\} = \sigma(y < x, w >), y \in Y,$$

$$\sigma(z) = \frac{1}{1 + exp^{-z}} - sigmoid\ function$$

Qualitative characteristics of the application of the specified model with the following parameters
LogisticRegression (multi_class='ovr', solver='lbfgs'):

Accuracy 0.971

	Precision	Recall	F1 score
1	0.76	0.98	0.85
0	1.0	0.97	0.98

Wall time: 1 min 10 s

The results shown by this classifier are better relative to the previous method with a comparable duration of operation, however, the work on identifying faulty situations (class 1) does not yet guarantee satisfactory operation in industrial mode.

6. Naive Bayesian classifier

The Bayesian classification is based on the maximum probability hypothesis, that is, an object d is considered to belong to the class cj (cj ∈ C) if the highest posterior probability is achieved:

$$max\, P\,(c_j|d).$$[6]

Bayesian formula:

$$P(c_j|d) = \frac{P(c_j)\cdot P(d|c_j)}{P(d)} \approx P(c_j)P(d|c_j),$$

where $P(d|c_j)$- the probability of encountering an object d among objects of class c_j, and $P(c_j)$,P(d) – prior probabilities of the class c_j and d.
Under the "naive" assumption that all features describing the classified objects are completely equal and not related to each other, then P (d | c_j) can be calculated as the product of the probabilities of encountering a feature x_i ($x_i \in X$) among objects of class c_j:

$$P(d|c_j) = \prod_{i=1}^{|x|} P(x_i|c_j),$$

where $P(x_i|c_j)$ - probabilistic assessment of the contribution of a feature x_i to the fact that $d \in c_j$.[7]
In practice, when multiplying very small conditional probabilities, there can be a loss of significant digits, and therefore, instead of the estimates of the probabilities P (x_i | c_j), the logarithms of these probabilities should be used. The logarithm is a monotonically increasing function, and, therefore, the class c_j with the largest value of the logarithm will remain the most probable. In this case, the decision rule of the naive Bayesian classifier takes the following form:

$$C^* = arg_{c_j \in C}\, max \left[log\, P\,(c_j) + \sum_{i=1}^{x} P(x_i|c_j) \right]$$

.

The resulting values of the MultinomialNB classifier from the Sklearn library turned out to be the following:

Accuracy 0.874

	Precision	Recall	F1 score
1	0.40	0.96	0.57
0	1.0	0.87	0.93

Wall time: 289 ms

The considered classifier, based on its description, works fundamentally differently - the speed of its work is much higher - however, the qualitative criteria, the main of which F-measure, are inferior to past classifiers.

7. Decision tree methodology

With this algorithm, the tree is built from top to bottom - from the root node to the leaves. At the first step of training, an "empty" tree is formed, which consists only of the root node, which in turn contains the entire training set. Next, you need to split the root node into subsets, from which the descendant nodes will be formed. For this, one of the attributes is selected and rules are formed that divide the training set into subsets, the number of which is equal to the number of unique values of the selected attribute. [8]
As a result of splitting, p (according to the number of attribute values) subsets are obtained and, therefore, p descendants of the root node are formed, each of which is assigned its own subset. Then this procedure is recursively applied to all subsets until the stop condition is reached.
For example, a partitioning rule should be applied to the training set, in which the attribute A, which can take p values: a_1, a_2, ..., a_p, creates p subsets S_1, S_2, ..., S_p, where examples will be distributed, in which the attribute A takes the corresponding value.
Moreover, N (C_j, S) is the number of examples of the class C_j in the set S, then the probability of the class C_j in this set is determined by the expression:

$$P = \frac{N(C_j S)}{N(S)},$$

where N (S) is the total number of examples in the set S.
The entropy of the sets S will be expressed as:

$$Info(S) = -\sum_{i=1}^{m} \frac{N(S_i)}{N(S)} \cdot log(\frac{N(C_j S)}{N(S)})$$

It will demonstrate the average amount of information required to determine the class of an example from the set S.
The same estimate, obtained after partitioning the set S by attribute A, can be written as:

$$Info_A(S) = \sum_{i=1}^{k} \frac{N(C_j S)}{N(S)} \cdot Info(S_i),$$

where S_i - i-th node, obtained by splitting by attribute A. After that, to choose the best branching attribute, you should use the criterion of the form:

$$Gain(A) = Info(S) - Info_A(S)$$

This criterion is called the criterion of information gain. This value is calculated for all potential split attributes and the one that maximizes the specified criterion is selected for the division operation.
The described procedure is applied to subsets S_i and further, until the values of the criterion cease to increase significantly with new partitions or a different stopping condition is met. In this case, when in the process of building a tree, an "empty" node is obtained, where not a single example will fall, then it must be converted into a leaf that is associated with the class most often found in the immediate ancestor of this node.
The DecisionTreeRegressor classifier with parameters Random_state = 15 and Min_samples_leaf = 25 showed the following characteristics:

Accuracy 0.964

	Precision	Recall	F1 score
1	0.71	0.97	0.85
0	1.0	0.97	0.98

Wall time: 975ms

When working with the decision tree method, the results are similar to the logistic regression method, however, the duration of training and forecasting in the decision tree method is much longer, which, with equal qualitative characteristics, puts the results of this method higher than others.

8. Gradient boosting method

Gradient boosting is a machine learning method that creates a decisive forecasting model in the form of an ensemble of weak forecasting models, usually decision trees, essentially developing the decision tree method. During boosting, the model is built in stages - an arbitrary differentiable loss function is also optimized in stages. [9]
For the problem of object recognition from a multidimensional space X with a label space Y, a training sample $\{x_i\}_{i=1}^N$ is given, where $x_i \in X$. In addition, the true values of the class labels for each object $\{y_i\}_{i=1}^N$ are known, where $y_i \in Y$. The solution to the prediction problem is reduced in this case to the search for a recognizing operator who can predict the labels as accurately as possible for each new object from the set X.
Let a family of basic algorithms H be given, each element $h(x,a) \in H: X \to R$ of which defined by some vector of parameters $a \in A$.
In this case, it is necessary to find the final classification algorithm in the form of the following composition: $F_M(x) = \sum_{m=1}^{M} b_m h(x, a_m), b_m \in R, a_m \in A$.
However, the selection of the optimal set of parameters $\{a_m, b_m\}_{m=1}^M$ is an extremely time-consuming task, therefore the construction of this composition should be carried out by means of "greedy" growth, each time adding the summand, which is the most optimal algorithm, to the sum.
At the step when the optimal classifier $F_{(m-1)}$ of length m - 1 has already been assembled, the task is reduced to finding a pair of the most optimal parameters $\{a_m, b_m\}$ for the classifier of length m:

$$F_M(x) = F_{m-1}(x) + b_m h(x, a_m), b_m \in R, a_m \in A$$

Optimality is understood here in accordance with the principles of explicit maximization of margins - this means that a certain loss function $L(y_i, F_m(x_i)) \to \min$ is introduced, showing how much the predicted answer $F_m(x_i)$ differs from the correct answer y_i. Next, you need to minimize the functionality of this error:

$$Q = \sum_{i=1}^{N\Sigma} L(y_i, F_m(x_i)) \to min$$

It should be noted that the error functional Q is a real function depending on the points $\{F_m(x_i)\}_{i=1}^N$ in the N-dimensional space, and this function is minimized by the gradient descent method. As the point for which the optimal increment should be found, we define F_{m-1} and the error gradient is expressed as follows:

$$\nabla Q = \left[\frac{\partial Q}{\partial F_{m-1}}(x_i)\right]_{i=1}^N = \left[\frac{\partial (\sum_{i=1}^N L(y_i, F_{m-1}))}{\partial F_{m-1}}(x_i)\right]_{i=1}^N = \left[\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}(x_i)\right]_{i=1}^N$$

By virtue of the gradient descent method, it is most beneficial to add a new term as follows:

$$F_m = F_{m-1} - b_m \nabla Q, b_m \in R,$$

where b_m is selected by linear search over real numbers R:

$$b_m = argmin \sum_{i=1}^{N} L(F_{m-1}(x_i) - b \nabla Q_i)$$

However, ∇Q is only a vector of optimal values for each object xi, and not a basic algorithm from the

family H, defined by ∀x∈X, so it is necessary to find $h(x,a_m)\in H$ that is most similar to ∇Q. To do this, it is necessary to re-minimize the error functionality using an algorithm based on the principle of explicit minimization of indents:

$$a_m = argmin \sum_{i=1}^{N} L(\nabla Q_i, h(x_i, a)) \equiv$$
$$\equiv train(\{x_i\}_{i=1}^{N}, \{\nabla Q_i\}_{i=1}^{N}),$$

which in turn corresponds to the basic learning algorithm.
Next, you need to find the coefficient b_m using linear search:

$$b_m = argmin \sum_{i=1}^{N} L(F_{m-1}(x_i) - b \cdot h(x_i, a_m)$$

The GradientBoostingClassifier implemented according to this algorithm with the parameters learning_rate = 0.3; n_estimators = 10; verbose = 1, subsample = 0.5 showed the following results:

Accuracy 0.993

	Precision	Recall	F1 score
1	0.94	0.99	0.97
0	1.0	0.99	1.0

Walltime: 3.95s

The gradient boosting method, like the decision tree method, is a method of enumerating classification parameters, which, in turn, determines their relative comparability in terms of training duration. However, the time spent in boosting to determine the ensemble of decision trees has a colossal effect - the accuracy in terms of the final state classes is the highest for this method.

9. Neural network

Taking into account the available analysis of data sources of the considered IT infrastructure monitoring system and the nature of this data, the MLP (Multi-Layer Perceptron) type was defined as a neural network - due to the absence of video surveillance systems as sources, and, consequently, the problem of video recognition of the classical verbose neural network of direct distribution will be sufficient to determine the effectiveness of its application. [10]
An MLP network is any multilayer neural network with a linear conductance function and a monotone limited activation function g_v common to all hidden neurons, depending only on the variety $t=s(v)-w_v$, which is a "smoothed step", as a rule, a hyperbolic tangent:

$$tng(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$

or logistic function:

$$\sigma(t) = \frac{tng(\frac{t}{2})+1}{2} = \frac{1}{1+e^{-t}}.$$

The activation function of output neurons can also be the same "smoothed step", or it can be identical $g_v(t)=t$, that is, each neuron v calculates the function:

$$u(v) = g_v\big((\sum w_e u(e)) - w_v\big).$$

The parameters of the edges we are called their weights, and the parameters of the vertices w_v are called displacements. In this case, which activation function is chosen - hiberbolic tangent or logistic - is indifferent: for any multilayer perceptron with an activation function tng calculating the function $F_{tng}(w,x)$, the same perceptron, in which the activation function in intermediate layers is replaced by a logical function σ, calculates the same the function itself for some other value of the parameter w ':

$$F_{tng}(w,x) = F_\sigma(w',x).$$

In accordance with the ideology of minimizing empirical risk with regularization of training of the perceptron calculating the function F(w,x), this is the search for a vector of weights and biases that minimizes the regularized total error:

$$E_T(w) = \varphi(w) + \sum_{i=1}^{N} E(F(w,x_i), y_i)$$

on some training set $T=((x_1,y_1),...(x_n,y_n))$. [11]
Training is most often carried out by the classical method of gradient descent; for its applicability, the activation functions of all neurons and the error and regularization functions must be differentiable. Practice shows that the speed of this algorithm is often inferior to others because of the huge dimension of the parameter w and the absence of explicit formulas for the derivatives of the function F with respect to w. [11]
The results of applying the MLPClassifier (max_iter = 100, random_state = 10) are as follows:
Accuracy 0.992

	Precision	Recall	F1 score
1	0.93	0.99	0.95
0	1.0	0.99	1.0

Wall time: 1 min 5 s
The neural network, as the user often expects, has shown high results of a qualitative assessment - they are essentially equal to the results of gradient boosting.

However, the duration of training and forecasting for a neural network is much longer than for methods based on building ensembles of decision trees - 1 minute versus several seconds.

10. Conclusion

The results of the application of various machine learning methods clearly prove the postulate of the "NoFreeLunch" theorem - it is the experimental tests that allow you to choose the most appropriate algorithm for solving a specific problem, taking into account specific initial data. In this case, it should be noted once again that the Accuracy characteristic is practically useless in comparing the results - it is more correct to evaluate the results by F-measure, and this should be done separately for each class.

Based on the applied sense of the task - to provide better monitoring of the IT infrastructure operation - the characteristics of training methods for data class "1" are more important, that is, for cases of real failures and infrastructure failures. At the same time, errors for class "0", in fact, will be additional incidents and, therefore, require additional labor from technical support specialists, which is certainly critical, but less important in comparison with the omission of real failures and failures. It is also worth noting the time parameters of the methods - the spread is truly colossal, from 289 milliseconds to 1 minute 15 seconds.

When comparing the comparison criteria, it is clearly seen that the gradient boosting method showed the optimal results of work - with a higher speed of this algorithm, it was able to learn better than other algorithms. When replicating an application on already large data sets (all IT services, a larger analysis horizon), training time is extremely important. Understanding this and the nature of the initial data, namely the absence of video and photo images in the initial data, allow us to conclude that the gradient boosting method is more than sufficient for solving the problem and using a neural network (showed similar results with a longer training duration) at this stage development of the considered IT infrastructure monitoring system in terms of the method of collecting information is not required.

References

[1] Bolshakov M.A. Preparation of a monitoring system for IT infrastructure for models of critical states based on neural networks // Science-intensive technologies in space research of the Earth. 2019. No. 4. p. 65-71.

[2] D.H. Wolpert, W.G. Macready No Free Lunch Theorems for Optimization // IEEE Transactions on Evolutionary Computation. 1997. №1. C. 1.

[3] G. Upton. Analysis of contingency tables. Translated from English and preface by Yu. P. Adler. Moscow. Finance and statistics. 1982. 143 p.

[4] M. Parsian Data Algorithms. Newton: O'Reilly Media, Inc., 2015. 778 c.

[5] D.W. Hosmer, S. Lemeshow Applied Logistic Regression. 2nd Ed. New York: John Wiley & sons, INC, 2000. 397 c.

[6] S. Ranganathan, K. Nakai, C. Schonbach Bayes′ Theorem and Naive Bayes Classifier. Encyclopedia of Bioinformatics and Computational Biology, Volume 1, Elsevier, c. 403-412. 2017

[7] Barber D. Bayesian Reasoning and Machine Learning. Cambridge University Press, c. 2012.

[8] T. Hastie The Elements of Statistical Learning // Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2 изд. Springer, 2008. 764 c.

[9] J.H. Friedman, Stochastic Gradient Boosting. Technical report. Dept. of Statistics, Stanford University, 1999.

[10] T. Windeatt Ensemble MLP Classifier Design. In: Jain L.C., Sato-Ilic M., Virvou M., Tsihrintzis G.A., Balas V.E., Abeynayake C. (eds) Computational Intelligence Paradigms. Studies in Computational Intelligence, vol 137. Springer, Berlin, Heidelberg

[11] V. Vapnik Principles of Risk Minimization for Learning Theory // Advances in Neural Information Processing Systems 4 (NIPS 1991). 1992. №4. C. 831-838.

[12] Gasnikov A.V. Modern numerical optimization methods. Universal Gradient Descent Method: A Tutorial. Moscow: MFTI, 2018.286 p. 2nd Ed