ANNOTATION

The graduate qualification work contains 74 pages of text, including 43 figures. The work is devoted to the development of the system of prediction and fault detection on laser cutting machines Navigator KS 12V. The peculiarity of the created system is the use of the latest achievements in the field of intellectual data analysis, as well as uniqueness.

Keywords - data mining, neural networks, clustering, time series, time series analysis, industrial internet of things.

ANNOTATION

Graduation paper contains 74 pages of text, including 43 pictures. The work is devoted to the development of a system for predicting and detecting faults on machines laser cutting Navigator KS12V. A feature of the created system is the use of the latest achievements. in the field of data mining, as well as well as uniqueness.

Keywords - data mining, neural networks, clustering, time series, time series analysis; industrial internet of things.

CONTENTS.

INTRODUCTION

The main purpose of this thesis is to develop a module for prediction and detection of errors and failures on laser cutting machines Navigator KS12V of VNITEP company. The problem of predicting failures is relevant not only for machines of this model, but also for other various types of production devices, since there are no generally accepted standards for tasks of this kind, and the tasks themselves are solved experimentally on the basis of the latest achievements of science.

The motivation for creating a system of fault prediction and detection is to save resources of Cespel, which uses Navigator machines, namely to reduce time and money spent on diagnostics and troubleshooting.

To determine the sources and types of faults in the thesis work, a functional and modular description of the Navigator KS12V machine tool was given. On the basis of this description the error and fault states were determined, as well as on the basis of recommendations of VNITEP engineers.

The solution to the problem posed required the identification of the best methods and approaches. For this purpose, the thesis investigated and analyzed existing solutions aimed at predicting and detecting device faults. All solutions described in the work are based on the principles of data mining and time series analysis. These two areas have also been described in the paper.

In the process of analyzing the subject area, the tasks that had to be solved to achieve the goal were identified. All tasks were listed.

In the process of analysis, the advantages and disadvantages of existing solutions were identified, and the choice of tools and methods was made on the basis of

analysis, namely a set of libraries for working with data using the Python language and methods for predicting and detecting failures.

The LSTM neural network was used to predict the time series of data from the machine tool. The paper analyzes the architecture of this neural network and describes additional methods for optimizing the results of this neural network.

Predicted values as well as general trends of the laser parameters: temperature and power are used to detect errors and malfunctions on the machine. The main criteria for faults are that the laser temperature and laser power go beyond the extremes for a long time.

In order to identify patterns for machine states, the kShape algorithm was used, which is aimed at clustering multivariate time series. This algorithm is the best for clustering time series. The paper describes the principles of the algorithm in comparison with other algorithms. Also, it is worth noting that the identified patterns are used by the gradient bousting algorithm, which is also described in the paper.

The developed module is a part of Omnicube software system. In this paper we described the system and the principles of integration of the developed module into the system.

1 SUBJECT MATTER ANALYSIS

1.1 The task of error and failure detection and prediction on the Navigator laser cutting machine

The main object of research of this work is the Navigator grain cutting machine of VNITEP company. This machine is installed in the company Cespel, engaged in the production of tank trucks and semi-trailers for various types of transportation: gasoline tankers, grain carriers, dump trailers, cement trucks, etc.

We will describe the functional and modular composition of this machine, as well as the problem of detecting and predicting errors and failures for the machine. In addition, we describe an approach to solve this problem through automation.

1.1.1 Description of the complex Navigator KS12V

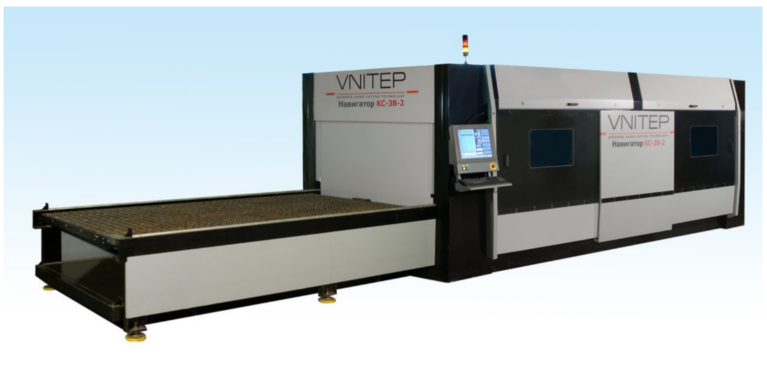


Figure 1 - Appearance of the complex Navigator KS12V

Navigator KS12V (Figure 1) is a sheet metal processing complex with a fiber laser, linear synchronous motor and numerical control (NC). Development and

This complex is supported by the Russian company VNITEP. The machine is the main product of this company.

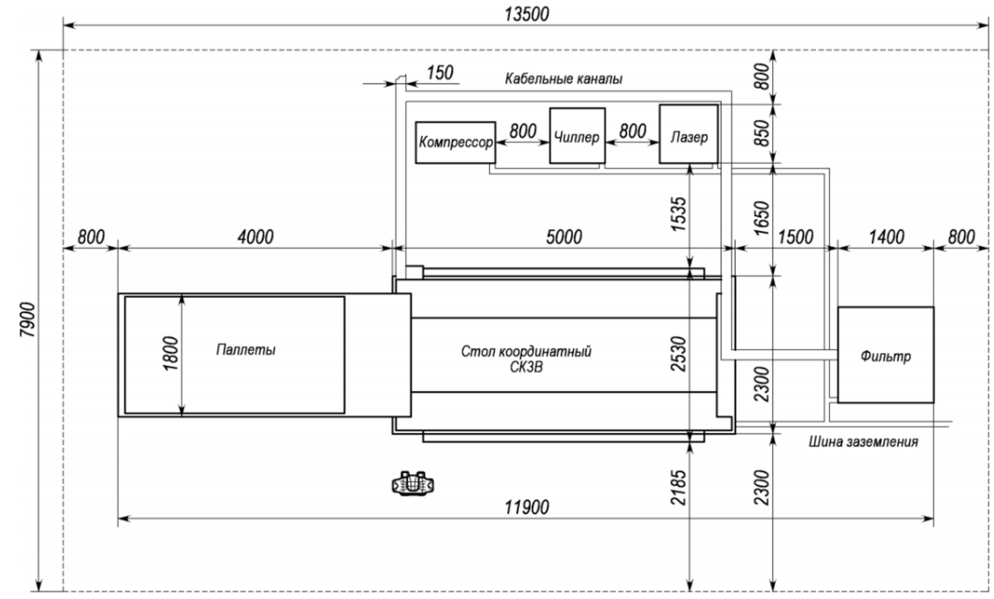


Figure 2 - Graphical description of the main machine parts

The complex consists of the following main modules (Figure 2): coordinate table, ytterbium laser, compressor, filtration and ventilation unit, chiller. The coordinate table consists of a CNC console with software, linear motors, linear ball guides, cable channels, safety channels, pallets for collecting technological waste, and an optical measuring system.

The machine's capabilities cover laser cutting tasks for most metals of various thicknesses up to 30 mm. Main metals used: armored steel, stainless steel, zirconium, brass.

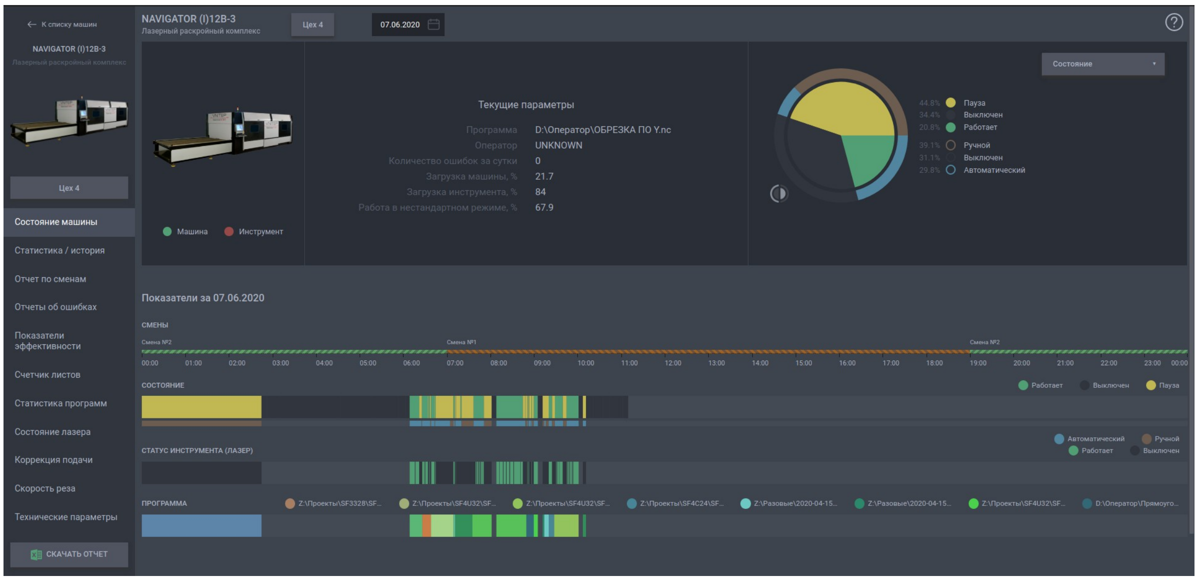
To operate the machine, the operator loads a program via the TNC's interface. The downloaded program is a set of machine instructions that the machine executes to cut a specific part in a specific format.

The object of the diploma work is the model of the described machine tool. Several machines of this model are used in Cespel company.

The task of the thesis will be solved in the context of the Omnicube system.

Omnicube develops and provides a universal platform for solving various tasks of intelligent monitoring in enterprises of various industries: industry, intelligent services for buildings, personnel monitoring, environmental monitoring, services for agricultural enterprises, tools for medical institutions. [45]

One of the clients of Omnicube is Cespel enterprise. Omnicube provides the functionality of collecting, processing and displaying data from the company's devices: machines, plants, production robots. The data itself is collected from integrated and installed sensors of these devices. In addition, Omnicube provides a user interface for the management and employees of Cespel. This interface shows statistics on various parameters and displays various statistical indicators.



Текущие параметры

Программа	D:\Оператор\ОБРЕЗКА ПО Y.nc
Оператор	UNKNOWN
Количество ошибок за сутки	0
Загрузка машины, %	21.7
Загрузка инструмента, %	84
Работа в нестандартном режиме, %	67.9

44.8%	●	Пауза
34.4%	●	Выключен
20.8%	●	Работает
39.1%	○	Ручной
31.1%	○	Выключен
29.8%	○	Автоматический

ПРОГРАММА: ● Z:\Проекты\SF3328\SF... ● Z:\Проекты\SF4U32\SF... ● Z:\Проекты\SF4U32\SF... ● Z:\Проекты\SF4C24\SF... ● Z:\Разовые\2020-04-15... ● Z:\Разовые\2020-04-15... ● Z:\Проекты\SF4U32\SF... ● D:\Оператор\Прямоуго...

Figure 3 - Main machine performance indicators

The main indicators (Figure 3) that are displayed in the interface are: status, laser status, running program. All of them

The status parameter shows the current operating mode of the machine and represents three modes: working (work), paused (off). The status parameter displays the current operating mode of the machine and represents three modes: working (work), off (off), pause. The tool (laser) status parameter displays the statistics of the machine laser operation and can have the following modes: automatic, manual, working, off. The working mode represents the use of the laser under the supervision of the operator. In addition, the main indicator is also the statistics of used programs for the machine.

All statistics are kept daily, as the work is done in shifts. These statistics can be checked in the statistics-history section. In addition, statistics are collected in the form of a shift report for each operator.

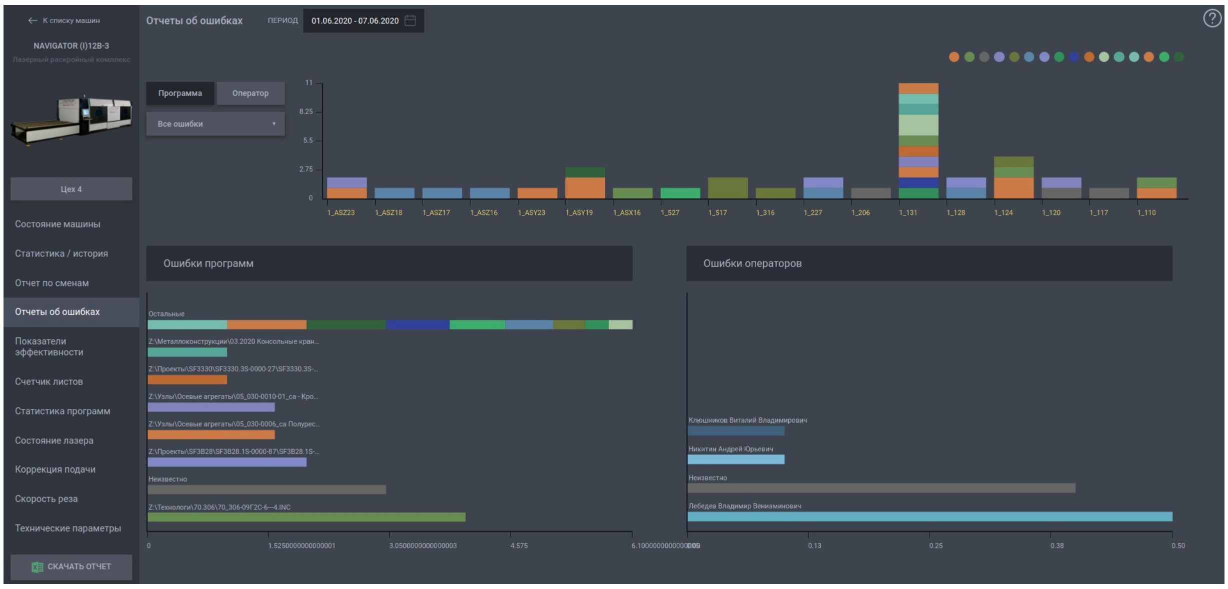


Figure 4 - Report on program and operator errors

The "Error Reports" section (Figure 4) displays statistics on faults of running programs and operator errors. In the main graph you can see the number of occurrences of an error (Y axis) having a certain code (X axis)

The "Performance Indicators" section (Figure 5) contains machine utilization, efficiency factor and tool utilization factor. The "Machine utilization" parameter represents the time

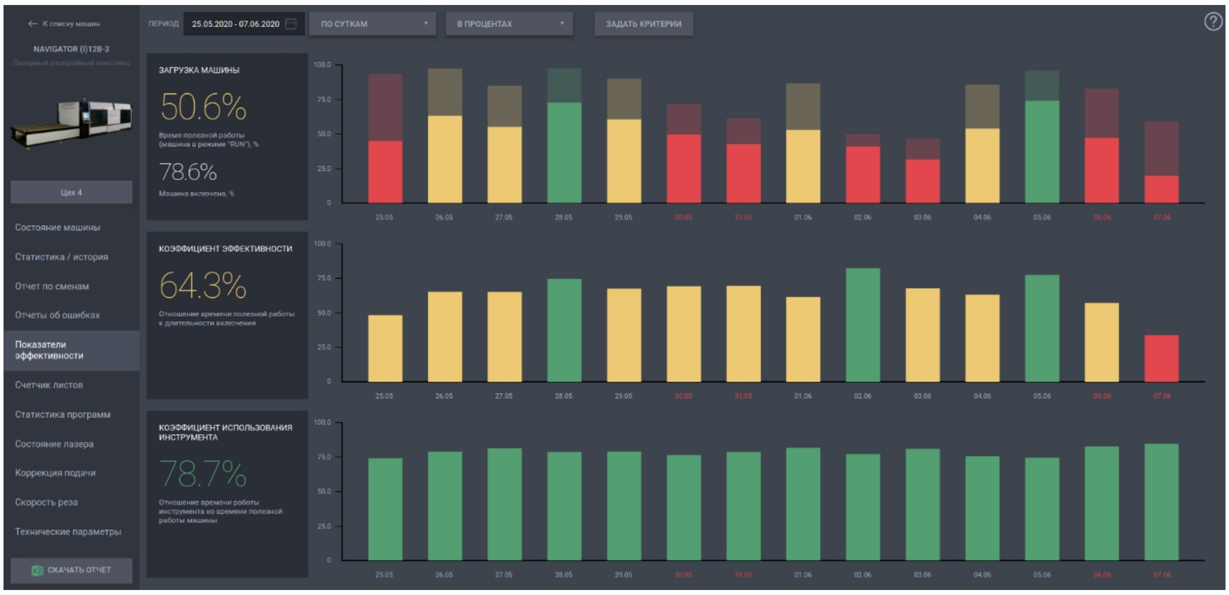


Figure 5 - Performance indicators

me of useful work. "Efficiency factor" - the ratio of useful work time to the duration of switching on. "Tool utilization factor" - ratio of tool operating time to machine operating time. All indicators are expressed in percentages or in hours. It is also possible to check the indicators both by shifts and by hours.

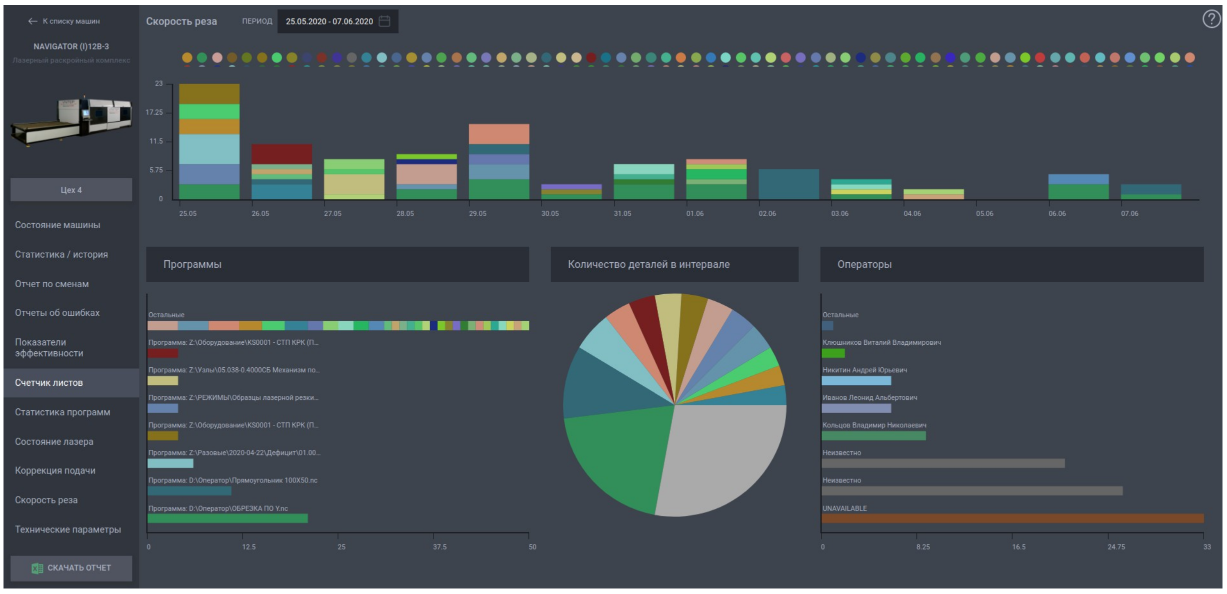


Figure 6 - Sheet counter statistics

The "Sheet counter" section (Figure 6) displays statistics on the number of metal sheets loaded into the machine. You can select the desired period to view the statistics.

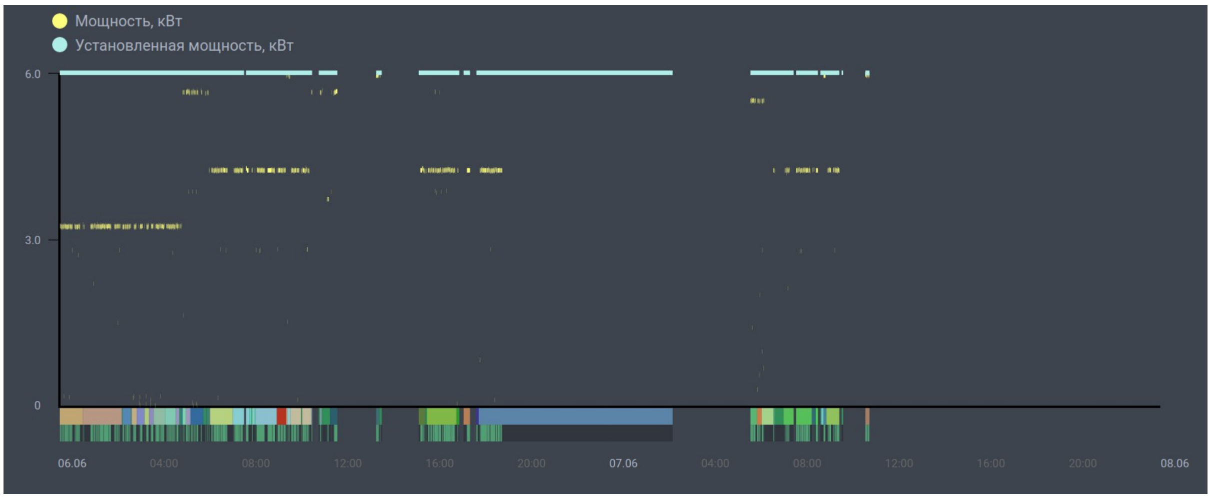


Figure 7 - Statistics on laser power used

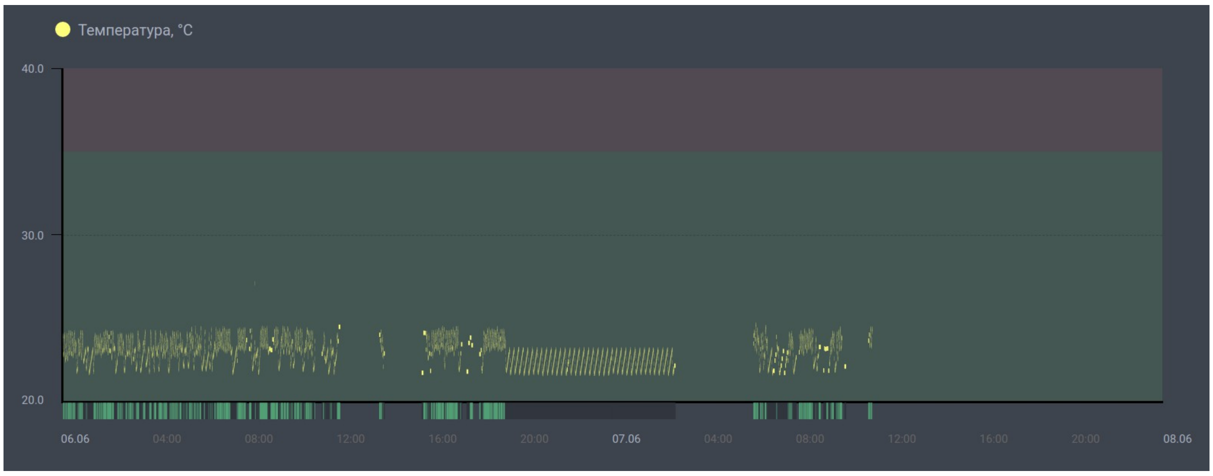


Figure 8 - Source temperature statistics

The "Laser Status" section contains three indicators of the laser status during use: the ratio of the actual laser power to the set power, the actual and set power (Figure 7), and the laser source temperature (Figure 8).

In addition, there are other sections: "Program Statistics" - statistics on used programs; "Feed Correction" - deviation by programs and operators; "Cutting Speed" contains data on cutting speeds

cutting, cutting speed correction, and these statistics regarding programs and operators. It is also possible to read the main technical data of the machine and download a report in format
.xls, which will contain all statistics on all indicators for the designated period.

1.1.2 Fault diagnosis problem

Efficient operation of industrial devices requires reliable service systems that must provide risk-free, fault-tolerant solutions, so preventing errors and failures is of paramount importance.

Typically, to prevent failures and errors, operators tend to perform maintenance based on manufacturers' recommendations and have to check critical parts on a regular basis, This requires experienced professionals with a high level of skill. This process is very expensive and labor intensive.

This problem is also present on Navigator model machines. Software and operator errors occur regularly, as well as critical failures such as optical system (laser) failure. Equipment wear and tear can also occur, such as XYZ axis drive failure.

At this point, operators need to constantly monitor the machine and perform complex technical analysis, which may require additional specialists. As mentioned above, this approach can be time-consuming and costly. This approach is unique to Navigator machines.

1.1.3 Automatic fault detection as a solution to fault diagnosis problems

The purpose of this thesis is to create a system for the detection and prediction of malfunctions on laser cutting machines Na vigator. The solution to this problem is the development and implementation of this system, as well as the description of the theoretical justification.

The solution of the task will bring positive results both for the company VNITEP and for its clients, including the enterprise Cespel, namely such a solution can reduce the time of operators and engineers, thus saving money and other resources of the enterprise. The developed solution will increase the efficiency of industrial processes of Sespel enterprise.

The main means for solving this task will be the statistics collected for the period 20172019 on various parameters of different modules: CNC, laser, components.

The main metrics will be:

– The ratio of the set laser power to the actual laser power, which allows you to track the wear trend of the laser head.
– Laser temperature is a parameter that can also be an indication of laser head wear and tear, and can also be a warning parameter of a possible deviation from the normal required temperature, which can be classified as operator error or machine error in general.
– Vibration sensor data - allows you to track the wear trend of the machine platform.
– Data from the XYZ drive sensors - allows you to identify errors and failures of the spatial laser drive.

1.1.4 Description of the approach to solving the problem of automatic diagnosis

stiches

Recently, the paradigm of the Industrial Internet of Things, which is part of the more general concept of the Internet of Things, has become widespread and provides ideas for integrating industrial equipment and sensors into a single system. Such a system can allow automated monitoring and analysis of critical parameters of industrial devices, without the involvement of operators, to detect and predict errors and failures [25].

Although the Industrial Internet of Things paradigm provides ideas for creating a system, it does not describe the ways and methods of realizing this system. This is due to the relatively recent emergence of this paradigm, which means that the methodology and principles of development are not yet established.

Since there are no established concepts at the moment, the development is based on different approaches, taking into account the specifics of industrial devices for which the system is being developed.

The problem of industrial IoT system development can be divided into two subproblems, each of which can be solved separately, but the solution of the second subproblem may depend on the solution of the first one:

1. Development of a subsystem for collecting, processing and storing data from various devices.
2. Development of a data analysis subsystem to detect and predict errors and failures.

Solving the first sub-problem can be the main solution to the second sub-problem, since data analysis requires a ready environment in which to analyze the data.

If the first subproblem can be solved by designing a graphical architecture and selecting appropriate tools, then to solve the second problem it is necessary to determine the approaches and methods for obtaining error and failure predictions, taking into account the peculiarities of the data.

It can be said that in order to have a system for detecting and predicting device errors and failures, two sub-problems must be solved. The first subproblem is not the subject of this thesis and is beyond the scope of this thesis, however, this thesis will describe the existing developed system with regard to its advantages and disadvantages, in the context of which the second subproblem will be solved.

1.1.5 Time series

The problem of predicting and detecting machine errors and failures can be characterized as a time series analysis problem, since the underlying data is a sequence of time-delimited events.

Time series is a series of data points indexed in time order. Time series analysis is a set of methods for analyzing time series data that are aimed at identifying significant patterns [7].

Time series have a temporal structure, that is, they are ordered in time, thus time series analysis differs from cross-sectional studies, which usually use the analysis of a sample or samples from the general population at a particular point in time. In addition, time series analysis differs from spatial analysis, which has the same fixed basis as cross-sectional studies, for example, spatial analysis may analyze geographical objects without considering the time component [14] .

There is a classification of time series [11] :

– univariate and multivariate.

– stationary and non-stationary.

– linear and nonlinear.

Let there be a process x_t where $t \geq 0$ or $-\infty < t < \infty$.

Univariate is a time series that is generated based on a single attribute, while multivariate has more than one, in other words in the univariate case x_t is a vector and in the multivariate case x_t is defined as a matrix.

Time series data can be represented as an $n \times d$ data matrix [8]:

$$\mathbf{D} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix} \tag{1}$$

Vector strings of the form:

$$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d}) \tag{2}$$

depending on the subject area have different names, e.g. entities, objects, examples, etc.

Vector columns of the form:

$$\mathbf{X}_j = (x_{1,j}, \ldots, x_{n,j}) \tag{3}$$

can also have different names: attributes, properties, variables, etc.

Stationarity is defined through the requirement that the co-local distributions of the vector (x_{t_1}, ..., $x_{(t)(k)}$) and the vector ($x_{(t)(1)+}$ ($_{t)}$, ..., $x_{(t)(k)+(t)}$) are equivalent. Dru

In other words, a stationary time series has repeating patterns, unlike a non-stationary one [6] .

Linear models are defined through the linearity properties of time series parameters, such as mean and variance. Nonlinear models, on the contrary, may have a nonlinear structure, which requires the use of nonlinear methods, such as nonlinear regression methods [5].

The most understudied are nonlinear nonstationary models [15]. Examples of realization of error detection and prediction system based on this type of models were not found and will not be discussed in this work, as this area is beyond the scope of the thesis topic.

The subtask outlined above can also be broken down into two elements:

1. error detection, anomaly detection.
2. Predicting the behavior of the device.

These problems can be solved both separately, where a different method will be used for each problem, and jointly through a single method. Each option will be considered in the existing examples.

1.1.6 Intelligent analysis of time series data

It can also be said that the task of detecting and predicting errors and failures is a task of data mining, namely, intelligent analysis of time series data. This abstraction is justified because there are similar problems that are solved using this paradigm.

Data mining is the process of identifying patterns from large amounts of data [19] .

Data mining is an interdisciplinary field that utilizes methods, ideas and principles from the following fields

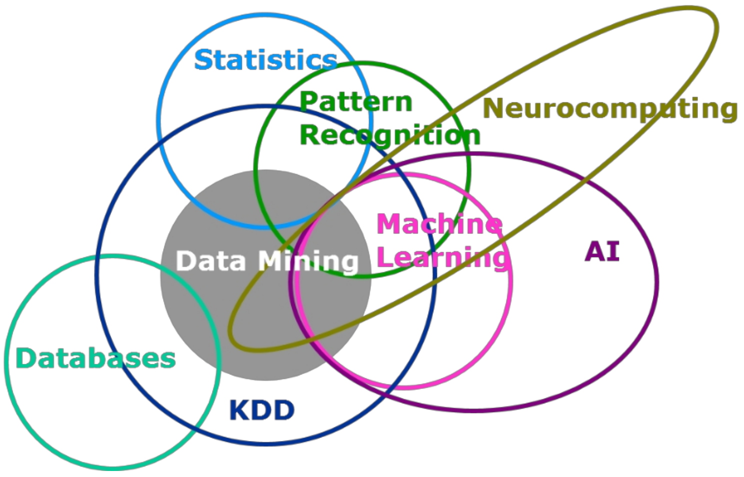


Figure 9 - The place of data mining among other fields

The following areas (Figure 9): statistics, machine learning, pattern recognition, computational neurobiology, and databases. It is worth noting that this area is part of a broader area knowledge database discovery.

In addition, data mining can be defined as a part of pattern recognition, which is concerned processing data from databases and identifying patterns related to a particular subject area [16] . The field of pattern recognition itself covers other fields such as signal processing and machine vision [17] .

Let us describe these areas and outline their main ideas and principles.

A database is an organized collection of data that is collected and stored using computer systems. A database is an organized collection of data that is collected and stored by computer systems.

Database theory supplies formalized methods and principles for the design and development of databases and DBMSs. Usually theoretically

There are two main types of database construction: relational (SQL) and non-relational databases (NoSQL). There is also sometimes a mixed type, which appeared relatively recently: new relational databases, which combines the best ideas and principles from relational and non-relational theories and approaches (NewSQL) [3].

In the context of data mining, databases are used as a source of data. The data themselves are processed before analysis, which is also part of the knowledge discovery process. The type and characteristics of data collection and storage in the database determine the subsequent steps of data analysis.

Knowledge database discovery (KDD) is a broader area of working with database data, of which data mining is a part. KDD is defined as a process of knowledge discovery (patterns), which is divided into main steps [44] :

1. selection - selection of data from databases according to certain criteria.
2. preprocessing - bringing the selected data into a suitable form (cleaning and deleting missing values, fixing attributes) for further analysis.
3. transformation - transformation of data into a suitable data structure, e.g. a data matrix (1).
4. Data mining - detecting patterns in data.
5. interpretation and evaluation (interpretation and evaluation).

The next step can also be to utilize, deploy, and exploit the discovered knowledge and patterns for the purposes of the subject area.

Statistics, pattern recognition, and machine learning are the method providers for each step of the KDD process. Each of these areas has its own characteristics, but they all also have much in common.

1.2 Analysis of existing solutions to the problem of detecting and predicting faults in devices

So, let's look at existing solutions for detecting anomalies, such as data errors and failures, as well as approaches for predicting and alerting users about them.

First, it can be said that time series forecasting models such as autoregressive model (AR), moving average model (MA), exponential model, moving average autoregressive model (ARMA), integrated moving average autoregressive model (ARIMA) depend on parameters derived from historical time series. These parameters are non-negative integers that refer to the order of the autoregressive part, the degree of the participating first difference, and the order of the moving average part. Although these processes can handle non-stationary data, they are limited in memorizing any state for any time interval [13].

In [26], the authors considered stream data as a partially observed Markov process [27]. Such a decision was made in order to create a composite system that consisted of different models: exponential models, seasonal models, and a model of the Cox log Gaussian process. This paper contains good ideas for the use of filters on a composite of partially observed Markov processes, as well as a theoretical justification for such use. However, a good developed theory and weak testing on real data did not show high efficiency of this approach. In this paper, the authors have learned only predict anomalies, but have not been able to identify them, and use this approach in real time.

Federated learning is one of the machine learning techniques that consists of training an algorithm on multiple decentralized peripherals or servers containing local data samples without sharing their data samples. The use of federated learning for filtering medical device network data was discussed in [28]. By using federated learning methods, the authors managed to improve the energy efficiency of data processing and analysis, i.e., they solved one of the most common problems in the field of the Internet of Things. The paper also describes the stages of local data analysis on the devices themselves and global data analysis on the server. For the first stage, the authors used adaptive filters [29], and for the second stage, matrix perturbation analysis [30]. The disadvantage of this approach is low flexibility, as it may be necessary to use correlation analysis of devices, and in distributed analysis correlation may be lost, thereby losing useful features. In addition, there are improved adaptive filters such as the kernel recursive filter [31]. It can also be said that matrix perturbation analysis may not take into account the asynchronous structure of preprocessed data.

For profile monitoring and fault detection purposes, in [32], the authors applied a nonlinear parametric regression model to develop a system that would be stable and insensitive to temperature changes in production practice. In [33], a monitoring method was developed that can automatically adapt the parameters of the control map (a method from control theory). In [34], the authors added all channel profiles and applied principal component analysis (PCA) to aggregated tonnage profiles to extract features. In [35], both statistical and wavelet features were used and extracted the

Each of these studies focused on analyzing an individual data profile, i.e., the analysis was performed with one-dimensional data. Each of these studies focused on analyzing an individual data profile, i.e., the analysis was performed with one-dimensional data. However, the sensor outputs of the device are typically multichannel, which requires multivariate time series analysis.

Multivariate data may have the following features [10]:

– As the number of sensors increases, the processing of each time series from each sensor becomes as more intensive, which has effect of increasing the computational load.
– The data has more outliers and noise compared to univariate data.
– Time series from different sensors have correlations at different levels, resulting in a large amount of redundant information, but possible correlations must be taken into account.
– Pre-downtime states typically last for some time and include equipment failure states. In addition, after a faulty sensor is repaired, there is also a delay period in the operational state. Therefore, the prediction model should have memory invariance with respect to such states.

These features of the data lead to the following research questions
I don't know what you're talking about:

– How to reduce redundant information without degrading prediction accuracy?
– How do you identify emissions and noise?
– How do we model long state memory?
– How do you accurately predict the current state based on historical data?

We will describe existing results that answer these questions. The authors in [36] applied multifactor PCA to reduce the

The author of the paper [37] extracted a set of several monitoring features from the high dimensional profile data in order to detect the nosiness of the instrument. The author of [37] extracted a set of multiple monitoring features from large size profile data to detect the tool nose. In another paper [38], the author applied uncorrelated multilinear principal component analysis (UMPCA) [39] for profile monitoring and fault diagnosis, which considered the channel interconnectivity of different profiles. Compared with PCA-based UMPCA, linear discriminant analysis (LDA) is also a classical method for feature extraction in data. However, conventional LDA cannot deal with multivariate data directly because it is a preassigned method for univariate problems.

In [40], an uncorrelated multilinear discriminant analysis was proposed for face recognition and image processing. This analysis works directly on multivariate data and extracts uncorrelated discriminant features through shadow vector projection. Compared to the UMPCA algorithm, which is unsupervised, such a method is a supervised multilinear feature extractor that will consider class information in feature extraction and may be more suitable for face recognition. Although there are some preliminary studies on the application of UMLDA for face recognition and image processing, few studies have been published in the literature on the use of UMLDA technology to analyze multidimensional data for automatic process control in industrial systems. In [41], the authors proposed a solution to the problem of real-time fault detection based on combining multivariate time series data into UMLDA models, However, the authors were unable to fully achieve the high efficiencies that other methods provide.

1.3 Statement of tasks to be solved

As it was mentioned above, the main purpose of the thesis is to create a system for detecting and predicting malfunctions on laser cutting machines Navigator KS12V of VNITEP company, installed at Sespel enterprise.

The main hypothesis is to achieve the goal through methods of intelligent analysis of multivariate time series data.

Let's finalize the tasks that need to be :

1. preparation and processing of accumulated data from the Navigator machine.
2. Initial descriptive analysis of the data.
3. Model selection and rationale for predicting multivariate time series of machine tool data.
4. selecting the optimal training window.
5. Definition of an adaptive prediction window.
6. Selecting and justifying the choice of clustering model in the data.
7. analyzing the resulting clusters.
8. selection of a data-driven model for fault detection.
9. model implementation and model testing.
10. integration of models into Omnicube's system.

1.4 Conclusions

In this section, the problem of detecting errors and failures on laser cutting machines of the Navigator KS12V model, as well as in devices in general, was outlined. The peculiarities of this problem as well as its theoretical aspects were described. In addition, the existing solutions based on different methods, approaches and ideas were analyzed. For each case a criticism was given, namely the advantages and disadvantages of each solution were outlined.

At the end of the section, tasks have been outlined whose solutions can help to achieve a solution to the problem of error detection and prediction.

2 DESCRIPTION OF METHODS AND TOOLS FOR IMPLEMENTATION

2.1 System for the Industrial Internet of Things

As described in the previous section 1, industrial data mining provides ideas for creating an intelligent system to automate and improve the efficiency of industrial devices. One of the many companies that are active in this area is Omnicube, which, as mentioned above, has its universal system for many tasks not only for the thinking Internet of Things, but for the Internet of Things in general, i.e., the tasks solved by this system are not limited to the industrial sector. The realization of a solution for fault detection and prediction is addressed in the context of this system. Let us describe this system.

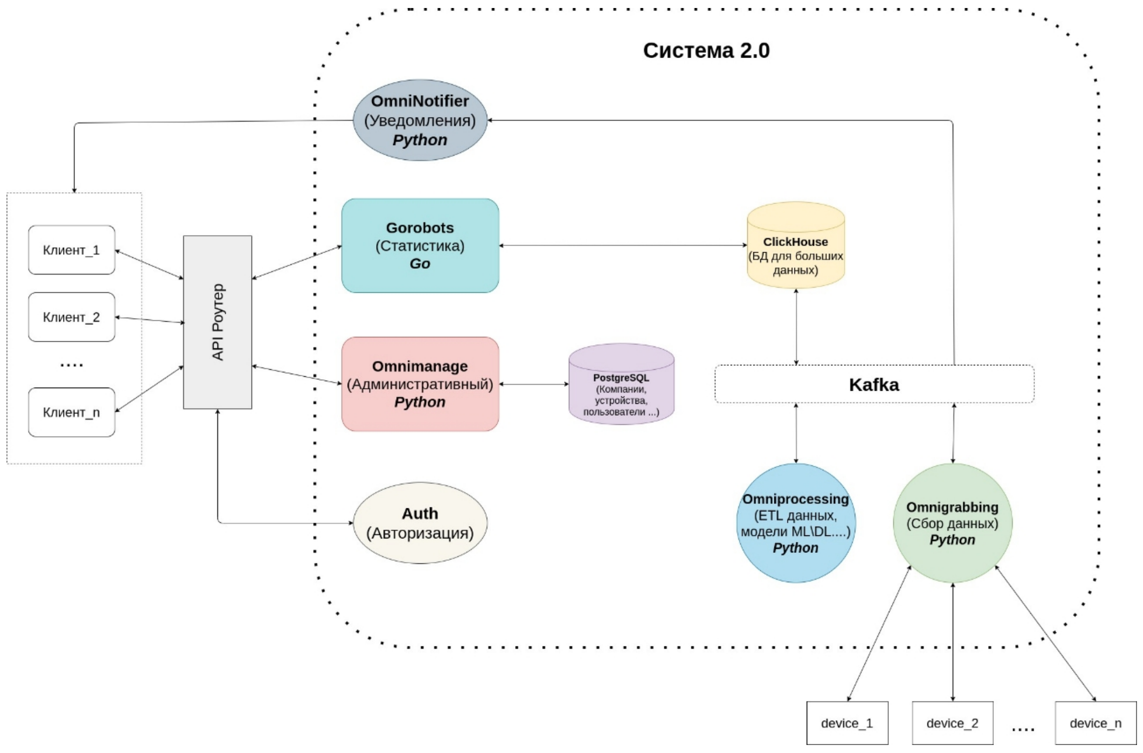

Система 2.0
OmniNotifier (Уведомления) Python
Gorobots (Статистика) Go
ClickHouse (БД для больших данных)
Omnimanage (Административный) Python
PostgreSQL (Компании, устройства, пользователи ...)
Kafka
Auth (Авторизация)
Omniprocessing (ETL данных, модели ML\DL....) Python
Omnigrabbing (Сбор данных) Python
API Роутер
Клиент_1
Клиент_2
....
Клиент_n
device_1
device_2
....
device_n

Figure 10 - Developed system for the Internet of Things

The system has a microservice architecture (Figure 10). Microservice architecture is the consideration of the developed application as a set of related services, each of which is responsible for some part of the overall system, as well as solves its own specific task. This approach has its advantages and disadvantages The main advantage is modularity, which allows you to develop and maintain components independently of each other. The disadvantages are: high complexity of development, complex process of deployment of the system on its own clusters, and, especially, on client clusters.

The main entities interacting with the system are clients and client devices. Each client device is connected to the system with the help of Omnicube engineers, taking into account the specifics of the devices themselves. For example, for the Navigator laser cutting machine, sensors were used, which were installed on critical nodes and from which the necessary information was collected. In addition, customer devices themselves sometimes provide functionality that allows data collection without installing sensors. Further, device data is aggregated and processed in a device-specific manner. The processed data is provided to customers in the form of a user interface, an example of which is the interface for the Cespel enterprise with data from the Navigator laser cutting machine.

The main nodes of the system are two microservices: omnimanage and gorobots. The omnimanage microservice is responsible for processing primary client data: information about the client, information about users of the client company, information about client devices (but not the device data itself), information about user roles for authorization, etc. Gorobots is responsible for processing the statistics collected from devices: it calculates basic statistical parameters (mean, variance, efficiency). Omnimanage is written in the Python language

using PostgreSQL database. Gorobots is written in Go language, data in this microservice is taken from ClickHouse database.

The omnigrabbing microservice is responsible for the collection and initial processing of client device data. this microservice converts the data into the required format for sending it to Kafka, and then uploads it to ClickHouse.

Kafka is a platform for real-time streaming data processing. The project aims to create a unified high-performance, low-latency platform for real-time streaming data processing. Kafka uses a binary protocol based on TCP, as well as systems for processing asynchronous data. This broker was chosen because of the large amount of incoming data. Kafka allows flexible solutions for processing large amounts of asynchronous data [46].

ClickHouse is a fast, open source OLAP database management system. This system focuses on columnar structure and also allows generating analytical reports using SQL queries in real time. The database was chosen because of the high load of storing a large amount of data. This DBMS allows processing several thousand queries per second, which is a key feature that distinguishes this system from other competitors [47].

Omnicube's system also includes auxiliary microservices:

– API router - a system for redirecting client requests.
– Auth is a service for user authorization.
– OmniNotifier is a system for customer notification.

The omniprocessing microservice (Figure 11) is the place where various models for monitoring and processing data are located, e.g.

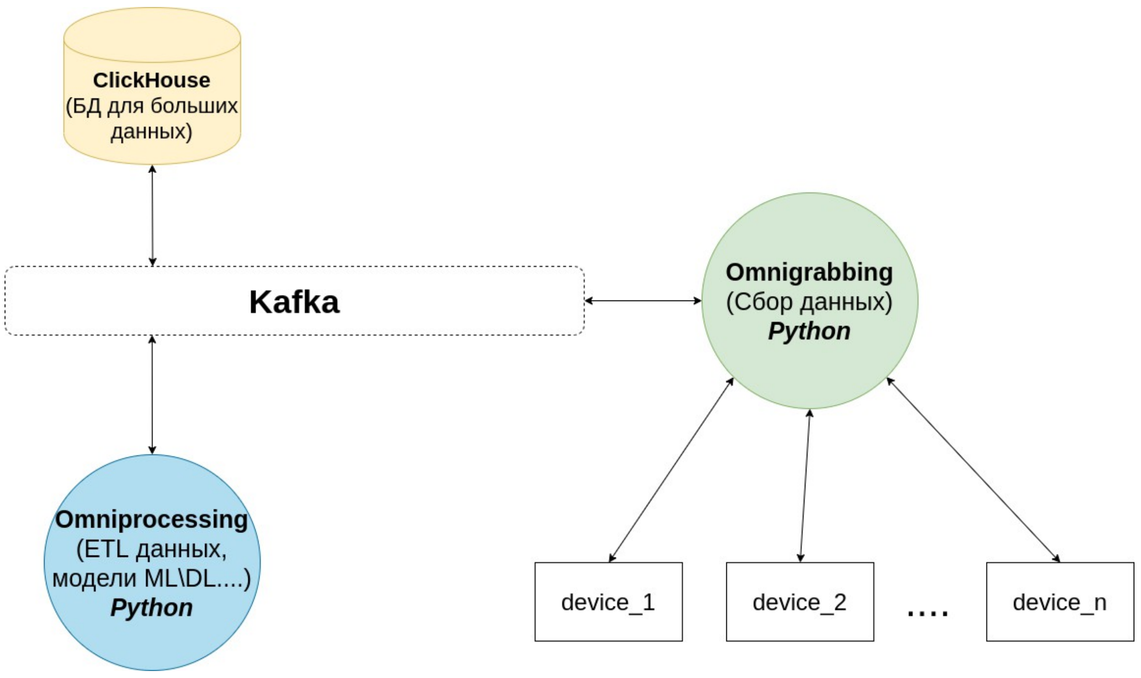


Figure 11 - Part of the system responsible for data analysis and processing

The main goals of this microservice are extraction, transformation and uploading. The main goals of this microservice are extraction, transformation and bootstrapping.

omniprocessing uses the Python framefork Faust, which allows data to be sent to the Kafka message broker queue. From Kafka, the data goes to the ClickHouse database. Thanks to this framework, it is possible to quickly build and embed machine learning pipelines [48]. The developed solution for Navigator machines is located exactly in the omniprocessing microservice.

2.2 LSTM neural network for time series forecasting

Recently, the use of neural networks has helped to improve the results of solving a large number of tasks in various spheres of human activity. This is due to the increased computing power of modern computers. It is this increased power that has allowed the theoretical research of many generations of scientists dealing with neural networks to be used to the maximum extent possible.

The field of time series forecasting has not been an exception to improve results based on the use of neural networks. The biggest breakthrough has been achieved by LSTM or longshort term memory networks, which is a kind of recurrent neural network architecture.

Recurrent neural networks, unlike unidirectional multilayer perceptrons, are capable of storing in memory various features that are necessary for the task at hand. In the case of time series, these can be long-term dependencies that can influence prognosis.

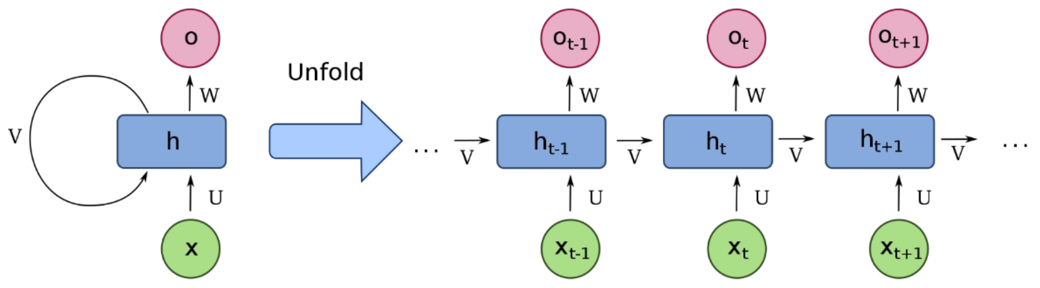


Figure 12 - Architecture of recurrent neural network

Figure 12 shows the architecture of a recurrent neural network, which is a directed sequence of neural network nodes, each of which depends on and influences other nodes based on a redistribution of weights.

Let there be a matrix $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ of input data to a neural network h, which changes its state from h_1 to h_n, given the sequential processing of the data X with different time steps t, and based on the computations of the previous steps:

$$h_t = f(U\,\mathbf{x}_{(t)} + V\,\mathbf{x}_{(t)-1}) \tag{4}$$

Function 4 is an activation function, such as the sigmo idal function σ or the hyperbolic tanh tangent, or a set of activation functions that depends on the particular network architecture. At each step t, the neural network produces a result o_t. Matrices U and V contain input and recurrent weights, respectively.

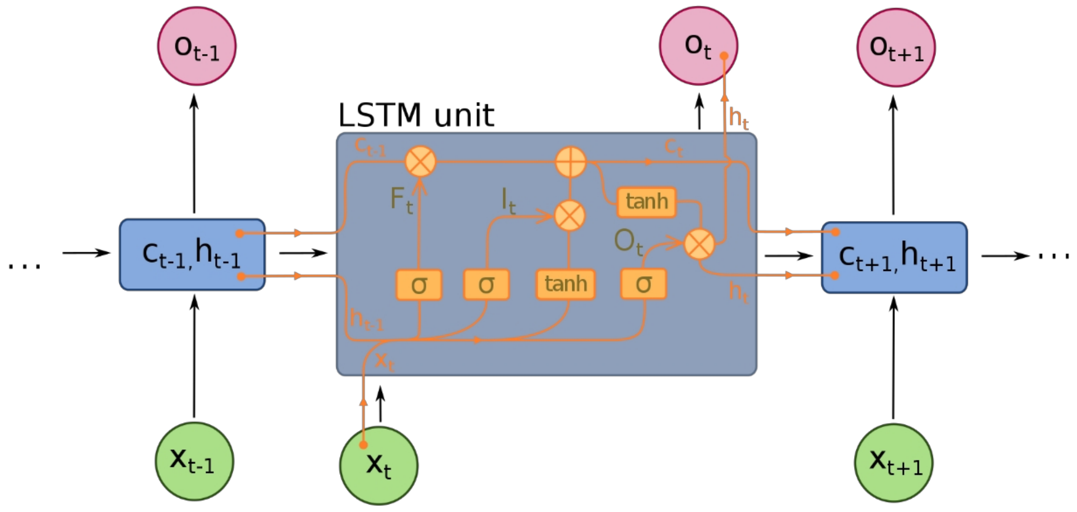


Figure 13 - LSTM architecture

Recurrent neural networks have different architectures, each of which has its own features, advantages and disadvantages. The LSTM architecture is the most effective for solving a wide range of problems (Figure 13).

$$
\begin{aligned}
&F_t = \sigma(W_{(F)}\, x_t + U_{(F)}\, h_{(t)-1} + \\
&b_F\,)\ I_t = \sigma(W_I x_t + U_I h_{(t)-} \\
&{}_1 + b_{(I)}) \\
&\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \qquad (5) \\
&O_t = \sigma(W_O x_t + U_O h_{(t)-1} + b_{(O)}) \\
&c_t = F_{(t)} \circ c_{(t)-1} + I_{(t)} \circ \tilde{c}_{(t)} h_{(t)} = o_t \circ \tanh(c_{(t)})
\end{aligned}
$$

The system of equations 5 describes the sequence of calculations of the new hidden state of the neural network h_t.

– F_t is the activation vector of the forgetting node.

– I_t is the activation vector of the input updatable node.

– $\tilde{c}_t$ is the activation vector of the incoming cell.

– O_t is the activation vector of the output node.

– c_t is the state of the cell at time t.

– W and U are weight matrices.

– b is the displacement vector.

2.3 Optimal learning window

Typically, the accuracy of a prediction model increases in proportion to the amount of training data, but this approach is not always appropriate. If the behavior or statistics of the incoming data changes, the trained model may fail to track these changes, leading to an accumulation of errors. This problem is called concept drift. To solve this problem, a moving window for re-training is used, which is generated from the latest data based on a given window size.

Choosing the optimal training window size of machine learning models is an open question. A large window size may have more accurate results, but it increases the complexity of the model, making it unsuitable for real-time applications, while a small window size may lead to an increase in error and, consequently, to a decrease in system reliability, overtraining or undertraining.

The authors of [42] proposed a method to determine the optimal size of the training window. The basic idea is to use least squares spectral analysis, also known as the LombaScargle method. This method works better on missing data, e.g., a similar method such as the fast transform

Fourier. In addition, fast Fourier transform requires uniform data distribution, which is not always possible when analyzing data from device sensors.

$$P_X(f) = \frac{1}{2\sigma^2}\left(\frac{\left(\sum_{n=1}^{N} (x(t_n) - \bar{x}) \cos(2\pi f (t_n - \tau))\right)^2}{\sum_{n=1}^{N} \cos^2(2\pi f (t_n - \tau))} + \frac{\left(\sum_{n=1}^{N} (x(t_n) - x) \sin(2\pi f (t_n - \tau))\right)^2}{\sum_{n=1}^{N} \sin^2(2\pi f (t_n - \tau))} \right) \qquad (6)$$

Function 6 is a description of the LombaScargle method. The optimal size of the training window is determined based on finding the largest periodic component occurring in the data. In other words, the size of the seasonality of the data is determined. However, before using this method, the data should be tested for seasonality to ensure that the method is correct.

$$\tan(4\pi f\tau) = \frac{\sum_{n=1}^{N} \sin(4\pi f t(_n))}{\sum_{n=1}^{N} \cos(4\pi f t(_n))} \qquad (7)$$

The parameter τ is defined in Equation 7.

2.4 Algorithm kShape

To detect anomalies or faults in machine data, it is necessary to use the basic principles of cluster analysis, which is the partitioning of data into clusters to identify laws of numbering.

Usually, the clustering problem is formulated as follows. Let us denote by $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ a set of n data, where $\mathbf{x}_i \in \mathbb{R}^m$. In clustering, the data X should be partitioned into k non-overlapping clusters $P = \{p_1, ..., p_k\}$ such that the sum of squares of the distances between the data instance $x_{(i)}$ and the centroid $\mathbf{c}_j$, which represents the concentration point of cluster $p_{(j)}$, is minimal:

$$P^* = \arg\min \sum_{j}^{k} \sum_{\mathbf{x}_i \in p(j)} dist(\mathbf{x}_i, \mathbf{c}_j) \qquad (8)$$

In Euclidean space such an optimization problem is NP-complete. The k-means method allows us to find a local optimum mediated by a random distribution of *k* centroids among the data.

There are different clustering algorithms in cluster analysis, each of which has its own specificity and may depend on the structure of the analyzed data.

In the last few decades, time series sequence clustering has received considerable attention, not only as a powerful research method in its own right, but also as one of many steps in data exploration and processing.

Most methods of time series analysis, including clustering, depend on the choice of a distance measure. A key issue in two time series sequences is how to handle the various distortions that characterize the time series.

Due to various difficulties and the need for invariance with respect to subject areas, much attention has been paid to the creation of new distance measures rather than to the creation of new clustering algorithms. It is generally believed that the choice of the distance measure is more important than the clustering algorithm itself. As a consequence, time-series clustering is mainly based on classical clustering methods: either by replacing the default distance measure with a more appropriate one for the time series, or by transforming the time series into "flat" data so that existing clustering algorithms can be used directly.

However, the choice of clustering method can affect accuracy, as each method expresses homogeneity and cluster separation differently, and efficiency, as the computational cost differs from one method to another. For example, spectral clustering or some variants of hierarchical clustering are more suitable for identifying clusters based on distribution density (based on areas of higher density) than separation methods such as the k-means method. On the other hand, the median method is more efficient than hierarchical or spectral methods in general.

Current approaches to shape-based clustering, which use partitioning methods with scale- and shift-independent distance measures, have two major drawbacks: (i) these approaches cannot scale to large data sets because they depend on computationally expensive methods or distance measures; (ii) these approaches were developed for specific subject areas or their effectiveness has only been shown for a limited number of data sets. Moreover, the most successful shape-based clustering methods handle phase invariance through local non-linear alignment of sequence coordinates, even though global alignment is often adequate.

The described approaches have never been comprehensively evaluated against each other, against other separation methods, or against different approaches such as hierarchical or spectral methods. This comparative analysis is only partially present in the various articles.

In [43], the authors proposed a new kShape algorithm for data shape-based time series clustering, which is efficient and independent of the specific subject domain. The kShape is based on a massive iterative refinement procedure similar to the one used by the k-means algorithm, but with significant differences. B

In particular, kShape uses both a different distance measure and a different method of calculating centroids than the k-means method. The kShape algorithm attempts to preserve the shapes of time series sequences when comparing them. To do this, kShape requires a distance measure that is invariant to mas stacking and shifting. Unlike other clustering approaches, kShape adapts a statistical measure of crosscorrelation.

Crosscorrelation is a measure of similarity between time-delayed signals that is widely used in signal and image processing. This statistical measure allows to determine the similarity of two sequences **x**= $(x_1, ..., x_m)$ and **y**= $(y_1, ..., x_m)$ even if they are not aligned with each other (and even if they have different lengths). Invariance with respect to shifts is achieved by fixing one sequence, e.g., **y**, and successively sliding **x** through **y**, where the scalar product of these sequences is computed.

In [43], the authors of the kShape algorithm described the kShape algorithm into three parts: kshape, extractshape and sdb. Let us describe each part.

The algorithm is first given a set of *znormalized* time series as well as k clusters, the number of which is determined in advance, which is a separate task.

$$z_i = \frac{x_i - \mu}{\sigma} \tag{9}$$

Equation 9 describes znormalization, which is the ratio of the deviation of a random element x_i from the mathematical expectation μ to the standard deviation σ.

```
def k_shape(x, k):
    iter = 0, idx = []
    centroids = [0,...,0]
    for i in range(100):
        old_idx = idx
        for j in range(k):
            centroids[j] = extract_shape(
                idx, x, j, centroids[j])
        distances = (1 - sbd(centroids[j], x))
        if old_idx == idx:
            break
```

Figure 14 - Algorithm kShape

Figure 14 shows the pseudocode of the kShape algorithm. At the first step, the necessary variables are initialized: the iteration counter (iter), the list for filling values based on clusters (idx), the list of zero-filled centroids (centroids), the length of which depends on the number of *k* clusters. Next, iterations take place that define the centroids and the clusters themselves, respectively. The authors calculated that 100 iterations will be enough to find a local optimum. The function extract_shape is intended to calculate a new centroid. The sbd function is designed to calculate the so-called shapedistance of normalization of crosscorrelation coefficients.

```
def extract_shape(idx, x, j, cur_center):
    a = []
    for i in range(len(idx)):
        x[i] = sbd(cur_center, x[i])
    S = x[i]*x[i]^T
    Q = I - 1/m * O
    M = Q^T * S * Q
    C' = Eig(M, 1)
    return C`
```

Figure 15 - Algorithm for calculating a new centroid based on the SBD distance

Figure 15 shows the pseudocode for computing the new centroid by maximizing the squares of the SBD distances (Equations 10 and 11). The matrices *I* and *O* represent a unit matrix and a unit-filled matrix. In Equation 10, *the* function *CC*(**x**, **y**) denotes crosscorrelation, R_0 is the automatic crosscorrelation function.

$$SBD(\mathbf{x},\mathbf{y}) = 1 - \max \frac{(CC)(\mathbf{x},\mathbf{y})}{\sqrt{R_{(0)}(\mathbf{x},\mathbf{x}) \cdot R_{(0)}(\mathbf{y},\mathbf{y})}} \qquad (10)$$

$$\mathbf{y}_k^* = \arg\max \sum_{\mathbf{x}_i \in P(k)} \mathrm{SBD}(\mathbf{x}_i, \mathbf{y}_k)^2 \qquad (11)$$

Based on the properties of crosscorrelation, the authors of the algorithm rewrote the original Equation 11 in the form of maximizing the Rayleigh ratio:

$$\mathbf{y}_k^* = \arg\max \frac{\mathbf{y}_t^T \cdot M \cdot \mathbf{y}_t}{\mathbf{y}_t^T \cdot \mathbf{y}_t} \qquad (12)$$

In Equation 12, the matrix *M* is Hermite and has a Schur decomposition in the form $M = Q^T \cdot S \cdot Q$. The computation of the matrices *Q* and *S* is shown in the listing

2. The authors found the maximum value of $\mathbf{y}_k^*$ as an eigenvector which corresponds to the maximum eigenvalue of the symmetric matrix M .

```
def sbd(x, y):
    length = 2^2(2*length(x)-1)
    CC = IFFT{FFT(x, length) * FFT(y, length)}
    NCC c = CC/(||x|| ||y|| )
    [value, index] = max(NCC)
    dist = 1 - value
    shift = index - length(x)
    if shift  0 then
        y_0 = [zeros(1 , shift), y(1 : end - shift)]
    else
        y_0 = [y(1 - shift : end), zeros(1 , -shift)]
```

Figure 16 - Efficient calculation of SBD distance

Figure 16 shows the algorithm for computing the SBD distance using fast and inverse fast Fourier transform (FFT and IFFT). Also, this listing describes the subsequent manipulations with the cross correlation measure obtained after working with the fast Fourier transform.

2.5 Gradient Busting

To detect faults, a model trained on the basis of the given fault criteria and identified clusters of time series is required. Since the criteria and clusters represent the identified features at the stage of analysis of each time series separately, it is necessary to use a model capable of recognizing features in all time series simultaneously, i.e., in the context of this paper, to determine the faults based on data from different sensors in the form of multidimensional time series.

A bousting approach is best suited for such task. Busting is a special algorithm that is based on the composition of other machine learning models. There are various modifications bousting, each of which has its own peculiarities, advantages and disadvantages. However, recently, the best results have been shown by the gradient bousting algorithm.

Gradient bousting is a variant of bousting, which is based on optimization of a differentiable loss function using the gradient descent algorithm. Let us describe this algorithm more formally.

Let there be a data set $\{(\mathbf{x}_i, y_i)\}^n_{i=1}$. It is necessary to find the approximation simulation $\hat{F}$ of the function $F(x)$ that minimizes the expectation of the loss function $L(y, F(x))$ (Equation 13).

$$\hat{F} = \arg\min \mathrm{E}[L(y, F(x))] \quad (13)$$

Gradient bousting takes y and searches for an approximation of $\hat{F}$ based on weighted functions $h_i(x)$ from some class H, called weak learning models (Equation 14).

$$\hat{F} = \sum_{i=1}^{M} \omega_i h_i(x) + const \quad (14)$$

In the first step, the function F_0 is computed, which represents initial minimization of the loss function.

$$F_0(x) = \arg\min \sum_{i=1}^{n} L(y_{(i)}, \omega) \quad (15)$$

As a result, the recurrence relation of the gradient descent method for the loss functions (Equation 16) and weights (Equation 17) is given.

$$F_m(x) = F_{m-1}(x) - \omega_m \left(\sum \nabla_{F_{(m-)(1)}} L(y_i, F_{(m)-1}(x_{(i)}))\right) \quad (16)$$

$$\omega_m = \arg\min \sum L(y_i, F_{m-1}(x_i) - \omega \nabla_{F_{(m-)(1)}} L(y_{(i)}, F_{(m)-(1)}(x_{(i)})))$$

(17

) By optimizing the parameters of the set of weak models, the gradient

of the

The bousting is capable of producing highly accurate results.

2.6 Tools for model development

For people involved in data analysis and machine learning, there are many tools available to solve their various problems. The tools can be either a complete platform (Matlab) or a set of additional software packages (Python, R).

The development of the system of fault detection and prediction on laser cutting machines will be carried out using the Python programming language. This language was chosen for two reasons. Firstly, the microservice, in which the developed solution will be implemented, as already described, is developed in the Python language. Secondly, Python language has flexible and high-quality tools for data manipulation and analysis, as well as tools for using the latest achievements of machine learning.

The main tools for data manipulation are the Numpy and Pandas libraries. These are the ones used for development.

The Numpy library is a set of functions for operations on multidimensional arrays. Although Numpy is used for the Python language, this library is almost entirely written in C, so it has high performance, unlike the standard tools provided by the Python language.

Pandas is a library that allows you to quickly and flexibly create complex datasets, and provides efficient methods of manipulating data of various types: numeric, categorical, temporal, binary, etc.

For the needs of descriptive statistics, as well as for data visualization, the Matplotlib library is used. This package contains a set of functions that allows you to build two-dimensional and three-dimensional graphs of various complexity. The wide functionality of Matplotlib allows you to analyze data visually, as well as to display the processed results.

For the needs of statistics and time series analysis, there are many third-party libraries for the Python language. The most developed are: SciPy, Statsmodels, Tslearn. SciPy is a library for complex scientific calculations. Statsmodels contains a set of most used statistical models. Tslearn is a library for time series analysis.

Many machine learning problems can be solved using the Scikitlearn machine learning library. This library provides functionality for solving a variety of machine learning problems: regression, classification, clustering, dimensionality reduction, etc. In addition, this library has the ability to prepare data for training and testing of machine learning models. This library will be used for such tasks.

Scikitlearn library covers a large number of machine learning tasks, but it does not have suitable tools for such area as deep learning. Also very often due to the lack of optimization of some tools, Scikitlearn library may not be suitable for use in running systems, so it is mainly used for prototyping. Therefore, libraries have been created in which these problems are solved. Neural networks, including those used in LSTM work, can be used with the help of TensorFlow and Keras libraries.

TensorFlow is a library for machine learning and deep learning in particular, written in Python and C++, with the added capability of utilizing CUDA, NVidia's parallel computing technology for graphics cards. TensorFlow allows you to visually and quickly train complex machine learning models, as well as quickly integrate and deploy the models in production environments. The library can also be used not only in Python, but also in JavaScipt, C++ and Swift.

Keras is a framework for deep learning tasks that includes tools for working with complex deep learning models - neural networks of various architectures. Keras is based on TensorFlow, which is an advantage because the framework and the library are able to work in concert. The Keras framework will be used to train the LSTM neural network.

2.7 Conclusions

This section analyzed Omnicube's system into which the Navigator machine tool error prediction and detection solution will be integrated.

The recurrent neural network LSTM was chosen for forecasting tasks. The section describes the advantages over other neural network architectures for time series forecasting tasks. A detailed description of LSTM network operation was given. The principle of determining the optimal training window was also described.

In addition, we analyzed clustering algorithms and selected and described the kShape algorithm for clustering multivariate time series. It is this algorithm that will help to qualitatively determine features of the Navigator machine time series data, and determine faults based on these features.

The section ends with a brief description of the tools used for the development, namely the analysis of libraries and frameworks for data analysis and machine learning tasks designed for the Python language.

3 DEVELOPMENT AND IMPLEMENTATION

3.1 Pre-processing and initial data analysis

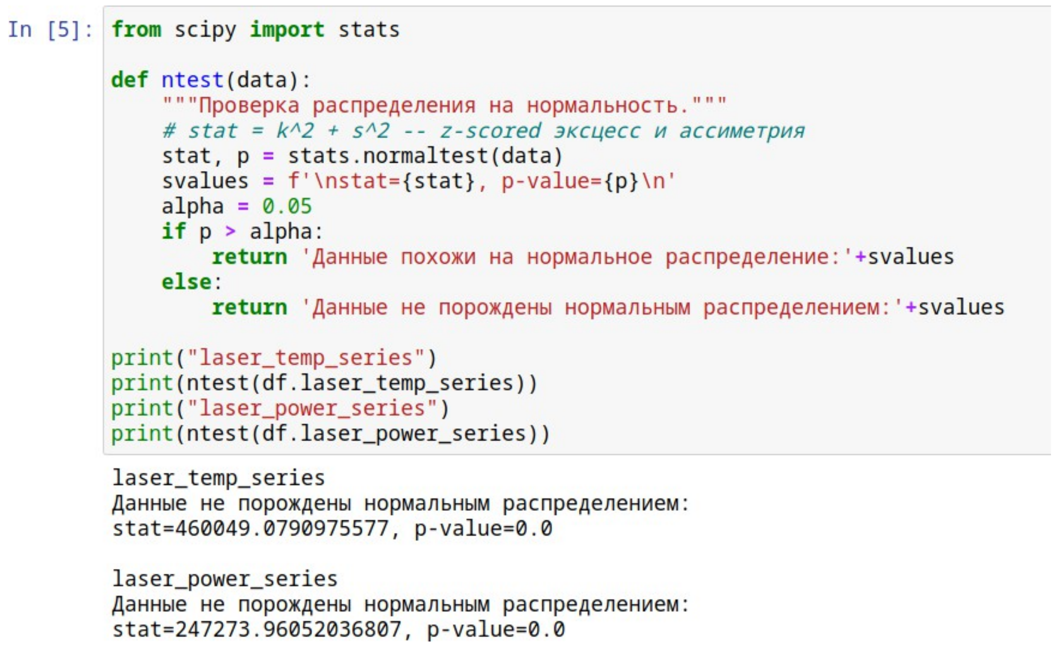
The data used was collected from the ClickHouse database. The period of collected data is from February 2 to June 11, 2020. The data is a set of about one million strings in JSON format, which have the following attributes:

- state - A categorical attribute representing the state of the machine.
- machine operator (operator) - full name of the machine operator performing work at the moment *t*.
- program name (program) - the name of the running program.
- laser mode (mode), e.g. automatic or manual.
- amperage.
- voltage.
- metal sheet counter (sheet counter).
- laser temperature.
- laser power.
- set laser power.
- time in Unix format.

The main parameters analyzed are: time, laser power and temperature, set laser power. At the time of data collection it was not possible to collect data on voltage and amperage, so the values of these parameters are zero. Also there was no possibility to take statistics on program errors, as all errors were recorded in connection with running programs. These problems will be solved in future developments.

3.1.1 Normality check

The first stage of the analysis was to check the distributions generated by the time series for closeness to normal. For this purpose, the K square d'Agostin test was used, which allows us to determine the measure of convergence of the initial distribution to the normal one. The hypothesis H_0 was that the given distribution is not normal. For this purpose, p-value estimation was used.



```
In [5]: from scipy import stats

        def ntest(data):
            """Проверка распределения на нормальность."""
            # stat = k^2 + s^2 -- z-scored эксцесс и ассиметрия
            stat, p = stats.normaltest(data)
            svalues = f'\nstat={stat}, p-value={p}\n'
            alpha = 0.05
            if p > alpha:
                return 'Данные похожи на нормальное распределение:'+svalues
            else:
                return 'Данные не порождены нормальным распределением:'+svalues

        print("laser_temp_series")
        print(ntest(df.laser_temp_series))
        print("laser_power_series")
        print(ntest(df.laser_power_series))
```

```
laser_temp_series
Данные не порождены нормальным распределением:
stat=460049.0790975577, p-value=0.0

laser_power_series
Данные не порождены нормальным распределением:
stat=247273.96052036807, p-value=0.0
```

Figure 17 - Testing of temperature and laser power distributions

Figure 17 shows that the temperature and power distributions do not have a normal structure.

The coefficients of kurtosis and skewness were calculated to better represent the structure of the data.

```
laser_temp_series
Эксцесс (острота пика распределения): 4.319019244019009
Ассиметрия (искаженность данных): -2.4806179934401724
```

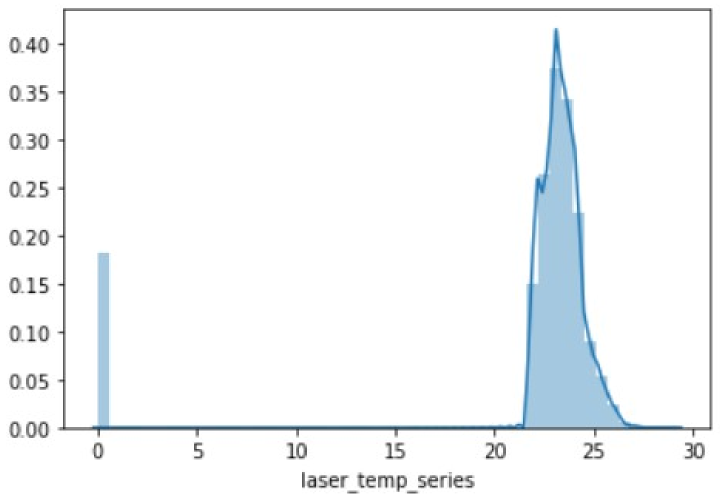


Figure 18 - Verification of the laser temperature distribution structure

```
laser_power_series
Эксцесс (острота пика распределения): 1.5045682121592465
Ассиметрия (искаженность данных): 1.5639472853029839
```

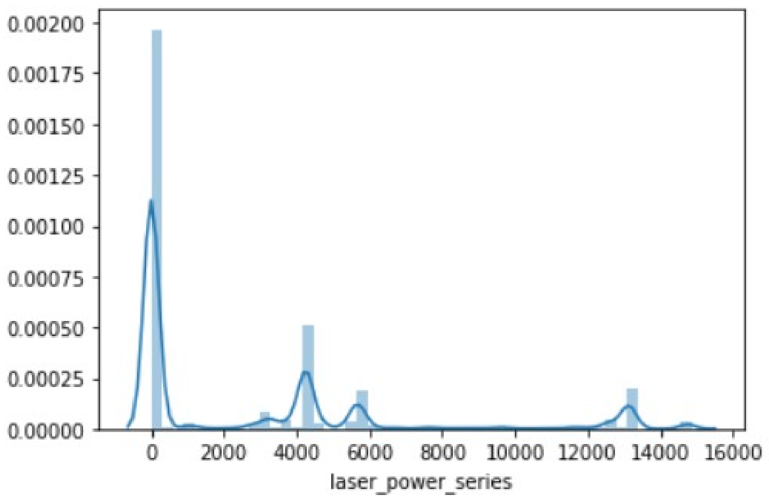


Figure 19 - Verification of the laser power distribution structure

Figures 18 and 19 show data on the structure of the temperature and laser power distributions. It can be seen that the structure of both distributions is complex, consisting of a mixture of distributions with nontrivial properties.

3.1.2 Trend analysis

Figure 20 shows that the data mostly have a stable trend by month. The dispersion of laser temperature data is low.

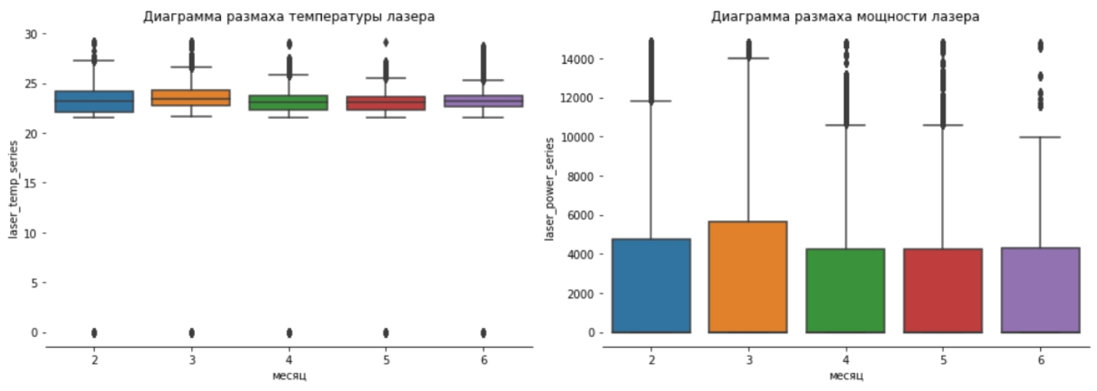


Figure 20 - Spread charts

Figures 21 and 22 show the trends by day, week and month. It is noticeable that the laser power trend is decreasing. This indicates that the laser head of the machine is starting to wear out, so this trend should be monitored.

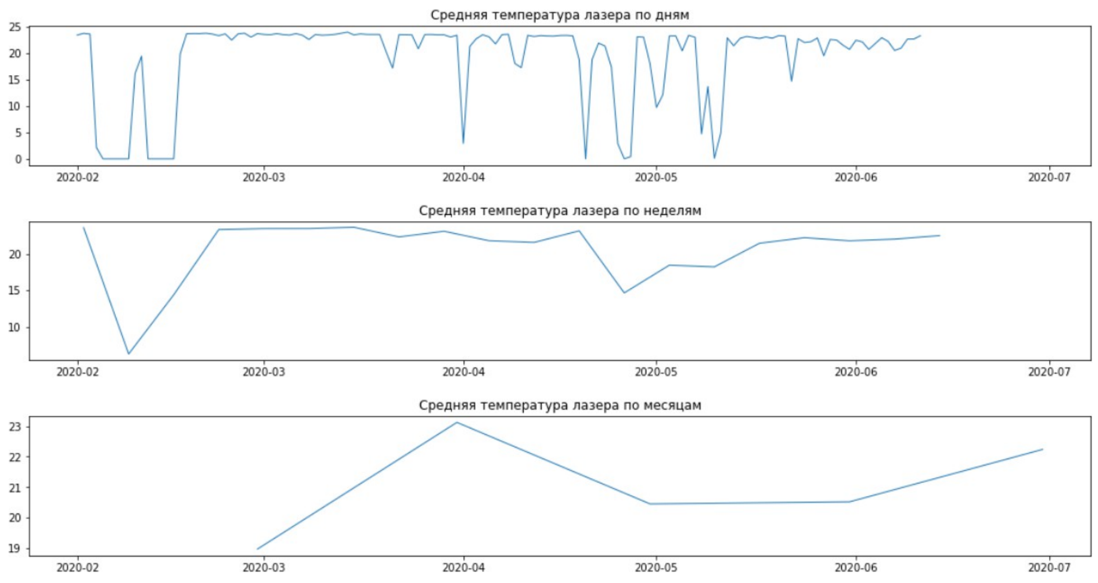


Figure 21 - Laser temperature trends

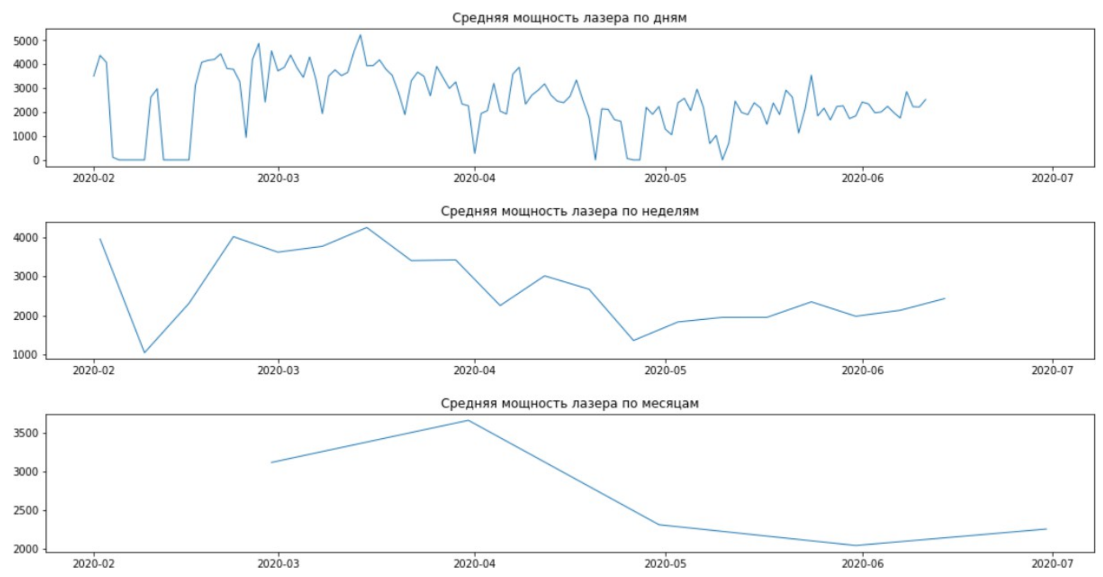


Figure 22 - Laser power trends

Figure 23 shows the time series over a short period of two parameters: actual and installed power. It can be seen that the installed capacity is constant, but it can be changed. The VNITEP engineers also recommend that regular exceeding of the actual power may lead to faster wear and tear, but as already described, the power trend is only decreasing so far. For a warning system, it is important to specify the requirements on the basis of which the laser power was allowed to be used above the set power for a certain time interval.

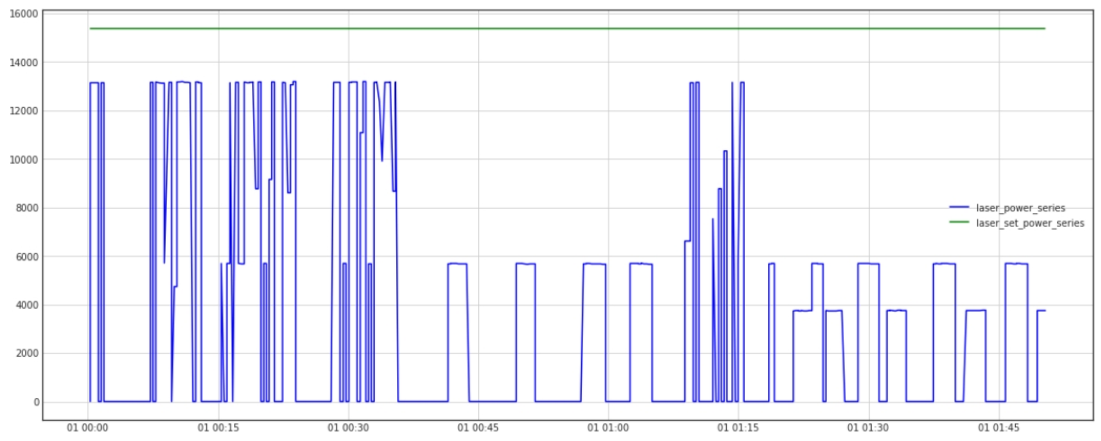


Figure 23 - Actual and installed capacity

Figure 24 is a representation of the apparent relationship between laser power and temperature. The data in the figure are normalized for easy comparison.

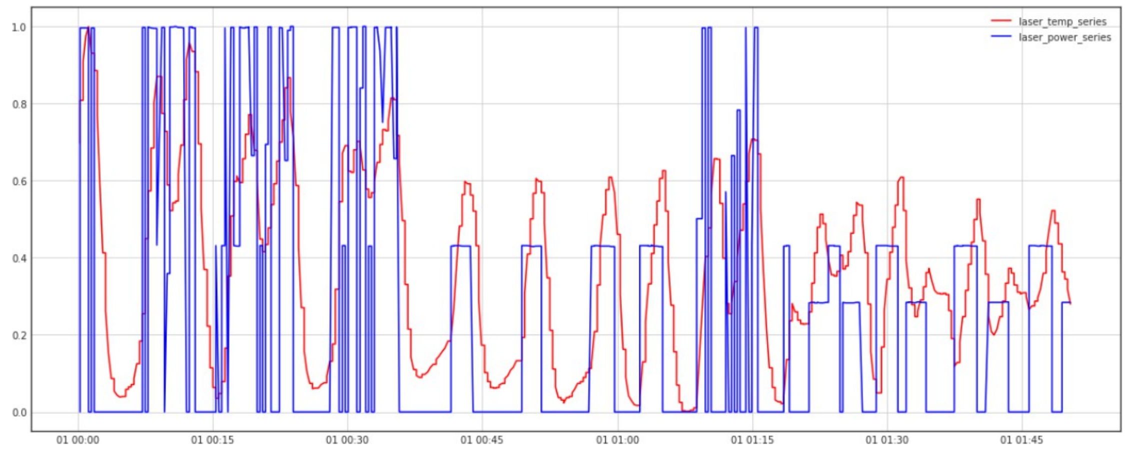
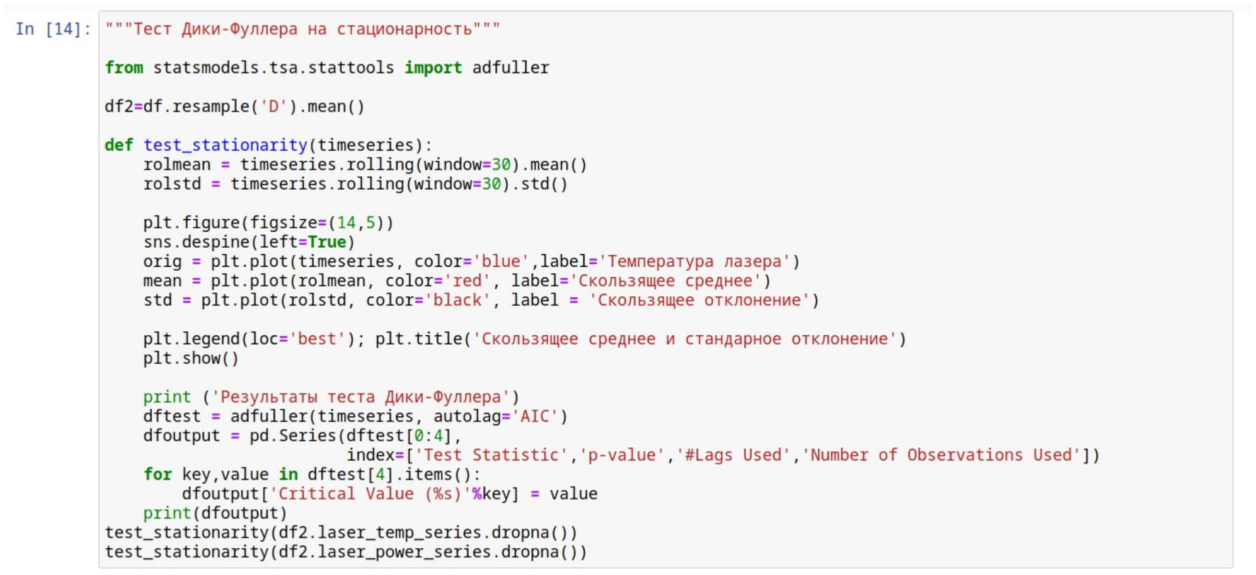


Figure 24 - Relation between temperature and laser power

3.1.3 Stationarity testing

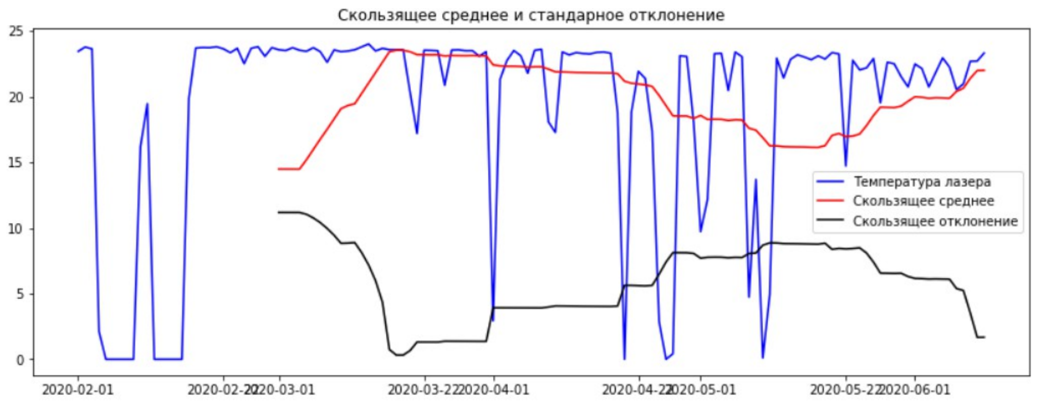
In order to fully utilize further more sophisticated methods, it is necessary to find out whether the time series under study are stationary. It is stationarity that allows to work with time series more efficiently. The DickeyFuller test can be used for this purpose. This test allows you to determine a measure of how stationary a given time series is.



```
In [14]: """Тест Дики-Фуллера на стационарность"""

         from statsmodels.tsa.stattools import adfuller

         df2=df.resample('D').mean()

         def test_stationarity(timeseries):
             rolmean = timeseries.rolling(window=30).mean()
             rolstd = timeseries.rolling(window=30).std()

             plt.figure(figsize=(14,5))
             sns.despine(left=True)
             orig = plt.plot(timeseries, color='blue',label='Температура лазера')
             mean = plt.plot(rolmean, color='red', label='Скользящее среднее')
             std = plt.plot(rolstd, color='black', label = 'Скользящее отклонение')

             plt.legend(loc='best'); plt.title('Скользящее среднее и стандарное отклонение')
             plt.show()

             print ('Результаты теста Дики-Фуллера')
             dftest = adfuller(timeseries, autolag='AIC')
             dfoutput = pd.Series(dftest[0:4],
                                  index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
             for key,value in dftest[4].items():
                 dfoutput['Critical Value (%s)'%key] = value
             print(dfoutput)
         test_stationarity(df2.laser_temp_series.dropna())
         test_stationarity(df2.laser_power_series.dropna())
```

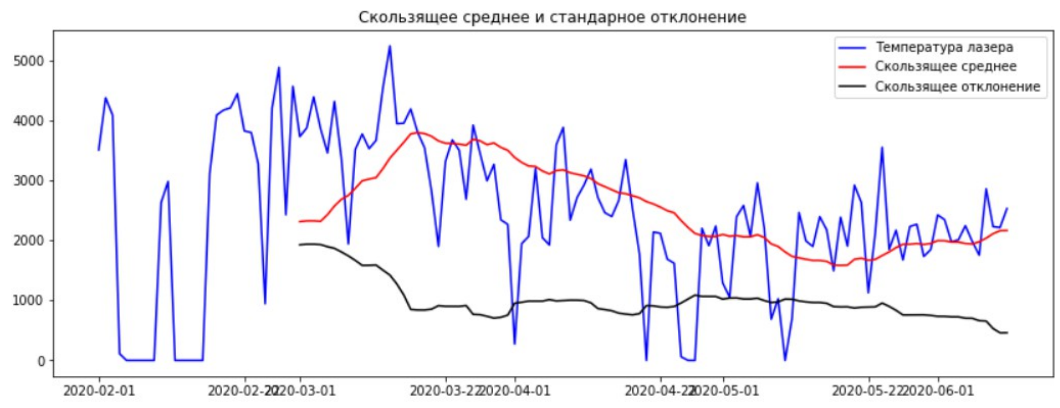
Figure 25 - Code for the DickeyFuller test

Figure 25 shows a code in which two time series of temperature and laser power are analyzed for stationarity. Both time series are averaged over days.



```
Результаты теста Дики-Фуллера
Test Statistic                  -5.048146
p-value                          0.000018
#Lags Used                       0.000000
Number of Observations Used    131.000000
Critical Value (1%)             -3.481282
Critical Value (5%)             -2.883868
Critical Value (10%)            -2.578677
dtype: float64
```

Figure 26 - Test results for laser temperature



```
Результаты теста Дики-Фуллера
Test Statistic                 -3.688439
p-value                         0.004279
#Lags Used                      2.000000
Number of Observations Used   129.000000
Critical Value (1%)            -3.482088
Critical Value (5%)            -2.884219
Critical Value (10%)           -2.578864
dtype: float64
```

Figure 27 - Test results for laser power

Figures 26 and 27 show the result of the test with additional parameters. The main condition for accepting the hypothesis of stationarity was p-value less than 0, 05, which means that the series has no unit roots (the characteristic equation of the autoregressive model of time series does not have roots equal modulo one). As can be seen from the figures, the series exhibit the properties of stationarity.

3.2 Architecture of the developed solution

Figure 28 shows the architecture in the form of a unidirectional graph, the nodes of which are modules of the developed solution, and the vertices are the data flow. The flow is divided into two parallel computation processes: prediction and feature generation.

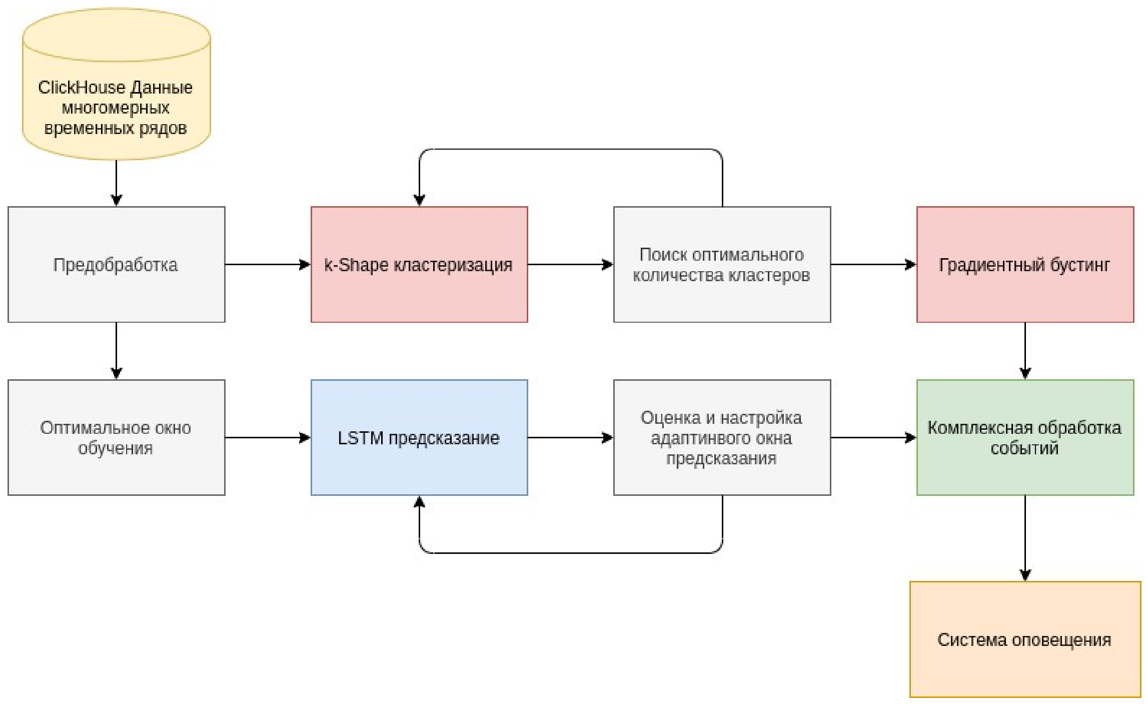

ClickHouse Данные многомерных временных рядов
Предобработка
k-Shape кластеризация
Поиск оптимального количества кластеров
Градиентный бустинг
Оптимальное окно обучения
LSTM предсказание
Оценка и настройка адаптивного окна предсказания
Комплексная обработка событий
Система оповещения

Figure 28 - Architecture diagram

The prediction is performed by a neural network of LSTM architecture, with additional features in the form of determining the optimal training window to improve accuracy and an adaptive prediction window, also to improve accuracy.

The kShape and gradient bousting algorithms are responsible for identifying additional features that characterize the dataset. This approach is necessary to identify additional patterns, some of which may be indicative of faults or near-fault states. It is possible that this block may not be helpful in troubleshooting, but it will contribute to a better understanding of the data.

The predicted LSTM data and the gradient bousting model are integrated into the complex event processing modules. The gradient-boosting model and some superimposed conditions determine whether the predicted time segments are potential faults or failures, or a normal state. If faults are detected, depending on the type of fault, the results are processed and the final data are sent to the fault reporting system.

The data stream can be processed recurrently in prediction and clustering blocks to improve accuracy.

3.3 LSTM training

Training models on time series can be done in different ways. The main way is training based on a sliding window. Suppose that there is a time series $TS=\{x_1, ..., x_n\}$. Then, a sliding window is a fragment of the TS series of size m with step s. Thus, the original series is split into segments of the same size:
$TS=\{S, ._{1}., \frac{S_n}{s}\}$. In this paper, the optimal window size is calculated
based on the periodogram constructed using the LombaScargle method, which was described in the previous section. It is also worth noting that the total training window, i.e., the segment of the entire time series, was chosen as the maximum, since neural networks are better trained on a larger amount of data.

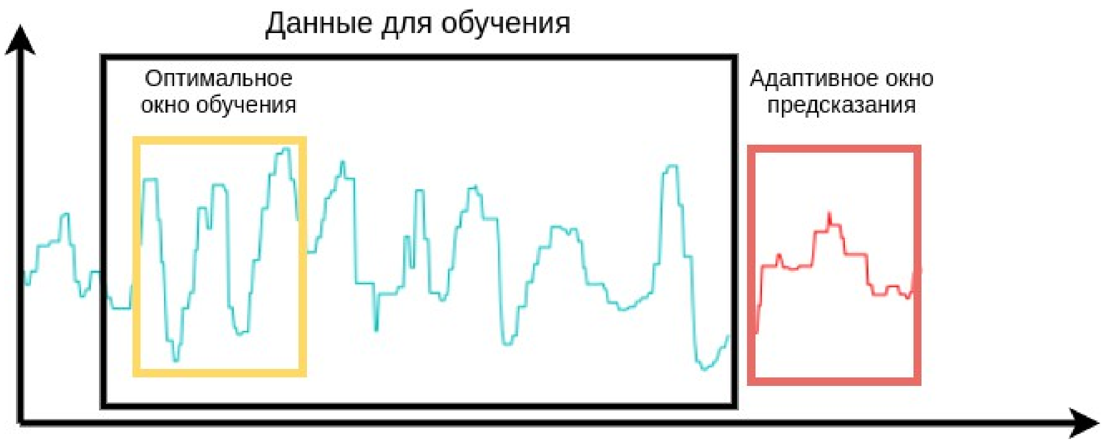


Figure 29 - Description of different time series windows

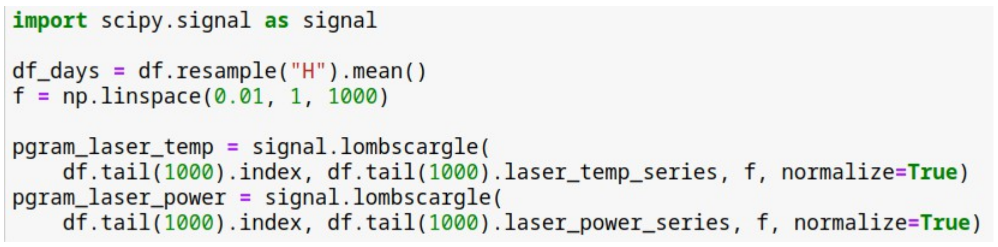
Figure 29 describes an approximate representation of different windows that can be customized based on certain optimality criteria. The diagram also includes a so-called adaptive window prediction, which is determined based on the smallest value of the loss function.

```
function PREDICTIONWINDOW($y_{act}, y_{pred}$)
    $MAPE = mean(abs((y_{act} - y_{pred})/y_{act}) * 100)$
    if $MAPE > 20\%$ then
        $PredictionWindow = PredictionWindow - 1$
    else if $MAPE < 5\%$ then
        $PredictionWindow = PredictionWindow + 1$
    else
        $PredictionWindow = PredictionWindow$
    end if
    return $PredictionWindow$
end function
```

Figure 30 - Adaptive prediction window algorithm

$$= \frac{1}{n}\sum_{t=1}^{n} \frac{A_t - F_t}{A_t} \qquad (18)$$

Figure 30 describes the pseudocode of the program for calculating the adaptive prediction window. MAPE (mean absolute percentage error) is the mean absolute percentage error, which is a measure of prediction accuracy and is expressed by formula 18, A_t is the real value and F_t is the predicted value.



```
import scipy.signal as signal

df_days = df.resample("H").mean()
f = np.linspace(0.01, 1, 1000)

pgram_laser_temp = signal.lombscargle(
    df.tail(1000).index, df.tail(1000).laser_temp_series, f, normalize=True)
pgram_laser_power = signal.lombscargle(
    df.tail(1000).index, df.tail(1000).laser_power_series, f, normalize=True)
```

Figure 31 - Code for using the LombaScargle method

The program code in Figure 31 calculates periodograms for hourly averaged time series of laser temperature and power. The Scipy library was used for this purpose. The data are automatically are normalized and the prevailing frequencies of the time series are determined on the basis of the given frequency.

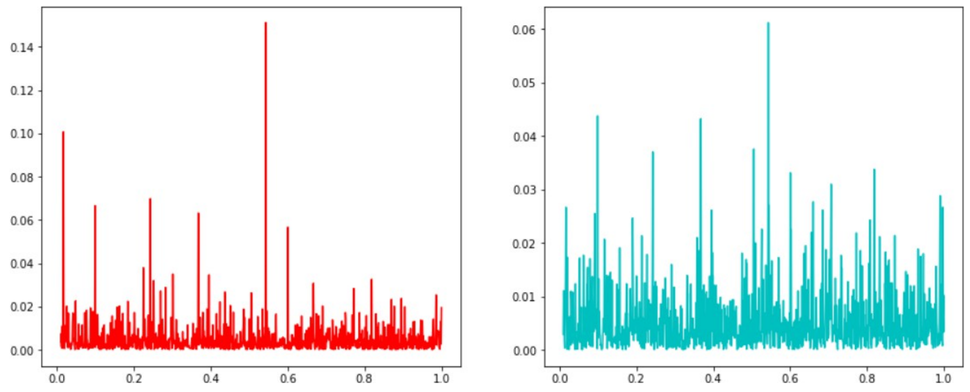
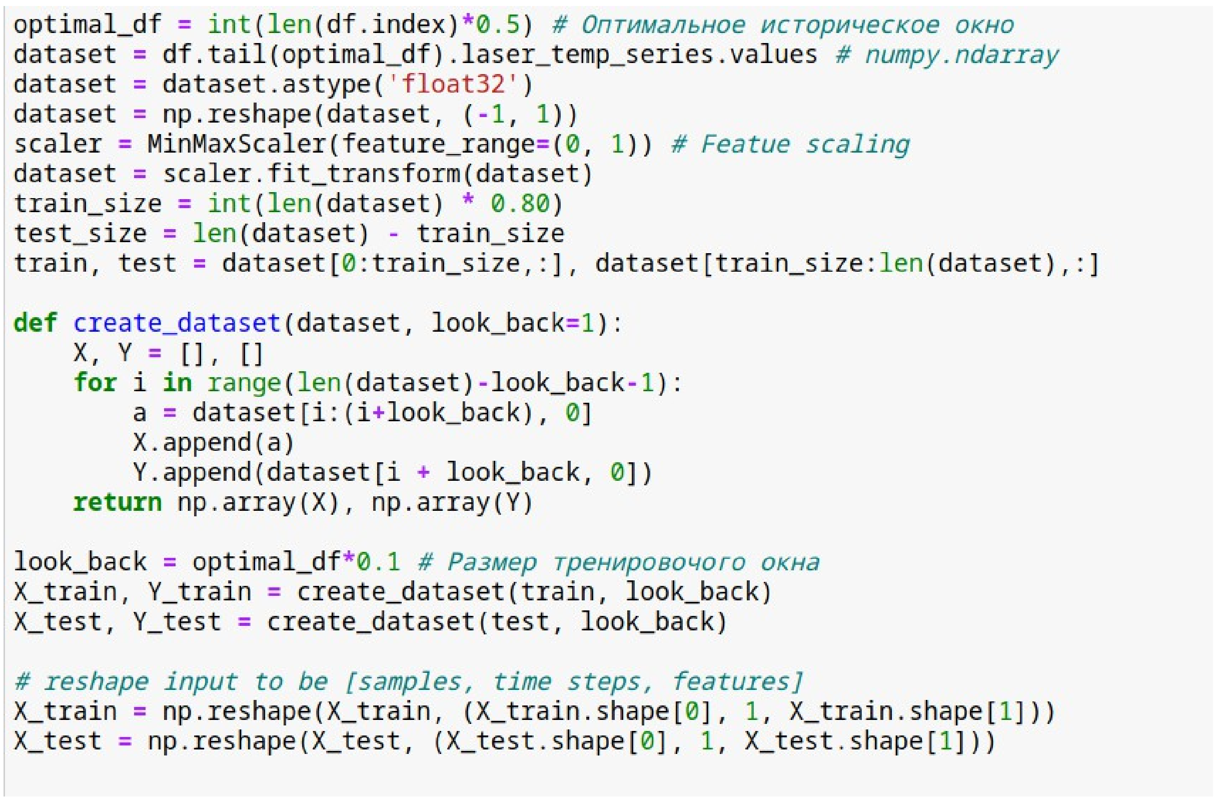


Figure 32 - Result of using the LombaScargle method

Figure 32 shows that there are almost no strongly emphasized frequencies in the data, but in both cases frequencies around the value of 0.5 predominate, so we can say that the optimal historical training window is to split the data into the last half.



```
optimal_df = int(len(df.index)*0.5) # Оптимальное историческое окно
dataset = df.tail(optimal_df).laser_temp_series.values # numpy.ndarray
dataset = dataset.astype('float32')
dataset = np.reshape(dataset, (-1, 1))
scaler = MinMaxScaler(feature_range=(0, 1)) # Featue scaling
dataset = scaler.fit_transform(dataset)
train_size = int(len(dataset) * 0.80)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]

def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)

look_back = optimal_df*0.1 # Размер тренировочого окна
X_train, Y_train = create_dataset(train, look_back)
X_test, Y_test = create_dataset(test, look_back)

# reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
```

Figure 33 - Preparation of laser temperature samples for LSTM training

Figure 33 shows the code for training data preparation. In this case, the code is described with splitting the samples into training and test samples according to the selected optimal windows in the form of

$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$. This code will be embedded and dynamically embedded in the future

change parameters of optimal training and prediction windows based on the described methods.

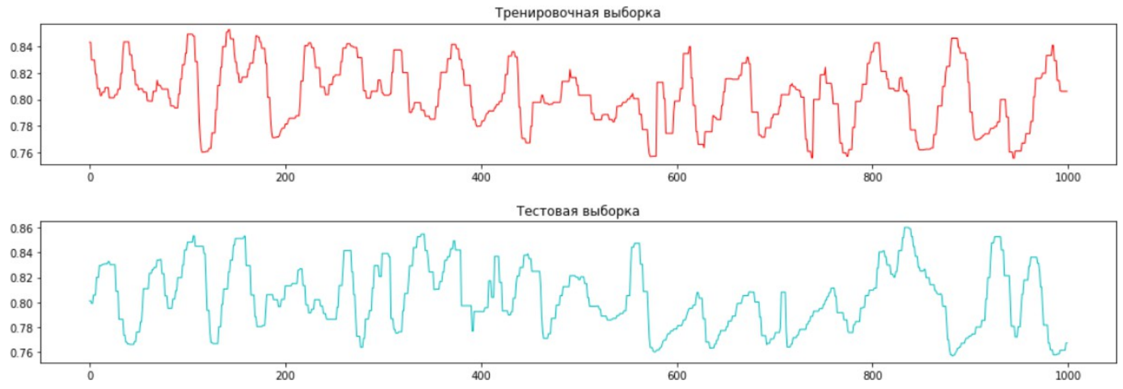


Figure 34 - Prepared laser temperature samples

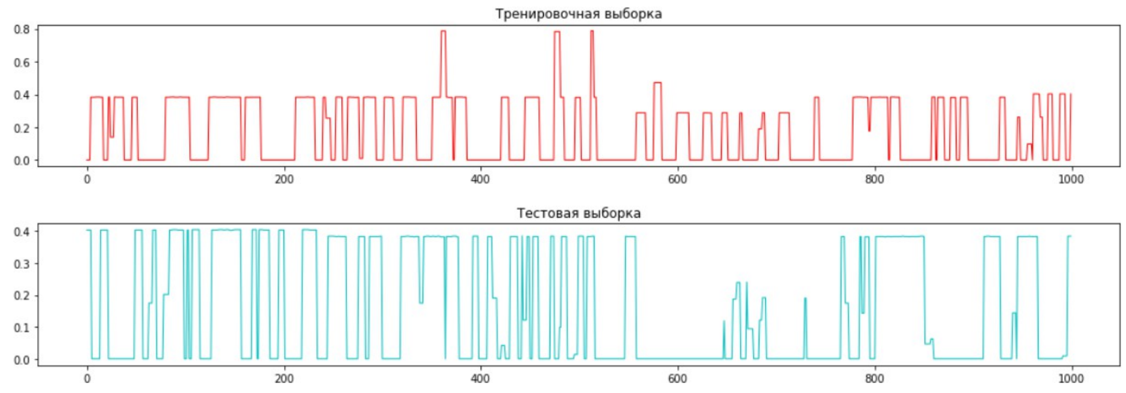
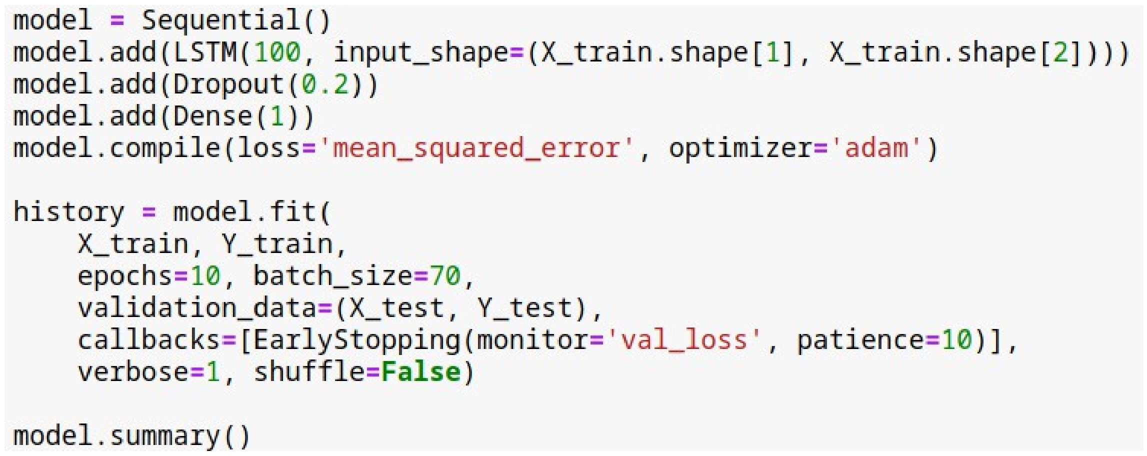


Figure 35 - Prepared laser power samples

The plots in Figures 34 and 35 show the first 1000 values of the training and test samples.



```
model = Sequential()
model.add(LSTM(100, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

history = model.fit(
    X_train, Y_train,
    epochs=10, batch_size=70,
    validation_data=(X_test, Y_test),
    callbacks=[EarlyStopping(monitor='val_loss', patience=10)],
    verbose=1, shuffle=False)

model.summary()
```

Figure 36 - Code for training the LSTM neural network

A number of parameters were chosen to train the LSTM neural network (Figure 36). For the network itself, 100 neurons were defined. Such a parameter is not necessarily optimal, but may be sufficiently suitable for most cases.

In addition, the Dropout regularization method is used for training, which is the random exclusion of a certain percentage of neurons in the process of training the neural network. This method is aimed at preventing overtraining of the network, and empirically, it was found that this type of regularization is the most effective for most cases. For the system under development, a Dropout of 20 percent was chosen, which, as well as the number of neurons, is not necessarily optimal, but it is an appropriate option.

A loss function in the form of RMS error was selected, as well as a method for stochastic optimization of this function Adam. The Adam method is considered to be the best method for optimizing loss functions for 2020. To find the optimal window, another loss function will be used as already described, namely the mean absolute percentage error.

For training, the size of the sent data was chosen to be 70 and the number of training epochs was chosen to be 10, which is sufficient.

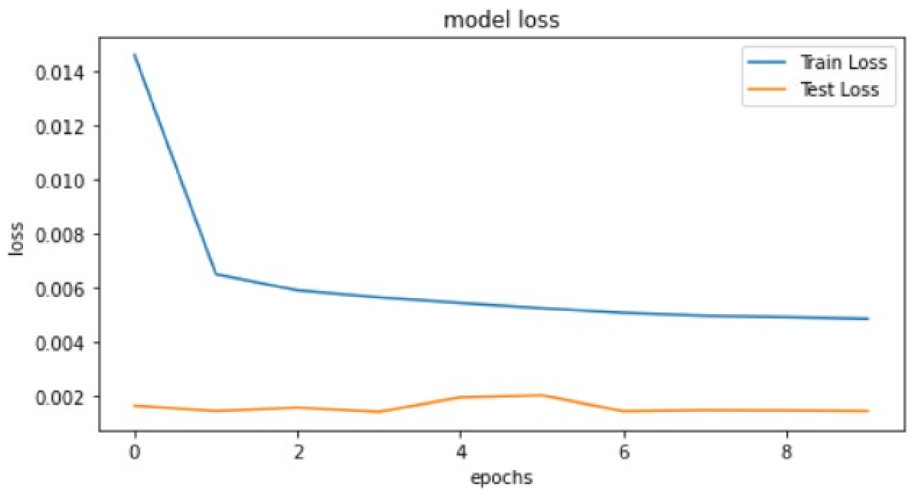


Figure 37 - Dynamics of loss functions during LSTM training for laser temperature

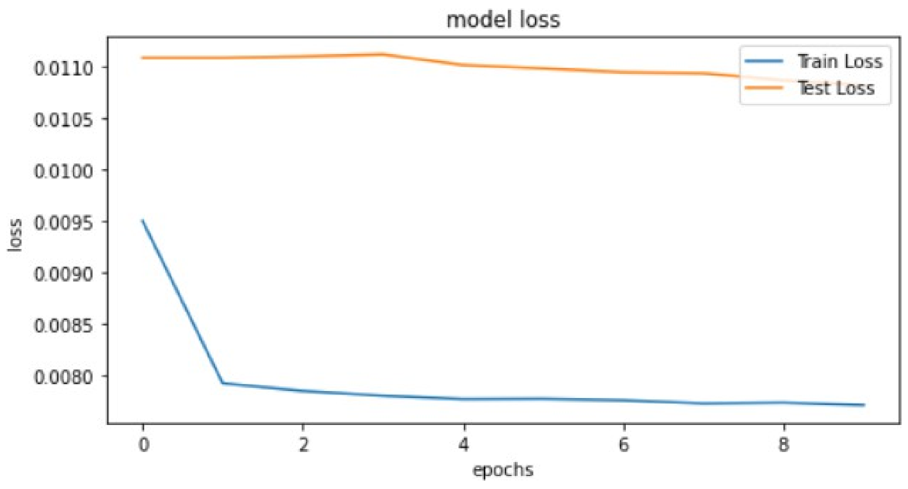


Figure 38 - Dynamics of loss functions during LSTM training for laser power

The graphs in Figures 37 and 38 show the dynamics of changes in the values of the loss functions on the training and test samples during training by epochs. For the laser temperature data, minimization was more successful than for the laser power data.

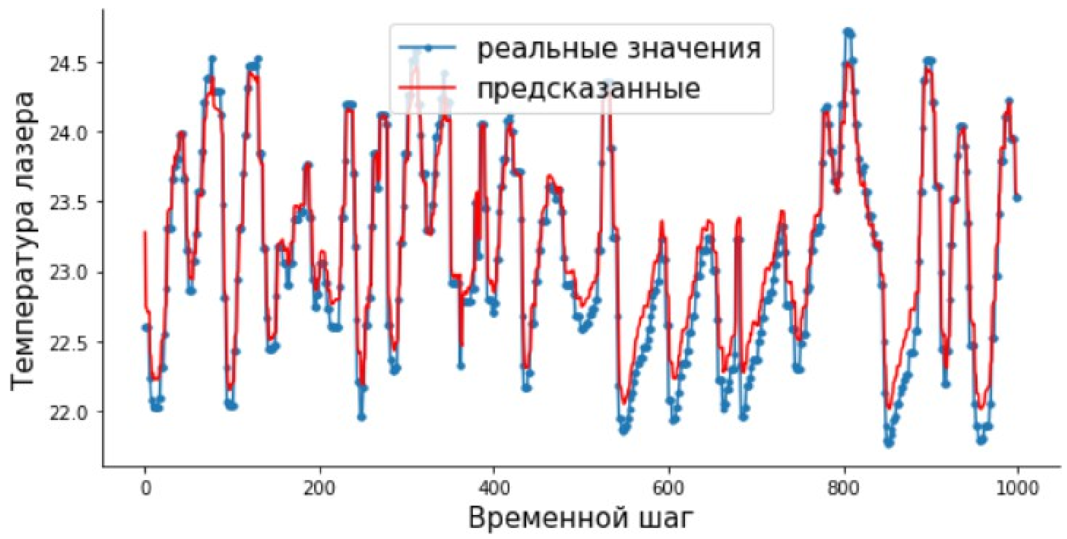


Figure 39 - Real and predicted values for laser temperature

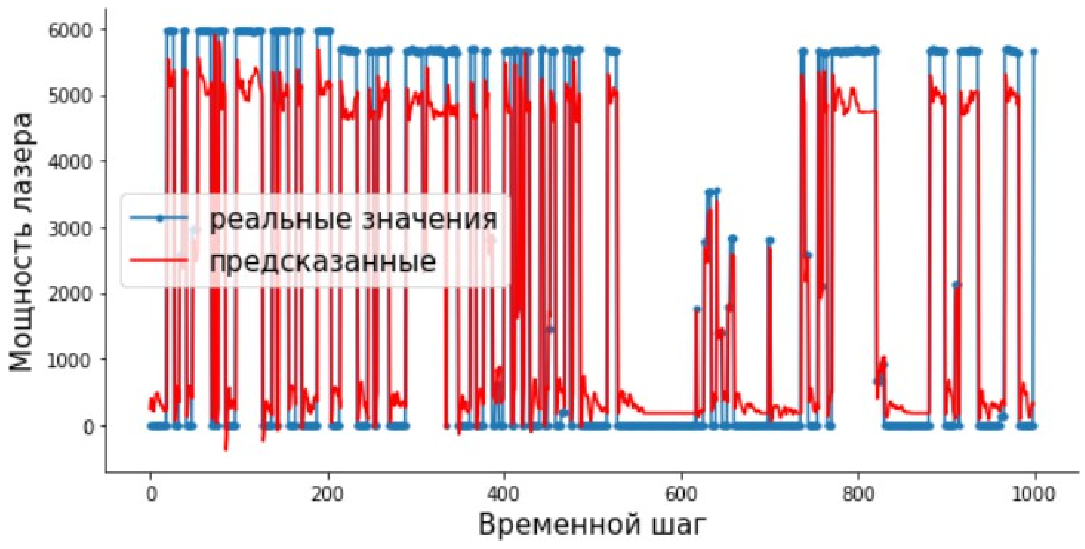


Figure 40 - Real and predicted values for laser power

The final results of the predicted values are plotted in Figures 39 and 40 for the first 1000 values. The predicted values for temperature were better than those for power. This result is due to the non-trivial structure of the laser power time series. However, this difference is not critical because the temperature depends on the power, which means that only temperature values can be used.

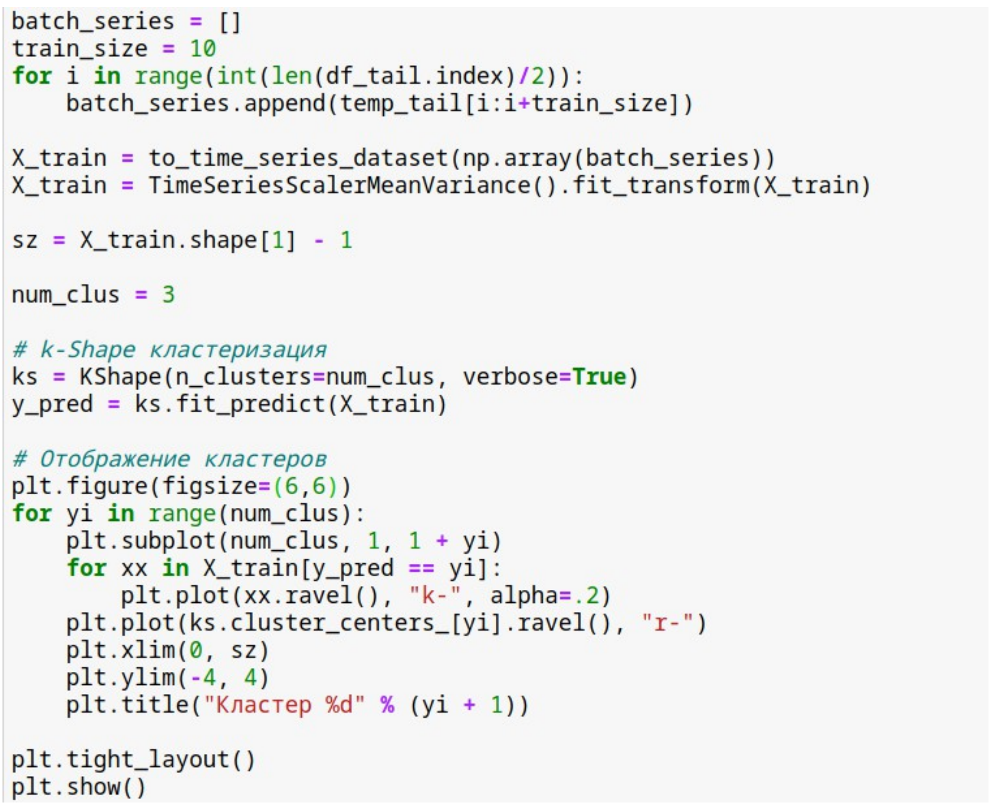
3.4 Clustering based on kShape

Clustering is an experimental step in the process of processing laser cutting machine data and may have uncertain results. However, this paper describes initial results using the kShape algorithm, which will be further improved by using optimization techniques.

Many clustering algorithms require preliminary setting of the number of clusters. This is a research task and has not yet been fully solved. Usually, the choice of the number of clusters depends on the specifics of the problem. In a fault recognition task, it is difficult to determine the appropriate number of clusters without using automatic optimization methods. However, three clusters have been defined: for the normal state, for the average state, and for the fault state.

Figure 41 shows the code for preparing laser temperature data, using the Tslearn-based kShape algorithm, and outputting the results using the Matplotlib library. A window length of 10 was chosen with a step size of half the training historical window. In addition, the data were normalized based on znormalization using the *TimeSeriesScalerMeanV* ariance() function.

Figure 42 shows the result of clustering of the laser temperature time series. From these results, it can be seen that three patterns of machine states are identified: decreasing, increasing, and steady state. Based on the identified patterns, it is possible to create a classification model for the predicted data, which will be used for gradient bousting in the future.



```
batch_series = []
train_size = 10
for i in range(int(len(df_tail.index)/2)):
    batch_series.append(temp_tail[i:i+train_size])

X_train = to_time_series_dataset(np.array(batch_series))
X_train = TimeSeriesScalerMeanVariance().fit_transform(X_train)

sz = X_train.shape[1] - 1

num_clus = 3

# k-Shape кластеризация
ks = KShape(n_clusters=num_clus, verbose=True)
y_pred = ks.fit_predict(X_train)

# Отображение кластеров
plt.figure(figsize=(6,6))
for yi in range(num_clus):
    plt.subplot(num_clus, 1, 1 + yi)
    for xx in X_train[y_pred == yi]:
        plt.plot(xx.ravel(), "k-", alpha=.2)
    plt.plot(ks.cluster_centers_[yi].ravel(), "r-")
    plt.xlim(0, sz)
    plt.ylim(-4, 4)
    plt.title("Кластер %d" % (yi + 1))

plt.tight_layout()
plt.show()
```

Figure 41 - Code for using the kShape algorithm

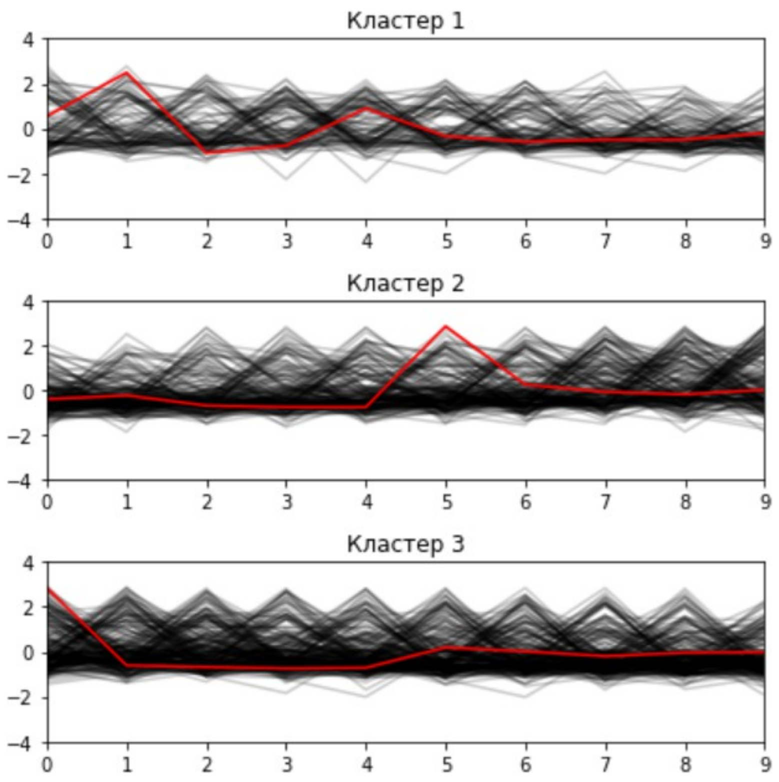


Figure 42 - Clustering results

3.5 Analysis of predicted values

The LSTM-predicted time series can be used for subsequent fault detection testing. Trend analysis and average of predicted values are used for fault detection.

Critical for the temperature value is a sustained high temperature above 120 degrees Celsius and a sustained low temperature below 30 degrees Celsius for a long period of time (about a day). For laser power, it is necessary to monitor the trend of its decrease over a day, as well as the exceeding of the used power above the set power within a minute. These limits for the studied parameters of the Navigator machine are determined on the basis of engineering requirements of the machine developers of VNITEP company.

The described constraints will be implemented in the Omniprocessing microservice based on a simple average calculation using the Gorobots statistics microservice, as well as on simple conditional constructs written in Python.

3.6 Commissioning

The entire described architecture of the developed module for prediction and fault detection will be integrated, as already described, into the Omniprocessing microservice. However, a number of problems related to clustering and the use of the gradient booing algorithm need to be solved, namely the determination of the optimal parameters for the algorithms, for example, the determination of the optimal number of clusters. The gradient bousting algorithm itself needs to be implemented based on optimized clustering, so it has not been described in this paper. Apart from that, The integration itself is a non-trivial task and requires solving problems related to the interfacing of all submodules taking into account the environment in which these submodules are integrated, namely in the Faust framework environment. Also a separate task is the use of Gorobots service for calculation of average trends and predicted values.

After solving the set tasks, the developed module will be deployed together with the whole Omnicube platform using Kubernetes. The fault report will be sent to the OmniNotifier notification service.

3.7 Conclusions

On the basis of preprocessing of data from the laser cutting machine Navigator KS12V, the primary analysis of features and properties of the selected parameters of the machine: temperature and laser power was carried out. For this purpose a number of tests were carried out: for normality and for stationarity of the series.

The optimal window for training the prediction model is identified. The use of prediction model - LSTM neural network is described. The hyperparameters of the model are characterized.

The use of the kShape algorithm for clustering is described, as well as the results of this use. Patterns of machine states, which can be further used by the gradient bousting algorithm, are identified. The optimization problems of this algorithm are outlined.

At the end of the section there is a description of the principles of integration with Omnicube system.

CONCLUSION

The result of this thesis is the solution of the problem of automatic prediction and fault detection on laser cutting machines Navigator KS12V of VNITEP company. To obtain the result a number of tasks were solved.

The problem of detecting errors and failures on laser cutting machines of the model Navigator KS12V, as well as in devices in general, was outlined. The peculiarities of this problem and its theoretical aspects are described. The existing solutions based on different methods, approaches and ideas are analyzed. For each case a criticism was given, namely the advantages and disadvantages of each solution were outlined. Also the problems, the solutions of which can help to achieve the solution of the problem of error detection and forecasting, were outlined.

The Omnicube system was analyzed, into which the Navigator machine tool error prediction and detection solution will be integrated.

On the basis of preprocessing of data from the laser cutting machine Navigator KS12V, the primary analysis of features and properties of the selected parameters of the machine: temperature and laser power was carried out. For this purpose a number of tests were carried out: for normality and for stationarity of the series.

The recurrent neural network LSTM was chosen for forecasting tasks. The advantages over other neural network architectures for time series forecasting tasks were described. A detailed description of LSTM network operation was given. The principle by which the optimal training window is determined was also described. The optimal window for training the prediction model was determined. The use of the prediction model was described

The hyperparameters of the model are characterized. The hyperparameters of the model are characterized.

In addition, clustering algorithms were analyzed, and the kShape algorithm for clustering multivariate time series was selected and described. The use of the kShape algorithm for clustering is described, as well as the results of this use. Patterns of machine states, which can be further used by the algorithm of graph bousting, are identified. The optimization problems of this algorithm are outlined. A brief description of the tools used for the development is given, namely the analysis of libraries and frameworks for data analysis tasks and tasks of machine learning tools designed for the Python language.

At the end there is a description of the principles of integration with Omnicube system.

In conclusion, it can be said that all the tasks set have been accomplished and the goal of the thesis has been achieved.

REFERENCE LIST

1. Loskutov A.Y. Analysis of time series. Course of lectures. - Faculty of Physics, Moscow State University, 2018. - 113 c.
2. Shitikov V.K. Classification, regression and other Data Mining algorithms using R. - Institute of Ecology of the Volga Basin RAS, 2017. - 351 c.
3. Martin Fowler. NoSQL: a new methodology for developing non-relational databases. - Williams, 2016. - 192 c. - ISBN 9785845918291.
4. Marc S. Paolella. Linear Models and TimeSeries Analysis. - Wiley, 2019. - 897 c. - ISBN: 9781119431855.
5. Douglas C. Montgomery. Introduction to time series analysis and forecasting. - Wiley, 2015. - 671 c.
6. Terence C. Mills. Applied Time Series Analysis. A practical guide to modeling and forecasting. - Academic press, 2019. - 356 c. - ISBN: 978012813117 6.
7. George Box. Time series analysis. Forecasting and control. - Wiley, 2016. - 709 c. - ISBN 9781118675021.
8. Mohammed J. Zaki. Data mining and analysis. // Fundamental Concepts and Algorithms. - Cambridge university press, 2018. - 562 c. - ISBN 978 0521766333.
9. Zhiqiang Ge. Multivariate statistical process control. - Springer, 2019. - 203
   c. - ISBN 9781447145134.

10. William W.S. Wei. Multivariate Time Series Analysis and Applications. - Wiley, 2019. - 528 с.

11. Zhihua Zhang. Multivariate Time Series Analysis in Climate and Environmental Research. - Springer, 2018. - 293 c. - ISBN 9783 319673394.

12. Ruey S. Tsay. Multivariate Time Series Analysis. - Wiley, 2016. - 522 c.

13. Kamil Feridun Turkman. NonLinear Time Series. Extreme Events and Integer Value Problems. - Springer, 2018. - 255 c. - ISBN 9783319070278.

14. Robert H. Shumway. Time Series Analysis and Its Applications. With R Examples. - Springer, 2019. - 202 c. - ISBN 9781441978646.

15. Tata Subba Rao. Time Series Analysis: Methods and Applications. - Elsevier, 2017. - 777 c. ISBN: 9780444538581.

16. Boris Mirkin. Core Concepts in Data Analysis: Summarization, Correlation and Visualization. - Springer, 2018. - 402 c.

17. Mehmed Kantardzic. Data mining. Concepts, Models, Methods, and Algorithms. - Wiley, 2020. - 661 c.

18. Charu C. Aggarwal. Frequent Pattern Mining. - Springer, 2019. - 85 c.

19. Jiawei Han. Data Mining. Concepts and Techniques. - Morgan Kaufmann Publishers, 2012. - 740 c.

20. Jure Leskovec. Mining of Massive Datasets. - Stanford, 2019. - 513 c.

21. Predictive Statistics. Analysis and Inference beyond Models. - Cambridge University Press, 2018. - 657 c. - ISBN 9781107028289.

22. Max Kuhn. Applied Predictive Modeling. - Springer, 2018. - 615 c. -

ISBN 9781461468486.

23. Richard Sutton. Reinforcement learning. - The MIT Press, 2018. - 548 c.

24. Mehryar Mohri. Foundations of Machine Learning. - The MIT Press, 2018. - 505 c.

25. Boyes, Hugh. The industrial internet of things (IIoT): An analysis framework. - Computers in Industry, 2018. - 112 c.

26. Jonathan Law. Composable models for online Bayesian analysis of streaming data. - Springer Statistics and Computing 28, 2017. - 1119-1137 c.

27. E. L. Ionides, C. Bretó, and A. A. King. Inference for nonlinear dynamical systems. Proceedings of the National Academy of Sciences of the United States of America, 103(49):18438-43, 2016.

28. Sunny Sanyal, Dapeng Wu, and Boubakr Nour. A Federated Filtering Framework for Internet of Medical Things. IEEE International conference on communications (IEEE ICC 2019).

29. Simon Haykin. Adaptive Filter Theory. - Prentice Hall, 2016. - ISBN 013 0484342.

30. G. W. Stewart. Matrix perturbation theory. - Wiley, 2017. - 234 c.

31. Steven Van Vaerenbergh. Kernel Recursive LeastSquares Tracker for Time Varying Regression. - IEE Transactions on neural networks and learning systems, vol. 23, No 8, 2018.

32. Paynabar K, Jin J, Agapiou J, Deeds P (2012) Robust Leak Tests for Transmission Systems Using Nonlinear MixedEffect Models. Journal of Quality Technology 44 (3):265278

33. Grasso M, Colosimo BM, Tsung F (2017) A phase I multimodeling

approach for profile monitoring of signal data. International Journal of Production Research 55 (15):43544377. doi:10.1080/00207543.2016.1251626

34. Lei Y, Zhang Z, Jin J (2016) Automatic Tonnage Monitoring for Missing Part Detection in MultiOperation Forging Processes. Journal of Manufacturing Science and EngineeringTransactions of the Asme 132 (5). doi:10.1115/1.4002531

35. Yang WA, Zhou Q, Tsui KL (2016) Differential evolutionbased feature selection and parameter optimization for extreme learning machine in tool wear estimation. International Journal of Production Research 54 (15):4703 4721. doi:10.1080/00207543.2015.1111534

36. Grasso M, Colosimo BM, Pacella M (2018) Profile monitoring via sensor fusion: the use of PCA methods for multichannel data. International Journal of Production Research 52(20):61106135. doi:10.1080/00207543.2014.916431

37. Zerehsaz Y, Shao C, Jin J (2016) Tool wear monitoring in ultrasonic welding using high-order decomposition. Journal of Intelligent Manufacturing 30 (2):657669. doi:10.1007/s10845-016-1272-4

38. Paynabar K, Jin J, Pacella M (2017) Monitoring and diagnosis of multichannel nonlinear profile variations using uncorrelated multilinear principal component analysis. IIE Transactions 45 (11):12351247. doi:10.1080/0740817x.2013.770187

39. Lu H, Plataniotis KN, Venetsanopoulos AN (2019) Uncorrelated Multilinear Principal Component Analysis for Unsupervised Multilinear Subspace Learning. IEEE Transactions on Neural Networks 20 (11):18201836. doi:10.1109/tnn.2009.2031144

40. Lu H, Plataniotis KN, Venetsanopoulos AN (2019) Uncorrelated Multilinear Discriminant Analysis With Regularization and Aggregation

for Tensor

Object Recognition. IEEE Transactions on Neural Networks 20 (1):103123. doi:10.1109/tnn.2008.2004625.

41. Feng Ye, Zhijie Xia, Min Dai, Zhisheng Zhang RealTime Fault Detection and Process Control Based on Multichannel Sensor Data Fusion. Journal of Intelligent Manufacturing, 2020.

42. Adnan Akbar, Abdullah Khan. Predictive Analytics for Complex IoT Data Streams. IEEE INTERNET OF THINGS JOURNAL, VOL. 10, NO. 10, 20 2017.

43. John Paparrizos, kShape: Efficient and Accurate Clustering of Time Series. Energy and Buildings Volume 146, July 1, 2017, Pages 2737 https://doi.org/10.1016/j.enbuild.2017.03.071

44. Data Mining Curriculum. - ACM SIGKDD Curriculum Committee, 2016. - 10 c.

45. Site of Omnicube company [Electronic resource]. - Access mode: https://www.omnicube.ru/, free. - (01.06.20)

46. Official Kafka documentation [Electronic resource]. - Access mode: https://kafka.apache.org/documentation/, free. - (10.05.20)

47. ClickHouse official documentation [Electronic resource]. - Access mode: https://clickhouse.tech/, free. - (10.05.20)

48. Faust framework [Electronic resource]. - Access mode: https://faust.readthedocs.io/en/latest/, free. - (10.05.20)