Anomaly States Monitoring of Large-Scale Systems with Intellectual Analysis of System Logs

Oleg Sheluhin, Andrey Osin
Moscow Technical University of Communications and Informatics
Moscow, Russia
{sheluhin, osin_a_v}@mail.ru

Abstract—**The article analyzes the paths and algorithms for automating the monitoring of computer system states by means of intellectual analysis of unstructured system log data in order to detect and diagnose abnormal states. This information is necessary for technical support to locate the problem and diagnose it accurately. Because of the ever-growing log size, mining data mining models are used to help developers extract system information. At the first stage, logs are collected with records of system states and information on the execution of processes. At the second stage, the log parser is used to retrieve a group of event templates, with the result that the raw logs are structured. At the third stage, after the logs are parsed into separate patterns, they are additionally represented as numerical vectors of attributes (attributes). The set of all vectors forms a matrix of signs. In the fourth stage, the feature matrix is used to detect anomalies of machine learning methods to determine whether the new incoming log sequence is abnormal or not. A decision tree was used as a classification method for machine learning. Using the example of a distributed HDFS data set, the effectiveness of the considered method for detecting anomalous system states is shown.**

I. Introduction

The main purpose of the system log is to record system states and important events at various critical points to help debug system failures and perform root cause analysis. Log data is an important and valuable resource for understanding the state of the system and its performance, so various system logs are an excellent source of information for online monitoring and the detection of anomalies [1], [2], [3], [4], [6].

By anomalous, we mean instances in a data set that do not correspond to the regular behavior of the system. The anomaly detection system consists of four successive stages: the collection of initial data, analysis and processing of the log, the extraction of features and numerical representation, the construction of an anomaly detection model [8], [12], [13], [14].

The purpose of the analysis and processing of system logs (parsing) is to separate the permanent and variable parts of messages and the formation of patterns (templates) of events [9], [10]. To use the observed unstructured data in machine learning algorithms, it is necessary to carry out their numerical representation [11]. To this end, after processing the log, one of the algorithms for the numerical representation is used [12].

At the next stage, possible paths and algorithms for automating the monitoring of computer system states by means of intellectual analysis of unstructured data of system logs are analyzed in order to detect and diagnose abnormal states.

The result of processing are clusters of messages, separated by event type. To do this, the log data is divided into different groups characterizing the sequence of messages. This is done by sequentially reading the log file and storing a certain number of events in the memory area (called a window) when processing a dataset. Windows can be fixed, sliding and session [13].

As a result of this step, a sequence of vectors is formed in the numerical form of each sequence of messages. To search for abnormal (emergency) states of the system, the vectors are processed by machine learning methods.

The aim of the work is to study the efficiency of machine learning algorithms for detecting abnormal (emergency) states of large computer systems by automated processing of unstructured data of a large amount of system logs.

II. Logs Processing Ways

Large-scale systems typically generate logs for recording system status and runtime information, each containing a timestamp and a log message indicating what happened. This valuable data can be used for a variety of purposes (for example, anomaly detection), and therefore logs are first collected for further use.

Log analytics involves searching the logs of billions of records, collecting and analyzing real-time data and a variety of visualizations. These features, combined with the flexibility of the data sources available, have made log analytics an excellent option for gaining visibility and understanding of the situation.

The relevance of a log file may vary from one person to another. Perhaps, the data of a particular log can be useful for one user, but not related to another user. Therefore, useful log data may be lost within a large cluster. Therefore, analyzing a log file is an important aspect today.

When managing real-time data, the user can use the log file to make decisions. But, since the amount of data increases, say, to gigabytes, it becomes impossible for traditional methods to analyze such a huge log file and determine the necessary data. By ignoring log data, a huge amount of relevant information will be missed.

Log data can be presented as a pivot table or file. In a log file or table, the records are ordered according to time. All software applications and systems produce log files. Some examples of log files are a transaction log file, an event log file, an audit log file, server logs, etc.

Logs are usually application specific. Thus log analysis is necessary task for extracting valuable information from a log file.

TABLE I. LOG TYPES

Log name	Log source	Information in the log data
Transaction log	Database management system	Consists of information about pending transactions, changes made by rollback operations, and changes that are not updated in the database. It is performed to save the ACID property (atomicity, consistency, isolation, durability property) during failures
Message log	Internet chat (IRC) and instant messaging (IM)	In the case of IRC, consists of server messages during the time interval during which the user is connected to the channel. On the other hand, to ensure user privacy, IM allows you to store messages in an encrypted form in the form of a message log. A password is required to decrypt and view these logs.
System log	Network devices such as web servers, routers, switches, printers, etc.	Syslog messages provide information about where, when and why, i.e. IP address, time stamp and log message. Contains two parts: the object (the source of the message) and security (the degree of importance of the log message)
Server log file	Web servers	It is created automatically and contains information about the user in the form of three parts, such as the IP address of the remote server, the time stamp and the document requested by the user.
Audit logs	Hadoop Distributed File System (HDFS) ANN Apache Spark.	Record all HDFS access activity occurring with the Hadoop platform
Demon logs	Docker	Contains detailed information about the interaction between containers, the Docker service, and the host machine. By combining these interactions, you can identify the container cycle and violation in the Docker service.
Pods	Kubernetes	This is a collection of containers that share resources, such as a single IP address and shared values.
Amazon CloudWatch Logs	Amazon Web Services (AWS)	Used to monitor applications and systems using log data, that is, to check for errors with the application and the system. It is also used to store and access system log data.
Quick logs	Openstack	These logs are sent to the Syslog and managed by the log level. Used to monitor the cluster, audit records, extract reliable server information, and more.

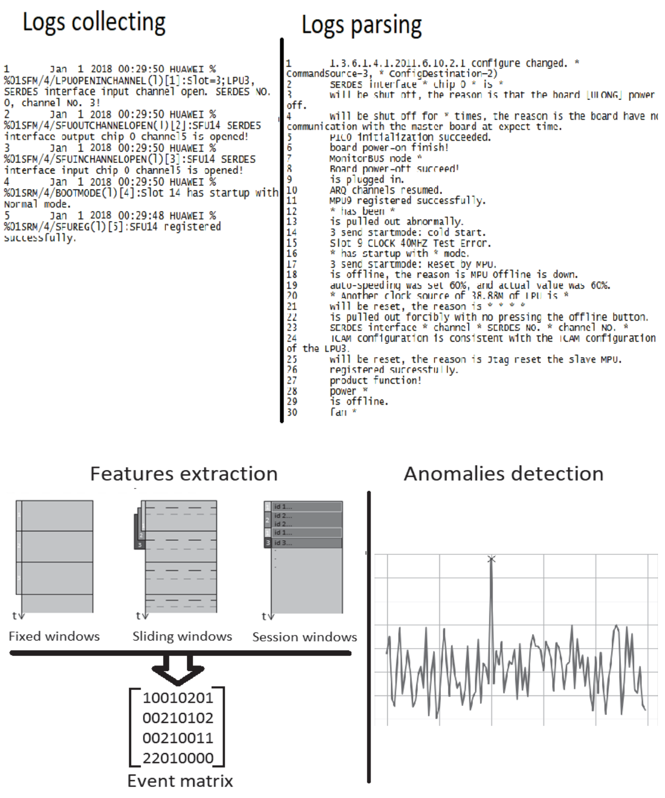
The main steps in the log processing steps are: collecting and clearing data; data structuring; data analysis. Log data is collected from various sources. The information collected must be accurate and informative, as the type of data obtained may affect performance. Therefore, information should be collected from real users. Each type of log contains a different kind of information.

Analysis of the structured log data can be performed in various ways, such as pattern recognition, normalization, classification using machine learning, correlation analysis, and much more. With the introduction of Data Mining to analyze logs, the quality of log data analysis is increasing. However, there are several problems with analyzing logs using data mining:

- Every day the amount of log data increases from megabytes to gigabytes or even petabytes. There is a need for additional log analysis tools;
- The log files do not always contain all the necessary information. So additional efforts are needed to extract useful data;
- Different numbers of logs comes from different sources. As a result, logs in various formats should be analyzed. Having different logs creates a data redundancy problem.

III. THE STRUCTURE OF THE SYSTEM LOG DATA PROCESSING ALGORITHM

The structure of the anomaly detection algorithm using data analysis of system logs includes four stages: log collection (logs), log analysis (parsing), feature extraction and anomaly detection [1].



```
1       Jan  1 2018 00:29:50 HUAWEI %
%01SFM/4/LPUOPENINCHANNEL(l)[1]:Slot=3;LPU3,
SERDES interface input channel open. SERDES NO.
0, channel NO. 3!
2       Jan  1 2018 00:29:50 HUAWEI %
%01SFM/4/SFUOUTCHANNELOPEN(l)[2]:SFU14 SERDES
interface output chip 0 channel5 is opened!
3       Jan  1 2018 00:29:50 HUAWEI %
%01SFM/4/SFUINCHANNELOPEN(l)[3]:SFU14 SERDES
interface input chip 0 channel5 is opened!
4       Jan  1 2018 00:29:50 HUAWEI %
%01SRM/4/BOOTMODE(l)[4]:Slot 14 has startup with
Normal mode.
5       Jan  1 2018 00:29:48 HUAWEI %
%01SRM/4/SFUREG(l)[5]:SFU14 registered
successfully.
```

```
1       1.3.6.1.4.1.2011.6.10.2.1 configure changed. *
CommandSource-3, * ConfigDestination-2)
2       SERDES interface * chip 0 * is *
3       will be shut off, the reason is that the board [ULONG] power
off.
4       will be shut off for * times, the reason is the board have no
communication with the master board at expect time.
5       PICO initialization succeeded.
6       board power-on finish!
7       MonitorBUS node *
8       Board power-off succeed!
9       is plugged in.
10      ARQ channels resumed.
11      MPU9 registered successfully.
12      * has been *
13      is pulled out abnormally.
14      3 send startmode: cold start.
15      Slot 9 CLOCK 40MHZ Test Error.
16      * has startup with * mode.
17      3 send startmode: Reset by MPU.
18      is offline, the reason is MPU Offline is down.
19      auto-speeding was set 60%, and actual value was 60%.
20      * Another clock source of 38.88M of LPU is *
21      will be reset, the reason is * * * *
22      is pulled out forcibly with no pressing the offline button.
23      SERDES interface * channel * SERDES NO. * channel NO. *
24      ICAM configuration is consistent with the ICAM configuration
of the LPU3.
25      will be reset, the reason is Jtag reset the slave MPU.
26      registered successfully.
27      product function!
28      power *
29      is offline.
30      fan *
```

Fig. 1. Steps in detecting system log anomalies

A. Logs collecting

Large-scale systems typically generate logs of system status and process execution information, each containing a timestamp and a log message indicating what happened in the system. This data can be used to detect anomalies.

An example of a fragment of the log file is shown in Fig. 2.

```
148     Jan  1 2018 00:13:15 HUAWEI
%%01SRM/4/BOOTMODE(l)[148]:Slot 14 has startup with Normal
mode.
164     Jan  1 2018 00:11:55 HUAWEI
%%01SRM/4/BOOTMODE(l)[164]:Slot 3 has startup with Normal
mode.
```

```
198        Jan  1 2018 00:11:23 HUAWEI
%%01SRM/4/BOOTMODE(l)[198]:Slot 11 has startup with
FastBoot mode.
```

Fig. 2. Log file fragment

B. Logs analysis

The system logs are usually not structured and contain free-form text. The purpose of the parse is to extract a group of event patterns, with the result that the raw logs can be structured.

The purpose of the message type search is to create message patterns that exist in the log file (Fig. 3). The wildcard characters "*" represent variable messages.

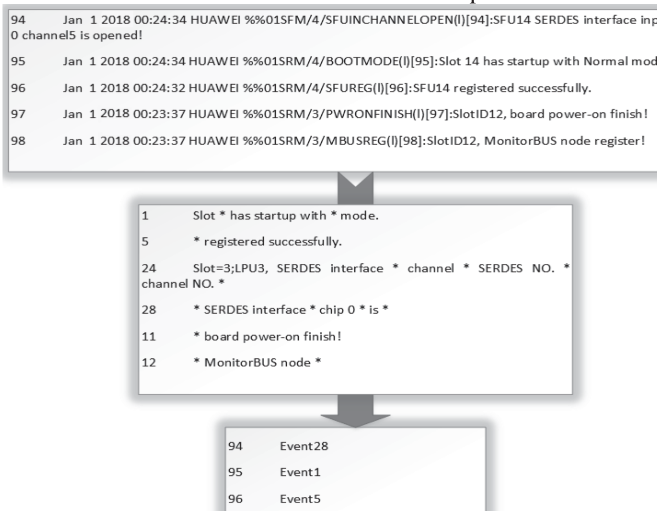
If each text line in the event log is considered a data point, and its individual words are considered attributes, then the clustering task is reduced to grouping similar log messages. For example, “Command has completed successfully” can be considered a 4-dimensional data point with the following attributes “Command”, “has”, “completed”, “successfully”.

As a result, each log message can be divided by a parser into a pattern (pattern) of an event (fixed part) and parameters (variable part).

C. Attributes extraction

After the logs are parsed into separate patterns, it is necessary to additionally present them in the form of numerical feature vectors, which makes it possible to apply machine learning methods at the next stage. To this end, the raw logs are first “sliced” into a set of logical sequences using various grouping methods, including fixed, sliding windows and session windows.

Fixed windows are based on a mark that contains the time each message appeared. Each fixed window has its own size (time interval). The window size is a constant value equal to Δt, for example, one hour or one day. Thus, the number of fixed windows depends on the specified window size. Events that occurred in one window are considered as a sequence of events.



```
94        Jan  1 2018 00:24:34 HUAWEI %%01SFM/4/SFUINCHANNELOPEN(l)[94]:SFU14 SERDES interface inp
0 channel5 is opened!

95        Jan  1 2018 00:24:34 HUAWEI %%01SRM/4/BOOTMODE(l)[95]:Slot 14 has startup with Normal mod

96        Jan  1 2018 00:24:32 HUAWEI %%01SRM/4/SFUREG(l)[96]:SFU14 registered successfully.

97        Jan  1 2018 00:23:37 HUAWEI %%01SRM/3/PWRONFINISH(l)[97]:SlotID12, board power-on finish!

98        Jan  1 2018 00:23:37 HUAWEI %%01SRM/3/MBUSREG(l)[98]:SlotID12, MonitorBUS node register!
```

```
1         Slot * has startup with * mode.

5         * registered successfully.

24        Slot=3;LPU3, SERDES interface * channel * SERDES NO. *
channel NO. *

28        * SERDES interface * chip 0 * is *

11        * board power-on finish!

12        * MonitorBUS node *
```

```
94        Event28

95        Event1

96        Event5
```

Fig. 3. System log parsing

In contrast to fixed windows, sliding windows are characterized by two attributes: the window size and the step size. For example, hourly windows, sliding in increments every five minutes. In general, the step size is smaller than the window size, which causes the overlapping of adjacent windows. The number of sliding windows is usually larger than fixed windows, which mainly depends on the window size and step size.

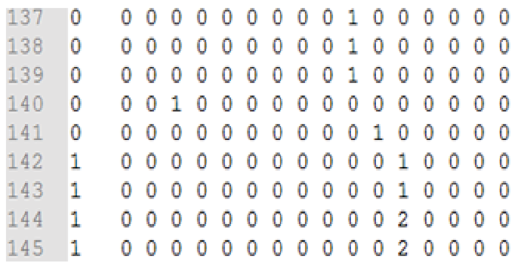
Session session windows are characterized by identifiers instead of a timestamp. Identifiers are used to denote different execution paths in some log data. For example, HDFS logs with block_id record allocation, recording, replication, deletion of a specific block. As a result, there is the possibility of grouping according to identifiers, where each session window has a unique identifier.

After constructing the log sequences, an X event matrix is generated. To this end, the number of occurrences of each log event is calculated in each sequence to form the event count vector. For example, if the event count vector is [0, 0, 2, 3, 0, 1, 0], this means that in this sequence, event 3 occurred twice, and event 4 occurred three times. Finally, the set of vectors constitutes the matrix of events X, where the entry Xi, j reflects the number of times the event j occurred in the i-th log sequence.

In the next step a vector object (event count vector) is created for each log sequence which is the number of occurrences of each event. The set of all vectors forms a matrix of attributes (attributes), which is also an event counter matrix.

In this paper, the division of the log was made by fixed time windows. A time window of 30 seconds was selected (Fig. 4).

Having obtained these vectors, one can apply machine learning methods to search for system alarm states. In order to avoid the concentration of alarms in one part of the data set, a random permutation is first performed.



```
137 0    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
138 0    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
139 0    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
140 0    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
141 0    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
142 1    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
143 1    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
144 1    0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0
145 1    0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0
```

Fig. 4. Fragment of a file with features vectors

When dividing the available data into a set of training and testing, we dramatically reduce the number of samples that can be used to study the model, and the results may depend on the random selection of a pair of sets (training, testing). Also, taking into account the fact that the number of abnormal instances is sufficiently small, it is necessary to use a different method of preliminary preparation of the data set for training and testing.

Therefore, in this paper, the next step was to use a procedure called cross-validation (CV). In an approach called k-Fold CV, the training set is broken up into k smaller sets. For each of the k-parts, the following procedure is implemented: first, the model is trained using k-1 data pieces as training data. The resulting model is then checked for the remainder of the data (that is, it is used as a test set to calculate a performance measure, such as accuracy). As a result, the performance indicator, assessed during cross-validation, is the average value calculated over all iterations.

D. Anomalies detection

A matrix constructed from such vectors can be used as an input for a machine learning model in order to create a model for the detection of anomalies. As an example of the presentation of syslog data, consider a distributed system dataset (HDFS) set up in open access [https://github.com/logpai/loghub]. HDFS logs are compiled in [16] using a cluster with 203 nodes on the Amazon EC2 platform.

Fig. 5 shows an example matrix for HDFS log blocks.

```
0 0 6 3 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-100000266894974466
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000297946873432694
0 1 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000321454365365927
0 0 3 4 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000495798604346871
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001506908074013832
0 0 1 2 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001589563216071319
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001816331768897973
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 0 1 0 0 0 3 0 0 0  %%blk_-1002202903781990841
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1002269362670389066
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk -002451733127902119
```

Fig. 5. Fragment of the event account matrix

Marked data for classifier training is presented in Fig. 6. The "1" label indicates the presence of an anomaly in this block.

```
%blk_-1008051957326381849  0
%blk_-1008065969389674647  0
%blk_-1008294375794221679  0
%blk_-1009207079038502874  1
%blk_-1009611663961467477  0
%blk_-1009738950938953120  0
%blk -1009742189976312427 0
```

Fig. 6. Sample log data layout

The constructed model can be used to determine whether the new incoming log sequence is anomalous.

IV. DATA STRUCTURE

We will use the BlueGene/L supercomputer log as an experimental dataset presented in https://www.usenix.org/cfdr-data and in [15]. The triggered log fragment contains 100,000 lines. Each of which includes a timestamp, a device name, a message itself, and a label indicating whether the message contains information about a specific type of error. Tags about the type of errors or abnormal (emergency) states of the system of the fragment in question are given in Table II.

To illustrate the operation of the alarm state detection algorithm based on log data, consider a binary classification that takes into account only one type of anomaly. As an anomalous (emergency) state, we consider errors like "APPSEV", "KERNMC" [9]. The number of instances of these types is quite large compared with the others presented in Table II.

Taking into account that the number of specifically abnormal copies in the available database is quite small, it is necessary to use the cross-validation procedure (CV). In the KV approach, called k-fold, the training set is divided into k smaller sets. At the beginning, for each of the k-parts, the model is trained using k-1 parts. The classification algorithm is then tested for the remainder of the data. The result of the evaluation of efficiency, assessed in case of CV, is the average value calculated for all iterations. To implement the cross-validation process, we used the K-Folds class from the sklearn library of the model_selection section [https://www.usenix.org/cfdr-data] and the value of the parameter k = 5. In order to avoid the concentration of abnormal messages in one part of the data set, they are pre-randomly rearranged [5], [8]. Table III shows the numerical values of the sample volumes for each iteration separately used in the process of training and testing the classifier, taking into account the 5-block HF.

TABLE II. LABELS OF ERROR TYPES

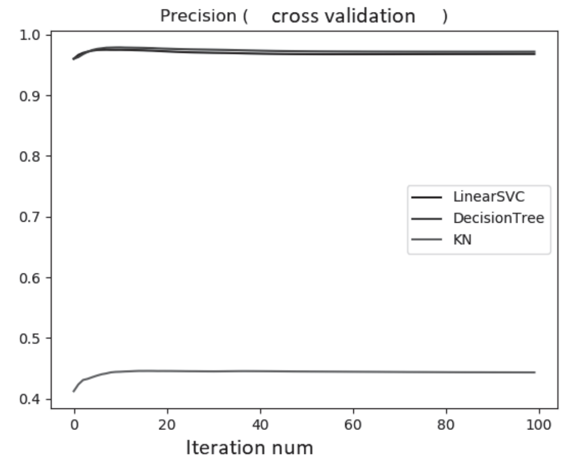
Error type label / count	Message example	Type of error or alarm conditions
APPSEV/ 7461	ciod: Error reading message prefix after LOGIN MESSAGE on CioStream[...]	App error
KERNTERM/ 512	rts: kernel terminated for reason 1004rts: bad message header: [...]	Kernel error
KERNMNTF/ 128	Lustre mount FAILED : bglio11 : block id : location	Kernel error
KERNMC/ 59	KERNEL FATAL machine check interrupt	Kernel error
KERNPOW/ 48	KERNEL FATAL Power deactivated: R05-M0-N4	Kernel error (power off)
R_DDR_STR/ 44	ddr: Unable to steer.*consider replacing the card	Card exchange
KERNRTSP/ 30	KERNEL FATAL rts panic! - stopping execution	Kernel error
KERNMICRO/ 16	KERNEL FATAL Microloader Assertion	Kernel error
KERNCON/ 4	KERNEL FATAL MailboxMonitor::serviceMailboxes() lib ido error: -1033 BGLERR IDO PKT TIMEOUT	Kernel error
Other/17		

TABLE III. DATASET CHARACTERISTICS FOR CROSS VALIDATION

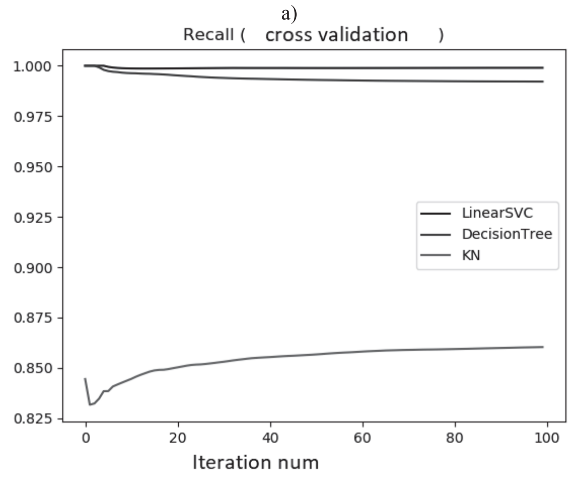
Anomaly/ Iteration num		Training sample		Test sample	
		Anomaly elements	Normal elements	Anomaly elements	Normal elements
KERNMC	1	45	1864	12	466
	2	45	1860	12	466
	3	48	1862	9	468
	4	46	1864	11	466
	5	44	1866	13	464
R_DDR_STR	1	35	1864	9	469
	2	34	1875	10	468
	3	37	1873	7	470
	4	37	1873	7	470
	5	33	1877	11	466
KERNRTSP	1	20	1889	8	470
	2	25	1884	3	475
	3	22	1888	6	471
	4	22	1888	6	471
	5	23	1887	5	472
APPSEV	1	52	1857	12	466
	2	49	1860	15	463
	3	53	1857	11	466
	4	49	1861	15	462
	5	53	1857	11	466

Fig. 7 shows the dependences of the change in the average value for the selected metrics for evaluating the quality of clustering during cross-validation.

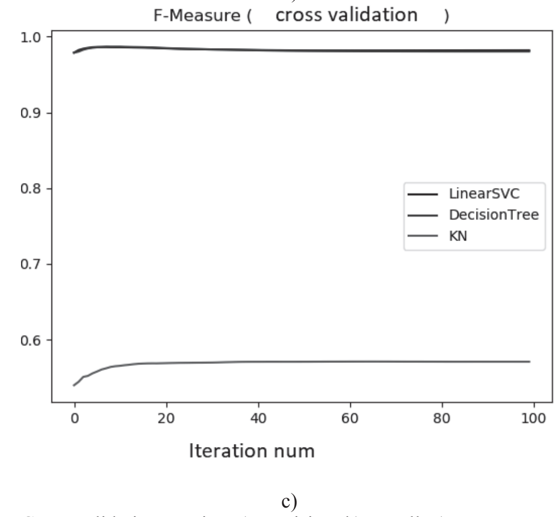
100 iterations of training and testing were conducted on a single data set using random permutation using the NumPy library's random.permutation () function developed by [http://www.numpy.org/]. In the absence of cross-validation, the ratio between the training and the test sample was 7/3. Comparison of the presented dependences shows that the CV allows to obtain characteristics that are more uniform and resistant to data oscillations.



a)



b)



c)

Fig. 7. Cross-validation metrics: a) Precision; b) Recall; c) F-measure

V. Metrics of classification algorithms

In machine learning tasks, various metrics are used to assess the effectiveness of the constructed models, such as: accuracy (precision), completeness (recall), F-measure (F-score), ROC-curves (Receiver Operating Characteristic curve - error curve), AUC- ROC and AUC-PR (Area Under Curve - area under the error curve and area along the pricison-recall curve)

After the classification, four types of results are possible: TP (True Positive - true positive), TN (True Negative - true negative), FP (False Positive - false positive), FN (False Negative - false negative).

The completeness (recall, sensitivity, sensitivity) shows the proportion of properly marked positive instances among all instances of the positive class:

$$recall = \frac{TP}{TP + FN}.$$

Precision is sensitive to the distribution of data while recall is not. Recall does not reflect how many samples are labeled as positive incorrectly, and accuracy does not give any information about how many positive samples are labeled incorrectly.

F-measure (F-score, F_β) combines the above two metrics - harmonic mean accuracy (precision) and recall:

$$F_\beta = \left(1 + \beta^2\right) \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}.$$

The F-measure reaches a maximum with recall and precision equal to one, and is close to zero if one of the arguments is close to zero.

ROC curve or error curve - a graph that allows to evaluate the quality of classification, displays the ratio between the sensitivity (TPR - True Positive Rate) of the algorithm and the fractions of negative objects that the algorithm predicted incorrectly (FPR - False Positive Rate) when the decision rule threshold is varied. The analysis of classifications using ROC is called ROC analysis.

$$FPR = \frac{FP}{FP + TN}.$$

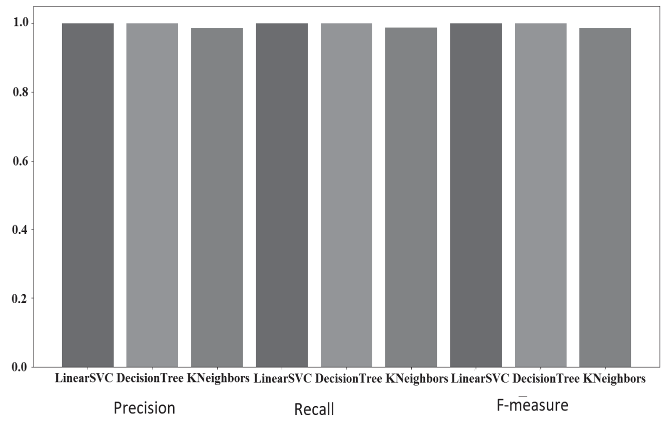
A quantitative interpretation of the ROC curve is given by the ROC-AUC index, the area under the ROC curve. In the ideal case, when the classifier does not make mistakes (FPR = 0, TPR = 1), the area under the curve is 1; if the classifier "guesses", then the AUC-ROC will tend to 0.5, since the classifier will produce the same amount of TP and FP. The area under the curve in this case shows the quality of the algorithm (more - better), besides this, the steepness of the curve itself is important - it is desirable to maximize TPR while minimizing FPR, which means that the curve should ideally tend to the point (0,1).

The AUC-ROC criterion is resistant to unbalanced classes and can be interpreted as the probability that a randomly selected positive object will be ranked higher by the classifier (will have a higher probability of being positive) than the randomly selected negative object.

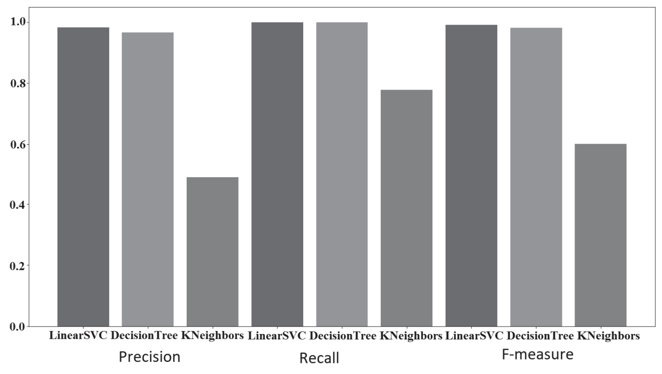
VI. Anomaly detection results

Consider the averaged results of the evaluation of three selected classification methods for the considered types of anomalous states: Decision Tree (DecisionTreeClassifier), Support Vector Vectors (LinearSVC), K-Nearest Neighbors (KNeighborsClassifier) [10], [11].

These methods are implemented in the library for Python - scikit-learn [http://scikit-learn.org/stable/index.html. The classification results are shown in Fig. 8.



a)



b)

Fig. 8. Estimation of error type: a) APPSEV; b) KERNMC

Based on the graphs obtained, it can be concluded that all three classifiers presented successfully cope with the task of classifying APPSEV anomalies. Presented in Fig. 8b, the averaged results of the evaluation of three selected classification methods for KERNMC type anomalies in the HF process show that the nearest neighbors method is significantly inferior to the two other SVC classification algorithms and the decision tree. This can be caused by both features of this type of anomaly, and a small number of anomalous samples in the sample at the training stage. In these samples, the number of abnormal copies did not exceed 1% of the total amount of data available for training.

As an example, Fig. 9 presents the ROC-curves for the binary classification of the KERNMC type anomaly.

As can be seen from the presented dependencies, the accuracy of detecting anomalous events lies within 0.92 ... 0.99 using the SVM and DecisionTree classification algorithms. The worst results belongs to classifier that implements the algorithm for K-nearest neighbors. For a more accurate comparison of the selected methods, a greater number of anomalous elements in the sample are needed.

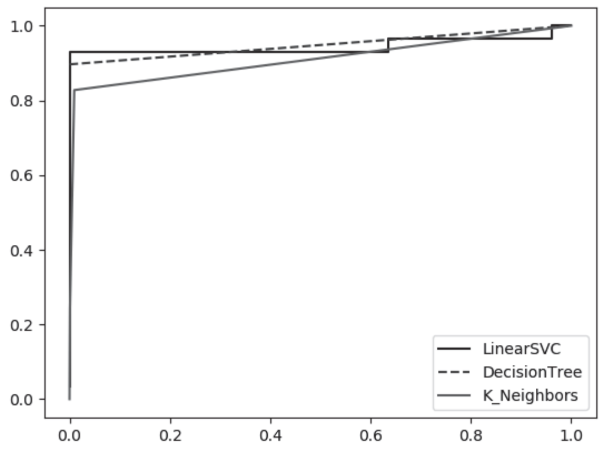


Fig. 9. ROC curves ("KERNMC" type error)

VII. Conclusion

For effective automatic detection and diagnosis of anomalous states based on unstructured system log data, a four-step algorithm based on the classification by machine learning methods has been proposed.

The central element of the considered algorithm is the formation of a matrix of attributes based on a group of templates in the parser after structuring the initial data of the system logs. The feature matrix was used to detect whether a new incoming log sequence is abnormal or not.

An analysis of the results obtained shows that the DecisionTree algorithm shows the best results in the binary classification of big data anomalies under consideration. The SVM algorithm is slightly inferior to it. The "nearest neighbors" algorithm does not work well with high dimensional data and therefore cannot be recommended for the problem of detecting anomalies based on unstructured data of system logs in which the dimension of vectors is comparable to the number of selected log event patterns.

References

[1] D. Zou, H. Qin, H. Jin, W. Qiang, Z. Han, X.Chen, «Improving Log-Based Fault Diagnosis by Log Classification». In Network and Parallel Computing, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 8707, pp. 446-458.

[2] Min Du, Feifei Li, Guineng Zheng, Vivek Srikumar, «DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning», in CCS '17 Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2017, pp.1285-1298.

[3] Daniel Dias Gonçalves, «Automatic Diagnosis of Security Events in Complex Infrastructures using Logs», Instituto Superior Tecnico, Universidade de Lisboa, May, 2015.

[4] Christophe Bertero, Matthieu Roy, Carla Sauvanaud, Gilles Tredan.Experience, «Log Mining Using Natural Language Processing and Application to Anomaly Detection», report on IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), 2017.

[5] Kristian Hunt, «Log Analysis for Failure Diagnosis and Workload Prediction in Cloud Computing», Stockholm, Sweden, 2016.

[6] Berkay Kicanaoglu, «Unsupervised Anomaly Detection in Unstructured Log-Data for Root-Cause-Analysis», Master's Degree Theses, Tampere University of Technology, 2015.

[7] Qiang Fu, Jian-Guang Lou, Yi Wang, Jiang Li, «Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis», in Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, December 06 - 09, 2009, pp. 149-158.

[8] R. Vaarandi, «A data clustering algorithm for mining patterns from event logs», in IPOM'03: Proc. of the 3rd Workshop on IP Operations and Management, 2003.

[9] Shilin He, Jieming Zhu, Pinjia He, Michael R. Lyu, «System Log Analysis for Anomaly Detection». Experience report on 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 23-27 Oct. 2016.

[10] Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu, «An Evaluation Study on Log Parsing and Its Use in Log Mining». In 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 28 June-1 July, 2016.

[11] Wei Xu, Ling Huang, Armando Fox, David Patterson, Michael I. Jordan, «Detecting Large-Scale System Problems by Mining Console Logs». In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, October 11 - 14, 2009.

[12] Tim Zwietasch, «Detecting Anomalies in System Log Files using Machine Learning Techniques». University of Stuttgart, 2014.

[13] O.I. Sheluhin, V.S. Ryabinin, M.A. Farmakovskiy, «Obnaruzhenie anomal'nyh sostoyanij komp'yuternyh sistem sredstvami intellektual'nogo analiza dannyh sistemnyh zhurnalov», Voprosy kiberbezopasnosti (in Russian), vol. №2(26), 2018.

[14] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, «An evaluation study on log parsing and its use in log mining», in Proc. of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2016.

[15] Jon Stearley, Adam Oliner, «What Supercomputers Say: A Study of Five System Logs», Stanford University Department of Computer Science Palo Alto, CA 94305, USA, Sandia National Laboratories Albuquerque, NM 87111 USA, 25-28 June, 2007.

[16] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan, «Large-scale system problem detection by mining console logs», in Proc. of the 22nd ACM Symposium on Operating Systems Principles (SOSP' 09), Big Sky, MT, October 2009.