Adv Syst Sci Appl 2019; **2**:23–32
http://ijassa.ipu.ru/index.php/ijassa/article/view/725

Machine Learning Algorithms for Automatic Anomalies Detection in Data Storage Systems Operation

Mikhail Hushchyn[1,2*], Andrey Sapronov[1,3], Andrey Ustyuzhanin[1,2]

[1]*National Research University Higher School of Economics, Myasnitskaya 20, 101000, Moscow, Russia*
[2]*Moscow Institute of Physics and Technology, Institutskiy per. 9, Dolgoprudny, Moscow Reg., 141700, Russia*
[3]*Joint Institute for Nuclear Research, Joliot-Curie 6, Dubna, Moscow Region, 141980, Russia*

Received April 9, 2019; Revised June 6, 2019; Published July 10, 2019

Abstract: Data storage reliability and availability play important role for a wide range of services and business processes. Manufacturers provide data storage systems that resistant to hardware and software failures but not for all cases. Well-timed detection of these failures helps to recover the system faster and prevent the failures before they occur. In this work a range of machine learning and time series analysis algorithms for failures detection for data storage systems is considered. The algorithms are applied and compared on the real system. Preliminary results show that binary classification methods demonstrate high failure detection and low false alarm rates. Time series prediction based approach shows similar results and outperforms one-class classification methods.

Keywords: machine learning, time series analysis, anomaly detection, data storage systems

1. INTRODUCTION

Modern data storage systems are built from hundreds of solid-state drives (SSDs) and hard disk drives (HDDs), networks and controllers to handle and process incoming data. A lot of data generated every day has to be properly saved for further usage. Storage systems have to provide high availability and reliability. Any failure in these systems will negatively affect any processes and applications that use the data and lead to significant revenue loss.

Any anomalous system behaviour can potentially be the result of a failure of its components and data loss. It requires from the storage system operators to explore it and, if necessary, take actions to resolve the problem. Timely anomaly detection allows to react on system failures faster decreasing shutdown time and preventing data loss. Anomalies can be the result of the system failures that have already happened or will happen in the near future. In the first case anomalies detection allows to reduce time delay for corrective actions. In the second case it gives time to take actions to prevent the failures and to reduce undesirable effects.

The paper has the following structure. Overview of related works is considered in Sec. 2. Data storage system that is used in this work for anomalies detection is described in Sec. 3. Discussion of different anomalies detection algorithms with examples and results is provided in Sec. 4. Finally, conclusions of this work are presented in Sec. 5.

*Corresponding author: hushchyn.mikhail@gmail.com

2. RELATED WORKS

There are a variety of anomaly detection methods [1, 2] applied in different domains of systems. These methods are the most widely used in computer networks to detect cyber attacks [3–7] and web-traffic anomalies [8–10]. There are a set of works where system logs of online service systems [11], supercomputers and distributed systems [12, 13] are analysed to detect various failures and anomalies. Several approaches were presented to predict failure of hard drives [14, 15] based on their SMART (Self-Monitoring, Analysis and Reporting Technology) data. Interesting results were demonstrated at the European Organization for Nuclear Research (CERN) where anomaly detection methods are used in data quality system for monitoring quality of data generated in high energy physics detectors [16].

The most promising methods of failures detection are based on machine learning and time series analysis algorithms. These approaches use clustering, one-class and binary classification algorithms for anomalies detection. Clustering methods [1, 2] suppose that normal and anomalous system behaviours form different clusters in some parameter space. The methods learn boundaries of these clusters and use them to distinguish different system states. This approach works well when the clusters are separable from each other. There are variety of clustering algorithms [17]: K-Means, Hierarchical Clustering, Gaussian Mixtures, DBSCAN and others.

Similarly, one-class methods learn system behaviour in the normal state. They suppose that normal states form dense clusters in system parameter space and recognize boundaries of these clusters. Everything that is beyond the boundaries is considered as an anomaly. The most used algorithms are Isolation Forest [18], Elliptical Envelope [19], One-class SVM [20], ASVDD [21] and WSVDD-CBA [22].

Binary classification [1, 2] is used when anomalies are known. It takes normal behaviour and anomalies as two classes and learns separation rule between them in parameters space. Then this rule is used to classify the system states. The most popular classifiers are Naive Bayes, Logistic Regression, SVM, Decision Tree, Random Forest, Gradient Boosting over Decision Trees and Artificial Neural Networks that are described in [17].

Time series analysis based methods [1, 2] are based on prediction of system parameters values in time. They use previous values of the parameters to predict the current one. Large deviations from the predictions indicate anomalies. There are a lot of predictive models for time series [23]: ETS models, Autoregression (AR) and Moving Average (MA) models, ARMA, ARIMA, Artificial Neural Networks and others. In this work several approaches of anomaly detection for data storage systems are considered.

3. TATLIN STORAGE DESCRIPTION

The goal of this work is to develop the algorithm of automatic failure detection for TATLIN [24] storage system. The system contains up to 4 storage controllers, Peripheral Component Interconnect Express (PCIe) fabric controller and up to 16 drive enclosures as it is shown in Fig. 3.1. Storage controllers are based on YADRO VESNIN [25] hardware platform. They provide user access to data, perform computations and run the TATLIN software. Each storage controller consists of 4 POWER8 Turismo SCM processors, 256 GB RAM, PCIe fabric connection adapter for data transfer and 2 Gigabit Ethernet (GbE) switches for internal network. Fabric controller integrates all components of the storage system and provides high data operation performance using fast SSDs. It has up to 96 SSDs with capacity of 2 TB each, 2 TB RAM cache, shared among storage controllers, 4 PCIe fabric adapters, Gigabit Ethernet switch for internal network and drive enclosure connection module. Drive enclosures host up to 96 Serial Attached SCSI (SAS) disks with capacity of 12 TB each to provide storage space for data that does not require high speed of operations, compared with SSD.

The storage reliability is provided by policies based on Reed-Solomon codes with minimal redundancy. It uses 25% redundancy that supports simultaneous failure of 2 drivers in 8 data

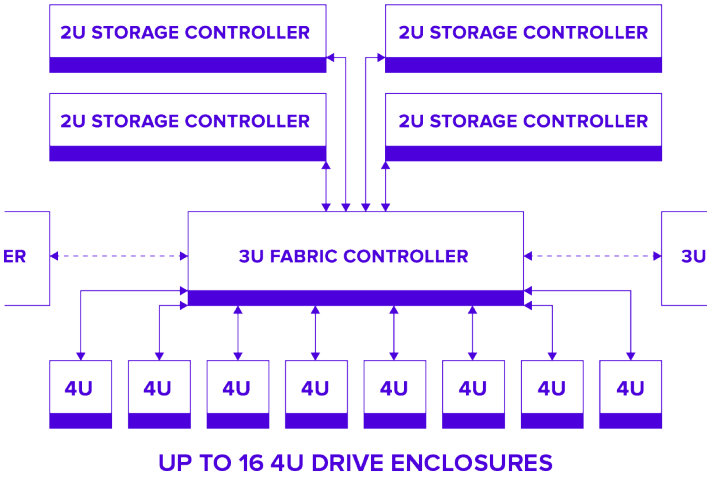


Fig. 3.1. TATLIN storage system.

+ 2 parity drives configuration. The fabric controllers provide shared access to the entire storage space for all storage controllers. Shared RAM cache gives hardware improvement by reducing CPU load in storage controllers for intensive data operations. The storage design excludes one point failure.

4. ANOMALIES DETECTION

4.1. Collected data

Behaviour of each component of the storage system is described by a set of parameters, that are used for anomalies detection. Examples of these parameters are CPUs and RAM usage, CPUs temperatures and fan speed for storage controllers, input and output traffics for network interfaces, SMART data for SSDs and SAS drives and others. Except hardware components information about logical data volumes inside storage pools is collected. Each volume is described by storage pool ID, capacity, state, read and write speed for each storage controller, number of read and write operations, average request processing time, average request queue size etc..

Several of known hardware failures for the all components are generated to test algorithms of anomalies detection. Parameters are measured approximately every 20 seconds and saved for the further analysis. For each measurement true state is recorded. There are two possible states: normal and failure.

Measurements of a component parameter are represented as a time series $\{(t_i, x_i)\}_{i=1}^{n}$, where x_i is a measured parameter value and t_i is a timestamp of the measurement. Each time series is divided into time bins with defined $width$. All measurements inside one bin are averaged. This aggregation helps to reduce noise of the measurements and to create time-regular observations of the parameters for further steps of the analysis. An example of time series after aggregation with $width = 60s$ is demonstrated in Fig. 4.2.

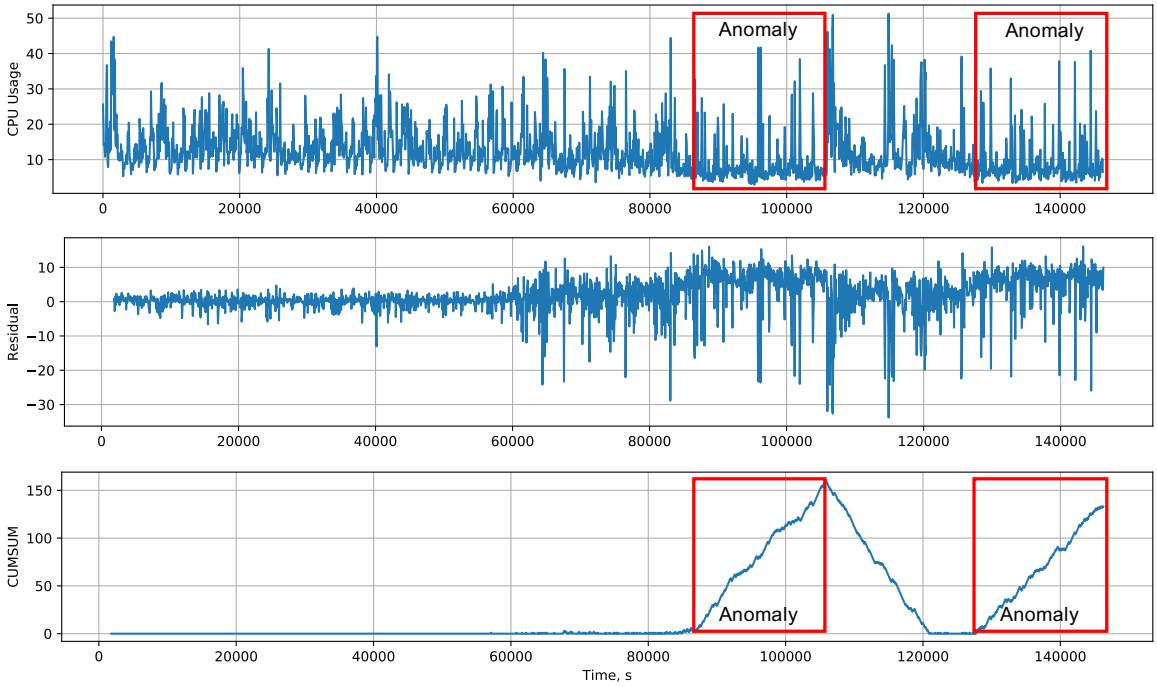


Fig. 4.2. (Top) Time series of percentage of CPU usage for one of the storage components of the storage system after aggregation. (Middle) Residual errors for the time series prediction. (Bottom) CUSUM test values for the anomalies detection.

4.2. Time series analysis

Consider time series analysis methods to detect anomalies. Time series analysis is used to predict parameter values of a component. To do this vector autoregression (VAR) model [23] is used. The principle of the model is following. Suppose a group of M time series is given: $\{y_{i1}, y_{i2}, ..., y_{it}\}_{i=1}^{M}$, where y_{it} is t-th measurement of i-th time series. Measurements of all time series with the same t are grouped into a vector $y_t = (y_{1t}\ y_{2t}\ \cdots\ y_{Mt})^T$. VAR model supposes that a vector of predicted parameters values $\hat{y}_t$ is a function of previous k measurements:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, ..., y_{t-k}) \tag{4.1}$$

where $\hat{y}_t$ is a predicted vector of parameters values; y_t is a vector of measured values of the parameters. For simplicity each time series has its own prediction model:

$$\hat{y}_{it} = f_i(y_{t-1}, y_{t-2}, ..., y_{t-k}) \tag{4.2}$$

where $\hat{y}_{it}$ is a predicted value for i-th time series. To approximate the functions f_i different machine learning algorithms [17] for the regression problem are used: Linear Regression model, Shallow Neural Networks, Random Forest and Gradient Boosting over Decision Trees Regressions. These algorithms are selected because they demonstrate high quality of the prediction and work well with multivariate time series. The models are trained with the following loss function:

$$L = \sum_t (\hat{y}_{it} - y_{it})^2 \tag{4.3}$$

Parameters of the models are optimized using Grid Search. The model with the lowest loss function value on the validation sample wins and used for anomalies detection. For each time series residual errors r_{it} are calculated:

$$r_{it} = \hat{y}_{it} - y_{it} \tag{4.4}$$

where r_{it} a residual error of a predicted value for i-th time series on t-th measurement. Example of a time series with two anomalies is shown in Fig. 4.2. It corresponds to the percentage of CPU usage for one of the storage controllers. The time series is aggregated as it is described in Sec. 4.1. To predict the current value $\hat{y}_{it}$ the previous 30 observations of this time series are used in this example. There are about 2500 observations that correspond to about 150000 seconds. The first 60000 seconds are used to train VAR model and the next 25000 seconds are used for the validation. Residual errors of the prediction are shown in Fig. 4.2. The figure shows larger residuals for the anomalies. To detect them the cumulative sum (CUSUM) [26] test is used. The test calculates new time series $\{T_t\}_{t=1}^{n}$, where the first value $T_1 = 0$ and the next values are estimated by the recurrent formula using residual errors:

$$T_t = \max(0, T_{t-1} + \epsilon_t) \tag{4.5}$$

$$\epsilon_t = \log \frac{f_0(r_{it})}{f_\infty(r_{it})} \tag{4.6}$$

where $f_\infty(r_{it})$ and $f_0(r_{it})$ are residual distributions for normal and anomalous states respectively. Normal distributions for the residuals are considered:

$$f_\infty(r_{it}) = \frac{1}{\sqrt{2\pi\sigma_\infty^2}} e^{-\frac{(r_{it}-r_\infty)^2}{2\sigma_\infty^2}} \tag{4.7}$$

$$f_0(r_{it}) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(r_{it}-r_0)^2}{2\sigma_0^2}} \tag{4.8}$$

where r_∞, σ_∞ are mean and standard deviation of residuals for normal states; r_0, σ_0 are mean and standard deviation of residuals for anomalous states; Residuals are not necessary have normal distributions. However, it was shown in [27] that even with strong deviations from the normal distribution CUSUM works well.

An anomaly is detected when the $T_t > T_{\text{alarm}}$. The distribution parameters and T_{alarm} are estimated during calibration on the validation sample without anomalies, where T_t has to take values close to 0 without growing trend. This approach allows to detect mean and variance changes of the residuals. An example of the CUSUM test is demonstrated in Fig. 4.2. The test value increases for anomalies and falls when the anomaly disappears. To avoid the test values relaxation after the anomaly T_t values can be reset right after the anomaly detected or by operator's demand.

4.3. One-class classification

One-class classification algorithms are used for anomalies and novelty detection in data. These algorithms learn boundaries of a normal class and everything that outside that boundary is considered as an anomaly. Isolation Forest [18] is one of the best one-class algorithms. Suppose a sample with N objects is given. Each object has D parameters. The idea of the Isolation Forest in the following:

Step 1 Select a subsample with $0 < n < N$ objects to build a new isolation tree.

Step 2 Randomly select a parameter.

Step 3 Randomly select an object from the current tree node. Use the selected parameter value of this object as a splitting rule for the node.

Step 4 Repeat steps 3 and 4 for the left and right children of the node. Stop the process when the termination conditions are satisfied.

Step 5 Repeat steps 1-4 to build forest of isolation trees.

According to [18], anomalies have shorter decision paths in isolation trees than normal objects. For an object x anomaly score $\hat{s}$ is estimated as:

$$s(x) = 2^{-\frac{E(h(x))}{c(N)}} \tag{4.9}$$

where $h(x)$ is a length of s decision path for the object in an isolation tree; $E(h(x))$ is an average length of the decision path for the object in the forest; $c(N)$ is an average length of the decision path for all N objects in the forest. The score takes values in $[0, 1]$. Objects with the score close to 1 are anomalous. On the other hand, objects with small score are normal. In case, when for all objects in a sample $s(x) \approx 0.5$, the sample has no anomalies.

Similar to the time series analysis approach, Isolation Forest uses vectors y_t of measured parameters values to estimate the anomaly score $\hat{s}_{it}$ for a measurement:

$$\hat{s}_{it} = f_i(y_t, y_{t-1}, ..., y_{t-k}) \tag{4.10}$$

An example of working of Isolation Forest is demonstrated in Fig. 4.3. The first 60000 seconds are used to train a classifier. The figure shows a slightly higher anomaly score for anomalies. All measurements with $\hat{s}_{it} \geq 0.5$ can be considered as anomalies. Results can be improved using CUSUM test similar to VAR models and will be considered further.

4.4. Binary classification

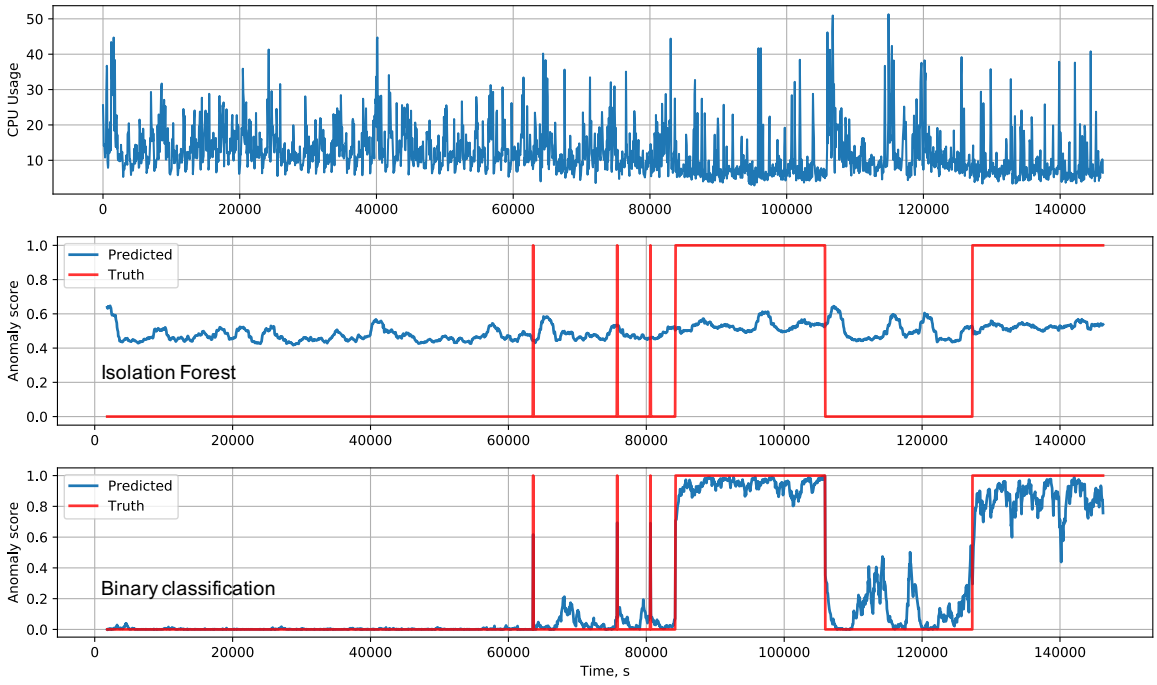


Fig. 4.3. (Top) Time series of percentage of CPU usage for one of the storage components of the storage system after aggregation. (Middle) Anomaly score predicted by Isolation Forest. (Bottom) Anomaly score predicted by Binary Classifier.

Binary classification is a powerful tool to detect known anomalies. Normal and anomalous measurements are considered as two classes. The classifier learns the separation surface

between them and is using it for anomalies detection for new measurements. Similarly to the VAR models, several binary classifiers [17] are considered: Logistic Regression, Naive Bayes Classifier, Shallow Neural Networks, Random Forest and Gradient Boosting over Decision Trees Classifiers. They are used to predict anomaly scores:

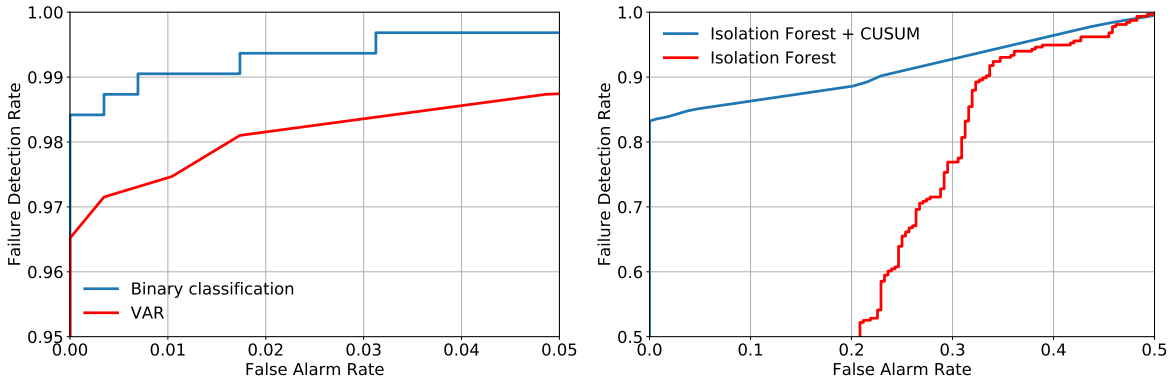
$$\hat{s}_{it} = f_i(y_t, y_{t-1}, ..., y_{t-k}) \tag{4.11}$$

where $\hat{s}_{it} \in [0, 1]$ is a predicted anomaly score. Parameters of the classifiers are optimized using Grid Search to minimize the logistic loss function:

$$L = -\sum_t (s_{it} \log\ \hat{s}_{it} + (1 - s_{it}) \log\ (1 - \hat{s}_{it})) \tag{4.12}$$

where $s_{it} \in \{0, 1\}$ is a true anomaly score for t-th measurement of i-th time series; The classifier with the lowest value of the loss function wins and is used for the anomalies detection. The example of the binary classification work is shown in Fig. 4.3. The first 110000 seconds are used to train the classifiers.

4.5. Models comparison



(a) (b)

Fig. 4.4. Dependencies of Failure Detection Rate from False Alarm Rate for different anomaly detection algorithms: (a) Binary classification and VAR; (b) Isolation Forest and Isolation Forest with CUSUM methods.

We collected data with induced hardware failures in different storage system components. A failure is considered as detected when at least one parameter of the system demonstrates anomalous behaviour. Time series for all parameters corresponded to one component are united into a group and used to estimate anomaly scores as described in previous sections. A measurement is anomalous when its anomaly score $\hat{s}_t > \tau$, where τ is a threshold value. For a set of τ values Failure Detection Rate (FDR) and False Alarm Rate (FAR) are calculated to measure the failures detection quality:

$$FDR(\tau) = \frac{1}{N_{\text{anomalies}}} \sum_i I[\hat{s}_i \geq \tau | s_i = 1] \tag{4.13}$$

$$FAR(\tau) = \frac{1}{N_{\text{normal}}} \sum_i I[\hat{s}_i \geq \tau | s_i = 0] \tag{4.14}$$

where $N_{\text{anomalies}}$ in the total number of anomalous measurements; N_{normal} is the total number of normal measurements. Dependencies of Failure Detection Rate from False Alarm Rate

	ROC AUC	FDR (FAR = 5%)	FDR (FAR = 1%)
Binary classification	0.999	0.997	0.991
VAR	0.997	0.987	0.975
Isolation Forest	0.755	0.0	0.0
Isolation Forest + CUSUM	0.956	0.852	0.836

Table 4.1. Quality metrics for the anomalies detection algorithms.

for different anomaly detection algorithms are shown in Fig. 4.4. These dependencies are also known as ROC-curves. Area under the ROC-curve (ROC AUC) is widely used in machine learning to measure quality of classification. ROC AUCs, Failure Detection Rates that correspond to False Alarm Rate values of 5% and 1% for different anomaly detection algorithms are presented in Tab. 4.1.

The results demonstrate that the binary classification method has the best quality. It detects larger than 99% of all anomalous states with False Alarm Rate below 5%. These results can be explained by the learning procedure of the method. During this procedure it recognizes separation surface between normal and anomalous states in the system parameters space. Then it uses this surface to distinguish the system states on new measurements.

VAR model shows similar results. It recognizes about 98% of all anomalous states with False Alarm Rate below 5%. However, unlike the Binary classification it does not use any information about anomalies. The model takes only the system parameters measurements for normal states to learn normal behaviour of the system. Any significant deviation from this behaviour considered as anomaly. This allows VAR to detect unknown anomalies.

Isolation Forest demonstrates high False Alarm Rates. Due to the noise in the parameters measurements Isolation Forest detects some normal measurements as anomalous and generates false alarms. It does not use any information about anomalies during the training procedure, that makes it harder to detect anomalies compared with binary classification. CUSUM test helps to reduce a number of false alarms and improves quality of the anomalies detection in a small FAR region as it is shown in Fig. 4.4.

5. CONCLUSION

Several approaches for automatic anomalies detection for data storage systems were tested and compared. These approaches are based on binary and one-class classification algorithms as well as on VAR models of time series analysis. The binary classification method has demonstrated the best detection quality. It allows to detect about 99% of anomalous states with False Alarm Rate of about 1%. The main limitation of this method is that it requires information about anomalies to learn how to detect them. This provides the high detection quality but makes it impossible to use the method for detection previously unseen anomalies.

To resolve this limitation two additional methods were considered. These methods do not use information about anomalies during their training procedures. They are able to detect previously unseen anomalies. Isolation Forest shows very small Failures Detection Rate in a region of False Alarm Rate below 5%. However, VAR model of time series analysis in combination with CUSUM test demonstrates promising results. It detects about 97% of anomalous states with False Alarm Rate is about 1%. The method learns normal behaviour of the system and detects any changes of it. In our future works we are going to continue developing methods of anomaly detection for data storage systems based on time series analysis methods. The goal is to investigate the methods that are able to recognize any anomalies without training procedure that uses information about these anomalies.

6. ACKNOWLEDGEMENTS

The research was carried out with the financial support of the Ministry of Science and Higher Education of Russian Federation within the framework of the Federal Target Program Research and Development in Priority Areas of the Development of the Scientific and Technological Complex of Russia for 2014-2020. Unique identifier RFMEFI58117X0023, agreement 14.581.21.0023 on 03.10.2017.

7. REFERENCE

REFERENCES

1. A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448 – 3470, 2007.
2. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.
3. M. Aiello, M. Mongelli, E. Cambiaso, and G. Papaleo, "Profiling dns tunneling attacks with pca and mutual information," *Logic Journal of IGPL*, vol. 24, p. jzw056, 09 2016.
4. E. Cambiaso, M. Aiello, M. Mongelli, and G. Papaleo, "Feature transformation and mutual information for dns tunneling analysis," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 957–959, July 2016.
5. M. Mongelli, M. Aiello, E. Cambiaso, and G. Papaleo, "Detection of dos attacks through fourier transform and mutual information," in *2015 IEEE International Conference on Communications (ICC)*, pp. 7204–7209, June 2015.
6. T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799 – 3821, 2007.
7. K. Limthong, "Real-time computer network anomaly detection using machine learning techniques," *Journal of Advances in Computer Networks*, pp. 1–5, 01 2013.
8. J. Hamidzadeh, M. Zabihimayvan, and R. Sadeghi, "Detection of web site visitors based on fuzzy rough sets," *Soft Computing*, 01 2017.
9. M. Zabihimayvan, R. Sadeghi, H. N. Rude, and D. Doran, "A soft computing approach for benign and malicious web robot detection," *Expert Syst. Appl.*, vol. 87, pp. 129–140, Nov. 2017.
10. M. Zabihi, M. V. Jahan, and J. Hamidzadeh, "A density based clustering approach for web robot detection," in *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 23–28, Oct 2014.
11. Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, (New York, NY, USA), pp. 102–111, ACM, 2016.
12. S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 207–218, Oct 2016.
13. W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, (New York, NY, USA), pp. 117–132, ACM, 2009.
14. C. A. C. Rincn, J. Pris, R. Vilalta, A. M. K. Cheng, and D. D. E. Long, "Disk failure prediction in heterogeneous environments," in *2017 International Symposium*

on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pp. 1–7, July 2017.
15. J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 383–394, June 2014.
16. M. Borisyak, F. Ratnikov, D. Derkach, and A. Ustyuzhanin, "Towards automation of data quality system for CERN CMS experiment," *Journal of Physics: Conference Series*, vol. 898, p. 092041, oct 2017.
17. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.
18. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, (Washington, DC, USA), pp. 413–422, IEEE Computer Society, 2008.
19. B. Koo, B. Shin, and T. Krijnen, "Employing outlier and novelty detection for checking the integrity of bim to ifc entity associations," in *ISARC 2017 - Proceedings of the 34th International Symposium on Automation and Robotics in Construction, 28 June - 1 July, Taipe, Taiwan*, pp. 14–21, International Association for Automation and Robotics in Construction I.A.A.R.C), 2017.
20. B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, pp. 1443–1471, July 2001.
21. R. Sadeghi and J. Hamidzadeh, "Automatic support vector data description," *Soft Comput.*, vol. 22, pp. 147–158, Jan. 2018.
22. J. Hamidzadeh, R. Sadeghi, and N. Namaei, "Weighted support vector data description based on chaotic bat algorithm," *Appl. Soft Comput.*, vol. 60, pp. 540–551, Nov. 2017.
23. R. J. Hyndman, G. Athanasopoulos, and OTexts.com, *Forecasting : principles and practice*. OTexts.com [Heathmont?, Victoria], print edition. ed., 2014 2014.
24. YADRO, "Tatlin storage description." `https://yadro.com/products/tatlin`, 2019.
25. YADRO, "Vesnin server description." `https://yadro.com/products/vesnin-tech`, 2019.
26. E. S. PAGE, "CONTINUOUS INSPECTION SCHEMES," *Biometrika*, vol. 41, pp. 100–115, 06 1954.
27. L. Koepcke, G. Ashida, and J. Kretzberg, "Single and multiple change point detection in spike trains: Comparison of different cusum methods," *Frontiers in Systems Neuroscience*, vol. 10, p. 51, 2016.

