DV2578 - Machine Learning Anomaly detection in Logs

Sri Sai Manoj Kommineni
19970612-8932
srko18@student.bth.se
Blekinge Institute of Technology
Karlskrona, Sweden.

I. Introduction

Many Internet services are run in big server clusters. In modern times, many companies are running these services on cloud computing environments provided by several companies such as Amazon, Microsoft and Google for easy scalability and maintenance costs primarily [20]. Designing, maintaining and deploying monitoring systems for such large and complex systems is difficult. When such services consisting of several hundreds of software components running on several computers malfunction, developers and operators need all the tools available to troubleshoot and diagnose problems in the system. There is one source of information that is available for almost every component of a software system that provides information about the original developers' ideas about noteworthy or unusual events for monitoring and problem detection [11]. Those are the typical, old console logs [39].

From the start of programming, logging has been used by developers [12], from mere print statements to complicated logging libraries, to record program variables, report runtime statistics, trace execution and print out full sentences to be read by a person, usually by the developer himself. However, modern large-scale services are usually developed by various developers, and the people going through the logs are part software integrators, part developers, part users and those who are given the duty to solve a problem. They probably are not the developers who have written the log statements. To make things more bad, current systems integrate external (most of the times open-source) components that are frequently updated, which may result in change of the log statements or the meanings of the log statements. Thereby extending the difficulty for the person reading the logs. Log files are typically huge files to analyse manually [8], [34] and not structured enough to analyse automatically. Hence, keywords such as "error" or "exception", are searched for using search tools, but this may not be effective or enough to identify the problem [38].

As abnormal or unusual log messages often lead to the cause of the problem, it is logical to consider log analysis as an anomaly detection problem in machine learning. However, it may not be the case that the presence, absence or frequency of a particular kind of message is enough to diagnose a problem. Usually, a problem manifests as an abnormality in relation to various types of log messages (correlations, relative frequencies, etc.). Therefore, instead of analysing the words in textual logs, we need to create features that precisely determine correlations among log messages, and perform anomaly detection on these features [38].

An outlier is defined as an observation which behaves differently from other observations such that we can suspect that the system is not going through its usual normal behaviour [10]. Outlier detection can be used for applications such as system fault detection and unexpected errors detection in databases [35]. Outlier detection is generally done on individual samples to predict outliers in the sample. In contrast, anomaly pattern detection on a data stream involves detecting the instance where the behaviour of the system is not usual and significantly different from the general behaviour, and pin pointing it [22], [35].

Anomaly detection in real-time streaming data has significant applications in a wide variety of industries such as finance, IT, security, medical, energy, e-commerce, and social media. An anomaly directly doesn't result in a problem. It just shows the change in behavioural pattern of the system. A change might be bad for the system, such as a bug in the software, or good, such as a system performance improvement after an update of a software component, which could be further examined after recognizing it. Anomaly detection in streaming applications is particularly challenging since there could be several data stream sources. Hence there is minimal scope for a human expert to intervene. Thus, there is a necessity for automation of detection of anomalies to identify and adapt to the changes and draw prediction. This project aims to detect the anomalies in logs using machine learning algorithms to identify the changes in the software systems, troubleshoot and recognize problems.

II. Dataset

HDFS (also known as Hadoop Distributed File System) is designed to run on clusters of servers for data processing.

This log set is generated by a HDFS system which was ran in a private cloud environment using benchmark workloads and manually labeled by hand-crafted rules to identify anomalies. The logs are traced using block IDs. A ground truth is assigned to each trace associated with a specific block Id. The dataset was create by [38] as part of their study to detect system run-time problems. The dataset contains 11,175,269 (about 11 million) lines of log statements which are labelled as anomalous or not anomalous.

III. Related Work

There have been various studies on anomaly detection for log analysis. The most significant methods used for anomaly detection can be categorized as statistical methods, supervised learning algorithms, unsupervised learning algorithms and deep learning algorithms.

Qiang Fu et al. [19] in this research paper have used statistical methods for log analysis and anomaly detection.

Peter Bodìk et al. [13] have used statistical methods to identify and classify crises (system problems), creating a knowledge base. They have used logistic regression algorithms for extracting features from logs obtained from several clusters of machines running in an enterprise level database. Using these features and applying statistical methods, crises were classified and identified.

Chen et al. [16] have extracted features from logs and used supervised learning for anomaly detection. Logistic regression has been used in specific.

Liang et al. [30] have used supervised algorithms along with other algorithms for anomaly detection using IBM BlueGene/L logs. They have used SVM (Support Vector Machine) algorithm for training and testing data. Several other clustering algorithms were also implemented for comparision.

Mostafa Farshchi et al. [18] have also implemented a supervised learning algorithm for anomaly detection in logs. They have used regression algorithms and statistical methods such as Pearson product-moment correlation [26] are used. Logs are collected from Amazon Ec2 computers [2] while running tools, Netflix Asgard [4] and CloudWatch [1].

Xu et al. [38] have applied Principal Component Analysis (PCA) algorithm, an unsupervised learning algorithm to detect anomalies in the logs. The logs used are from Hadoop Data File System (HDFS) [3] and Darkstar online game server [9]. The logs are parsed and features such as log message count vectors are extracted. These feature vectors are used for PCA and the results are visualized using decision trees.

Weixi Li [28] have used Natural Language Processing (NLP) techniques on logs for feature extraction. And they have used unsupervised learning algorithms K-means and DBScan in combination with Artificial Neural Network (ANN) for anomaly detection using the features.

Hamooni et al. [21] have applied unsupervised algorithm called LogMine which clusters the logs based on tokens generated by parsing logs using the general structure of the logs. This was an extension of study done by Ning et al. who have applied DBScan and a modification of DBScan algorithm for log analysis [33].

Lin et al. [31] have applied unsupervised learning algorithm for clustering, called LogCluster. Logs from Hadoop-based application are used for this study. The logs have been parsed and vectorized by applying NLP (Natural Language Processing) techniques such as IDF (Inverse Document Frequency) and Contrast-based Event Weighting. These features were used to cluster the algorithms.

Lou et al. [32] have parsed logs and used unsupervised learning algorithm called in variant mining to cluster the logs and extract features such as log count and log message groups. Statistical methods are applied on these features to identify anomalies.

Shilin He et al. [24] have reviewed and evaluated six log-based anomaly detection methods, three supervised methods, namely logistic regression [13], decision tree [16] and SVM (Support Vector Machine) [30], and three unsupervised methods, namely Log Clustering [31], PCA (Principal Component Analysis) [38] and In-variants Mining [32], based on F-measure and run time. HDFS data set used in [38] and BGL dataset used in [34]. In this study, invariant mining outperformed other algorithms in terms of F-measure.

Brier et al. [14] introduced a dynamic rule based anomaly detection method for identifying security breaches in log records after appplying unsupervised learning algorithms for classifying the logs.

Vinay Kumar et al. [37] have applied a deep learning algorithm called S-LSTM (Stacked Long short-term memory) to detect anomalies in logs. The model resulted in an accuracy of 99.6 on the dataset used.

Silin He et al. [23] have applied NLP (Natural Language Processing) technique IDF (Inverse Document Frequency) weighting extract features from the logs. And an usupervised learning algorithm based on hierarchical agglomerative clustering called Log3C was applied, performing better than PCA (Principal Component Analysis) done by Xu et al. [38] and invariants mining done by Lou et al. [32].

Zhang et al. [40] applied a new algorithm LOGROBUST based on natural language processing (NLP) techniques and deep learning algorithms. The log words were vectorized using Fast text algorithm [27] which also takes in semantics of the words and then features are extracted from the logs. A Bidirectional LSTM [25] algorithm was appled to detect anomalies in logs. This model has outperformed models proposed in [13] (Logistic Regression), [30] (Support Vector Machine), [32] (Invariant Mining) and [38] (Principal Component Analysis).

Li et al. [29] has also used NLP techniques and deep learning algorithms for anomaly detection. The logs have been parsed to extract features and then logs have been vectorized by using sentence embeddings using Bidirectional Encoder Representation from Transformers (BERT) algorithms [17]. These features are used to train an Attention Based Bidirectional LSTM (Long short term memory) algorithm which is a modification of Bidirectional LSTM network [25].

IV. Algorithms

As seen in the studies [29], [40], the deep learning methods have produced good results, but these methods are computationally expensive and also require the developer to label several logs which is labour intensive since logs are ubiquitous. Hence, we haven chosen to experiment using Unsupervised learning algorithms in combination with Natural Language Processing techniques. Cyntihia et al. have done an analysis on algorithms Local Outlier Factor(LOF), Isolation Forest and Logistic regression for credit card fault recognition [15] and found LOF and Isolation Forest to be performing better. In this project, unsupervised learning in combination with NLP techniques are explored as NLP techniques are giving good results. The algorithms K-means, DB-Scan, LOF and Isolation Forest have been implemented for this project.

- GloVe, Global vectors for word representations is an unsupervised machine learning algorithm for words to be represented as vectors. This is a method for obtaining pre-trained word embeddings. It considers global count statistics not only the local information [36].
  It learns on co-occurence matrix and trains word vectors, so that the co-occurance ratio can be predicted. In comparision with word2vec , GloVe is very similar, despite their starting points. All the log messages are removed of log variables extracted and log text is converted to word embeddings which vectors of size 100.
- Cosine Distances are calculated between embeddings and a dictionary of types of log messages (all the variables are removed and only text is extracted) is maintained. Since all the variables are extracted from the log message, similar types of log message would be very similar. Hence, looking for cosine similarity would be suffice, rather than using traditional clustering methods, which look for the ideal centroid for the clusters and are computationally costly. 41 types of log messages were identified in the dataset using this method.
- K-means algorithm is applied with one cluster and the data points outside certain threshold value are classified as anomalies. K means was applied for 300 iterations and was run 10 times with random initial points as centroids of the data. Euclidean distance metric was used for measuring distance. The K-means algorithm was fitted with data size of 100,000 lines.
  The euclidean distance from the center for the data points can be observed in 1. For more clarity, a scatter plot was also plotted to get more clarity for setting the threshold values. From figures 1 and 2, the threshold was set at 250,000.
- DB Scan is applied with epsilon value of 4 and minimum samples = 2, the distance metric used is Euclidean distance. A data size of 100,000 lines of logs was fit into the algorithm. Clusters with value '-1' are the outliers in the dataset, since they could not cluster into any cluster group.

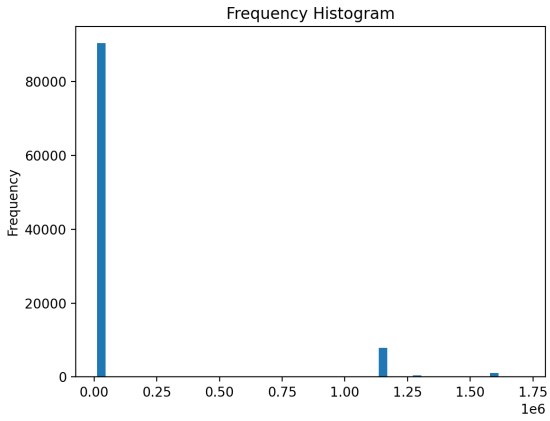


Fig. 1. *Distribution of data over distance, x-axis: frequency of datapoints, y-axis: distance from the centre of cluster*

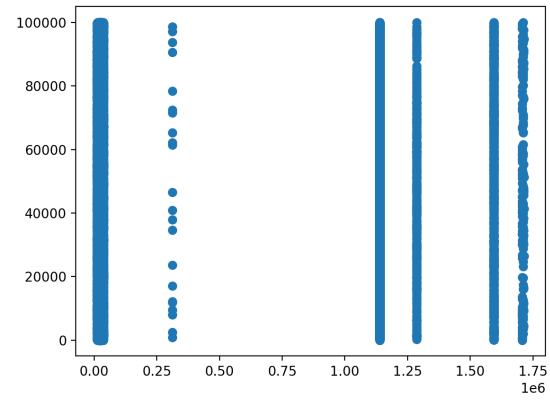


Fig. 2. *Distribution of data over distance, x-axis: data point index value ranging from 0 to 100,000, y-axis: distance from the centre of cluster*

- Isolation Forest was applied on a data size of 100,000 lines of logs, with all the features pre-processed. It has a contamination value of 0.1. It resulted in average anomaly scores as seen in figure 3. In the figure, low average anomaly scores signify more anomalous behaviours of data.
- Local Outlier Factor identifies anomalies by observing the neighbours of each data point. It compares the density of the total data sample with the local density of each data point. If the local density of each data point is lower, it is classified as an outlier. KNN is applied to determine the density for Local Outlier Factor algorithm, n=20 is given as parameter for this algorithm and 100,000 lines of logs are fit into the model.

V. Implementation

Apache Spark open source software [3], scikit-learn [6], scipy [7] and spark-nlp [5] have been used for implementation in python programming language. The anomaly detection

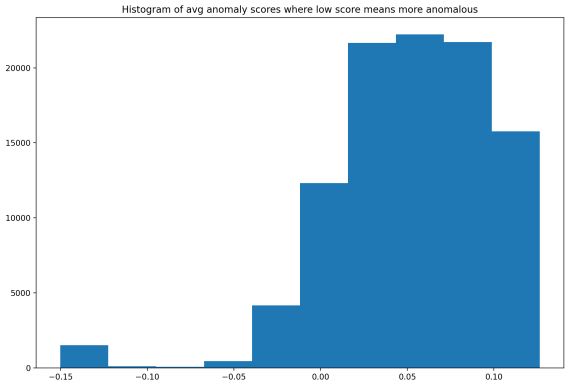


Fig. 3. *Isolation Forest Anomaly Scores, x-axis: frequency of datapoints, y-axis: anomaly score of data point*

process involves log parsing, feature extraction and applying algorithm which can be seen in figure 4.

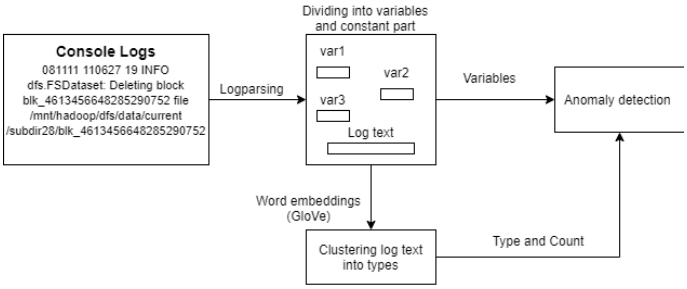


Fig. 4. *Anomaly detection*

A. Log parsing

The logs are parsed using regular expressions from the console logs. The logs are parsed initially based on the general log structure and then more variables are extracted based on observation. The parsing consists of date, time, blk_id, location, log type, log message type (seen as log message info), IP address, port number and log message. The log statements after parsing can be seen in the figure 5.

The log statement can be divided into two parts, variable part and the constant part.

1) Variable part of the logs:: The variable part of the log statement includes all values that can be unique for the log such as the date, time, blk_id, location, IP address and port number. These values change based on the log statement, hence they are called log variables. The following are extracted using regular expressions.

1) Date: It is in the format DDMMYY (Date-Month-Year) and is first 6 characters of logs.
2) Time: It is in the format HHMMSS (Hours-Minutes-Secs) and is the next 6 characters of the log.
3) Block Id: It starts with "blk_id" and ends with a sequence of characters (numbers and symbols).
4) Location: The location is a numeric value.
5) Log Type: Log type tells us the category of message the log is conveying such as INFO, WARN, etc..
6) IP: IP address mentioned in log statements.
7) Port: Port numbers of the IP address used by the application.

2) Constant part of the logs:: The constant part of the logs contains the log message, other than the variables in the log message part, the log message remains constant for other log statements too. It is extracted using regular expressions. For example, consider the following two log messages:
"Sent block blk_-9848064687919812344 of size 21312 from /11.350.10.16"
"Sent block blk_-9848064687919812345 of size 21312 from /12.240.9.12"
We can seen that excluding the numerical part (variable part), the remaining remains constant for both the log messages. So, it is known as the constant part.

B. Feature Extraction

Since all machine learning algorithms require numerical values for predictions, the numerical features are extracted from the parsed log statements shown in figure 5. Log statements are highly correlated [38] based on different variables. So, the following are the features extracted:

1) **Block_Id:** A dictionary of block_ids is made and each block_id was assigned with a value in range of 1 to 1008. Any new blk_id will be added to the dictionary and an incremental value will be added.
2) **Location:** Location value is taken as it is.
3) **IP:** A dictionary of IP addresses is created and each IP is in range of 1 to 106 in the current dataset. Any new IP address will be added to the dictionary and an incremental value will be added.
4) **Port:** Port number is taken as it is.
5) **Time:** The date and time variables are taken in seconds starting from 0 for the first log.
6) **Log Type**: A dictionary of Log Types is made. The data set contains only one value, more values will be added to the dictionary when needed.
7) **Log Message Info:** A dictionary of Log Message Info is made. The current data set has only six values, more values will be added to the dictionary when needed.
8) **Log Message Type:** The log message from the log parsing process is stripped of numeral values, thus removing all variable values in the log message and extracting the constant part. Then, word embeddings are used to vectorize the log statement.

   GloVe: A pretrained model of GloVe was trained with 930,000 lines of logs from the data set. This helps improve the co-occurrence matrix of the GloVe model

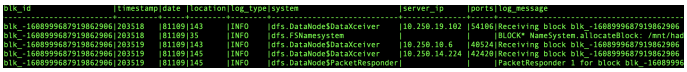
to vectorize the data better.

Classifying based on cosine similarity: A dictionary of log message types is made, new values are added if no log message type, which is stored as vector using GloVe embeddings, in the dictionary has a cosine similarity value more than 0.95, i.e., the log messages should be matching more than 95%. Clustering is done using cosine similarities.

A total of 41 different log message types are found in the dataset and are stored as a dictionary.

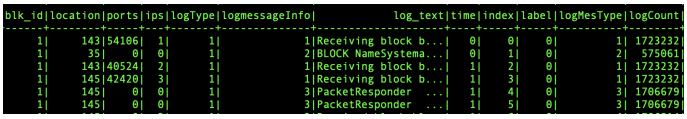
9) **Log Count:**A dictionary for log count value for each log message type is maintained. This feature helps in recording frequency of the log message.

All the above feature values can be seen in figure 6, extracted from values shown in figure 5.



blk_id	timestamp	date	location	log_type	system	server_ip	ports	log_message
blk_-160899968791986298	6	203518	81109	143	INFO	dfs.DataNode$DataXceiver	10.250.19.102	54106	Receiving block blk_-160899968791986298
blk_-160899968791986298	6	203518	81109	35	INFO	dfs.FSNamesystem			BLOCK* NameSystem.allocateBlock: /mnt/had
blk_-160899968791986298	6	203519	81109	143	INFO	dfs.DataNode$DataXceiver	10.250.10.6	40524	Receiving block blk_-160899968791986298
blk_-160899968791986298	6	203519	81109	145	INFO	dfs.DataNode$DataXceiver	10.250.14.224	42420	Receiving block blk_-160899968791986298
blk_-160899968791986298	6	203519	81109	145	INFO	dfs.DataNode$PacketResponder			PacketResponder 1 for block blk_-160899968

Fig. 5. *Parsed Logs*



blk_id	location	ports	ips	logType	logmessageInfo	log_text	time	index	label	logMesType	logCount
1	143	54106	1	1	1	Receiving block b...	0	0	0	1	1723232
1	35	0	0	1	2	BLOCK NameSystema...	0	1	0	2	575061
1	143	40524	2	1	1	Receiving block b...	1	2	0	1	1723232
1	145	42420	3	1	1	Receiving block b...	1	3	0	1	1723232
1	145	0	0	1	3	PacketResponder ...	1	4	0	3	1706679
1	145	0	0	1	3	PacketResponder ...	1	5	0	3	1706679

Fig. 6. *Features Extracted*

C. Anomaly Detection

Anomalies are detected by using algorithms as discussed in previous section.

VI. Result

As seen in figure 7, the performance metrics for K-means algorithm has increased as more data is being processed but it got flattened very quickly. As seen in figure 8, the precision increased while all other metrics got worsened as more data is processed while implementing DB-Scan. As seen in figure 9, the performance metrics of Isolation Forest algorithm got flattened very quickly as more data is processed. As seen in figure 10, the performance metrics of Local Outlier Factor (LOF) algorithm got flattened very quickly as more data is processed.

In table I and figure 11, we can see the performance metrics from different algorithms compared. Run time, accuracy, precision and F1-score are the metrics measured.

VII. Conclusion

We can conclude that Local Outlier Factor clearly outperformed other algorithms based on the performance metrics.

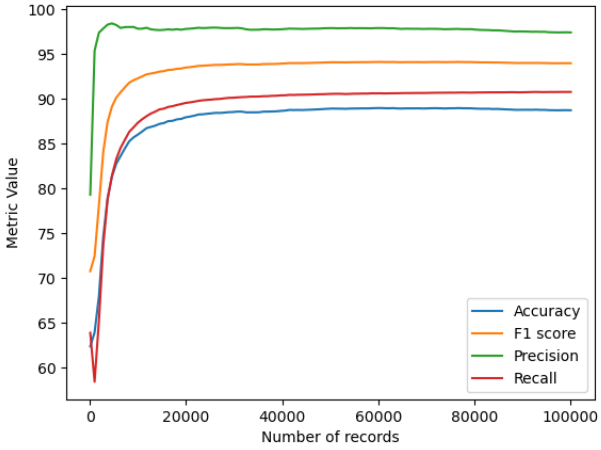


Fig. 7. *K-means metrics*

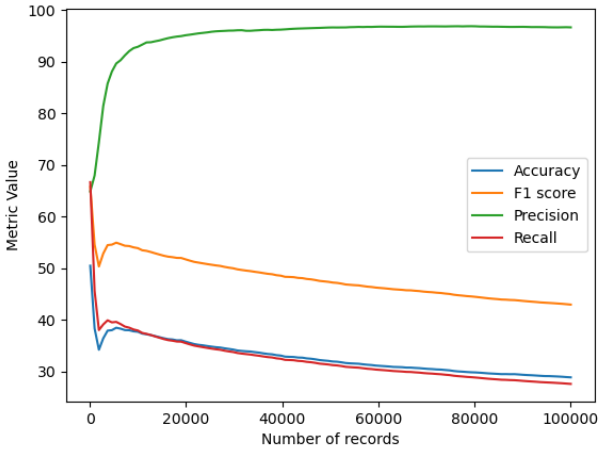


Fig. 8. *DBScan metrics*

However, better metrics are required to make the model usable in real life. K-means performed better than DBScan and Isolation Forest.

TABLE I
Metrics Comparision

	K-means	DBScan	LOF	Isol. Forest
Run time	4.1814	0.2890	5.0800	3.0400
Precision	0.9741	0.9661	0.9718	0.0701
Recall	0.9078	0.2763	0.9768	0.2298
Accuracy	0.8873	0.2890	0.9500	0.8835
F1 Score	0.9398	0.4297	0.9743	0.1074

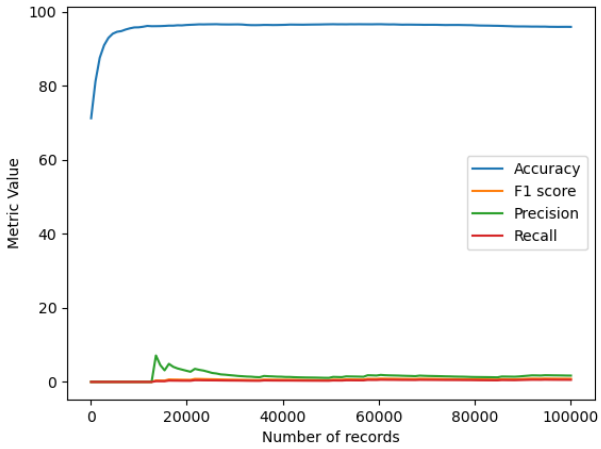


Fig. 9. *Isolation Forest metrics*

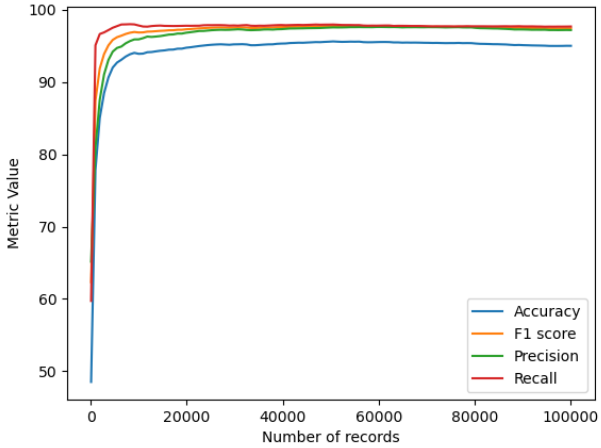


Fig. 10. *Local Outlier Factor metrics*

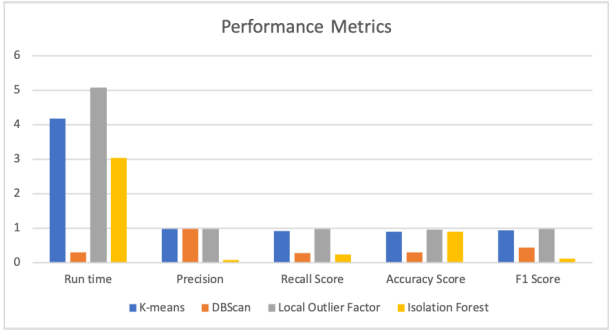


Fig. 11. *Histogram of metrics*

REFERENCES

[1] Amazon CloudWatch - Application and Infrastructure Monitoring.

[2] Amazon EC2.

[3] Apache Hadoop.

[4] Asgard.

[5] John Snow Labs | NLP & AI in Healthcare.

[6] scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation.

[7] SciPy.org — SciPy.org.

[8] Understanding customer problem troubleshooting from storage system logs. FAST '09, USA.

[9] DarkstarProject/darkstar, November 2020. original-date: 2014-01-11T01:31:17Z.

[10] Charu C Aggarwal. An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer, 2017.

[11] Devika Ajith. A survey on anomaly detection methods for system log data. *International Journal of Science and Research (IJSR)*, 8:23, 07 2019.

[12] RW Barnitz and GE Terwilliger. Application of data-logging and programming techniques to steel mill processes. *IRE Transactions on Industrial Electronics*, pages 24–32, 1959.

[13] Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B. Woodard, and Hans Andersen. Fingerprinting the datacenter: Automated classification of performance crises. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, page 111–124, New York, NY, USA, 2010. Association for Computing Machinery.

[14] J. Breier and J. Branišová. A dynamic rule creation based anomaly detection method for identifying security breaches in log records. *Wireless Personal Communications*, 94(3):497–511, 2017. cited By 14.

[15] P. Caroline Cynthia and S. Thomas George. An outlier detection approach on credit card fraud detection using machine learning: A comparative analysis on supervised and unsupervised learning. *Advances in Intelligent Systems and Computing*, 1167:125–135, 2021.

[16] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pages 36–43, 2004.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[18] M. Farshchi, J. Schneider, I. Weber, and J. Grundy. Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis. In *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, pages 24–34, 2015.

[19] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 ninth IEEE international conference on data mining*, pages 149–158. IEEE, 2009.

[20] Robert L Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[21] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1573–1582, New York, NY, USA, 2016. Association for Computing Machinery.

[22] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[23] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, page 60–70, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 207–218. IEEE, 2016.

[25] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv:1508.01991 [cs]*, August 2015. arXiv: 1508.01991.

[26] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, Melbourne, Australia, 2nd edition edition, 2018.

[27] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. FastText.zip: Compressing text classification models. *arXiv:1612.03651 [cs]*, December 2016. arXiv: 1612.03651.

[28] Weixi Li. Automatic log analysis using machine learning: awesome automatic log analysis version 2.0, 2013.

[29] X. Li, P. Chen, L. Jing, Z. He, and G. Yu. Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 92–103, 2020.

[30] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 583–588, 2007.

[31] Q. Lin, H. Zhang, J. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 102–111, 2016.

[32] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li. Mining invariants from console logs for system problem detection. pages 231–244, 2019. cited By 38.

[33] Xia Ning, Geoff Jiang, Haifeng Chen, and Kenji Yoshihira. 1HLAer: a System for Heterogeneous Log Analysis. 2014.

[34] Adam Oliner and Jon Stearley. What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 575–584. IEEE, 2007.

[35] Cheong Hee Park. Outlier and anomaly pattern detection on data streams. *The Journal of Supercomputing*, 75(9):6118–6128, 2019.

[36] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. *arXiv preprint arXiv:1708.03390*, 2017.

[37] R. Vinayakumar, K. P. Soman, and P. Poornachandran. Long short-term memory based operation log anomaly detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 236–242, 2017.

[38] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, 2009.

[39] Wei Xu, Ling Huang, Armando Fox, David A Patterson, and Michael I Jordan. Mining console logs for large-scale system problem detection. *SysML*, 8:4–4, 2008.

[40] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati, Furao Shen, and Dongmei Zhang. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, page 807–817, New York, NY, USA, 2019. Association for Computing Machinery.