Comparative study of machine learning algorithms for anomaly detection in Cloud infrastructure

[1]Mr. Satish Kumbhar, [2]Nikita Paranjape, [3]Rasika Bhave, [4]Akshay Lahoti
[1]Assistant Professor, Department of Computer Engineering, College of Engineering, Pune, Maharashtra 411005
[2]B.Tech Student, Department of Computer Engineering, College of Engineering, Pune, Maharashtra 411005
[3]B.Tech Student, Department of Computer Engineering, College of Engineering, Pune, Maharashtra 411005
[4]B.Tech Student, Department of Computer Engineering, College of Engineering, Pune, Maharashtra 411005
Email: [1]*ssk.comp@coep.ac.in*, [4]*lahotias14.comp@coep.ac.in*
Contact: [1]+919860574798, [2]+918390023697, [3]+919607210054, [4]+919987985460

***Abstract*—**Cloud is one of the emerging technologies in the field of computer science and is extremely popular because of its use of elastic resources to provide optimized, cost-effective and on-demand services. As technology started to grow in scale and complexity, the need for automated anomaly detection and monitoring system has become important. Inappropriate exploitation of Cloud resources can often lead to faults like crashing of VMs, decreased efficiency of cloud system etc. thereby leading to violations of the Service Level Agreement (SLA). These faults are often preceded by anomalies in the behavior of the VMs. Hence, the anomalies can be used as indicators of faults which potentially violate the SLAs. We have created a system that will monitor the VMs, detect anomalies and warn the system administrator before any problem escalates. We present in this paper a comparative study of various machine learning algorithms used for detecting anomalies in cloud

Keywords- *Anomaly detection; Machine Learning; Cloud infrastructure*

*****

I. INTRODUCTION

Anomaly detection is the process of finding the patterns in a dataset whose behavior is abnormal or unexpected. Such unexpected behavior is also termed as an anomaly or an outlier [9]. The anomalies cannot always be categorized as an attack but it can be a surprising, previously unknown behavior which can escalate into a bigger problem. This can potentially violate the SLA [1].

Machine Learning automatically trains a model from historic data without being explicitly programmed and improves its accuracy with experience. This model predicts values of data in near future. In anomaly detection, the machine learning algorithm is supplied initially with the data. This data is a mixture of normal and anomalous data that has been labelled. These algorithms create models that are then used to predict whether the cloud system is in anomalous state or not [8].

This paper is organized as follows. Section II explains the setting up of the infrastructure. Section III describes the methods used to simulate anomalies on the VMs. Section IV explains how the data was collected and preprocessed. Section V describes the machine learning algorithms used. Section VI will put forth the results we got on the testing data and lastly Section VII describes the future scope of this research.

II. SETTING UP INFRASTRUCTURE

A. *Cloud*

OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service, whereby virtual servers and other resources are made available to customers [2]. It can be installed as mentioned in [3]. For testing, we installed OpenStack on a machine which has 4 processors, 32 GB of RAM.

B. *Metrics monitoring tool*

We have installed Zabbix, enterprise open monitoring software for networks and applications, to monitor our virtual machines and collect data. This entails installing a Zabbix server on one virtual machine, and a Zabbix agent on all other virtual machines that we wish to monitor. The agent monitors the virtual machine it is installed on and periodically sends data to the server (data is sent per minute). The server machine needs to be configured to accept data from all the agents and agents need to be configured to send data to the correct server [4].

III. SIMULATING ANOMALIES

In order to find out how the VM behaves when an anomaly occurs, we simulated some anomalies. There are several tools to simulate such anomalies. These tools can be used as commands on the terminal and do not require a user interface. We used the stress tool [5]. By varying different command line parameters passed to these tools, we were able to customize the severity of each of the following anomalies [6].

A. *CPU Utilization*

It creates number of process specified by the user and consumes the CPU resource for a specified amount of time.

Syntax: stress -c <no_of_processes> -t <time_to_run>

Example: stress -c 4 -t 180s

B. *Available Memory*

Available memory is the amount of RAM that is free.

Syntax: stress -m --vm-bytes <Number_of_units_of_memory> <Unit_of_memory> -t <time_for_which_thge_process_is_run>s

Example: stress -m --vm-bytes 300M -t 180s

C. *Disk IO wait Time*

It is the amount of time CPU has spends doing read/write operations. High disk IO time indicates that CPU has spent more time doing disk input output instead of using that time for computations.

Syntax: stress -d <Number_of_processes> --hdd-bytes <Number_of_bytes><unit_of_bytes> -t <time for which the process is run>s

Example: stress -d 2 --hdd-bytes 512M -t 180s

IV. Data Collection and Preprocessing

We collected data that was anomalous and normal both so that not only will our machine learning algorithms be able to identify exactly which anomaly took place, but will also be able to distinguish between a normal and an anomalous state. We have collected equal amounts of data of each anomalous state and the normal state. This is to ensure that the machine learning algorithm will have equal opportunity to identify patterns of each anomaly and normal data. If the normal data largely outnumbered anomalous data, there is a possibility that the model classifies all data as normal data and still gets good accuracy on the testing dataset. This data is collected as each metric per minute. Some data metrics were dependent upon others. Eg average CPU load per 15 minutes was dependent on CPU load per minute. We compiled only the independent metrics. A full list of metrics given as an input to the various machine algorithms are in table (1). We kept a track of when we had simulated any anomalies. A python script took that data as input and labelled the collected data. Normal data was class 1, memory overuse was class 2, swap space utilization - class 3, CPU overload - class 4, excess disk I/O - class 5. This data is stored as a csv file and can be used to train any model.

TABLE I. List of Extracted parameters

Avg CPU load/ minute	Free swap space	Incoming traffic on virbr0
CPU steal time	CPU idle time	CPU user time
Outgoing traffic on virbr0	No of processes run / minute	CPU computation time
No. of CPU interrupts/min	Free inodes on root	Incoming traffic on virbr0-nic
Used disk space on root	CPU interrupt handling time	Available memory size
Outgoing traffic on virbr0-nic	Incoming traffic on ens32	CPU soft interrupt handling time
No. of CPU switches/min	Free disk space on root	Outgoing traffic on ens32
CPU system time	CPU I/o wait time	Class assigned

V. Machine Learning Algorithms

A. *logistic Regression*

It is a logistic function to convert the output of a linear regression into classes. Higher linearity between the feature and the target variable contributes to better performance of the Logistic Regression model. In the multiclass case as ours, the training algorithm uses the one-vs-rest (OvR) scheme. The logistic regression class of scikit-learn implements regularized logistic regression using the ‘liblinear’ library, ‘newton-cg’, ‘sag’ and ‘lbfgs’ solvers. It can handle both dense and sparse input [7].

B. *Linear discriminant analysis*

Linear Discriminant Analysis can be used to perform supervised dimensionality reduction, by projecting the input data to a linear subspace consisting of the directions which maximize the separation between classes. The dimension of the output is necessarily less than the number of classes, so this is in general a rather strong dimensionality reduction, and only makes senses in a multiclass setting [10]. We have used LDA to obtain the 10 most important parameters that best separate the classes [7].

C. *K-nearest neighbor*

K-NN is a simple, non-parametric lazy learning technique used to classify data based on similarities in distance metrics. The class of a new datapoint is dependent on classes of K of its nearest datapoints.It does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point. We have used k = 5 for classification, owing to trial and error to get the best possible accuracy for our dataset [7].

D. *Decision Tree*

CART (Classification and Regression Trees) supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node. Scikit-learn use an optimized version of the CART algorithm. New data points are classified according to its feature values at each level of the tree [7].

E. *Naïve Bayes*

There are many cases where the statistical dependencies or the causal relationships between system variables exist. It can be difficult to precisely express the probabilistic relationships among these variables. To take advantage of this structural relationship between the random variables of a problem, a probabilistic graph model called Naïve Bayesian Networks (NB) can be used. GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian [7] [11].

F. *Support Vector Machine*

These are a set of related supervised learning methods used for classification and regression. Support Vector Machine (SVM) is widely applied to the field of pattern recognition. SVC and NuSVC implement the "one-against-one" approach for multi- class classification. If n_class is the number of classes, then n_class * (n_class - 1) / 2 classifiers are constructed and each one trains data from two classes [7].

G. *Neural Network*

It is a set of interconnected nodes designed to imitate the functioning of the human brain. Each node has a weighted connection to all other nodes in neighboring layers. Individual nodes take the input received from connected nodes and use the weights together with a simple function to compute output values. The user specifies the number of hidden layers as well as the number of nodes within a specific hidden layer. In a multiclass classification problem, the output layer of the neural network contains several nodes. The Multilayer Perceptions (MLP) neural networks have been very successful in a variety of applications and producing more accurate results than other existing computational learning models [6]. They are capable of approximating to random accuracy, any continuous function as long as they contain enough hidden units [7].

VI. Experimental Results

We collected data over one month. This contained about 1000 data points of every anomaly and normal Table 2: Performance of Algorithms on test data scenarios. We have split this data into two chunks. 80% of the data is used to train the models and 20% to test them. This was done with 10 fold cross validation.

We used the following metrics to assess the different methods

Accuracy = TP / TP + TN

Precision = TP / TP + FP

Recall = TP / TP + FN

F measure = (2 * Precision * Recall) / (Precision + Recall)

TABLE II. Performance of Algorithm on Test data

Machine Learning Algorithm	Accuracy	Std. deviation
Logistic Regression	0.964298	0.002801
Linear Discriminant Analysis	0.982502	0.003136
K Neighbor Classifier	0.995051	0.001322
Decision Tree Classifier	0.999647	0.000433
Naive Bayes	0.762373	0.012196
Support Vector Machines	0.878913	0.010643
Neural Network	0.868010	0.061652

TABLE III. Comparison of the performance of the precision, recall and F1-score

Algorithm	Precision	Recall	Fscore
Logistic Regression	0.97	0.97	0.97
Linear Discriminant Analysis	1.00	1.00	1.00
K Neighbor Classifier	1.00	1.00	1.00
Decision Tree Classifier	1.00	1.00	1.00
Naive Bayes	0.97	0.77	0.82
Support Vector Machines	0.93	0.88	0.88
Neural Network	0.85	0.89	0.86

We observed that we get really good results from logistic regression, K nearest neighbor classifier and decision tree classifier. SVM and Neural network results vary as we vary the arguments that we pass to the classifier.

VII. Future Scope

Here we have used only 7 popular algorithms for comparisons. However there may be better algorithms that can give higher accuracy. One can also think of combining two or more algorithms for better results, for e.g. LDA can be used for dimensionality reduction [10] and the reduced number of parameters can be supplied as features in a neural network. Reducing the parameters and extracting the essential ones will decrease the amount of time required to train the algorithms. Another area of study is the value of arguments passed to the algorithms, which can change their performance values.

References

[1] G. Aceto, A. Botta, W. de Donato and A. Pescapè, "Cloud monitoring: Definitions, issues and future directions," 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET), Paris, France, 2012, pp. 63-67.

[2] https://en.wikipedia.org/wiki/OpenStack

[3] https://docs.openstack.org/devstack/latest/

[4] https://tecadmin.net/install-zabbix-agent-on-ubuntu-and-debian/

[5] https://people.seas.harvard.edu/~apw/stress/

[6] https://linux.die.net/man/1/stress

[7] http://scikit-learn.org/stable/documentation.html

[8] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao and F. Liu, "Evaluating machine learning algorithms for anomaly detection in clouds," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 2716-2721.

[9] G. Aceto, A. Botta, W. de Donato and A. Pescapè, "Cloud monitoring: Definitions, issues and future directions," 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET), Paris, France, 2012, pp. 63-67.

[10] F. Song, D. Mei and H. Li, "Feature Selection Based on Linear Discriminant Analysis," 2010 International Conference on Intelligent System Design and Engineering Application, Changsha, 2010, pp. 746-749.

[11] F. Doelitzscher, M. Knahl, C. Reich and N. Clarke, "Anomaly Detection in IaaS Clouds," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, 2013, pp. 387-394

[12] Dr. Chinthagunta Mukundha," Anomaly Detection in Cloud Based Networks and Security Measures in Cloud Date Storage Applications", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064