Univariate and Multivariate Gaussian Models for Anomaly Detection in Multi Tenant Distributed Systems

Prof. Pravin Ramdas Patil[1], Dr. Geetanjali Kale[2]
Assistant Professor[1]
Head of Department and Associate Professor[2]
Department of Computer Engineering, SCTR's Pune Institute of Computer Technology,
Pune, Maharashtra, India[1, 2]

***Abstract*—Due to the flaws in shared memory, settings, and network access, distributed systems on a network always have been susceptible to cyber intrusions. Co-users on the same server give attackers the chance to monitor the activity of many other users and launch an attack when those users' security is at risk. Building completely secure network topologies immune from risks and assaults has traditionally been the goal. It is also hard to create an architecture that is 100 percent safe due to its open-ended nature. The precise parameters and infrastructure design whereby the strike is instantiated are a constant which can always be detected regardless of the sort of attack. This work now have the chance to simulate any abnormality and subsequent attack possibilities using network parameter values thanks to the increased usage of algorithms for machine learning and data-gathering tools. This work proposes a Gaussian model to forecast the likelihood of an attack occurring depending on certain system parameters. This work model a univariate and a multivariate Gaussian model on the training dataset. This work makes use of various threshold values to predict whether the data point is an inlier or an outlier. This research examines accuracies for various threshold values. An important challenge in an anomaly detection situation is class imbalance. As long as this work just utilizes training data, a class imbalance is not a problem. Our data-driven results show that combining machine learning with Gaussian-based models might be a useful tool for analyzing network intrusions. Although more steps are being made to boost digital space security, machine learning algorithms may be utilized to examine any abnormal behavior that is left uncontrolled.**

Keywords*—Multi-tenant distributed system; anomaly detection; outlier detection; machine learning; Gaussian model

I. INTRODUCTION

One of today's most demanding technologies is cloud computing. Cloud computing offers an infinite quantity of IT facilities to deliver amazing computing speed, but on the flip side, it has serious security problems with public clouds for multitenant cloud environments. Most government and commercial companies are compromising with the limited IT resources and performance from existing resources since they are not migrating their sensitive and private data over the public cloud due to security concerns. The aforementioned problems will be solved by finding a way to protect private space over public clouds.

Multiple clients can use the services provided by multi-tenant distributed systems. As a result, each client has access to the activity of the others. By being one of the clients of such a system and taking advantage of such surveillance, attackers can launch assaults against one or more other tenants of the system [1]. To stop any entity in the system from suffering damage, such an attack must be promptly detected [2]. The scourge of attacks in such distributed systems has been a hot topic among researchers despite improvements in cyber security measures. Although cyber security protections have improved, experts continue to focus on the problem of intrusions in such distributed multi-tenant systems. Multiple tenants can cohabit on the same network thanks to multi-tenant distributed systems (MTDS). The MTDS service provider does not inquire about the tenant's motivations when they request co-allocation. This situation presents a chance for renters with bad intentions to observe and collect confidential information about the target occupants. [4] Because the attacker tenant has access to sensitive information, the tenant may prepare an attack that has a greater likelihood of success. [3]

There have already been several attempts to use a variety of techniques to identify the existence of intrusions in distributed applications. [5], [6] Earlier, the emphasis was on applying statistical techniques to compute specific function values, but more recently, cutting-edge approaches including deep learning have been applied. In this regard, artificial neural networks have been investigated.

Although rule-based engines were used to identify assaults, they frequently fall short of spotting any newly discovered threats. Transfer learning may be helpful in this situation, but there is no guarantee that the variables of the source work and the destination job are identical, which has been a significant obstacle to its application. [7]

This work suggests a Gaussian-based classifier strategy in this research for identifying the potential for intrusions in a multi-tenant distributed system to identify inliers and outliers. This work defines a threshold value. This work also looks at the accuracy of different threshold values. Authors are thankful to Patil and Ingale [8] for providing us with the dataset.

Section II of paper includes literature survey of research work done in the area of network attack detection. It explores

Machine learning algorithms used to detect network attacks and to improve cyber security. Section III describes experimentation performed to create and collect dataset. As network attack is not a continuously or regularly occurring event hence lesser number of attacks are performed to create dataset. This dataset includes majority non attack instances and very few attack instances. This section includes statistical and graphical representation of collected dataset. Section IV explains creation of univariate and multivariate Gaussian models for anomaly detection and respective models performance analysis. Section V contains conclusion of research work done.

II. Related Work

Network attack detection has historically made heavy use of signature-based detection. This approach uses an analysis of an attack's "signature," or distinctive qualities, to foretell potential hazards in the future [9]. Methods to discover the best attack signatures were suggested by Hilker et al. [10]. Han et al. [11] advocated crafting network traffic using several attributes. The system cannot identify any new attacks that were previously undiscovered owing to a lack of knowledge about them, which is a significant problem with this technique. Additionally, each new effort to locate signatures requires human labor in addition to time.

Additionally, there have been initiatives to employ machine learning algorithms in this field. Algorithms based on supervised learning have traditionally been used to identify network attacks. [12] For assault detection, Zseby et al. favoured the use of selecting features and subsequent mapping [13]. Evolutionary algorithms were used by Rafique et al. [14] to evaluate the effectiveness of classifying malware. The chance of assault is extremely low, it should be highlighted, therefore a model may get away with forecasting all data as non-negative and yet show good accuracy, making the entire process exceedingly costly.

Prior strategies likewise emphasized the application of boosting techniques and feature reduction in transfer learning. TrAdaBoost was introduced by Dai et al. [15] and reweights the data from the positive and negative classes to give the uncommon examples that indicate attacks more weight in the outcome. TCA-transfer component analysis was used by Pan et al. to feature project the domains closer to one another in the common space [16]. HeMap is a technique created by Shi et al. [17] that projects features using linear transformations. Patil and Ingale [8] tackled the class imbalance problem and used an ensemble based meta classifier to detect anomaly.

The detection of assaults has also been done using model-based methods. This strategy falls under the category of transfer learning and makes the crucial assumption that the source task and the target task share at least some parameters or model priors. Bekerman demonstrated how transfer learning may help increase the resilience of malware detection in uncharted situations. [17].

A noteworthy finding in all of these prior methods was that the stark class disparity seen in network assaults was hardly discussed. Additionally, due to this imbalance, effectiveness of other measures should also be discussed in order to shed light on the results that were produced. We model a Gaussian model on the training dataset. The advantage of this method is that class imbalance does not cause any hindrance.

Research community is contributing towards improving cyber security and security of multi-tenant distributed systems. Despite being all these efforts, attackers are successfully able to place compromised or virtual machine having anomaly to reside with target virtual machine. This leads to increase in the probability of having successful attack on a target virtual machine. Detection of new types of attack possible because of co-residence, co-location and co-tenant of attacker virtual machine with a target virtual machine is still remains a challenge to researchers. Univariate and Multivariate Gaussian models are created to detect network attacks. Performance analysis of individual models created is performed.

III. Dataset Preparation

A. Dataset Collection

Dataset has been collected by Patil and Ingale [8] by using Netdata, a programme for real-time performance monitoring that creates system logs. The logs have been collected across 28 files. This work combines all the files into a single dataset for easy handling. The dataset consists of 4986 inliers instances and 60 outlier instances with 63 columns. All the columns names are noted in Table I.

B. Dataset Preparation

Contributors dropped the column ‘anomaly score’ as it is generated by the software. Authors also separate ‘label’ from the remaining dataset. Authors also drop the columns whose standard deviation is less than 0.3 but also store the original dataset. Contributors are left with 36 columns in the remaining dataset. This work plot some of the important columns as a categorical plot except anomaly score from Fig. 1 to 12. Authors don’t have to worry about class imbalance because model on the training dataset while training is done.

TABLE I. Column Names

Sr. No.	Column name
1	app_cpu_sys_netdata
2	app_cpu_sys_apps.plugin
3	app_cpu_sys_tc-qos-helper
4	app_cpu_sys_go.d.plugin
5	app_cpu_sys_logs
6	app_cpu_sys_ssh
7	app_cpu_sys_system
8	app_cpu_sys_kernel
9	app_cpu_sys_other
10	app_cpu_usr_netdata
11	app_cpu_usr_apps.plugin
12	app_cpu_usr_tc-qos-helper
13	app_cpu_usr_go.d.plugin
14	app_cpu_usr_logs
15	app_cpu_usr_ssh
16	app_cpu_usr_system
17	app_cpu_usr_kernel
18	app_cpu_usr_other
19	app_mem_netdata
20	app_mem_apps.plugin
21	app_mem_tc-qos-helper
22	app_mem_go.d.plugin
23	app_mem_ssh

24	app_mem_cron
25	app_mem_system
26	app_mem_other
27	app_mem_X
28	app_soc_ssh
29	app_soc_system
30	app_soc_other
31	app_soc_X
32	sda_writes
33	ops_sda_writes
34	utilization
35	packets_received
36	packets_sent
37	packets_delivered
38	socket_used
39	udp_packets_received
40	udp_packets_sent
41	avail
42	Dirty
43	Writeback
44	sys_cpu_softirq
45	sys_cpu_user
46	sys_cpu_system
47	sys_cpu_iowait
48	switches
49	interrupts
50	io_out
51	ip_received
52	ip_sent
53	net_received
54	net_sent
55	pgio_out
56	Running
57	Free
58	Used
59	Cached
60	Buffers
61	Uptime
62	Label
63	anomaly_score

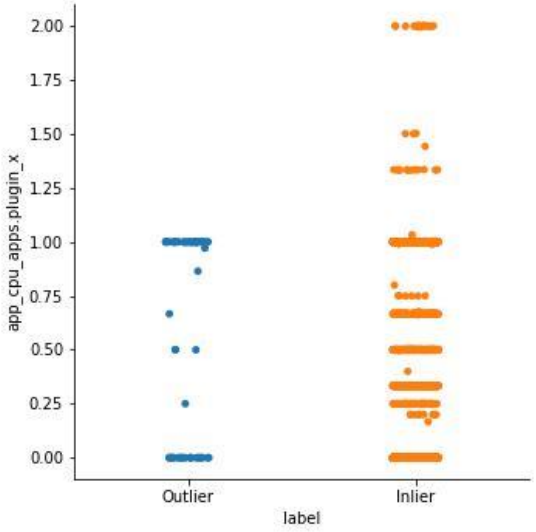

Fig. 1. Categorical plot of app_cpu_apps.plugin_x.

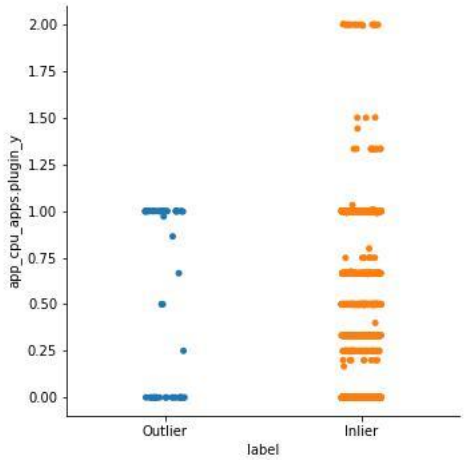

Fig. 2. Categorical plot of app_cpu_apps.plugin_y.

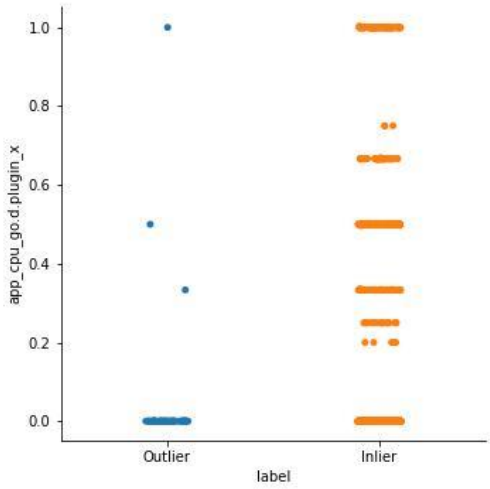

Fig. 3. Categorical plot of app_cpu_go.plugin_x.

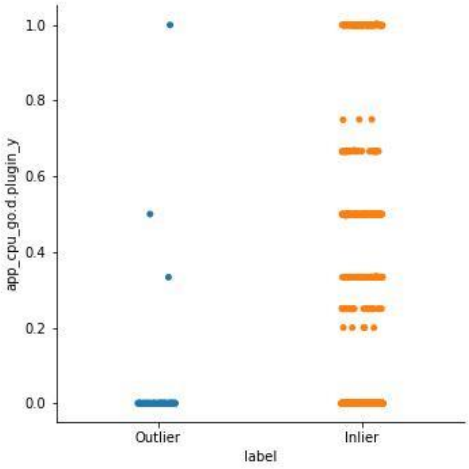

Fig. 4. Categorical plot of app_cpu_go.plugin_y.

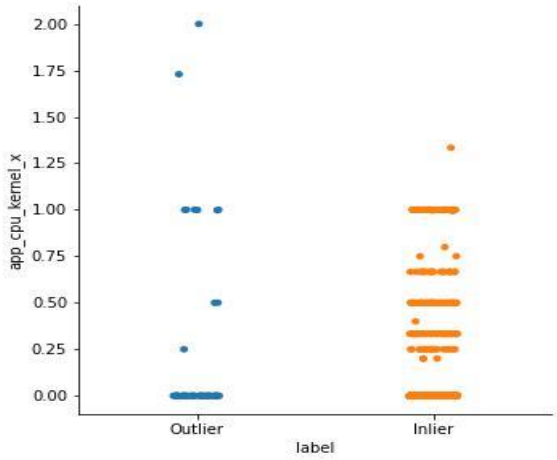

Fig. 5. Categorical plot of app_cpu_kernel_x.

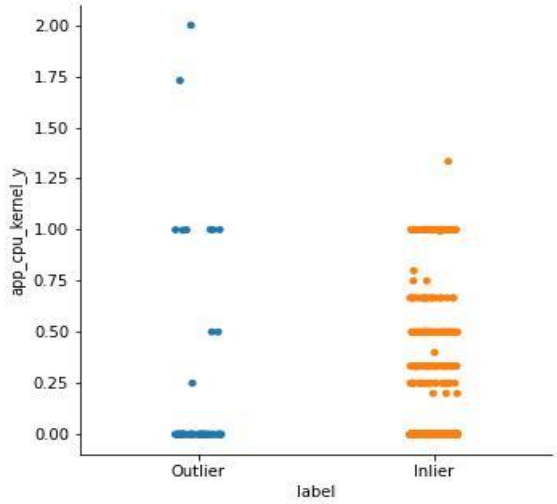

Fig. 6. Categorical plot of app_cpu_kernel_y.

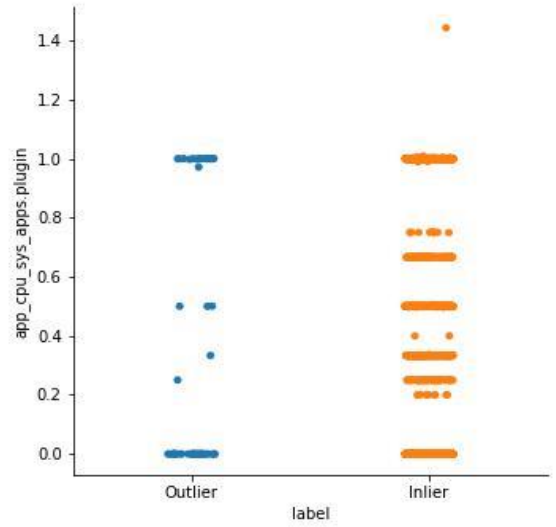

Fig. 7. Categorical plot of app_cpu_sys_apps.plugin.

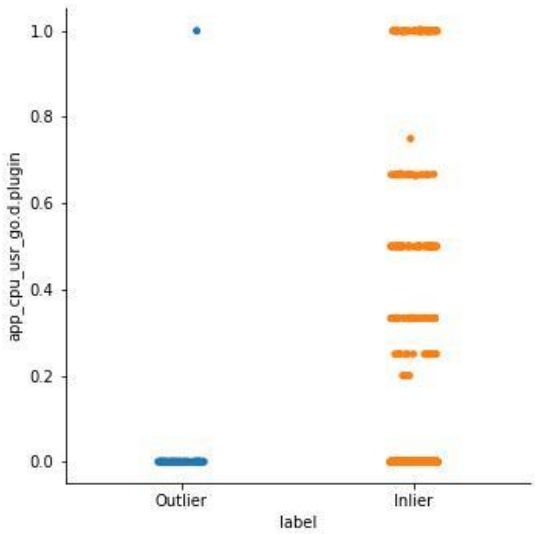

Fig. 8. Categorical plot of app_cpu_usr_go.d.plugin.

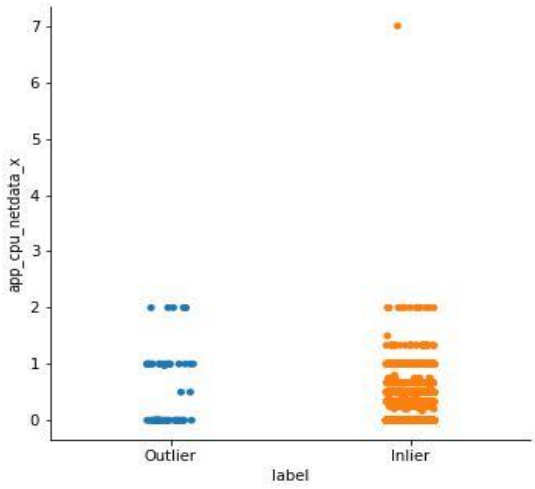

Fig. 9. Categorical plot of app_cpu_netdata_x.

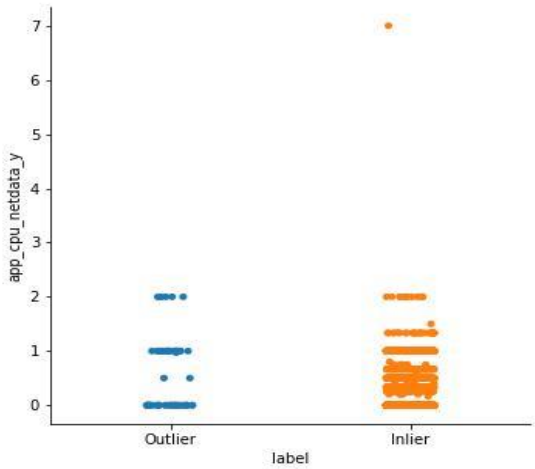

Fig. 10. Categorical plot of app_cpu_kernel_y.

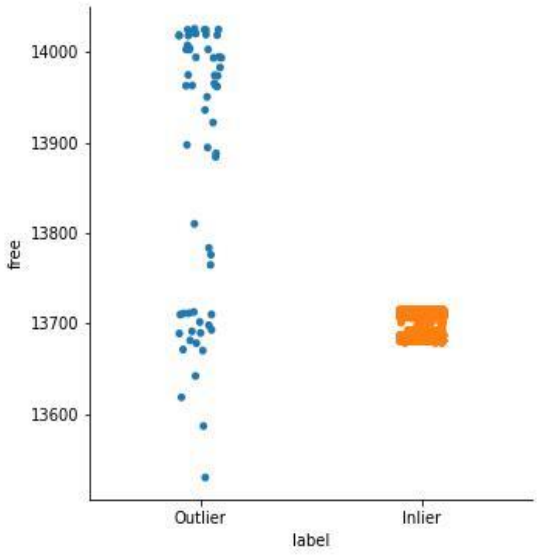

Fig. 11. Categorical plot of free.

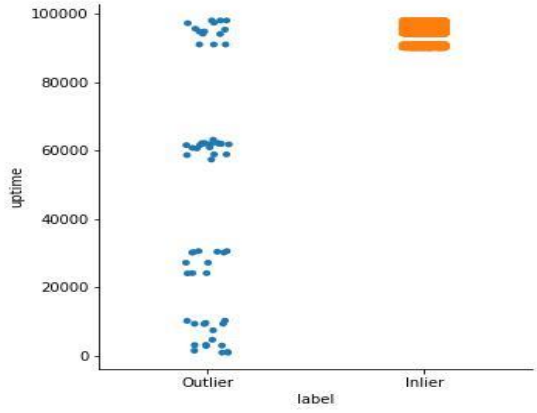

Fig. 12. Categorical plot of uptime.

Authors then standardize the dataset as there is need to perform PCA on it. PCA is applied by a keeping 98% variance. After applying PCA, Dataset have 38 columns in the original dataset and 18 columns in the dataset on which columns were removed having standard deviation less than 0.3. This work plots the first two components of the new dataset on a 2D axis as shown in Fig. 13. Authors also perform PCA on the original dataset.

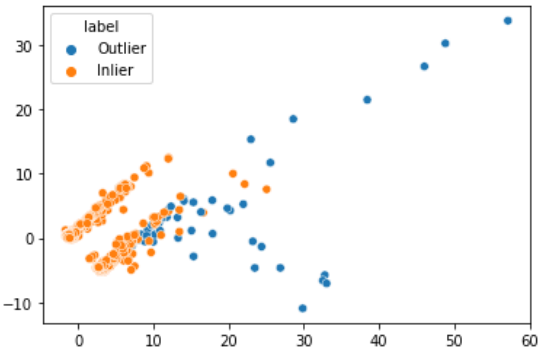

Fig. 13. First two components of the dataset after PCA.

This work, as shown in Fig. 14, plots the first three components of the new dataset on 3D axes. Here authors can clearly see a separation between inliers and outliers.

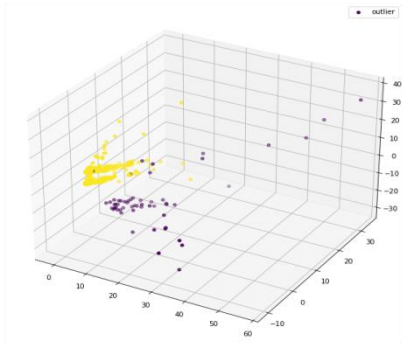

Fig. 14. First three components of the dataset after PCA.

Now authors have two datasets, one with all columns and another with columns left after removing columns with a standard deviation less than 0.3. Authors apply PCA to both datasets. This work split both datasets into three sets, train, test, and cross-validation set. The training set consists of 4000 inliers. The testing set consists of 586 inliers and 30 outliers. The cross-validation set consists of 400 inliers and 30 outliers.

IV. Gaussian Model for Anomaly Detection

A. Univariate Gaussian Model

Gaussian distribution is a continuous probability density function for a real-valued random variable in statistics. It is given by Eq. (1).

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1)$$

Where f(x) is the probability density function, μ is the mean and σ is the standard deviation.

This work calculates the mean and standard deviation of each column of both datasets and model a Gaussian distribution on all columns. The final probability is calculated by taking the product of the probabilities of all columns. Negative logarithms of probabilities are plotted as histograms as shown in Fig. 15 to 19. Fig. 15 shows probabilities of train inliers. Fig. 16 shows probabilities of test set. Fig. 17 shows probabilities of test set with columns having standard deviation less than 0.3 removed. Fig. 18 shows probabilities of cross validation set. Fig. 19 shows probabilities of cross validation set with columns having standard deviation less than 0.3 removed.

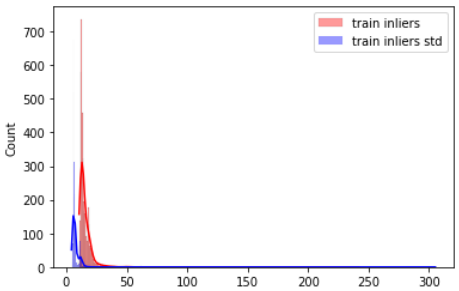

Fig. 15. Probabilities of train inliers.

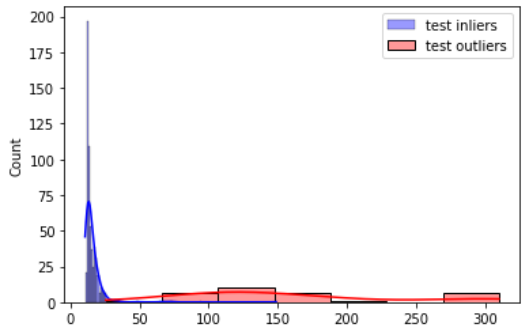

Fig. 16. Probabilities of test set.

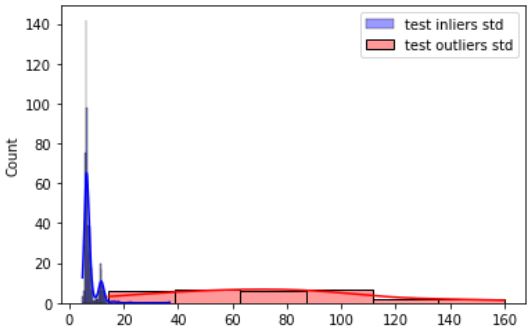

Fig. 17. Probabilities of test set with columns having standard deviation less than 0.3 removed.

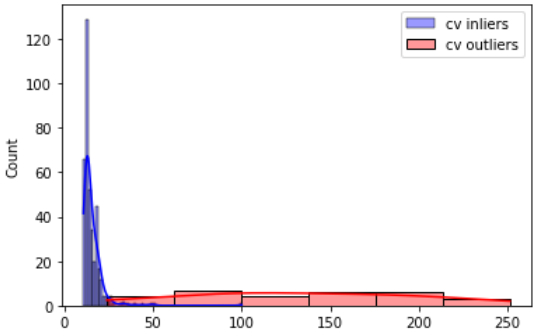

Fig. 18. Probabilities of cross validation set.

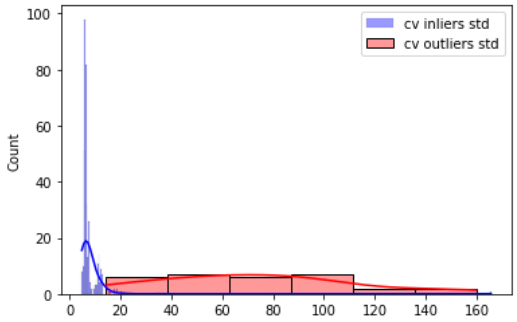

Fig. 19. Probabilities of cross validation set with columns having standard deviation less than 0.3 removed.

Authors set a threshold probability value to classify the test and cross-validation set. Different thresholds are set and accuracy is observed. Table II and III show accuracies for the original dataset.

TABLE II. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR TRAIN AND TEST INLIER SET

Threshold	Train accuracy	Test inlier accuracy
1e-10	0	0
1e-15	63.625	64.16
1e-20	89.725	89.078
1e-25	95.925	96.075
1e-30	97.825	97.78
1e-35	98.625	98.12
1e-40	99.05	98.63
1e-45	99.175	98.63

TABLE III. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR TEST OUTLIER, CROSS-VAL INLIER AND OUTLIER SET

Threshold	Test outlier accuracy	Cross Val inlier accuracy	Cross Val outlier accuracy
1e-10	100	0	100
1e-15	100	66.25	100
1e-20	100	91.75	100
1e-25	100	96.75	96.66
1e-30	96.66	98.0	93.33
1e-35	96.66	98.5	90.0
1e-40	93.33	99.25	90.0
1e-45	93.33	99.25	90.0

Table IV and V shows accuracies for the dataset whose columns were removed which had a standard deviation of less than 0.3.

TABLE IV. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR TRAIN INLIERS, TEST INLIERS AND TEST OUTLIERS SET

Threshold	Train accuracy	Test inlier accuracy	Test outlier accuracy
1e-10	82.475	83.95	100
1e-15	98.475	98.805	96.66
1e-20	99.4	99.65	93.33
1e-25	99.575	99.65	93.33
1e-30	99.6	99.65	99.65
1e-35	99.675	99.658	90.0
1e-40	99.725	99.82	90.0
1e-45	99.725	99.82	90.0

TABLE V. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR CROSS-VAL INLIERS AND OUTLIERS

Threshold	Cross Val inlier accuracy	Cross Val outlier accuracy
1e-10	86.25	96.66
1e-15	98.25	96.66
1e-20	99.5	90
1e-25	100	86.66
1e-30	100.0	86.66
1e-35	100.0	83.33
1e-40	100.0	80.0
1e-45	100.0	76.66

B. *Multivariate Gaussian Model*

The multivariate normal distribution, multivariate Gaussian distribution, or joint normal distribution are expansions of the one-dimensional normal distribution to higher dimensions in probability theory and statistics. It models the probability in one shot instead of calculating individual probabilities and multiplying them. Multivariate Gaussian distribution is given by the Eq. (2).

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}}|\Sigma|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^{T}\right\} \quad (2)$$

Where μ is the length-d row vector of means of all columns, ∑ is the covariance matrix of shape d x d. d is the number of features.

Authors calculate the mean and covariance matrices of both datasets to model a multivariate Gaussian distribution. Authors set a threshold value and classify the dataset between inlier and outlier and calculate accuracies for various threshold values. Negative logarithms of probabilities are plotted as a histogram as shown in Fig. 20 to 24. Fig. 20 shows probabilities of train inliers. Fig. 21 shows probabilities of test inliers and test outliers. Fig. 22 shows probabilities of test inliers and test outliers with columns having standard deviation less than 0.3 removed. Fig. 23 shows probabilities of cross-val inliers and cross-val outliers. Fig. 24 shows probabilities of cross-val inliers and cross-val outliers with columns having standard deviation less than 0.3 removed.

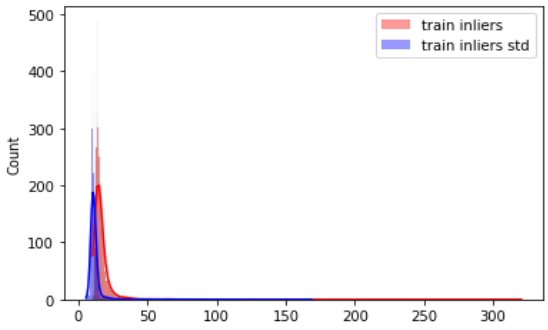

Fig. 20. Probabilities of train inliers.

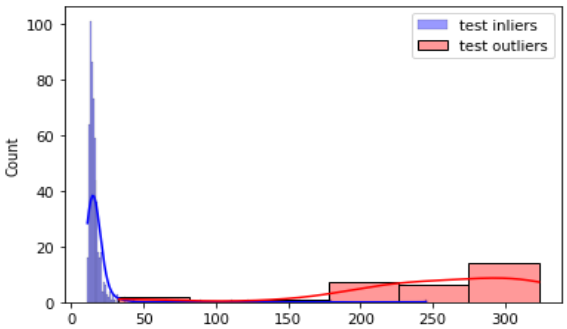

Fig. 21. Probabilities of test inliers and test outliers.

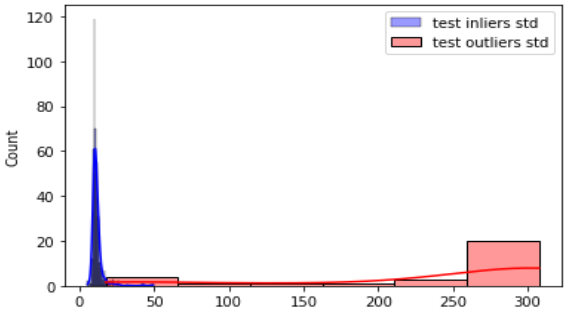

Fig. 22. Probabilities of test inliers and test outliers with columns having standard deviation less than 0.3 removed.

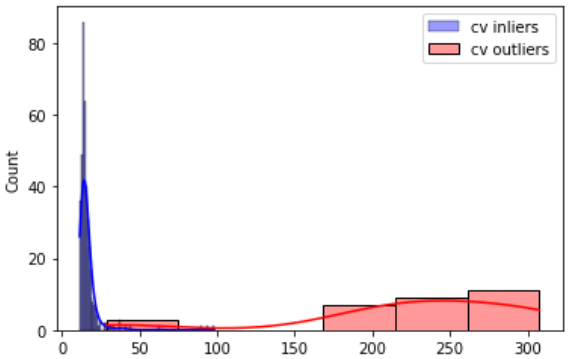

Fig. 23. Probabilities of cross-val inliers and cross-val outliers.

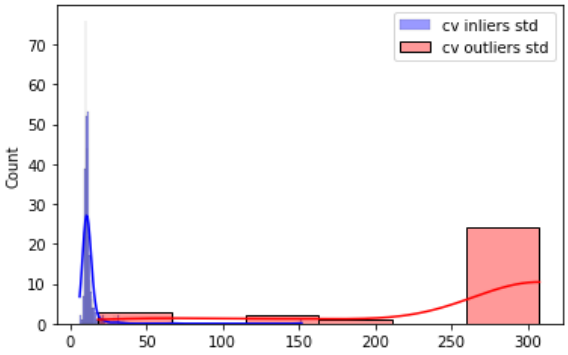

Fig. 24. Probabilities of cross-val inliers and cross-val outliers with columns having standard deviation less than 0.3 removed.

Table VI and VII show accuracies for the original dataset.

TABLE VI. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR TRAIN INLIERS, TEST INLIERS, AND TEST OUTLIERS SET

Threshold	Train accuracy	Test inlier accuracy	Test outlier accuracy
1e-10	0.0	0.0	100.0
1e-15	29.2	30.88	100.0
1e-20	83.1	84.47	100.0
1e-25	93.075	94.02	100.0
1e-30	96.325	97.26	100.0
1e-35	97.575	97.78	93.33
1e-40	98.5	98.80	90.0
1e-45	98.8	99.146	90.0

TABLE VII. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR CROSS-VAL INLIERS AND CROSS-VAL OUTLIERS SET

Threshold	Cross Val inlier accuracy	Cross Val outlier accuracy
1e-10	0.0	100.0
1e-15	30.0	100.0
1e-20	82.5	100.0
1e-25	94.0	100.0
1e-30	96.0	100.0
1e-35	97.5	96.66
1e-40	98.25	96.66
1e-45	98.75	96.66

Table VIII and IX show accuracies for the dataset whose columns were removed which had a standard deviation of less than 0.3.

TABLE VIII. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR TRAIN INLIERS, TEST INLIERS, AND TEST OUTLIERS

Threshold	Train accuracy	Test inlier accuracy	Test outlier accuracy
1e-10	16.675	16.21	100.0
1e-15	94.125	93.68	100.0
1e-20	97.2	97.95	96.66
1e-25	98.625	99.48	93.33
1e-30	99.125	100.0	93.33
1e-35	99.325	100.0	93.33
1e-40	99.4	100.0	93.33
1e-45	99.45	100.0	93.33

TABLE IX. VARIATION OF ACCURACY WITH DIFFERENT THRESHOLD VALUES FOR CROSS-VAL INLIERS AND CROSS-VAL OUTLIERS

Threshold	Cross Val inlier accuracy	Cross Val outlier accuracy
1e-10	18.0	100.0
1e-15	93.25	100.0
1e-20	97.25	96.66
1e-25	98.0	93.33
1e-30	98.75	93.33
1e-35	99.25	90.0
1e-40	99.5	90.0
1e-45	99.5	90.0

V. CONCLUSION

This work states that univariate and multivariate Gaussian models for anomaly detection are successfully created. Data imbalance is not an issue here because these models fit on the train set and this work uses a threshold to predict inliers and outliers. This work examines the trend between various threshold values and accuracies. The proposed method, a Gaussian model to forecast the likelihood of an attack occurring based on certain system parameters uses a univariate and a multivariate Gaussian model on the training dataset and examines accuracies for various threshold values. It also addresses the challenge of class imbalance in anomaly detection situations. This method presents the successful creation of univariate and multivariate Gaussian models for anomaly detection. The data imbalance is not an issue in these models because they fit on the train set and use a threshold to predict inliers and outliers. The study also examines the relationship between various threshold values and accuracies. For univariate Gaussian model variation of accuracy with different threshold values ranges up to 99.175 percent and for train accuracy up to 98.6 percent for test inlier accuracy and up to 100 percent for test outlier accuracy. For multivariate Gaussian model variation of accuracy with different threshold values ranges up to 99.45 for train accuracy, up to 100 for test inlier accuracy and up to 100 for test outlier accuracy with validation.

Future work is about using deep learning techniques such as auto encoders. Machine learning is revealing a plethora of potential for cybersecurity aficionados to explore as more and more data is gathered, specifically with the data that they already own. When this work talks about escalating warfare in the internet age, timely automated identification of any threats or suspicious conduct can avoid a number of mistakes from occurring.

REFERENCES

[1] Mohammad-Mahdi Bazm, Thibaut Sautereau, Marc Lacoste, Mario Sudholt, Jean-Marc Menaud, ¨ ”Cache-Based Side-Channel Attacks Detection through Intel Cache Monitoring Technology and Hardware Performance Counters”, FMEC2018 - The Third IEEE International Conference on Fog and Mobile Edge Computing, Apr 2018, Barcelona, Spain. IEEE, pp.1-6. ¡hal-01762803¿

[2] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, “Malware guard extension: Using sgx to conceal cache attacks”, arXiv preprint arXiv:1702.08719,2017.

[3] C. Disselkoen, D. Kohlbrenner, L. Porter, and D. Tullsen, “Prime+abort: A timer-free high-precision l3 cache attack using intel TSX”, in 26th USENIX Security Symposium (USENIX Security 17), (Vancouver, BC), pp. 51–67, USENIX Association, 2017.

[4] M. Chiappetta, E. Savas, and C. Yilmaz, “Real time detection of cache-based side-channel attacks using hardware performance counters”, Applied Soft Computing, vol. 49, pp. 1162–1174, 2016.

[5] Ziqi Wang, Rui Yang, Xiao Fu, Xiaojiang Du, and Bin Luo, ”A shared memory based cross-vm side channel attacks in iaas cloud”, In Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference on, pages 181–186. IEEE, 2016.

[6] A. Valdes, K. Skinner, Adaptive, ”Model-Based Monitoring for Cyber Attack Detection”, International Workshop on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2000.

[7] Deri, Luca, and Alfredo Cardigliano. "Using cyberscore for network traffic monitoring." In 2022 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 56-61. IEEE, 2022.

[8] P. Patil and R. Ingle, "Meta-ensemble based classifier approach for attack detection in multi-tenant distributed systems," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154077.

[9] A. Valdes, K. Skinner, Adaptive, ”Model-Based Monitoring for Cyber Attack Detection”, International Workshop on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2000.

[10] Michael Hilker, Christoph Schommer,” Description of bad-signatures for network intrusion detection”, n Conf.s in Research and Practice in Information Technology Series, vol. 54, ACSW, 2006, pp. 175–182

[11] N. Stakhanova, M. Couture, A. A. Ghorbani, ”Exploring network-based malware classification”, 2011 6th International Conference on Malicious and Unwanted Software

[12] F. Iglesias, T. Zseby, ”Analysis of network traffic features for anomaly detection”, Mach. Learn.101(1-3), 59–84 (2014).

[13] M. Z. Rafique, P. Chen, C. Huygens, W. Joosen, ”Evolutionary algorithms for classification of malware families through different network behaviors”, Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Pages 1167-1174.

[14] W. Dai, Q. Yang, G. -R. Xue, Y. Yu, ”Boosting for transfer learning”, 24th International Conf. on Machine Learning(ICML) 2007

[15] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, ”Domain adaptation via transfer component analysis”, IEEE Transaction on Neural Netw.22(2), 199–210, 2011.

[16] Shi, Q. Liu, W. Fan, P. S. Yu, R. Zhu, ”Transfer learning on heterogenous feature spaces via spectral transformation”, IEEE International Conf. on Data Mining (ICDM), 2010.

[17] D. Bekerman, B. Shapira, L. Rokach, A. Bar, ”Unknown malware detection using network traffic classification”, 2015 IEEE Conference on Communications and Network Security (CNS).