System performance anomaly detection using tracing data analysis

Iman Kohyarnejadfard
Prof. Daniel Aloise and Prof. Michel Dagenais

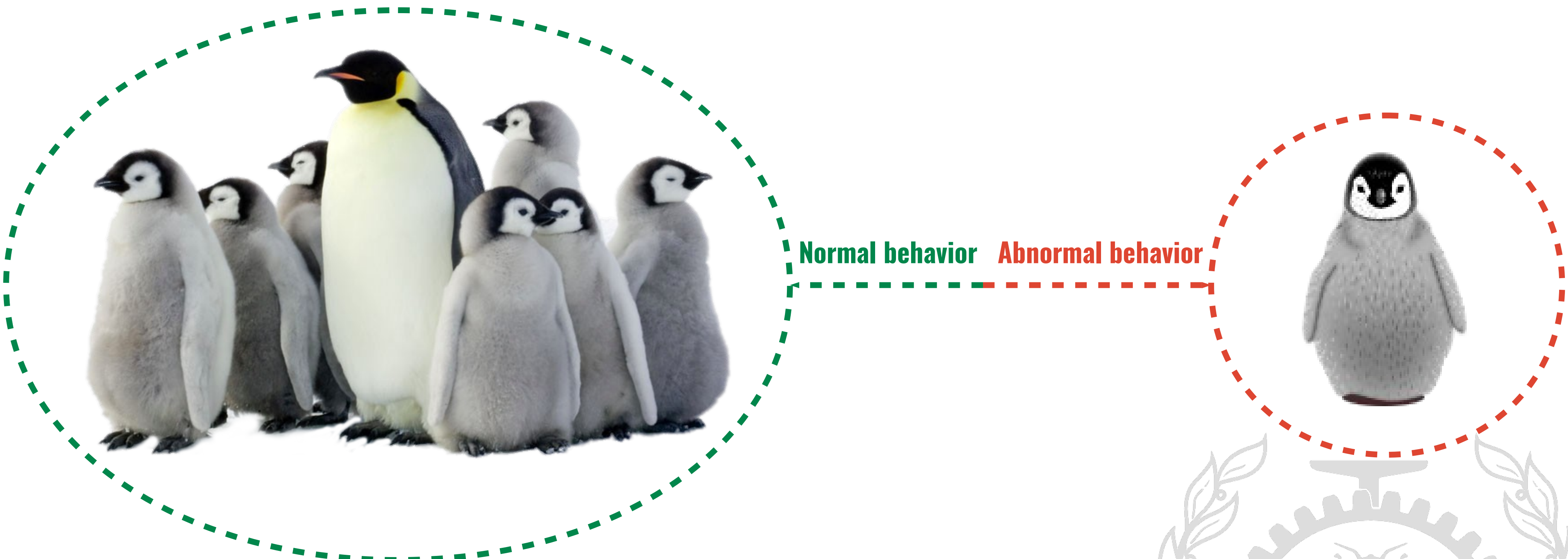
POLYTECHNIQUE
MONTREAL

Agenda


Results
03
01
Introduction
02
Methodology and
Implementation
04
Conclusions &
Questions

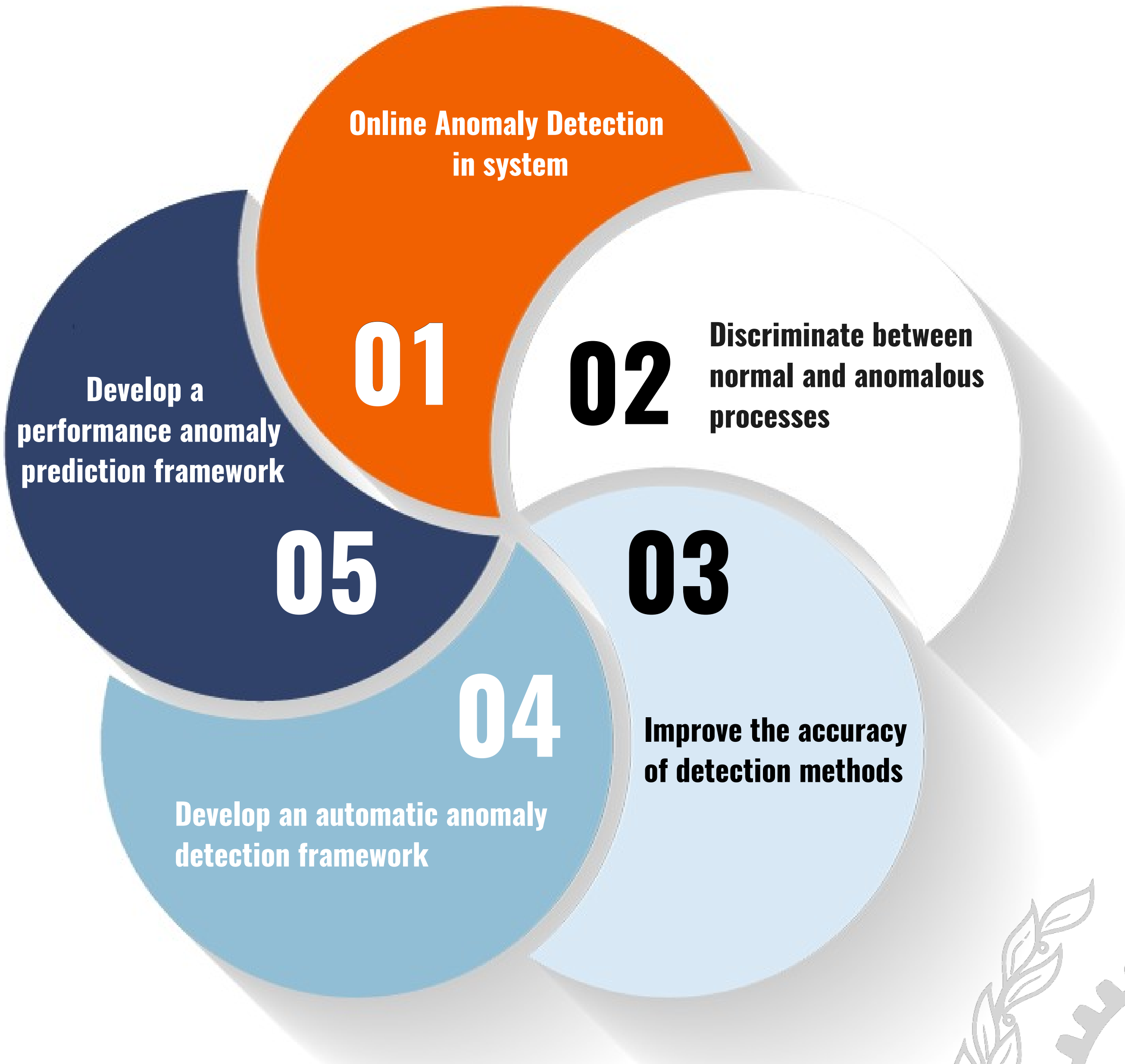
Anomaly Detection

- Anomaly is something different which deviates from the common rule.
- Anomalies are patterns in data that do not conform to a well defined notion of normal behavior.
- Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior.
- Many anomaly detection techniques have been developed for various application domains.



The figure retrieved from: https://pngtree.com

Motivation


Online Anomaly Detection in system
01
02
Discriminate between normal and anomalous processes
03
Improve the accuracy of detection methods
04
Develop an automatic anomaly detection framework
05
Develop a performance anomaly prediction framework

Challenges

01 Defining a normal region that encompasses every possible normal behavior is very difficult.

02 Normal behavior keeps evolving and the current notion of normal behavior might not be sufficiently representative in the future.

03 The exact notion of an anomaly is different for different application domains.

04 Availability of labeled data for training/validation of models used by anomaly detection techniques is a major issue.

Why system calls?

01 System Call is a program signal for requesting a service from the system kernel.

02 System calls can represent low-level interactions between a process and the kernel in the system.

03 system call traces generated by program executions are stable and consistent during program's normal activities so that they can be used to distinguish the abnormal operations from normal activities.

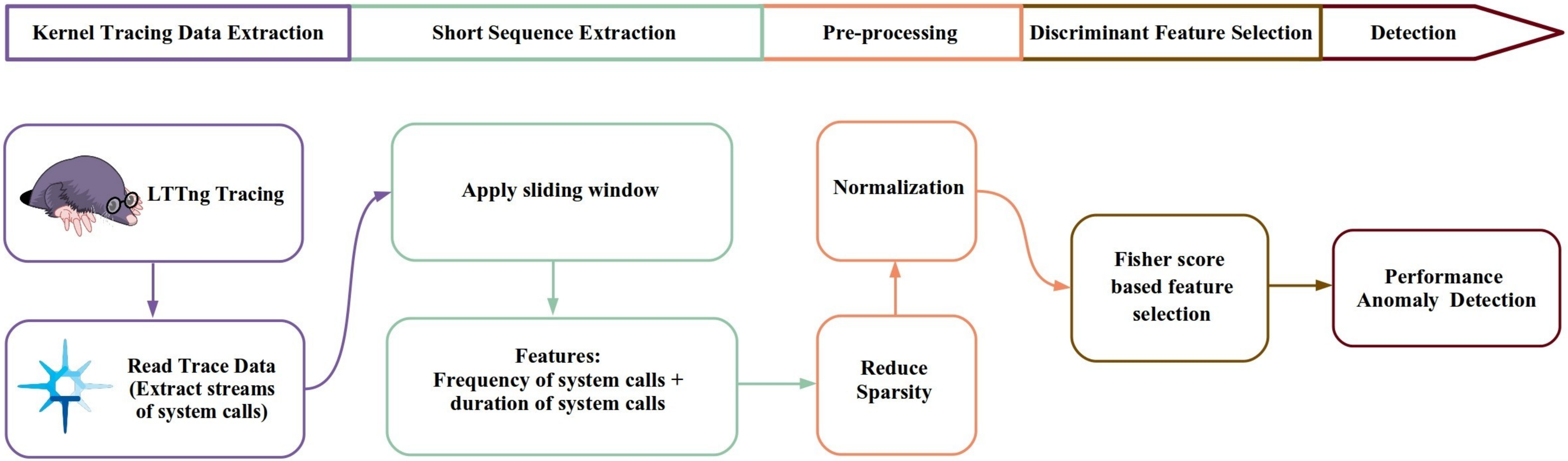
04 System call streams are enormous, and suitable to use in machine learning. A single process can produce thousands system calls per second.

05 We can use three different representations of system calls: n-grams of system call names, histograms of system call names, and individual system calls with associated parameters.

06 System call sequences can provide both momentary and temporal dynamics of process behavior.

Methodology

- The methodology is based on collecting streams of system calls produced by all or selected processes on the system, and sending them to a monitoring module.
- Machine learning algorithms are used to identify changes in process behavior.
- The methodology uses a sequence of system call count vectors or sequence of system call duration vectors as the data format which can handle large and varying volumes of data.



Our Use Case

The open source MySQL synthetic benchmarks tool, Sysbench, with oltp test in complex mode.

A virtual machine with different workloads, such as:
I. (CPU problem) Setting the VM's CPU cap to too low (e.g., 1 CPU core, while running 8 threads of MySQL)
II. (Memory problem) Setting the memory cap to too low (e.g., 256 MB memory, while the MySQL table is of size 6 GB)

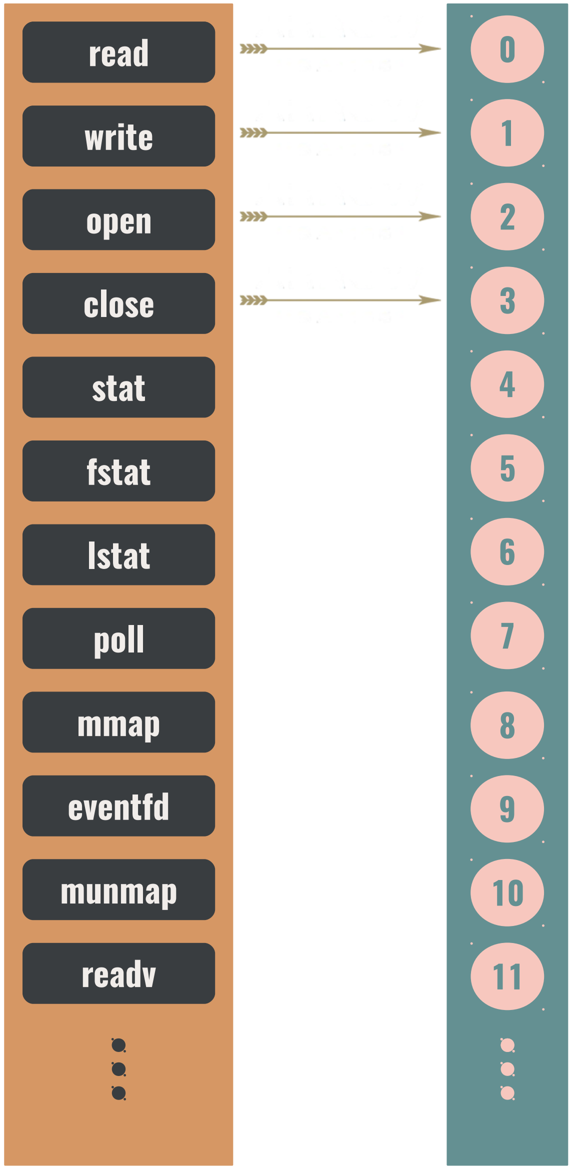
Sliding window = 10k system calls
overlapping size = 100 system calls

18k normal and anomalous samples

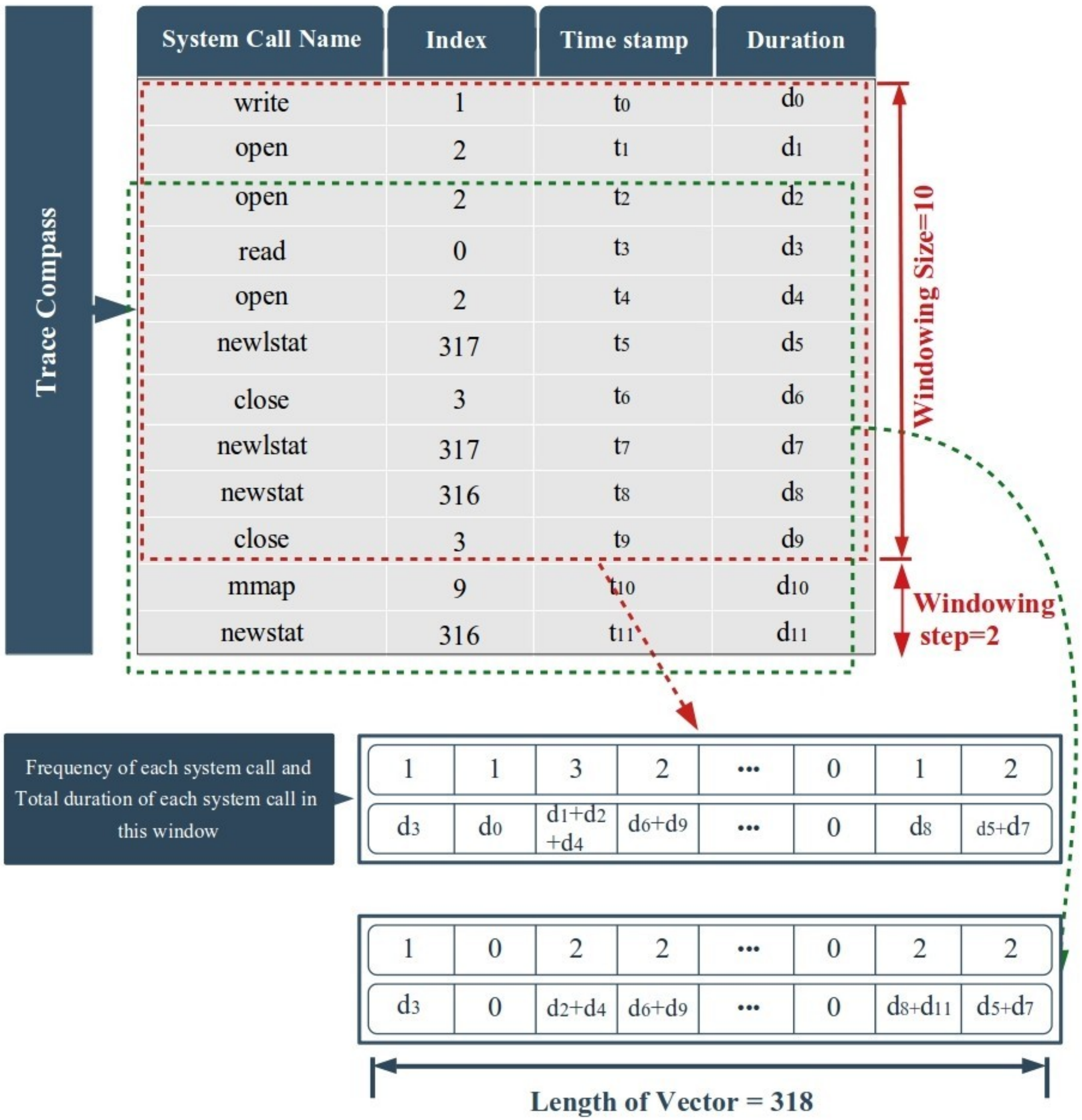
The benchmarking tool is run on virtual machines with different configurations and varying load on resources; LTTng is used to record the different tracing data streams.

Trace compass is used to read tracing data, create tables of system calls and construct the initial vectors to use in machine learning part.

Indexes instead of names



Read Trace


Trace Compass
System Call Name
Index
Time stamp
Duration
write 1 t0 d0
open 2 t1 d1
open 2 t2 d2
read 0 t3 d3
open 2 t4 d4
newlstat 317 t5 d5
close 3 t6 d6
newlstat 317 t7 d7
newstat 316 t8 d8
close 3 t9 d9
mmap 9 t10 d10
newstat 316 t11 d11
Windowing Size=10
Windowing step=2
Frequency of each system call and Total duration of each system call in this window
1 1 3 2 ••• 0 1 2
d3 d0 d1+d2 +d4 d6+d9 ••• 0 d8 d5+d7
1 0 2 2 ••• 0 2 2
d3 0 d2+d4 d6+d9 ••• 0 d8+d11 d5+d7
Length of Vector = 318

Preprocessing

1. **Scaling**

   It selects the same number of samples from each class without considering any order in vectors.

2. **Normalization**

   The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

3. **Sparsity**

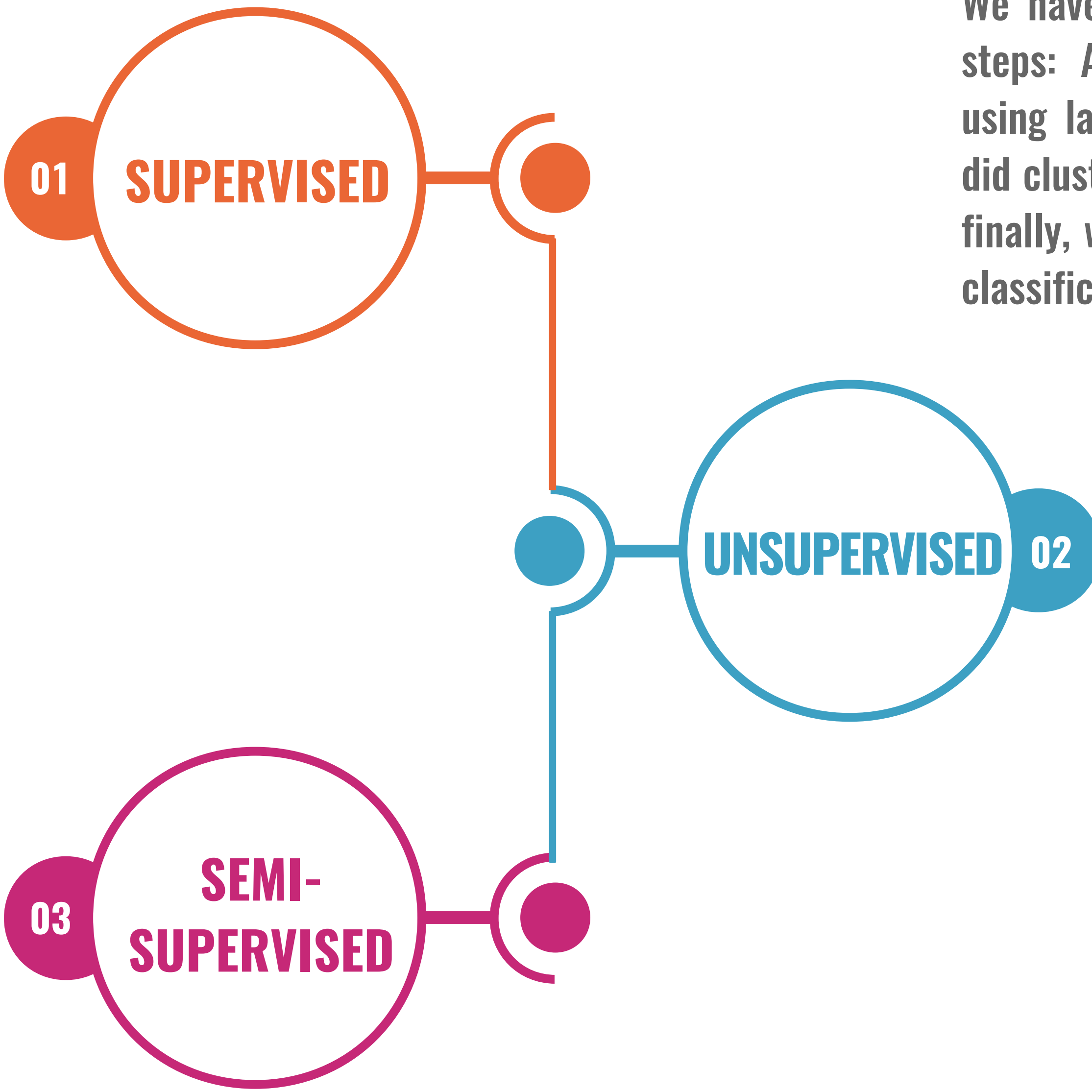
   Sparse matrices are common in machine learning. They occur in some data collection processes or applying certain data transformation techniques like one-hot encoding or count vectorizing.

4. **Fisher score**

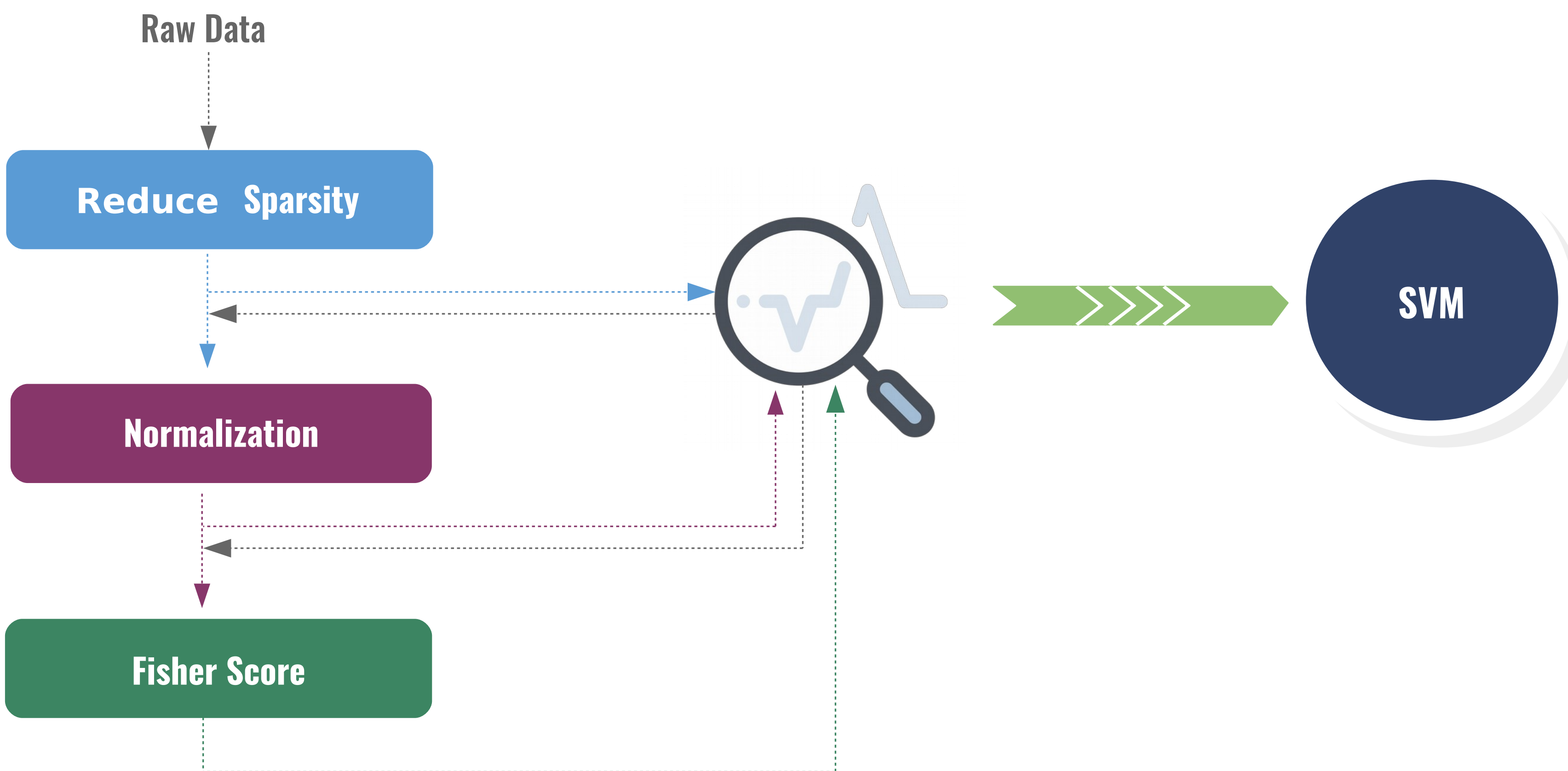
   It selects each feature independently according to their scores under the Fisher criterion, which leads to a suboptimal subset of features.

Learning part

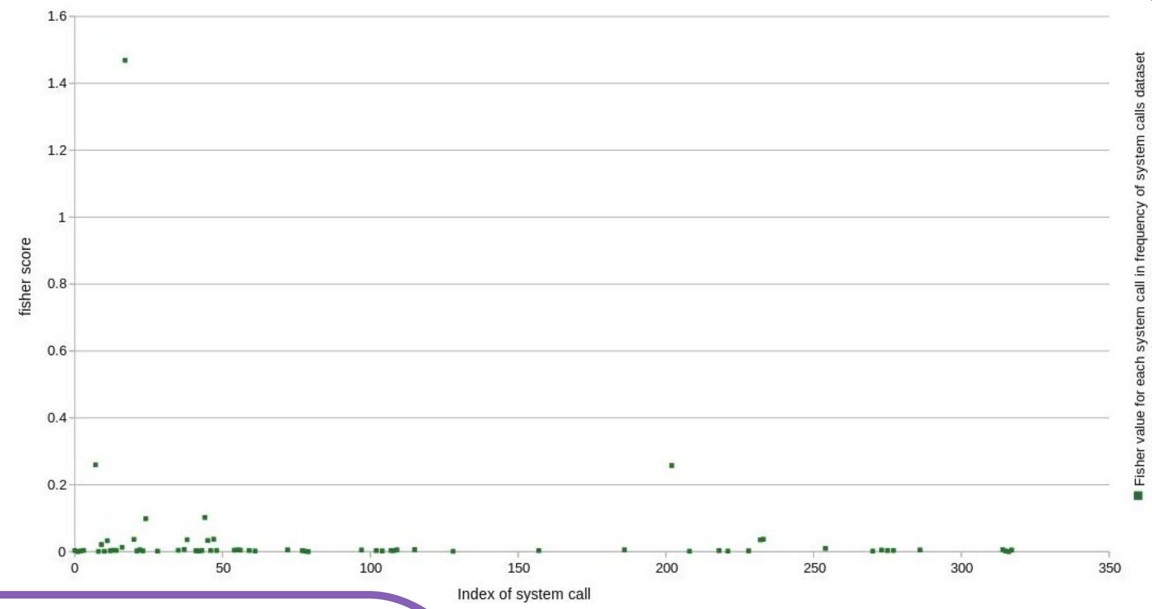
We have analysed the created dataset in 3 steps: At first, we trained the SVM classifier using labelled data. In the second step, we did clustering on samples (unsupervised) and finally, we used parameters estimated in the classification step to cluster the samples.

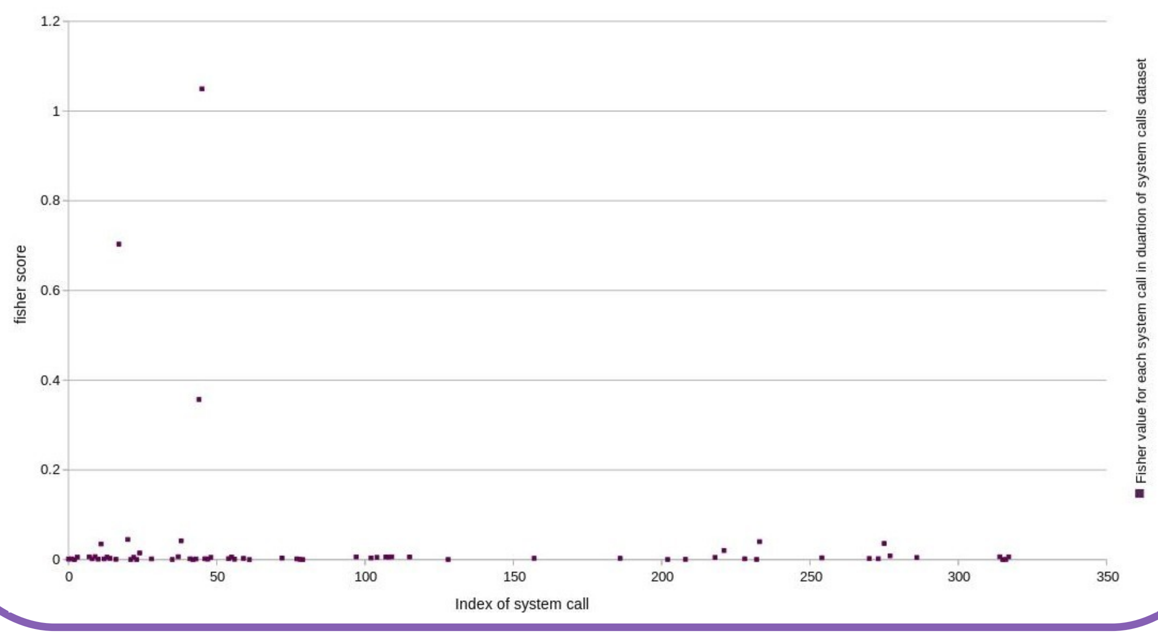


Supervised Learning



The Fisher score for each system call in frequency-based approach

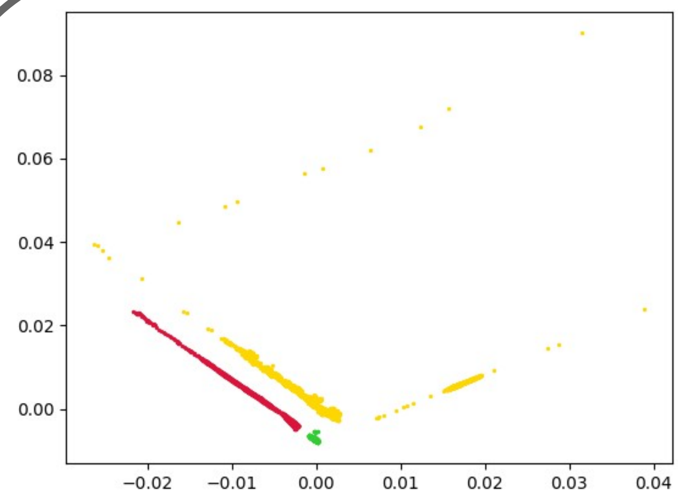


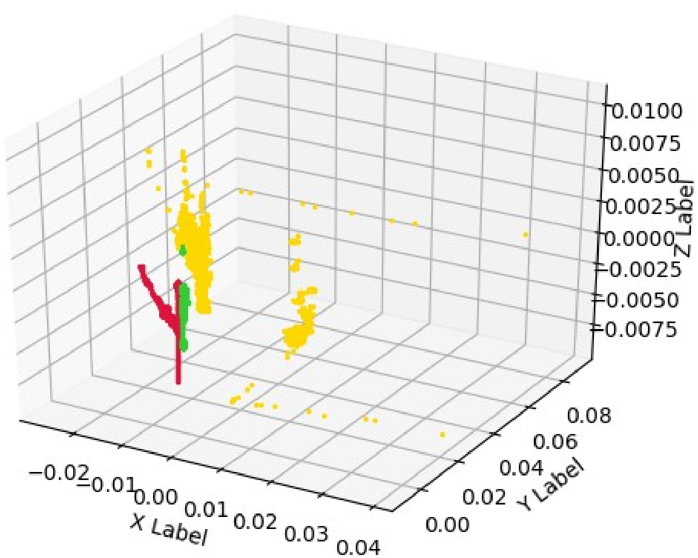


The Fisher score for each system call in duration-based approach

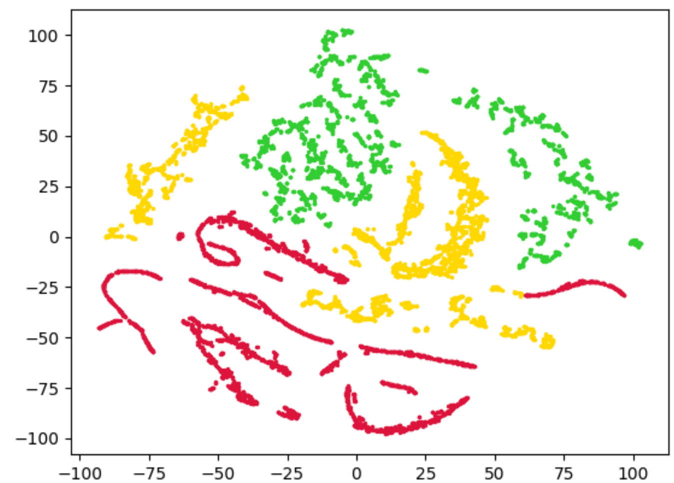
Unsupervised Learning

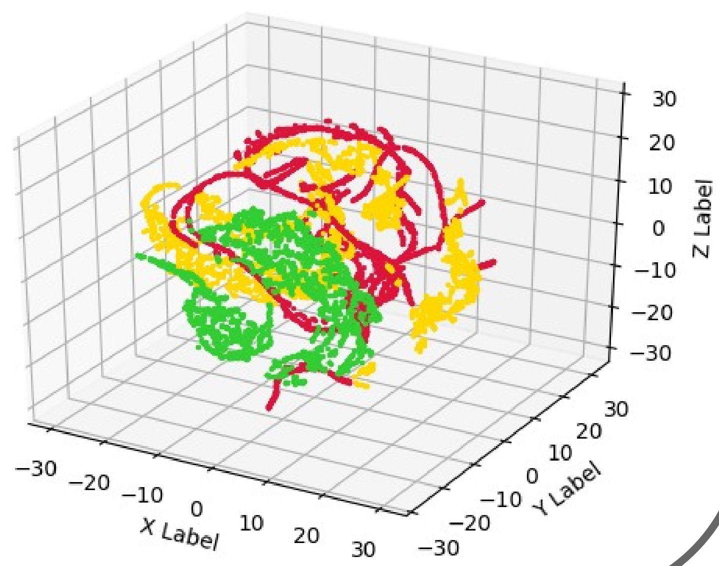
Projection of Dataset using PCA



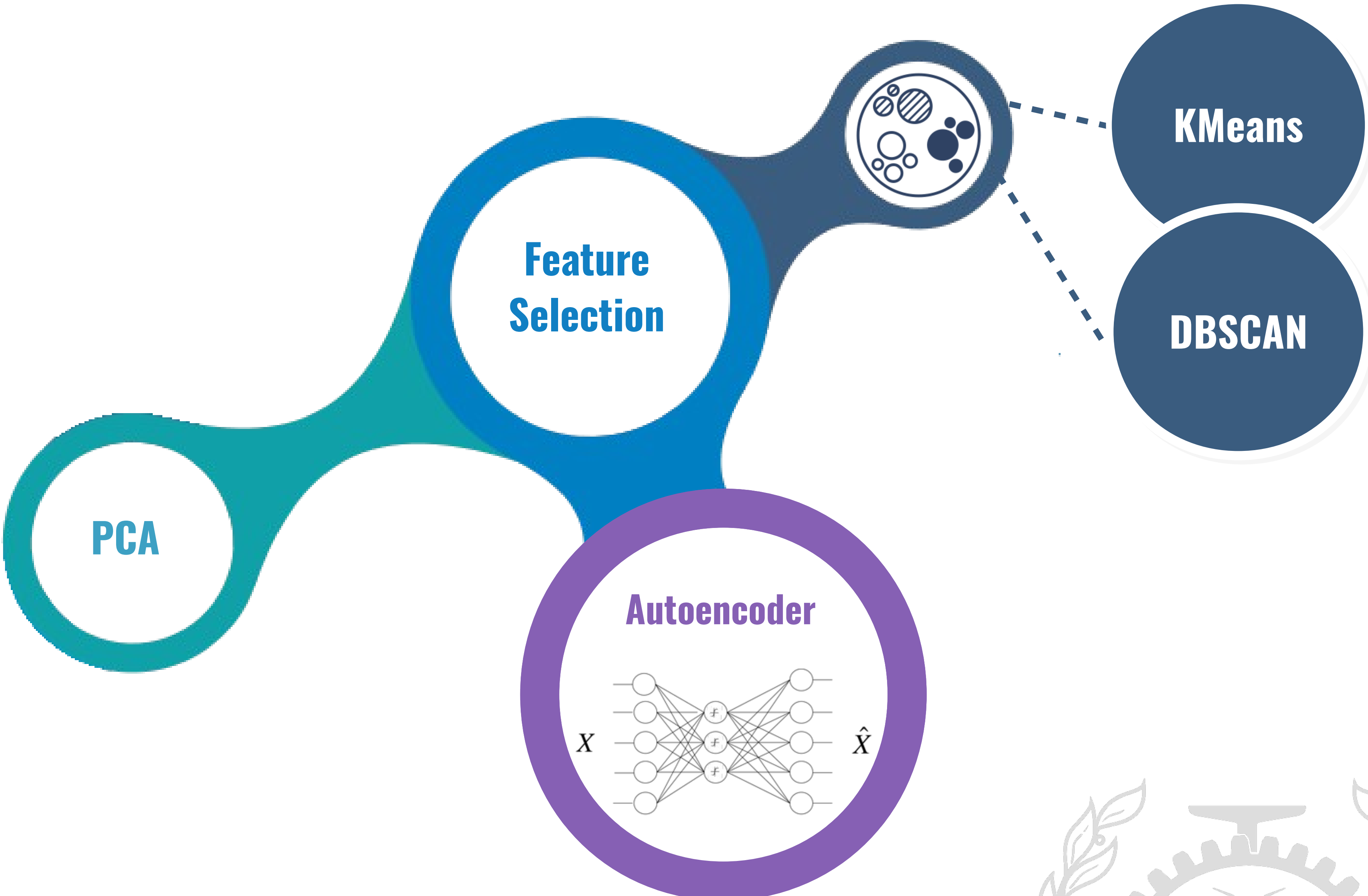


Projection of Dataset using TSNE

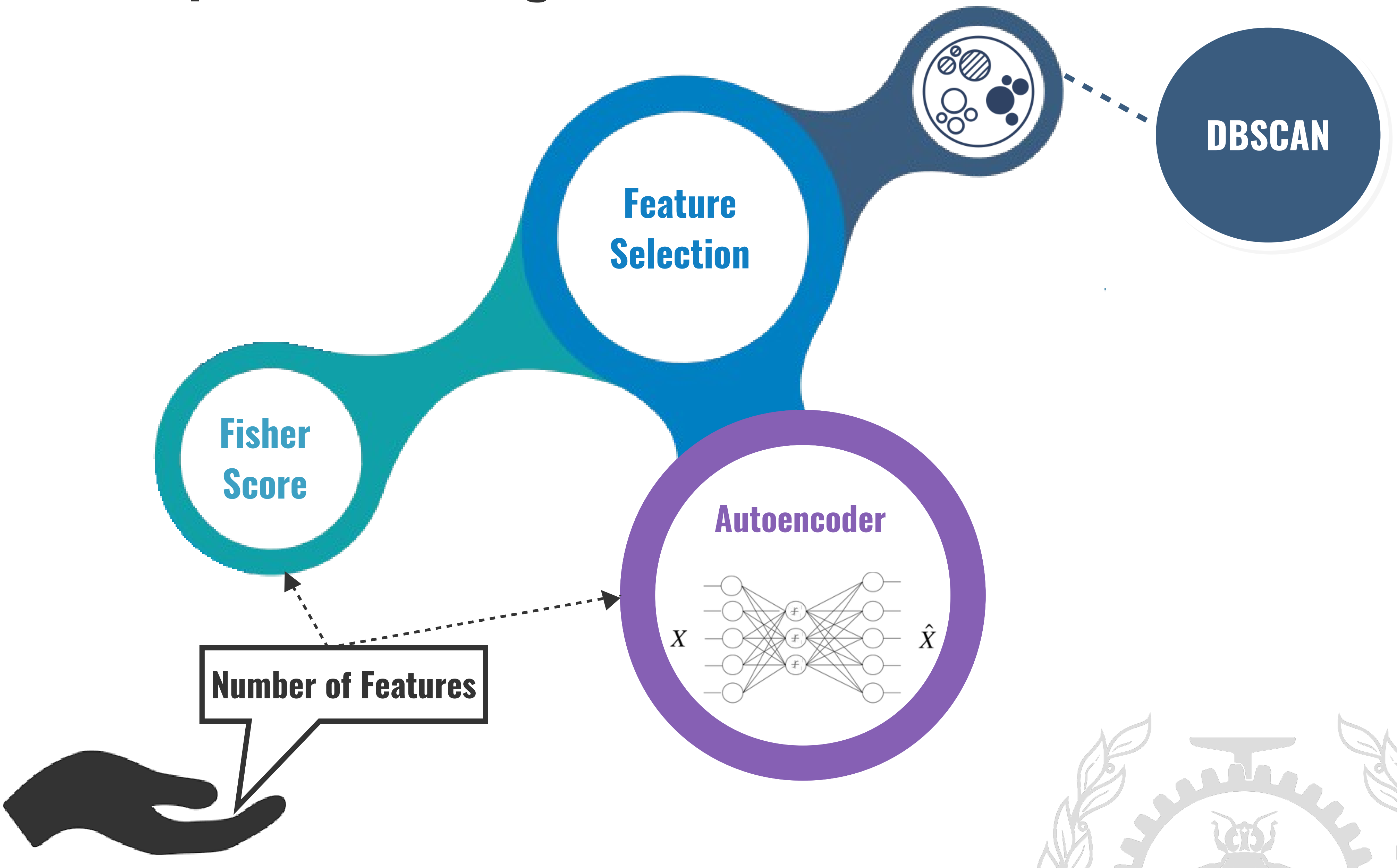




Unsupervised Learning

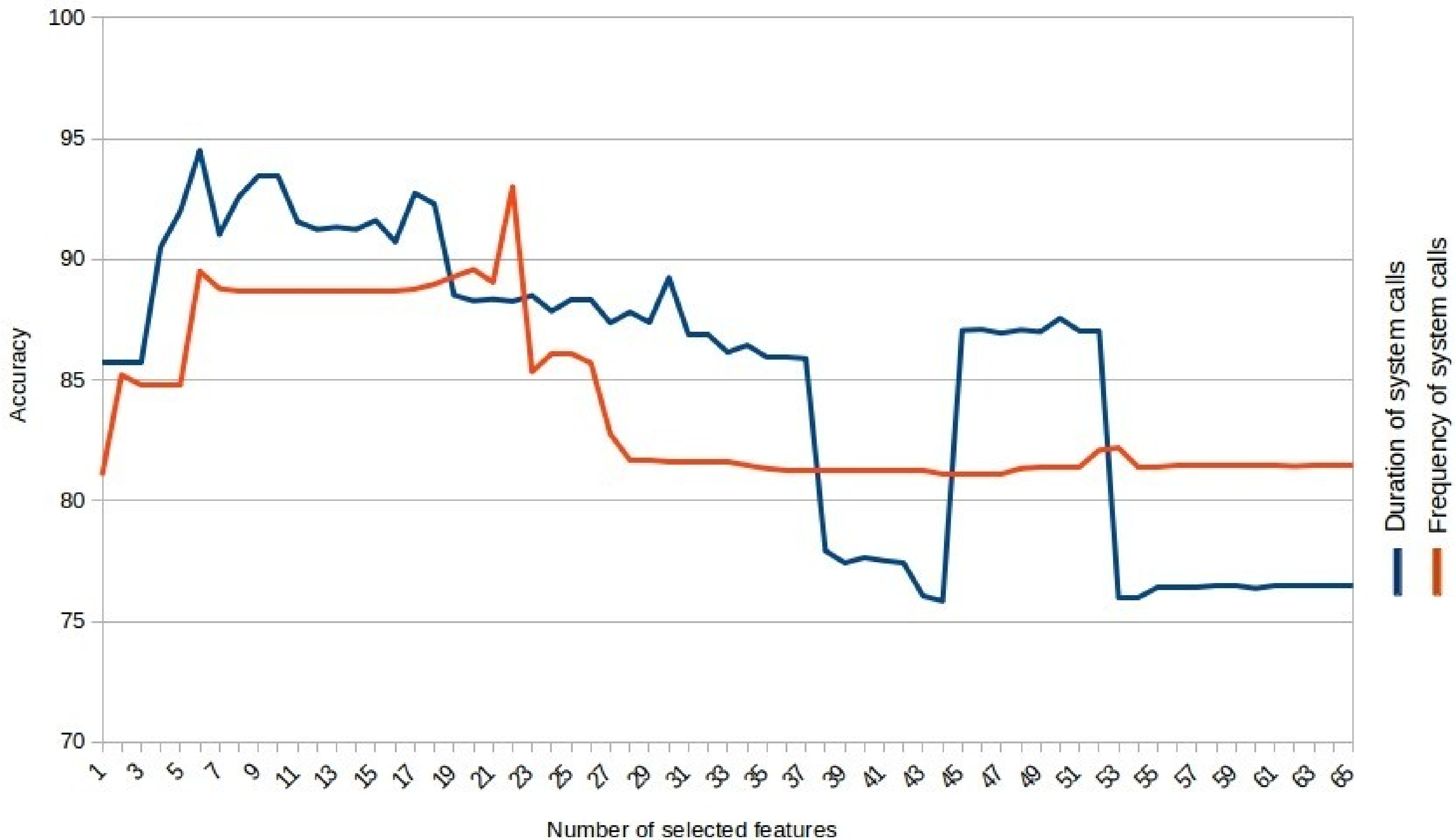


Semi-Supervised Learning



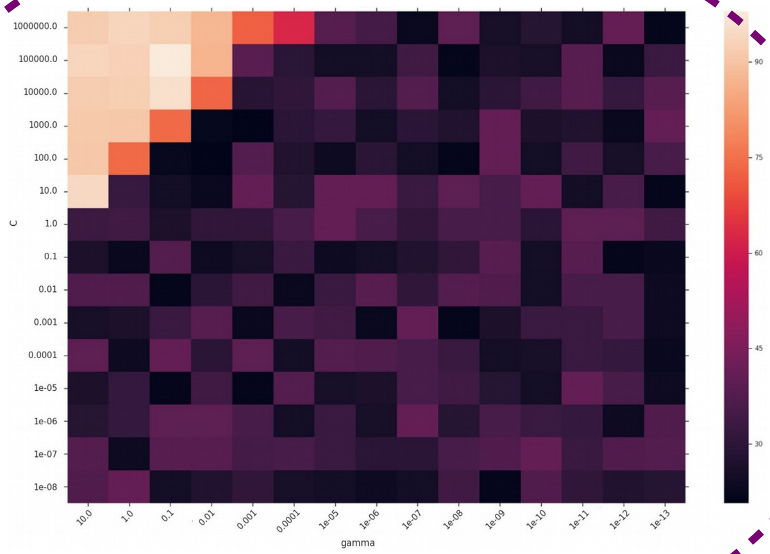
The figure retrieved from: https://www.vectorstock.com

Results



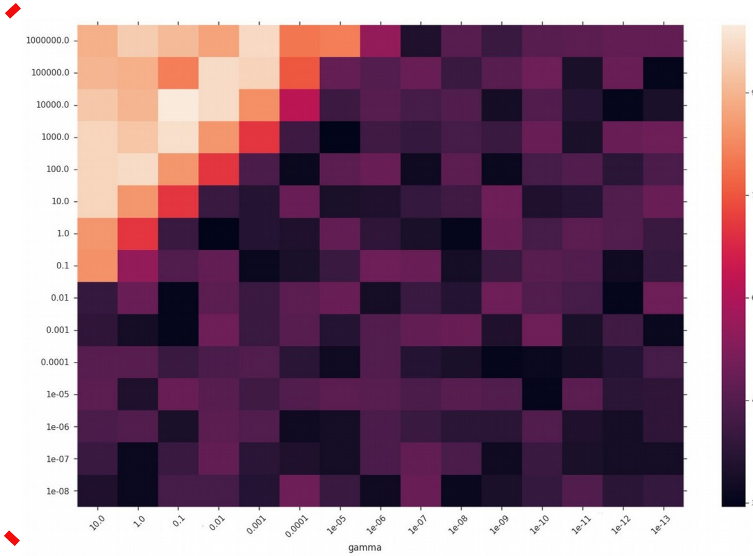
Supervised Learning accuracy versus different number of top-ranked features

Results

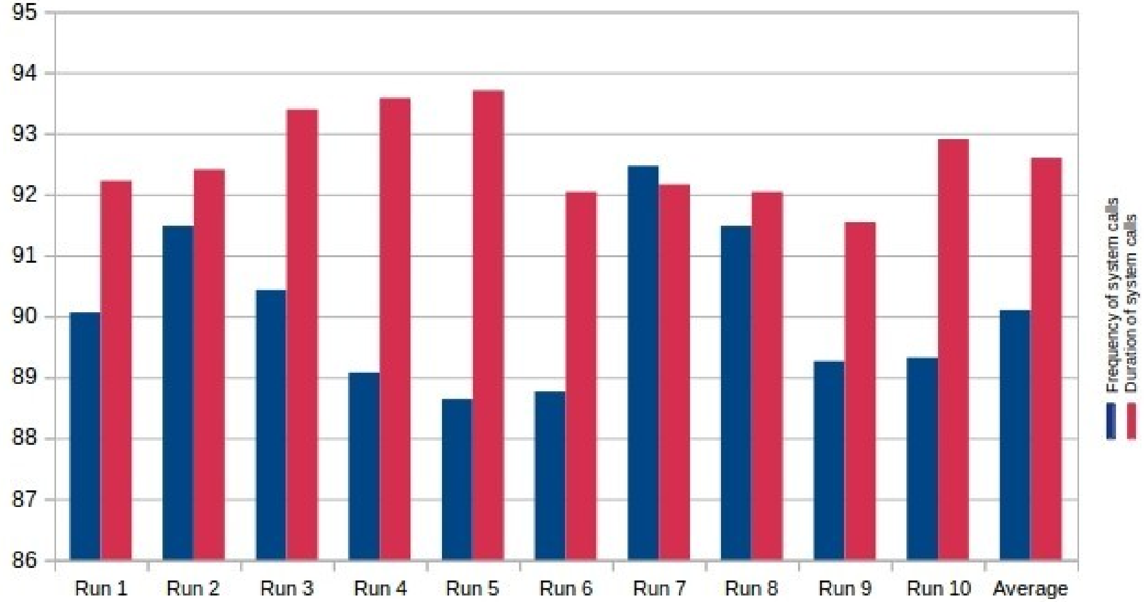


Heat map of the frequency-based anomaly detection accuracy using different γ and C.

Heat map of the duration-based anomaly detection accuracy using different γ and C.



Results



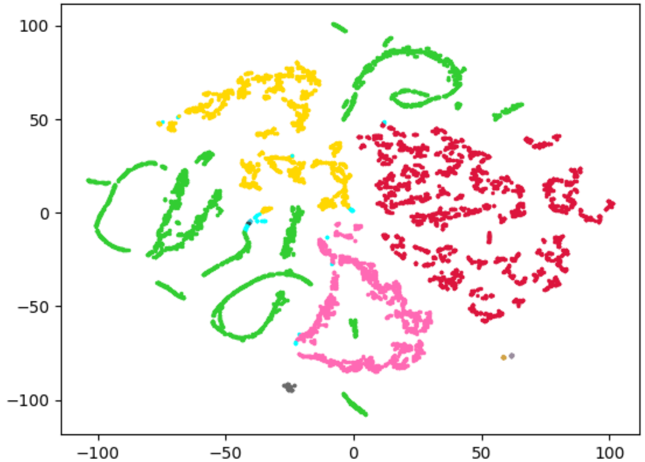
Accuracy of the supervised learning approach on multiple runs.

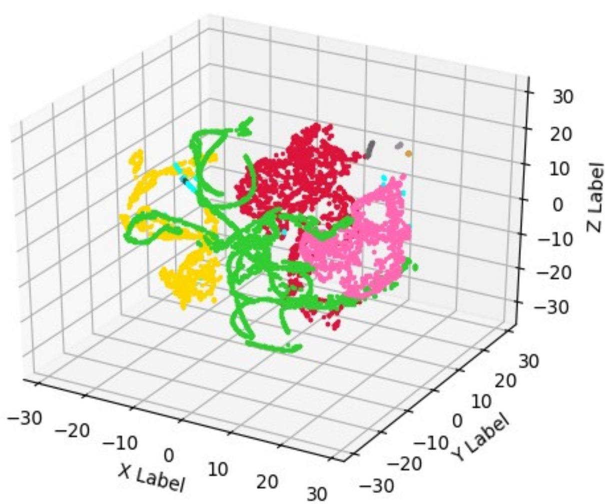
	Duration			Frequency		
	Acc	Prec	Rec	Acc	Prec	Rec
RBF	**0.925**	**0.848**	**0.806**	**0.900**	0.857	**0.911**
SIG	0.906	0.796	0.660	0.886	0.884	0.879
POLY	0.911	0.810	0.674	0.882	**0.913**	0.744

The performance of the proposed RBF based anomaly detection approach compared to the Sigmoid and polynomial based methods.

Results

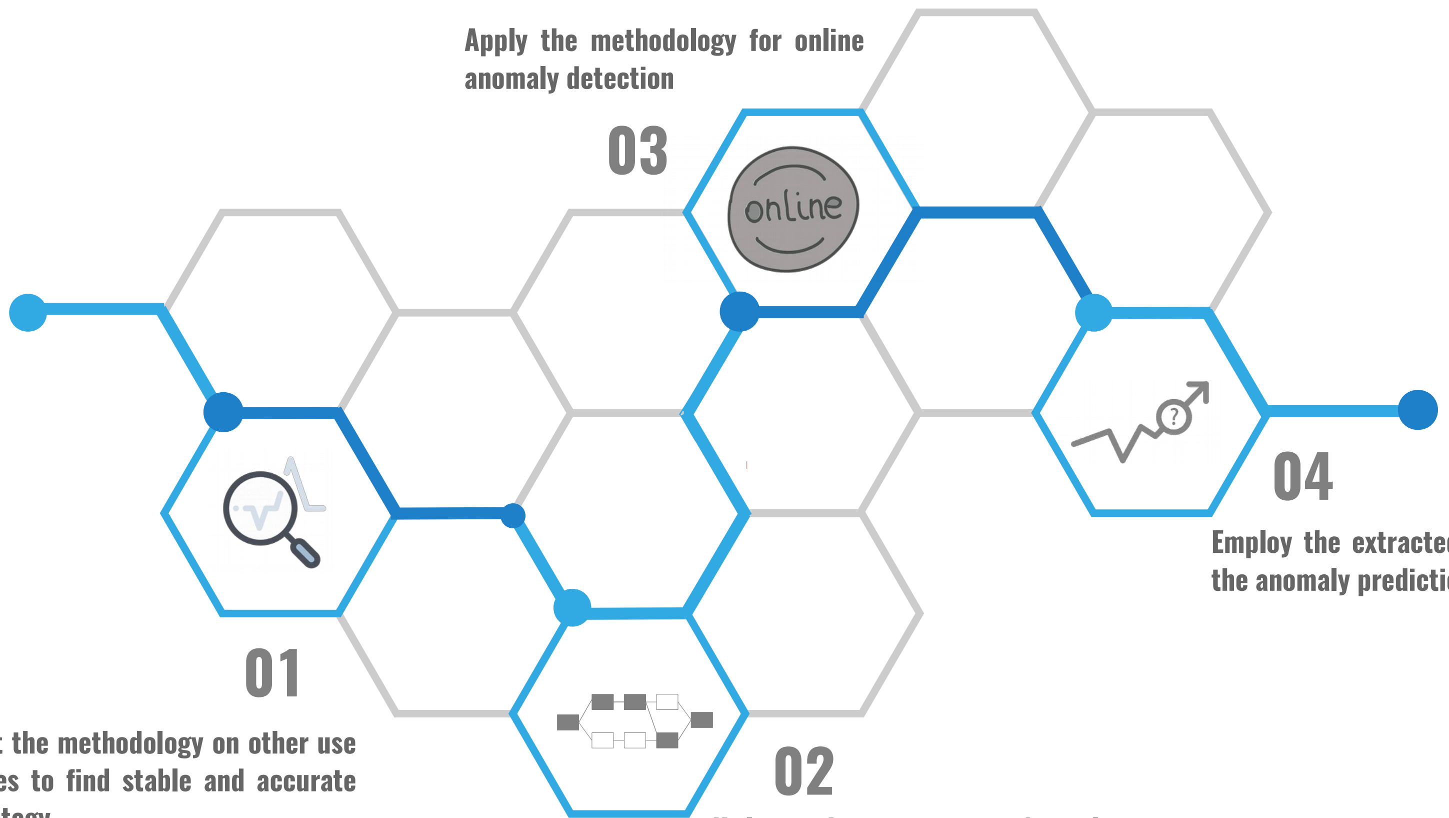
Projection of clustered samples using DBSCAN & Fisher Score





We gained <u>ARI=0.8471</u> by utilizing DBSCAN clustering method with eps=0.001 and Fisher Score feature selection with (number of top-ranked features)=3

Future Directions



01
Test the methodology on other use cases to find stable and accurate strategy.

02
Utilize other metrics and analysis such as critical path data extraction to improve the performance.

03
Apply the methodology for online anomaly detection

04
Employ the extracted features in developing the anomaly prediction framework.

Thank you for your attention!

Questions?

Iman.kohyarnejadfard@polymtl.ca
https://github.com/Kohyar

References

[1] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for unix processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, SP '96, pages 120–, Washington, DC, USA, 1996. IEEE Computer Society.
[2] Z. Xu, X. Yu, Y. Feng, J. Hu, Z. Tari, and F. Han. A multi-module anomaly detection scheme based on system call prediction. In 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pages 1376–1381, June 2013.
[3] E. Eskin, , and S. J. Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, volume 1, pages 165–175 vol.1, June 2001.
[4] A. Liu, C. Martin, T. Hetherington, and S. Matzner. A comparison of system call feature representations for insider threat detection. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pages 340–347, June 2005.
[5] Dae-Ki Kang, D. Fuller, and V. Honavar. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pages 118–125, June 2005.
[6] R. Canzanese, S. Mancoridis, and M. Kam. System call-based detection of malicious processes. In 2015 IEEE International Conference on Software Quality, Reliability and Security, pages 119–124, Aug 2015.
[7] Michael Dymshits, Ben Myara, and David Tolpin. Process monitoring on sequences of system call count vectors. 2017 International Carnahan Conference on Security Technology (ICCST), pages 1–5, 2017.
[8] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. Deep learning for classification of malware system call sequences. In Byeong Ho Kang and Quan Bai, editors, AI 2016: Advances in Artificial Intelligence, pages 137–149, Cham, 2016. Springer International Publishing.
[9] Mathieu Desnoyers and Michel Dagenais. The lttng tracer : A low impact performance and behavior monitor for gnu / linux. In OLS Ottawa Linux Symposium, 2006.
[10] Tracecompass. https://projects.eclipse.org/projects/tools.tracecompass. Accessed: 2019-01-30.
[11] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.
[12] Ulrich H.-G. Kre. Advances in kernel methods. chapter Pairwise Classification and Support Vector Machines, pages 255–268. MIT Press, Cambridge, MA, USA, 1999.
[13] J. Alvarez Cid-Fuentes, C. Szabo, and K. Falkner. Adaptive performance anomaly detection in distributed systems using online svms. IEEE Transactions on Dependable and Secure Computing, pages 1–1, 2018.