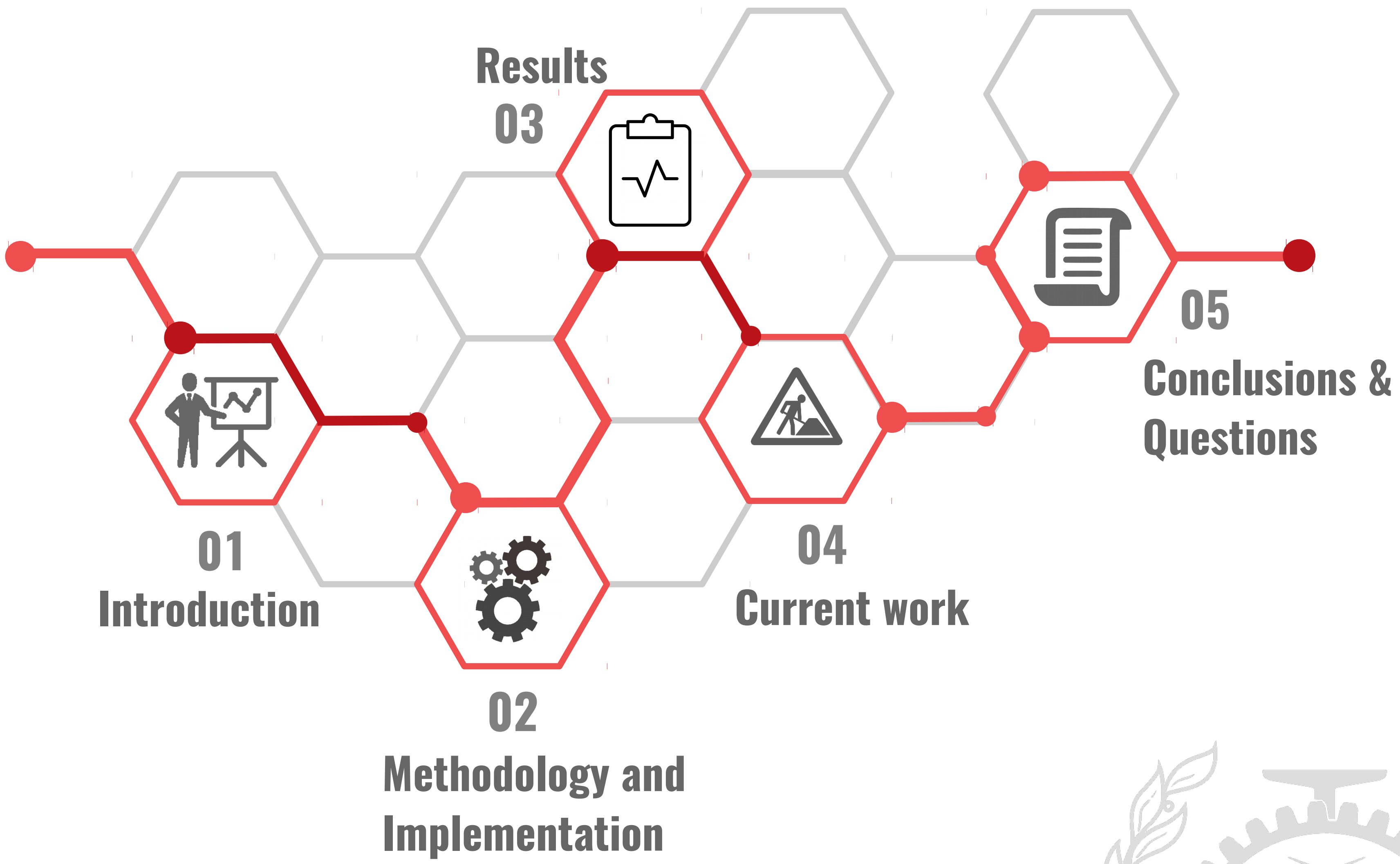
System performance anomaly detection using tracing data analysis

Iman Kohyarnejadfard
Prof. Daniel Aloise and Prof. Michel Dagenais

POLYTECHNIQUE
MONTREAL

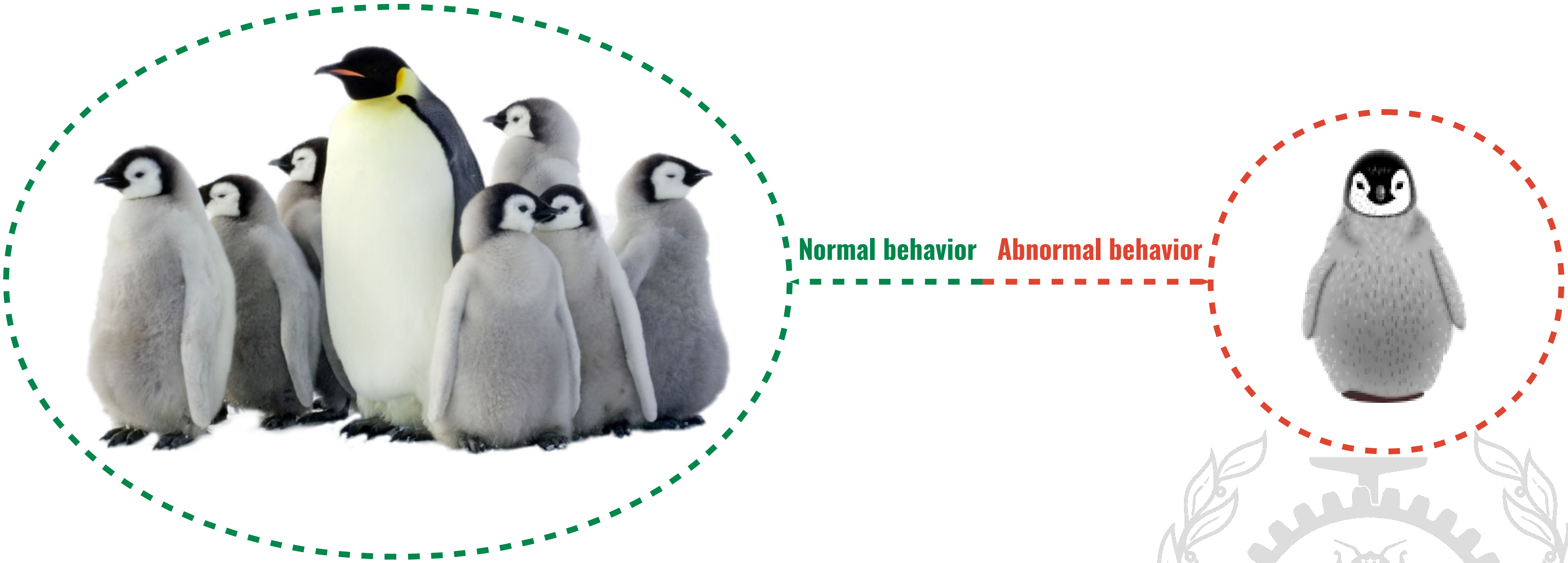
Agenda



Results
03
01
Introduction
02
Methodology and
Implementation
04
Current work
05
Conclusions &
Questions

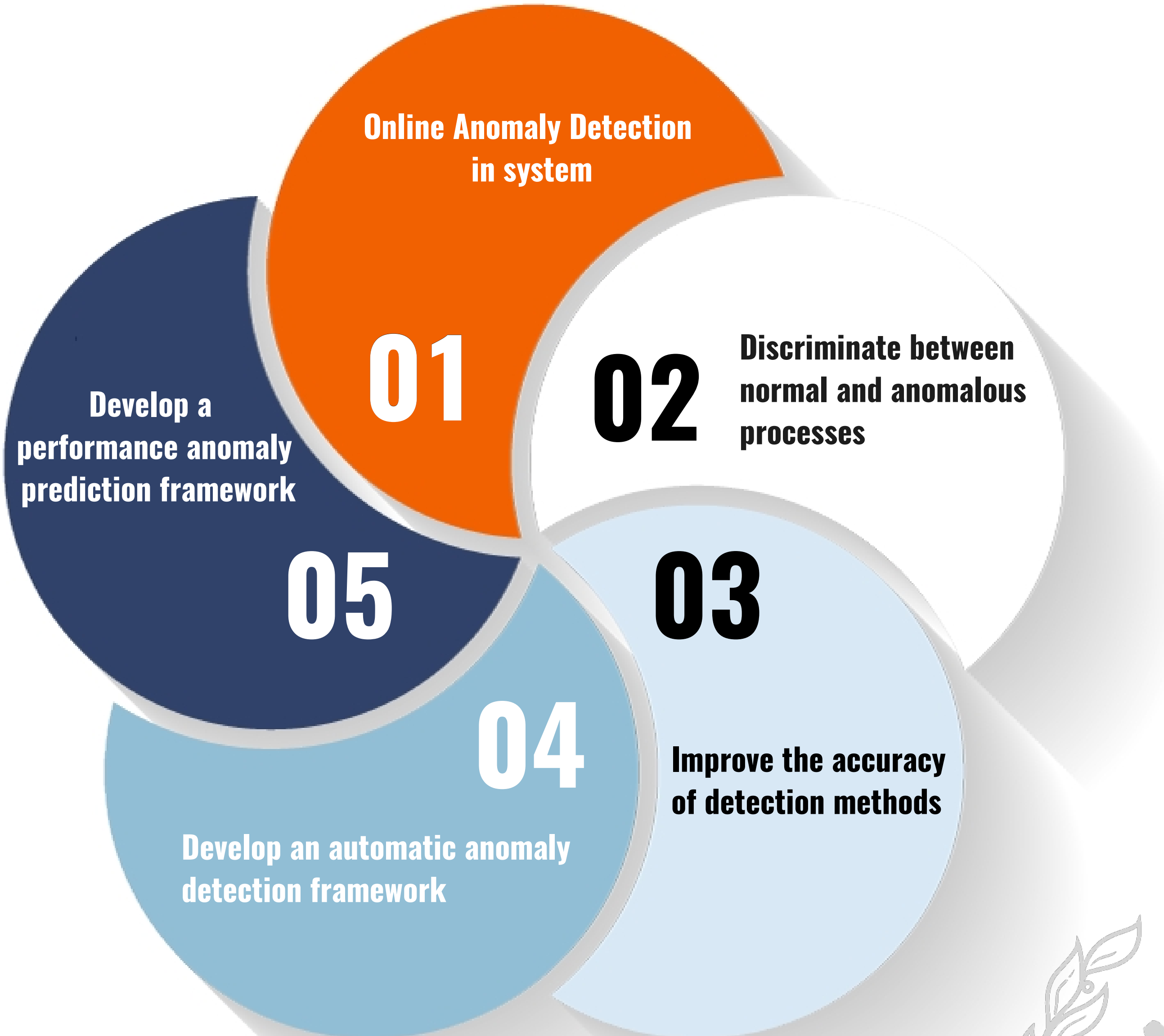
Anomaly Detection

- 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.

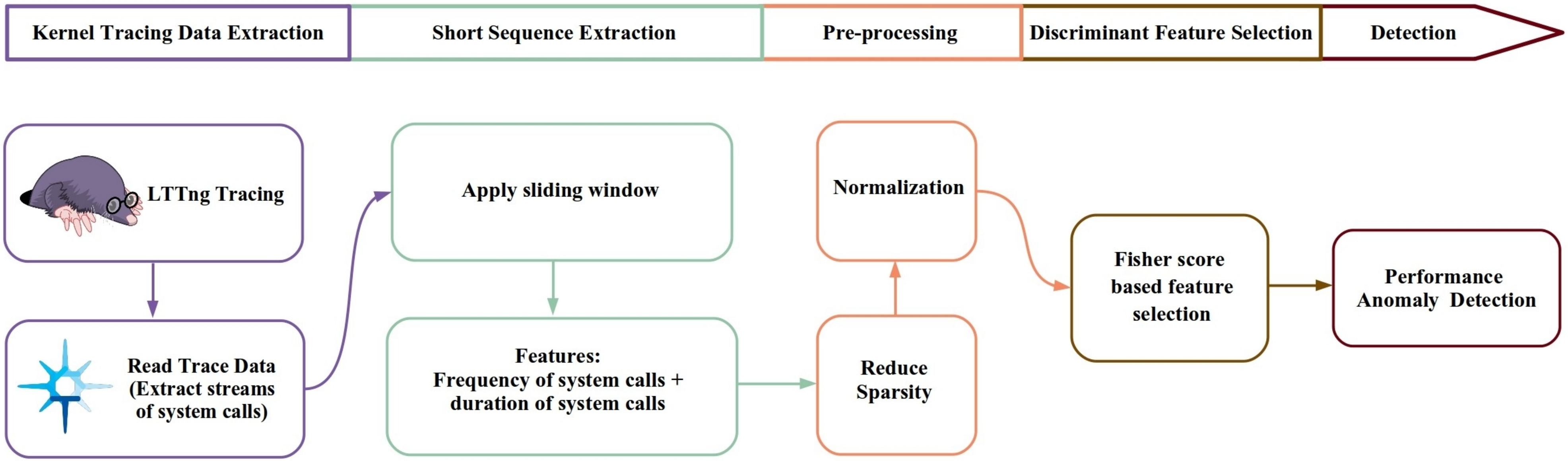
04 System call streams are enormous, and suitable to use in machine learning. A single process can produce thousands of system calls per second.

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.

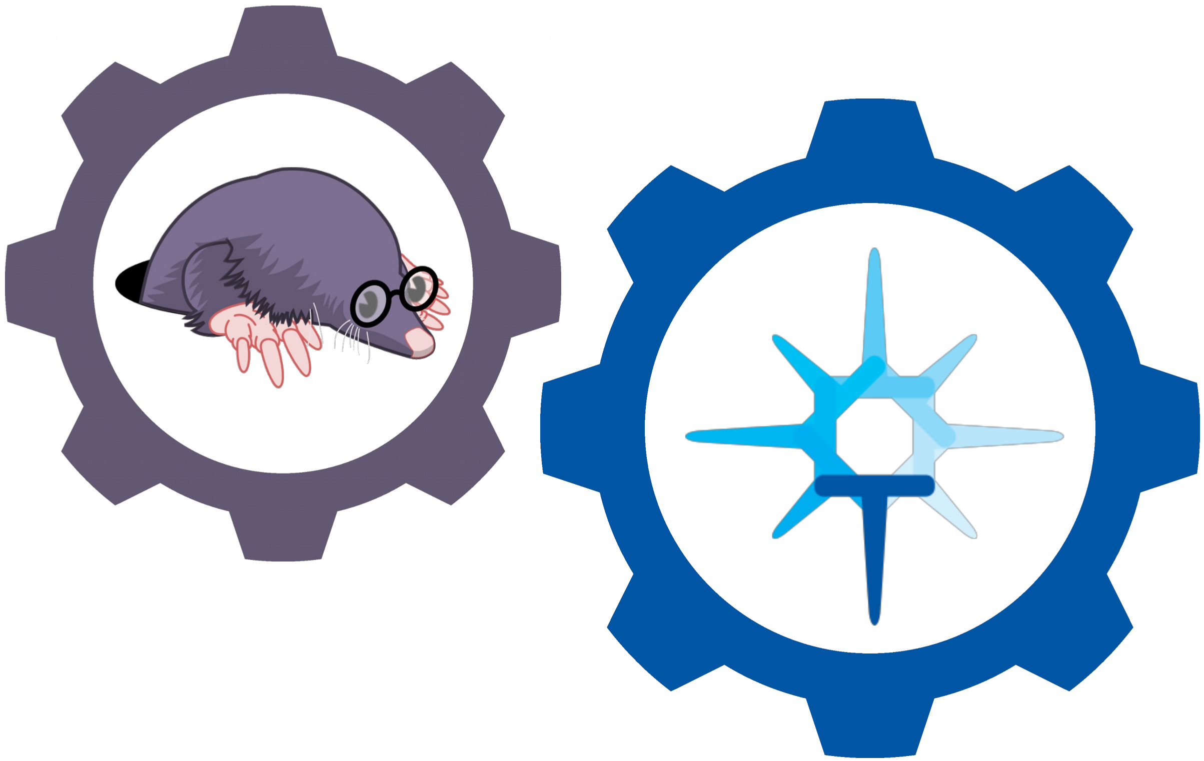
06 System call sequences can provide both momentary and temporal dynamics of process behavior.

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.

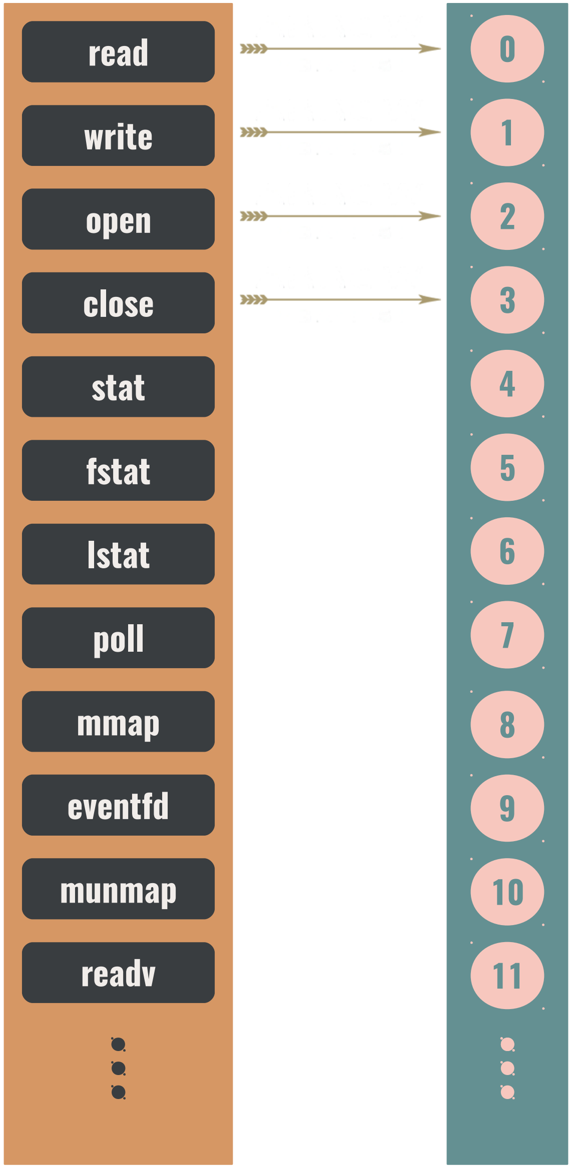


The benchmarking tool is run on virtual machines with different configurations and varying load on resources; LTTng is used to record the different tracing data streams.

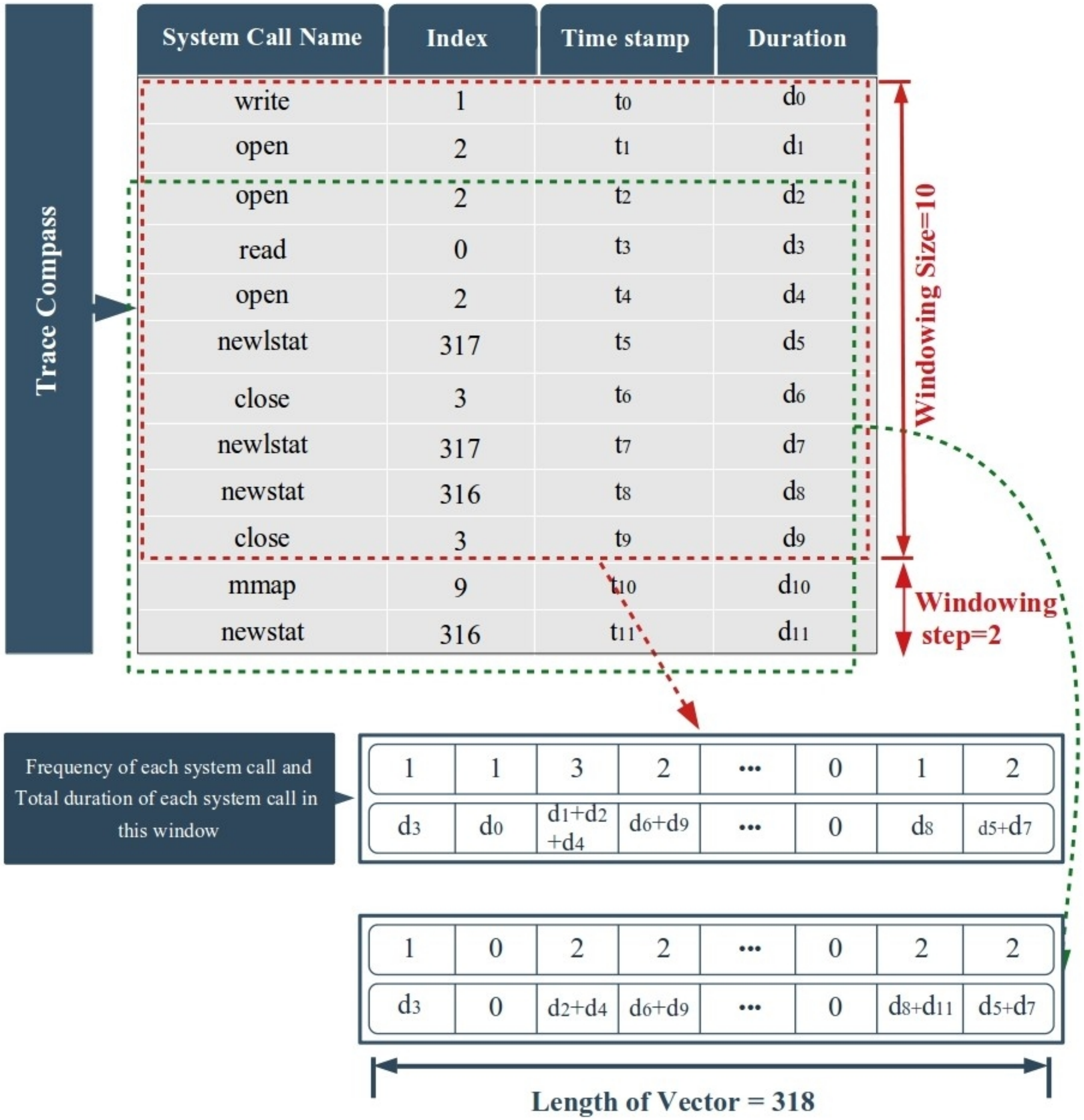


Trace compass is used to read tracing data, create tables of system calls and construct the initial vectors to use in the machine learning part.

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

Use Case 1

The open source MySQL synthetic benchmarks tool, Sysbench, with OLTP test (OnLine Transaction Processing) in complex mode.

A virtual machine with different workloads, such as:
I. (CPU problem) Setting the VM CPU cap too low (e.g., 1 CPU core, while running 8 threads of MySQL)
II. (Memory problem) Setting the memory cap 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

Use Case 2

The Chrome UnderStress 1.0 chrome extension is used to open, close, and refresh many light and heavy pages in Chrome with configurable speed.

Faults are simulated by running this Chrome extension on the VMs with different amount of CPU and memory resources.

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

18k normal and anomalous samples

Learning Module

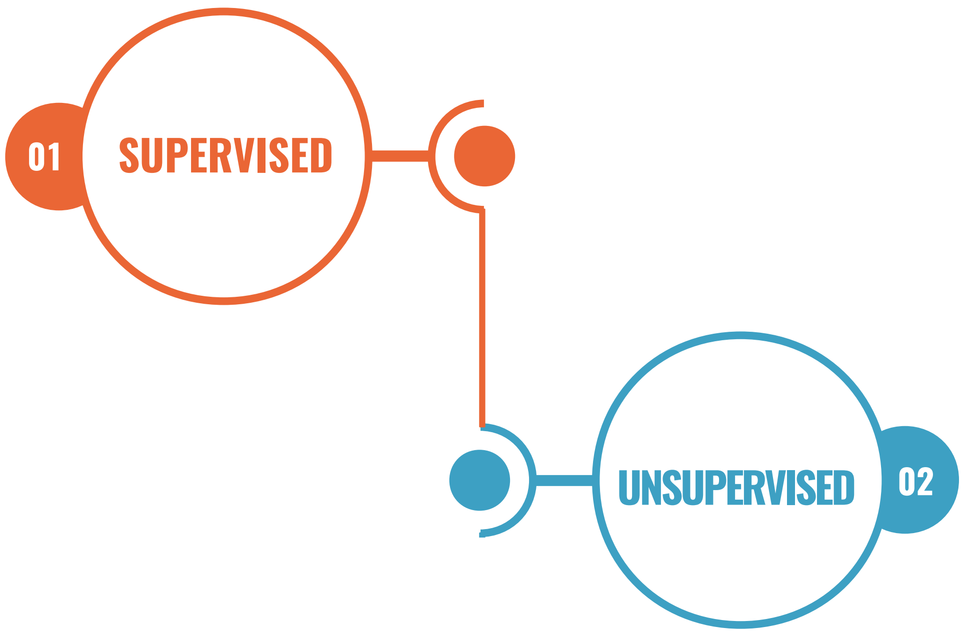
Supervised, unsupervised, and semi-supervised techniques can be used to detect performance anomalies in processes, depending on the volume of available labeled data.

01 SUPERVISED

first, we assume that enough labeled training samples are available.

Learning Module

Supervised, unsupervised, and semi-supervised techniques can be used to detect performance anomalies in processes, depending on the volume of available labeled data.

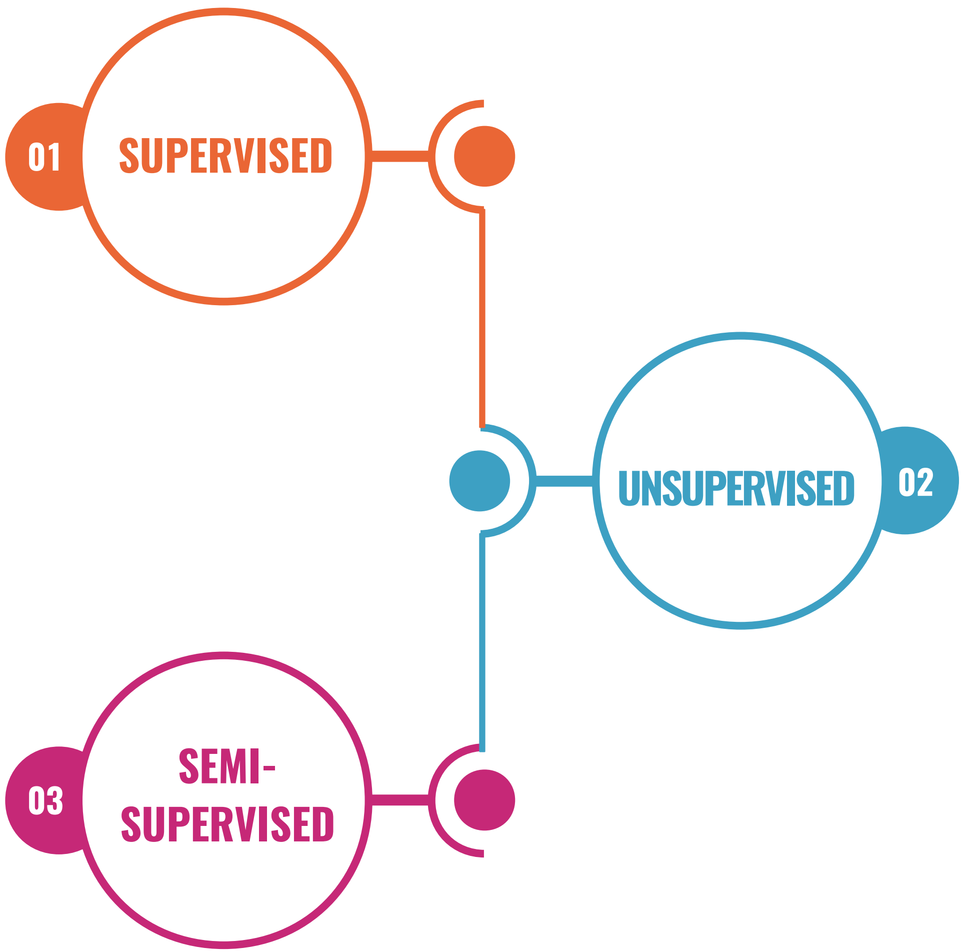


first, we assume that enough labeled training samples are available.

Since providing labeled data for the whole data distribution is not always possible, we propose to use an unsupervised approach.

Learning Module

Supervised, unsupervised, and semi-supervised techniques can be used to detect performance anomalies in processes, depending on the volume of available labeled data.



first, we assume that enough labeled training samples are available.

Since providing labeled data for the whole data distribution is not always possible, we propose to use an unsupervised approach.

However, unsupervised approaches usually present worse classification performance than supervised methods in practice given that no priori information is exploited. Therefore, in order to introduce a from of supervision into the unsupervised approach and improve the detection performance, we propose a semi-supervised approach.

Supervised Model

1 Reduce Sparsity

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

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 Feature Selection

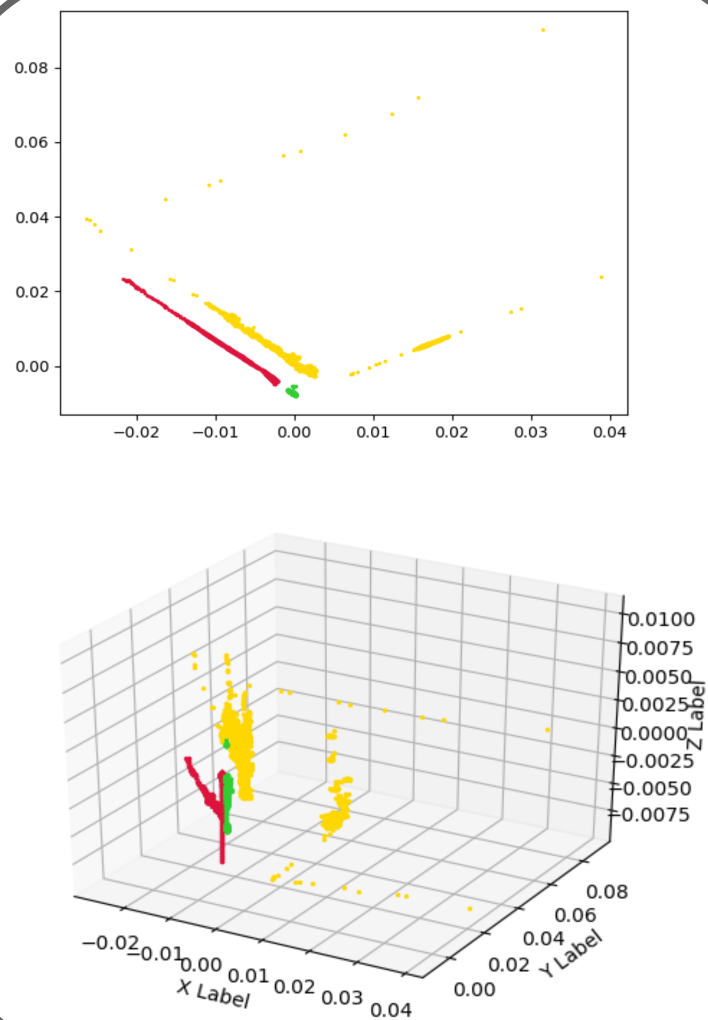
It selects each feature independently according to their scores under the Fisher criterion, which leads to a suboptimal subset of features.

4 Supervised Learning

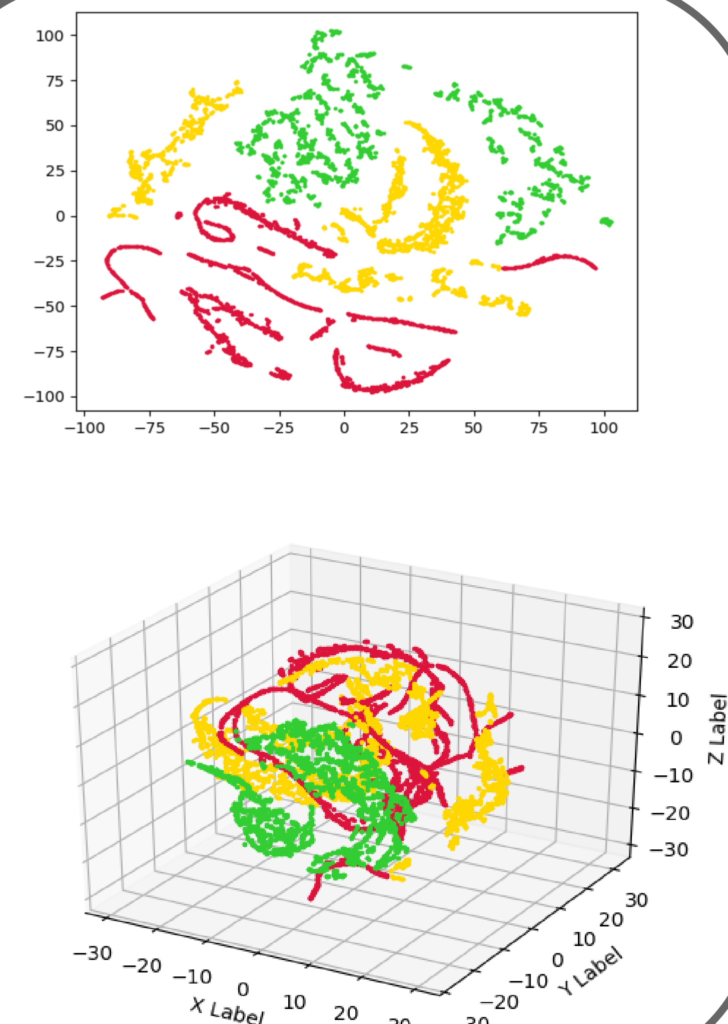
Supervised multi-class anomaly detection is done in this step using SVM with rbf kernel function.

Unsupervised Model → Reduce Sparsity → Normalization → K-Means

Projection of Dataset using PCA

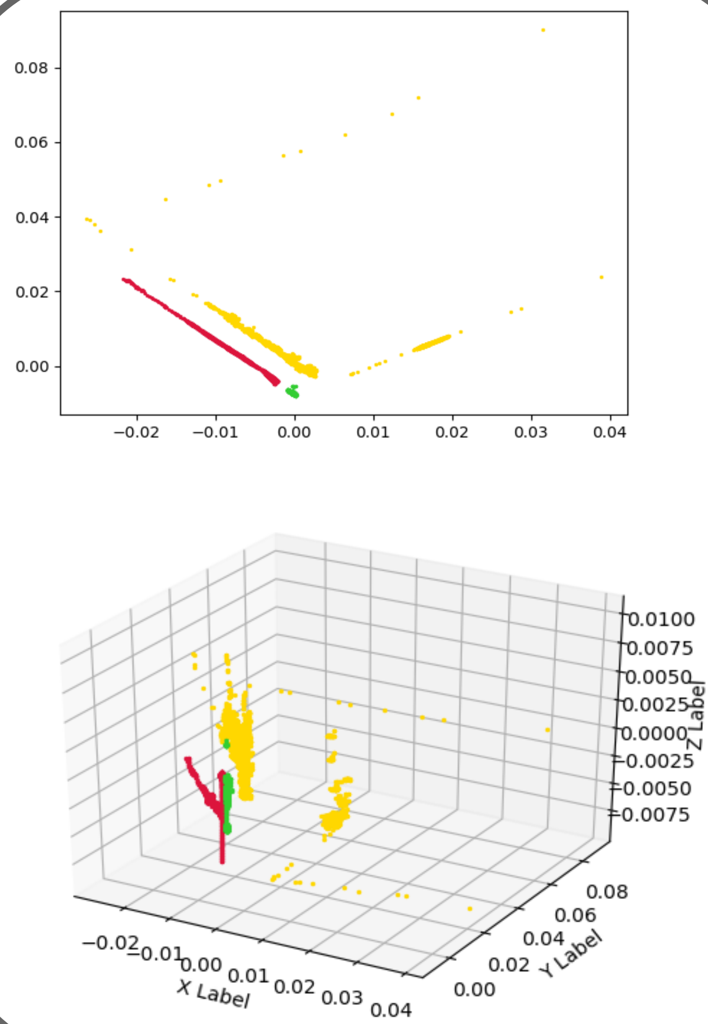


Projection of Dataset using TSNE

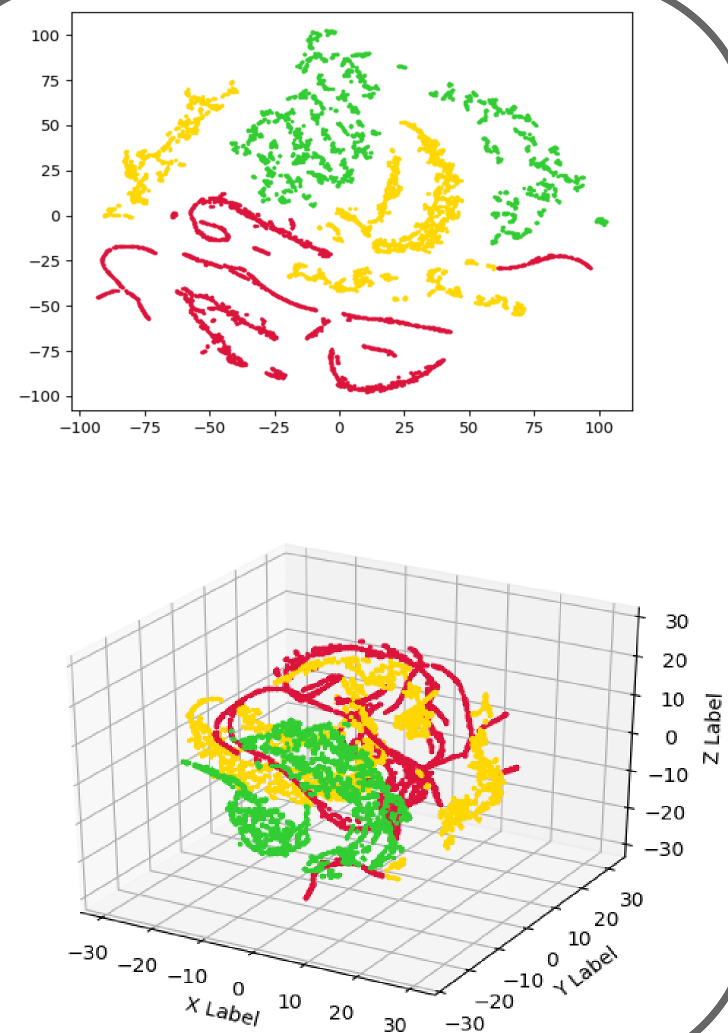


Unsupervised Model → Reduce Sparsity → Normalization → K-Means / DBSCAN

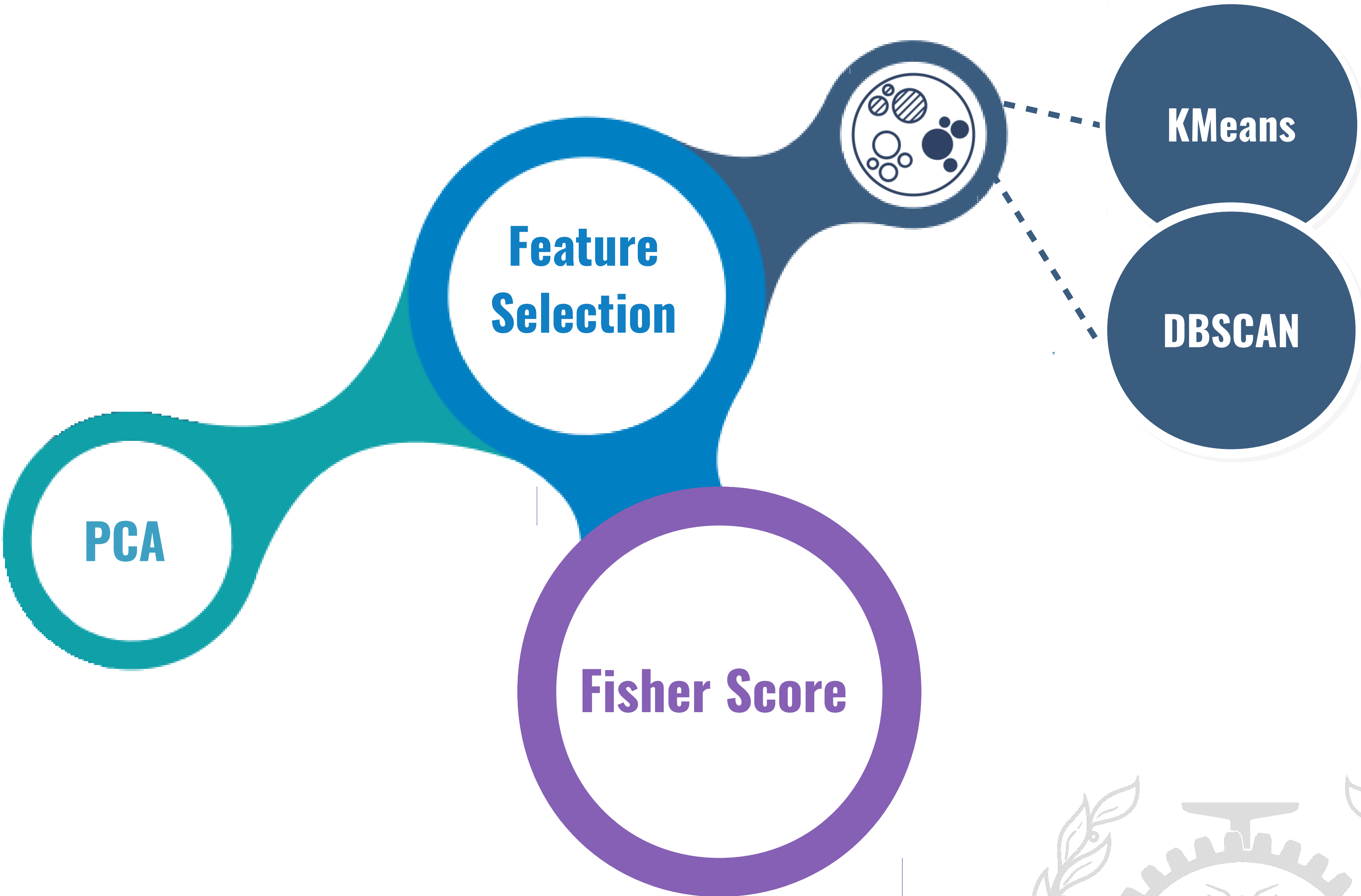
Projection of Dataset using PCA



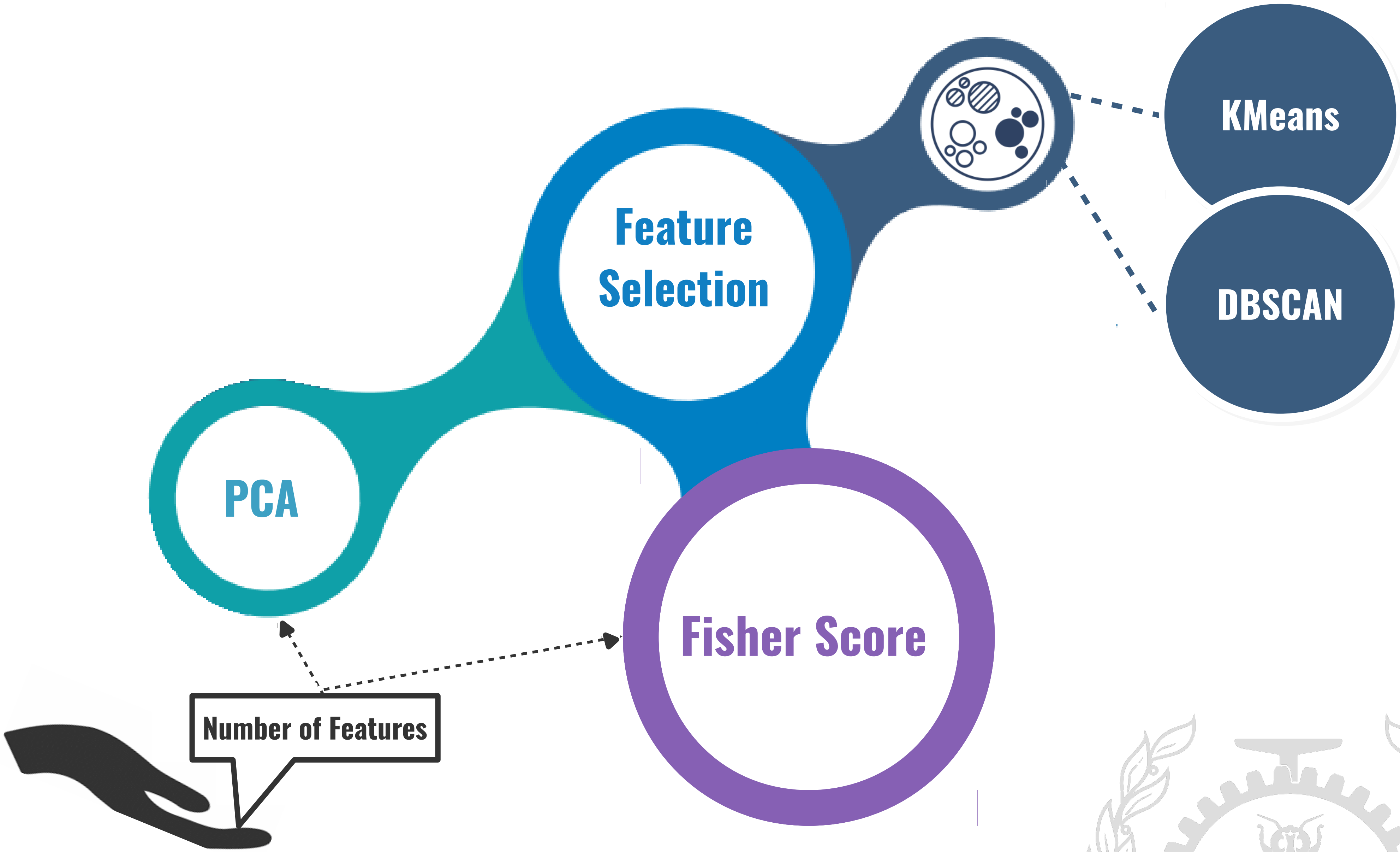
Projection of Dataset using TSNE

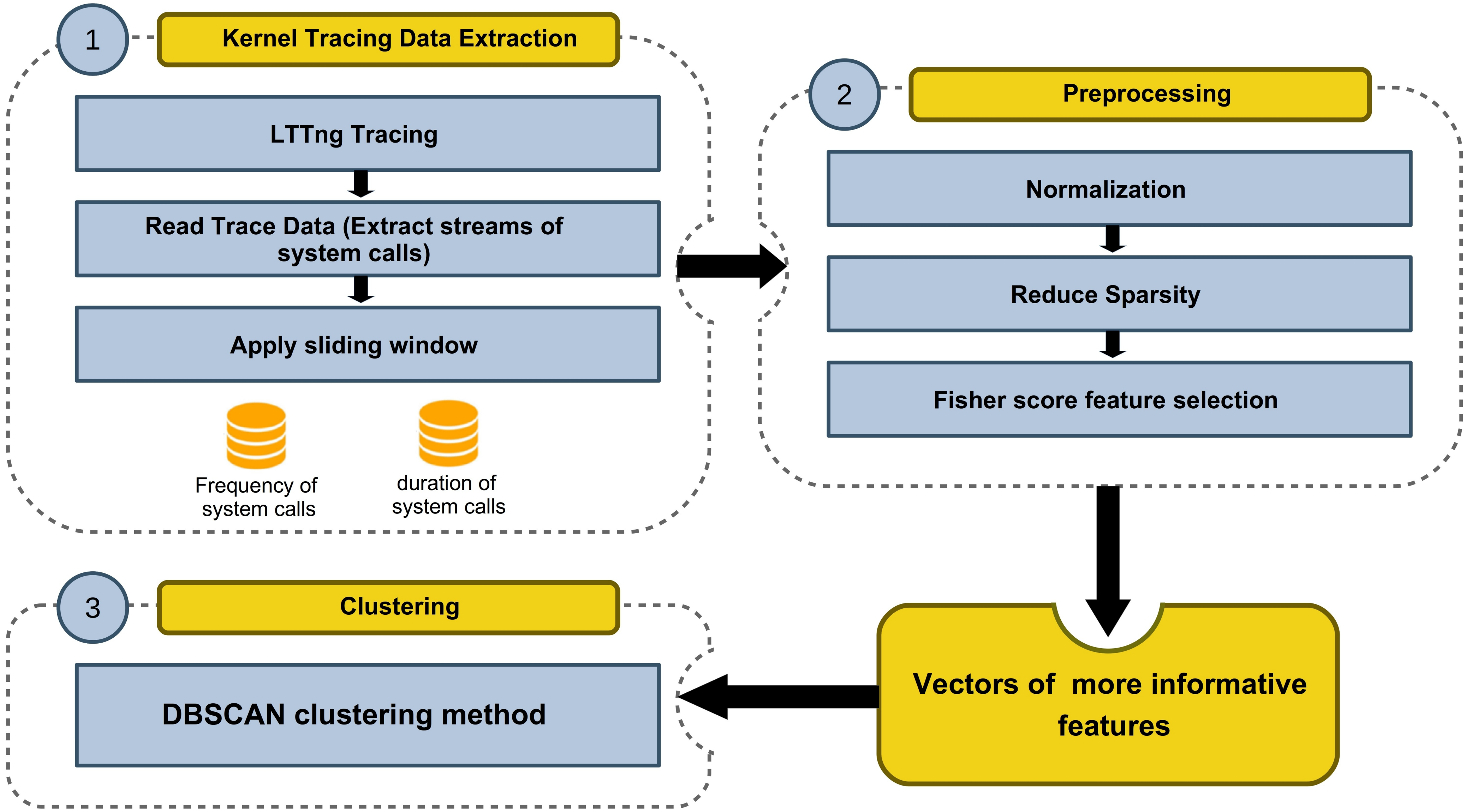


Semi-supervised Model



Semi-supervised Model

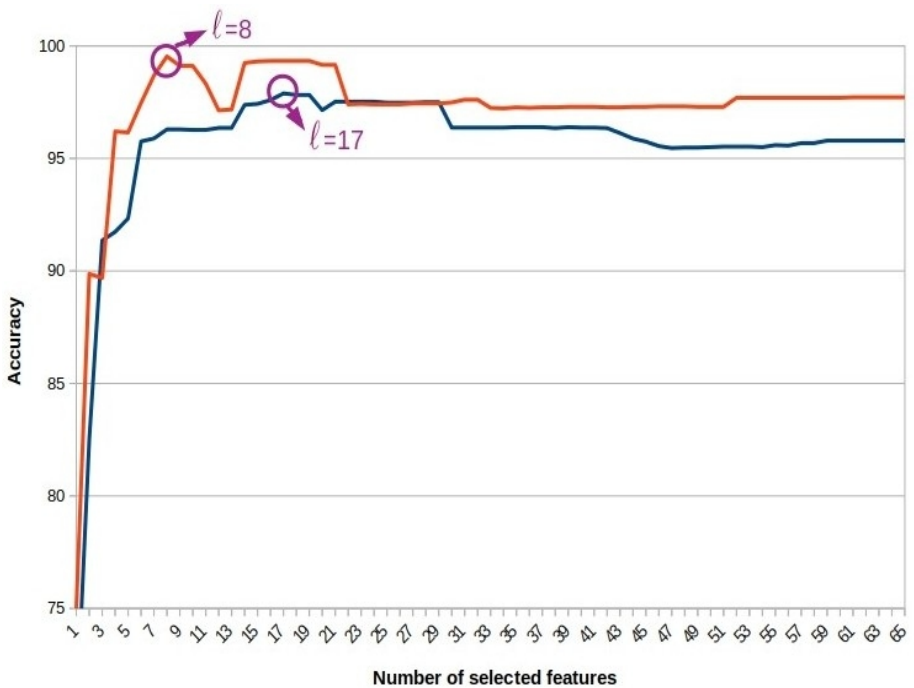




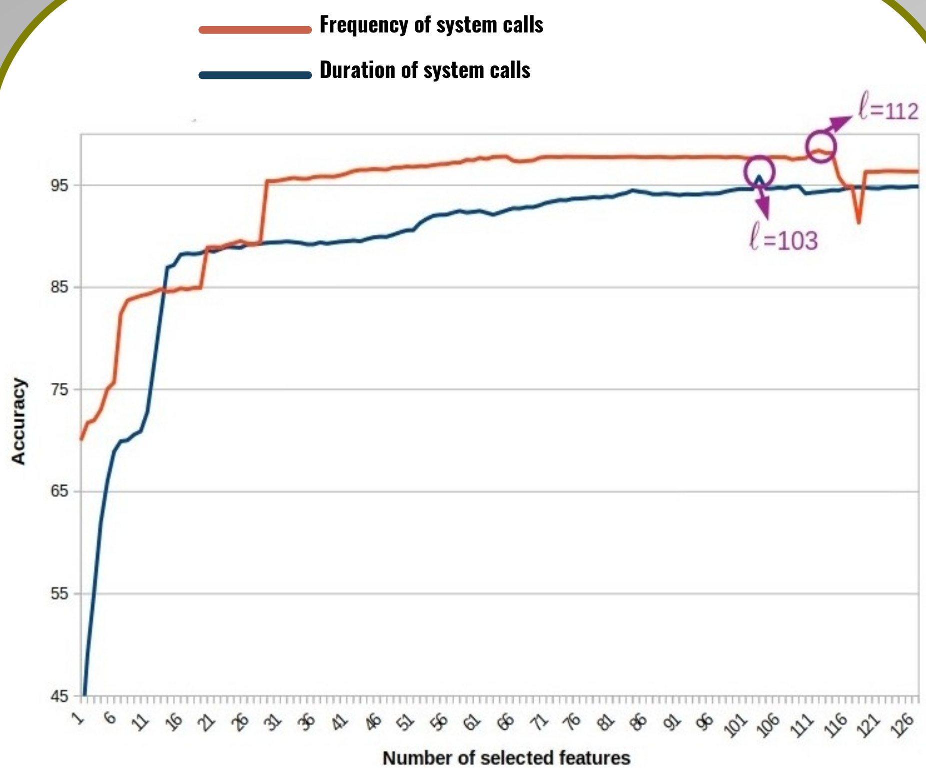
The architecture of the proposed Semi-supervised framework

RESULTS

Experimental results of the supervised method



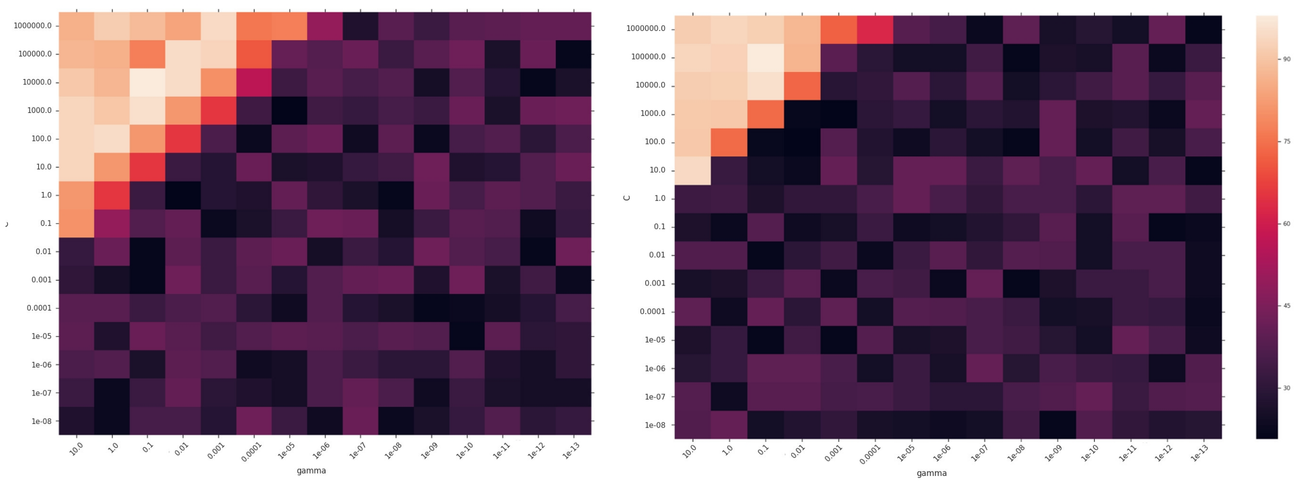
MySQL dataset



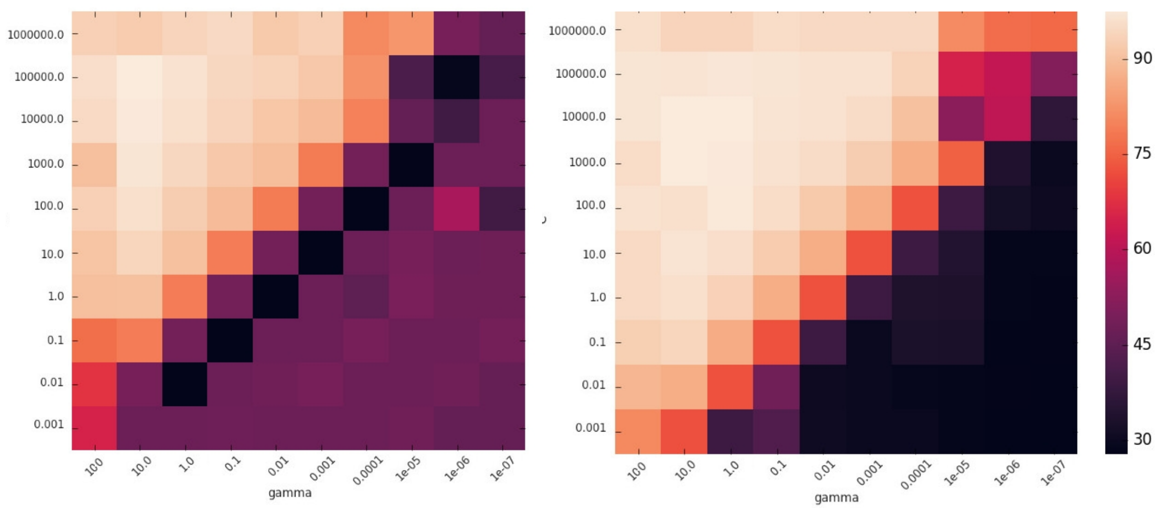
Chrome dataset

SVM-based anomaly detection accuracy versus the different number of top-ranked features;

Heat map of the duration-based and frequency-based supervised anomaly detection accuracy using different parameters γ and C for the Mysql dataset



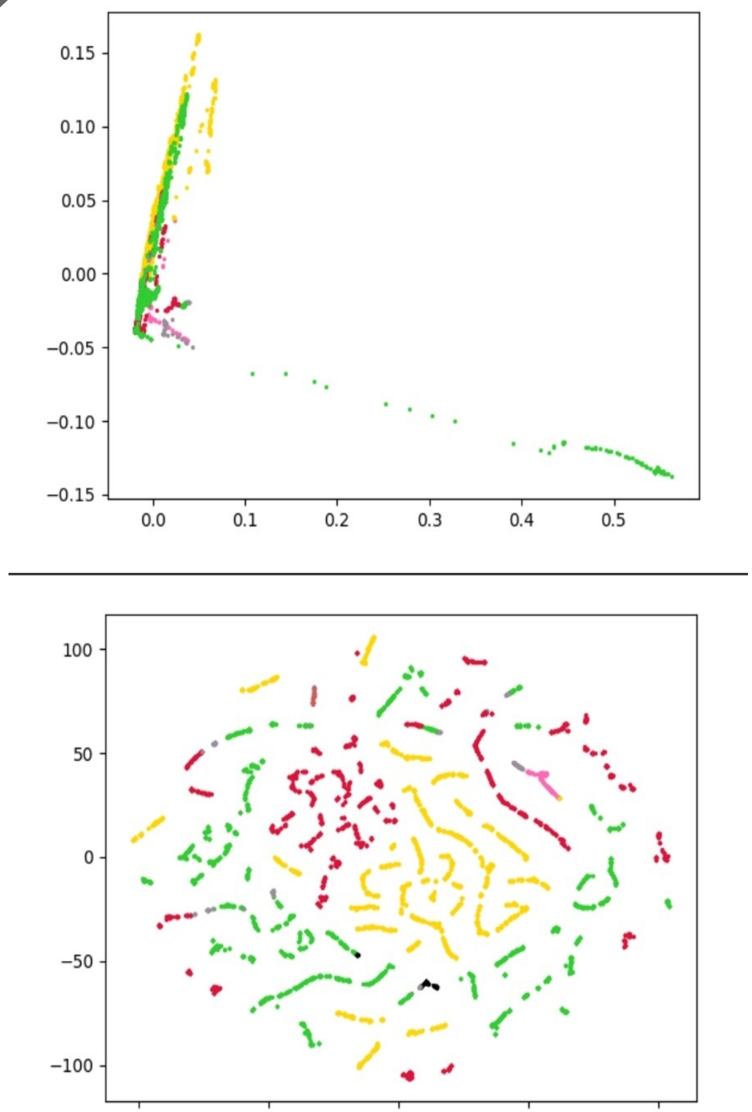
Heat map of the duration-based and frequency-based supervised anomaly detection accuracy using different parameters γ and C for the Chrome dataset



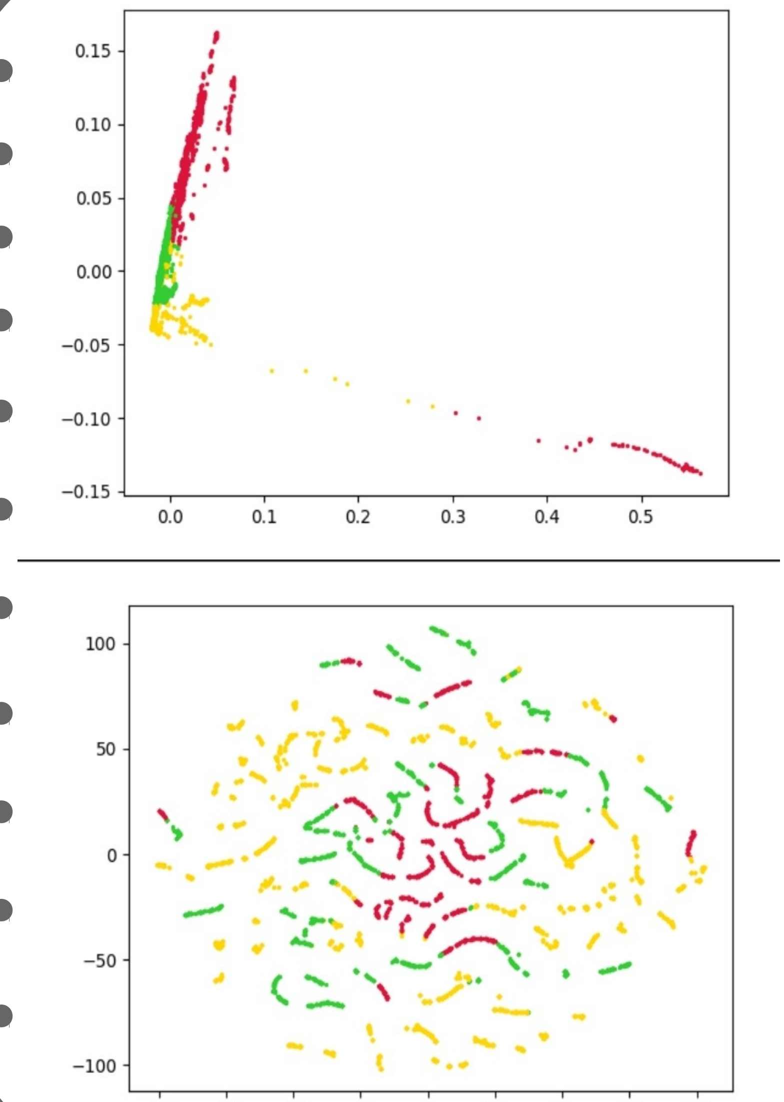
		Accuracy	Precision	Recall
MySQL Process	Frequency(ℓ=17)	0.928	0.989	0.968
	Duration(ℓ=8)	0.937	0.988	0.978
Chrome Process	Frequency(ℓ=103)	0.951	0.990	0.994
	Duration(ℓ=112)	0.959	0.991	0.985

The performance of the proposed supervised anomaly detection approach

The visual result of dbscan clustering after choosing l = 103 features with the highest fisher score



The visual result of K-Means clustering after choosing l = 103 features with the highest fisher score



DBSCAN

Validation of DBSCAN based semi-supervised technique on original features versus where the Fisher score feature selection method is applied

	Frequency-based Data set	ARI	Number of clusters	Duration-based Data set	ARI	Number of clusters
MySQL Process	Original Features($\epsilon = 10^{-3}$)	0.281	17	Original Features($\epsilon = 10^{-3}$)	0.278	18
	Fisher Score($\ell = 17$ and $\epsilon = 10^{-3}$)	0.874	8	Fisher Score($\ell = 8$ and $\epsilon = 10^{-3}$)	0.855	8
Chrome Process	Original Features($\epsilon = 5 \times 10^{-4}$)	0.254	21	Original Features($\epsilon = 10^{-3}$)	0.127	27
	Fisher Score($\ell = 103$ and $\epsilon = 5 \times 10^{-4}$)	0.823	9	Fisher Score($\ell = 112$ and $\epsilon = 10^{-3}$)	0.701	11

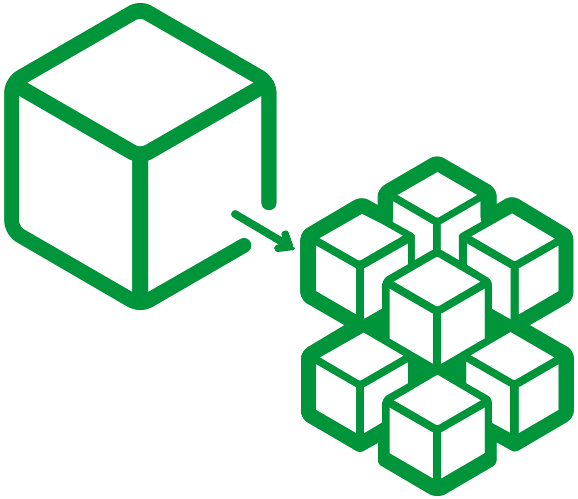
Current and future work

Analyzing the root cause of anomalies in microservice systems

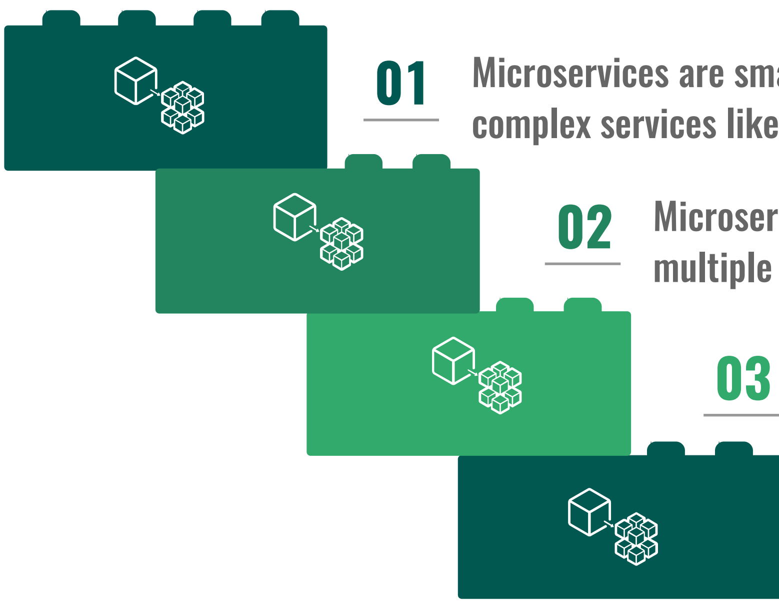
Learning system performance Anomalies using tracing data analysis

Anomaly detection in microservice systems using tracing data and Machine Learning

Anomaly detection in microservice systems using tracing data and Machine Learning



The concept of DevOps and agile approaches like microservice architectures and Continuous Integration becomes extremely popular since the need for flexible and scalable solutions increased.



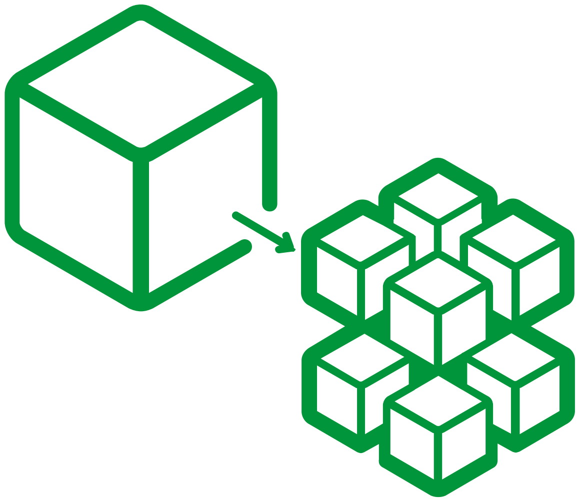
01 Microservices are small services that are interconnected with many other microservices to present complex services like web applications.

02 Microservices provide greater scalability and make distributing the application over multiple physical or virtual systems possible.

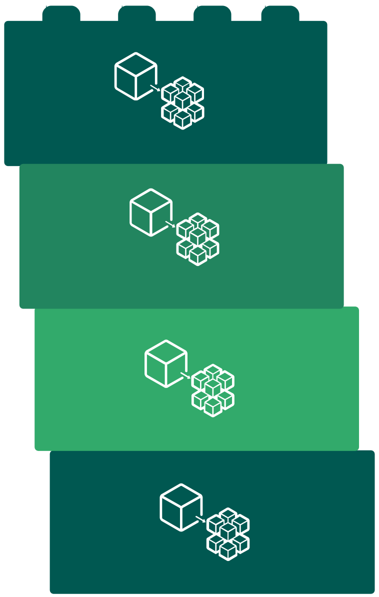
03 Microservices architecture tackles the problem of productivity and speed by decomposing applications into smaller services that are faster to develop and easier to manage; if one microservice fails, the others will continue to work.

04 Each microservice can be written using different technologies, and they enable continuous delivery.

Anomaly detection in microservice systems using tracing data and Machine Learning



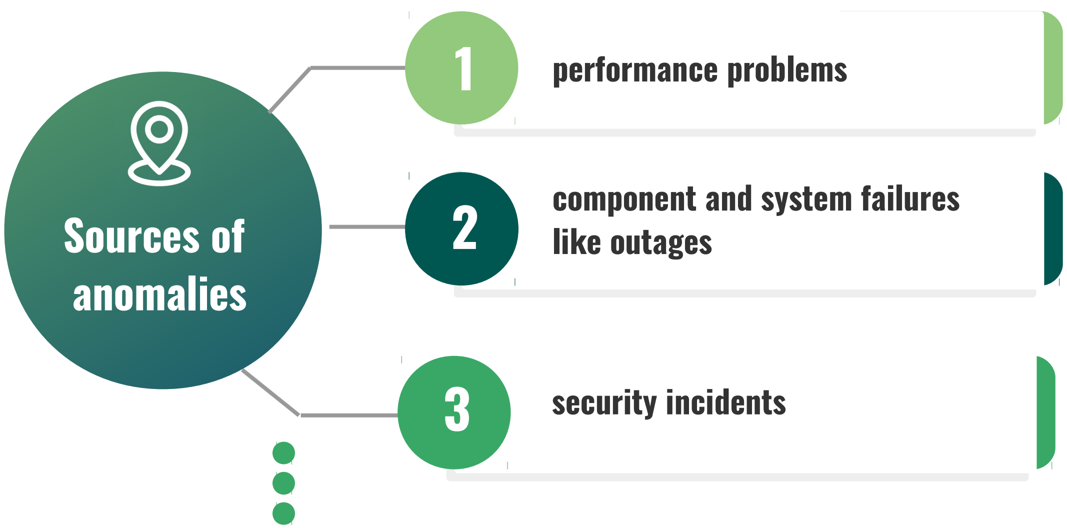
The concept of DevOps and agile approaches like microservice architectures and Continuous Integration becomes extremely popular since the need for flexible and scalable solutions increased.



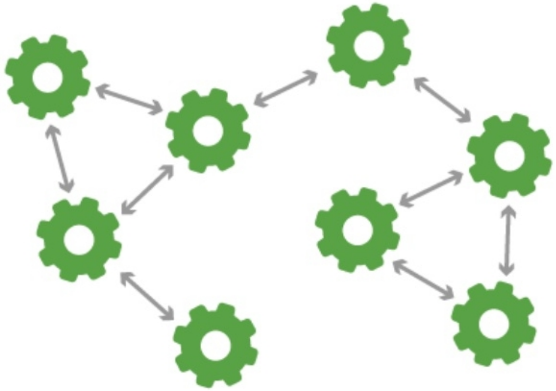
Despite all these benefits, by increasing the degree of automation and distribution, application performance monitoring becomes more challenging because microservices are possibly short-lived and may be replaced within seconds.

Hence new requirements in the way of anomaly detection have emerged as these changes could also be the cause of anomalies.

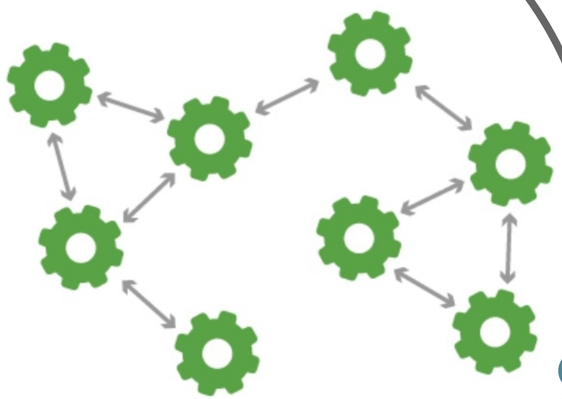
One of the essential parts of AIOps platforms is to detect the anomaly rapidly and decrease the reaction time before it leads to a service or system failure.



Microservices mostly use Representational State Transfer (REST) as a way to communicate with other microservices



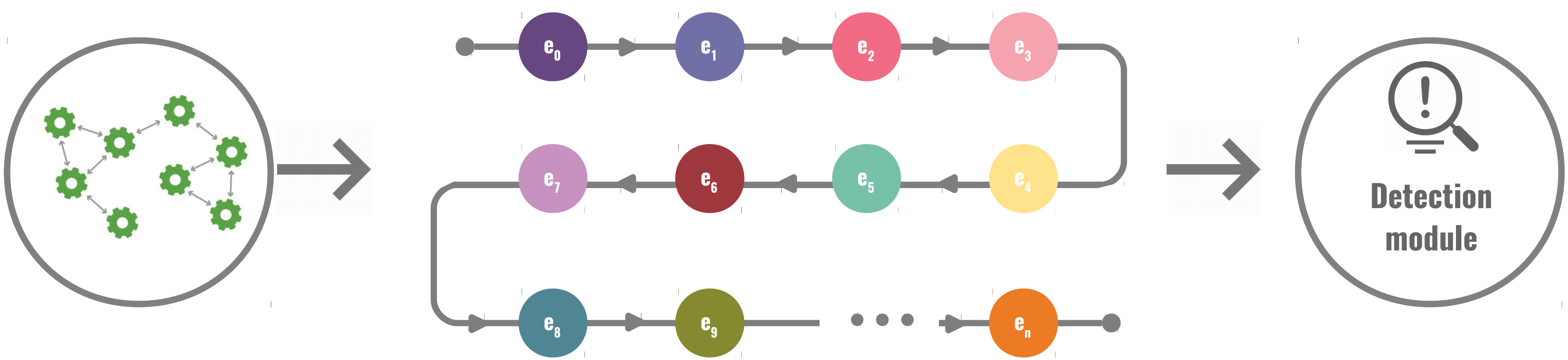
Microservices mostly use Representational State Transfer (REST) as a usual way to communicate with other microservices



1. LTTng is the primary tool we use in the data collection step.
2. Traces in microservice architectures are composed of events (spans).
3. An event contains information describing the state, performance, and further service characteristics at a given time t_i.
4. Similar to structures for natural languages, we can convert the tracing data into sequential data to utilize and improve analysis methods developed in these domains.

A trace T is represented as an enumerated collection of events sorted by the timestamps (e_0, e_1, ..., e_n).

Each event in the trace contains some attributes such as ID, parent ID, protocol, host address, return code, URL, response time, and timestamp.



In the detection phase, we use this sequential information to make a prediction and compare the predicted output against the observed value.

A limited number of events can be the result of an action. Therefore few of the possible events can appear as the next event in the sequence.

The LSTM network can be used in this part to learn the possible sequence of events and predict the next event.

The anomaly is reported if the event observed in the next timestep from the original sequence is different from the predicted one.

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.