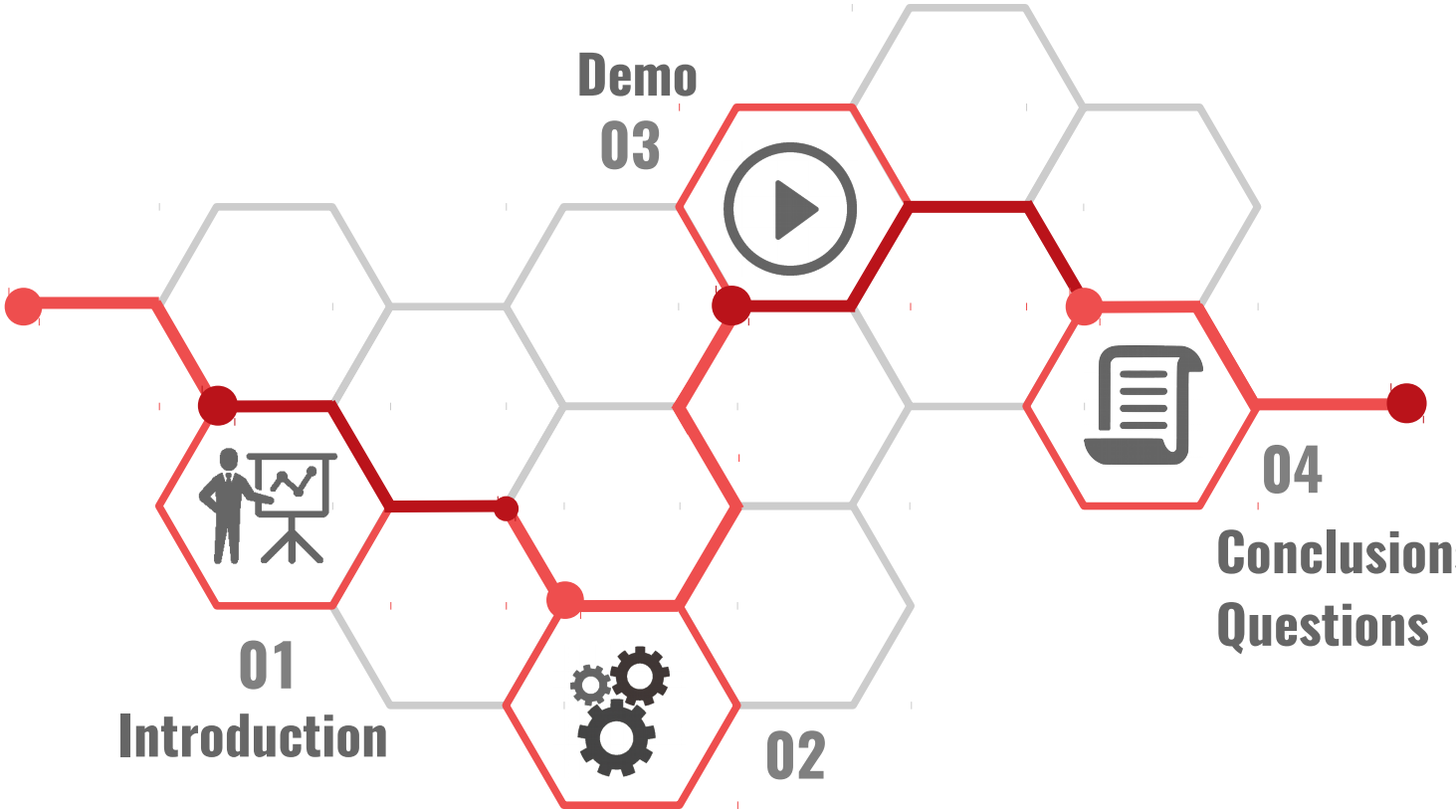
Anomaly Detection using streams of system calls

Iman Kohyarnejadfard
Prof. Daniel Aloise and Prof. Michel Dagenais

POLYTECHNIQUE
MONTREAL

Agenda


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

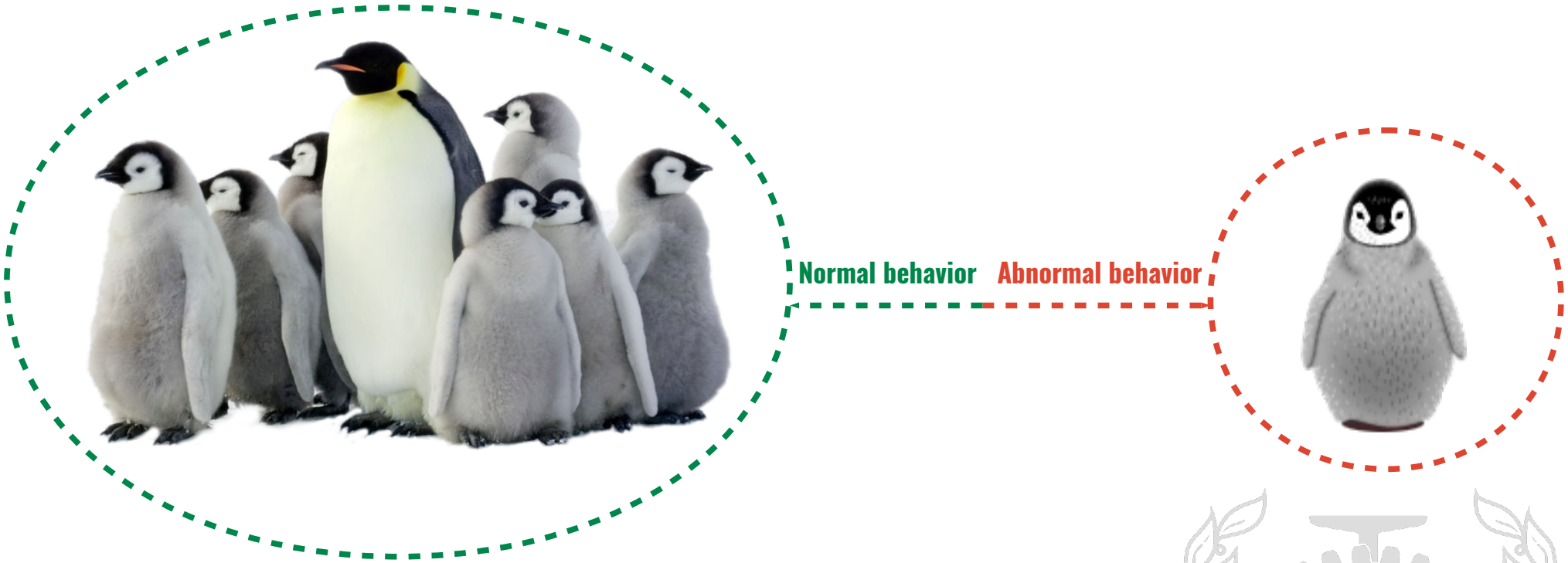
Anomaly

- 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.

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

Anomaly Detection

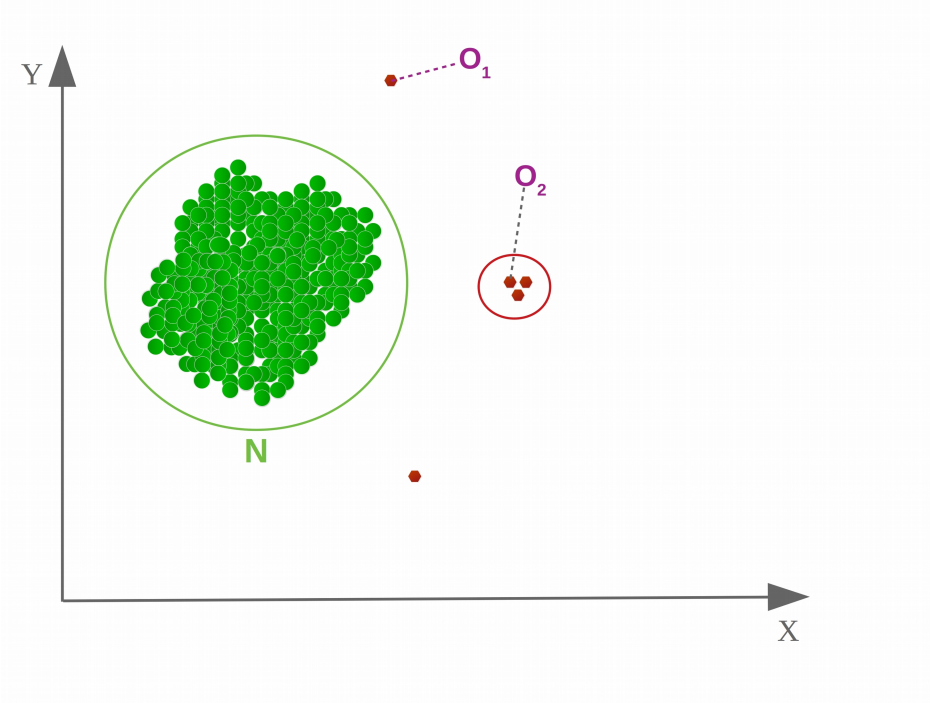
- Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior.
- Many anomaly detection techniques have been developed for various application domains.
- Anomalies in data translate to significant, and often critical, actionable information in a wide variety of application domains.



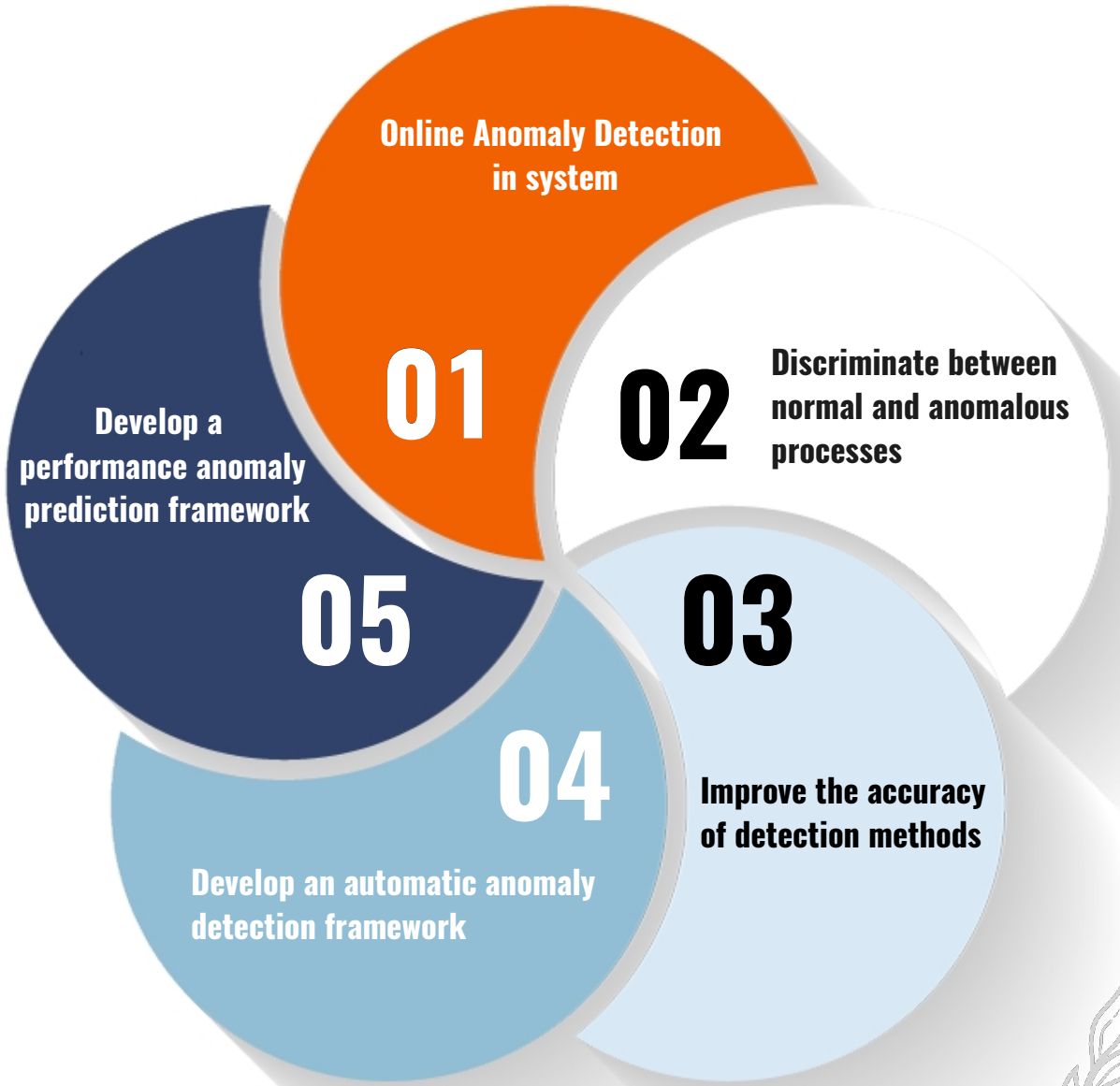
The figure retrieved from: https://pngtree.com

Anomaly Detection

- The data has a normal region: N
- Most observations lie in this region.
- Points that are sufficiently far away from this region, for example o_1 and o_2 are anomalies.



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

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

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

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

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.

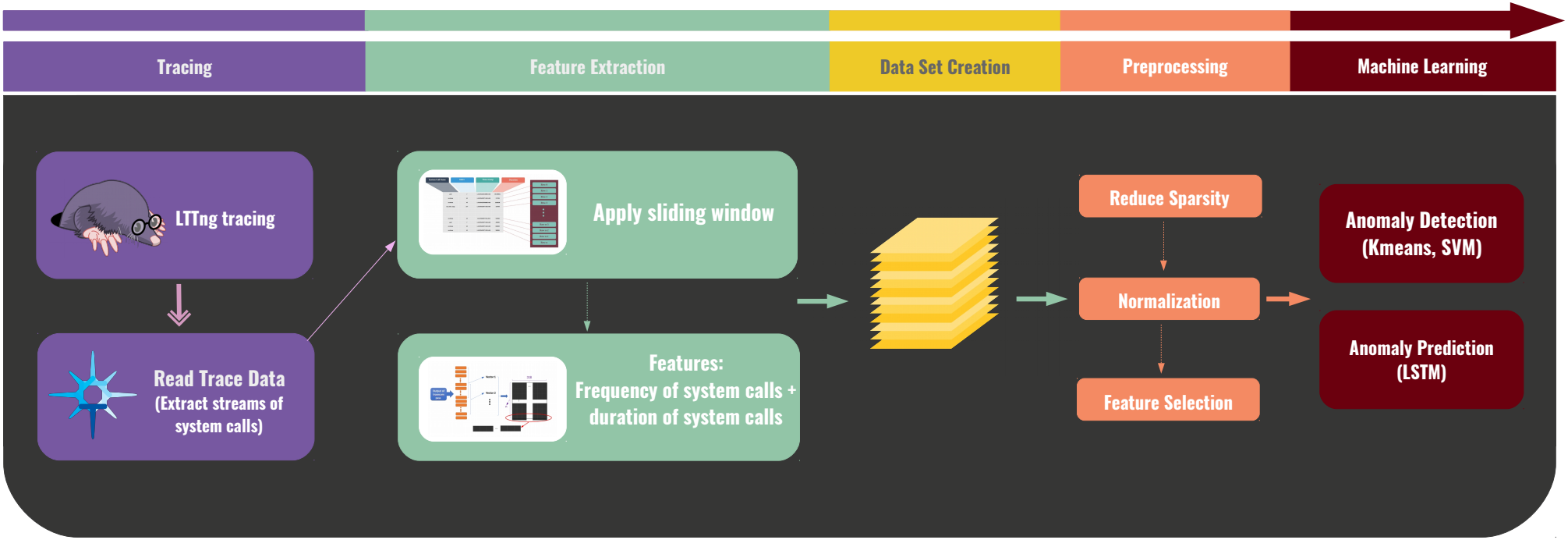
04 System call streams are enormous, and suitable to use in machine learning. A single process can produce thousands system calls per second.

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 part.
- 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

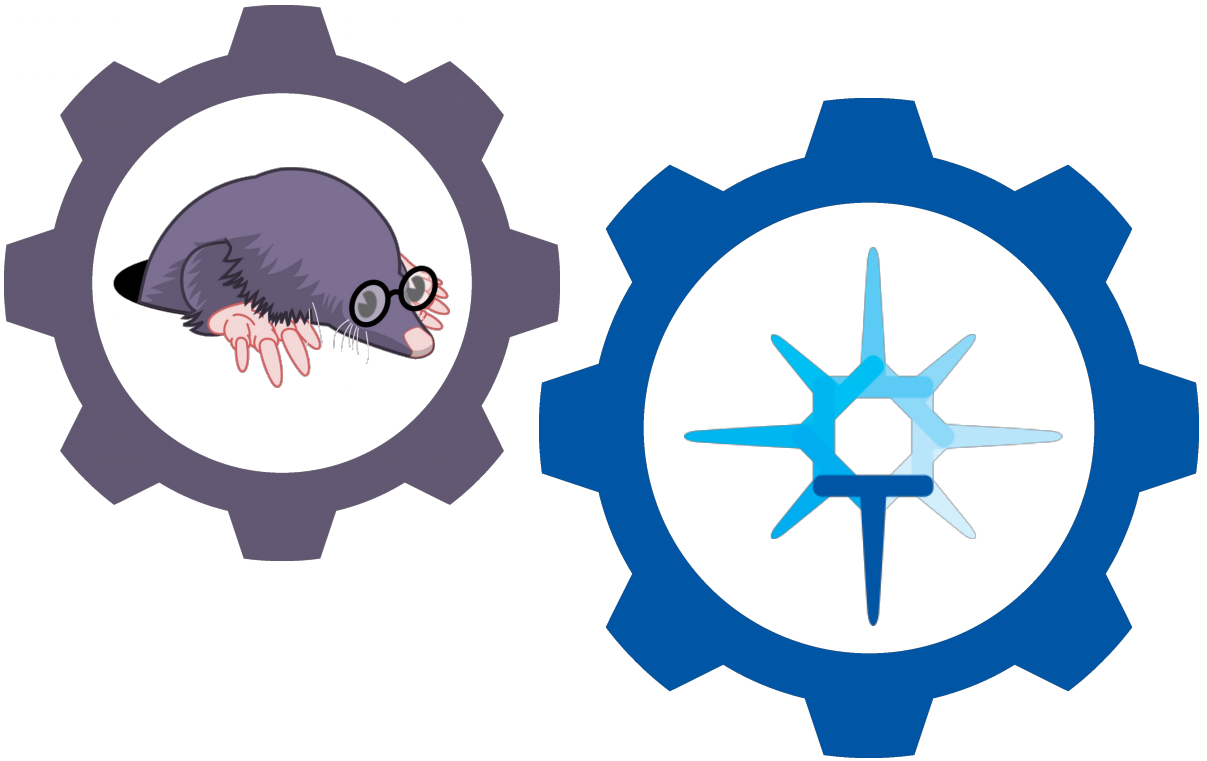
The open source MySQL synthetic benchmarks tool, Sysbench, with oltp test in complex mode.

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

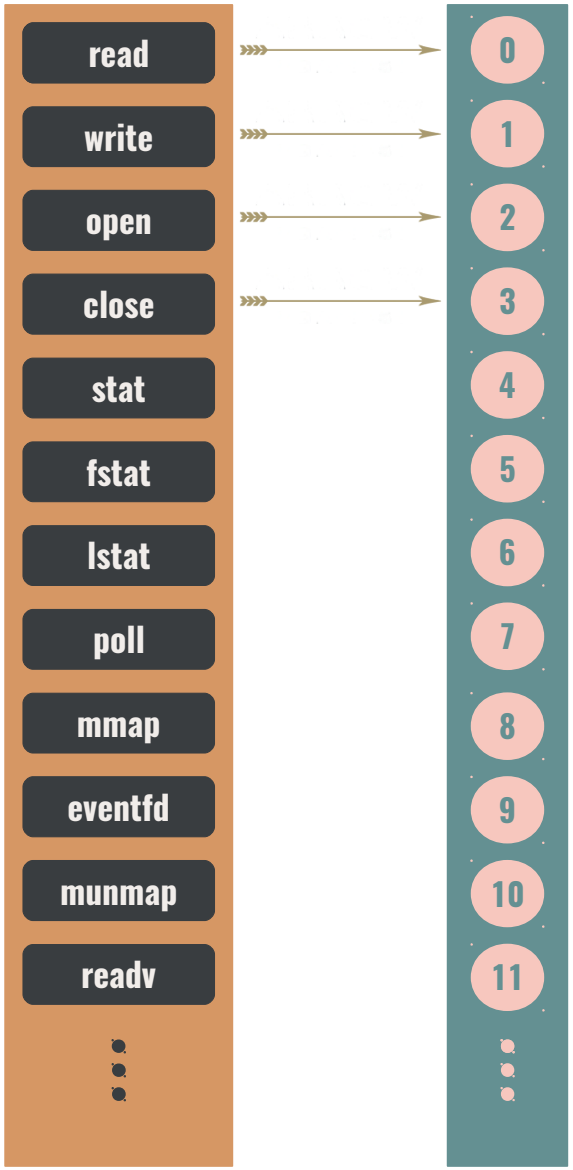
9000 normal samples vs 9000 anomalous ones (including Memory and CPU problem)

The benchmarking tool is run on virtual machines with different configurations and varying load on resources; LTTng is used to keep our multiple tracing data.

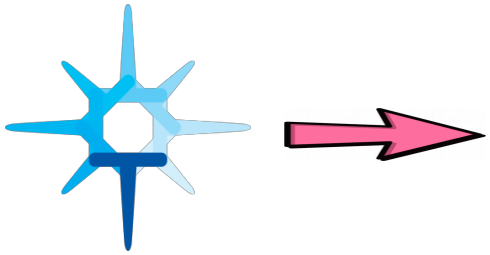


Trace compass is used to read tracing data, create tables of system calls and construct the initial vectors to use in machine learning part.

Indexes instead of names



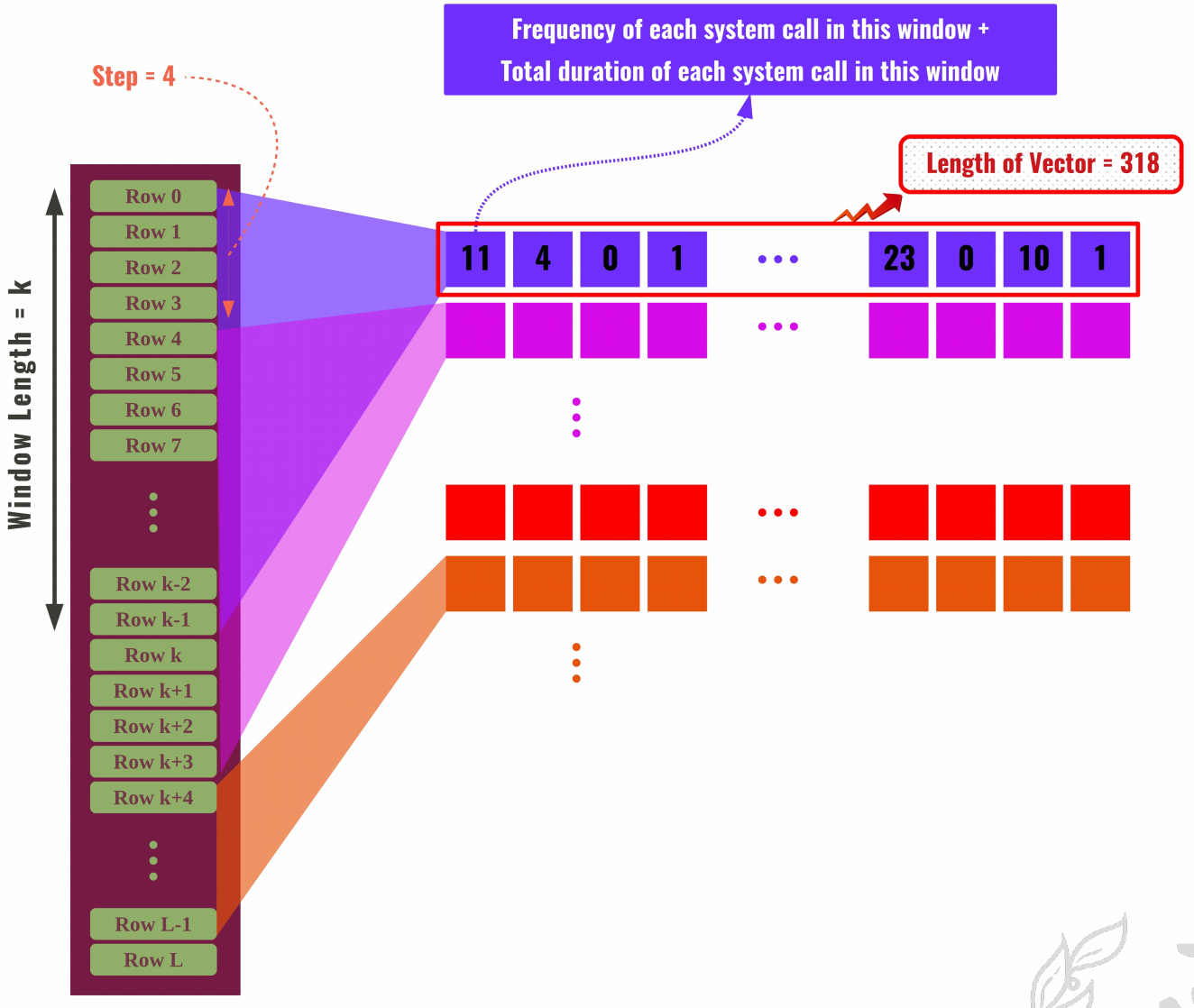
Read Trace



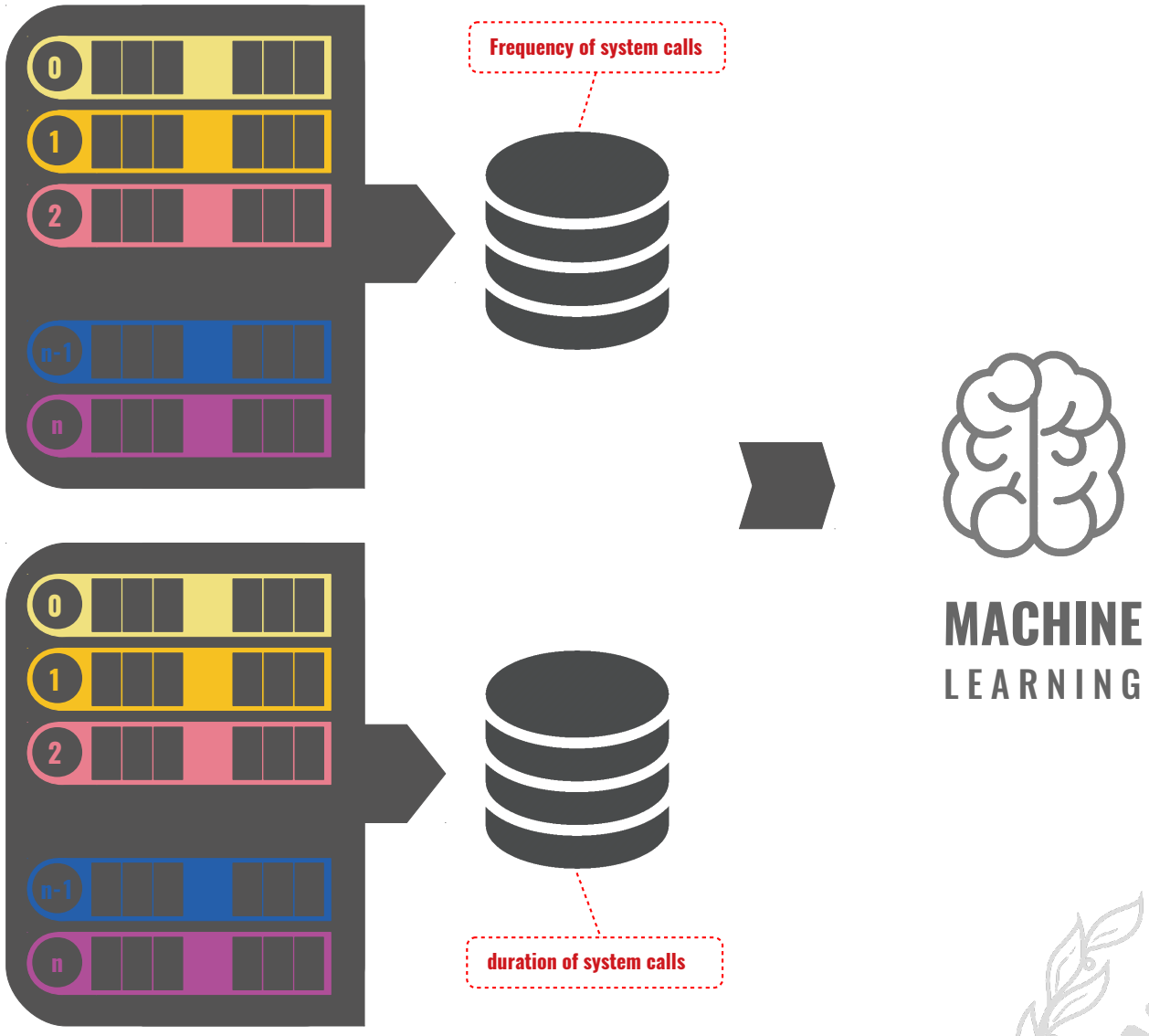
System Call Name	index	Time stamp	Duration
poll	7	1,54178425914609E+018	591390041
recvfrom	45	1,54178425977145E+018	175726
recvfrom	45	1,54178425976938E+018	2262238
sync_file_range	277	1,54178425977153E+018	126748
recvfrom	45	1,5417842597714E+018	534946
poll	7	1,54178425977154E+018	459203
recvfrom	45	1,54178425977145E+018	605839
recvfrom	45	1,54178425977155E+018	569994

Row 0
Row 1
Row 2
Row 3
Row n-3
Row n-2
Row n-1
Row n

Windowing


Frequency of each system call in this window +
Total duration of each system call in this window
Step = 4
Length of Vector = 318
Window Length = k
Row 0
Row 1
Row 2
Row 3
Row 4
Row 5
Row 6
Row 7
Row k-2
Row k-1
Row k
Row k+1
Row k+2
Row k+3
Row k+4
Row L-1
Row L
11
4
0
1
23
0
10
1

Data Set Creation



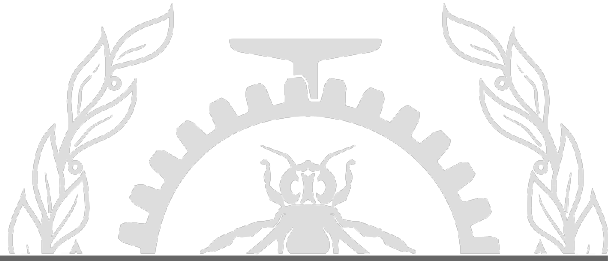
Preprocessing

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

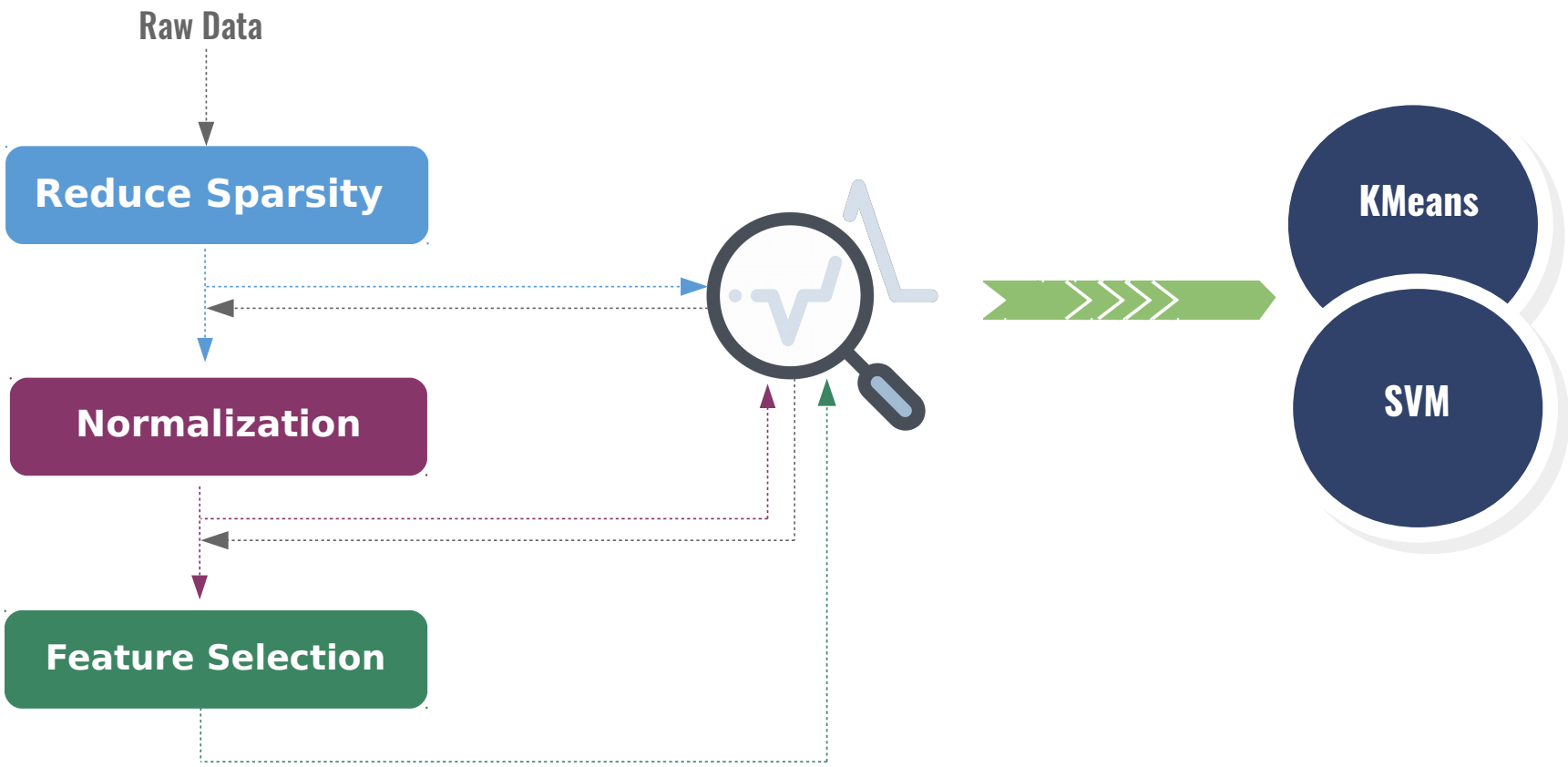
2. **Normalization**
   The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

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.

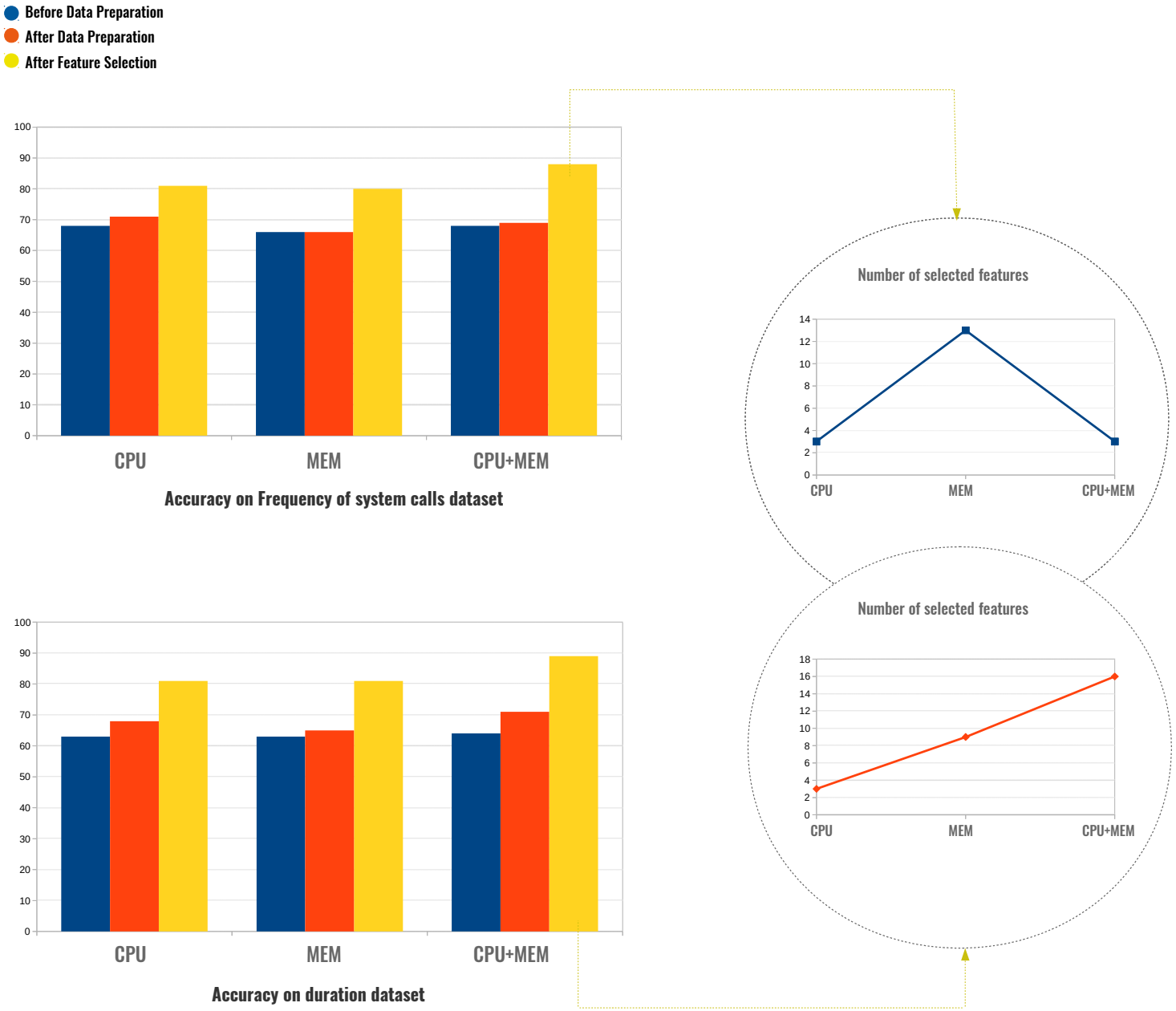
4. **Fisher score**
   It selects each feature independently according to their scores under the Fisher criterion, which leads to a suboptimal subset of features.



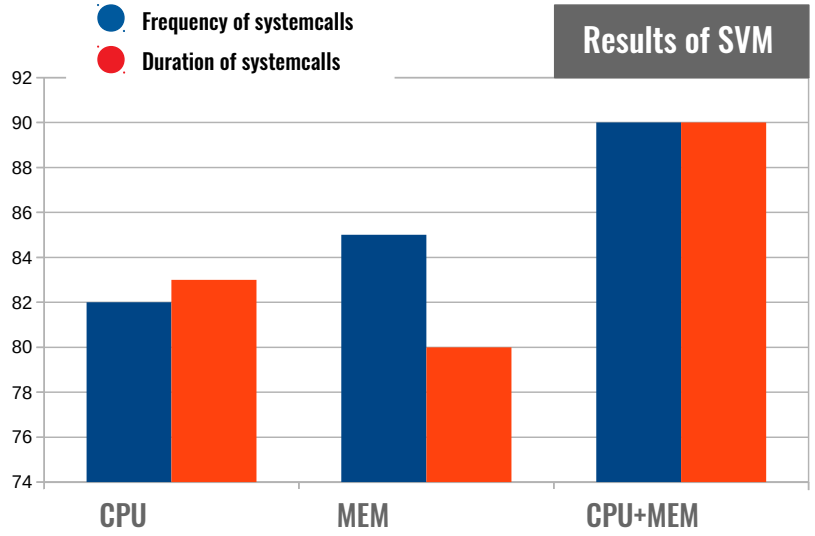
Learning part



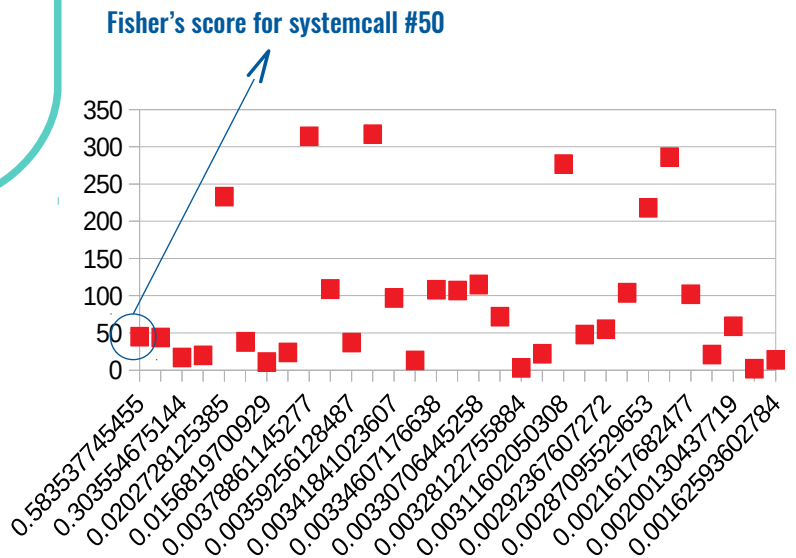
Results

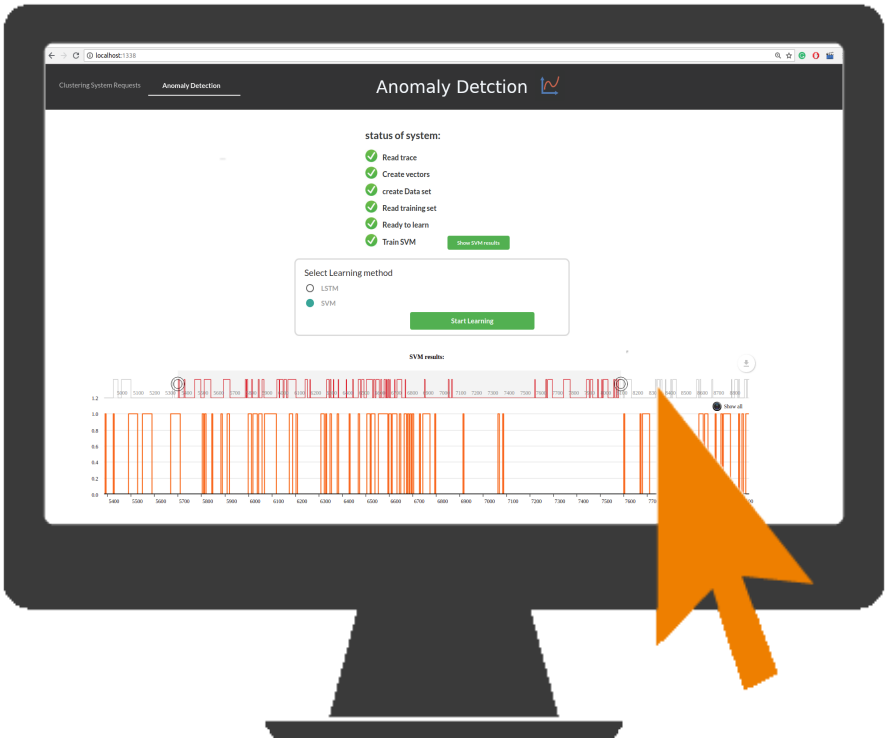

Before Data Preparation
After Data Preparation
After Feature Selection
100
90
80
70
60
50
40
30
20
10
0
CPU
MEM
CPU+MEM
Accuracy on Frequency of system calls dataset
Number of selected features
14
12
10
8
6
4
2
0
CPU
MEM
CPU+MEM
100
90
80
70
60
50
40
30
20
10
0
CPU
MEM
CPU+MEM
Accuracy on duration dataset
Number of selected features
18
16
14
12
10
8
6
4
2
0
CPU
MEM
CPU+MEM

Results

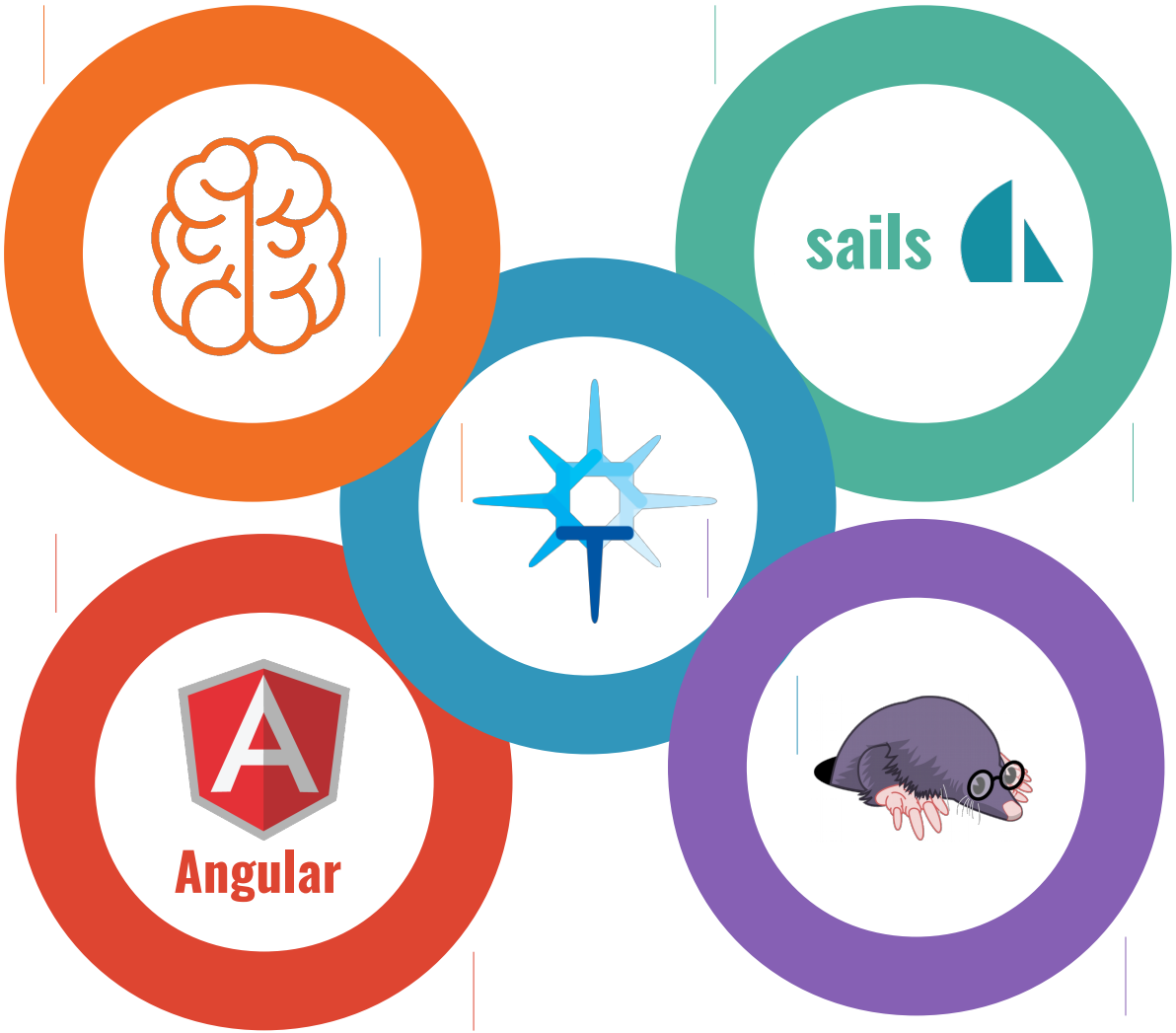


30 high values of fisher score for frequency of systemcalls

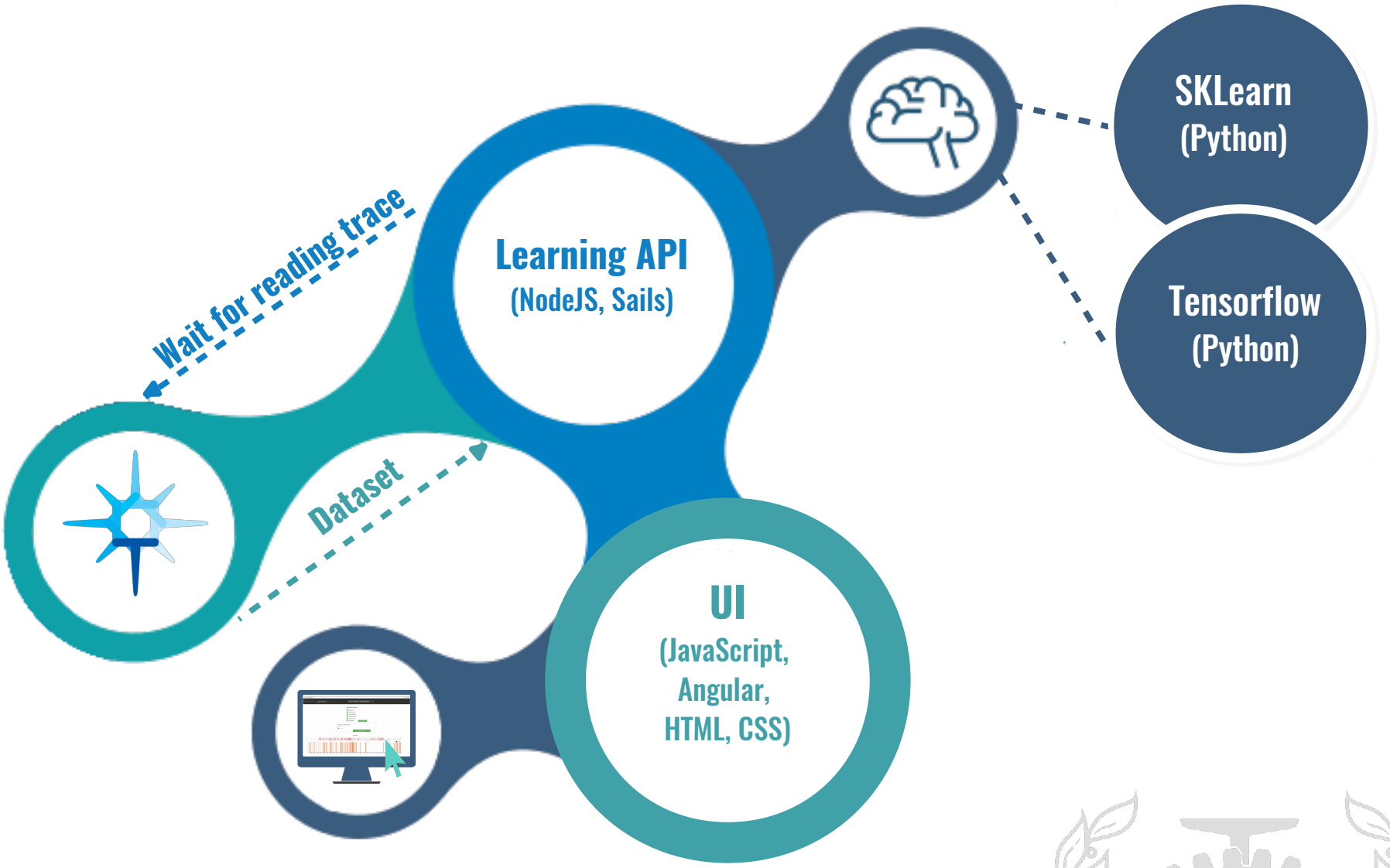



localhost:1338
Clustering System Requests
Anomaly Detection
Anomaly Detction
status of system:
Read trace
Create vectors
create Data set
Read training set
Ready to learn
Train SVM
Show SVM results
Select Learning method
LSTM
SVM
Start Learning
SVM results:
Show all

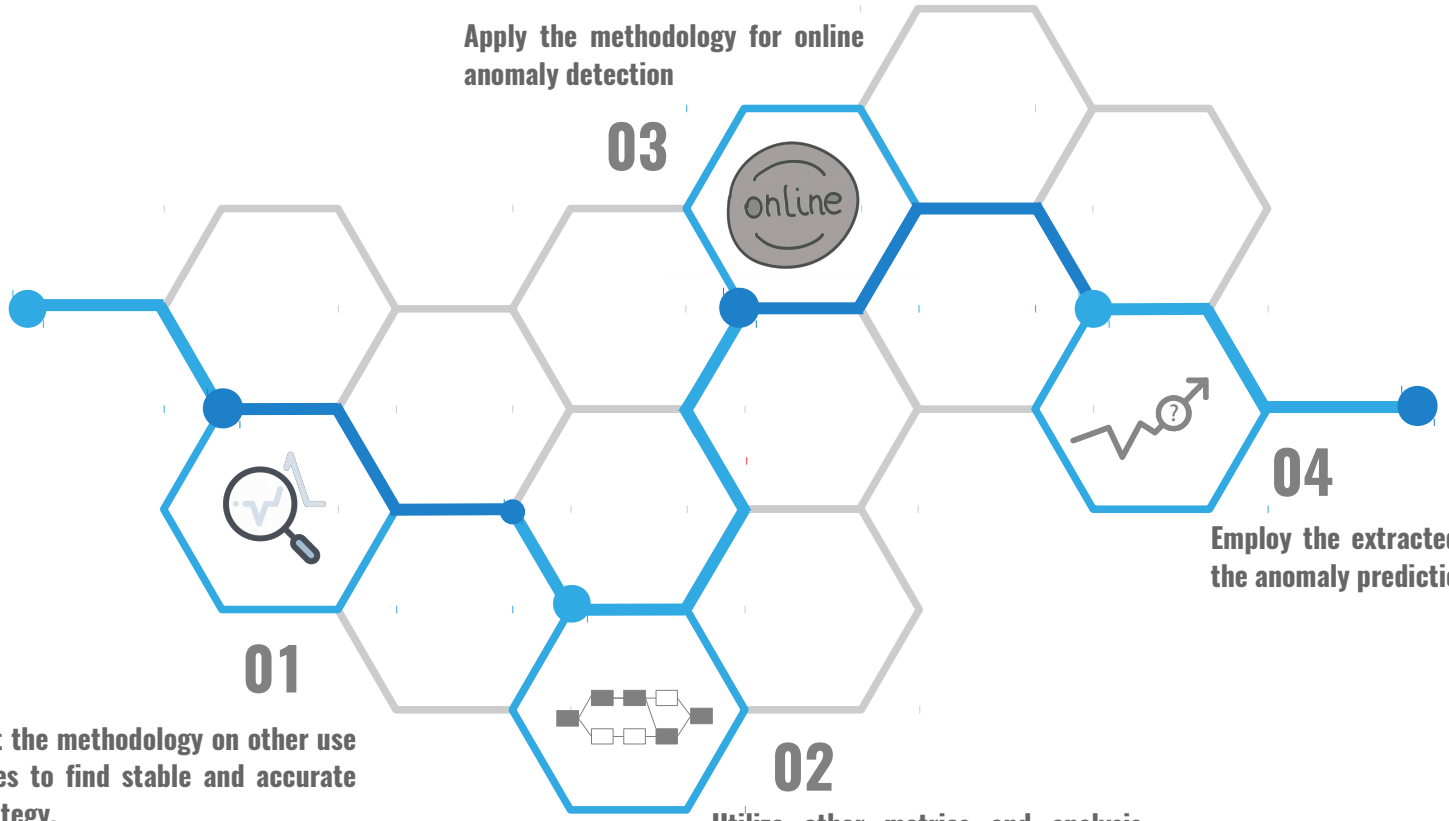
Technologies used in this framework



Framework Architecture



Future Directions



03

Apply the methodology for online anomaly detection

04

Employ the extracted features in developing the anomaly prediction framework.

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.

Thank you for your attention!

Questions?

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

References

1. Gebai M. et al., A thorough analysis of kernel and userspace tracers on Linux : design , implementation and overhead, Journal of CSUR, 2018
2. Kolosnjaji, Bojan, et al., "Deep Learning for Classification of Malware System Call Sequences", Springer International Publishing, 2016
3. R. Canzanese, S. Mancoridis and M. Kam, "System Call-Based Detection of Malicious Processes," 2015 IEEE International Conference on Software Quality, Reliability and Security, Vancouver, BC, 2015, pp. 119-124
4. Amiri M. et al., Survey on prediction models of applications for resources provisioning in cloud, Journal of Network and Computer Applications, 2017