Tracing IoT devices for anomaly detection purposes

Robin Gassais

Mai 10, 2018

École Polytechnique de Montréal

DORSAL lab

Agenda

Remind

Machine learning?

- Why using Machine Learning?
- Dataset creation and preprocessing

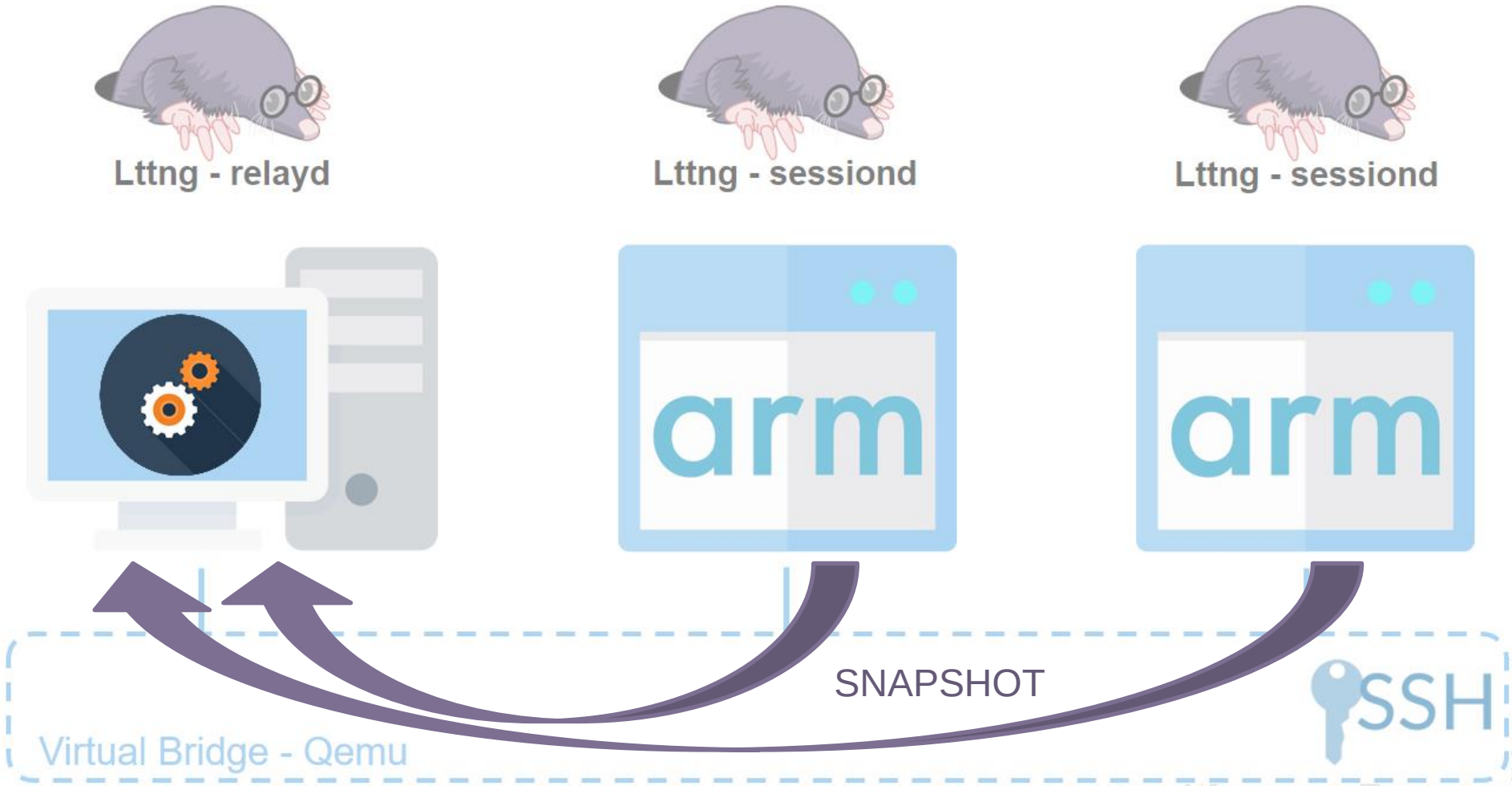
Anomaly detection

- Misuse detection
- Outlier detection
- Sequence matching

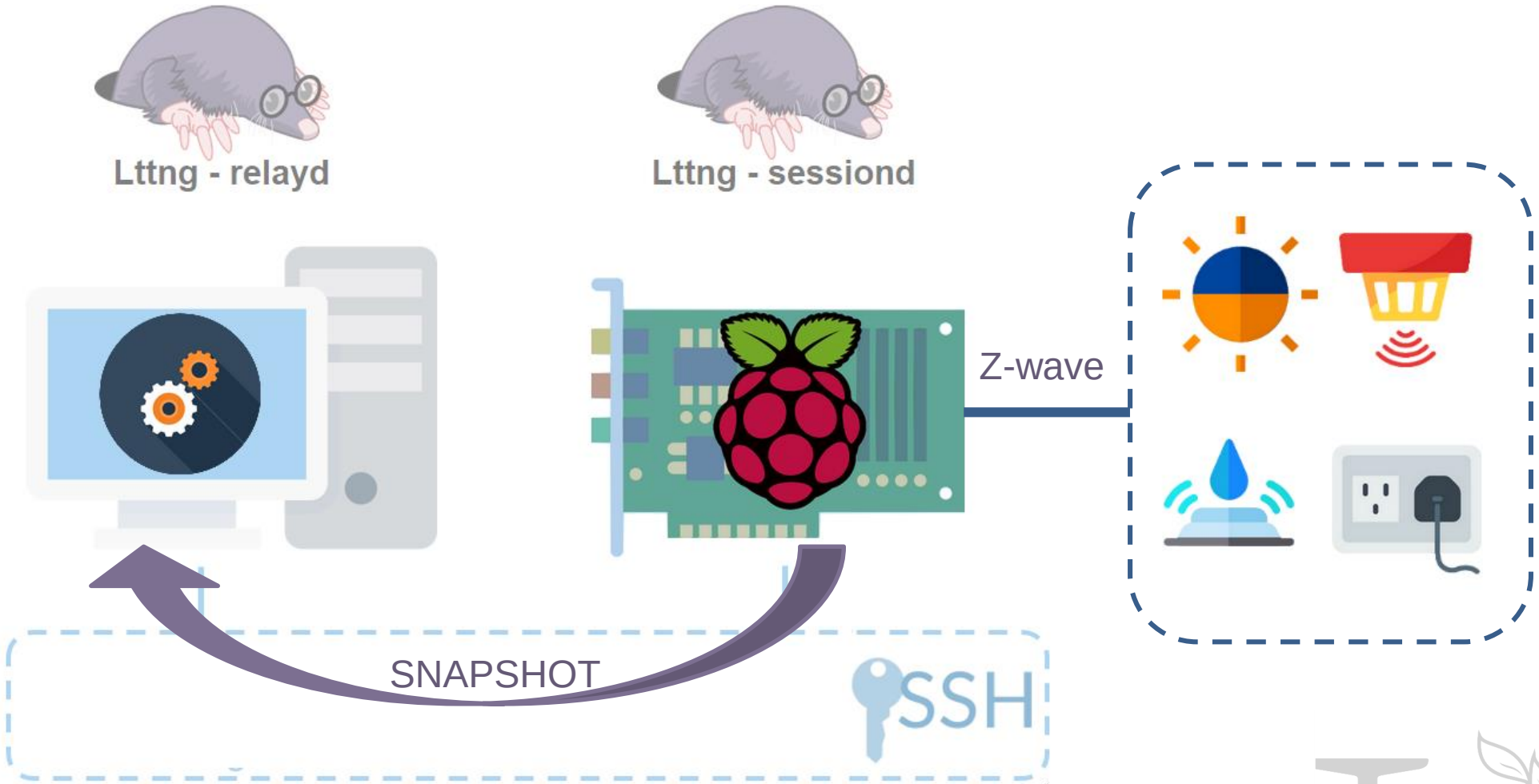
Use-case – Intrusion Detection

Future work

Remind


Lttng - relayd
Lttng - sessiond
Lttng - sessiond
arm
arm
SNAPSHOT
SSH
Virtual Bridge - Qemu

Remind



Machine Learning

Machine learning

Why using machine learning?

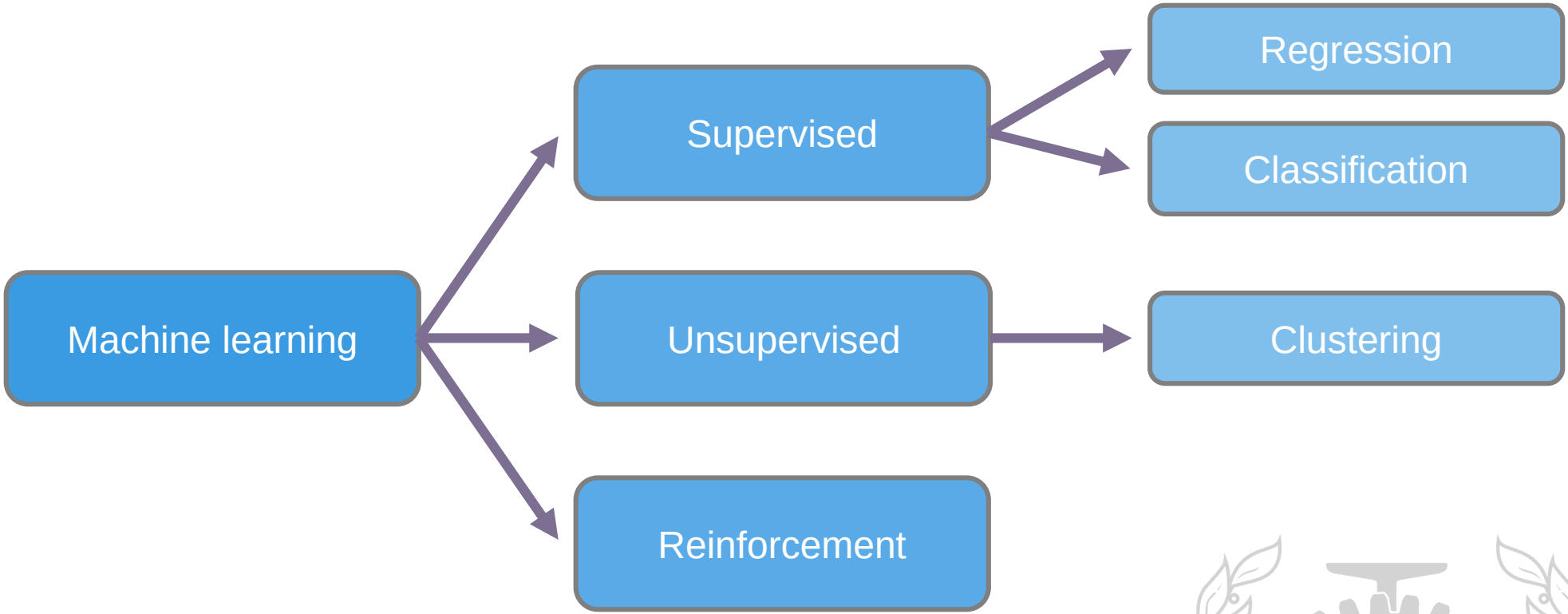
Machine learning is a set of techniques and algorithm that "give computer systems the ability to "learn" [...] **with data**, without being explicitly programmed"

Source : Paraphrased from Samuel Arthur - "Some Studies in Machine Learning Using the Game of Checkers"

Machine learning

Why using machine learning?

Machine learning is a set of techniques and algorithm that "give computer systems the ability to "learn" [...] **with data**, without being explicitly programmed"



Source : Paraphrased from Samuel Arthur - "Some Studies in Machine Learning Using the Game of Checkers"

Machine learning

Why using machine learning?

Machine learning is a set of techniques and algorithm that "give computer systems the ability to "learn" [...] **with data**, without being explicitly programmed"

- Help to create precise rules to detect anomalies
- Can detect new behavior we didn't think about
- Several python libraries that simplify usage

Source : Paraphrased from Samuel Arthur - "Some Studies in Machine Learning Using the Game of Checkers"

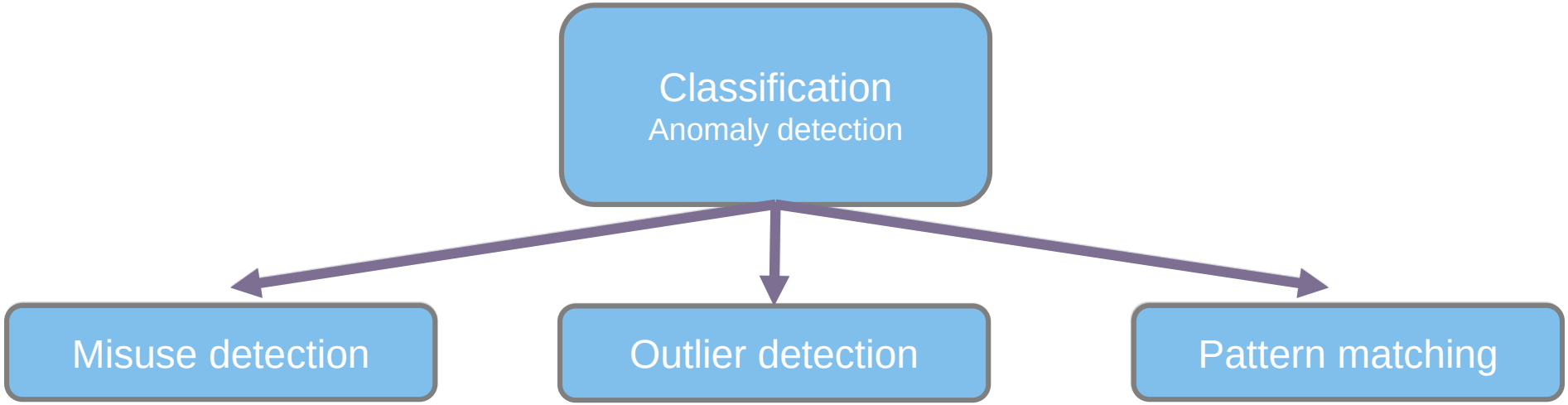
Machine learning

Dataset Creation

- Preprocess the traces and extract features
- Generate precise labels
- Problem : find a “balance” between normal and abnormal behavior

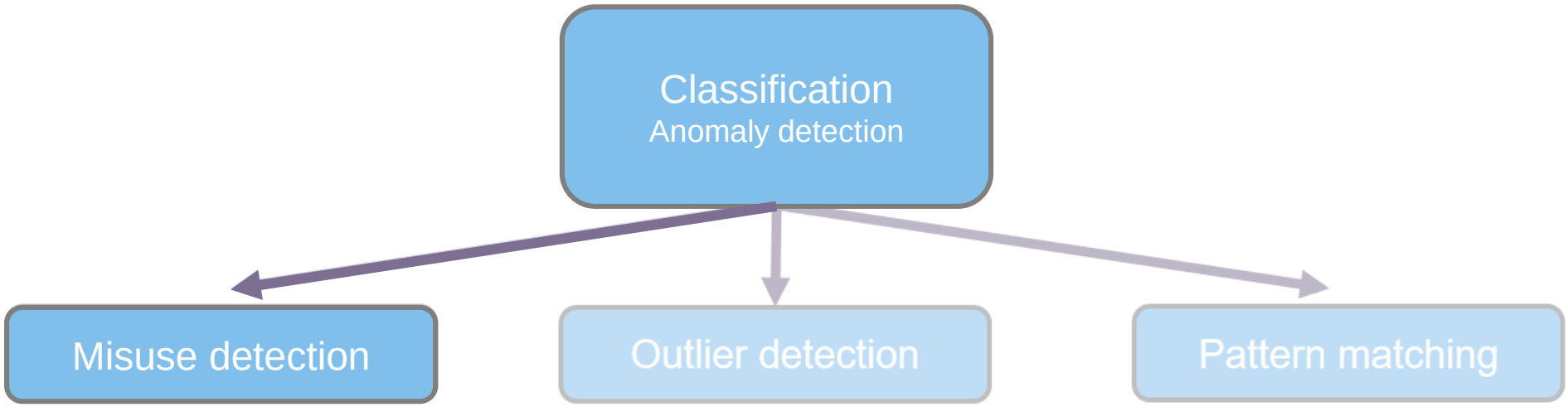
Anomaly detection

Anomaly detection



Anomaly detection

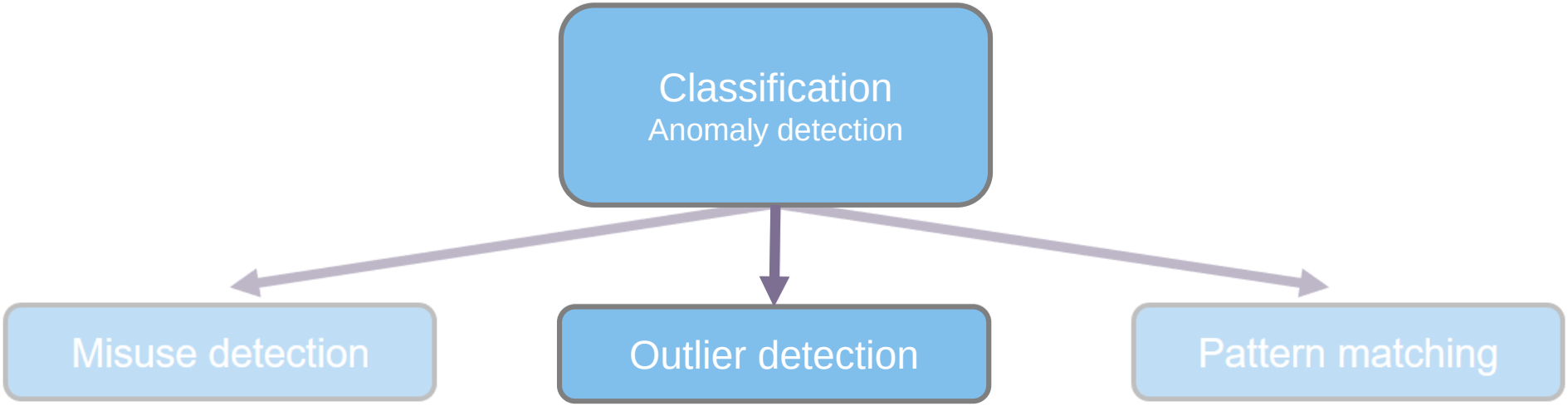
Misuse detection



- Decision Tree
- Random Forest
- Gradient Boosted Tree
- Support Vector Machines
- Multilayer Perceptron

Anomaly detection

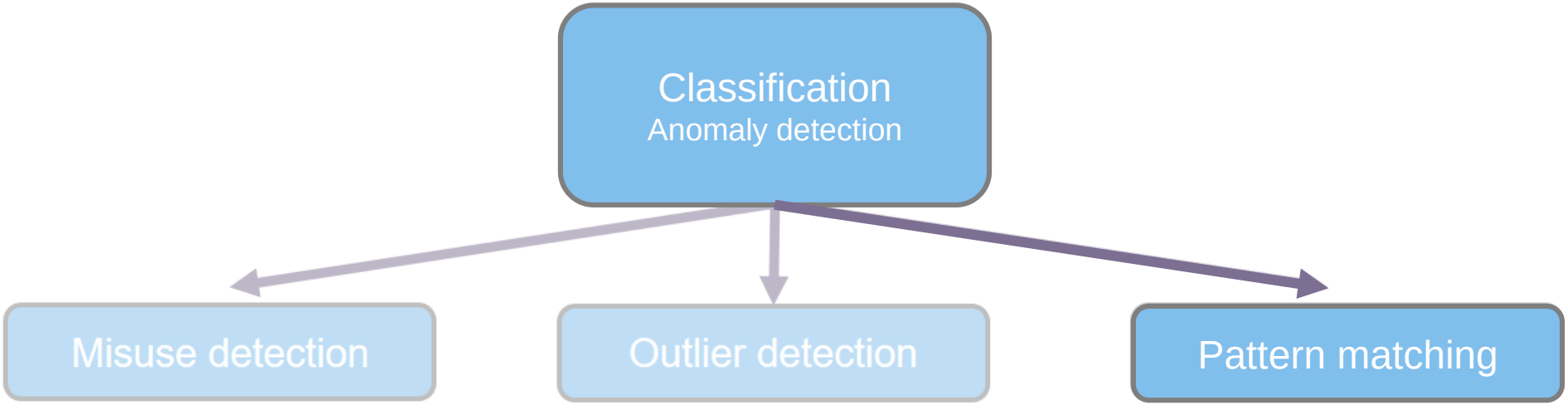
Outlier detection



- OneClass SVM
- Isolation Forest

Anomaly detection

Pattern matching



- LSTM

Use-case – Intrusion Detection

Use-Case – Intrusion Detection

Dataset Generation

Two steps to get the traces :

- Trace the kernel with a normal behavior – creation of databases
- Trace the kernel while being attacked

Getting the right data :

- Dataset cleanup
- Labeling with databases
- Processing data

Dataset :

- 177 895 events
- 60% label 0 | 40% label 1
- Syscall, sched, net, …
- Several custom feature

Use-Case – Intrusion Detection

Preprocessing

- Extract the features
- Create synthetics events
- Use other events to create new features

Syscall_open : {filename : "sudoTracing.sh", d_timestamp :1780004334 , pathname : "/etc/login.defs", p_name : "sudoTracing.sh"}

Syscall_read : {filename : "sudoTracing.sh", d_timestamp :1780004334, p_name : "sudoTracing.sh"}

...

Syscall_readlinkat : {pathname : "/proc/sudoTracing.sh/exe", d_timestamp : 1780004334, p_name : "system-journal"}

...

Net_dev_queue : {p_name : "sshd", d_timestamp : 890024167, source_port : 22, dest_port : 41640, saddr : 1322077235, daddr : 1322077219 }

Use-Case – Intrusion Detection

Tools

Tracing:

- Configuration file to enable the events
- Automatically receive snapshots and analyze them

Dataset creation:

- Creation and clean up dataset, handle multiple sources
- Easy labeling with custom rules

Preprocessing:

- Babeltrace Python API + custom scripts : CTF traces -> Array of number, custom features
- Pluggable with Scikit-learn and Keras

Use-Case – Intrusion Detection

Results

	Decision Tree	GBT	MLP	SVM	LSTM
Accuracy	98,48%	97,88%	55,41%	52,60%	89,40%
Precision	97,45%	96,22%	52,89%	55,33%	
Recall	98,86%	99,27%	56,54%	80,74%	
F1	98,11%	99,90%	40,93%	50,45%	

Obtained with cross validation – 5 folds – test dataset

Use-Case – Intrusion Detection

Results

- Attack 1 : mkdir infected -> touch infected.sh -> chmod u+x ./infected/infected.sh
- Attack 2 : ./infected2.sh

	Decision Tree	GBT	MLP	OneClass SVM
Detection latency attack 1	5,54 s	5,59s	X	32,93s (But lot's of FN)
Detection latency attack 2	X	X	4,74s	28,31s (But lot's of FN)

(Still under development)

Future Work

- Optimization of the algorithm
- More evolved attacks
- Combination of algorithm to improve the detection performance?
- Tradeoff between snapshot frequency, number of tracepoints and performance of the device

Future Work

Thank you!

Questions? Suggestions? Solutions?

robin.gassais@polymtl.ca

github.com/Blouglou2/TracingIDS