Dependability Challenges in Safety-Critical Systems: the adoption of Machine Learning

Andrea Bondavalli,

RCL Group - University of Florence – Italy

e-mail: andrea.bondavalli@unifi.it

RCL

RESILIENT COMPUTING LAB

UNIVERSITÀ DEGLI STUDI FIRENZE

DIMAI

DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Outline

- Introduction and motivation
- Safety and AI (and Machine Learning) artifacts
- Understanding Safety Systems solutions based on ML with associated challenges.
- Move along two cases:
  - ML as Controller coupled with a safety monitor
  - ML as a safety monitor

- Discussion and Conclusion

The advent of AI and ML

- It is a fact that AI solutions and Machine Learning algorithms are pervading all the areas and sectors of our automated life.
- They show superior performance as they learn from data and do not require the designer to master at start the complexity of any problem.
- They are (a bit more slowly) pervading also safety critical applications and systems……

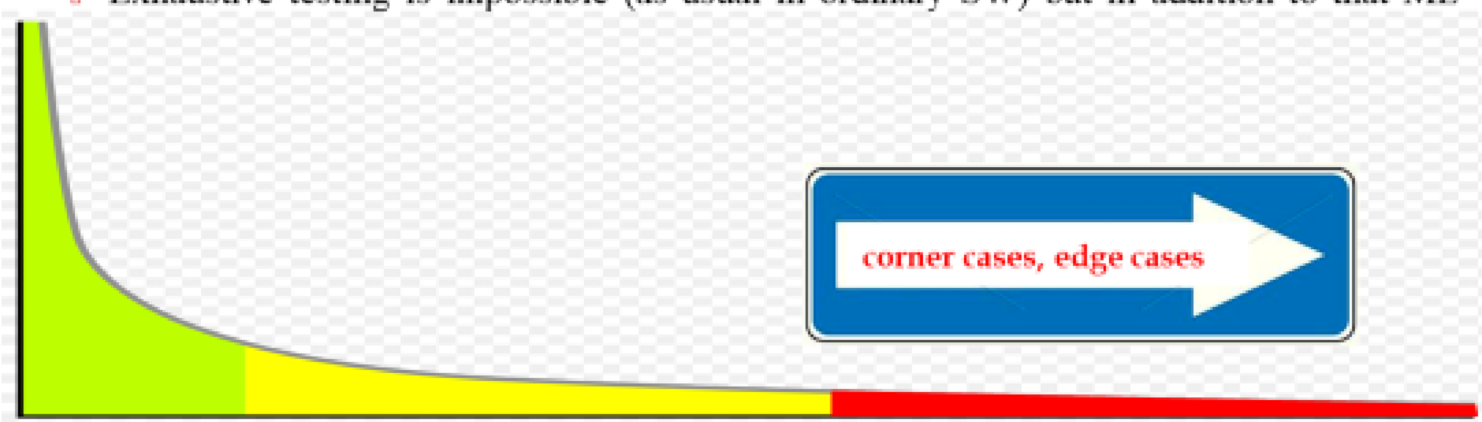
Safety critical systems and AI

Safety Critical System: any system whose failure might have severely unacceptable consequences regarding human lives, environment or society

Difficulties in AI enabled critical applications

AI/ML used in safety-critical functions:

- Lack of clear functional specifications
- Non-deterministic and probabilistic outputs
- Limitations of the training data
- Non-explainable ML (i.e., black box)
- Exhaustive testing is impossible (as usual in ordinary SW) but in addition to that ML



Henrique Madeira, 80th Meeting of the IFIP 10.4 Working Group on Dependable Computing and Fault Tolerance, Virtual - 25 June 2021 — 27 June 2021 6

Henrique Madeira, DEI-FCTUC, 2021

ML and safety assurance

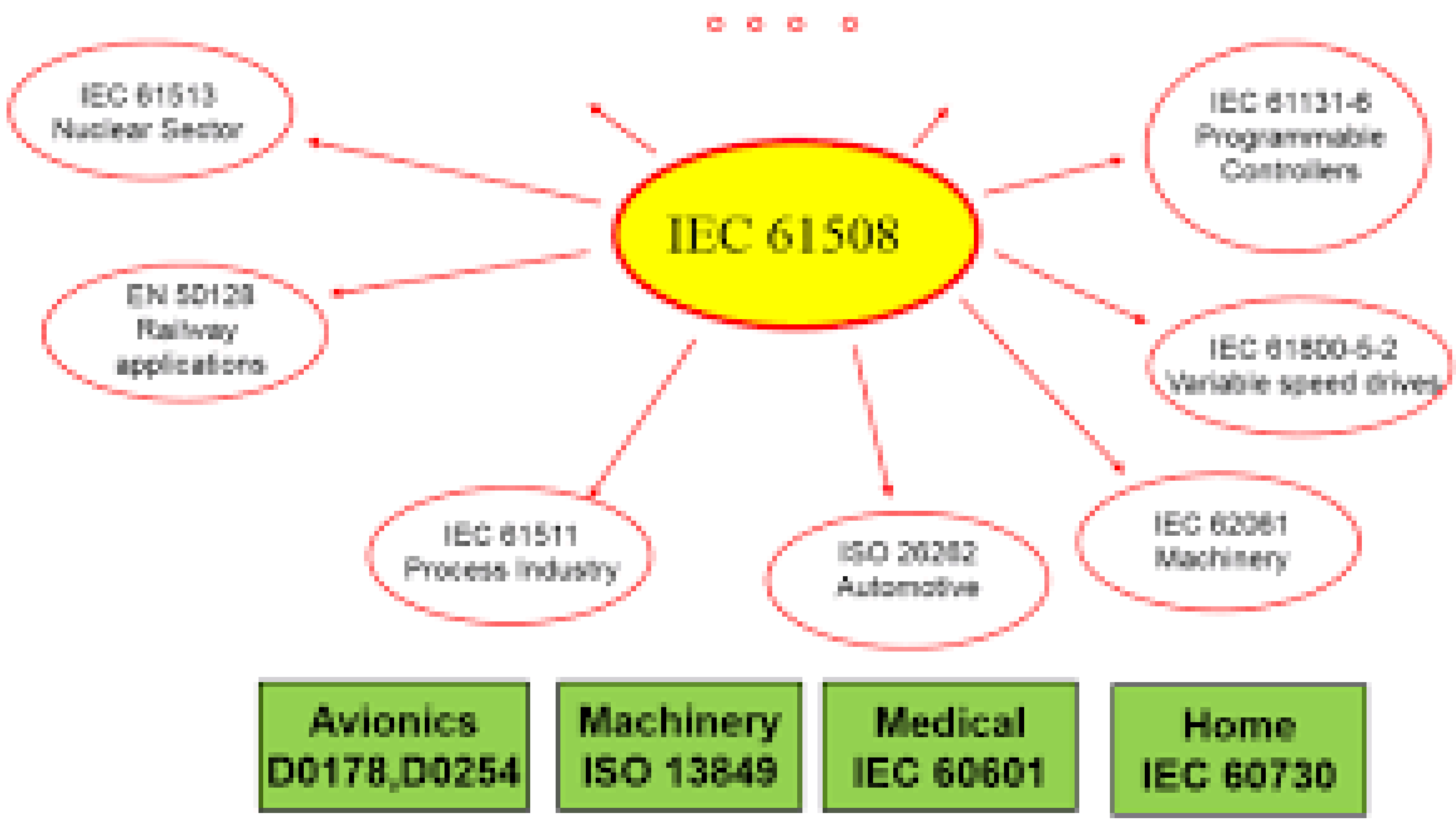
- **How to assure safety and security** in safety-critical applications making use of ML?
- **How to demonstrate** that one can trust on safety-critical applications incorporating Machine Learning?

Can we do that for e.g., self-driving cars?

- Millions of vehicles
- Billions of driving hours
- Huge pressure to cut cost
- Very high criticality

ML and safety standards

Existing safety standards did not evolve to cover ML technologies.



ML and safety standards

Most standards define the SIL for a function not for a component:

> specifying the SIL or ALR for an ML algorithm is not easy. It relies on the architecture of the incorporating system

Safety cases needed to derive the safety requirements at the ML algorithms level and to support evidence of their proper behavior.

Who is going to solve the problems?

Artificial intelligence

The solution is more and better AI

Problems:

- **How to assure safety and security** in AI enabled safety-critical applications?
- **How to demonstrate** that one can trust on AI enabled safety-critical applications?

More

- Robust AI models
- Non-symbolic AI
  - Larger training data sets and bigger and more complex neural networks
  - Interpolation vs extrapolation
- Explainable AI
- Ensembles
- ...

Today's talk target

While waiting for the problem to be **SOLVED** we have been focusing on understanding:

The properness and the risks of Incorporating Machine Learning Algorithms into Safety-Critical Systems

Safety Management and ML Algorithms

Safety management: practices to achieve or maintain safety through fail-aware, failsafe, or fail-operational mechanisms.

To be used also in case of ML-based component incorporated into SCS

To deal with faults resulting from ML-based components:

- Safety envelope
- Runtime verification and fail-safe

RCL
RESILIENT COMPUTING LAB

2 Different Scenarios

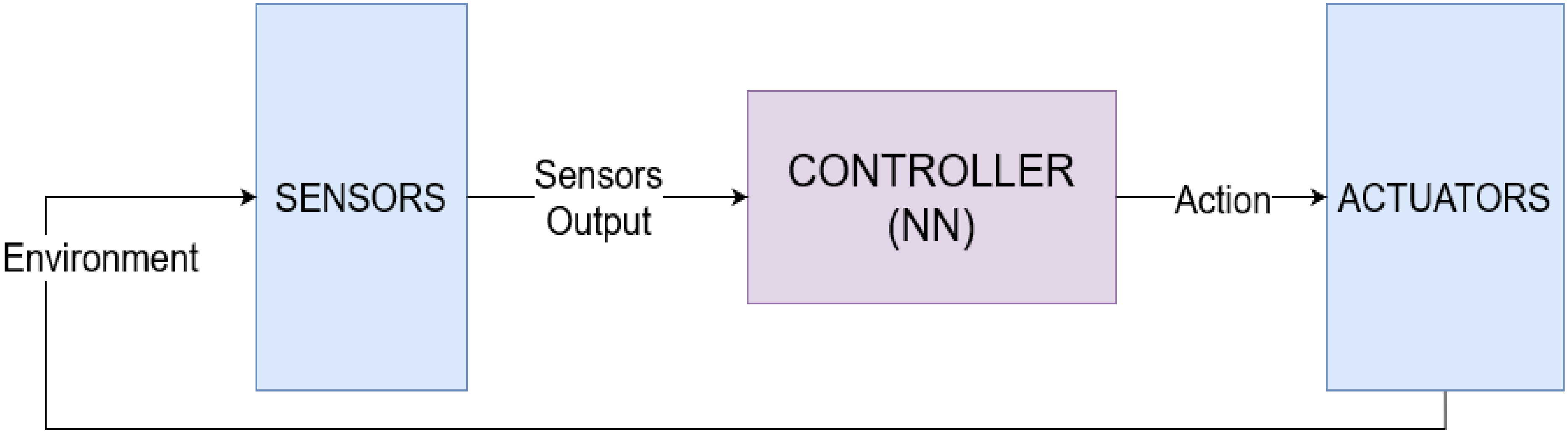
- The remainder of this talk will develop along two different scenarios:
  - ML **performing the system function** (taking decisions) and a simple safety monitor to check and take safety measures (e.g. stop the system)
  - ML as a **binary classifier performing the safety monitor** role (error/anomaly/intrusion detection) and triggering safety measures

ML Performing the system function

Autonomous Vehicles

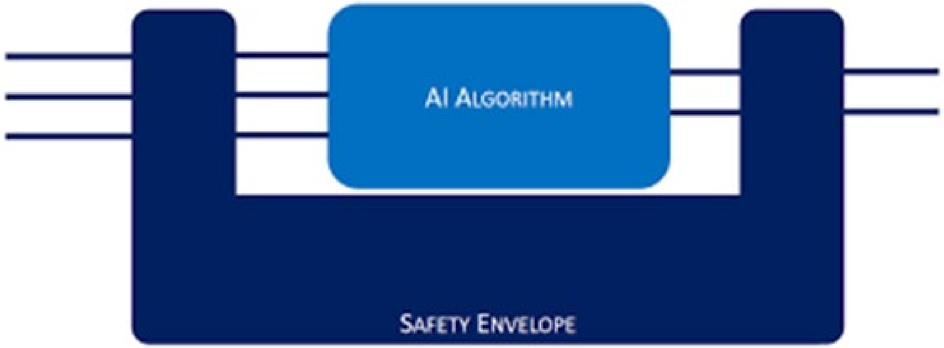
- Autonomous vehicles are one of the greatest examples of the power of machine learning

- These systems are controlled by a neural network (or an ensemble of neural networks), which we call "Controller", trained with huge amount of data to perform the driving task



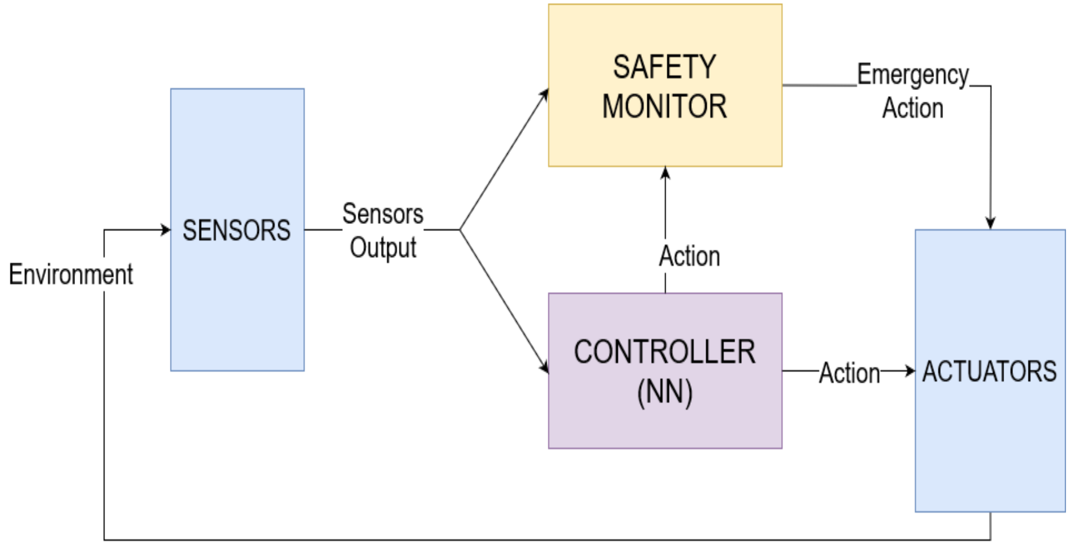
Safety Envelope

- Given that we cannot trust these control systems ("Controllers", for brevity) to be safe enough, it is natural to apply independent safety subsystems ("Safety Monitors" or SM)



- Ideally, a safety monitor is much simpler than a Controller, so that, once verified, it gives strong confidence that it will perform to the level of reliability (and hence of vehicle safety) assessed

A Vehicle – System view



Controller: an end-to-end deep neural network,

Safety Monitor: performs safety checks based on the sensors output *and* the action chosen by the Controller. It will perform an emergency brake If the Controller breaks the "safe braking distance" rule

An Experiment

Controller: a neural network trained with the Deep Deterministic Policy Gradient (DDPG) algorithm.

- The implementation is provided by the "*Framework for Reinforcement Learning Coach*", an open-source project developed by Intel's AI Lab

The Safety Monitor is in charge of detecting hazardous events

- It receives LiDAR data and process them using «classic» algorithms such as ground segmentation and clustering.
  if the Controller breaks the "safe braking distance rule" the monitor will perform an emergency brake

Controller – Monitor interactions

- the *state space* of the System (Controller + Safety Monitor):

Safe States: all the states in which the Controller does not need the intervention of the Safety Monitor, and the Monitor does not intervene

Mitigation States: the Controller's behaviour would lead to a system failure (accident), but the Monitor correctly prevents the crash.

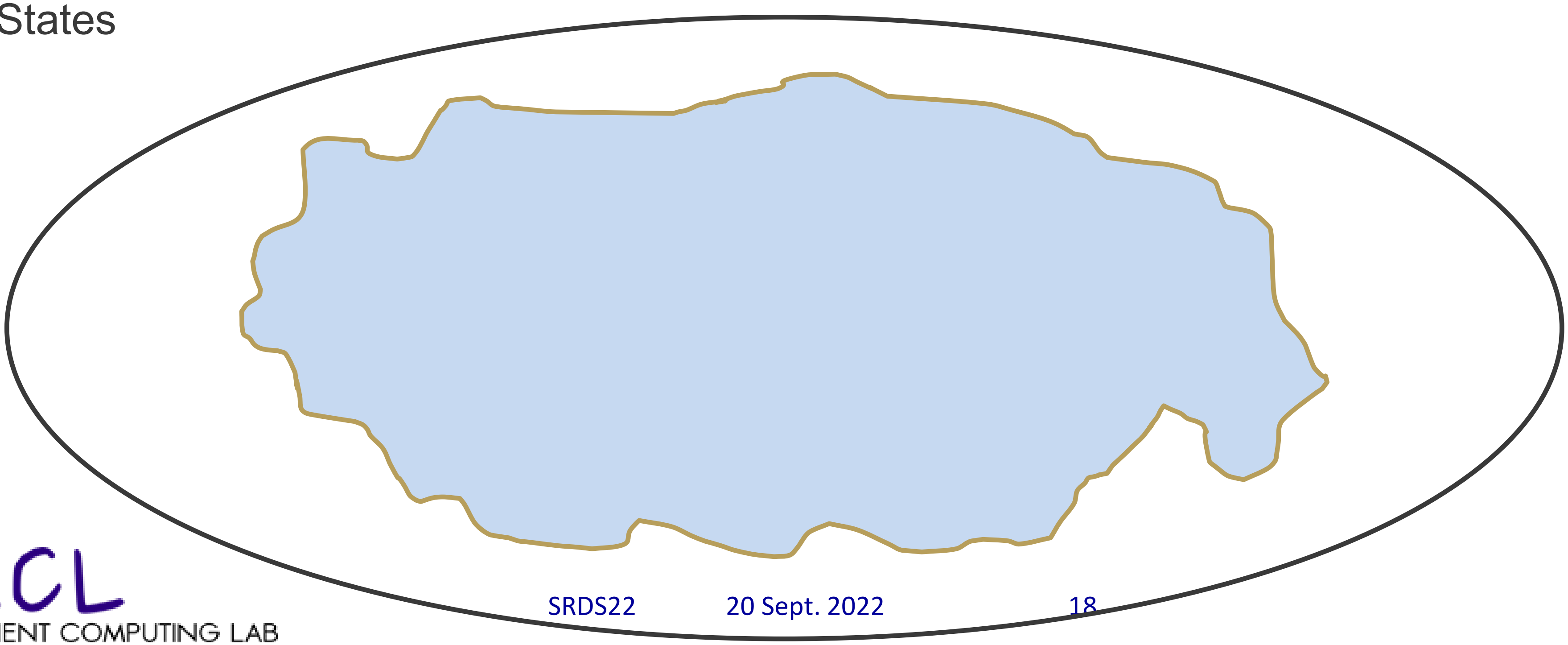
False Alert States: the states in which the Controller does not need the intervention of the Safety Monitor, but the Monitor wrongly intervenes.

Accident States: all the states in which the Controller's behavior leads to a crash which are not solved by the Monitor.

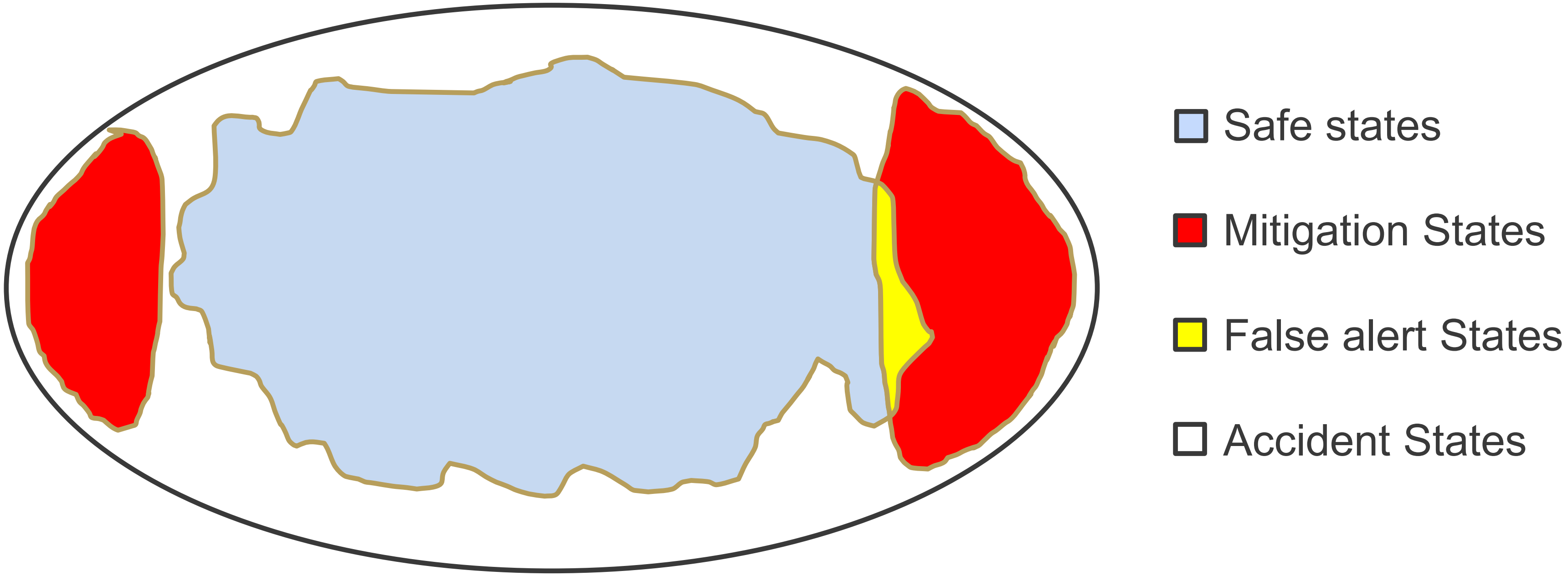
Controller training

- Assume the Controller was trained for some time, giving us this picture of the state space:

- Safe states
- Accident States



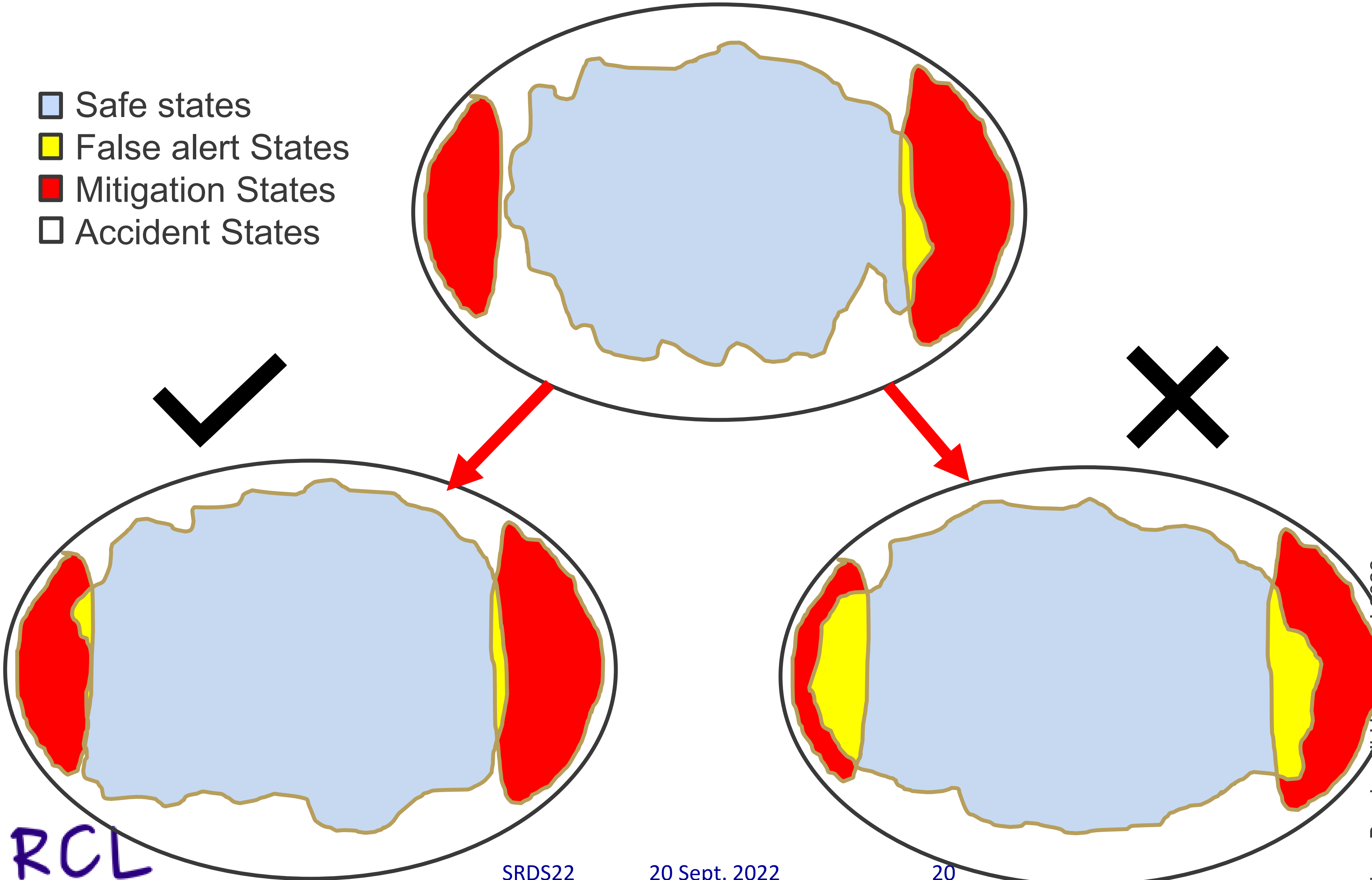
Adding a Safety Monitor



To improve the overall system's safety, it would be natural to improve the Controller by performing further training activity

Further Training: Possible evolutions

- Safe states
- False alert States
- Mitigation States
- Accident States

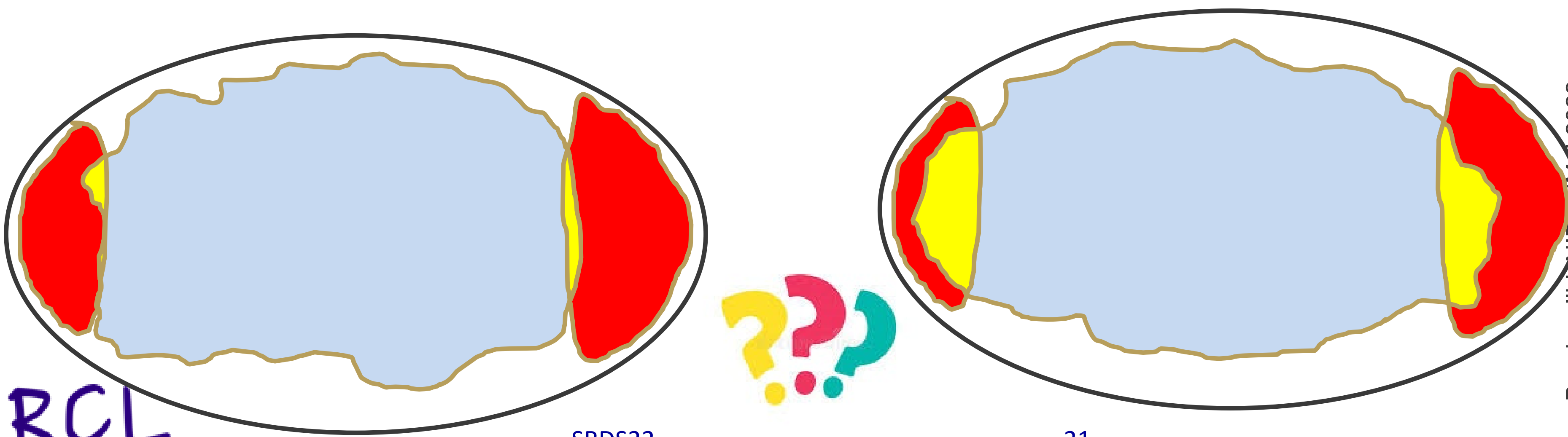


Wich will be the actual evolution?

We can't estimate in advance how the modifications (learning) will change the Controller's performance, but…

At the same time, the Safety Monitor is a "simpler" component, designed to react to specific hazardous events and, in general, not subject to changes

Assuming constant "coverage" in safety monitors while the primary system evolves is a potentially dangerous fallacy.



The Problem

The uncontrolled evolution of a machine-learning component raises questions from the point of view of safety assurance, especially when paired with other components such as the Safety Monitor

Our experiment

The Controller can be trained until it meets the desired performance.

The Safety Monitor is a "simple" submodule, (using classic techniques).
Its behaviour can change only if re-implemented.

The Controller is trained in *without* Safety Monitor. After the whole system (Controller + Safety Monitor) is tested.

We used CARLA, an open-source urban driving simulator

Controller Test Run

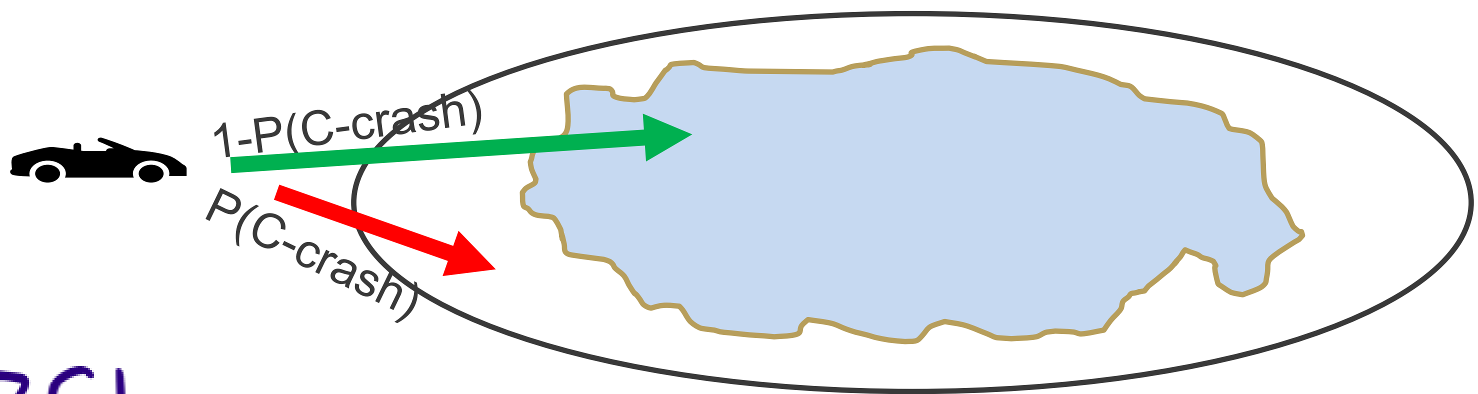
The Controller is first tested alone. A test run of the Controller ends when:

- *a collision happens* or
- *all the target destinations are reached*

We define the event C-crash (Controller crash) as

"a crash would occur without a SM"

From which we compute: – **P(C-crash)** = $\frac{number\ of\ C-crashes}{kilometers\ driven}$



Safe states
Accident
States

RCL
RESILIENT COMPUTING LAB

Testing the Safety Monitor

- The runs of the Controller are replayed, and the Safety Monitor introduced
- Simulations run at a fixed time-step: we can thus compute the *time* ***t*** necessary for the SM to prevent a collision, if it happened in that specific run

- All the alerts of the SM *before* time ***t*** do not trigger the emergency brake, but are recorded (as False Alarms)

- All the alerts of the SM *after* time ***t*** trigger the emergency braking

Measures

From the recorded counts of these basic events we derived the metrics of interest:

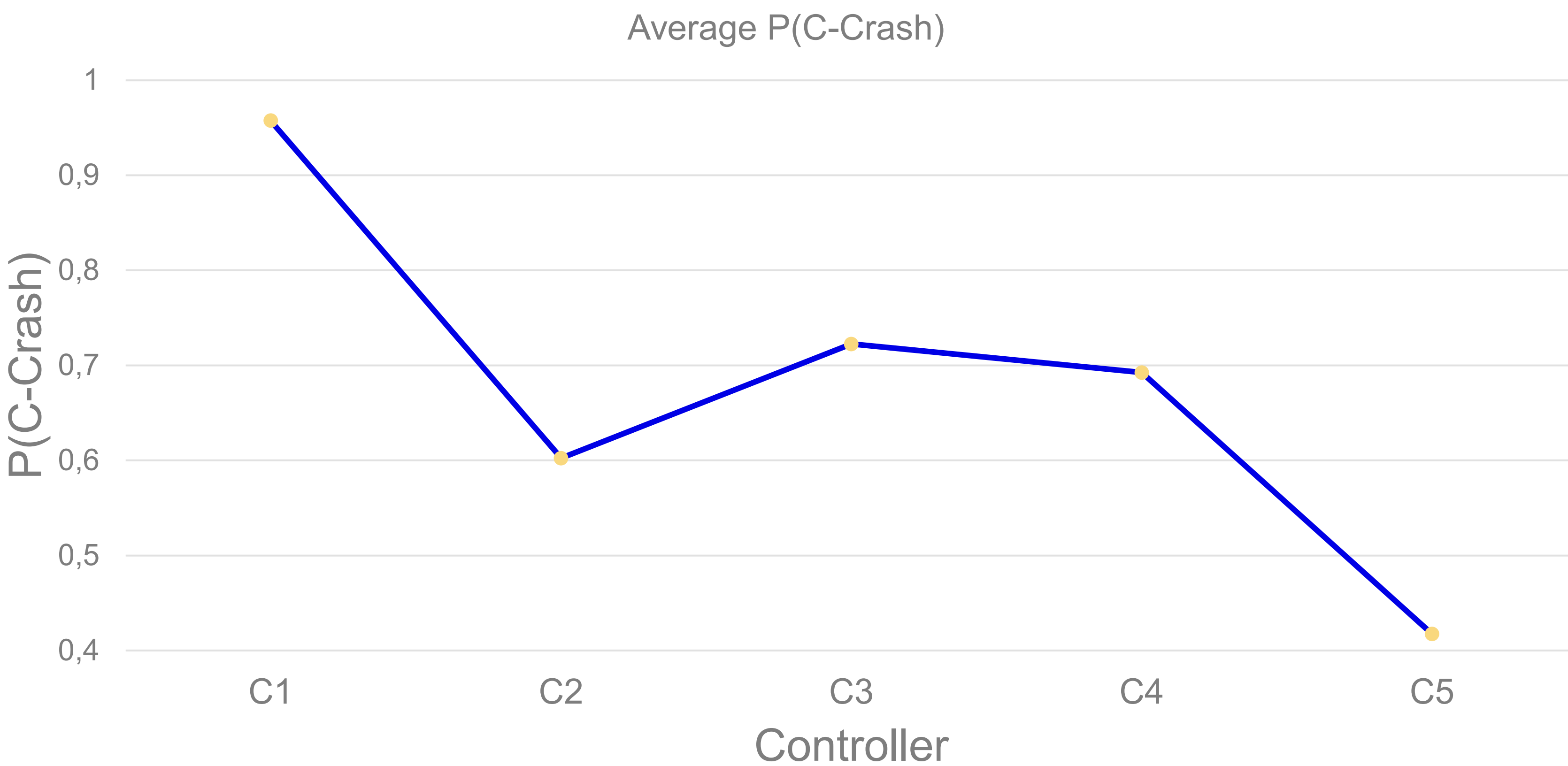
- **Coverage (COV)** = $\frac{number\ of\ SIs}{number\ of\ C\text{−}crashes}$

- **P(crash)** = $\frac{number\ of\ crashes}{kilometers\ driven}$

- **False Alarm Rate (FAR)** = $\frac{number\ of\ FAs}{number\ of\ FAs + number\ of\ TNs}$

Results – Controller P(C-crash)

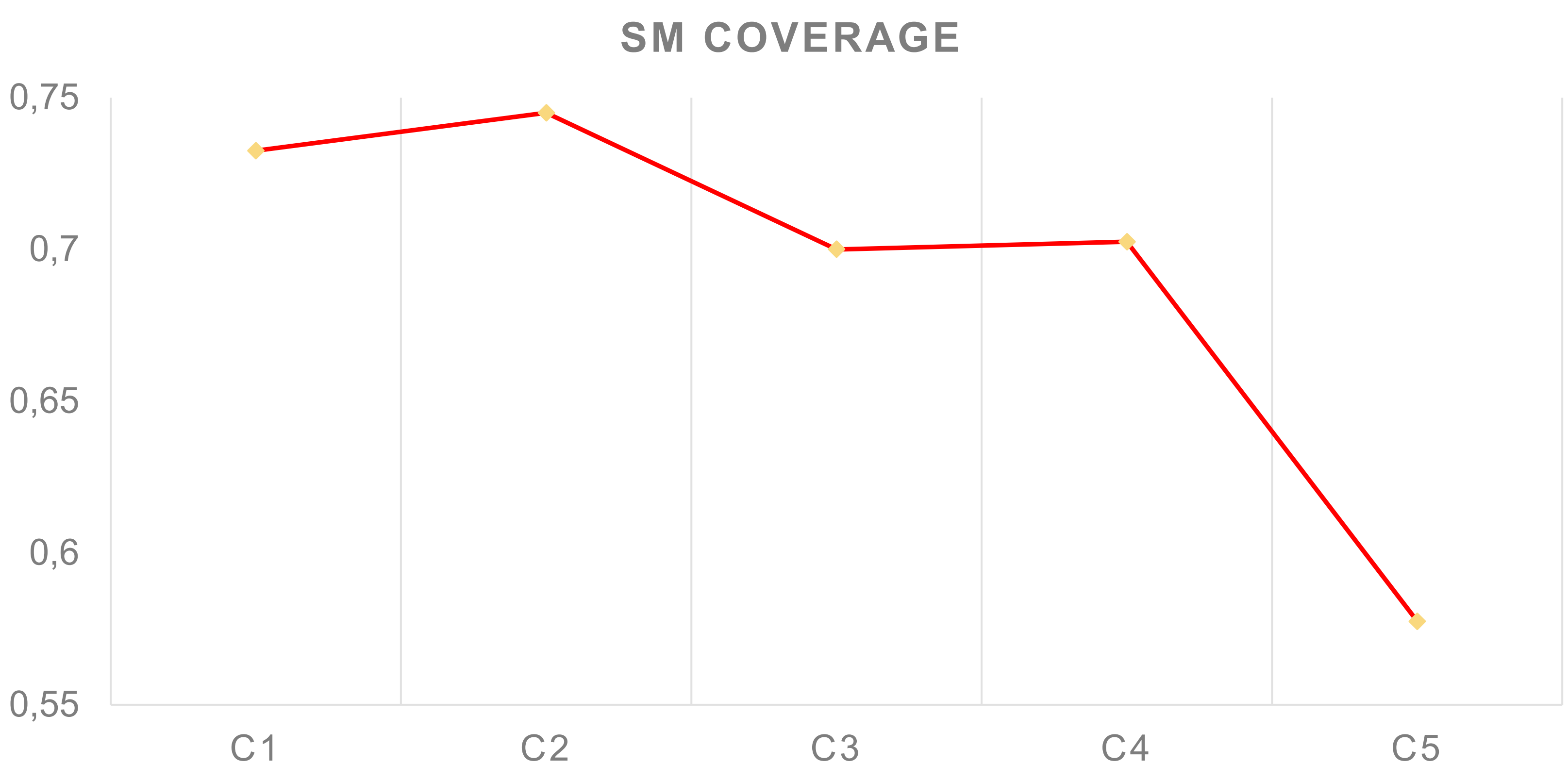
The Controller was tested at 5 different stages $C_1 \ldots C_5$



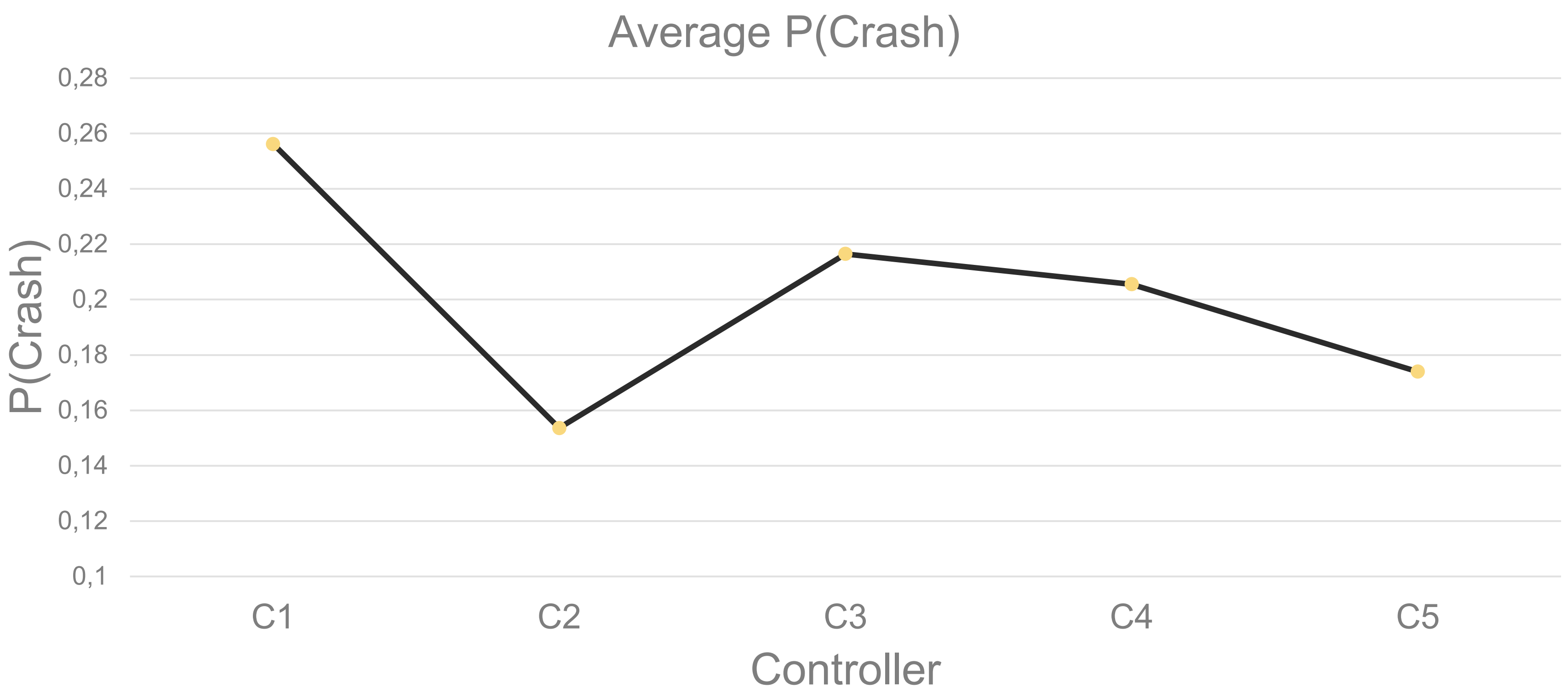
Probability of the Controller alone causing a crash, i.e., P(C-crash)

Safety Monitor Coverage

Coverage of the Safety Monitor when applied to the Controller at different learning stages. We can see that its efficacy is at its minimum when combined with the "best" Controller



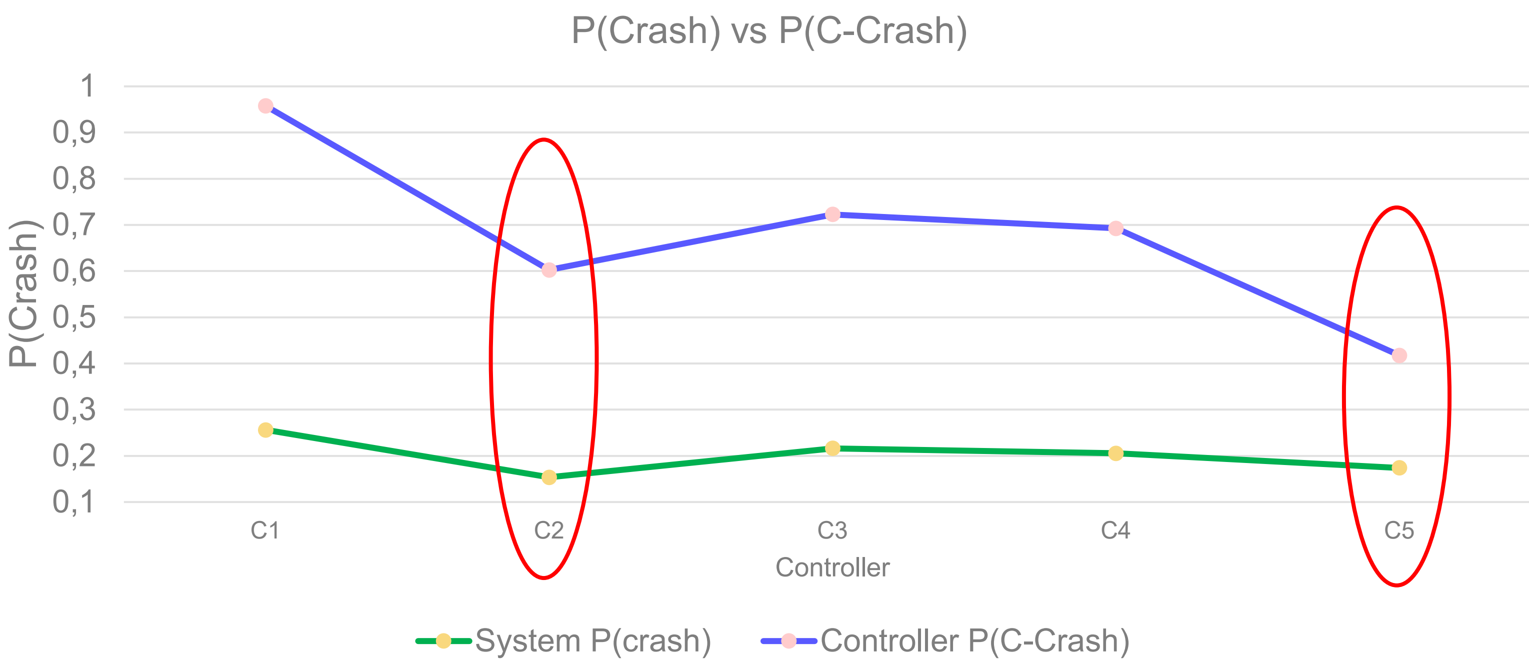
Results - System



Andrea Bondavalli UNIFI-DiMaI 2022

Best system combination

By comparing P(C-Crash) and P(Crash) we can see that adding the Safety Monitor *always* reduce the likelihood of a crash



Most IMPORTANT: We can observe that, although C5 is ***significantly*** better than the other Controllers, the System **performs better when using Controller C2**

Remarks

► It is common practice in systems design to build and analyse pieces in isolation and then enjoy some 'composability' when putting things together.

► If we used this approach as we *did not* change the Safety Monitor, we might have expected to observe a coverage between 70% and 75% of the Monitor

► As training provided good results: C5 P(C-Crash) is quite lower than previous controllers, one would expect the System to improve with the improvement of the Controller!

Lesson learnt

▶ **BUT….** not only the COV(erage) of the Safety Monitor drastically decreased when combined with Controller C5

▶ but even the **safety of the *whole system* got worse**!

▶ we observed one example of the **possibly dangerous *emergent phenomena*** that can raise by combining Machine Learning and "static" software

ML as Safety checker

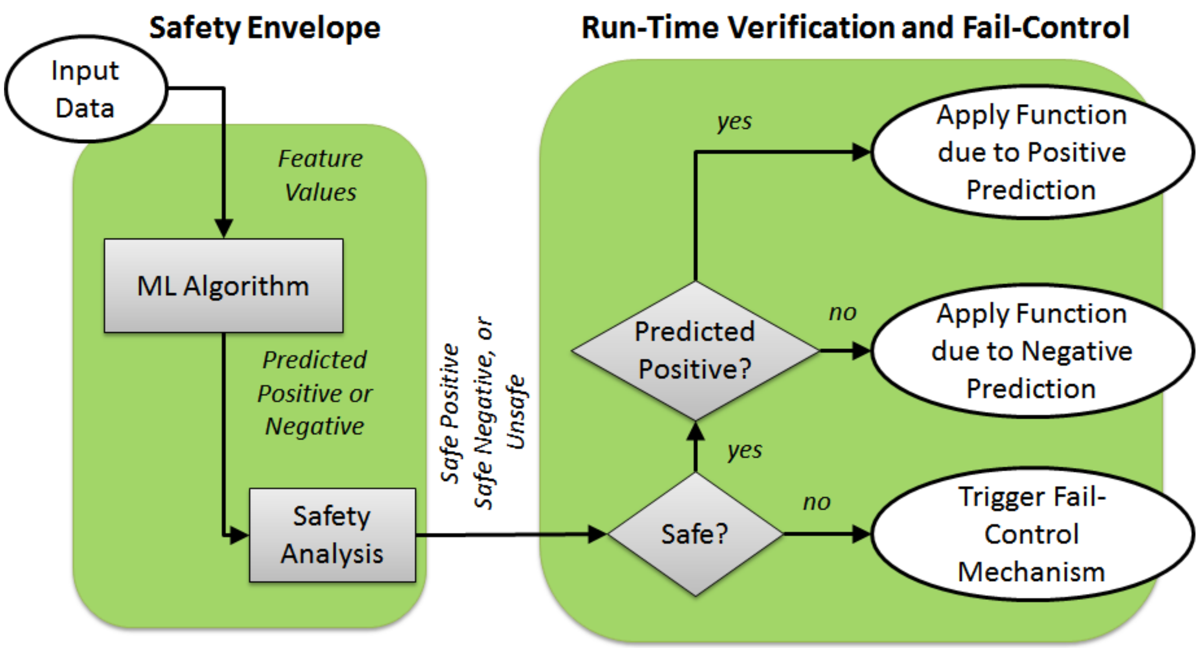
An architecture for binary predictions

A Binary classifier acting as the safety monitor → **need to have confidence in prediction to safely apply its decisions**

We want to use an ML algorithm that either

i) provides predictions that are sufficiently safe to be used, or

ii) triggers fail-control mechanisms.



Safety and misclassifications

Misclassifications may either be **FNs** (real problems predicted as normal) or **FPs** (normal situations predicted as problems).

- FNs are the primary and direct trigger to catastrophes
- FPs may indirectly lead to unsafe situations.

We assume here **that only FNs** are the cause of safety issues.

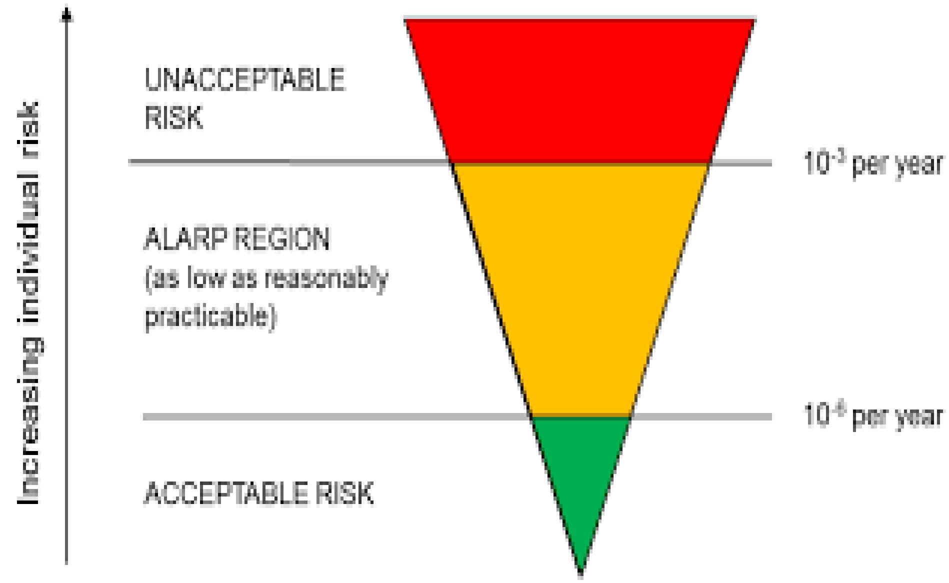
Safety does not mean that critical events will never occur in a system.

It guarantees that the risk (combination of likelihood and impact) of a threat is tolerable according to the requirements.

Acceptable Level of Risk (from standrds Standards)

a **SMALL** number of FNs, FN* (or residual FNs) may be admitted

Depending on the Acceptable Levels of Risk (ALR) derived from the safety requirements



- ALR is a commonly used concept in safety standards to specify the tolerable hazard.

normally defined as

- THR tolerable hazard rate or
- PFD probability of failure on demand

Properness of a ML algorithm

Our target becomes

To assess the properness a given ML algorithm to be used as safety monitor.

Answering the following question:

Can it be safely used or does its usage bring to an unacceptable risk?

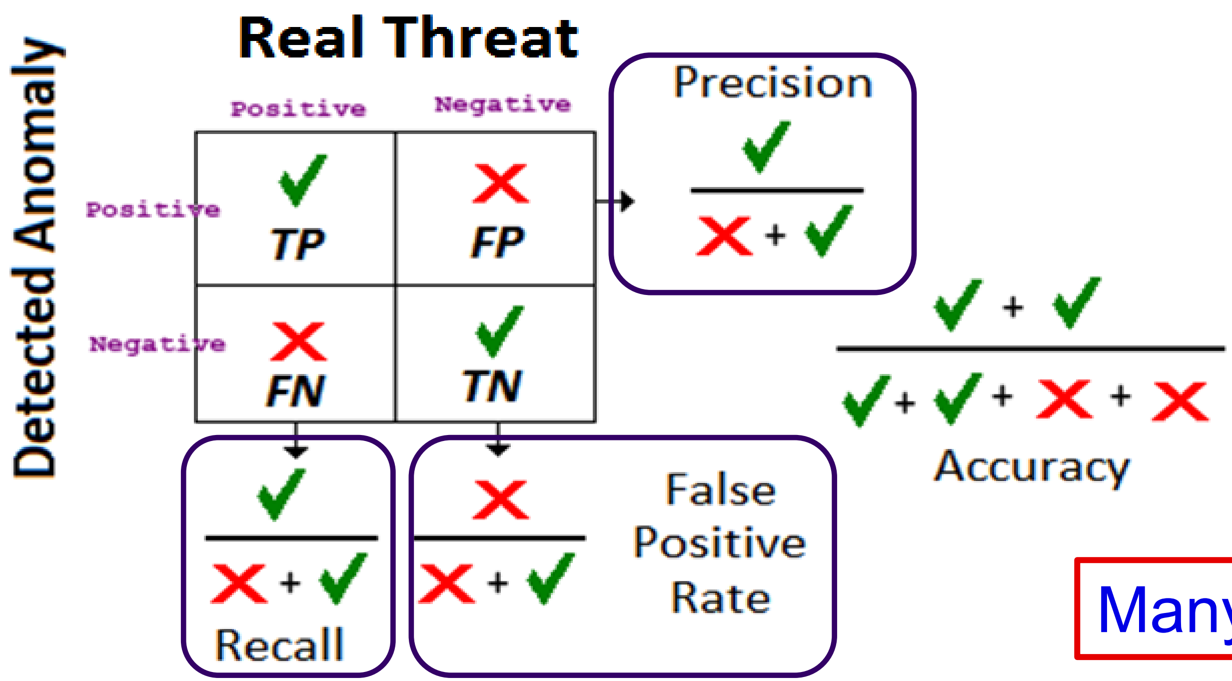
Measures for assessing ML prediction algorithms

- The effectiveness of predictions are assessed depending on specific indicators.

Given a data point and the judgement of an algorithm

→

4 outcomes which populate a confusion matrix used to derive metrics

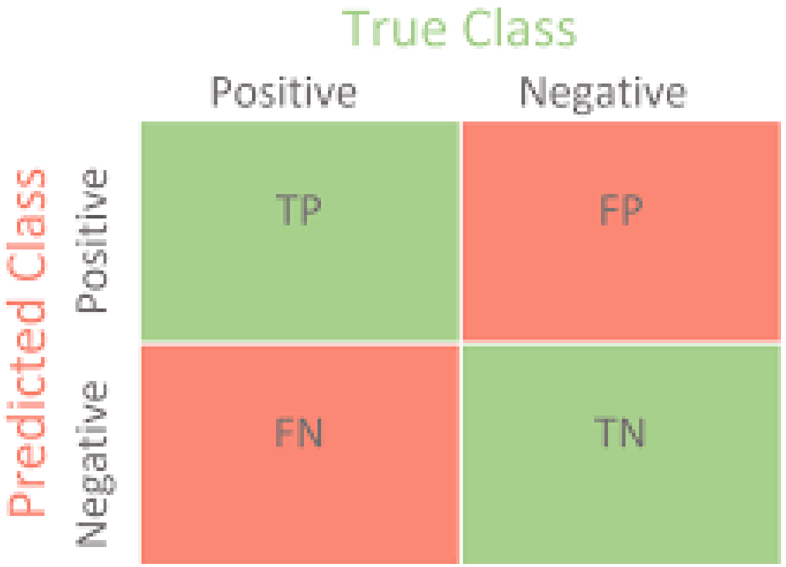


Many metrics exist

Do we have the proper metrics?

Normally
1) (True) NEGATIVES are much more than positives

2) Recall, Precision and their combination do not consider TN which is the most populated class.



Metrics based on the confusion matrix (i. e. based on the number of misclassifications) may not adequately describe all the aspects of the behavior of an ML algorithm.

They may not be able to help us in answering to our question.

Structure of ML algorithms

Any ML algorithm (used as a binary classifier) devises a mathematical model from a training data set. Once training is completed it makes predictions through a function:

dp_label = alg.predict(dp)

alg.predict(dp): alg.decisionfunction(alg.score(dp))

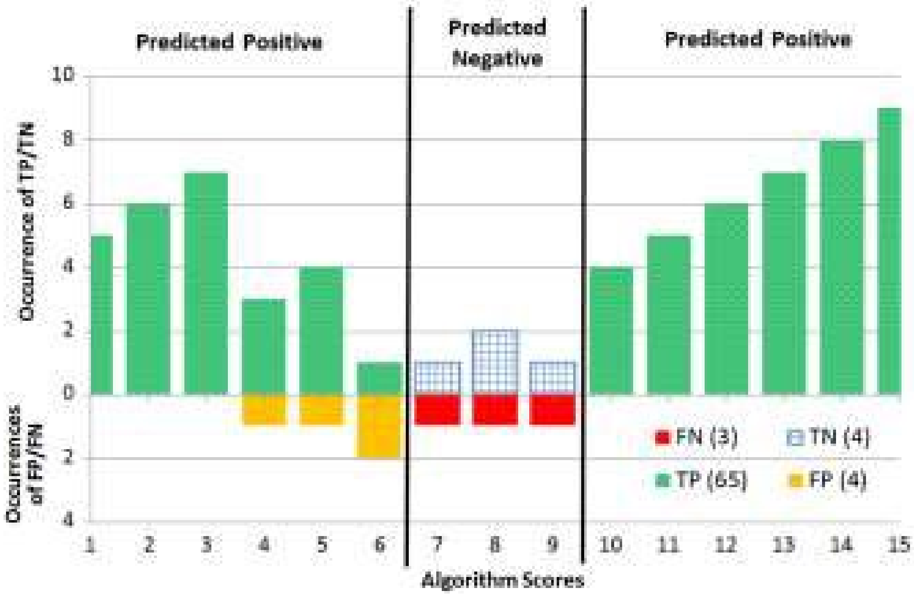
alg.score(dp)
is a numeric score (depending on the type of algorithm)

alg.decisionfunction(num_score)
converts a numeric score into a binary label

- dp — a single data point,
- dp_label — the (binary) prediction for a data point

Example. ML algorithm alg1

- numeric scores in the range of [1;15]
- Binary decision:
  - negative if 7 ≤ score ≤ 9, positive otherwise.
- test set of 76 data points
- Results: 65 TP, 4 TN, 4 FP and 3 FN,
- 90.7 Accuracy
- and 94.9 F1-Score.

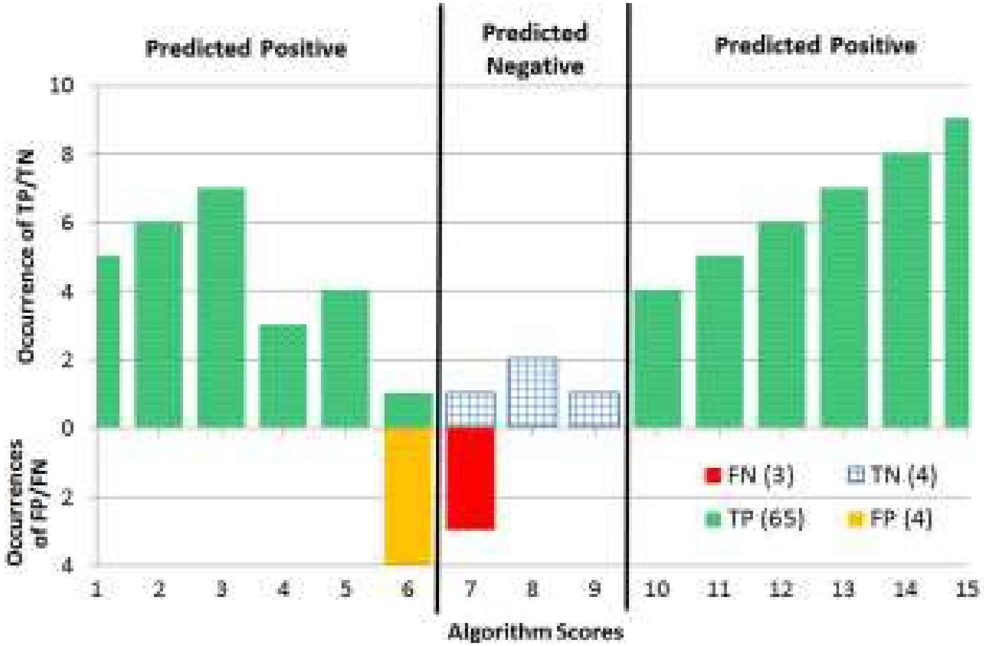


Andrea Bondavalli UNIFI-DiMaI 2022

RCL RESILIENT COMPUTING LAB

ML algorithm alg2

range [1;15]
Same binary decision
test set of 76 data points
Result: 65 TP, 4 TN, 4 FP and 3 FN,
90.7 Accuracy
and 94.9 F1-Score.



- ▶ SAME AS ALG 1
- ▶ BUT DIFFERENT DISTRIBUTION

Andrea Bondavalli UNIFI-DiMaI 2022

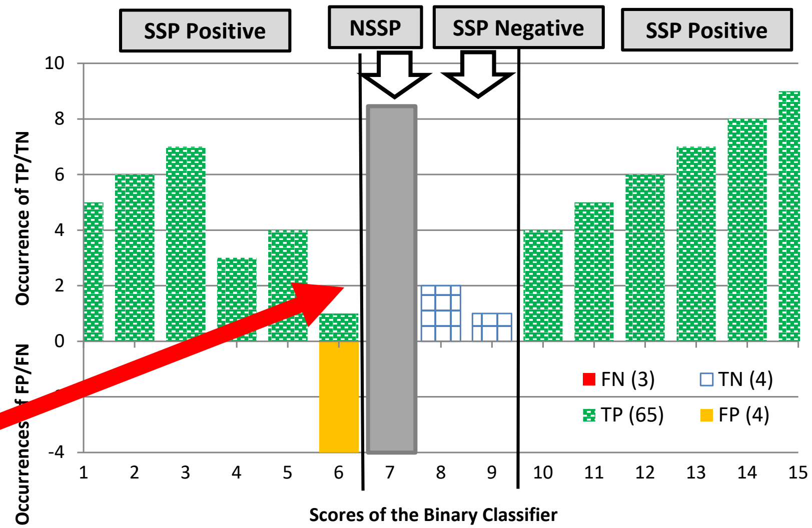
RESILIENT COMPUTING LAB

Difference between Alg1 and Alg 2

- same confusion matrix (the number of misclassifications is the same).
- Misclassifications by alg2 only on scores in the range [6; 7] while by ALG 1 in [4:8].
- **This difference is not captured by the confusion matrix and all the metrics based on the number of TP, TN, FP, and FN.**

From Binary to Semi-Ternary Classification

- analyze the distribution of misclassifications
- identify which numeric scores may generate misclassifications, (especially FNs)
- consider this subset of scores as not sufficiently safe
- Identify an area containing not sufficiently safe (NSSP) predictions, while the rest predictions are sufficiently safe (SSP)



- 76 data points: 69 SSP Positive, 3 SSP Negative, and 4 NSSP.
- 72 predictions that are safe, and 4 predictions that are not safe

RCL
RESILIENT COMPUTING LAB

FN*

- Not all FNs must be inside the NSSP
- there can be a residual small percentage, labelled as FN*, which appear as SSP.

- How many FNs can be in the SSP?
- Determined according to the ALR:

probability of FN* within SSP ≤ ALR.

SSP and NSSP

- how to separate SSP from NSSP?
- (and derive metric scores to quantitatively assess safety of an ML algorithm).

Safety of predictions defined based on the

ALR

derived by the safety specialists.

- We developed algorithms that given an ALR derive SSP_{ALR} and $NSSP_{ALR}$ values

Safety-Oriented Metrics

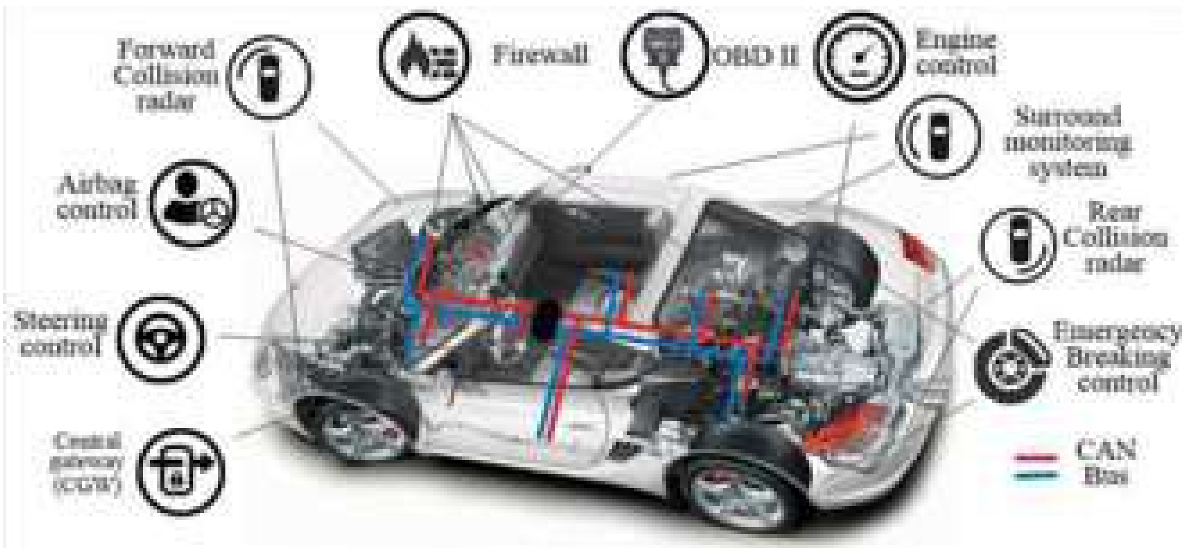
- The SSP_{ALR} and $NSSP_{ALR}$ ouput by our algorithms can then be used to compute:
- Sufficiently Safe Prediction rate

$$SSPr(ALR) = \frac{SSP_{ALR}}{NSSP_{ALR} + SSP_{ALR}}$$

- No Prediction rate – NPr(ALR):
  - $NPr(ALR) = 1 - SSPr_{ALR} = \frac{NSSP_{ALR}}{NSSP_{ALR} + SSP_{ALR}}$

EXAMPLE:

AN ML-BASED INTRUSION DETECTION SYSTEM FOR CONTROLLER AREA NETWORK (CAN) BUS



A representation of the architecture of car with a CAN Bus

Successful Security attacks will impair safety!!

Methodology and Experimental Setup

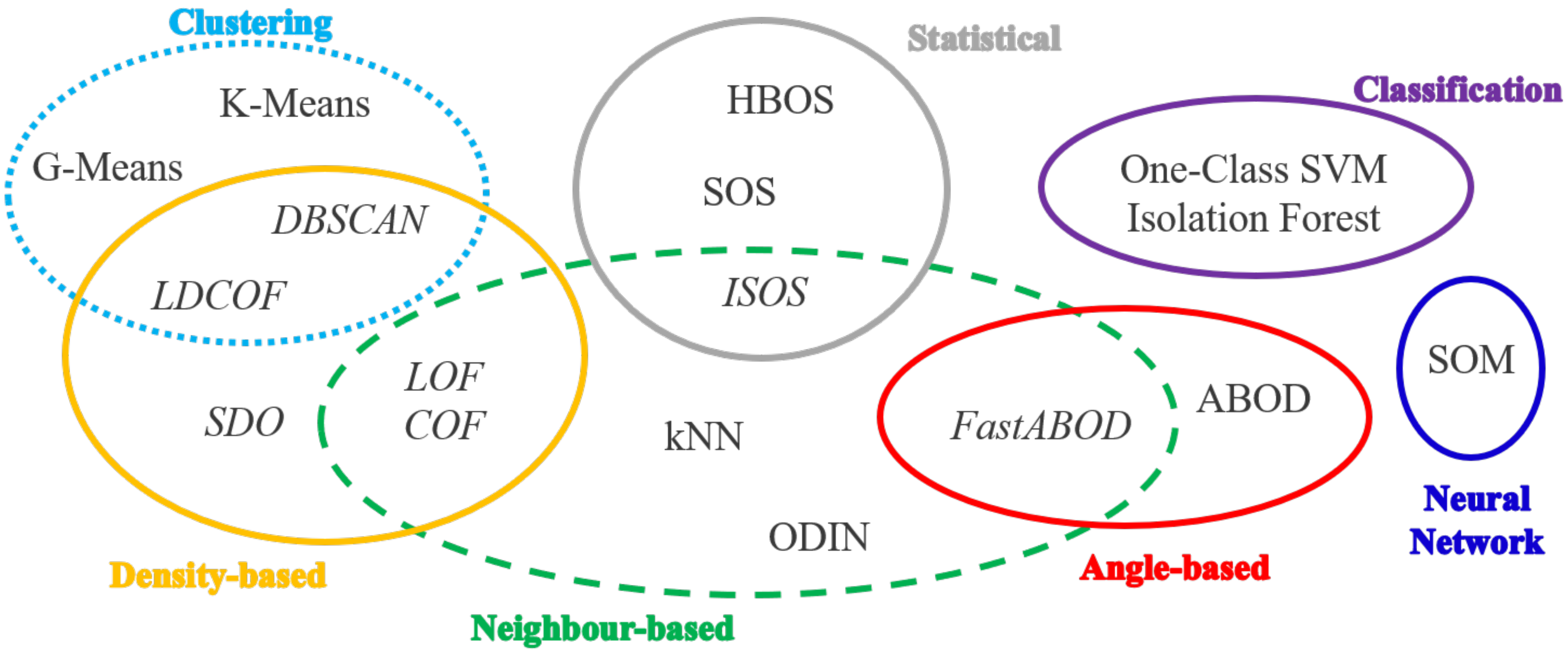
- Public datasets on attacks to build a solid baseline for our experimental study.
- Unsupervised algorithms (have potential in detecting both known and unknown – (zero day)
- Metrics: the two new metrics (with ALR set to 0.01), and many from the literature.
- A framework to run experiments: the RELOAD tool.

Attacks and (Public) Datasets

Datasets used in this study: name, release year, data point used, number of ordinal and categorical features, number and percentage of attacks.

Name	Year	# Data Points	Features		Attacks	
			Ord.	Cat.	#	%
ADFANet	2015	132 002	5	6	3	11.3
CICIDS17	2017	200 000	77	5	5	79.7
CICIDS18	2018	200 000	77	5	6	26.2
CIDDS	2015	200 000	5	7	4	14.4
ISCX12	2012	200 000	4	10	4	43.5
NSLKDD	2009	148 516	37	5	4	40.7
SDN20	2020	200 000	63	5	5	66.6
UGR16	2016	200 000	4	6	5	3.3
UNSW-NB15	2015	175 341	38	6	8	6.5

Unsupervised algorithms



We selected 12!

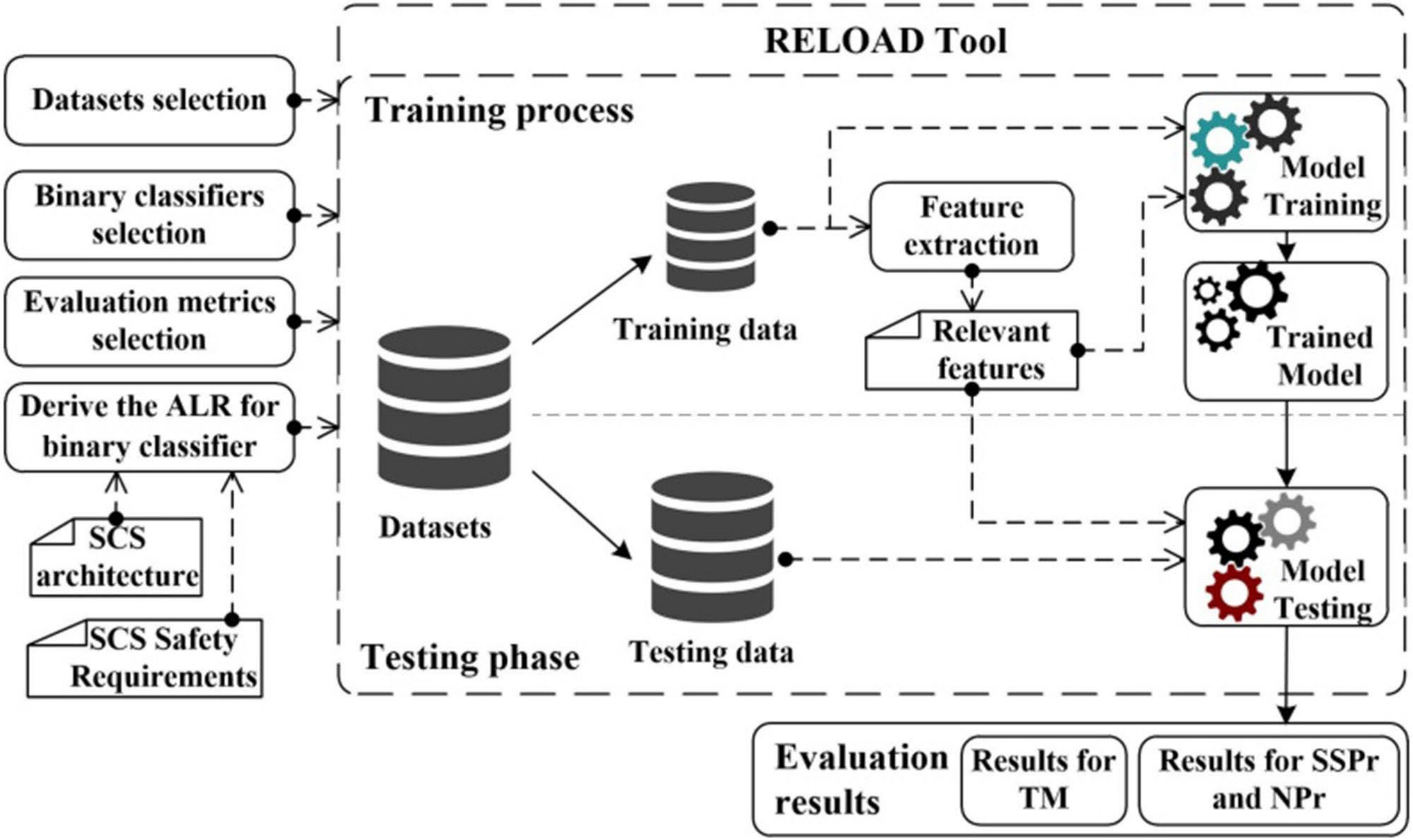
Evaluation metrics

- SSPr(0.01) and NPr(0.01) - using an ALR 0.01

- Most common metrics: Accuracy (ACC), Precision (P), Recall (R), False Positive Rate (FPR), F1-Score, Matthews Coefficient (MCC), Area Under the ROC Curve (AUC), Area under the Convex Hull of the ROC Curve (AUCH),
- Less used metrics: Gini index, H-measure (H), Kappa Statistics (KS), Youden Index, and Precision-Recall-Gain curve (PR-Gain).

- ACC(0.01) and MCC(0.01), Accuracy and Matthews Coefficient restricted to SSP(0.01): provide a measure **on the detection performance when providing sufficiently safe predictions**

The overall methodology



12 algorithms on 9 datasets: 108 experiments

The overall Evaluation

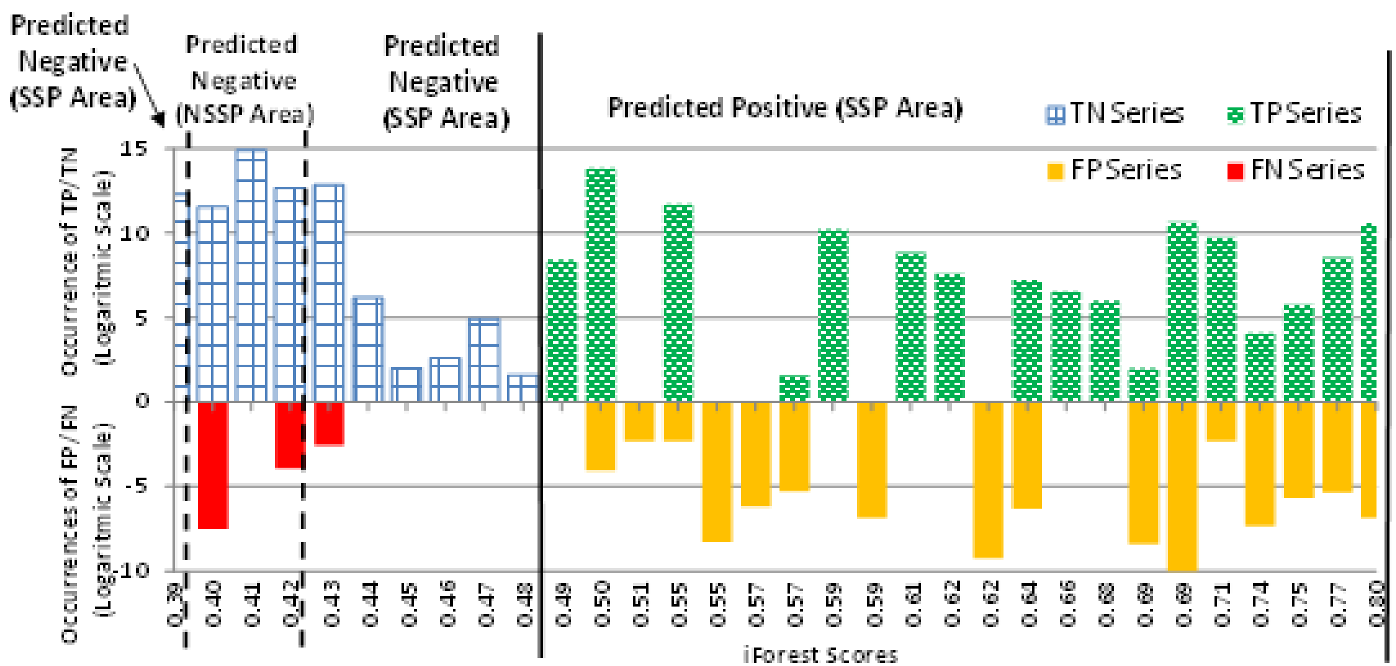
A portion of the results (metric scores) of applying the algorithms to the datasets, ordered by decreasing SSPr. Highlighted cases are those that are being explored through plots in this presentation

Case ID	Algorithm	Dataset	Traditional Metrics															Distribution-Based		
			FN %	FPR	P	R	F1	F2	MCC	ACC	AUC	AUCH	H	Gini	KS	Youden	PR-Gain	SSPr (0.01)	MCC (0.01)	ACC (0.01)
1	FastABOD	ADFANet	0.01	0.010	0.977	0.999	0.988	0.995	0.983	0.993	0.99	1.00	0.98	0.99	0.99	0.99	0.98	100.00	0.98	0.99
2	LOF	ADFANet	0.00	0.079	0.851	1.000	0.919	0.966	0.885	0.946	0.97	0.97	0.85	0.93	0.94	0.00	0.85	100.00	0.89	0.95
3	SVM	ADFANet	0.00	1.000	0.310	1.000	0.473	0.692	0.002	0.310	0.59	0.79	0.51	0.19	0.58	0.00	0.31	100.00	0.00	0.31
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	LOF	SDN20	0.01	0.900	0.691	0.999	0.817	0.918	0.262	0.701	0.56	0.71	0.36	0.12	0.43	0.00	0.69	100.00	0.26	0.70
16	iForest	SDN20	0.00	0.232	0.896	1.000	0.945	0.977	0.829	0.923	0.99	0.99	0.93	0.99	0.95	0.54	0.90	100.00	0.83	0.92
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	SOM	UNSW	0.01	1.000	0.379	0.999	0.550	0.753	0.000	0.379	0.56	0.65	0.09	0.13	0.20	0.02	0.38	100.00	0.00	0.38
20	SOM	SDN20	0.13	0.878	0.696	0.998	0.820	0.918	0.281	0.707	0.92	0.95	0.75	0.84	0.79	0.01	0.70	99.87	0.29	0.71
21	SVM	SDN20	0.13	0.894	0.693	0.998	0.817	0.917	0.261	0.702	0.92	0.95	0.76	0.84	0.80	0.00	0.69	99.82	0.26	0.70
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
31	ODIN	SDN20	0.12	0.395	0.835	0.990	0.906	0.955	0.691	0.862	0.60	0.74	0.39	0.21	0.45	-0.11	0.83	96.10	0.65	0.86
32	SDO	CIDDS	0.20	0.611	0.510	0.996	0.674	0.836	0.440	0.625	0.56	0.71	0.19	0.12	0.37	-0.34	0.51	95.92	0.40	0.60
33	SOM	CIDDS	0.08	0.783	0.448	0.998	0.619	0.801	0.308	0.521	0.58	0.72	0.20	0.16	0.43	0.00	0.45	95.55	0.21	0.48
34	SVM	CIDDS	0.08	0.781	0.449	0.998	0.619	0.801	0.308	0.522	0.58	0.72	0.20	0.16	0.43	0.00	0.45	95.53	0.21	0.48
35	FastABOD	SDN20	0.09	0.290	0.873	0.995	0.930	0.968	0.778	0.900	0.82	0.88	0.49	0.64	0.64	0.31	0.87	95.18	0.73	0.90
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
41	KMEANS	UNSW	0.73	0.765	0.439	0.980	0.607	0.787	0.290	0.518	0.63	0.68	0.13	0.26	0.24	0.00	0.44	91.74	0.01	0.44
42	GMEANS	UNSW	1.24	0.746	0.443	0.969	0.608	0.783	0.289	0.526	0.52	0.62	0.11	0.04	0.22	0.00	0.44	91.34	0.10	0.45
43	LOF	UNSW	6.58	0.811	0.384	0.827	0.525	0.672	0.220	0.651	0.57	0.64	0.08	0.14	0.18	0.16	0.43	90.32	0.22	0.63
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
66	iForest	ADFANet	0.03	0.397	0.713	0.984	0.827	0.915	0.636	0.794	0.84	0.89	0.65	0.69	0.77	0.58	0.71	69.08	0.00	0.71
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
77	ODIN	CICIDS18	1.97	0.075	0.972	0.950	0.946	0.951	0.876	0.938	0.67	0.97	0.81	0.94	0.87	0.74	0.90	51.30	0.00	0.93
78	ODIN	ISCX	0.80	0.468	0.139	0.872	0.240	0.425	0.219	0.559	0.69	0.77	0.04	0.37	0.45	0.00	0.13	50.03	0.00	0.14
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

iForest on ADFANet - ID 66

- Very few FNs, → very high Recall (99.2)….. However, many **FNs co-locate with TNs** ending in NSSP

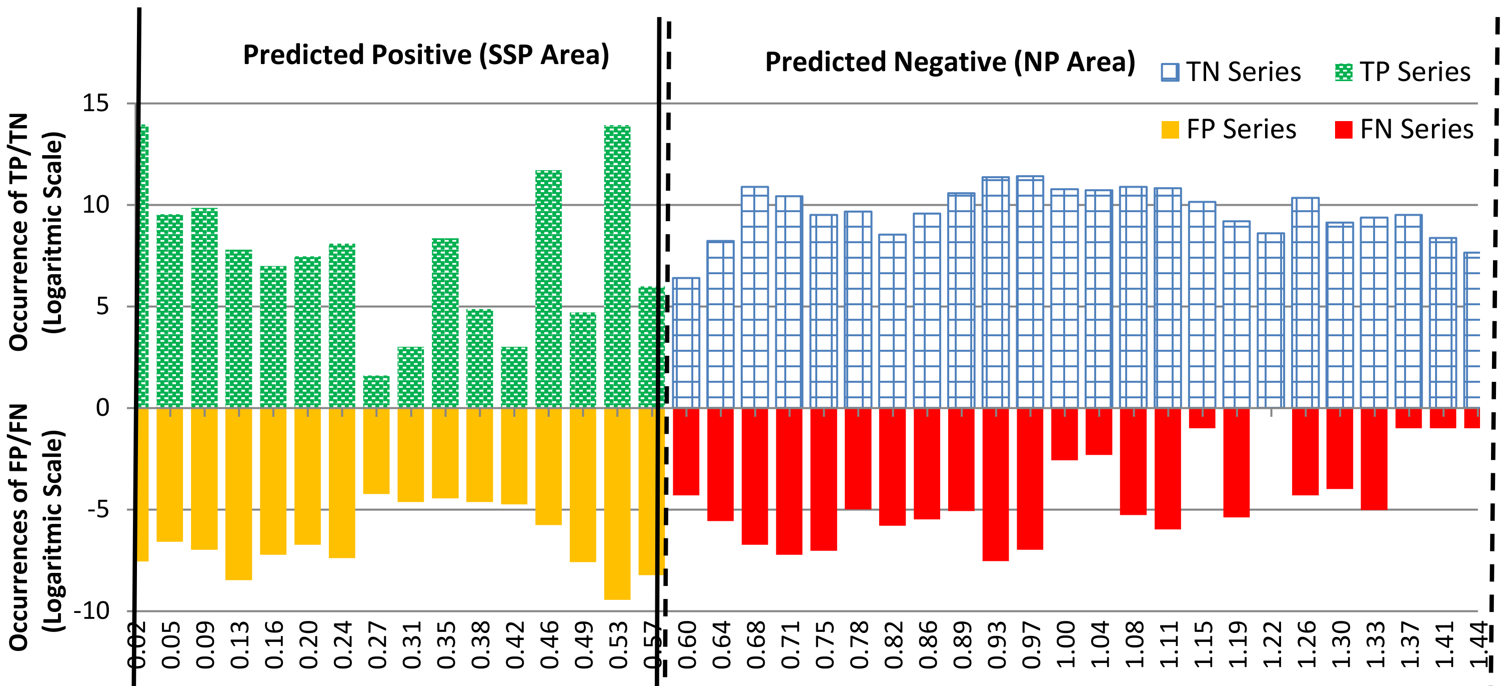
- SSPr(0.01)=69.08% → **only 69.08% sufficiently safe.**



RCL
RESILIENT COMPUTING LAB

ODIN on CICIDS18 - ID 77

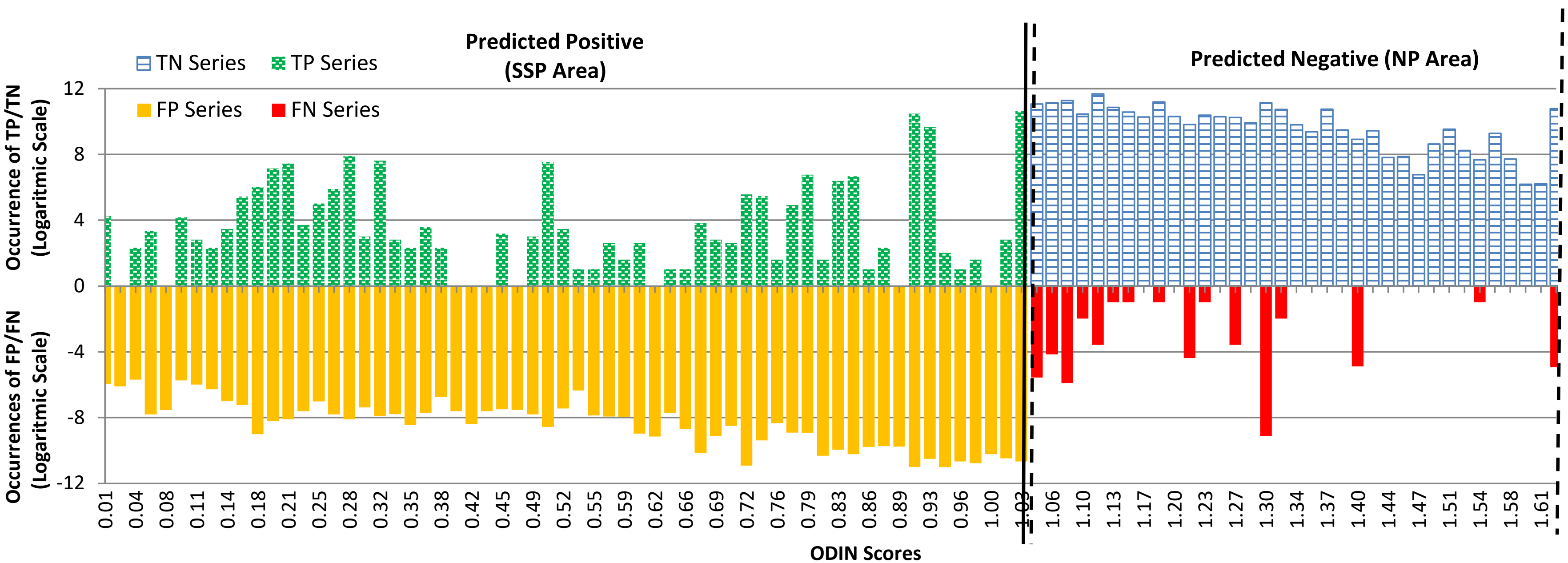
Example of very poor SSPr even with relatively low FN% and high Recall
Only 1.97% FNs but scattered distribution…….
→ NPr = 48.7% and SSPr = 51.3%



ODIN to ISCX - ID 78

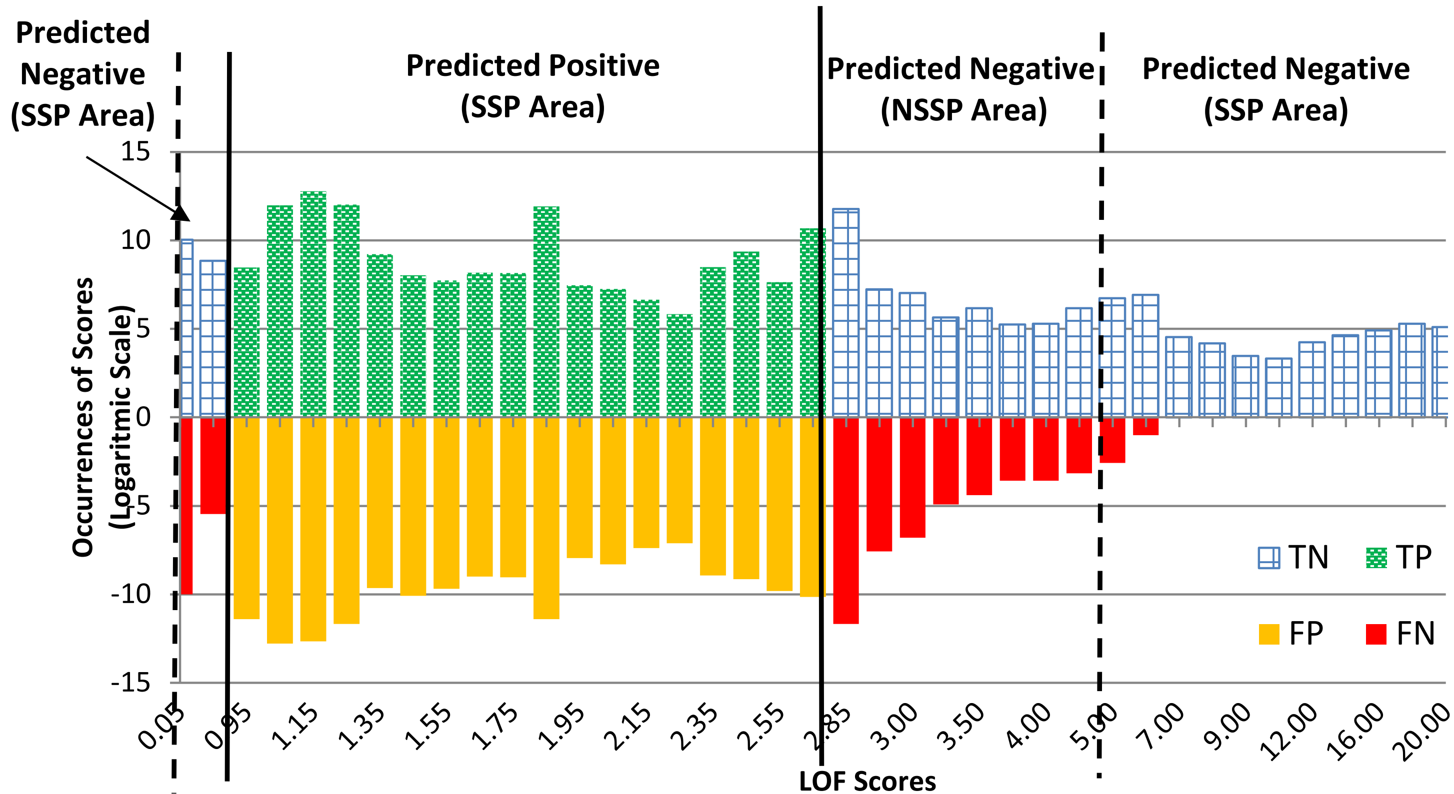
Very low FN 0.8%, but SSPr 50.0% only!!.

The distribution of FNs (red bars) overlaps completely with TNs (blue patterned bars), which all become NSSP.



LOF on UNSW - ID 43

- High FN 6.58%.
- Here FNs are mostly in a relatively small area allowing for a quite high SSPr of 90.32%.



Traditional Metrics

- 43 cases out of our 108 had SSPr(0.01) above 90%.→ (no more than 10% of the predictions are NSSP)
- We derived the "best" 43 cases for each of the traditional metrics.
  - (Recall shares 36, all the others less than 20!!)

Number of cases that result in a SSPr(0.01) ≥ 90 and fit in the best 43 for traditional metrics.

R	F2	Youden	H	F1	PR-Gain	AUC	AUCH	Gini	P	KS	MCC	ACC	FPR
36/43	18/43	16/43	16/43	14/43	13/43	10/43	10/43	10/43	10/43	9/43	6/43	6/43	4/43

RCL
RESILIENT COMPUTING LAB

Comments

3 examples of low FN but scattered..

1 with high FN but concentrated ……

- It is evident that SSPr catches aspects of the behavior of ML algorithms which escape traditional metrics!!
- Metrics based on distributions should be used together to traditional ones:

  Cases with perfect SSPr but many FPs…..

  IDs 1 and 3: both achieve SSPr of 100,

  but 1 shows an accuracy of 0.993

  while 3 an accuracy of only 0.310 (and MCC = 0).

- **3 would be not usable because of availability**

Conclusions: Take from this journey

ML as Controller coupled with a safety monitor

- Nasty surprises – more learning improved the ML controller but WORSENED the system
- Need for joint management and testing

ML as as a safety checker

- care with measures and proper derivation from safety cases
- Selection of proper ML need deep analysis combining traditional and ad hoc measures

ML can bring huge benefit to Safety critical systems but integration needs a lot of attention!!

Relevant papers

- Zoppi, T., Ceccarelli, A., Bondavalli, A. Evaluation of Anomaly Detection Algorithms Made Easy with RELOAD. International Symposium on Software Reliability Engineering, ISSRE, 2019, pp. 446–455,
- T. Zoppi, A. Ceccarelli, T. Capecchi, A. Bondavalli. Unsupervised Anomaly Detectors to Detect Intrusions in the Current Threat Landscape. **ACM/IMS Trans. Data Sci.** 2, 2, Article 7 (April 2021), 26 pages.
- Zoppi, A. Ceccarelli and A. Bondavalli. MADneSs: A Multi-Layer Anomaly Detection Framework for Complex Dynamic Systems. **IEEE Transactions on Dependable and Secure Computing**, vol. 18, no. 2, pp. 796-809, 1 March-April 2021.
- Terrosi, F, Strigini, L. and Bondavalli, A. Impact of Machine Learning on Safety Monitors. Proc. of 41st international conference on Computer Safety, Reliability and Security -SAFECOMP 22 – Munich, Germany.
- Gharib, M., Zoppi, T., Bondavalli, A., 2021. Understanding the Properness of Incorporating Machine Learning Algorithms in Safety-critical Systems. In: Proceedings of the 36th Annual ACM Symposium on Ap-plied Computing. ACM, pp. 232-234.
- Mohamad Gharib, Tommaso Zoppi, Andrea Bondavalli. 2022. On the Properness of Incorporating Binary Classification Machine Learning Algorithms Into Safety-Critical Systems. **IEEE Transactions on Emerging Topics in Computing**. Early access (https://www.computer.org/csdl/journal/ec/5555/01/09788529/1DUa6JUMQyk)

Credits to: Francesco Terrosi,Tommaso Zoppi, Mohammad Gharib, Lorenzo Strigini, Andrea Ceccarelli and the entire RCL-Group@UNIFI

QUESTIONS??