Benchmarking Machine Learning-based Online Failure Prediction Models

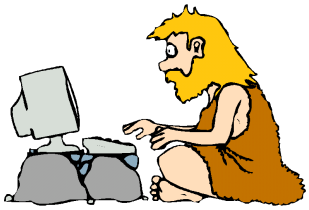
Marco Vieira

mvieira@dei.uc.pt
University of Coimbra, Portugal

1

Marco Vieira

- Full Professor at the University of Coimbra
- BSc and MSc in Informatics Engineering
- PhD in Computer Science
- Dependability and Security Assessment and Benchmarking
- Fault Injection and Vulnerability & Attack Injection
- Online Failure Prediction
- Resilience Benchmarking
- Robustness and Security Testing
- Software Verification & Validation



2

Coimbra, Portugal

- City in the center of Portugal
  - 200 Km to the north of Lisbon
- ~ 150 000 people
- Most activity around the University
- Many centuries of history!

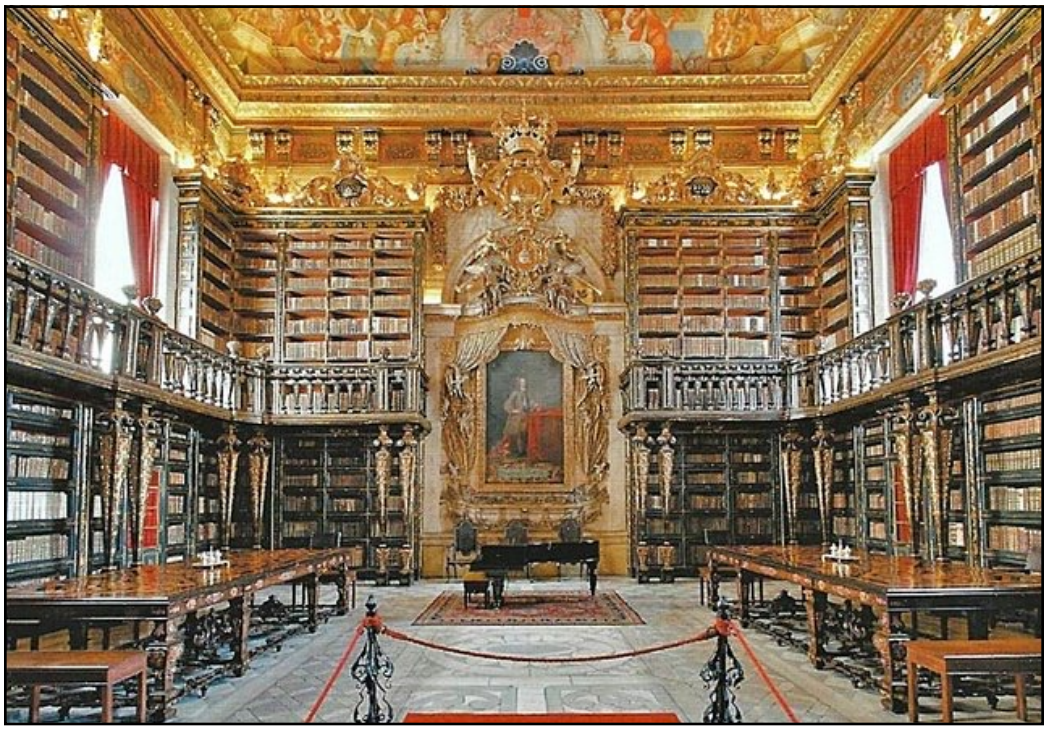
University of Coimbra

- One of the oldest in the world
  - Created in 1290
- 8 schools
  - Sciences and Technology
  - Law
  - Medicine
  - Pharmacy
  - Economics
  - Arts and Humanities
  - Psychology and Education Sciences
  - Sport Sciences and Physical Education
- About 25000 students
  - ~20% of which are foreigners

http://www.uc.pt

5

6



7

8

Benchmarking Machine Learning-based Online Failure Prediction Models

João Campos, **Marco Vieira**

mvieira@dei.uc.pt
University of Coimbra, Portugal

9

What is Online Failure Prediction (OFP)?

*Technique that attempts to **predict failures** by using past data and the current system state*

-- Salfner, Lenk, Malek, "A survey of online failure prediction methods," ACM Computing Surveys, 2010

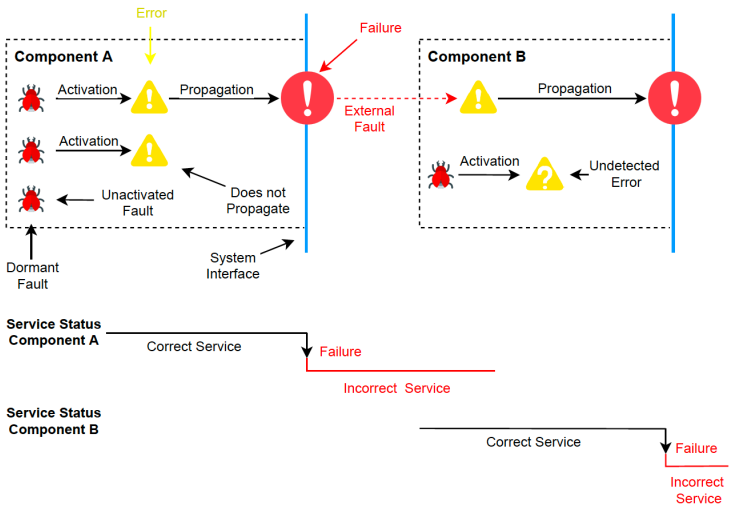
- Allows taking preemptive measures to mitigate possible hazards
  - Failures: crash, hang, performance degradation…
  - Hazards: data corruption, data loss, service degradation…
  - Measures: saving data, restarting a system, live migration…
- Past data is used to train a model
- Current system state is collected and fed to the model for predictions

10

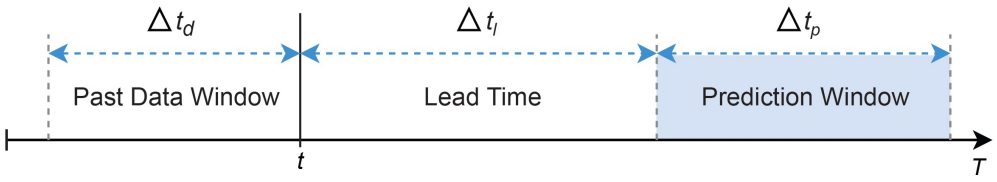
Why Online Failure Prediction?

- Software-intensive systems are becoming more and more complex
- Complexity makes fault avoidance and fault detection very difficult
  - Systems end-up being shipped with residual faults
- Fault tolerance mechanisms are expensive and not applicable in many types of systems
- Particularly relevant in the context of residual faults (e.g. software bugs) that cannot be tolerated by other fault tolerance mechanisms
  - Seen as the ultimate barrier against system failures!
- Useful for large servers, including cloud and virtualized systems, where the costs and risks associated with a failure are not negligible

Fault-Error-Failure-Fault



The Online Process



What are the Challenges?

- Despite its potential, OFP is still not widely adopted…
- Developing effective failure prediction models is not simple
  - Yet again the complexity issue…
- Lack of data for training models
- Modern software systems evolve over time
- Techniques and tools for assessing and comparing alternative solutions for a particular system installation are not available

Addressing some Challenges…

- Machine-Learning (ML) for developing prediction models
  - ML algorithms have shown their ability to adapt and extract knowledge in a variety of complex problems
- Software fault injection to facilitate the generation of failure data
  - Fault injection has been used for decades in the validation of critical systems
- Online learning to detect and handle system evolution and thus keep models updated
- **Benchmarking for assessing and comparing solutions**
  - Benchmarking has been used in multiple domains for assess and compare alternative systems, components, tools…

What is Benchmarking?

*Procedure that allows a **fair and sound assessment and comparison** of alternative solutions according to some characteristics*

- Typical components: metrics, workload, setup and procedure
- Performance benchmarking:
  - Metrics: response time, throughput…
  - Workload: transactions, computations…
- Dependability benchmarking:
  - Metrics: response time, throughput, availability, time to recover…
  - Workload: transactions, computations…
  - Faultload: software faults, hardware faults…

What is Benchmarking?

- Security benchmarking:
  - Metrics: response time, throughput, availability, time to recover…
  - Workload: transactions, computations…
  - Attackload: *you know what I mean, right?*
- Resilience benchmarking, trustworthiness benchmarking…
- The benchmark design must fulfill a set of properties:
  - Representativeness, repeatability, portability, non-intrusiveness, promptness, ease of use and implementation

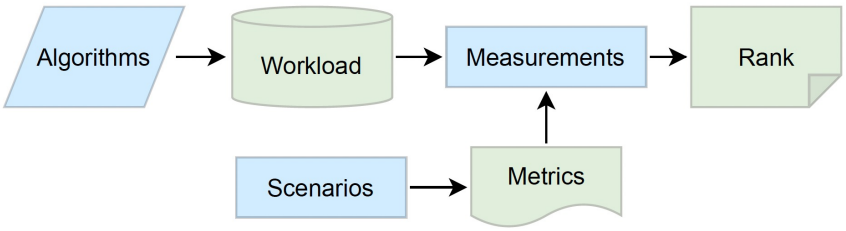
Benchmarking for OFP

- Guidelines for implementing a procedure to assess and compare failure prediction models
  - Assuring a fair and sound comparison, while fulfilling the key properties
- Choosing the adequate metrics for the assessment
  - Depending on the application scenario
- Preparing and validating the workload
- Comparing the alternative models
- Selecting the most adequate predictor (or set of predictors)

Outline

- Benchmarking framework
- Key components of the framework:
  - Scenarios
  - Metrics
  - Workload (dataset)
- Benchmarking procedure
- Case Study: Linux OS (version 3.16.82)

Benchmarking Framework



Scenarios

- Requirements representing real contexts, having constraints with different criticality, where failure prediction will be used

- High-criticality: failures missed may be problematic
  - Goal: select a model that is able to predict the highest number of failures
  - Example: home-banking system
- Medium-criticality: failures may be missed to reduce false-alarms
  - Goal: model reporting few false-alarms at the cost of missing some failures
  - Example: email server
- Minimum-criticality: every false-alarm is a cause of concern
  - Goal: select a model reporting the lowest number of false-alarms
  - Example: forum server

Metrics

- Allow characterizing the effectiveness of the algorithms
  - Must be easy to understand and allow the comparison among alternative algorithms from different perspectives and scenarios

- One main metric is proposed to rank the tools
  - According to the goal of the scenario in the specific context of OFP
- A tiebreaker metric is used only when there is a tie
  - i.e., no statistical difference, as defined in the benchmarking procedure

Scenario	Metric	Tiebreaker
High-criticality	informedness $*$ recall	recall
Medium-criticality	f-measure	precision
Minimum-criticality	markedness	precision

Informedness

- How consistently a predictor predicts the outcome of a TP and a TN
  - i.e., how informed a predictor is for the specified condition, versus chance

$$Informedness = \frac{TP}{TP+FN} + \frac{TN}{FP+TN} - 1$$

- Counterpart of recall, it is a measure of how informed the model is about positives and negatives
- Due to the data imbalance, most of the time this metric prioritizes models predicting more failures, but still considering FPs
  - However, the metric is not concerned with which condition is better predicted
- To overcome this behavior, we rank the models based on the product of their informedness and recall
  - Which makes this suitable for the high-criticality scenario

Markedness

- How consistently the predictor has the outcome as a marker, i.e., how marked a condition is for the specified predictor, versus chance

$$Markedness = \frac{TP}{TP+FP} + \frac{TN}{FN+TN} - 1$$

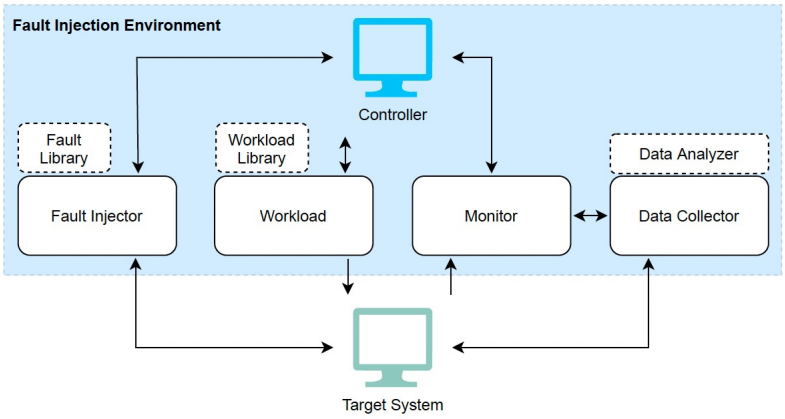
- Counterpart of precision, it is a measure of trustworthiness of positive and negative predictions
- The metric considers the proportions of the predicted positives and negatives that are correctly classified
- As failures are rare, each FP will lower the metric more than a FN
  - Thus, satisfying the requirements for the minimum-criticality scenario, by minimizing the FPs whilst also classifying some failures

Workload (Dataset)

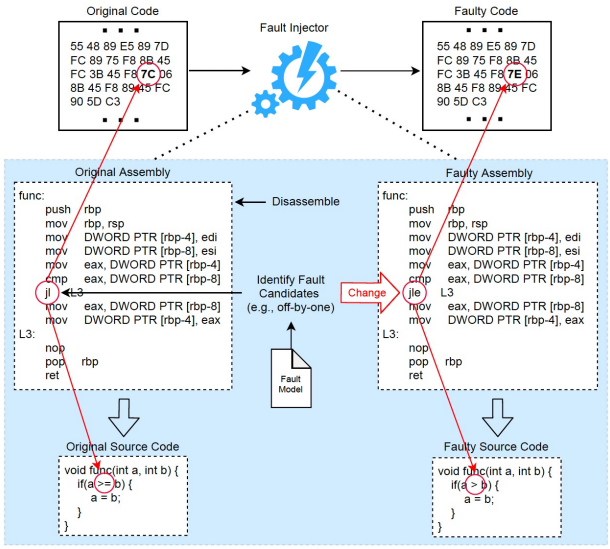
- Data needed to train and test the failure prediction algorithms.
  - Should mimic the behavior of the target system, taking into account the hardware and software, workload, failure modes, etc.

- Real workloads
  - Contain data collected from real environments
- Realistic workloads
  - Artificial, based on a subset of operations from real systems in the domain
  - Realistic workloads of failure data can be generated through fault injection
- Synthetic workloads
  - Artificially generated based on assumptions and rules
  - Representativeness is highly questionable

Fault Injection to Generate Data

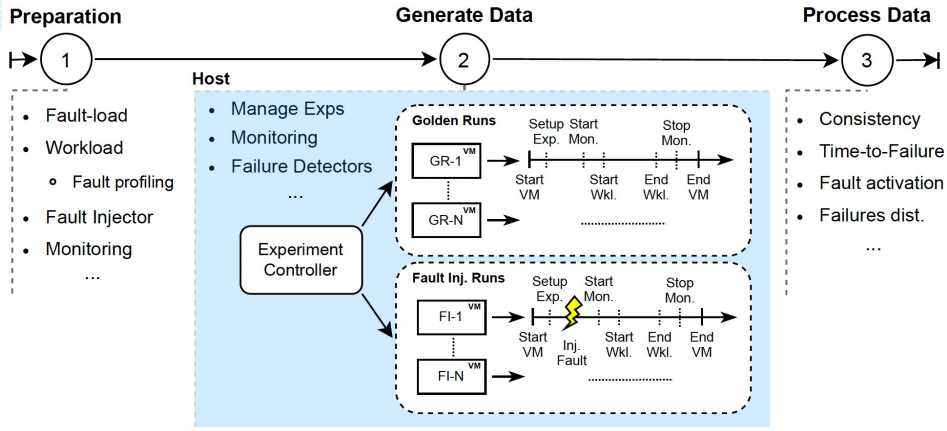


Machine-Code Fault Injector



27

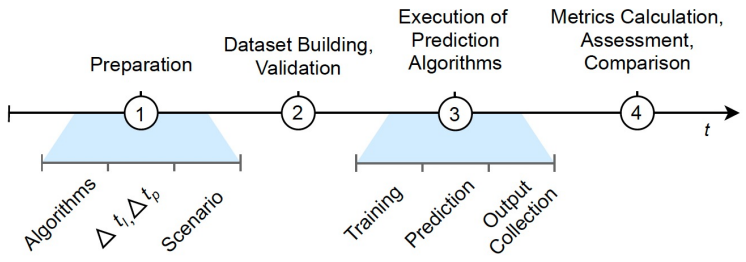
Data Generation Process



28

Benchmarking Procedure

- Rules that must be followed, including the phases that must be conducted, towards the assessment of the performance



Preparation

- Consists of identifying the set of parameters for benchmarking the failure prediction models
  - e.g., the failure modes to predict, the intervals relative to the failure prediction task Δ_{tl} and Δ_{tp}
- Includes the selection of the algorithms and their configurations, as well as any other techniques and configurations necessary
  - e.g., dimensionality reduction, sliding windows
- Another key aspect is to decide on the scenario taking into account the environment in which the predictor will operate

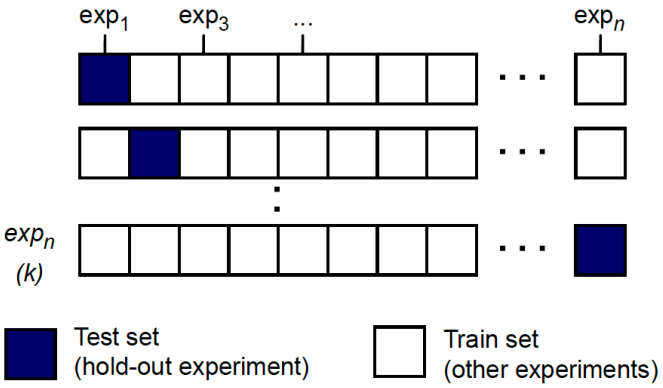
Dataset Building and Validation

- A dataset is a simple table where the columns are divided into:
  - Set of features: the monitored variables
  - Target: whether a failure will or not occur within the specified Δ_{tl}, Δ_{tp}
- Instances should be in the same sequence as they were collected
  - To allow the evaluation of techniques considering order (e.g., sliding windows)
- No "false-predictors" can exist
  - i.e., features directly related to the target
- The dataset should only contain sanitized data
  - e.g., in case of data generated through fault injection, runs that fail immediately after injection should be eliminated, as these are not representative
- …

Metrics Calculation

- Each predictor must be evaluated using a test datasets to get an estimate of the generalization error
- Given the experiment-based nature of OFP techniques such as *k-fold* cross validation are not adequate
  - e.g., there is no sequence between multiple failures, and failures will always be at the end of the experiments
- We observed that using k-fold cross-validation for OFP leads to overfitted solutions
  - When our best models (selected using k-fold cross-validation) were put online they failed to detect almost all the incoming failures…
- Experiment-wise Leave-one-out Cross-Validation (ELOOCV)

Experiment-wise Leave-one-out Cross-Validation



Case Study: Linux Failure Data

- Target: Linux OS (version 3.16.82)

- Data generation:
  - *stress-ng* CPU profile as workload
  - Fault injector from [Yoshimura et al., 2012], ported to kernel version 3.16.82
  - 262 numeric features observed
  - Only about 1.2% of the fault injection experiments could be used

Workload	Fault Injection	Golden
cpu	4472	100

- The datasets were labeled according to different pairs of Δ_{tl} ,Δ_{tp}
  - e.g., for [20, 40] we had approximately 18000 non-failure samples and 69, 239, 437, 435, and 153 hang, crash, CPU, memory, and kernel samples respectively

Failure Modes

- Fail-stop failures:
  - Crash: OS crashes
  - Hang: OS hangs
  - Performance: deviates more than 5% from the lowest baseline value
  - Infinite execution: workload has not finished after 15 minutes
  - I/O: cannot write to disk
  - File system corruption: using *fsck* functionality
- Non-fail-stop failures:
  - CPU/execution-related: e.g., invalid opcode
  - Memory-related: e.g., segmentation fault
  - Kernel-related: e.g., recursive fault

Techniques used in the Experiments

Process / Step	Techniques
$\Delta t_l, \Delta t_p$	[20, 40, 60]/[20, 40, 80]
Scenarios	*High/Medium/Minimum-criticality*
Feature Selection	Variance, Correlation
Sampling	Random under/oversampling, ADASYN
Sampling Ratios	Over.: [2, 4], Under.: [1, 10], Over. and Under.: [(2, 1), (2, 5), (4, 1), (4, 5)]
Algorithms	DT, Bagging, RF, XGBoost

Overall Results (Fail-stop Failures)

Scen. \ Rank		1st		2nd		3rd
crash	*High*	RF (0.956)	>	XGB. + ROS (2) + RUS (1) + Corr. (0.935)	=	Bag. + RUS (10) + Corr. (0.865)
	Med.	RF + ADA. (2) + Corr. (0.525)	>	Bag. + RUS (10) + Corr. (0.287)	>	XGB. + Corr. (0.132)
	Min.	RF + ADA. (2) + Corr. (0.375)	>	Bag. + RUS (10) + Corr. (0.168)	>	XGB. + Corr. (0.070)
hang	*High*	RF + ROS (4) (0.998)	>	Bag. + ROS (4) + Corr. (0.996)	=	XGB. + ADA. (4) + Corr. (0.994)
	Med.	Bag. + ROS (4) (0.989)	=	RF + ROS (4) + Corr. (0.982)	>	XGB. + Corr. (0.513)
	Min.	Bag. + Corr. (0.999)	=	XGB. + ROS (2) + Corr. (0.999)	>	RF + ROS (4) + Corr. (0.971)

$a \tilde{>} b$; a has significant differences from b
$a = b$; a does not have significant differences from b
(#.###) ; value of the performance metric according to the scenario
RF=Random Forest (RF); **Bag**=Bagging; **XGB**=XGBoost; **DT**=Decision Tree (DT); **ADA**.=ADASYN; **ROS**=Random Oversampling; **RUS**=Random Undersampling; **Corr.**=Correlation (feature selection)

Benchmarking Properties

- Promptness
  - Most experiments were completed in less than a couple of hours
  - Directly related to the dimension of the datasets
- Intrusiveness
  - Does not require any kind of modification to the failure prediction models
- Portability
  - Can be used to assess and compare any kind of failure prediction model
- Repeatability
  - ELOOCV assures that the results are not biased by the data split or data leaks
- Representativeness
  - Related to the dataset, which is generated by means of fault injection

Highlights and Limitations

- This does not intend to be an extensive research of ML algorithms and techniques or even the impact of different Δ_{tl}, Δ_{tp} values
- We try to demonstrate:
  - How the performance of the algorithms varies under different conditions
  - How complicated it is to choose the best model for different scenarios
  - How our proposal can assist in the assessment and comparison task
- We do not dwell on the specific performance of the models as the focus is on comparison and ranking amongst the different solutions

Take-Away(s)

- Properly assessing prediction models is a complex task
  - Given the plethora of algorithms and techniques and their inherent limitations
- The lack of common procedures for supporting comparisons makes it difficult to promote the widespread use of OFP
- The evaluation should be conducted using data from the system in which OFP will be deployed
- Different metrics are needed to effectively characterize and rank the algorithms under evaluation

Some Ideas for the Future

- Assessing the performance of the models in a real-world production system
- Researching concept drift, batch, and online learning to keep the models updated
- Time series and deep learning to increase the predictive performance
- White-box explainable ML algorithms to infer and determine the causes of failure
- Generative ML techniques to generate failure data
- …

Q&A

Recall

- Proportion of failures that are correctly identified as such:

```
recall = TP / (TP + FN)
```

- Metric is used only as a tiebreaker in the high-criticality scenario, where the best model is the one that correctly predicts most failures
  - The TN and FP are not considered, and the data is highly imbalanced

Precision

- Proportion of positive predictions that were correctly classified:

```
precision = TP / (TP + FP)
```

- Adequate tiebreaker for both medium- and minimum-criticality scenarios, where the purpose is also to reduce/avoid FPs
  - i.e., from a list of models predicting the same number of failures, the best is the one with fewer FPs

F-Measure

- Harmonic mean of precision and recall, where TPs have twice the weight of the FPs
- Suitable for the medium-criticality scenario where it is preferable to predict fewer failures than having an added cost due to FPs

Algorithms' Hyperparameters

Alg.	Hyperparameters
DT	**min_samp_split**: [001, .01, 2], **max_feat.**: [.1, .55, None], **min_samp_leaf**: [.001, .01, 1]
RF	**estimators**: 100, **max_feat.**: [.1, .55, auto], **min_samp_leaf**: [.001, .01, 1], **min_samp_split**: [.001, .01, 2]
Bag.	**max_features**: [.1, .55, 1.0], **estimators**: 100 **max_samples**: [.1, .55, 1.0],
XGBs.	**estimators**: 100, **learning_rate**: [.1, .3], **max_depth**: [7], **subsample**: [0.7, 1], **min_child_weight**: [1, 5], **colsample_bytree**: [.7, 1]

Overall Results (Non-fail-stop Failures)

cpu	*High*	Bag. + RUS (1) + Corr. (0.877)	=	XGB. + RUS (1) + Corr. (0.839)	>	RF + RUS (1) (0.786)
	Med.	Bag. (0.758)	>	RF (0.638)	>	XGB. + ADA. (4) (0.552)
	Min.	Bag. (0.947)	>	RF + Corr. (0.911)	>	XGB. + ADA. (4) (0.446)
memory	*High*	Bag. + ADA. (4) + RUS (1) (0.960)	=	RF + ADA. (4) + RUS (5) (0.956)	>	XGB. + ROS (2) + RUS (1) (0.949)
	Med.	Bag. (0.742)	>	RF + Corr. (0.714)	>	DT (0.675)
	Min.	Bag. + ADA. (4) (0.995)	>	RF + ROS (2) (0.810)	>	DT (0.651)
kernel	*High*	RF + ADA. (2) + Corr. (0.982)	>	Bag. + ROS (4) + RUS (5) + Corr. (0.951)	>	DT + ADA. (4) (0.949)
	Med.	RF + ADA. (2) (0.331)	>	Bag. + ROS (4) + Corr. (0.187)	>	XGB. (0.181)
	Min.	RF + ADA. (2) (0.266)	>	Bag. + ROS (4) (0.103)	>	XGB. (0.100)

Benchmarking Machine Learning-based Online Failure Prediction Models

João Campos, **Marco Vieira**

mvieira@dei.uc.pt
University of Coimbra, Portugal