Practical Challenges and Insights in Concept Drift Research

Discovering Drift Phenomena in Evolving Landscape (DELTA 2024)
ACM SIGKDD 2024 workshop

Heitor Murilo Gomes
https://heitorgomes.com/

Victoria University of Wellington, New Zealand

VICTORIA UNIVERSITY OF WELLINGTON TE HERENGA WAKA

https://capymoa.org/

TAIAO Time-Evolving Data Science and Artificial Intelligence for Advanced Open Environmental Science

Heitor Murilo Gomes

Senior lecturer at the Victoria University of Wellington (VuW) in New Zealand. Before joining VuW, Heitor was co-director of the AI Institute at the University of Waikato.

Research interests: ML for data streams, ensemble learning, semi-supervised learning, concept drift detection/adaptation, …

PI of a few research projects ranging from applied to fundamental research, i.e. ML for energy prediction, novel methodologies for stream learning, …

Leads the ***capymoa*** (https://capymoa.org/) open source library for data stream learning, and provide support for MOA (Massive On-line Analysis).

https://heitorgomes.com/

About this talk

Brief introduction

Practical challenges

- Simulation

- Evaluation

- Using drift detectors

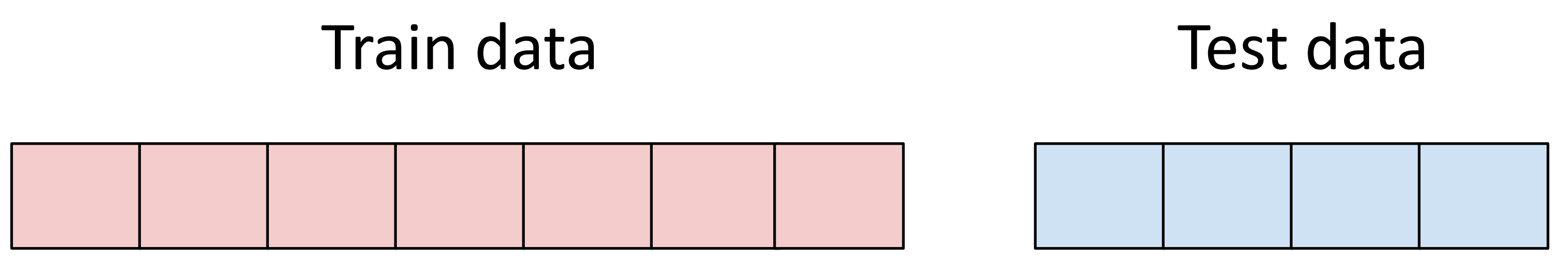
Other challenges

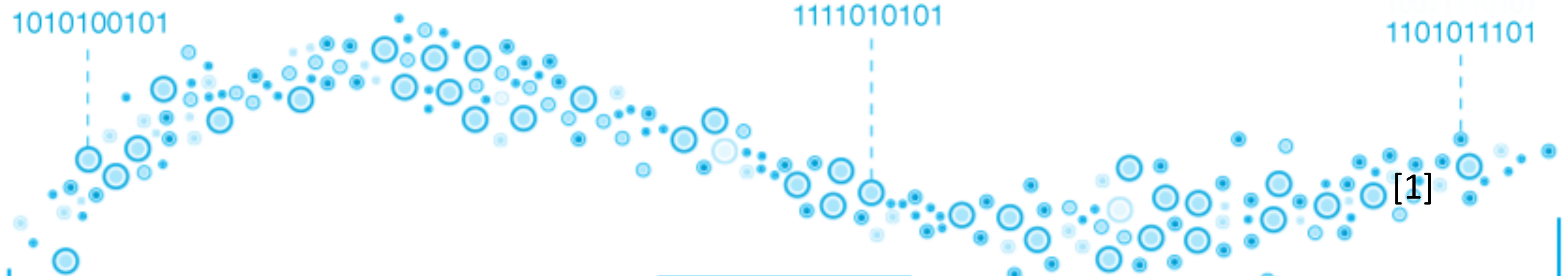
- Recurrent concept drifts

- Delayed & Unsupervised drift detection

Brief Introduction

Batch Learning vs Stream Learning (SL)





Learns from **static** data

Uses a **large** amount of **computing**

Can only **predict** after (**extensive**) **training**.

Should **re-train** after a **concept drift**

Learns from a **stream of data**

Incrementally online learn form instance/mini-batch.

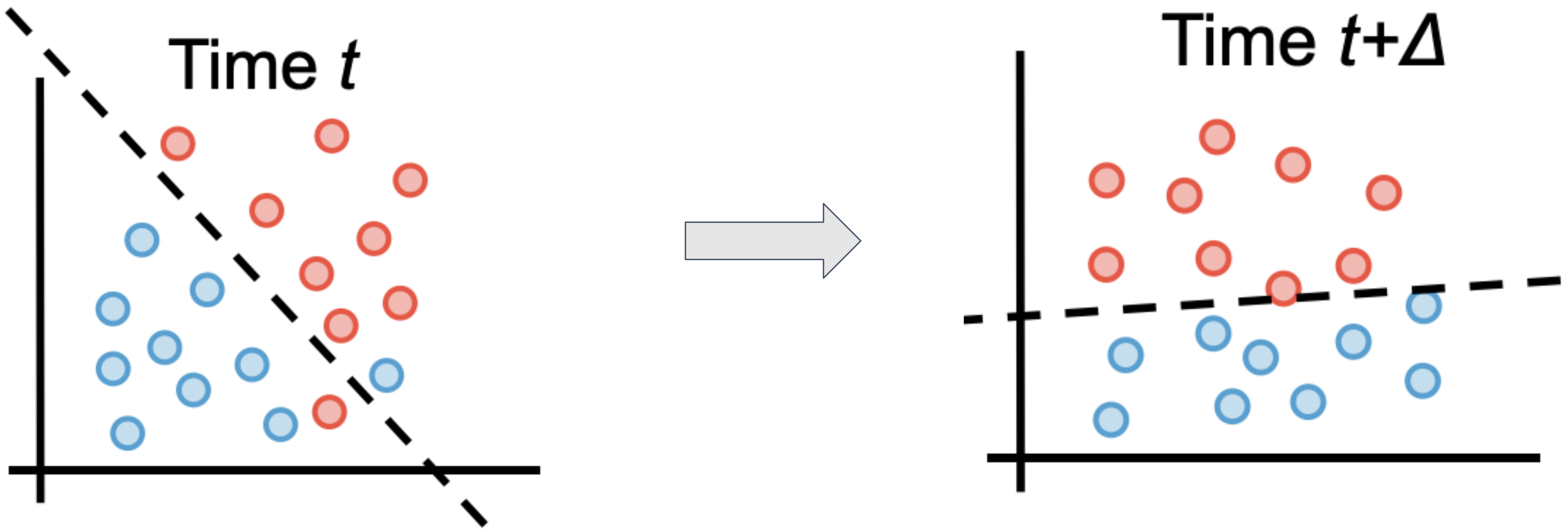
Should use **limited computing** resources.

Able to **predict** at **any given moment**.

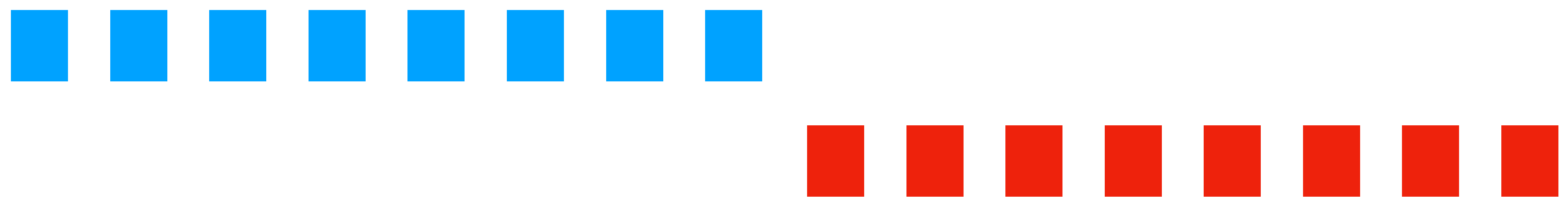
Should **adapt** to **concept drifts online**.

[1] https://www.onaudience.com/resources/what-is-data-stream-and-how-to-use-it/

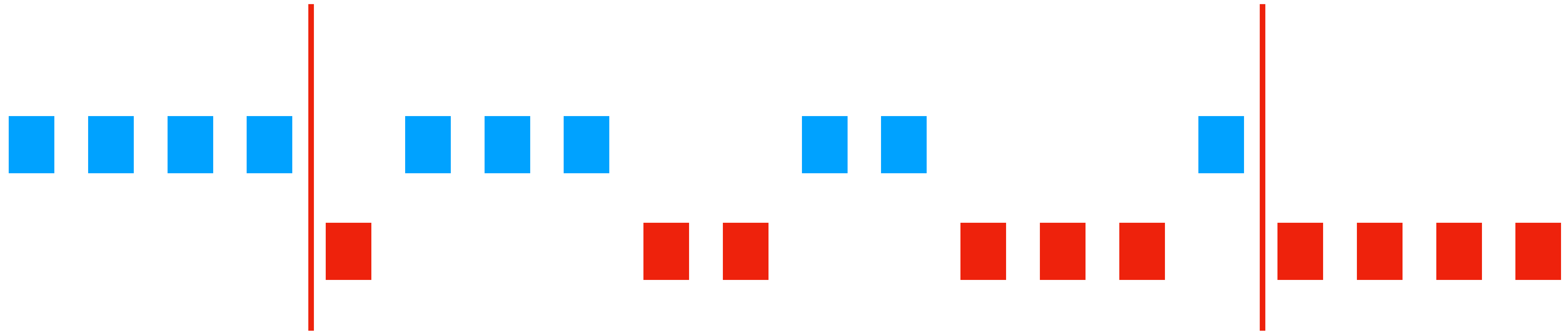
Concept Drift



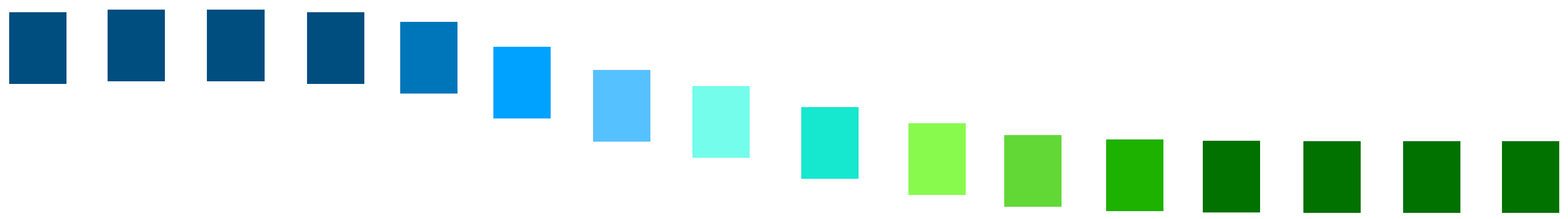
Concept Drift (categorisation)



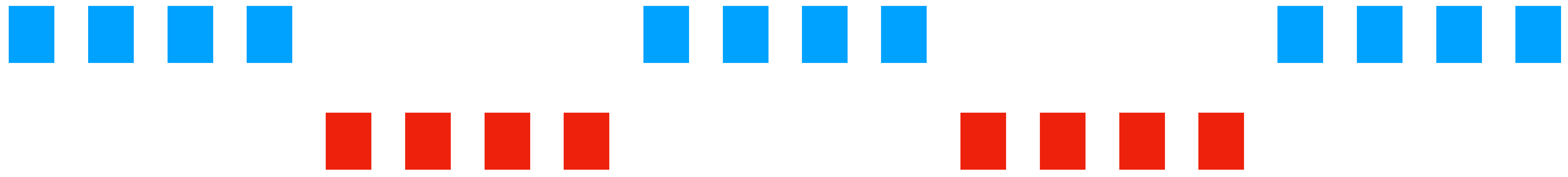
abrupt drift



gradual drift



incremental drift



recurrent concept drift

Practical challenges

CapyMOA

Machine learning for data streams

https://capymoa.org/

https://github.com/adaptive-machine-learning/CapyMOA

v0.7.0

CapyMOA

A machine learning library for streaming data based on four pillars:

- **Efficiency**
- **Interoperability**
- **Accessibility**
- **Flexibility**

First released on May 03, 2024

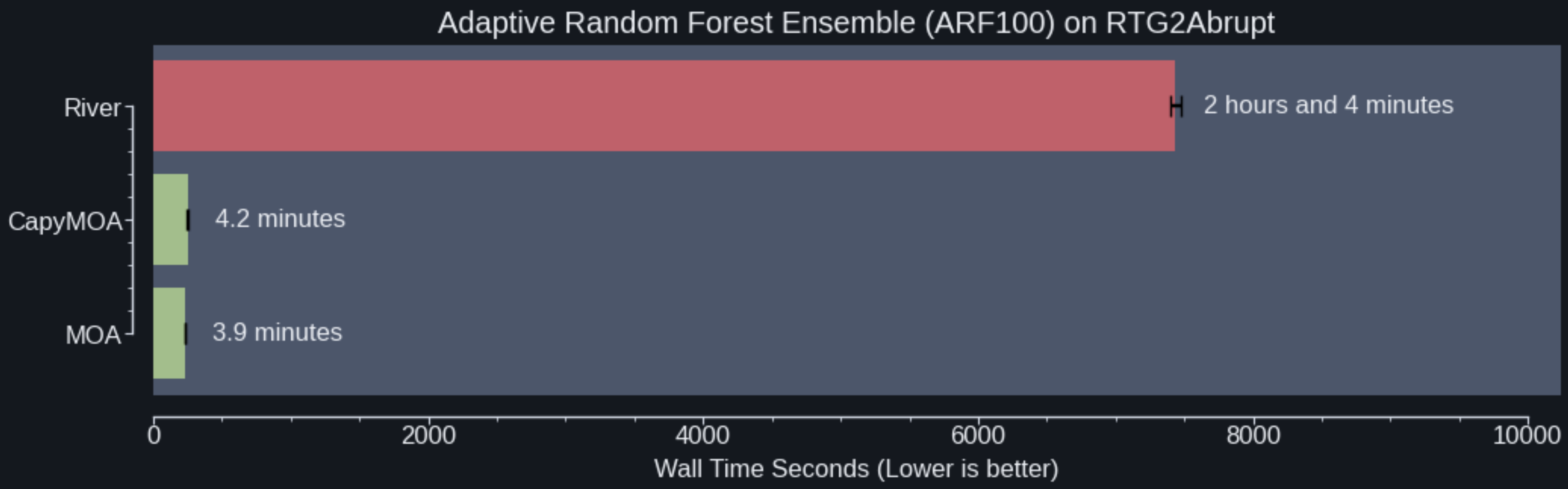
Other frameworks: **MOA** (java)[1], **river** (python)[2] and **scikit-multiflow** (python)[3]

[1] Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., & Seidl, T. (2010). Moa: Massive online analysis, a framework for stream classification and clustering. In *Workshop on applications of pattern analysis* (pp. 44-50). PMLR.

[2] Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T. and Bifet, A., 2021. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, *22*(110), pp.1-8.

[3] Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, *19*(72), 1-5.

Why? Efficiency



Reproducibility: https://github.com/adaptive-machine-learning/CapyMOA/blob/main/notebooks/benchmarking.py

Simulating Concept Drifts

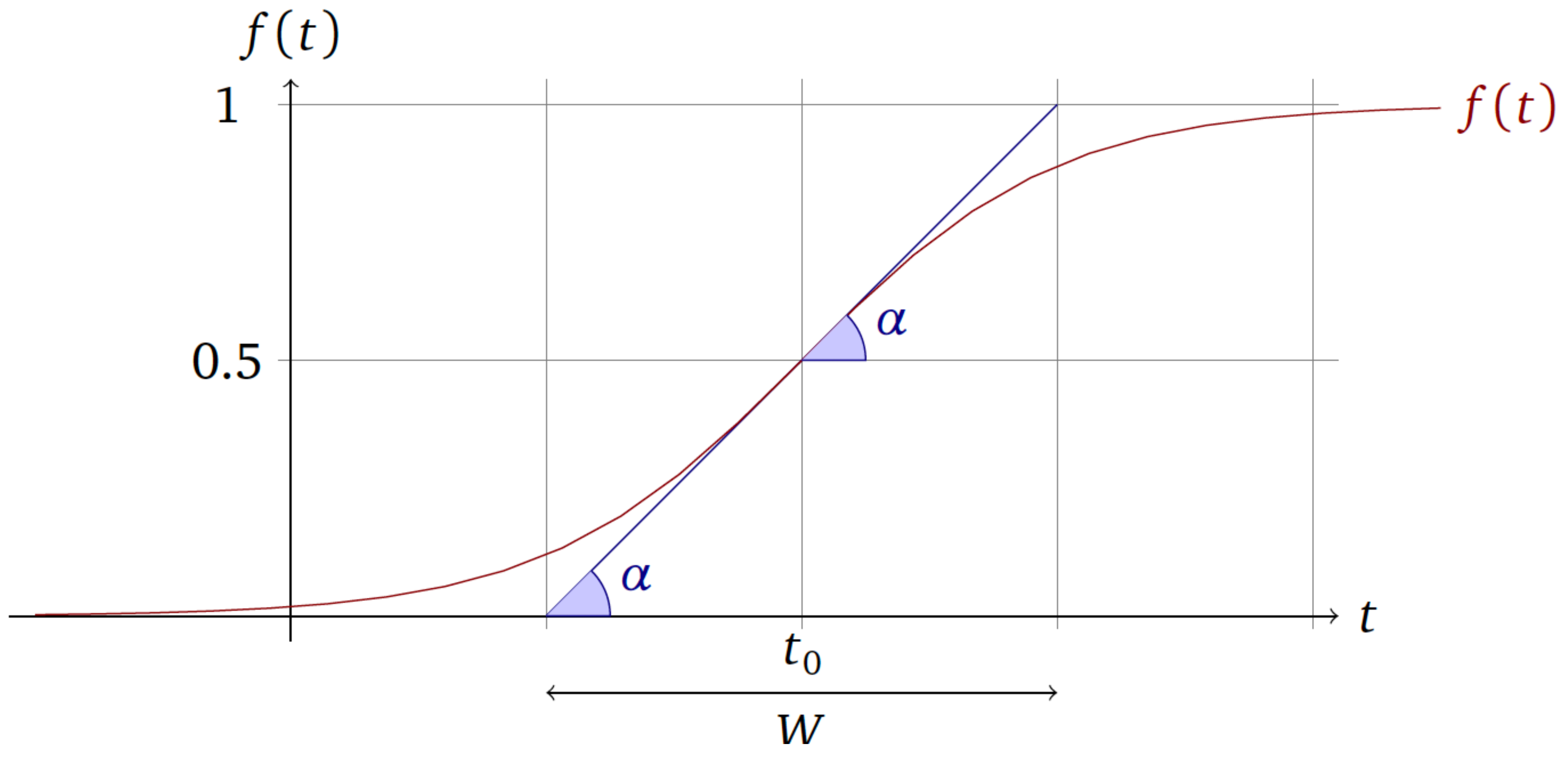
Concept drift is hard to define in a real data stream

Thus, studying it using real data can be challenging

One approach is to use synthetic data for studying and benchmarking algorithms

Concept Drift Framework

“Model a concept drift event as a **weighted combination of two pure distribution** that characterizes the target concepts before and after the drift.” [Bifet et al, 2011]



[Bifet et al, 2011] Bifet, A., & Kirkby, R. (2011). Data stream mining a practical approach. Chapter 2.7.1

Recursive definition

- Most tools (MOA[1], river[2], scikit-multiflow[3], …) uses a recursive approach to specify concept drift locations like:

CDS(**CDS**(**SEA**(1), **SEA**(2), 1000), **SEA**(3), 2000)

- Where we specify the drift **position** and the **width** of a drift (if it is a Gradual Drift) recursively.

- This can lead to some confusion depending on where the recursion is placed

[1] Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., & Seidl, T. (2010). Moa: Massive online analysis, a framework for stream classification and clustering. In *Workshop on applications of pattern analysis* (pp. 44-50). PMLR.

[2] Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T. and Bifet, A., 2021. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, *22*(110), pp.1-8.

[3] Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, *19*(72), 1-5.

Explicit list

In CapyMOA [1], the concepts and drifts are clearly outlined on a list format through a **DriftStream**

Drift position + drift width:

- The **start** and **end** of a **concept** is determined by the presence of an **AbruptDrift** or **GradualDrift** object

DriftStream([SEA(1), AbruptDrift(position=1000), SEA(2), GradualDrift(**position**=2000, **width**=500), SEA(3)**])**

The GradualDrift can also be specified in terms of start and end:

DriftStream([SEA(1), AbruptDrift(position=1000), SEA(2), GradualDrift(**start**=1750, **end**=2250), SEA(3)**])**

- This can make the drift locations more explicit and easy for new comers

[1] CapyMOA, https://capymoa.org

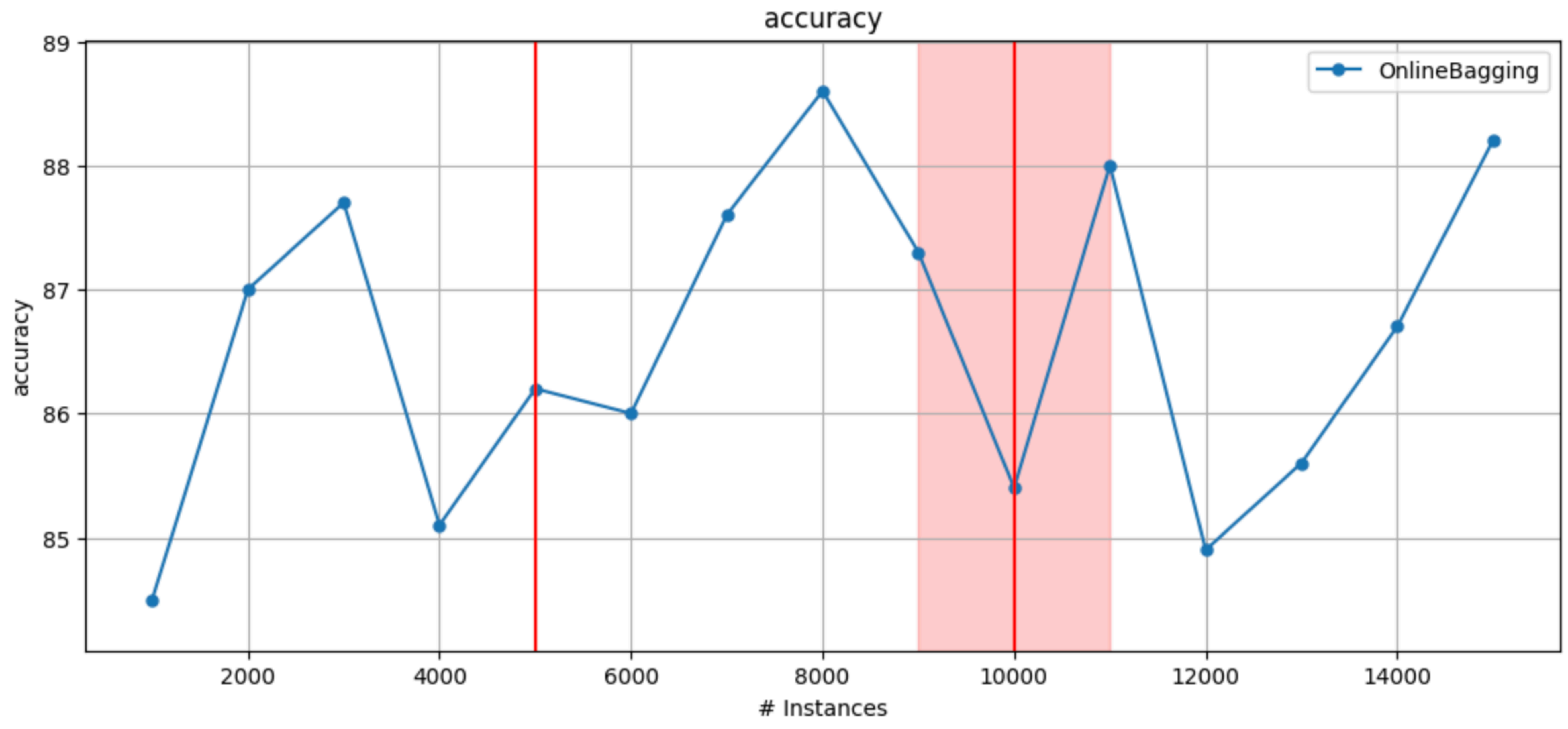
Code example

```
DriftStream(stream=[SEA(function=1),
                    AbruptDrift(position=5000),
                    SEA(function=3),
                    GradualDrift(position=10000, width=2000),
                    SEA(function=1)])
```

Both approaches will generate a similar output.

The second one use CapyMOA generic API to invoke the MOA CLI

```
Stream(moa_stream=ConceptDriftStream(),
                    CLI='-s
(ConceptDriftStream -s
generators.SEAGenerator -d
(generators.SEAGenerator -f 3) -p 5000 -w
1) -d generators.SEAGenerator -w 2000 -p
10000 -r 1')
```

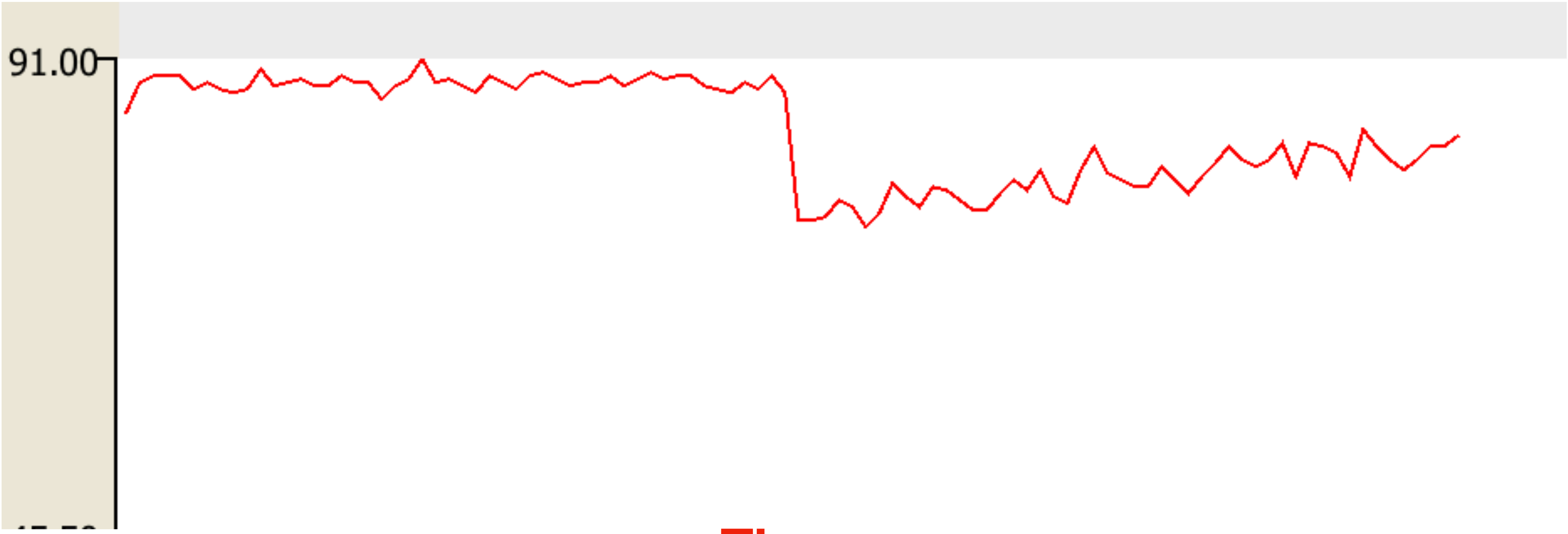


See more on the drift stream tutorial at https://capymoa.org/notebooks/04_drift_streams.html

Evaluation (detectors)

Common approach (proxy): **“Attach the method to a classifier, if the accuracy goes up, then the detector works”**

Accuracy



Time

Not necessarily the detector is successful in detecting changes, maybe it is just <u>randomly resetting the classifier</u>!

We must use specific metrics to evaluate a detector

Evaluation (detectors)

Important: we need the ground-truth of drift location for some of these

Some Metrics:

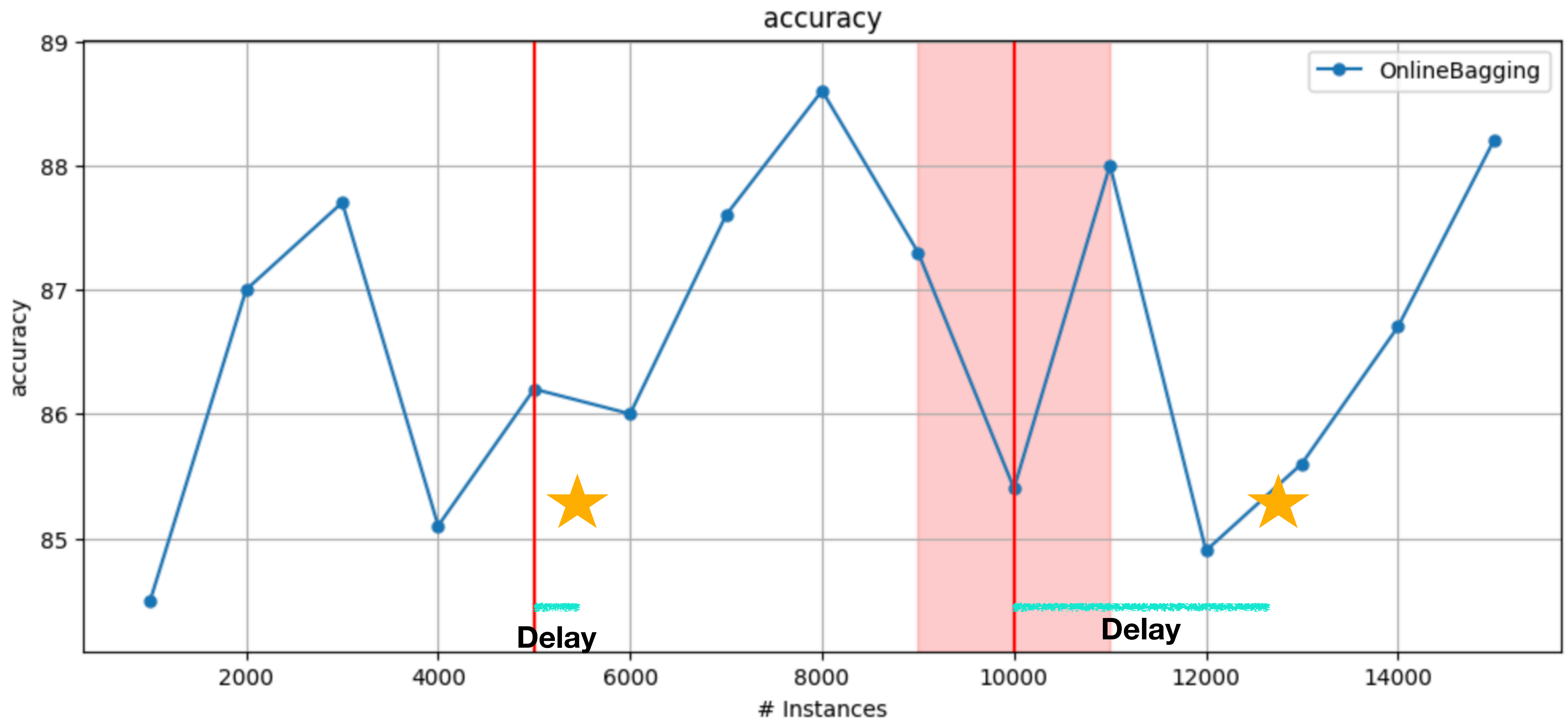
- Recall, Precision, …
- Number of detections
- Mean Time between False Alarms (MTFA)
- Mean Time to Detection (MTD)
- And others: MDR, ARL, MTR, …

Bifet, A. (2017). Classifier concept drift detection and the illusion of progress. In *Artificial Intelligence and Soft Computing ICAISC, 2017*

Evaluation (detectors)

We might want to only account for detections if they are within a **max_delay**

Let's assume in the example below that **yellow stars** are detections, we can observe some delay between the drifts (red vertical lines or rectangles (gradual)) and detections.



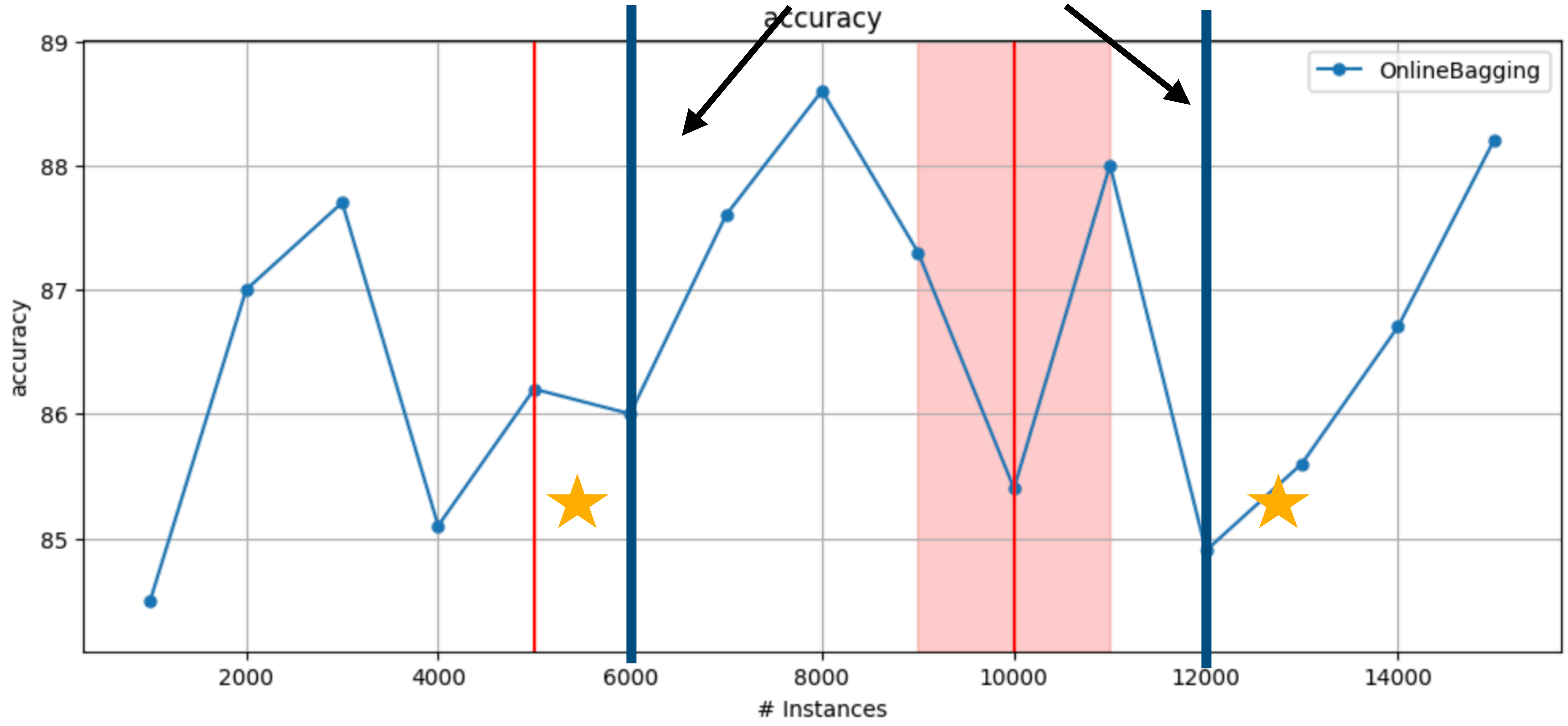
See more on *https://capymoa.org/notebooks/drift_detection.html*

Evaluation (detectors)

We might want to only account for detections if they are within a **max_delay**

Let's assume in the example below that **yellow stars** are detections, we can observe some delay between the drifts (red vertical lines or rectangles (gradual)) and detections.

We can also specify **max_delay** to determine when a detection should be considered a TP



See more on *https://capymoa.org/notebooks/drift_detection.html*

Using Drift Detectors

Ideally, using concept drift detectors should be straightforward

There should be some way to **update** it with new values, and some way of **detecting that a drift** has been detected

Bifet, A. (2017). Classifier concept drift detection and the illusion of progress. In *Artificial Intelligence and Soft Computing ICAISC, 2017*

Using Drift Detectors

```
# Create a learner nb, create a detector, declare the stream, …

while stream_sea1drift.has_more_instances() and i < max_instances:
    instance = stream_sea1drift.next_instance()
    pred = nb.predict(instance)
    evaluator.update(instance.y_index, pred)

    is_correct = int(pred == instance.y_index)

    detector.add_element(is_correct)

    if detector.detected_change():
        print('Change detected at instance: ' + str(i))

    nb.train(instance)
# …
```

See notebook **05_KDD2024_solutions.ipynb** at *https://adaptive-machine-learning.github.io/kdd2024_ml_for_streams/*

Recurrent Concept Drifts

Recurrent Concept Drifts

- by **concept transition**
- by **time of recurrence**

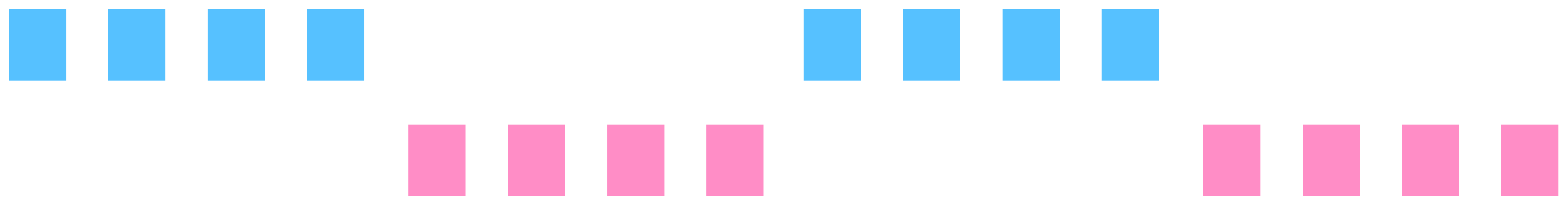
N Gunasekara, B Pfahringer, HM Gomes, A Bifet, Y S Koh. **Recurrent Concept Drifts on Data Streams.** International Joint Conferences on Artificial Intelligence (IJCAI), 2024

Recurrent Concept Drifts

- by **concept transition**

Recurrent Concept Drifts

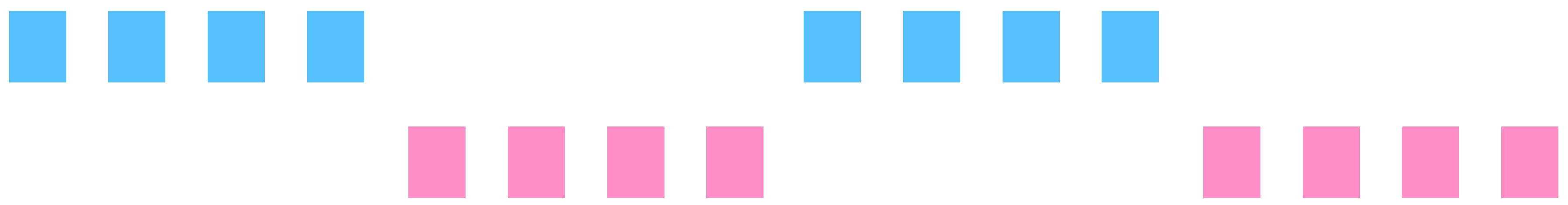
- by **concept transition**



a) abrupt recurrent drift

Recurrent Concept Drifts

- by **concept transition**

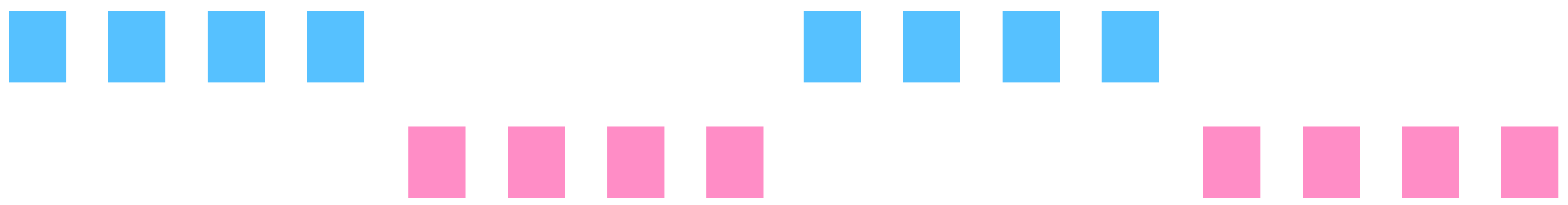


a) abrupt recurrent drift

b) incremental recurrent drift

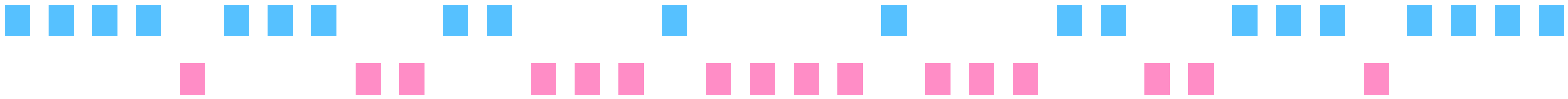
Recurrent Concept Drifts

- by **concept transition**



a) abrupt recurrent drift

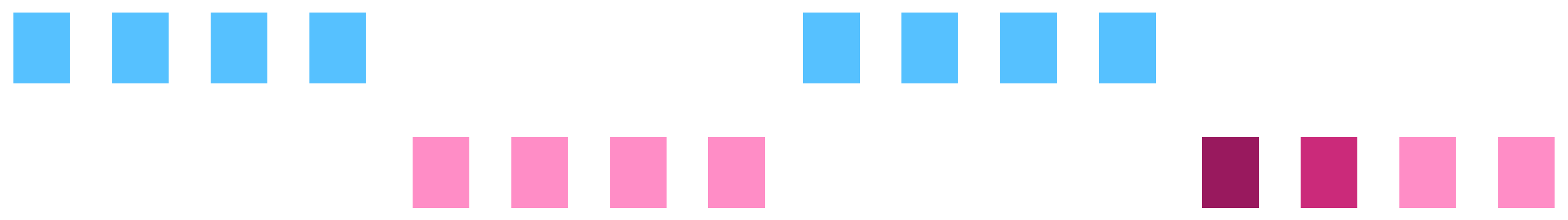
b) incremental recurrent drift



c) gradual recurrent drift

Recurrent Concept Drifts

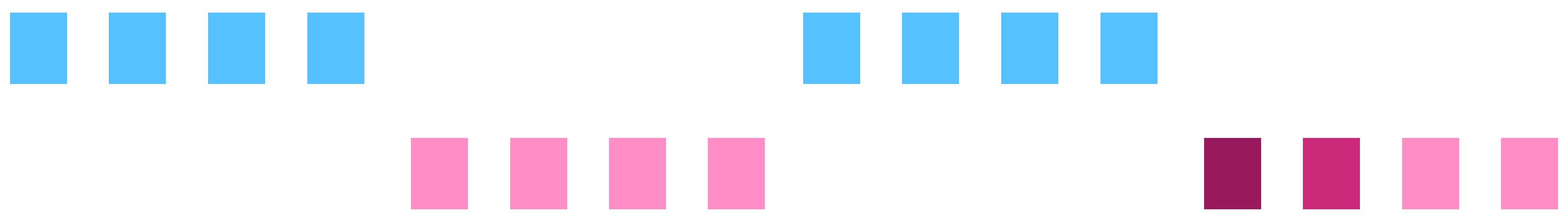
- by **concept transition** cont..



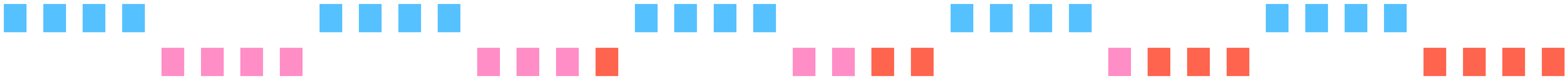
d) partial recurrent drift

Recurrent Concept Drifts

- by **concept transition** cont..



d) partial recurrent drift



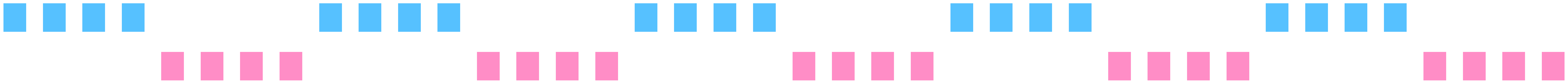
e) evolving recurrent drift

Recurrent Concept Drifts

- by **time of recurrence**

Recurrent Concept Drifts

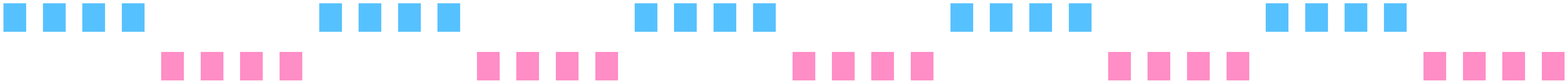
- by **time of recurrence**



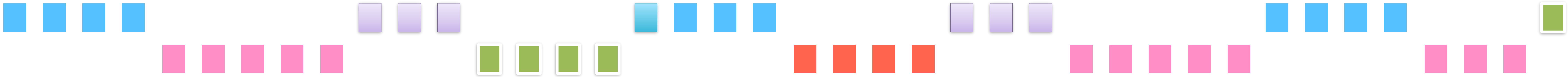
a) periodic recurrent drifts

Recurrent Concept Drifts

- by **time of recurrence**



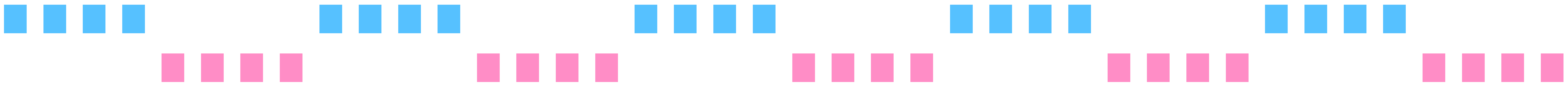
a) periodic recurrent drifts



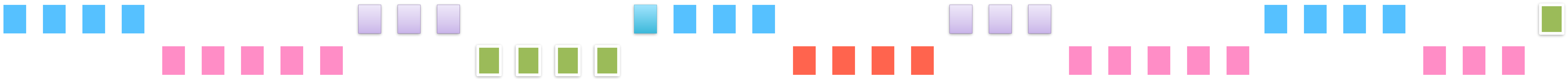
b) semi-periodic recurrent drifts

Recurrent Concept Drifts

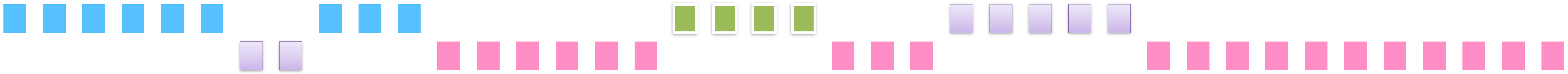
- by **time of recurrence**



a) periodic recurrent drifts



b) semi-periodic recurrent drifts



c) random recurrent drifts

Drift Re-Cap

- Many types of **drifts**
  - abrupt, incremental, gradual and recurrent
- Many types of **recurrent concept drifts**
  - by transition
  - by recurrence

Learners that cope with recurrent CD

- Ideally, these methods attempt to retain the knowledge acquired on a given concept and reuse it whenever that concept reappears.

- Simple example: Maintain a pool of learners, somehow identify if a concept is related to a given learner from the pool.

- Challenges:
  - identify the compatibility of two concepts
  - maintain the pool of learners

Methods

Design components:

- **DD**: Drift Detection
- **DP**: Drift Prediction
- **LM**: Labels Missing
- **LN/A**: Labels Not Available
- **MetaL**: Meta Learning
- **MetaF**: Meta Features
- **Clust**: Clustering
- **CEqSim**: Conceptual Equivalence/Concept Similarity
- **Ens**: Ensemble
- **CPool**: Concept Pool.

Please refer to section 3 of the survey for more information

N Gunasekara, B Pfahringer, HM Gomes, A Bifet, Y S Koh. **Recurrent Concept Drifts on Data Streams.** International Joint Conferences on Artificial Intelligence (IJCAI), 2024

Methods

	Sec	Method	Year	DD	DP	LM	LN/A	MetaL	MetaF	Clust	CEqSim	Ens	CPool
	3.1	LEARN++*	2011									X	X
Explicit Handling of Recurrences (model for each data batch)		PMRCD	2012									X	X
		Dynse	2018									X	X
		ASE	2017								X	X	
		GraphPool	2018								X	X	X
	3.2	RCD	2013	X				X			X	X	X
Meta Learning (act as a wrapper algorithm to determine the best model/s for the current concept)		CPF	2016	X				X			X		X
		ECPF	2019	X				X			X		X
		PEARL	2022	X				X				X	X
	3.3	REDLLA	2012	X		X				X	X		X
		SUN	2012	X		X				X	X		X
		ContexTrac	2012	X						X	X		X
Clustering		ESCR	2021	X		X				X	X	X	
		CDMSE	2021	X		X				X		X	
		CCP	2010						X	X		X	
		UClust	2020	X		X	X		X	X	X		X
		CDCMS	2020	X						X		X	
	3.4	MM-PRec	2016	X	X			X			X		X
		PCCF	2016		X								
Drift Prediction		BLPA	2017	X	X								
		CPRD	2019	X	X								
		ProSeed	2016	X	X								
		ProChange	2018	X	X	X							
		MDP	2018	X	X				X	X			
		Nacre	2021	X	X							X	
Meta Features	3.5	SELeCT	2022						X				X
		FiCSUM	2023	X					X				

Design components: **DD**: Drift Detection, **DP**: Drift Prediction, **LM**: Labels Missing, **LN/A**: Labels Not Available, **MetaL**: Meta Learning, **MetaF**: Meta Features, **Clust**: Clustering, **CEqSim**: Conceptual Equivalence/Concept Similarity, **Ens**: Ensemble, **CPool**: Concept Pool.

Please refer to section 3 of the survey for more information

N Gunasekara, B Pfahringer, HM Gomes, A Bifet, Y S Koh. **Recurrent Concept Drifts on Data Streams.** International Joint Conferences on Artificial Intelligence (IJCAI), 2024

Evaluation

- Under recurrent concept drift
  - Model performance
  - Drift Detection performance

Evaluation

Relative Performance

- **compares** the performance of classifier A **against a baseline classifier** B.
  - **at instance** i: $log(B_{error_i}/A_{error_i})$ [1]
  - **Cumulative Accuracy Gain**: $\sum (accuracy(A) - accuracy(B))$ [2]

[1] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. Machine learning, pages 317–346, 2013.

[2] Ocean Wu, Yun Sing Koh, Gillian Dobbie, and T Lacombe. Probabilistic exact adaptive random forest for recurrent concepts in data streams. Int. J. Data Sci. Anal., pages 1–16, 2022.

Evaluation

Model Selection for Each Concept

- measures the strength of the relationship between each $< model, context >$ pair [1].
  - **context**: an underlying condition that results in a concept
  - **measures** the **context linkage** for model reuse (from a model pool)

[1] Ben Halstead, Yun Sing Koh, P Rid- dle, R Pears, M Pechenizkiy, and Albert Bifet. Recurring concept memory management in data streams: exploiting data stream concept evolution to improve performance and transparency. DM and KD, pages 796–836, 2021

Evaluation

Drift Detection on **Synthetic Data** (drift points known in advance)

- **True drift** points are **compared** to the **drift detected points** to detect **Type I** (FP) and **Type II** (FN) errors. [1-4]

[1] Robert Anderson, Yun Sing Koh, and Gillian Dobbie. Predicting concept drift in data streams using metadata clustering. In IJCNN, pages 1–8. IEEE, 2018.

[2] Alexandr Maslov, Mykola Pechenizkiy, Indre Žliobaite, and Tommi Kärkkäinen. Modelling recurrent events for improving online change detection. In SDM, pages 549–557. SIAM, 2016.

[3] David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, and Russel Pears. Detecting volatility shift in data streams. In ICDM, pages 863–868. IEEE, 2014.

[4] Yun Sing Koh, David Tse Jung Huang, Chris Pearce, and Gillian Dobbie. Volatility drift prediction for transactional data streams. In ICDM, pages 1091– 1096. IEEE, 2018.

Open Source Software & Benchmark Datasets

- Most methods have **custom** open source implementations
- **Traditional streaming datasets**
  - Real world: may not know the reoccurrence
  - Synthetic : reproducibility
- **CapyMOA recurrent concept API**

Please refer to: Section 5 & 6 of the survey for more information

N Gunasekara, B Pfahringer, HM Gomes, A Bifet, Y S Koh. **Recurrent Concept Drifts on Data Streams.** International Joint Conferences on Artificial Intelligence (IJCAI), 2024

Delayed & Unsupervised Drift Detection

Delayed & Unsupervised Drift Detection

- Most concept drift detection algorithms are applied to the **univariate stream of correct/incorrect** classifier predictions

- Such strategies require that labeled data is available as soon as possible to respond to concept drifts in a timely fashion

- Despite their intrinsic differences, most drift detectors **trigger when** the observed **model's predictive performance starts to degrade**

Gomes, H.M., Grzenda, M., Mello, R., Read, J., Le Nguyen, M.H. and Bifet, A., 2022. A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Computing Surveys*, *55*(4), pp.1-42.

Delayed & Unsupervised Drift Detection

Terminology

Delayed drift detection: The label will arrive at some point in the future, and it will be used for feeding the learner with a **delayed** univariate stream of correct/incorrect predictions

Unsupervised drift detection: The label **will not arrive**, thus the detection should be based on the input data or the output of the learner itself

Žliobaite, Indre. "Change with delayed labeling: When is it detectable?." In *2010 IEEE international conference on data mining workshops*, pp. 843-850. IEEE, 2010.

Delayed Drift Detection (Example)

- Experiment with data generated using the AGRAWAL generator with **3 abrupt concept drifts** (at instances 25, 000, 50, 000, and 75, 000).
- Used an ensemble algorithm capable of detecting and adapting to changes by resetting base models whenever **changes** are **detected** on their **univariate stream of correct/incorrect** predictions.
- Figure depicts the amount of concept drifts detected (y-axis) over the processing of 100,000 instances with and without delayed labelling.
- The detections for the “No delay” experiment shows a high rate of detection immediately after the concept drifts, except for a few arbitrary drift signals in-between the concept drifts.

Delayed Drift Detection (Example)

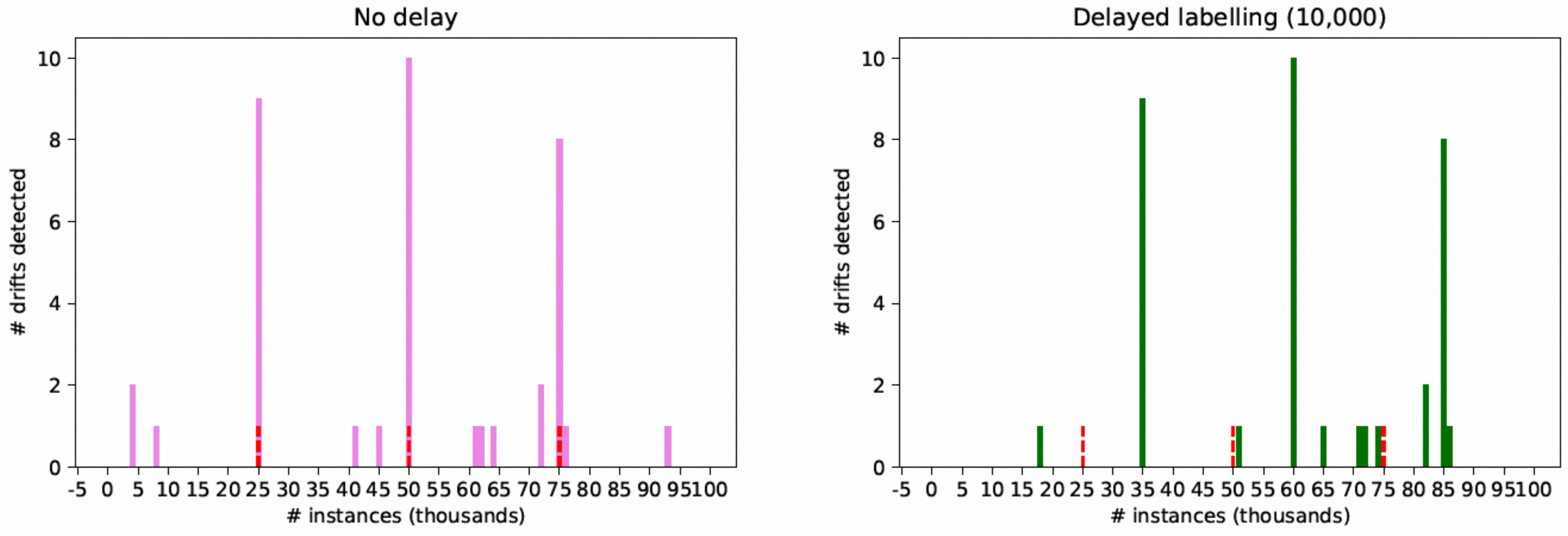


Fig. 7. Drifts detected by a 10 learner SRP model using ADWIN on AGRAWAL with and without labelling delay. Red dotted vertical lines indicate the location of concept drifts.

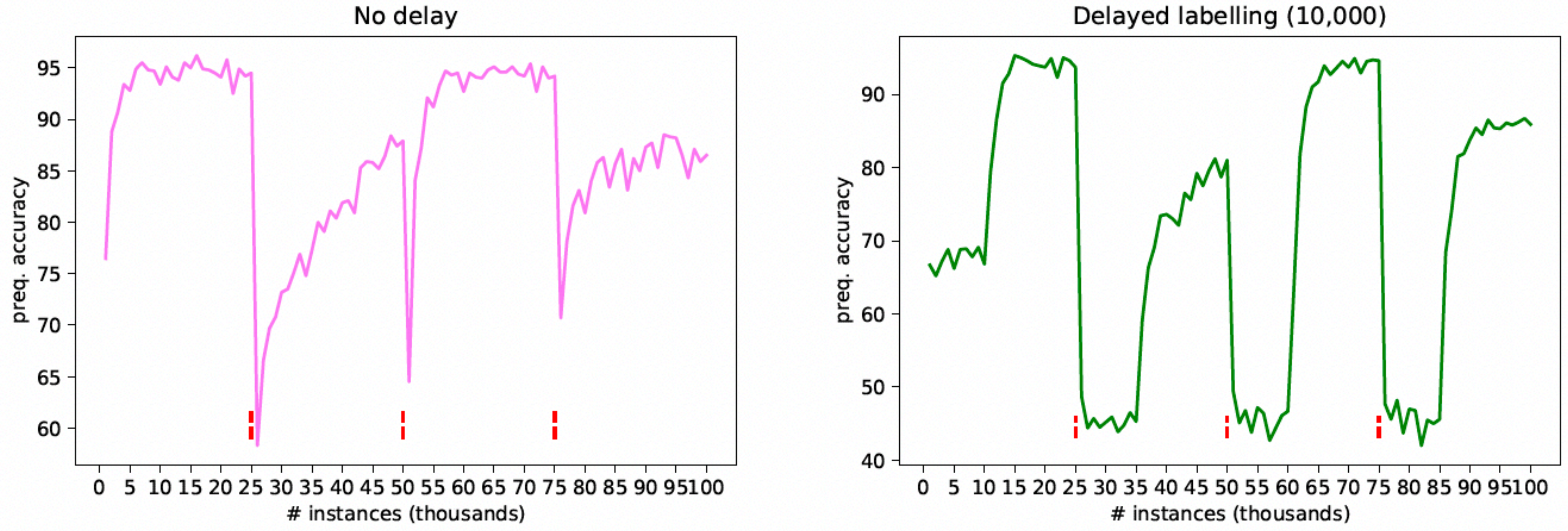


Fig. 8. Accuracy by a 10 learner SRP model using ADWIN on AGRAWAL with and without labelling delay. Red dotted vertical lines indicate the location of concept drifts.

Gomes, H.M., Grzenda, M., Mello, R., Read, J., Le Nguyen, M.H. and Bifet, A., 2022. A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Computing Surveys*, *55*(4), pp.1-42.

STUDD: Unsupervised Concept Drift Detection using a Student–Teacher Approach

- Detecting concepts drifts in the absence of labeled data

- **Procedure.** A predictive model (teacher) is built using an initial batch of labelled training data. The teacher's predictions are used as class labels to train a surrogate model (student), which will learn to mimic the teacher. A drift detection algorithm is used to identify variations in the mimicking error of the student.

- **Hypothesis.** If the mimicking error increases, then it means that a concept drift has occurred.

Cerqueira, V., Gomes, H. M., Bifet, A., & Torgo, L. (2023). STUDD: A student–teacher method for unsupervised concept drift detection. Machine Learning, Springer

Conclusions

Conclusions

- Practical aspects w.r.t. CD: simulate, evaluate, utilise
- Opportunities in identifying drifts on partially and delayed labeled settings
- Opportunities w.r.t. recurrent drifts in the intersection with Online Continual Learning
  - **Datasets** (OCL **->** Recurrent SL)
  - Model **pool management** techniques (OCL **<-** Recurrent SL)

Thank you!

- Consider trying **CapyMOA** for your drift detection needs!

Contact: heitor.gomes@vuw.ac.nz

Discord

https://discord.gg/RekJArWKNZ

https://github.com/adaptive-machine-learning/CapyMOA