Navigating Complex Machine Learning Challenges in Streaming Data

ECML Tutorial 2024

Heitor Murilo Gomes[1*], Marco Heyden[2]
Maroua Bahri[3,4]
https://heymarco.github.io/ecml24-streamingchallenges/

* Corresponding author: heitor.gomes@vuw.ac.nz

[1] Victoria University of Wellington, New Zealand, [2] KIT, Germany, [3] INRIA Paris, France,
[4] Sorbonne Université, France

https://capymoa.org/

Classification algorithms

Hoeffding Tree*

* Also known as Very Fast Decision Tree (VFDT)

Goal: Grow a decision tree incrementally

This means that after every new training instance, the tree may grow

Key question: When should a split happen?

Hypothesis: A small sample is often enough to choose a near optimal split decision

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *ACM SIGKDD*.

Hoeffding Bound

It is a statistical inequality that provides a theoretical guarantee on the convergence of sample averages to the true mean with a high probability

In other words, the **Hoeffding Bound** helps in determining whether **the observed differences in the attributes' merit (purity) are statistically significant** or merely due to random variation

Wassily Hoeffding (1963) Probability Inequalities for Sums of Bounded Random Variables, Journal of the American Statistical Association, 58:301, 13-30, DOI: 10.1080/01621459.1963.10500830

Hoeffding Bound

When should we split a node?

Let X_1 and X_2 be the top 2 most informative attributes*

Is X_1 a stable option?

Hoeffding bound, split on X_1 if $G(X_1) - G(X_2) > \epsilon$

Where $G(*)$ is a purity measure
(e.g. Gini index, Information gain)

* The top attributes to split, the ones that will cause the splits to be “purer”

Hoeffding Bound

When should we split a node?

Let X_1 and X_2 be the top 2 most informative attributes*

Is X_1 a stable option?

Hoeffding bound, split on X_1 if $G(X_1) - G(X_2) > \epsilon$

Where $G(*)$ is a purity measure
(e.g. Gini index, Information gain)

$$\epsilon = \sqrt{\frac{R^2 \ln 1/\delta}{2n}}$$

R = Range of observed random variable

δ = The desired probability of the estimate not being within ϵ of its expected value

n = Number of observed instances

* The top attributes to split, the ones that will cause the splits to be “purer”

Hoeffding Tree

wrap-up

- ϵ decreases with n (or the more instances observed)

- HT builds a tree that converges to the tree built by a batch learner given sufficiently large data

- A *grace period* can be used to avoid "splitting too fast"

- There are better options w.r.t. theoretical guarantees (See McDiarmid Trees*), but HTs still works well in practice

* Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2012). Decision trees for mining data streams based on the McDiarmid's bound. *IEEE Transactions on Knowledge and Data Engineering*.

Other Streaming Decision Tree algorithms

- The **Extremely Fast Decision Tree (EFDT) [1]** algorithm improves upon the Hoeffding Tree by using a more relaxed criterion for splitting nodes, allowing it to grow the tree more quickly.
    - EFDT can revisit and revise earlier splits if better splits are found later, which can lead to subtree pruning and potentially sudden drops in accuracy.
- **PLASTIC [2]** avoids EFDT's subtree pruning by restructuring the tree rather than pruning it when revising splits.

Streaming Data session, 11:00am, Thursday

    - This allows PLASTIC to maintain accuracy by rearranging the tree structure without discarding information

[1] Manapragada, Chaitanya, Geoffrey I. Webb, and Mahsa Salehi. Extremely fast decision tree. ACM SIGKDD International, 2018
[2] M. Heyden, H. M. Gomes, E. Fouché, B. Pfahringer, and K. Bohm. Leveraging Plasticity in Incremental Decision Trees. ECML-PKDD, [To Appear] 2024

Bagging

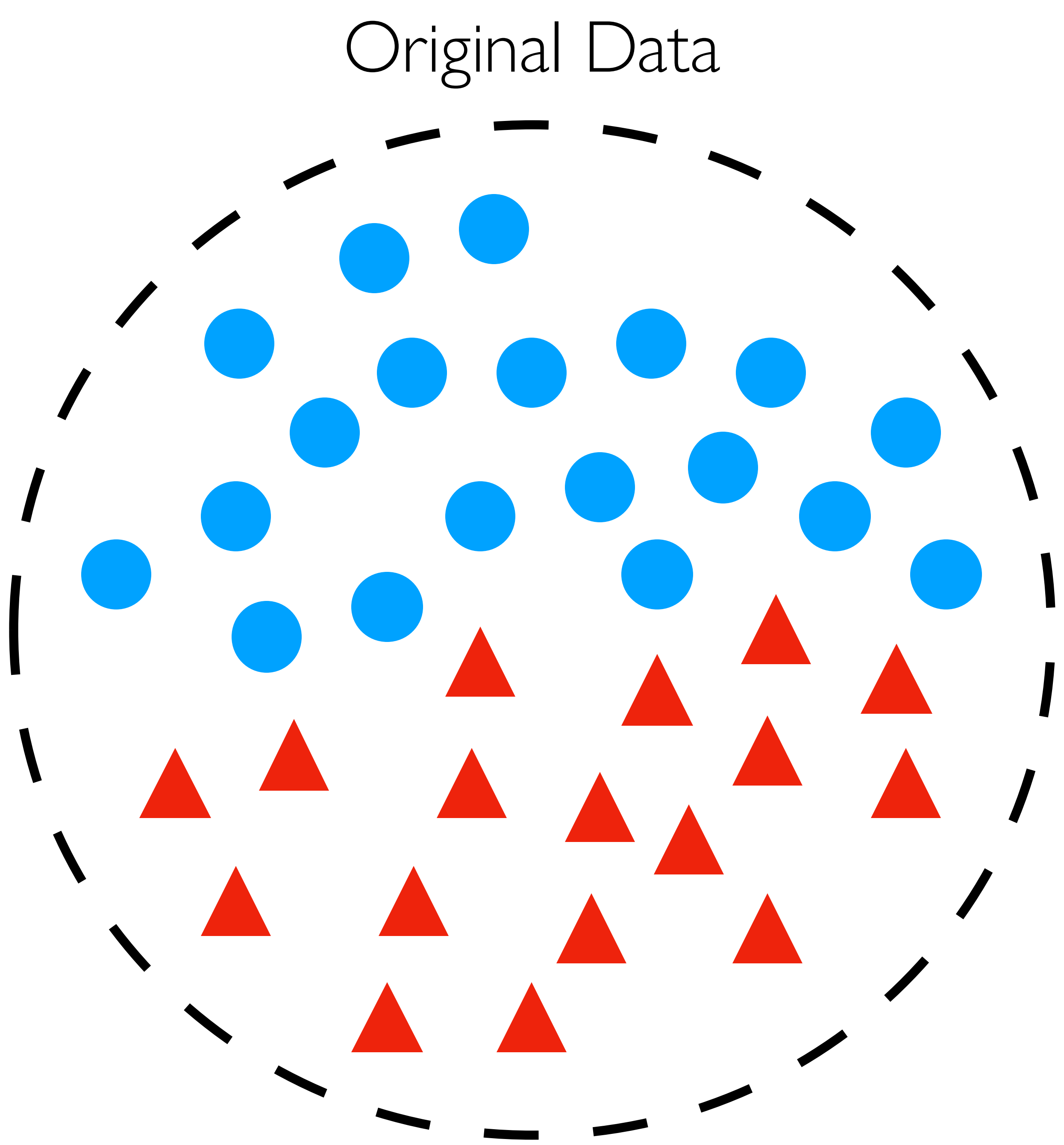
Bootstrap **Agg**regat**ing**

Bagging trains each model of the ensemble with a bootstrap sample from the original dataset.

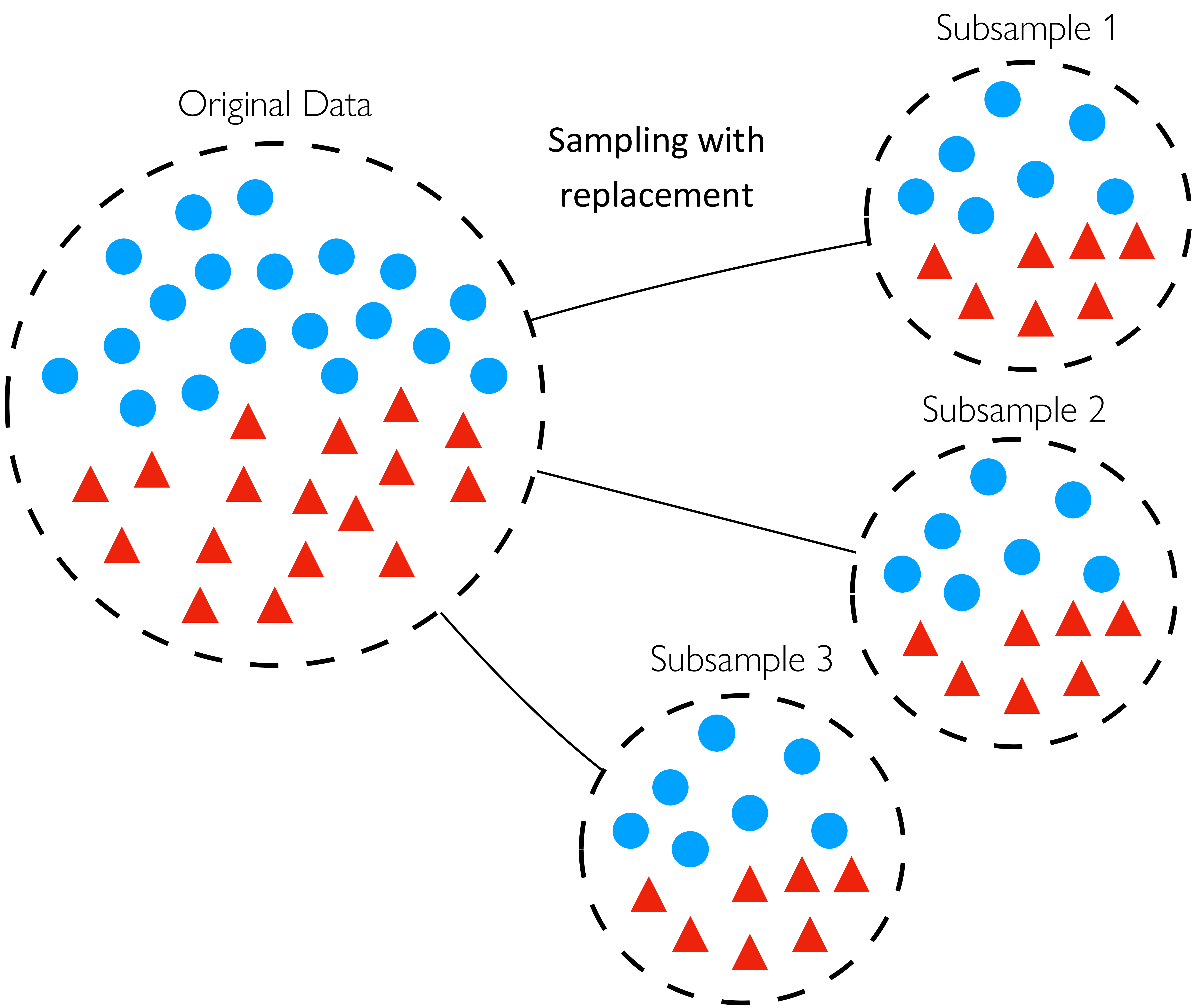
Every bootstrap contains each original sample **K** times, where **Pr(K=k)** follows a binomial distribution.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

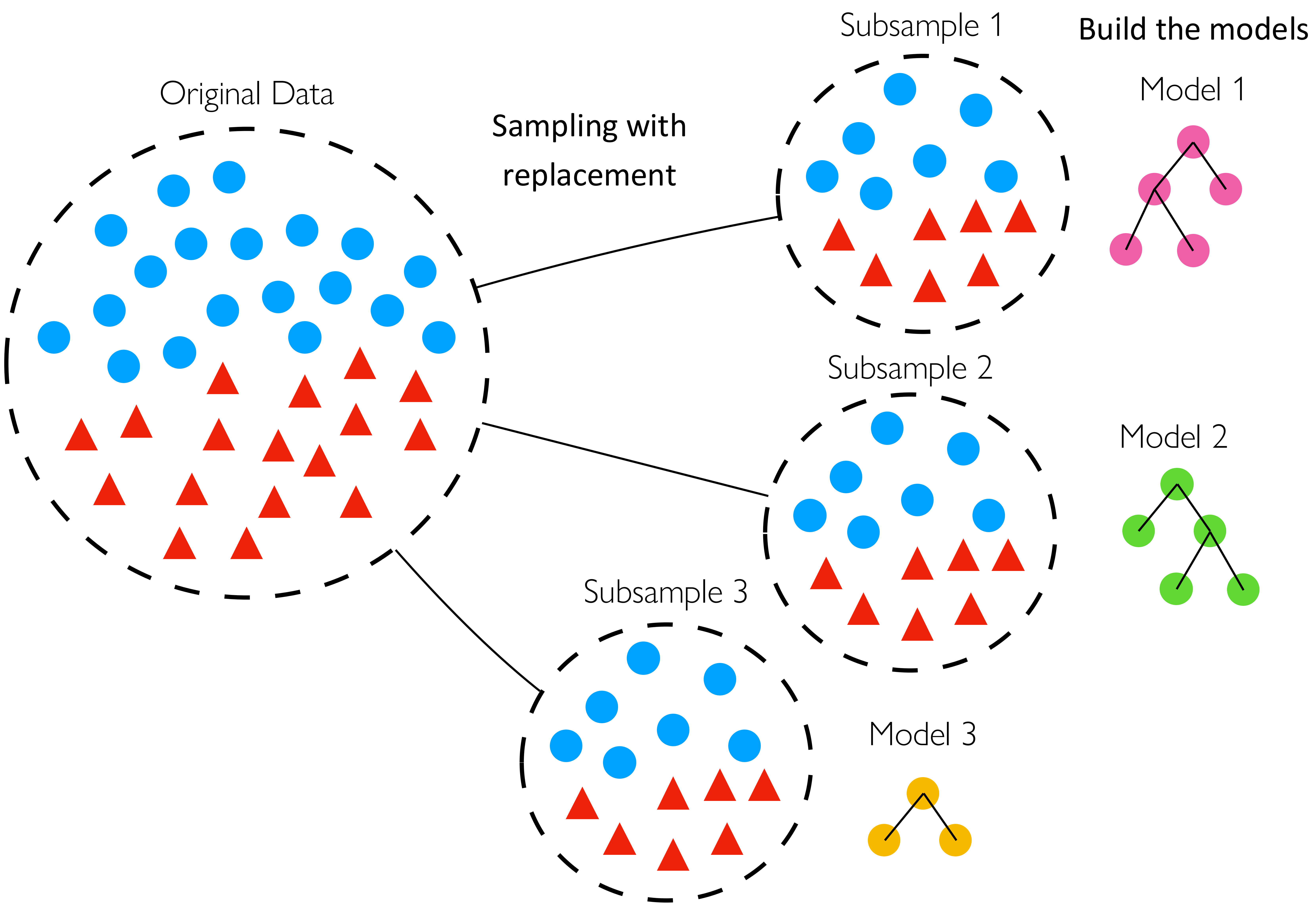
Bagging



Bagging


Original Data
Sampling with replacement
Subsample 1
Subsample 2
Subsample 3

Bagging


Original Data
Sampling with
replacement
Subsample 1
Subsample 2
Subsample 3
Build the models
Model 1
Model 2
Model 3

Bagging

On average for each subsample:

~64% of the instances are from the original dataset

~37% are repeated instances

~37% of the original instances are not present*

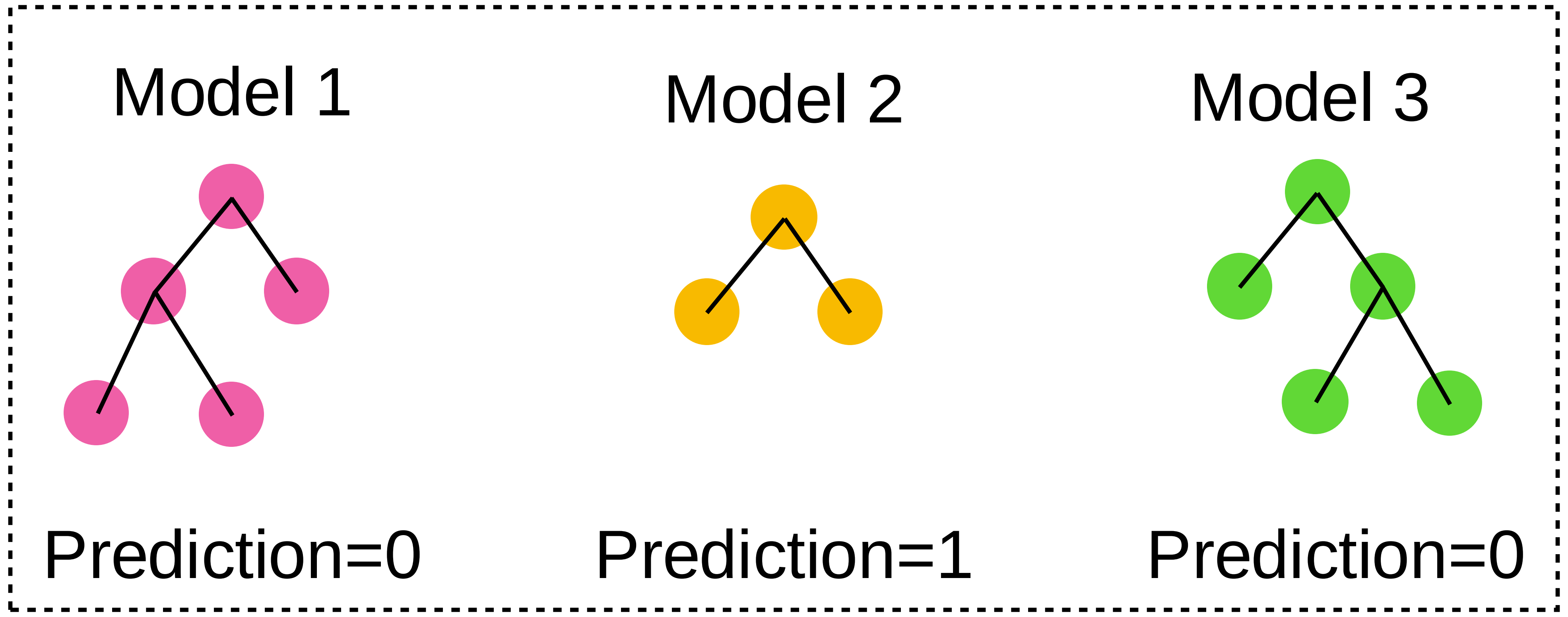
* Out-Of-Bag (OOB)

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Bagging

The **predictions** of each learner are **aggregated** using majority vote to obtain the final prediction.

Prediction for a given instance X…



Ensemble

Prediction=0

Online Bagging

- We cannot apply Bagging directly to data streams…
- Unfeasible to store all data before creating each bootstrap subsample

<u>We need to build the subsamples online</u>

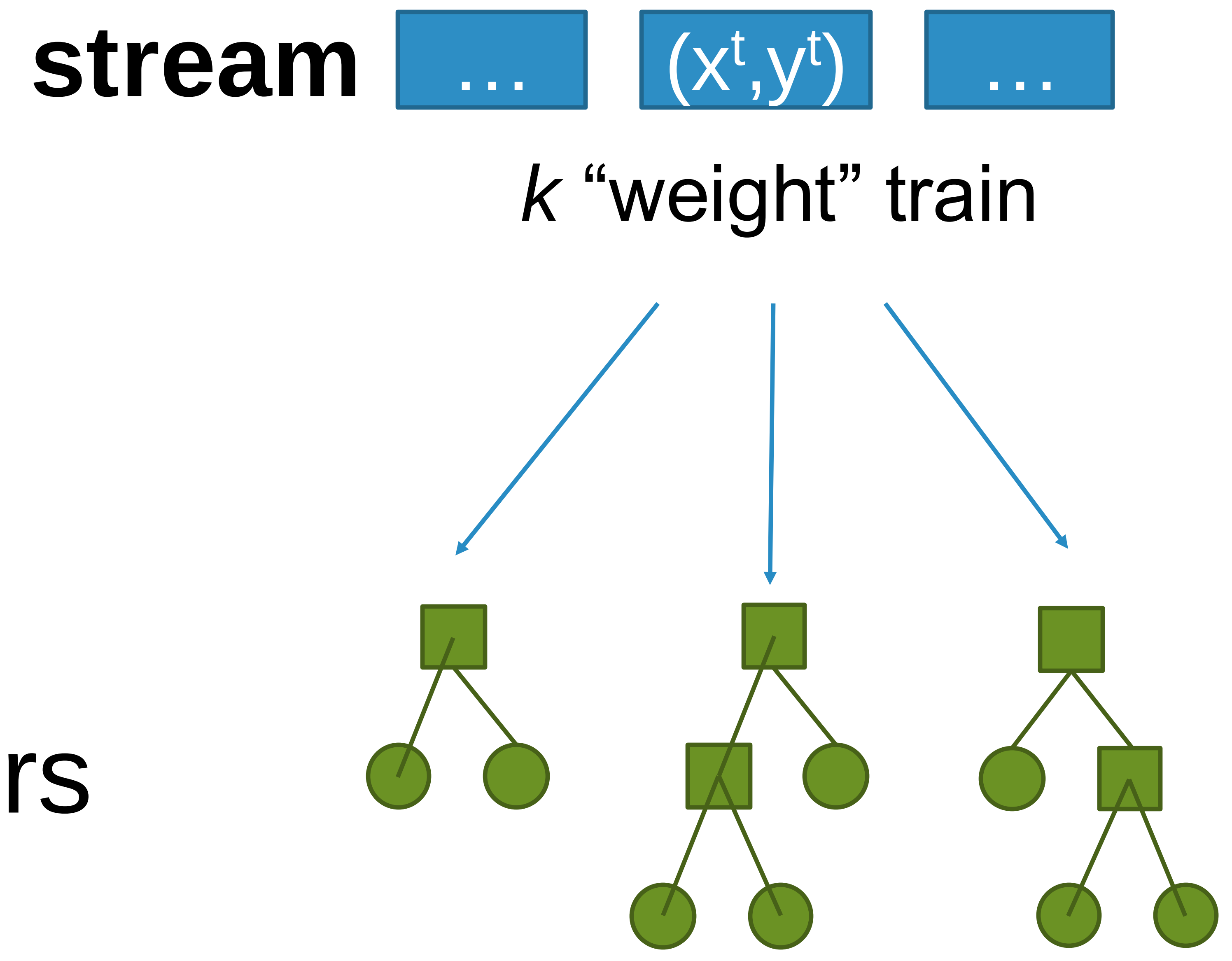
N. Oza and S. Russel "Online bagging and boosting" Artificial Intelligence and Statistics, 2001

Online Bagging

- Given a dataset with **N** samples
- In Bagging, every bootstrap contains each original sample **K** times, where **Pr(K=k)** follows a binomial distribution
- Oza and Russel found out that for large **N**, the binomial distribution tends to a **Poisson(1)** distribution
- Online Bagging instead of sampling with replacement, gives each example a weight according to **Poisson(1)** distribution

N. Oza and S. Russel "Online bagging and boosting" Artificial Intelligence and Statistics, 2001

Online Bagging



$$k \leftarrow Poisson(\lambda=1)$$
$$\textbf{if } k > 0 \textbf{ then}$$
$$\quad l \leftarrow FindLeaf(t, x)$$
$$\quad UpdateLeafCounts(l, x, k)$$

Practical effect: train learners with different subsets of instances.

N. Oza and S. Russel "Online bagging and boosting" Artificial Intelligence and Statistics, 2001

Subsamples

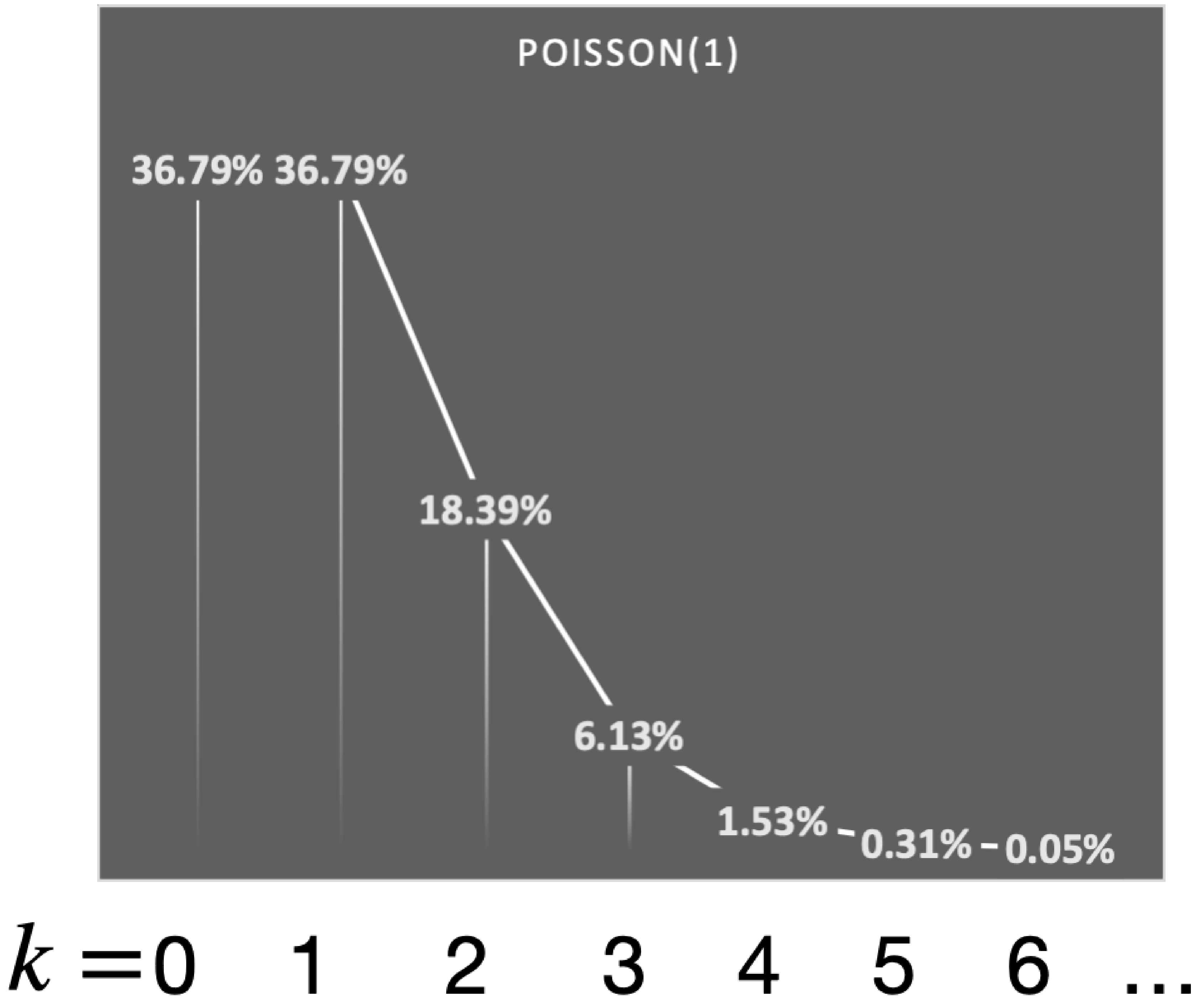
Batch bagging

~64% from the original dataset

~37% are repeated

~37% are not present

Online bagging



Adaptive Random Forest (ARF)

Streaming version of the original Random Forest by Breiman

Uses a variation of the Hoeffding Tree

Main differences:

Online bagging, base learner & detectors

Overview:

1. Online bagging
2. Random subset of features
3. Drift detector for each tree

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *ACM SIGKDD*.

Breiman, L. (2001). Random forests. *Machine learning.*

Gomes, H. M., Bifet, A., Read, J., …, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning.*

ARF: Detect and Adapt

- One **Warning** and one **Drift** detector per base model
- Relies on the **Adaptive WINdow** (ADWIN) algorithm for detection (other algorithms could be used)
- ***Background*** **learners** are started once a warning is detected, their subspace of features may not correspond to the subspace of features used by the *“foreground”* learner.
- Once a drift is detected, the ***background*** **learner replaces** the ***“foreground”*** **learner**.

Boosting and Gradient Boosting

- XGBoost and CatBoost are popular **batch gradient boosting** methods
- One key challenge when adapting such algorithms other than a stream setting includes <u>concept drift recovery</u>

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD. ACM, 2016.
Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. NeurIPS.

Boosting on Streams

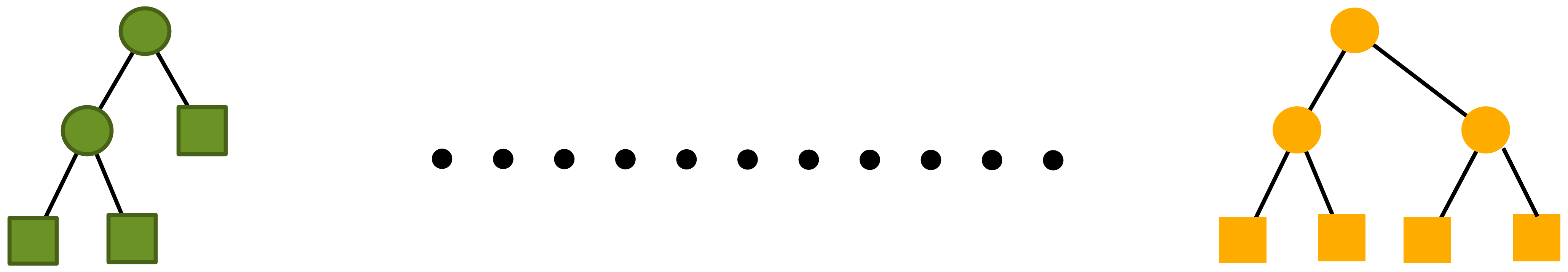
- [2001] **OzaBoost** [1] uses weights from a Poisson(1) distribution to train multiple times using a given instance (similar to Online Bagging)
- [2012] **Online Smooth Boost** [2] is analogous to batch SmoothBoost, thus it uses a smooth distribution for weight assignment
- [2020] **Gradient boosted AXGB** [3] use Mini-batch trained XGBoost as its base learners and adjusts the booster when concept drifts are detected by ADWIN
- [2024] **Streaming Gradient Boosted Trees (SGBT)** [4] **performs better than** bagging-based stream learners

[1] N. Oza and S. Russel "Online bagging and boosting" Artificial Intelligence and Statistics, 2001
[2] Chen, Shang-Tse, Hsuan-Tien Lin, and Chi-Jen Lu. "An online boosting algorithm with theoretical justifications." International Conference on International Conference on Machine Learning. 2012.
[3] Montiel, J., Mitchell, R., Frank, E., Pfahringer, B., Abdessalem, T., Bifet, A.: Adaptive xgboost for evolving data streams. In: 2020 IJCNN
[4] Gunasekara, N., Pfahringer, B., Gomes, H., & Bifet, A. (2024). Gradient boosted trees for evolving data streams. Machine Learning, 113(5), 3325-3352.

Regression algorithms

Adaptive Random Forest Regression

- Similar to ARF for classification
- builds **regression trees**
- for **prediction**, uses **mean** of **predictions** (by each tree)



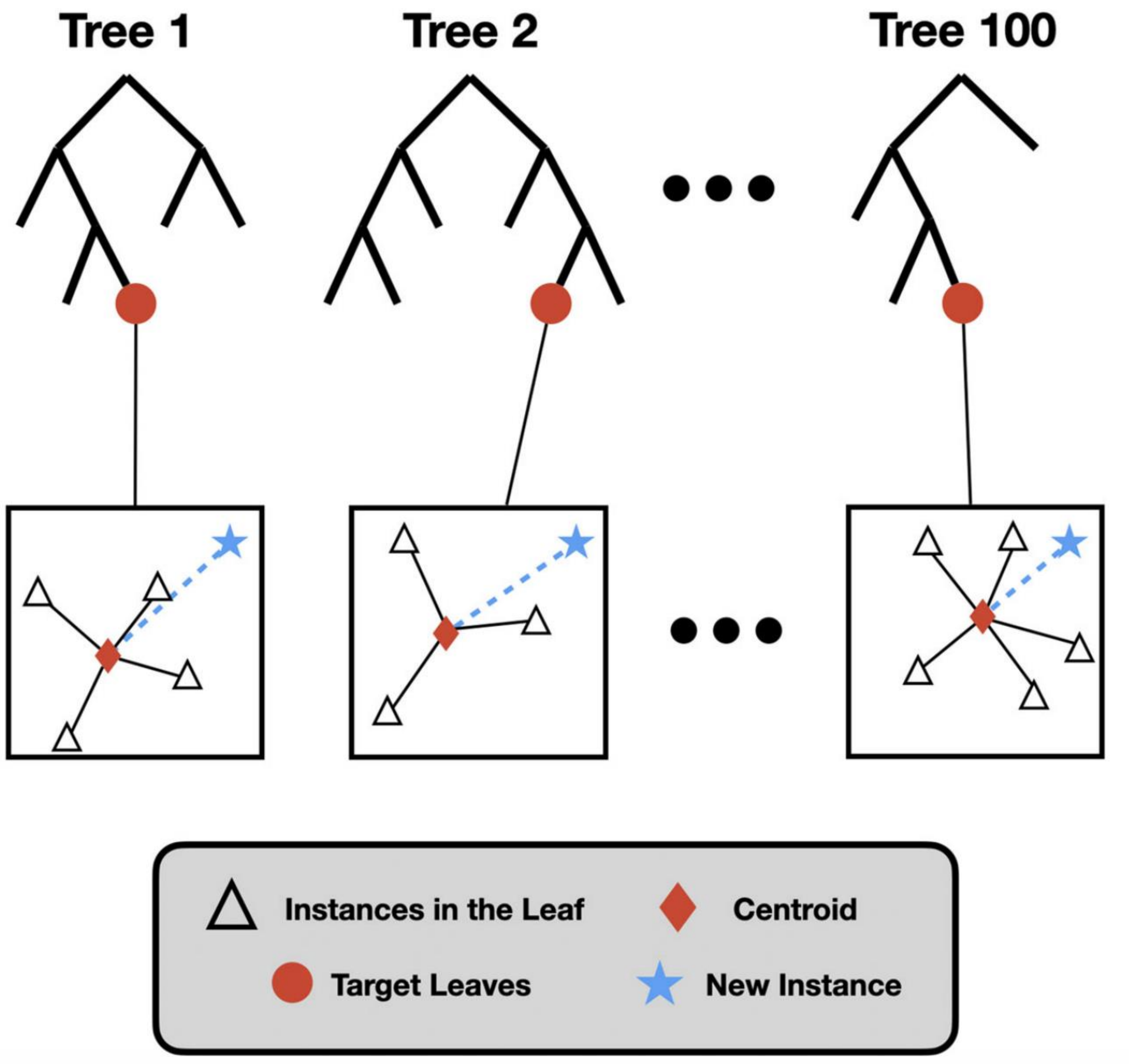
Gomes HM, Barddal JP, Ferreira LEB, Bifet A (2018) Adaptive random forests for data stream regression. ESANN

Self-Optimizing k-Nearest Leaves (SOKNL)

- **Extends** Adaptive Random Forest Regression
- Generates **a representative data point (centroid)** in each leaf by **compressing** information **from all instances in that leaf**
- During **prediction**, calculates **distances** between **input instance** and **centroids** for **relevant leaves**
- Uses **only k leaves** with **smallest distances** for **prediction**
- **Dynamically tuning k** values based on **historical information**

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "SOKNL: A novel way of integrating K-nearest neighbours with adaptive random forest regression for data streams"
Data Mining and Knowledge Discovery (2022)

Self-Optimizing k-Nearest Leaves (SOKNL)



Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "SOKNL: A novel way of integrating K-nearest neighbours with adaptive random forest regression for data streams" Data Mining and Knowledge Discovery (2022)

Prediction Intervals

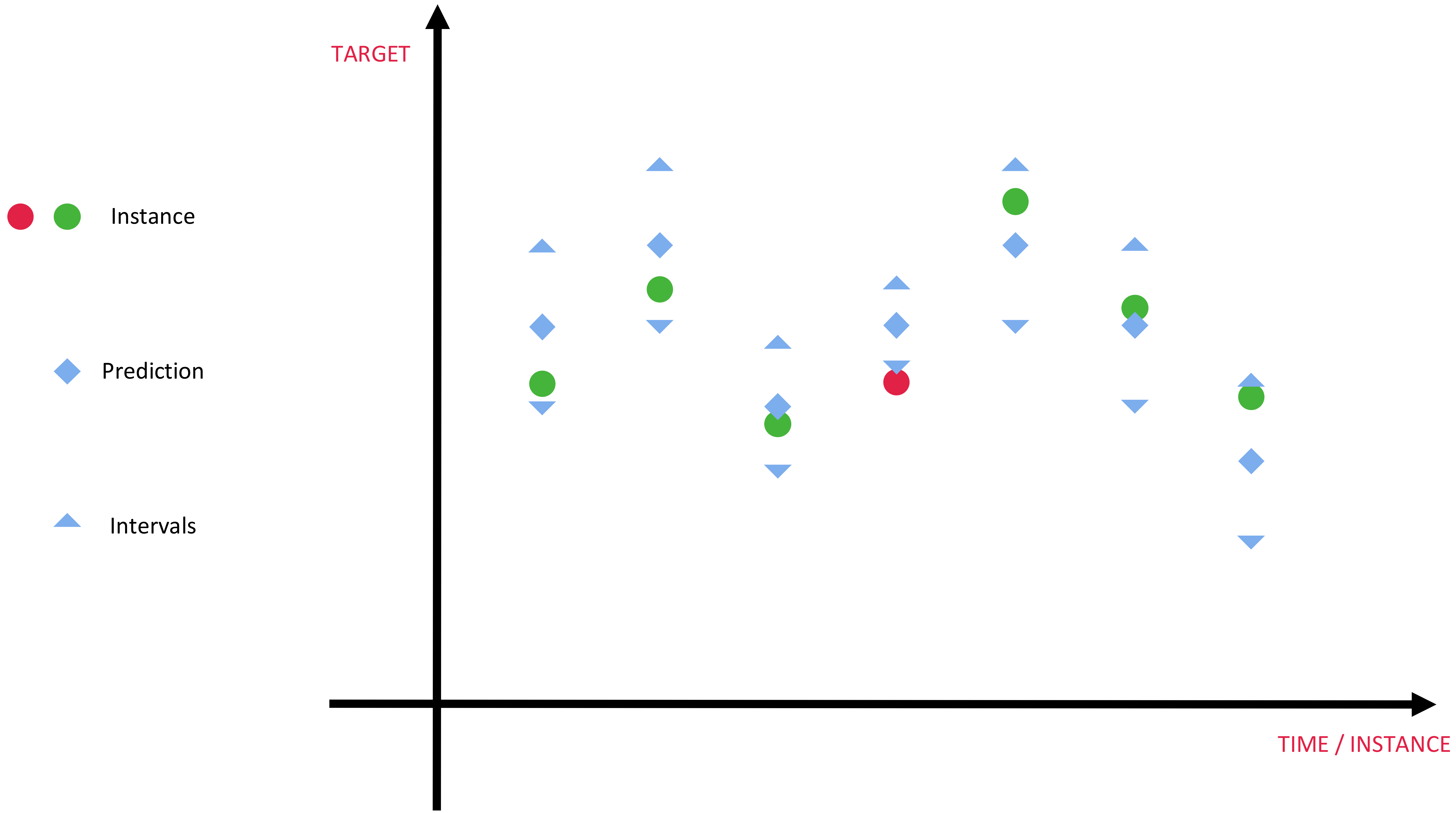
Prediction Intervals

Prediction Intervals (PIs) are very useful improve our confidence in predictions yield in regression tasks

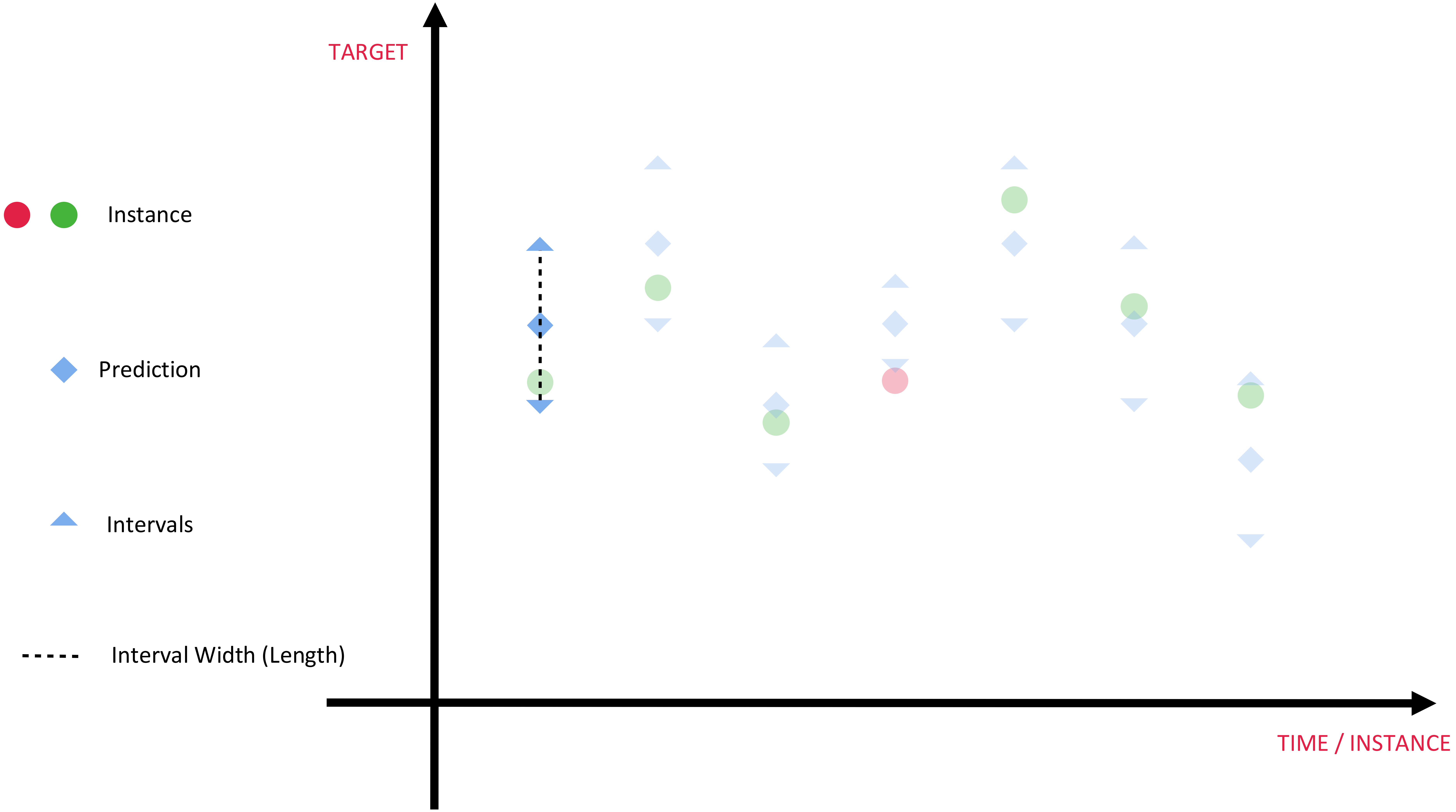
Challenge

Traditional PI methods were not designed to adapt to evolving streams (i.e. those with concept drift)

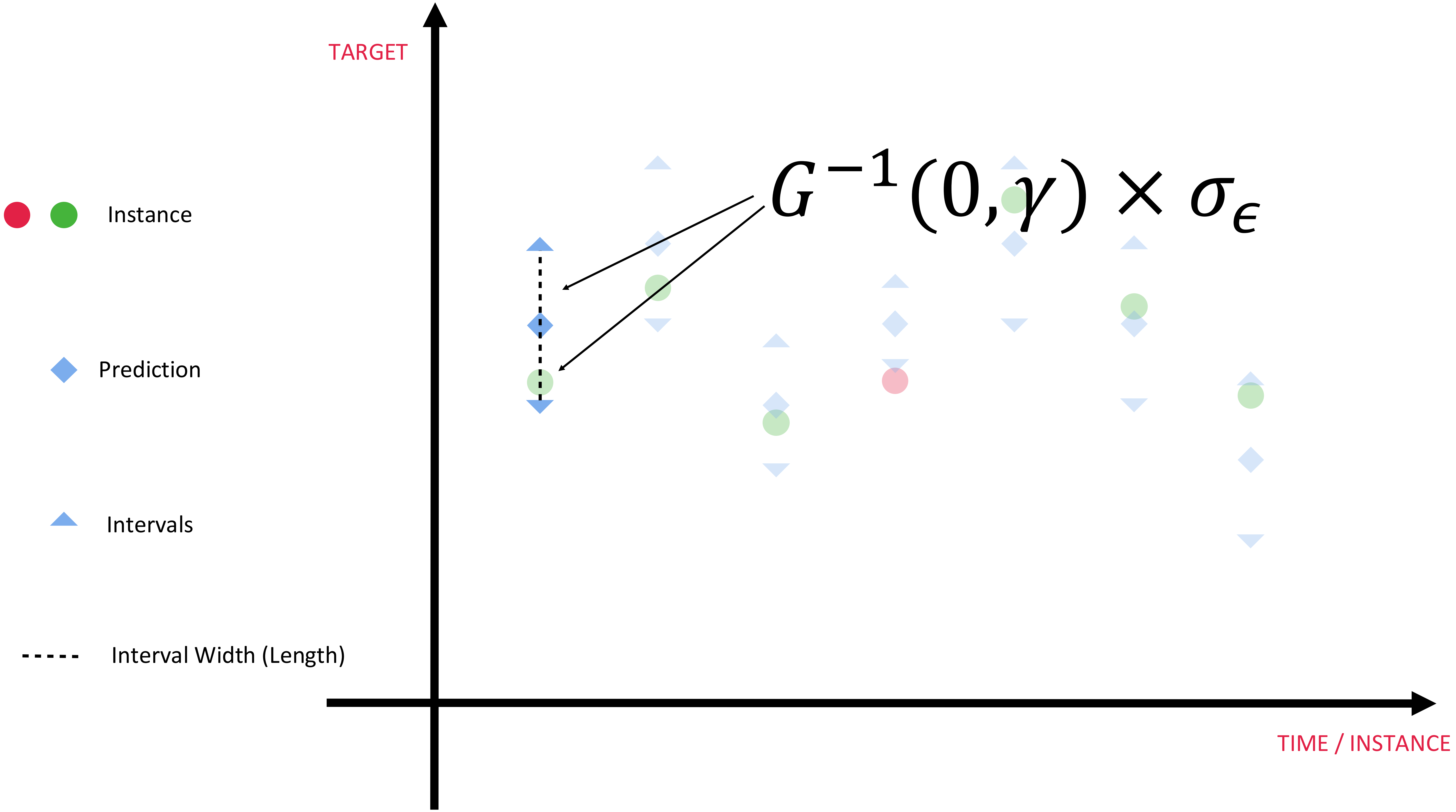
Prediction Interval



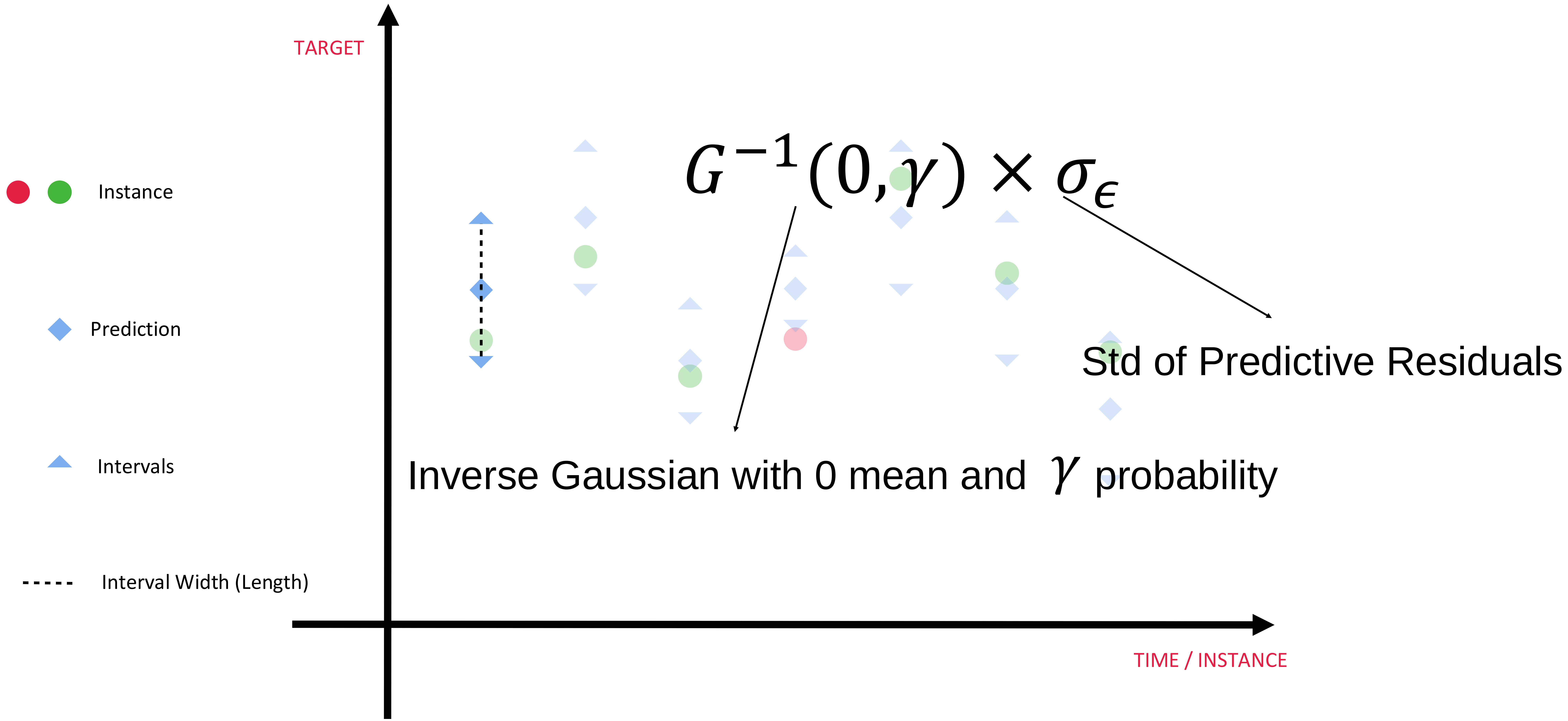
Prediction Interval



Mean and Variance Estimation (MVE)



Mean and Variance Estimation (MVE)



Mean and Variance Estimation (MVE)

$$\Pr\left(y \in \left[\hat{y} - G^{-1}(0,\gamma) \times \sigma_\epsilon, \hat{y} + G^{-1}(0,\gamma) \times \sigma_\epsilon\right]\right) \approx \gamma$$

$$\mathrm{PI}_{\mathrm{MVE}} \in \left(\hat{y} - G^{-1}(0,\gamma) \times \sigma_\epsilon\ , \hat{y} + G^{-1}(0,\gamma) \times \sigma_\epsilon\ \right)$$

γ : Confidence Level / Significance Level

Adaptive Prediction Interval (AdaPI)

$$\Pr\left(y \in [\hat{y} - \mathcal{S} \times G^{-1}(0,\gamma) \times \sigma_\epsilon, \hat{y} + \mathcal{S} \times G^{-1}(0,\gamma) \times \sigma_\epsilon]\right) \approx \gamma$$

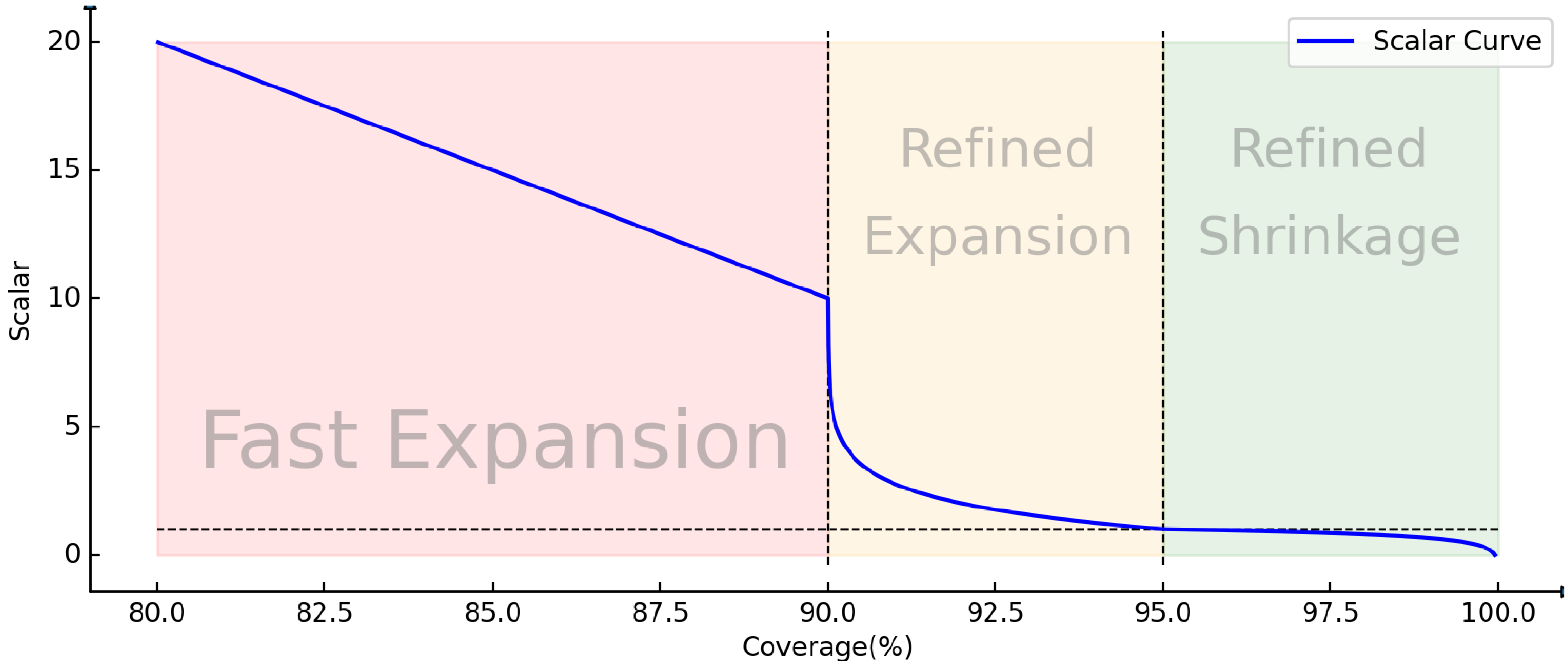
$$\mathrm{PI}_{\mathrm{AdaPI}} \in (\hat{y} - \mathcal{S} \times G^{-1}(0,\gamma) \times \sigma_\epsilon , \hat{y} + \mathcal{S} \times G^{-1}(0,\gamma) \times \sigma_\epsilon)$$

γ : Confidence Level / Significance Level

$\mathcal{S}$: Scalar

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Scalar for AdaPI



Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Evaluation Metrics for PI

$$\text{Coverage} = \frac{1}{N}\sum_{i=1}^{N} I_i$$

$$NMPIW = \frac{\frac{1}{N}\sum_{i=1}^{N}(P_{u_i} - P_{l_i})}{R}$$

NMPIW : Normalized Mean Prediction Interval Width

R : Range of Target Values

P_u, P_l : Upper and Lower Bounds of Prediction Intervals

Expansion Case

Datasets: Bike

Blue area: AdaPI

Red area: MVE

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Shrinkage Case

Datasets: MetroTraffic

Blue area: AdaPI Red area: MVE

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Switching Case

Datasets: Abalone

Blue area: AdaPI Red area: MVE

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Dealing with Drifts

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression." PAKDD (2024)

Prediction Intervals Summary

Prediction Interval (PI) is essential for uncertainty quantification in regression tasks.

Challenges

Traditional PI methods are not suitable for dynamic data streams.

Solution

Mean and Variance Estimation (MVE); ADAPI*

Evaluation

Coverage

Normalised Mean Prediction Interval Width (NMPIW)

Yibin Sun, B Pfahringer, H M Gomes, and A Bifet. "Adaptive Prediction Interval for Data Stream Regression.” PAKDD (2024)

Practical examples

02_ECML2024_supervised.ipynb