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*.

Bagging

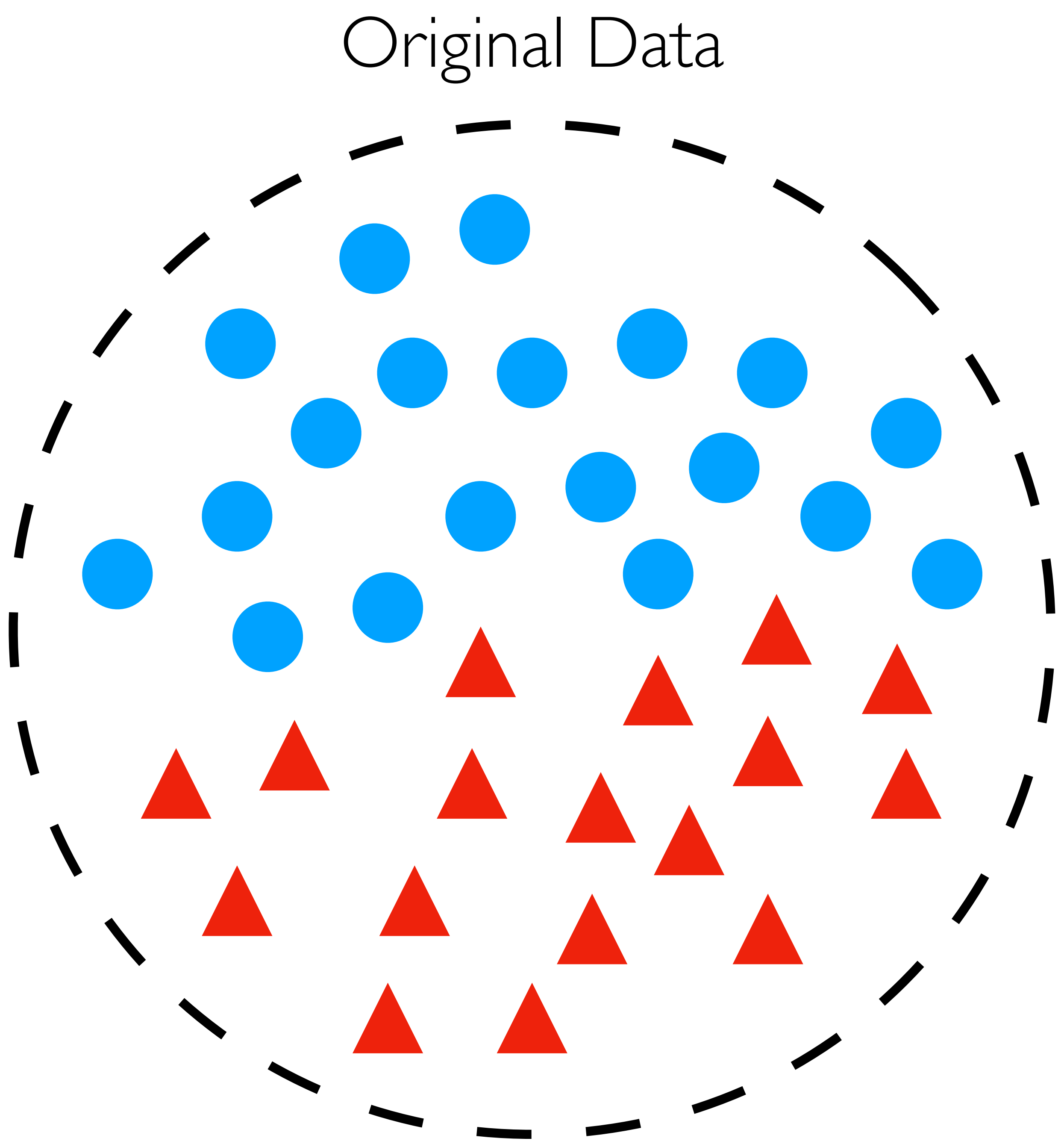
Bootstrap **Agg**regat**ing**

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

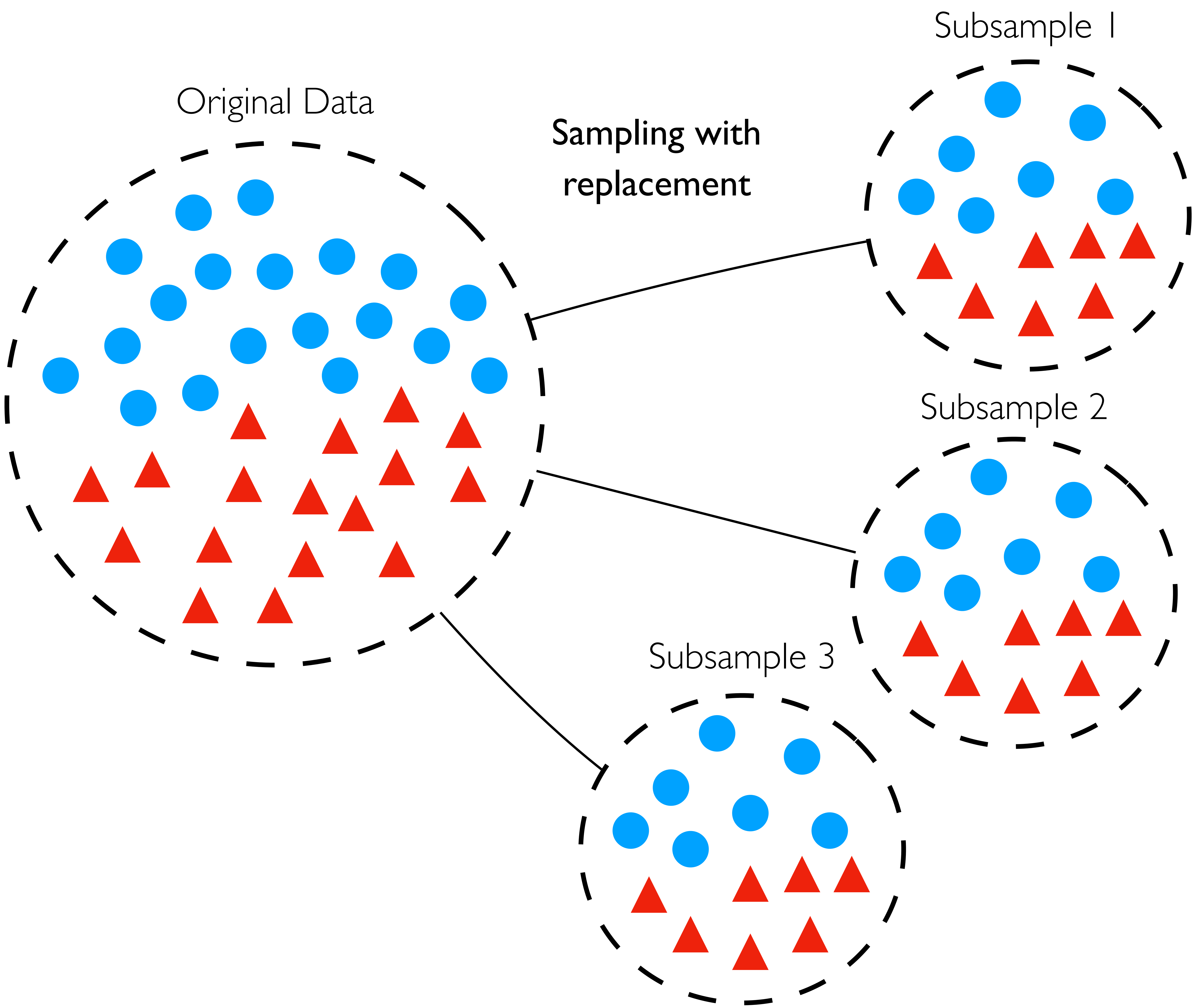
Every bootstrap contains each original sample **K** times, where **Pr(K=k)** follows a binomial distribution.

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

Bagging

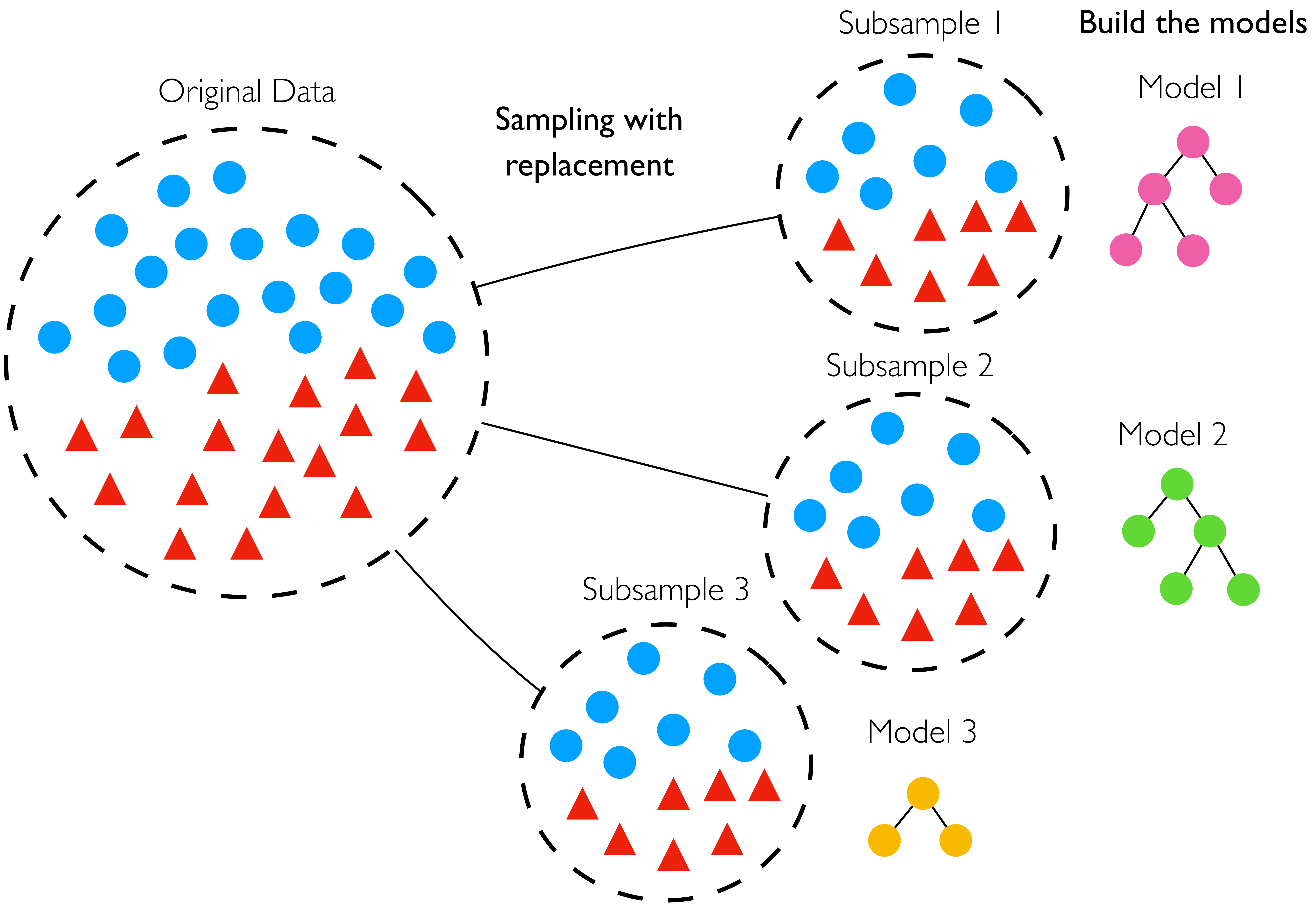


Bagging



Original Data
Sampling with replacement
Subsample 1
Subsample 2
Subsample 3

Bagging


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

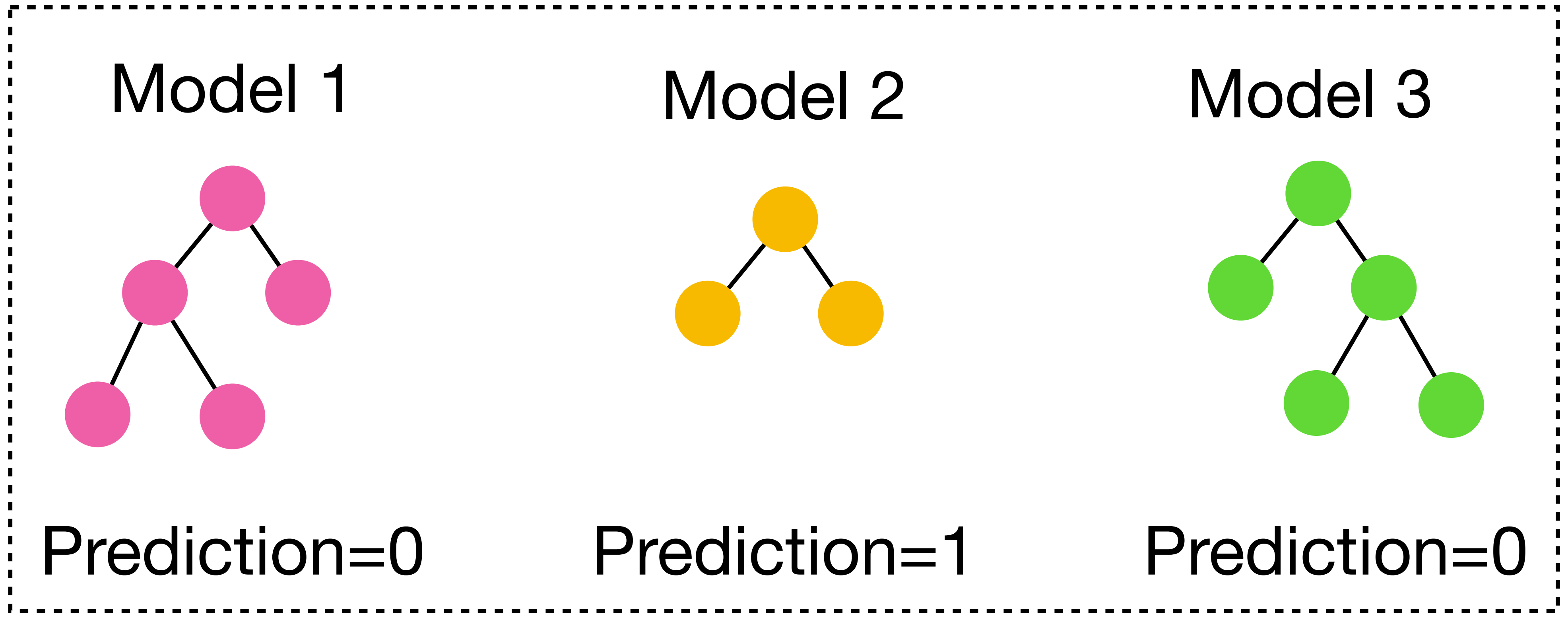
* Out-Of-Bag (OOB)

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

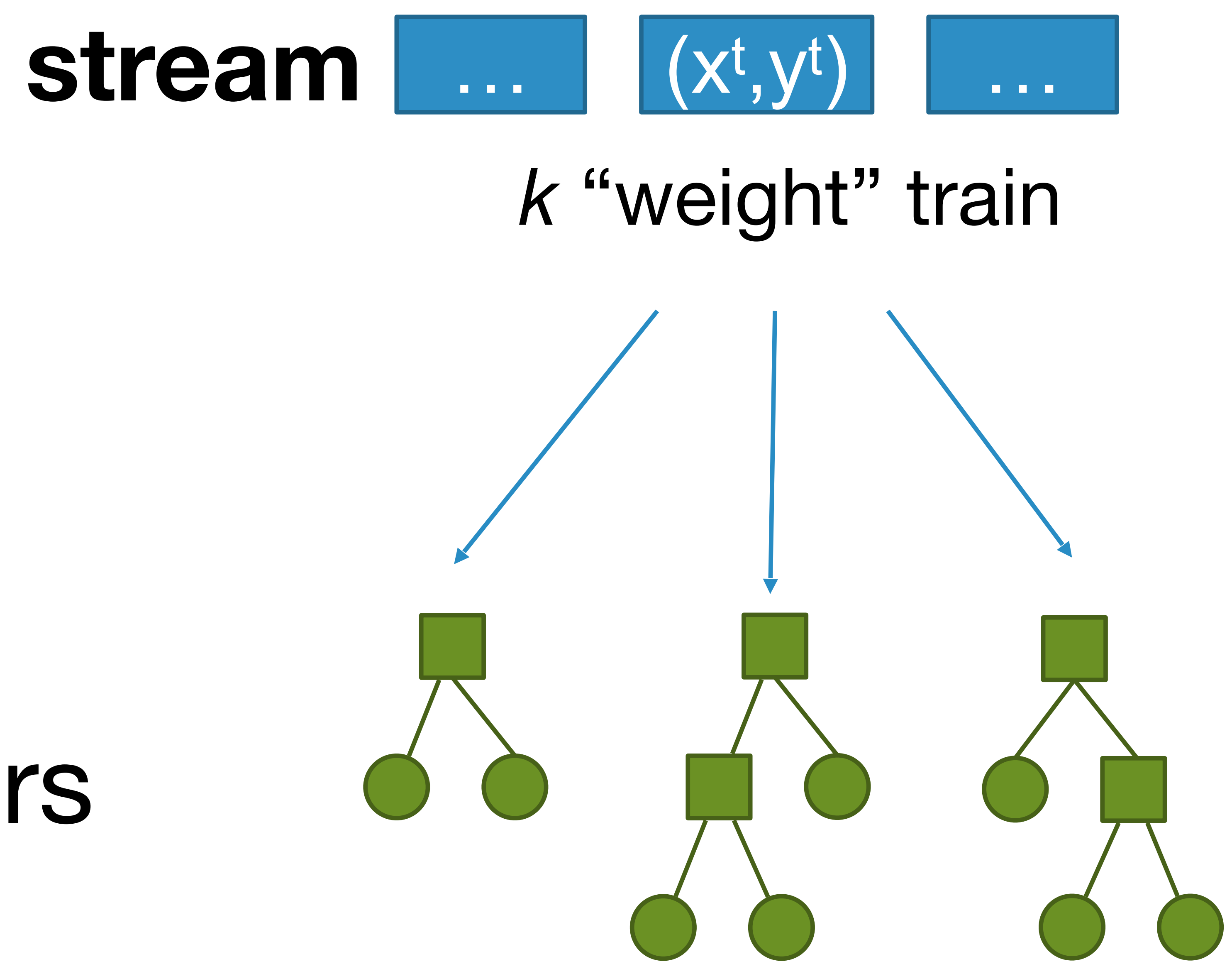
Online Bagging

- 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

$k \leftarrow Poisson$ (λ=1)
if $k > 0$ **then**
    $l \leftarrow FindLeaf(t, x)$
    $UpdateLeafCounts(l, x, k)$



Practical effect: train learners with different subsets of instances.

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

Subsamples

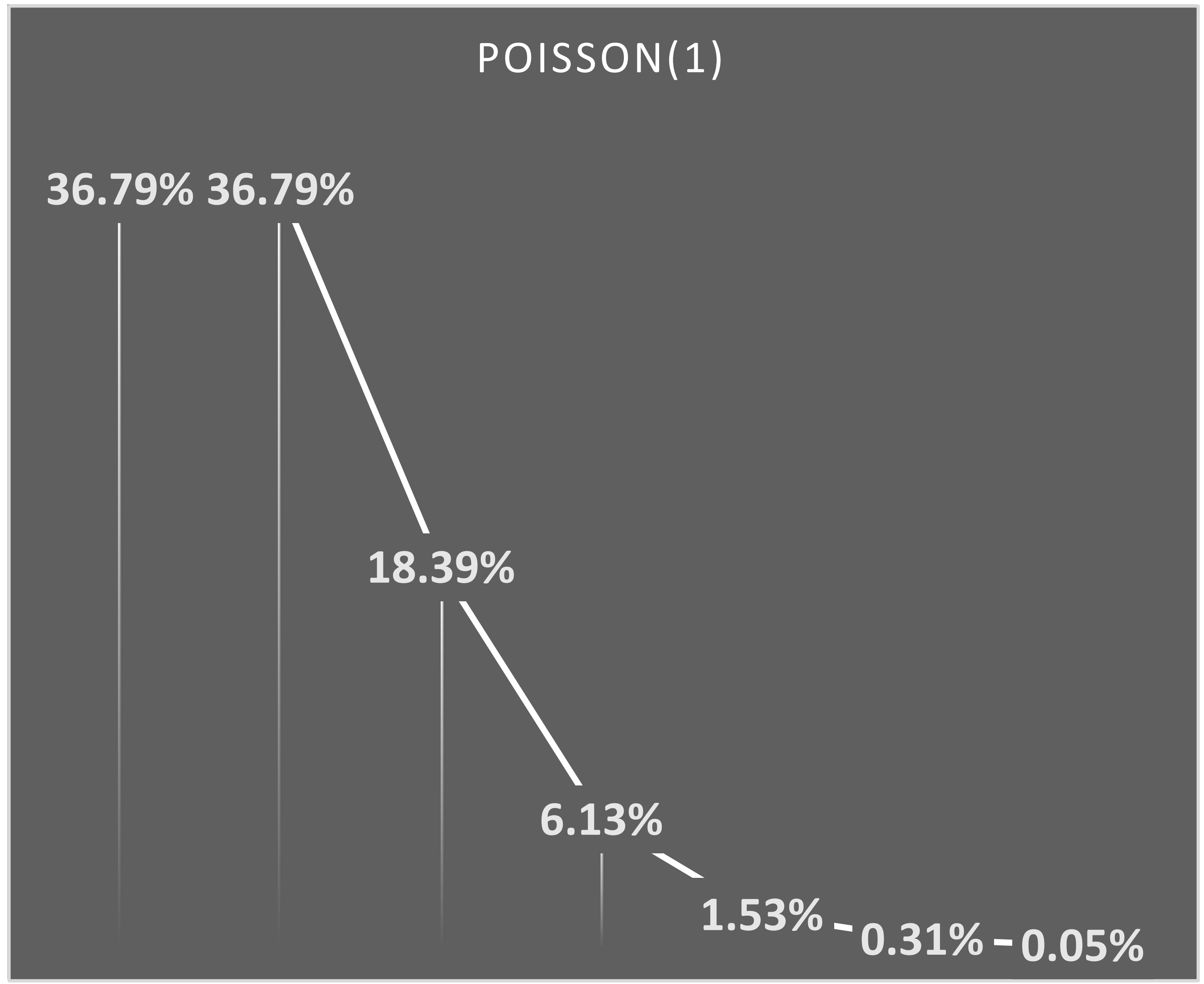
Batch bagging

~64% from the original dataset

~37% are repeated

~37% are not present

Online bagging



$k =$ 0 1 2 3 4 5 6 …

Adaptive Random Forest (ARF)

Streaming version of the original Random Forest by Breiman

Uses a variation of the Hoeffding Tree

Main differences:

Bootstrap aggregation and the **base learner**

Overview:

1. Online bagging
2. <u>Random subset of features</u>
3. Drift detector for each tree

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

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

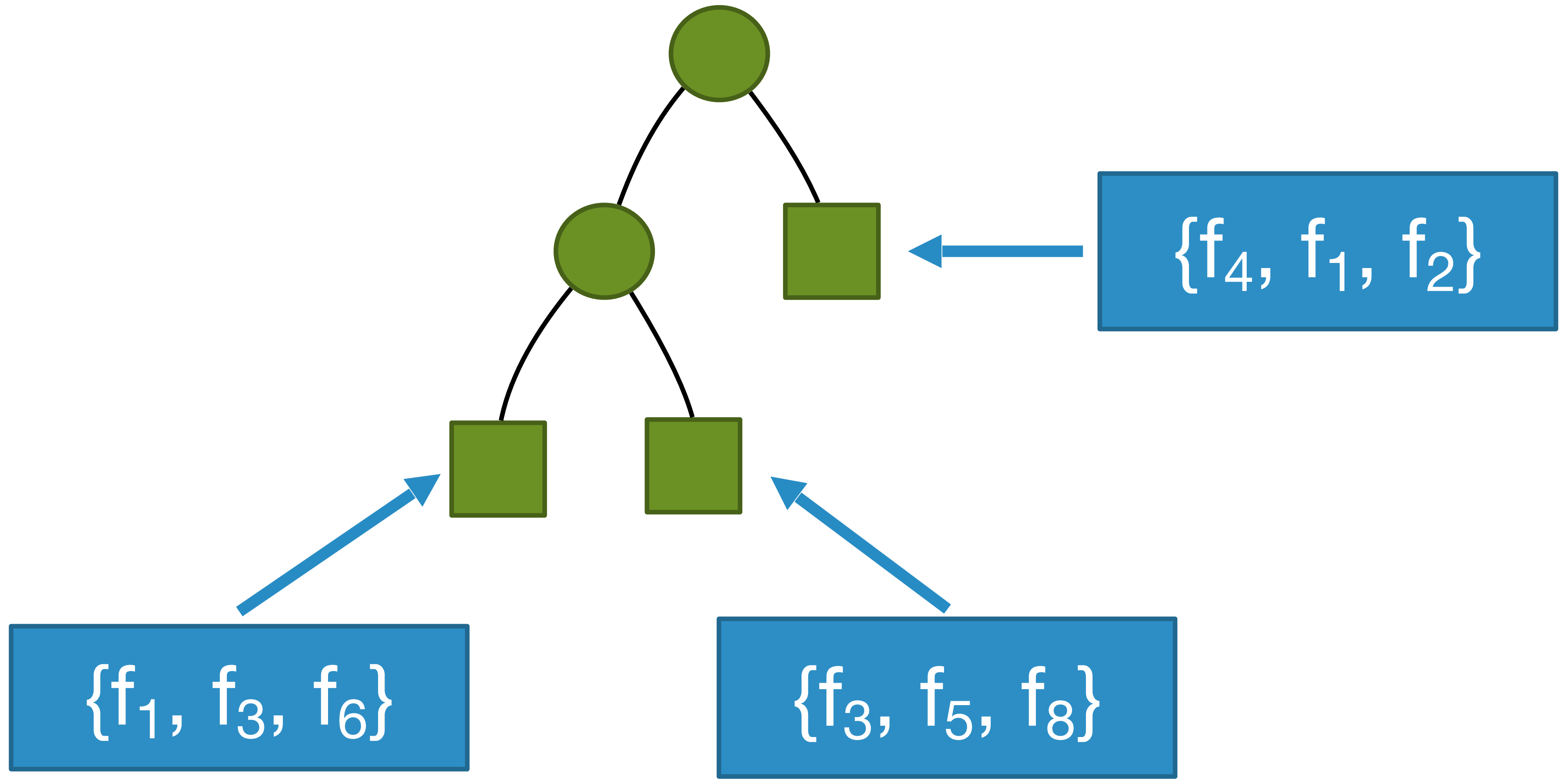
ARF: Drift Detection and Adaptation

- 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**.

Randomizing the feature set

Local randomization

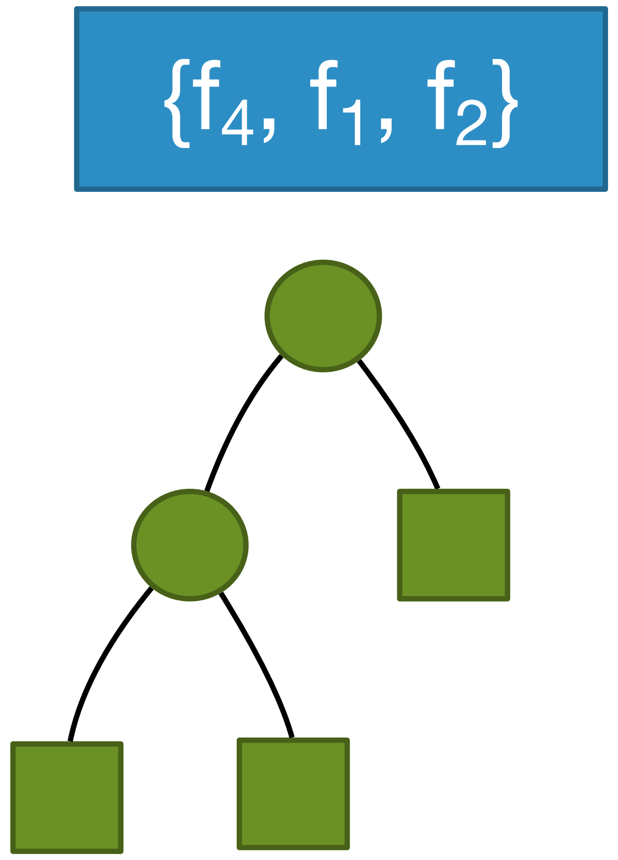
Random Forest



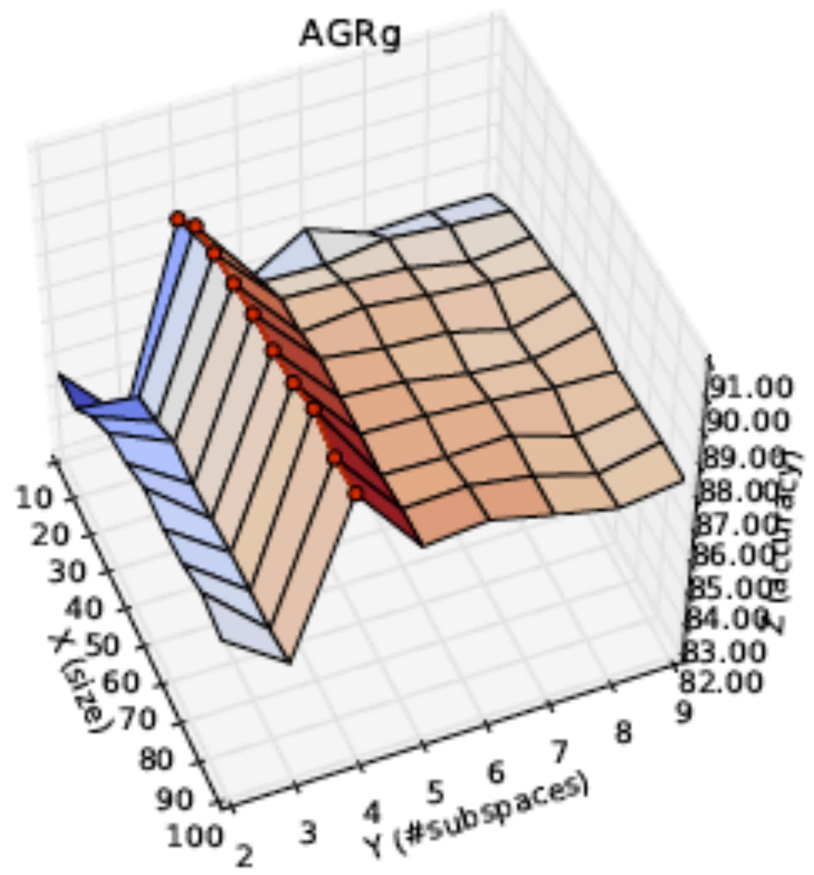
Global randomization

Random Subspaces

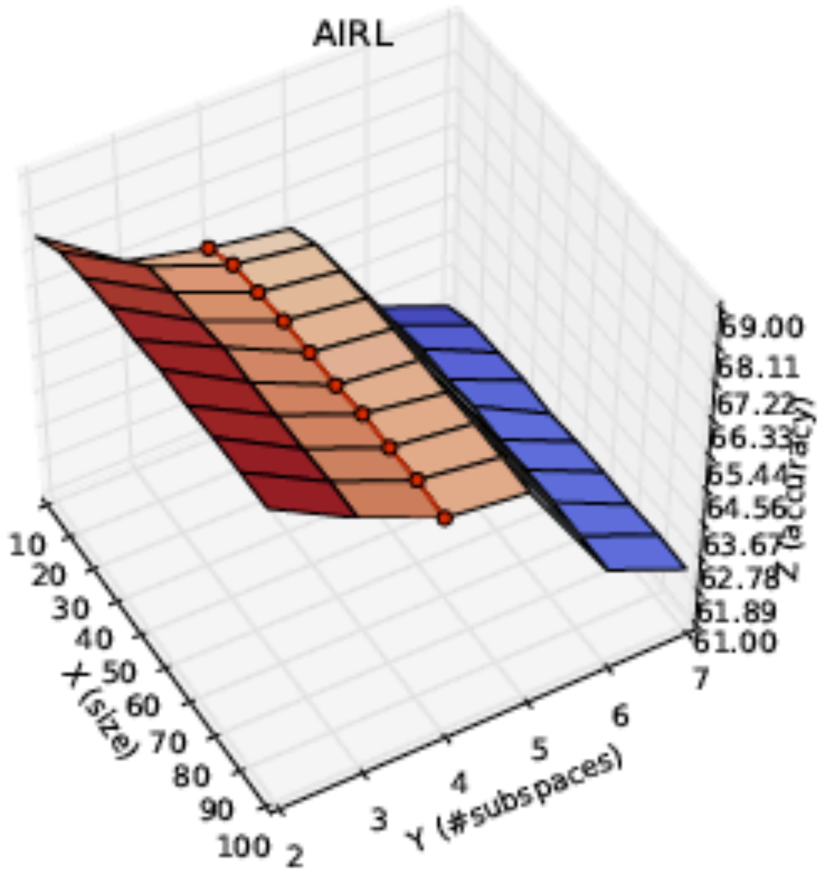
Random Patches



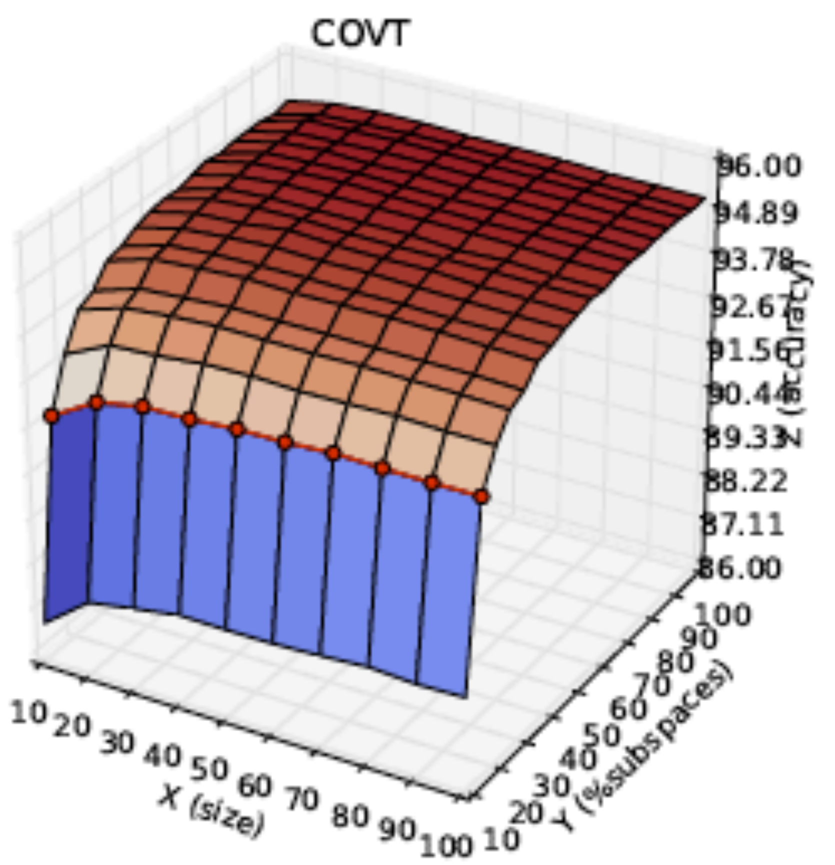
Impact of subspace size



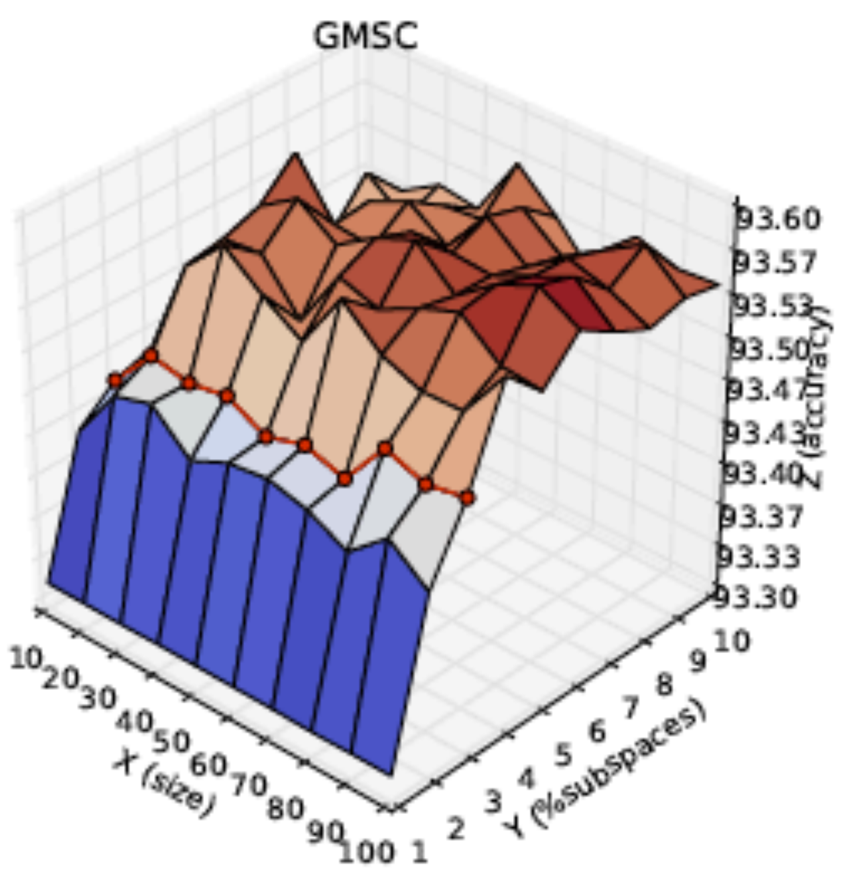
(a) AGR$_g$



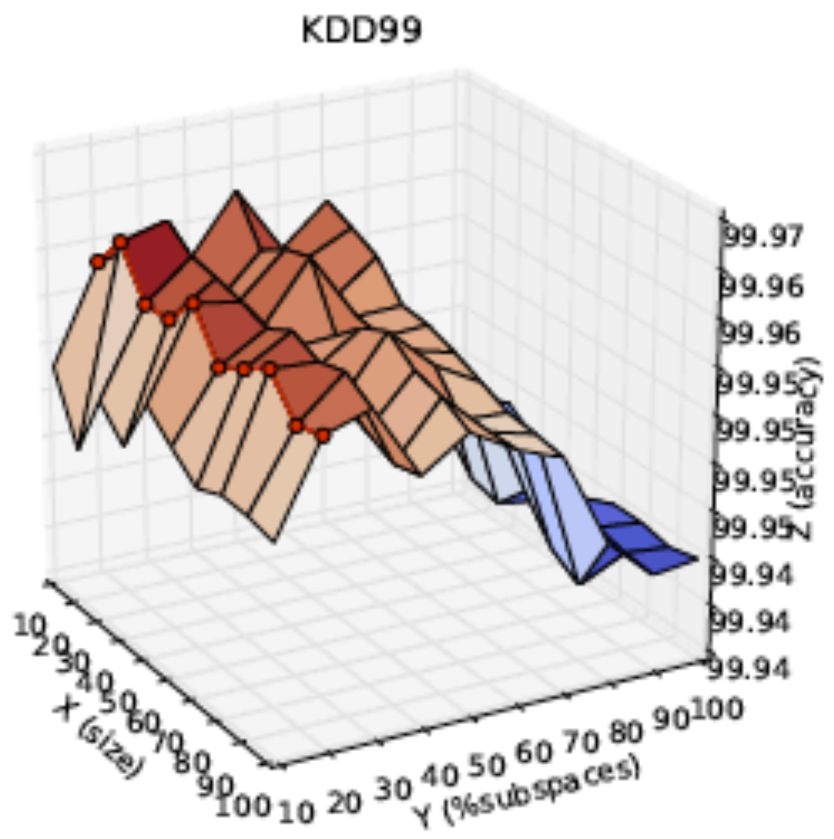
(b) AIRL



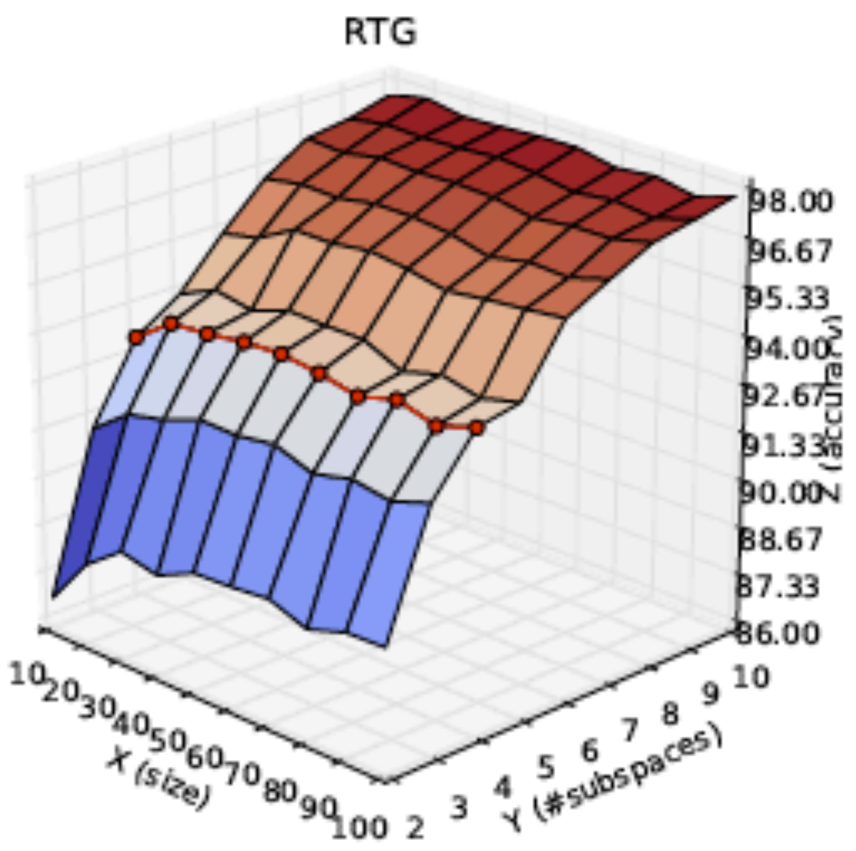
(c) COVT



(d) GMSC



(e) KDD99



(f) RTG

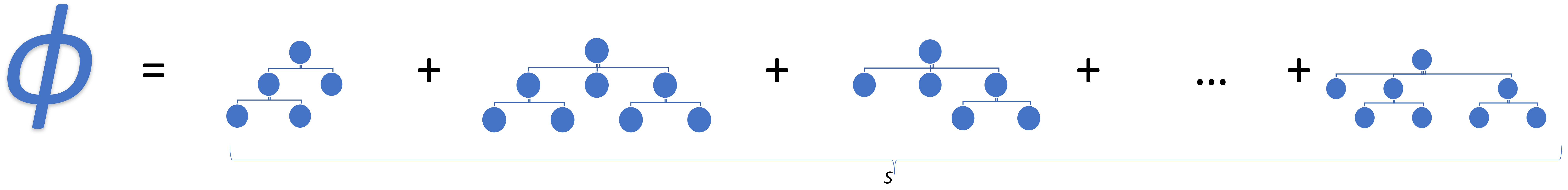
Boosting

- XGBoost[1] and CatBoost[2] are popular **batch boosting** methods
- **Challenges** to streaming:
    - **adjusting** the **booster online** after a **concept drift**

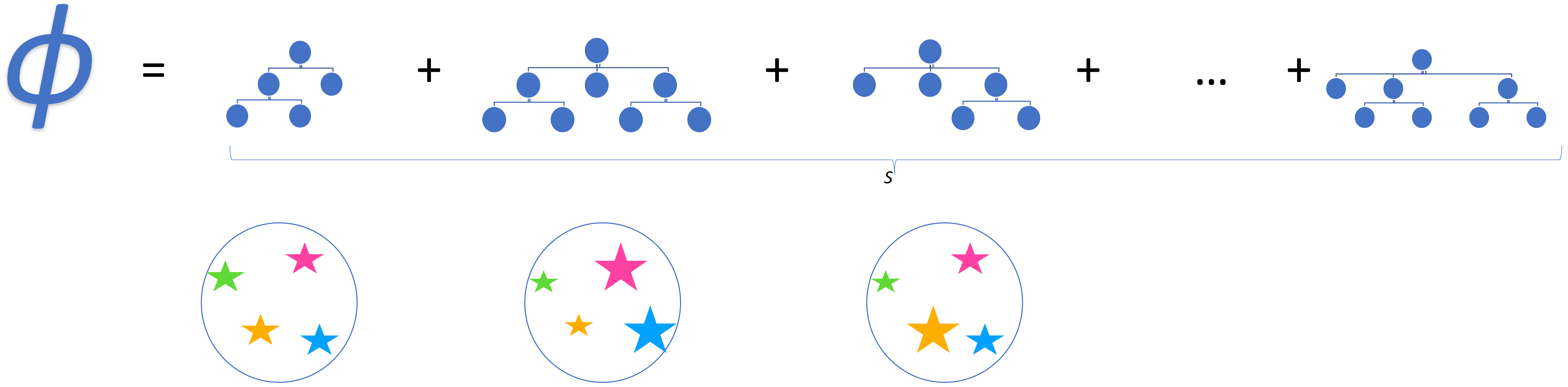
[1] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discov and Data Mining, pages 785–794. ACM, 2016.
[2] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems, 31.

Boosting

The ensemble is built in an **additive** manner, **sequentially** adding trees.

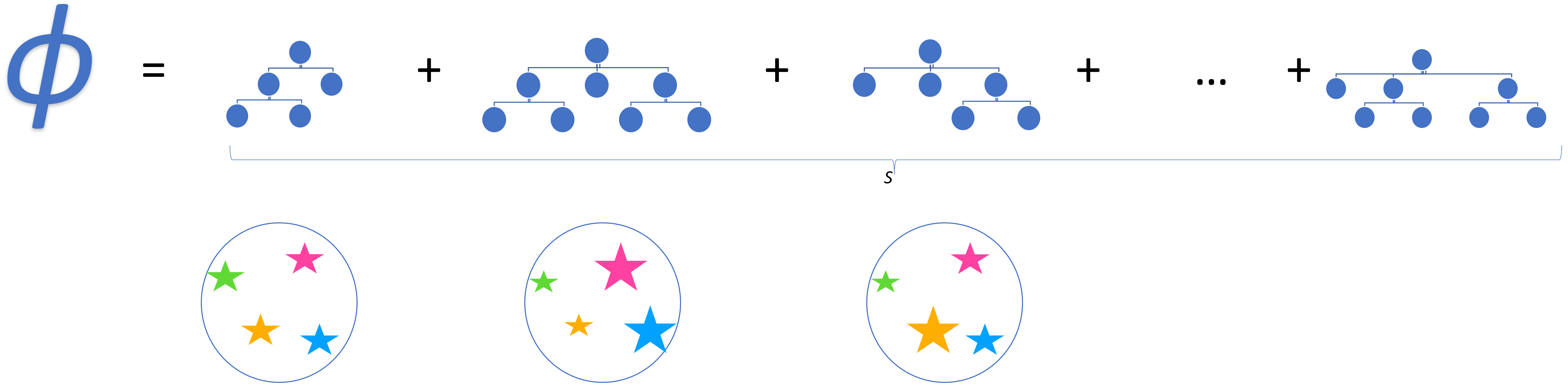


Boosting



The **previous learner's prediction (loss)** is considered to assign a **weight** to an instance.

Gradient Boosting



Gradient boosting uses **gradient information** to **assign weights** to instances.

Boosting on Streams

- Trees are configured in a **boosting** setup
- OzaBoost [1] uses **weights** from a **Poisson(1)** distribution to **train multiple times** using a given instance.
  - Similar to Online Bagging
- **Online Smooth Boost** [2] is analogous to batch SmoothBoost for imbalanced data.
  - uses a **smooth distribution** for weight assignment
- Gradient boosted AXGB [3] use
  - **mini-batch trained XGBoost** as its base learners
  - **adjusts the booster** when concept drifts are detected by **ADWIN**
- Challenges
  - **Not as good as bagging** based stream learners [3]
  - AXGB **only supports binary class** problems
- Streaming Gradient Boosted Trees (SGBT) **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

Streaming Gradient Boosted Trees (SGBT)

- Uses ***weighted squared loss*** explained in [1,2]
  - with ***hessian(h_i)* as the weight** and ***gradient over hessian(g_i / h_i)*** as **the target** *considering previous boosting* step:

$$\sum_{i=1}^{n} \frac{1}{2} h_i (f_s(x_i) - g_i/h_i)^2 + \Omega(f_s) + constant$$

penalises the complexity of the tree

- This allows one to use ***any streaming regression tree*** instead of the one used in XGBoost [2].

Gunasekara, N., Pfahringer, B., Gomes, H., & Bifet, A. (2024). Gradient boosted trees for evolving data streams. Machine Learning, 113(5), 3325-3352.
[1] Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics 28(2), 337–407 (2000)
[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016.

Streaming Gradient Boosted Trees (SGBT)

- Utilises **trees** with:
  - **drift detectors** to monitor **standardised absolute error**.
  - **grows a background** tree when it **reaches a warning zone**.
  - **replaces the active** tree **with background** tree when it **reaches a danger zone**.
- Uses a **subset of features**.
- Multi-class support
  - **committee of regression trees** at a **given boosting step**.
  - **binary SGBT** for **each class**.

Gunasekara, N., Pfahringer, B., Gomes, H., & Bifet, A. (2024). Gradient boosted trees for evolving data streams. Machine Learning, 113(5), 3325-3352.

SGBT re-cap

- **Online boosting under concept drift** is more challenging due to the **sequential ensemble setup**.
- SGBT allows **each tree to monitor its error** and **adjust to concept drifts** without **sacrificing predictive** power
  - Supports multi class problems

Ensembles re-cap

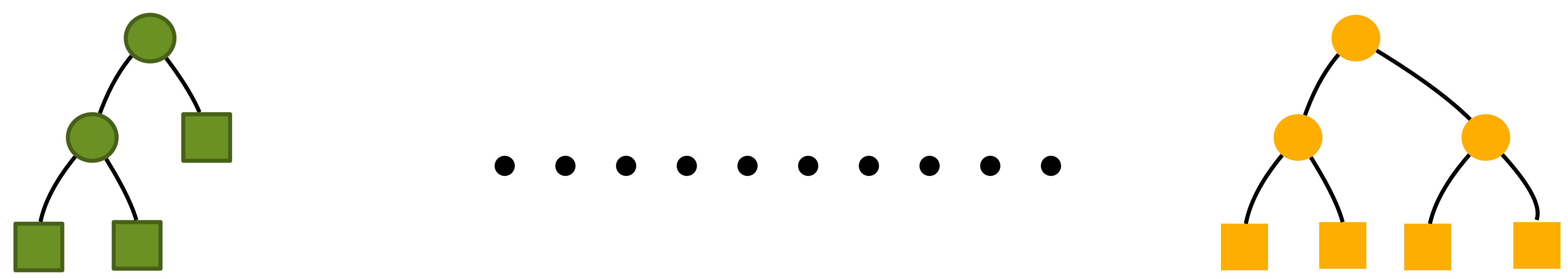
- Use **Poisson** distribution to derive weights for multiple training iterations
- More advance methods use **drift detectors** to adapt to changes
- **SGBT** performs better than bagging based methods
- Latest developments
  - Use **task parallelism** for **bagging ensembles** using **mini-batches** [1, 2]

[1] G. Cassales, H. M. Gomes, A. Bifet, B. Pfahringer and H. Senger, Improving the performance of bagging ensembles for data streams through mini-batching, Information Sciences,Volume 580, 2021, Pages 260-282, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2021.08.085.
[2] G. Cassales, H. M. Gomes, A. Bifet, B. Pfahringer and H. Senger, "Balancing Performance and Energy Consumption of Bagging Ensembles for the Classification of Data Streams in Edge Computing," in IEEE Transactions on Network and Service Management, vol. 20, no. 3, pp. 3038-3054, Sept. 2023, doi: 10.1109/TNSM.2022.3226505

Regression algorithms

Adaptive Random Forest Regression

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



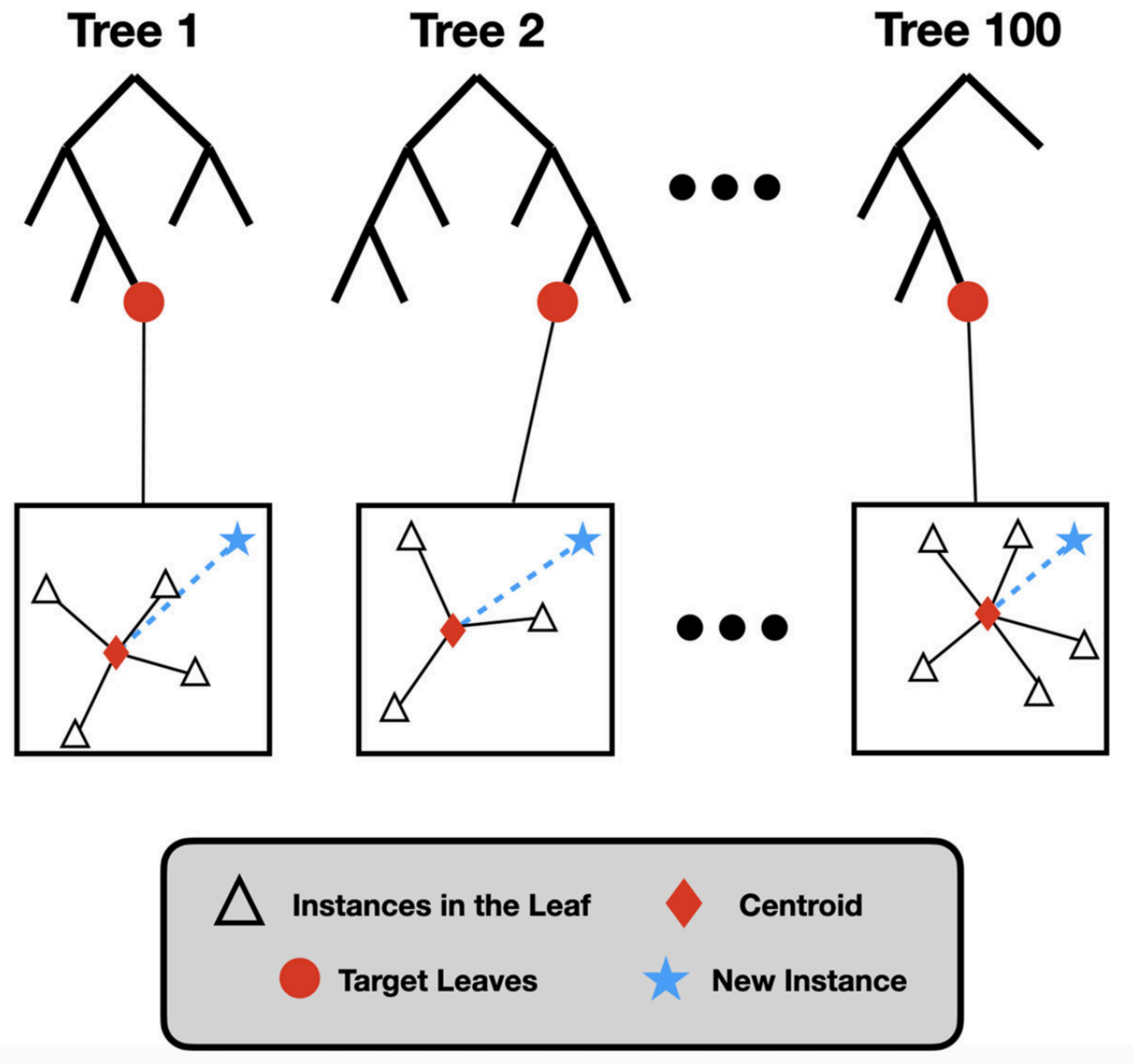
Gomes HM, Barddal JP, Ferreira LEB, Bifet A (2018) Adaptive random forests for data stream regression. ESANN

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)

Self-Optimizing k-Nearest Leaves (SOKNL)

at time t

K :	1	2	**3**	4	5
Error:	9.85	7.26	6.97	8.66	8.20

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)

at time t + Δ

K :	1	2	3	4	**5**
Error:	9.94	8.65	8.24	7.95	7.61

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)

Practical examples

IJCAI_2024_supervised.ipynb