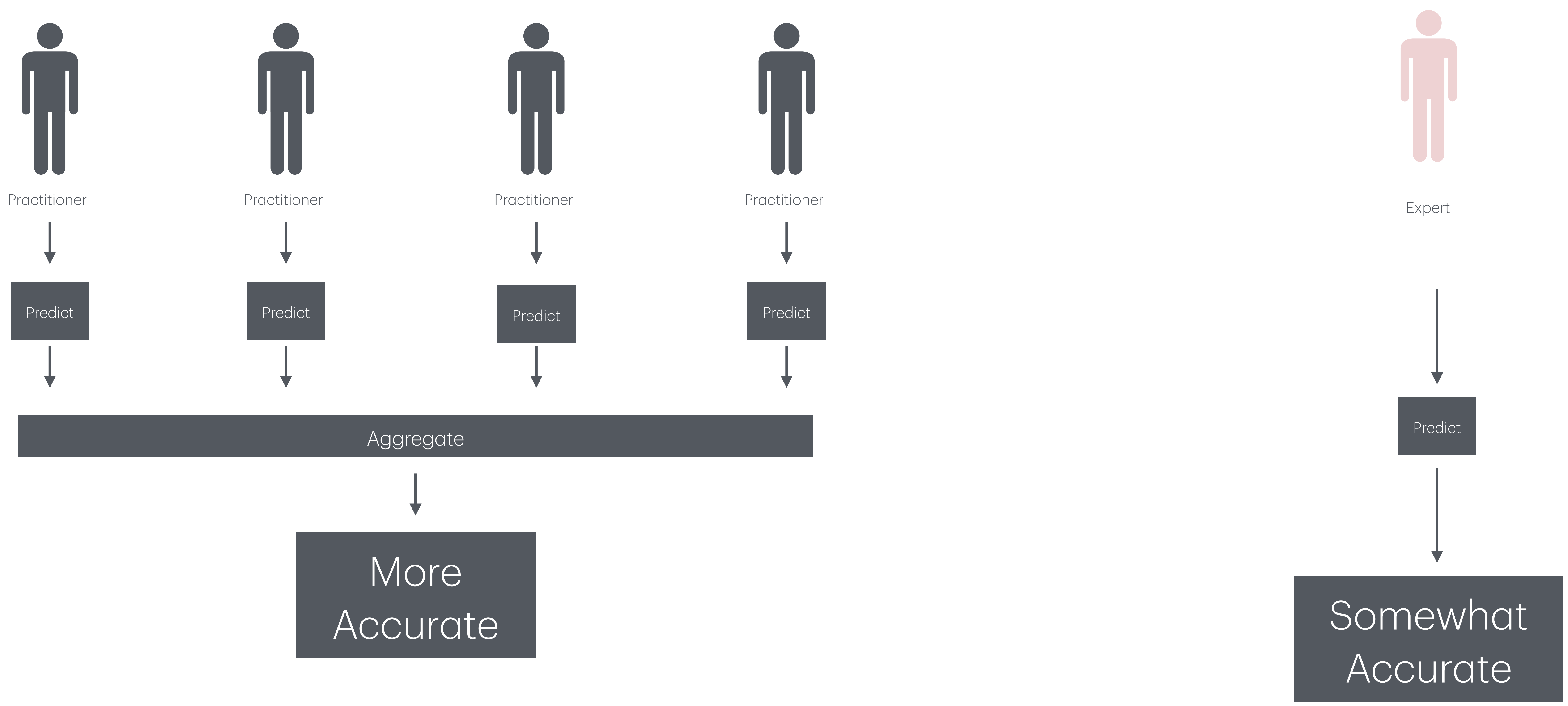
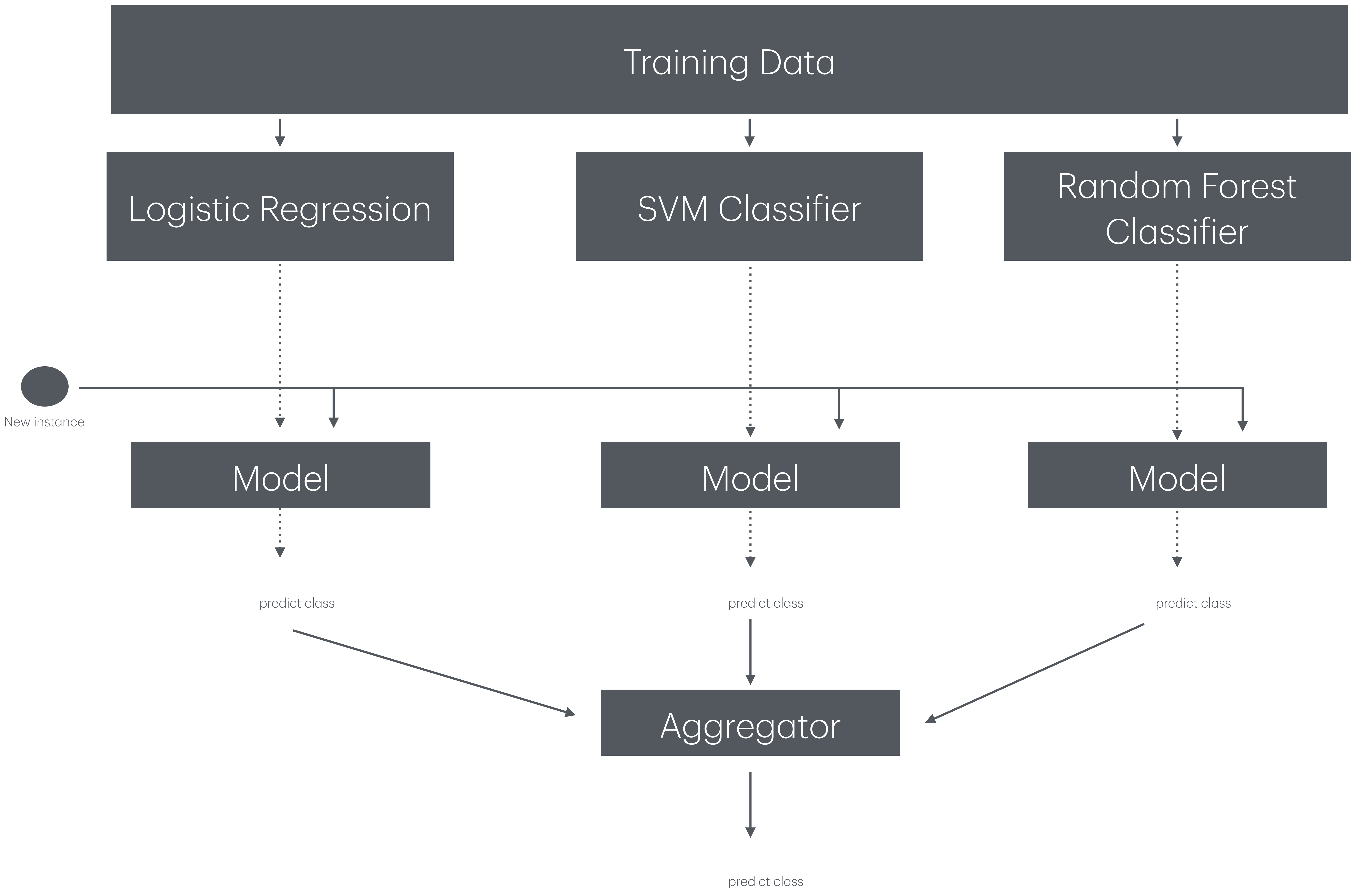
Ensemble


Practitioner
Practitioner
Practitioner
Practitioner
Predict
Predict
Predict
Predict
Aggregate
More Accurate
Expert
Predict
Somewhat Accurate

Hard Classifier



Aggregator -> Majority hard-vote classifier
- mode

Law Large Numbers

In probability and statistics, a Bernoulli process is a finite or infinite sequence of binary random variables, so it is a discrete-time stochastic process that takes only two values, canonically 0 and 1. The component Bernoulli variables Xi are identically distributed and independent. Prosaically, a Bernoulli process is a repeated coin flipping, possibly with an unfair coin. Every variable Xi in the sequence is associated with a Bernoulli trial or experiment.

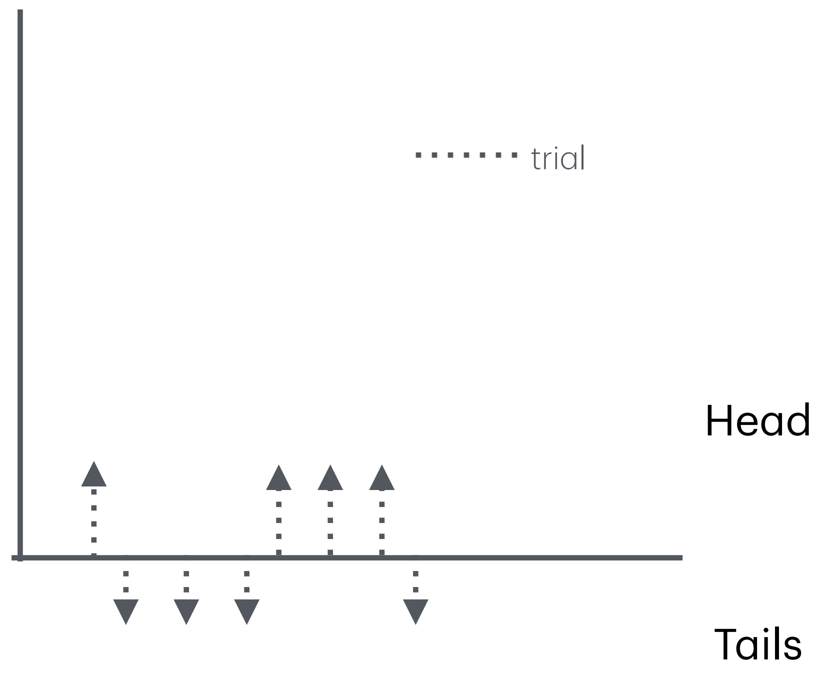
Wikipedia

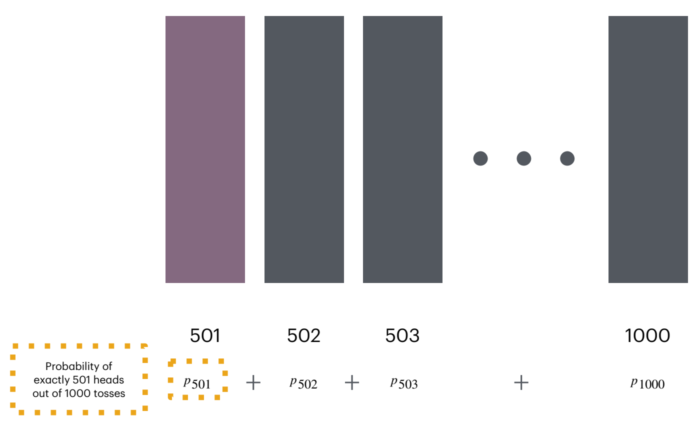
binomial distribution

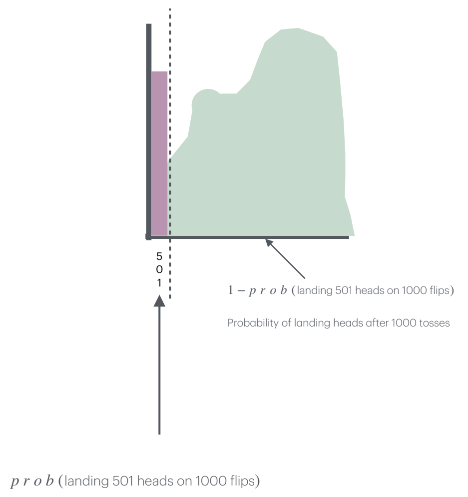
$$P(k) = \binom{n}{k} p^k q^{n-k}$$

Features

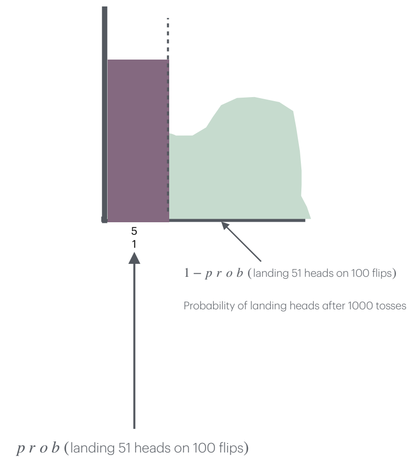
Head=1	Tails =0





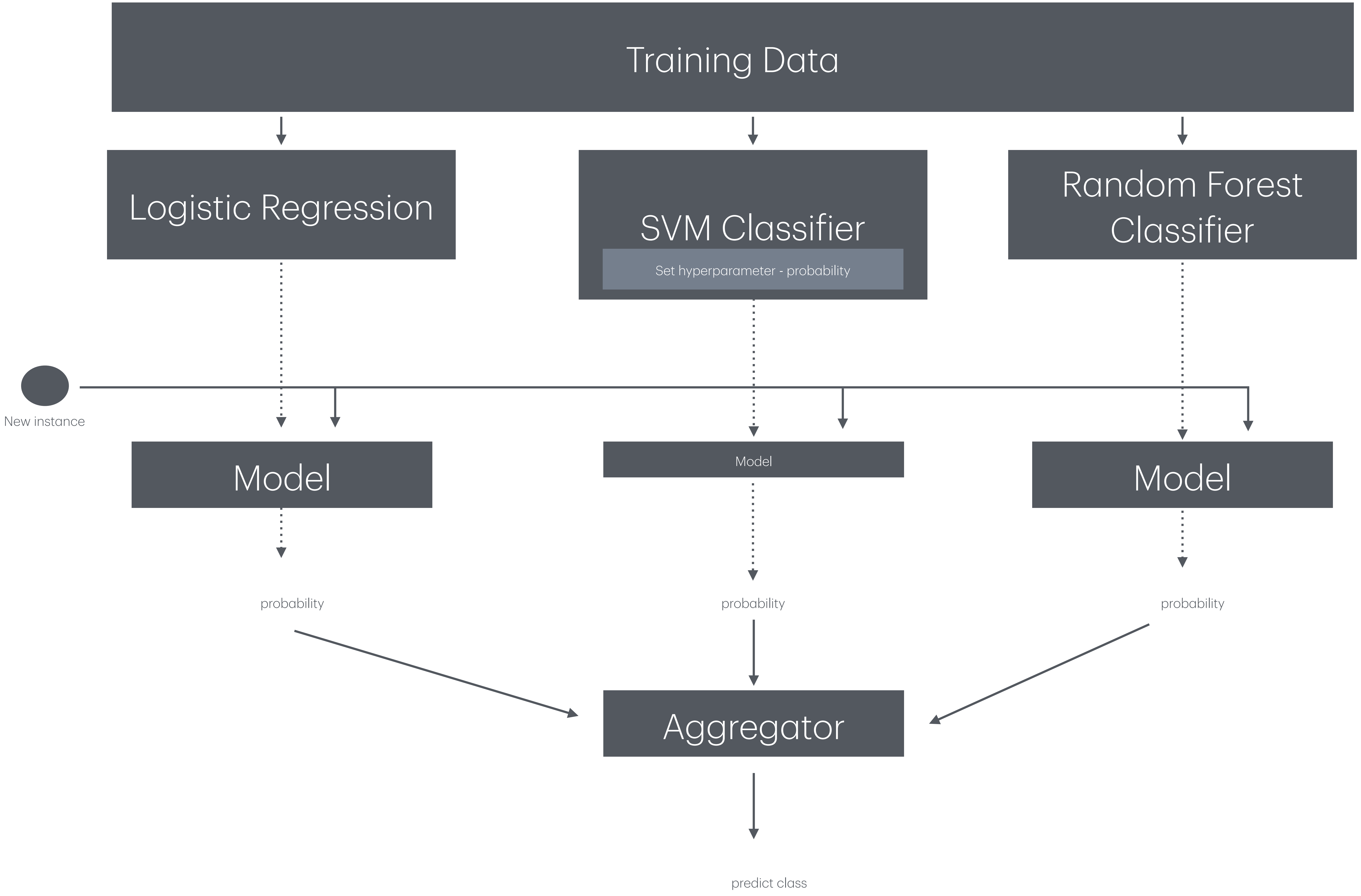


Larger rv distribution



Smaller rv distribution

Soft Classifier

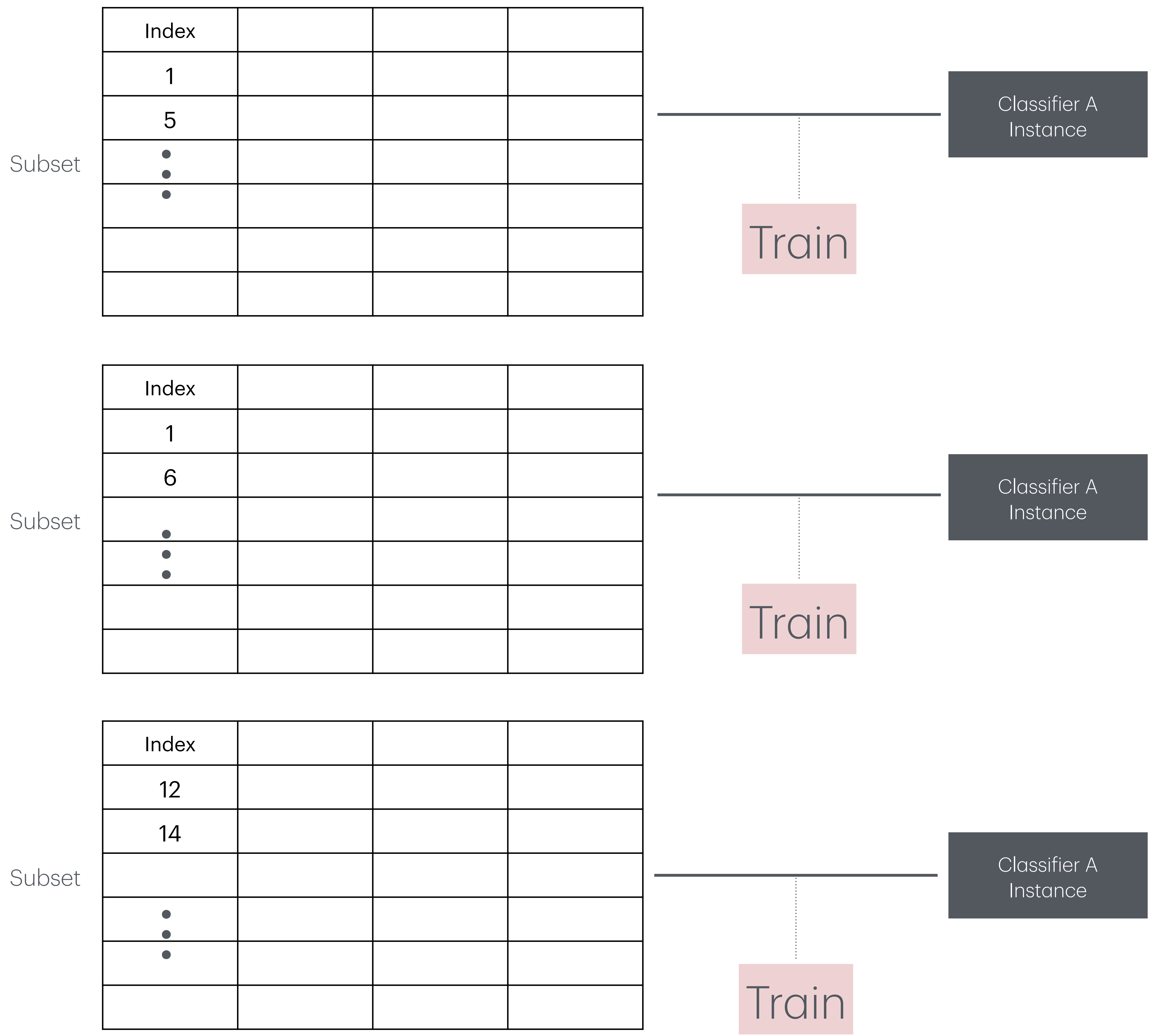


Aggregator -> Majority soft-vote
Selects class with highest probability averaged across all classifiers

Bagging

Classifier A

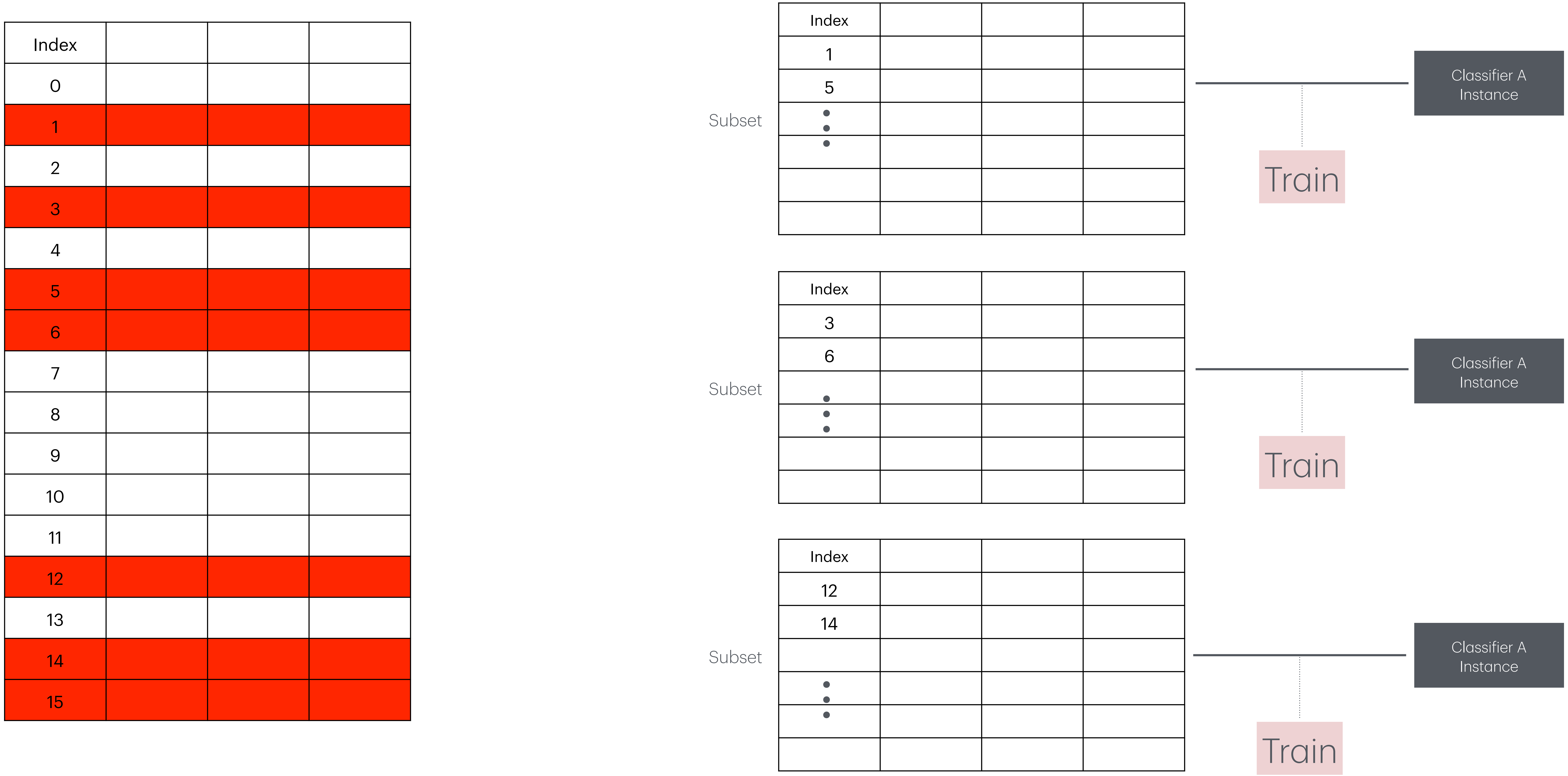
Index			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Sampling replacement. Indices can be reused by differing subsets

Pasting

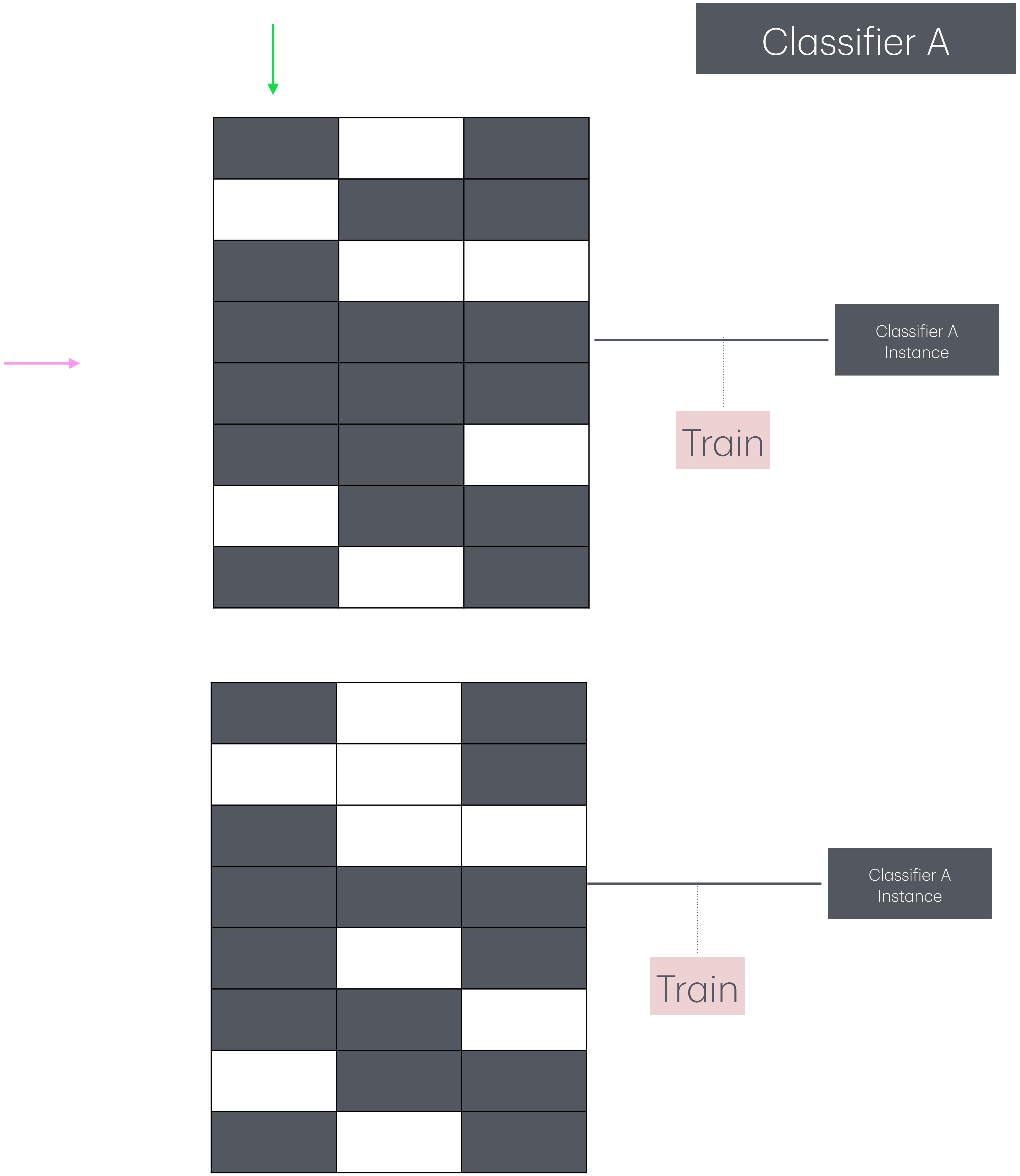
Classifier A



Sampling without replacement. Indices in red are not able to be fetched after usage

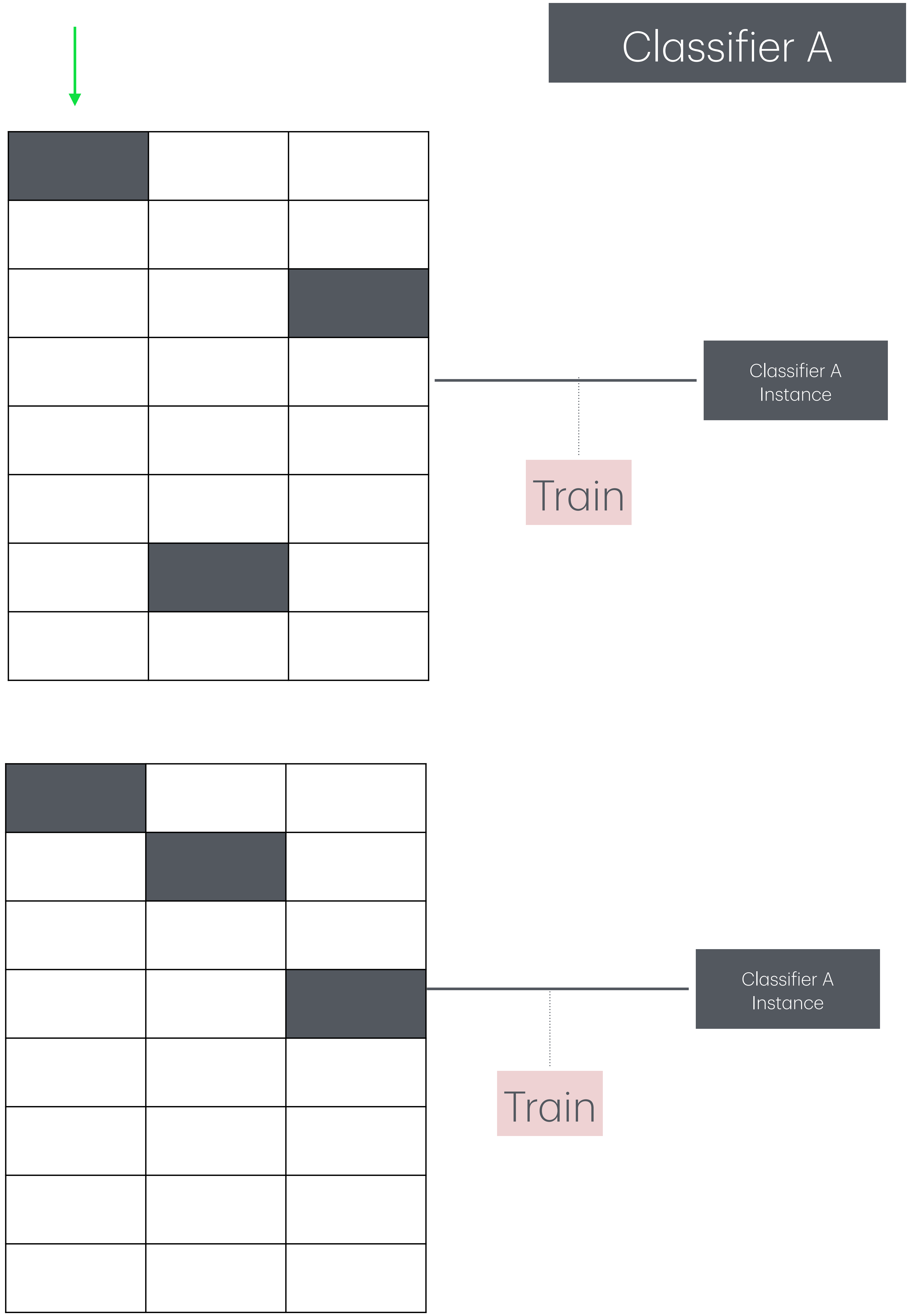
Random Patches

Sampling both
- Training instances
- Features



Random Subspaces

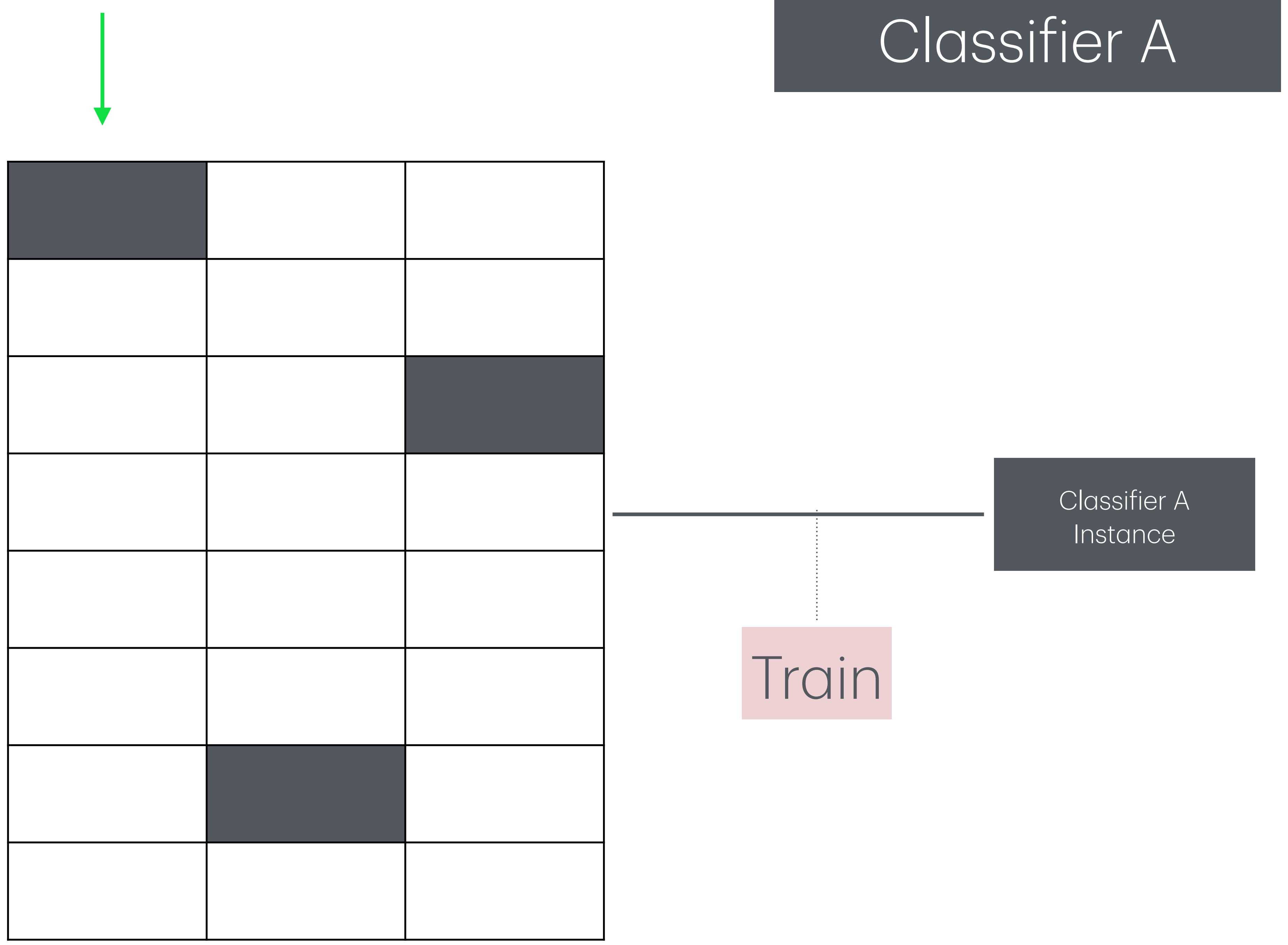
Sampling
- Features



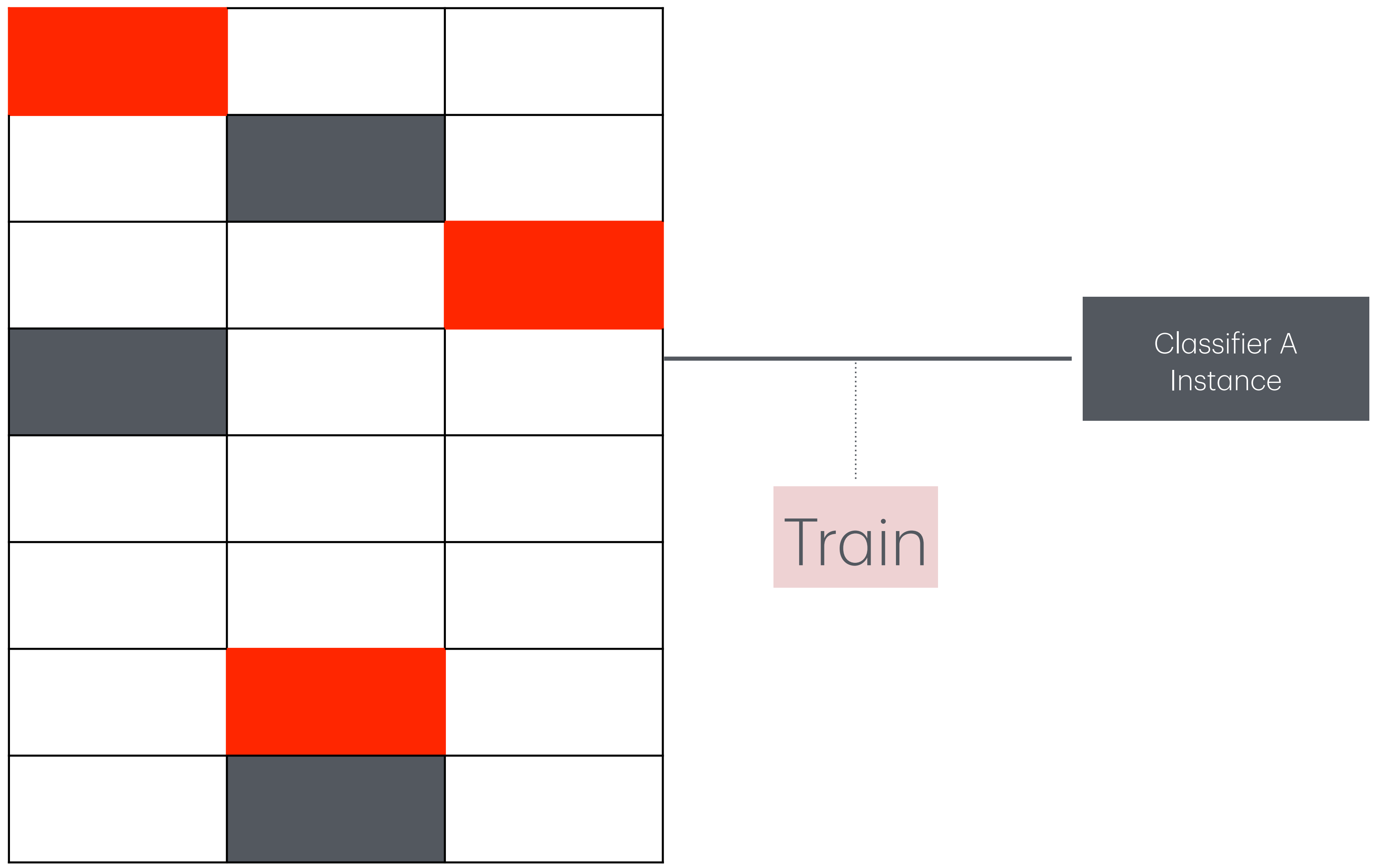
Random Subspaces (Pasting)

Classifier A

Sampling
- Features



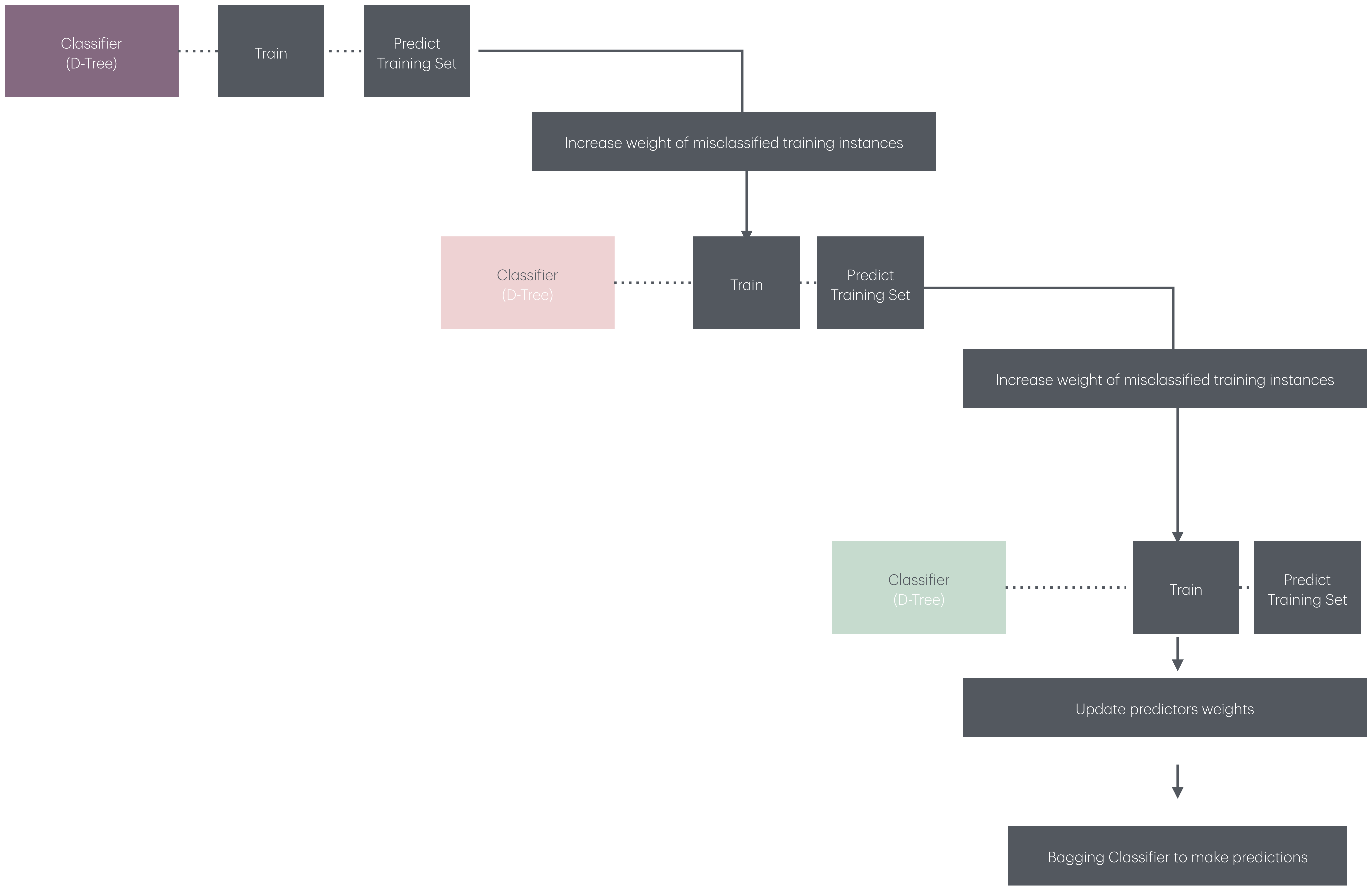
Sampling without replacement. Indices in red are not able to be fetched after usage



Feature Importance

Quick method to find feature importance

Boosting AdaBoost



AdaBoost Weights

$$w_{all} = \frac{1}{m}$$

X	Y	W
		w_r11
		w_r12
		w_r13
		w_r14

Classifier

Train

Predict Training Set

X	Y_	Y	W
			w_r11
			w_r12
			w_r13
			w_r14

$$r_1 = \frac{\sum_{i=1}^{m} w^i \cdot (Y = Y\_)}{\sum_{i=1}^{m} w^i}$$

$$r_1 = \frac{w^{i=2} + w^{i=4}}{w^{i=1} + w^{i=2} + w^{i=3} + w^{i=4}}$$

Calculate Predictor Weighted Error (r_1)

r_1

Predictor Weight $\alpha(r_1)$

α

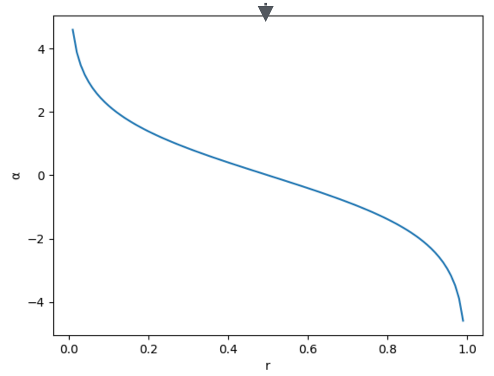
X	Y_	Y	W
			w_r11
			w_r12 * exp(α)
			w_r13
			w_r14 * exp(α)

Weight update (Only the misclassified)

Normalize weights

$$\frac{w^i}{\sum_{i=1}^{m}}$$

X	Y_	Y	W
			w_r11
			w_r12
			w_r13
			w_r14



X	Y	W
		w_r11
		w_r12
		w_r13
		w_r14

Classifier

Train

Predict Training Set

X	Y_	Y	W
			w_r11
			w_r12
			w_r13
			w_r14

$$r_2 = \frac{\sum_{i=1}^{m} w^i \cdot (Y = Y\_)}{\sum_{i=1}^{m} w^i}$$

$$r_1 = \frac{w^{i=2}}{w^{i=1} + w^{i=2} + w^{i=3} + w^{i=4}}$$

Calculate Predictor Weighted Error (r_2)

r_1

Predictor Weight $\alpha(r_2)$

α

X	Y_	Y	W
			w_r11
			w_r12 * exp(α)
			w_r13
			w_r14

Weight update (Only the misclassified)

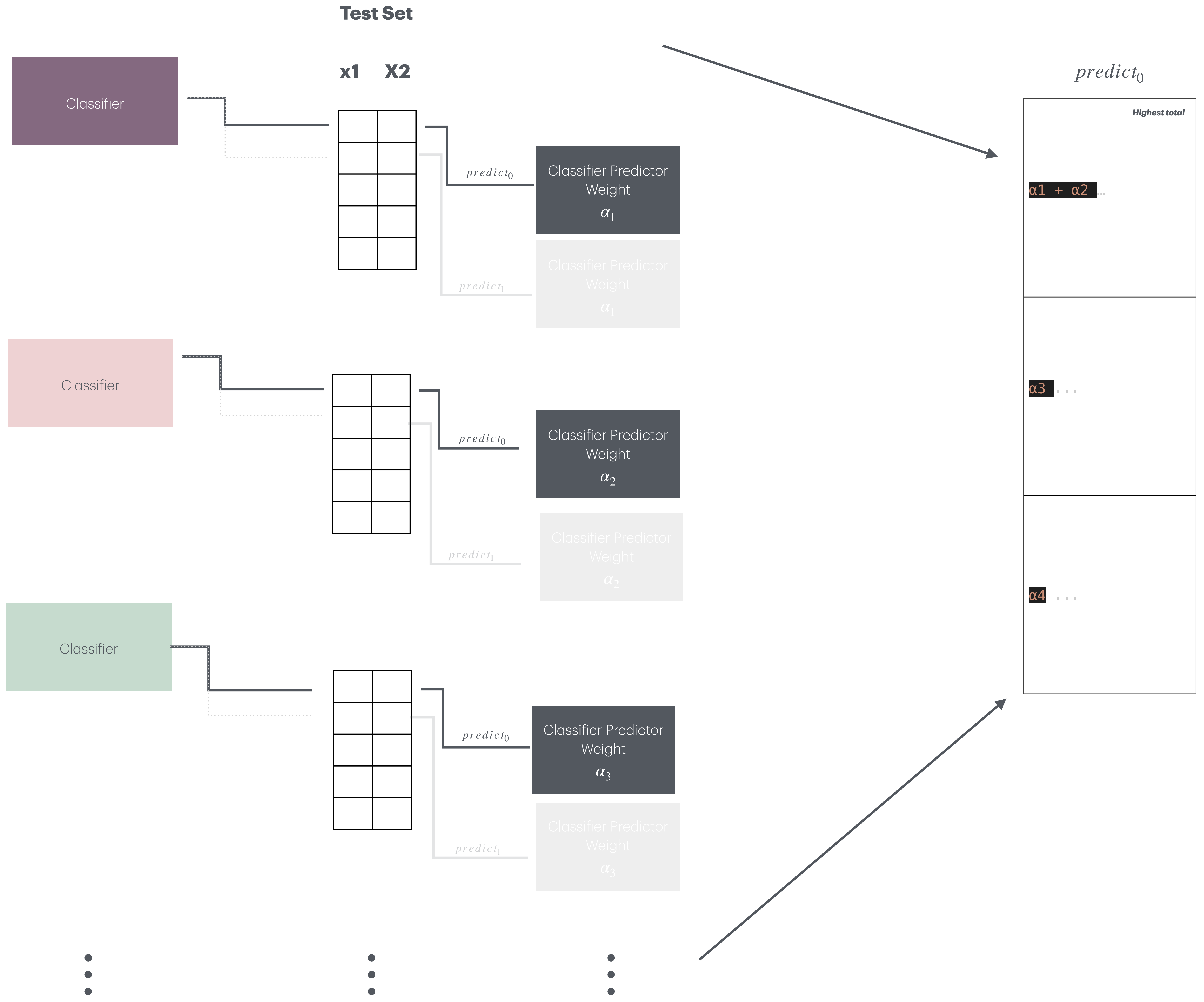
Normalize weights

$$\frac{w^i}{\sum_{i=1}^{m}}$$

X	Y_	Y	W
			w_r11
			w_r12
			w_r13
			w_r14

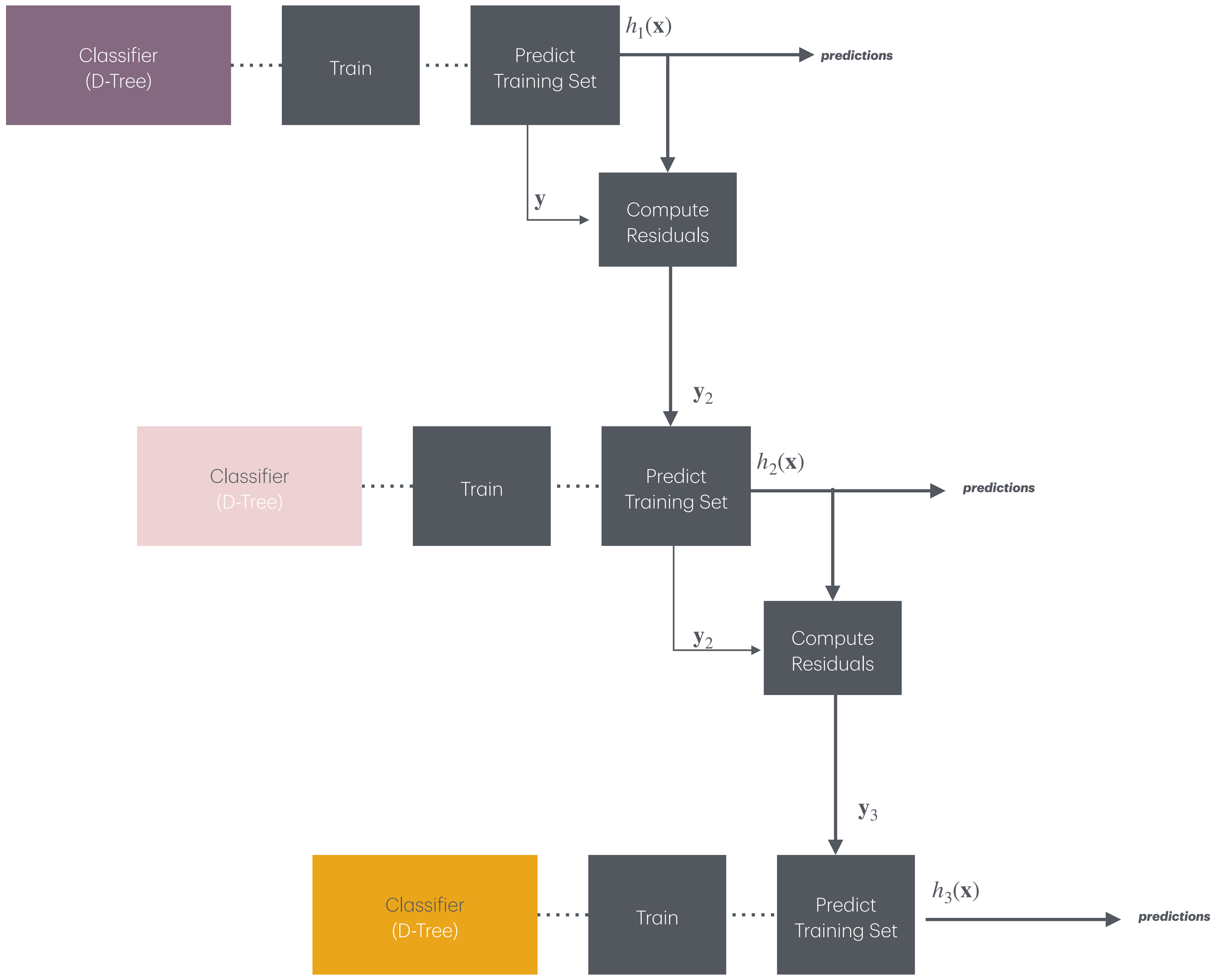
⋮

Predictions AdaBoost



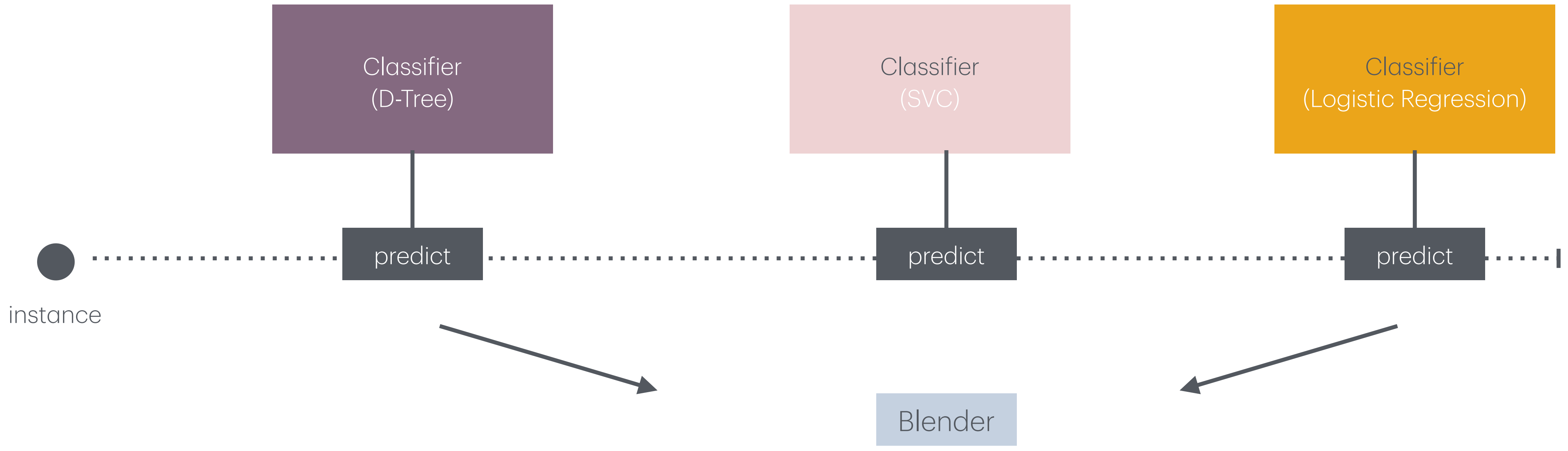
Contrived example predicts class = 0 since it has the highest weight votes

Gradient Boosting AdaBoost

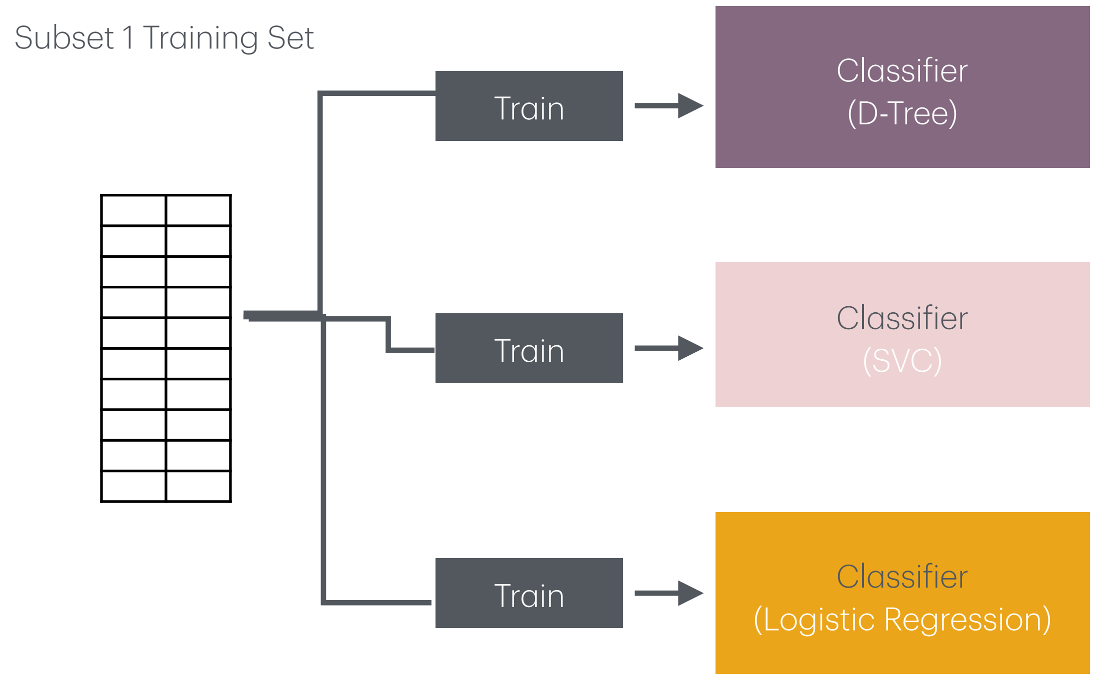


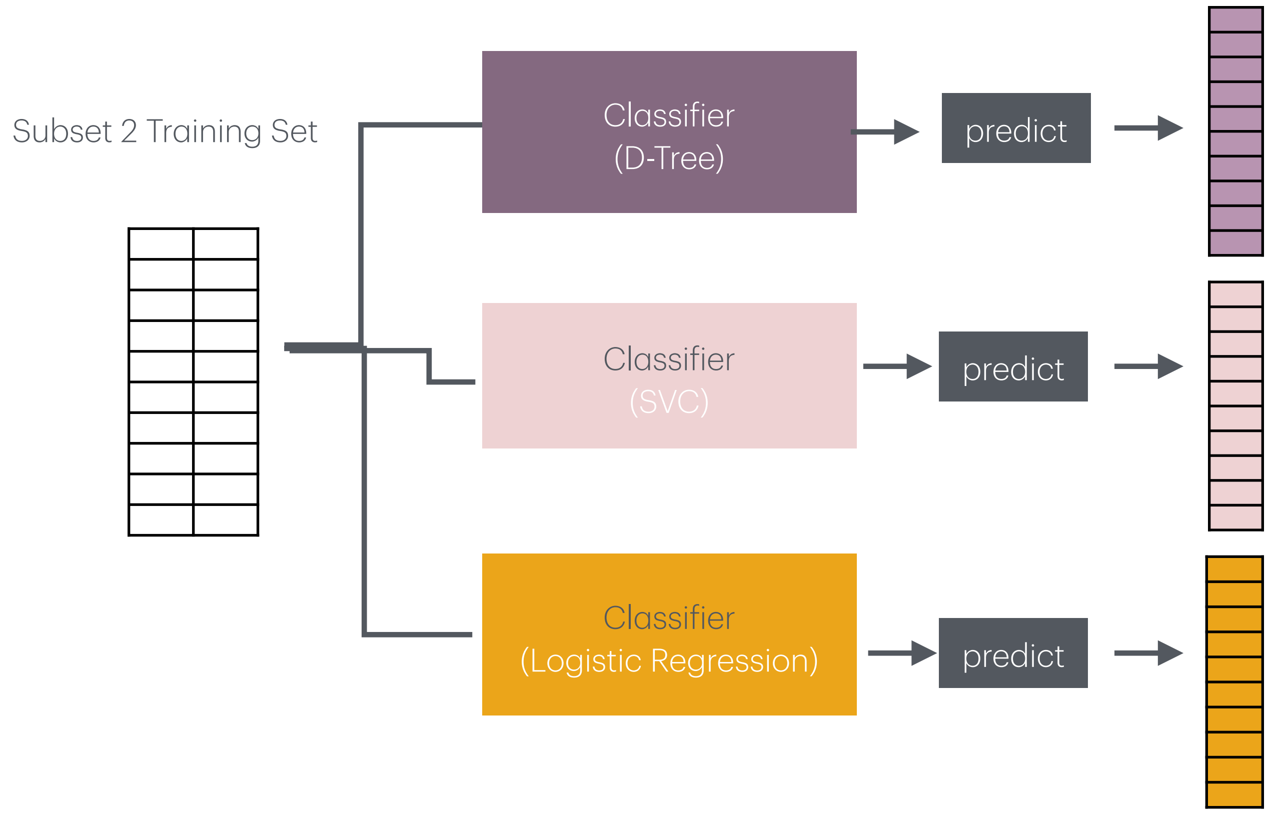
Stacking

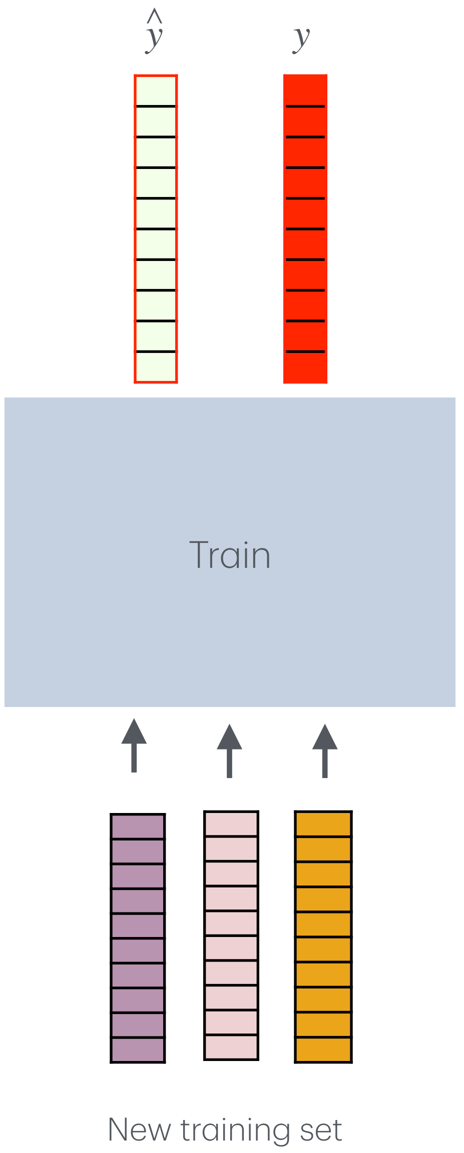
Train a model to perform aggregation



Layer 1:
Models trained on Subset 1







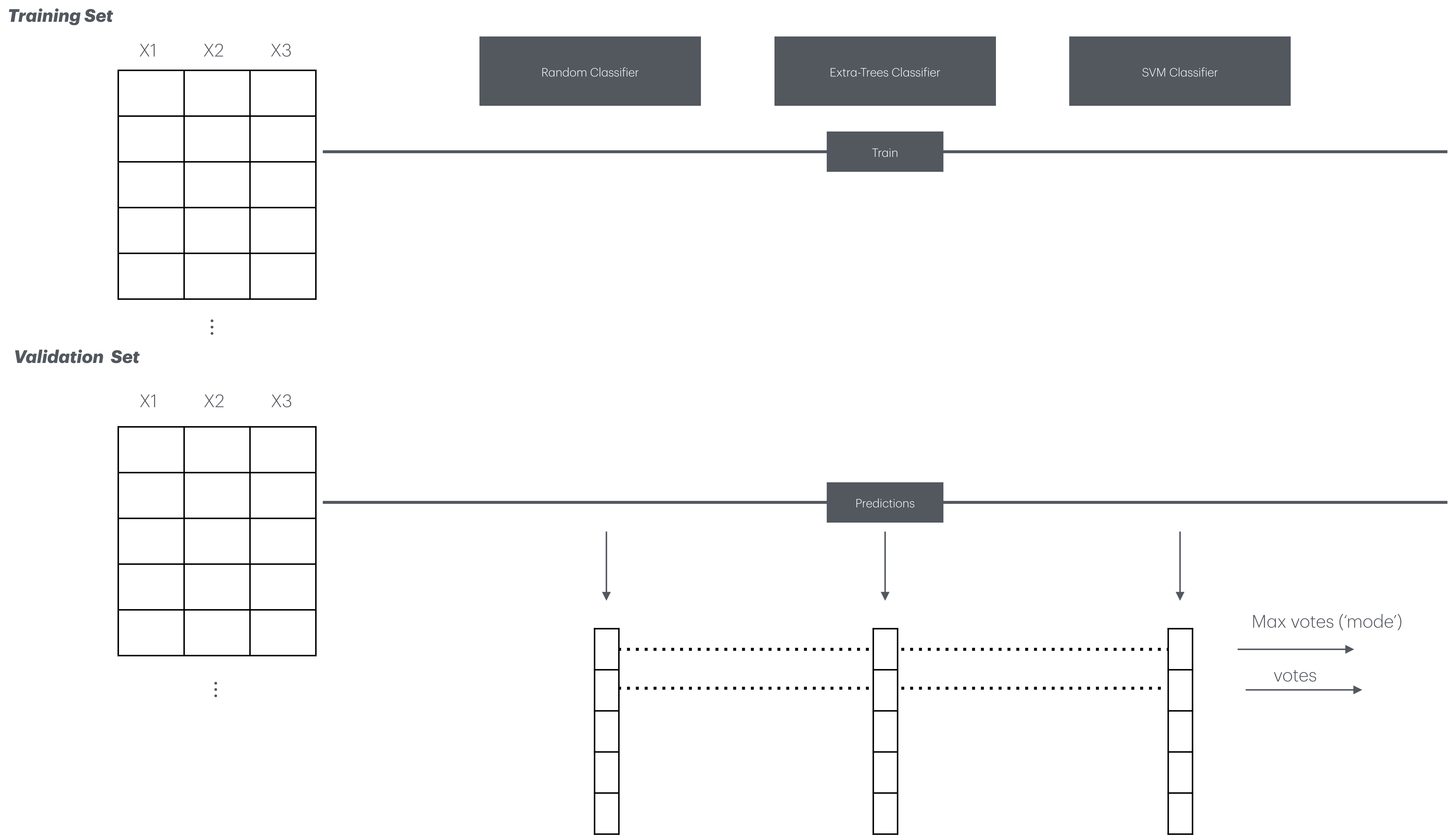
Layer 2:

Models trained on Subset 2 predictions

Blender learns to predict target values given first layer predictions

Learning from the aggregate rather than learning from scratch

Example 8 : Hard Voting



Example 8 : Soft Voting