Recap – AdaBoost

Rina BUOY

AMERICAN UNIVERSITY OF PHNOM PENH

STUDY LOCALLY. LIVE GLOBALLY.

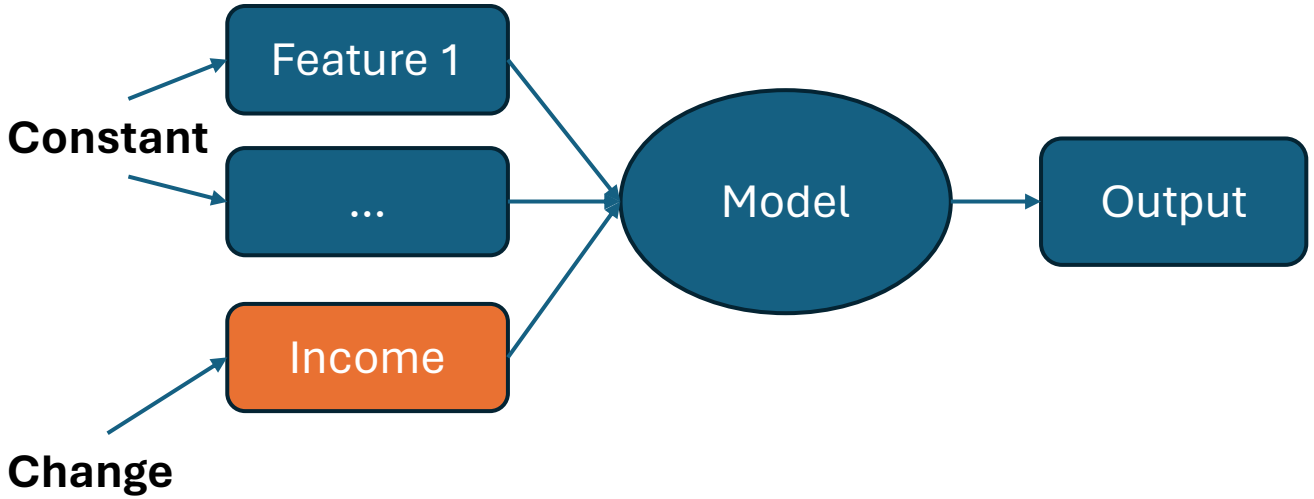
Feature Importance

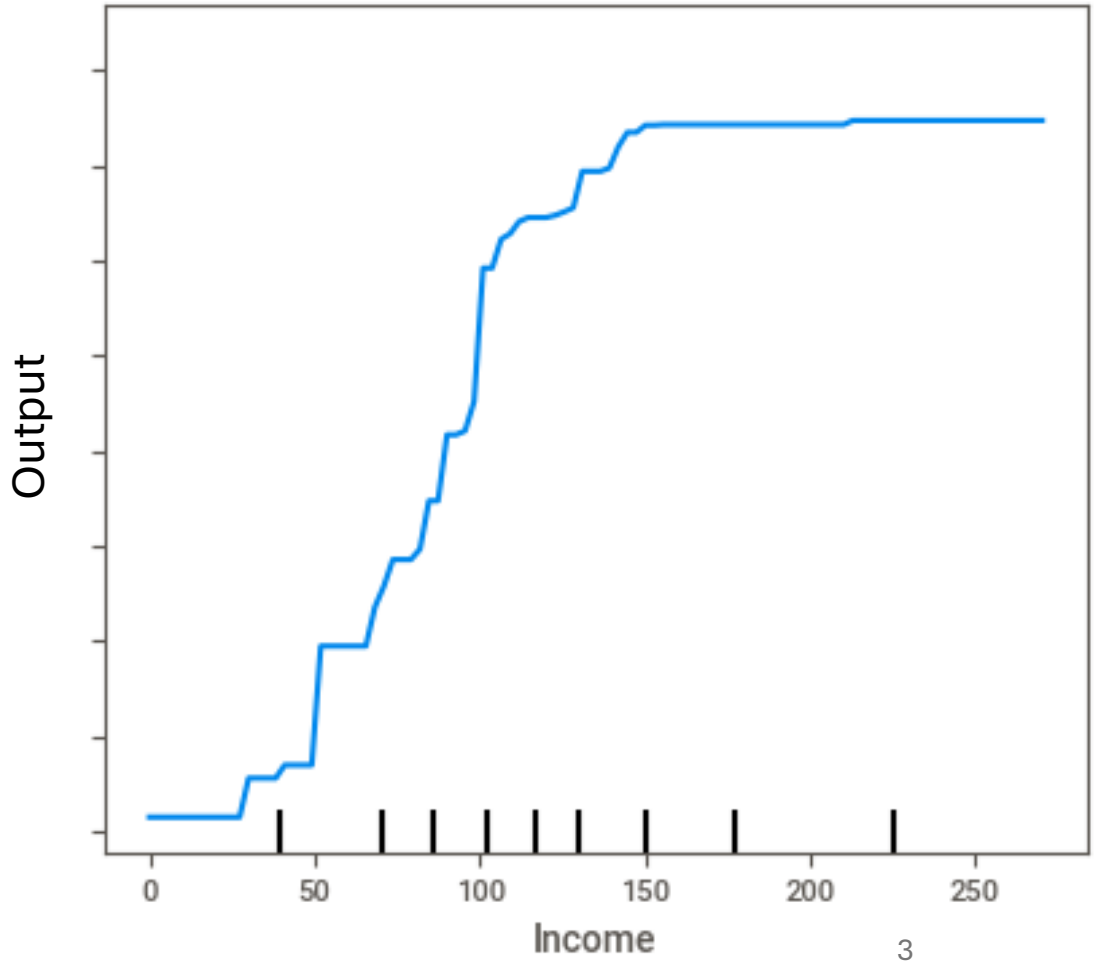
- A Partial Dependence Plot (PDP) is a graphical representation of the marginal effect of a feature on the predicted outcome of a machine learning model.

- It shows how the prediction changes as the value of a single feature varies while keeping other features constant.

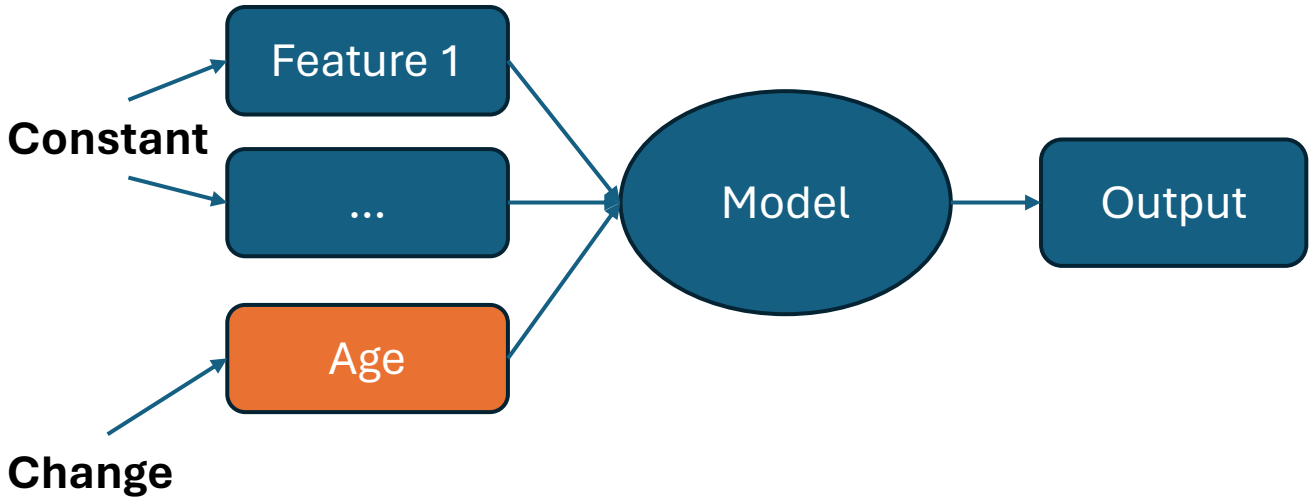
- PDPs are commonly used to interpret the relationship between individual features and the target variable in black-box models such as ensemble methods and deep learning models.

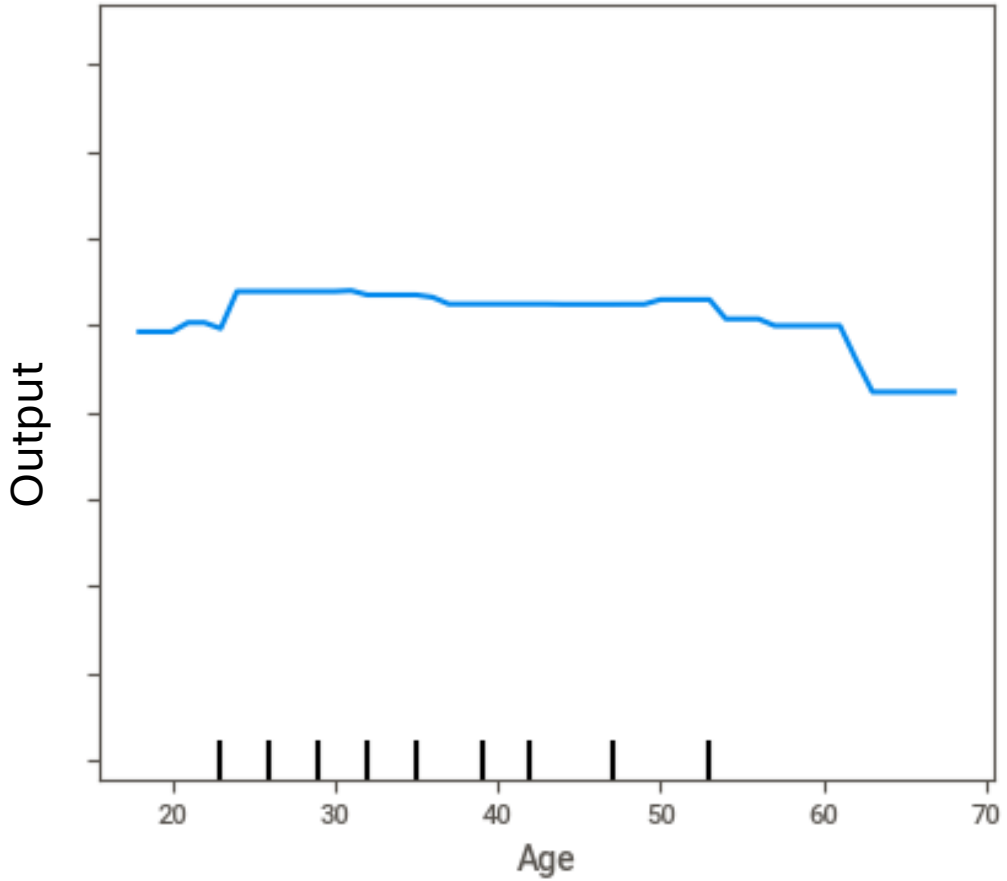
Partial Dependencies Plot





Partial Dependencies Plot





Which one is more important ?
Instead of visualizing, can we do this algorithmically ?

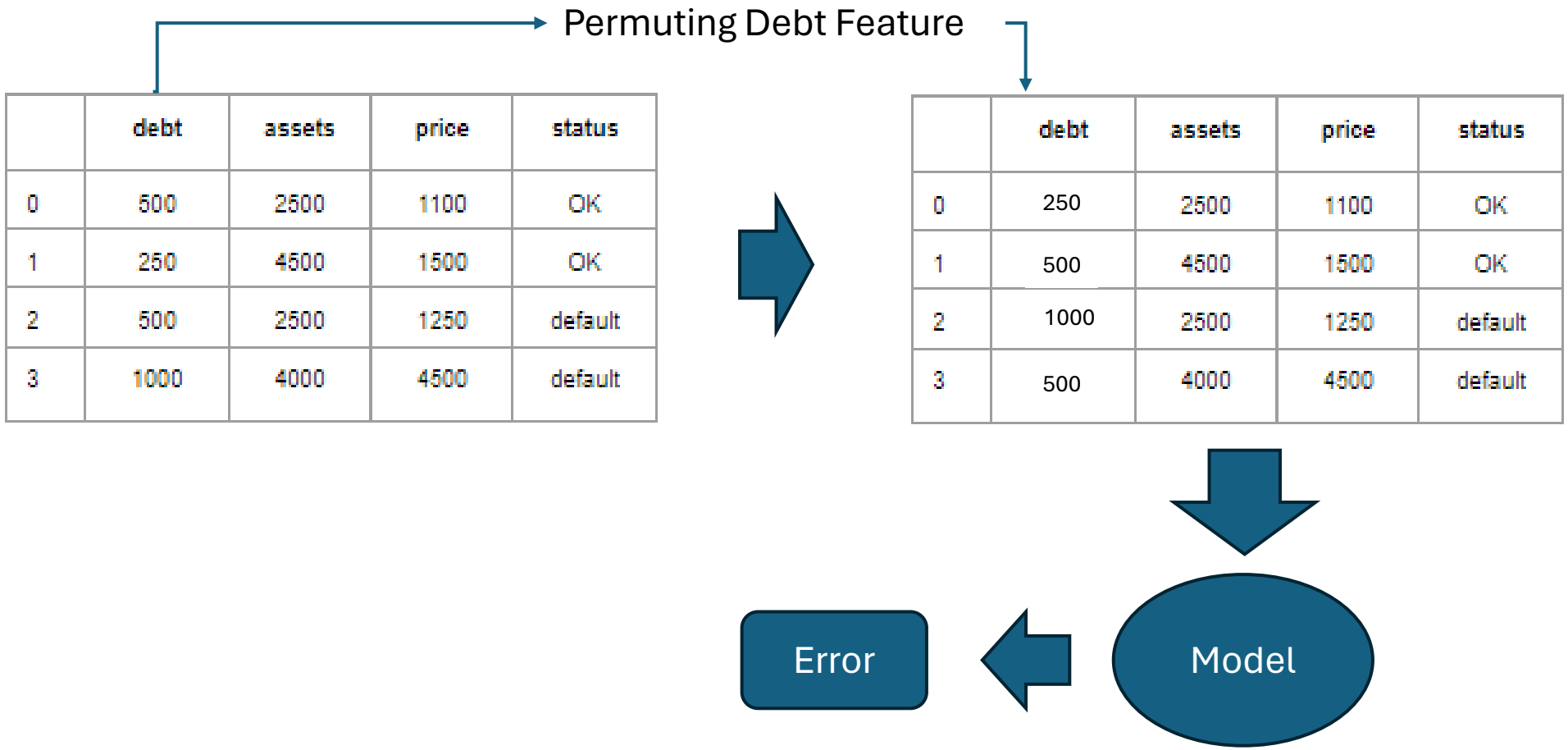
Permutation Feature Importance

- Permutation feature importance is a method used to measure the importance of each feature in a machine learning model.

- It evaluates the impact of shuffling (permuting) the values of each feature on the model's performance.

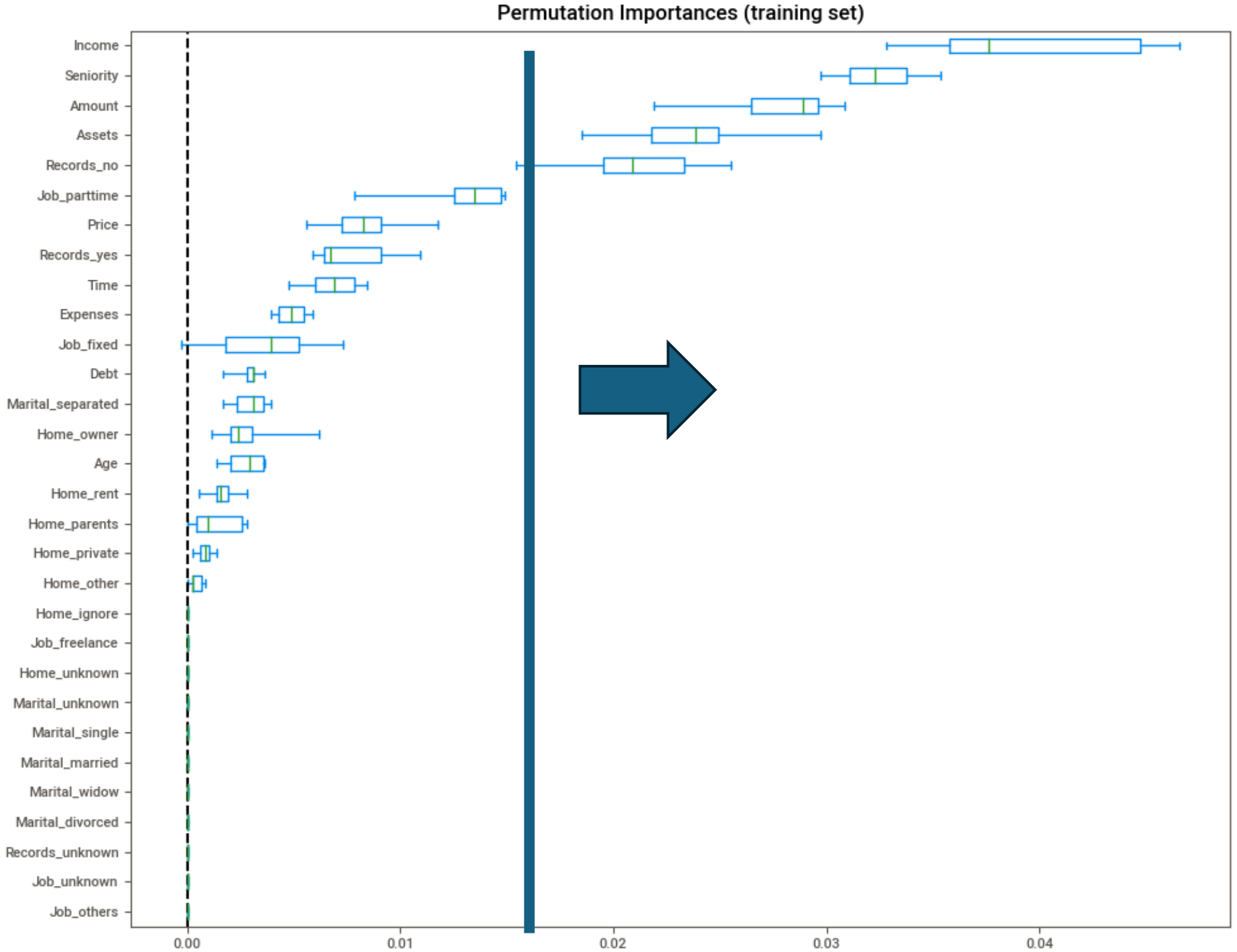
- The basic idea is to assess how much the model's performance decreases when the values of a particular feature are randomly shuffled, compared to the original feature values.

Permutation Feature Importance



	debt	assets	price	status
0	500	2500	1100	OK
1	250	4500	1500	OK
2	500	2500	1250	default
3	1000	4000	4500	default

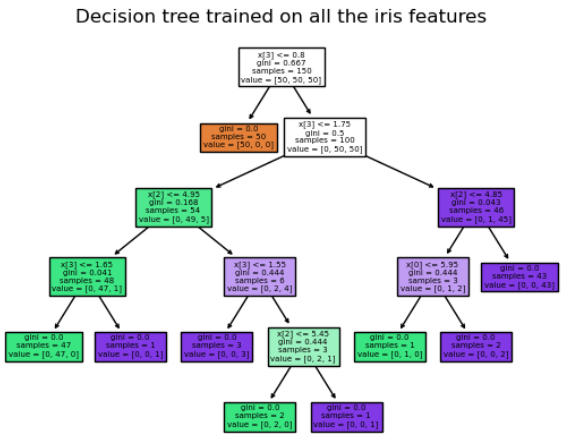
	debt	assets	price	status
0	250	2500	1100	OK
1	500	4500	1500	OK
2	1000	2500	1250	default
3	500	4000	4500	default


Permutation Importances (training set)
Income
Seniority
Amount
Assets
Records_no
Job_parttime
Price
Records_yes
Time
Expenses
Job_fixed
Debt
Marital_separated
Home_owner
Age
Home_rent
Home_parents
Home_private
Home_other
Home_ignore
Job_freelance
Home_unknown
Marital_unknown
Marital_single
Marital_married
Marital_widow
Marital_divorced
Records_unknown
Job_unknown
Job_others
0.00
0.01
0.02
0.03
0.04

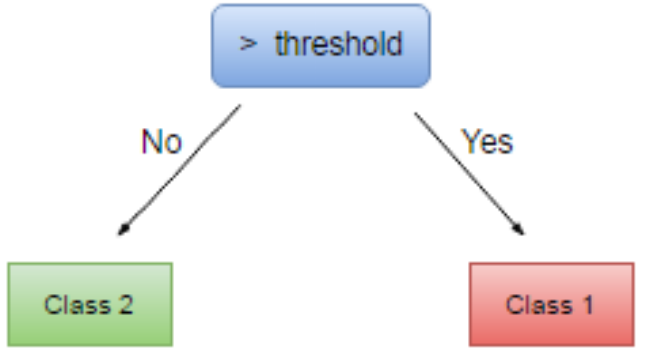
Boosting

- Boosting refers to a family of machine learning meta-algorithms which combine the outputs of many “weak” classifiers into a powerful “committee”.
- Each of the weak classifiers alone may have an error rate which is only slightly better than random guessing.

A deep decision tree



A decision stump (i.e., weak classifier)

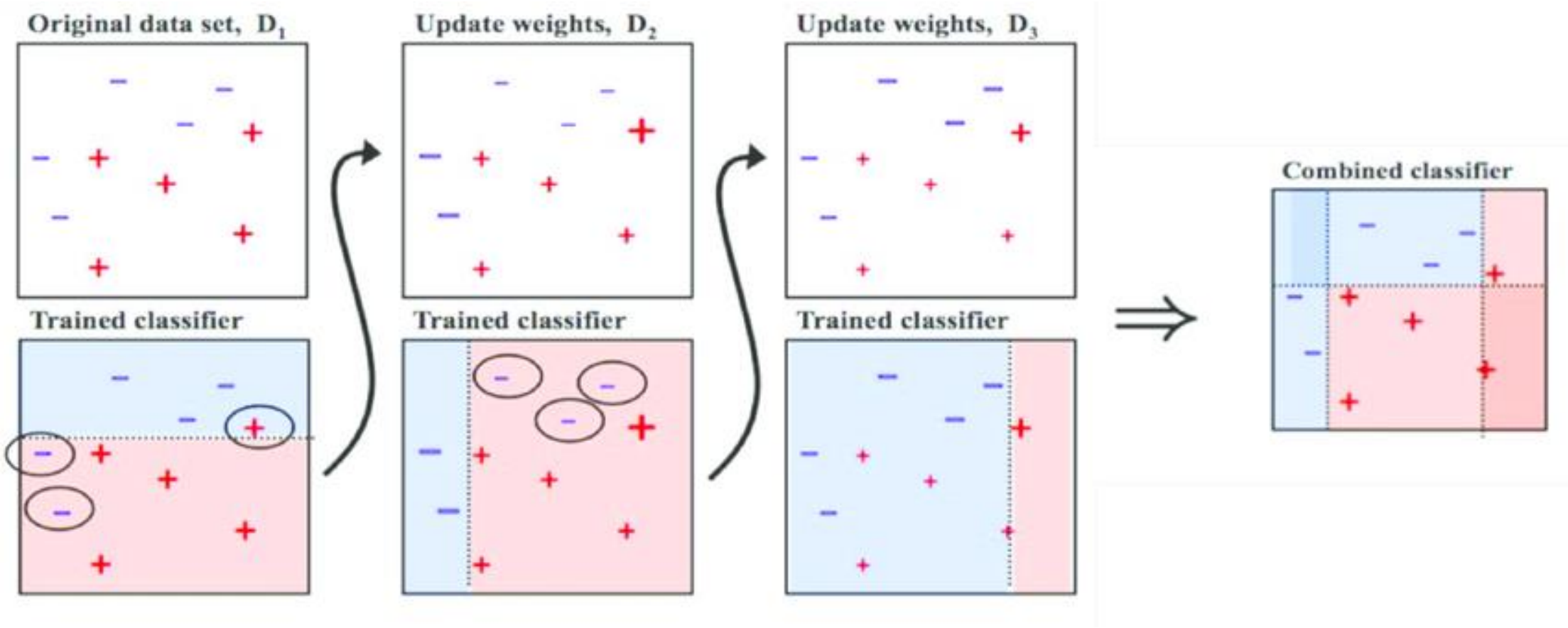


Adaptive Boosting

- AdaBoost (Adaptive Boosting) is a particular boosting algorithm in which we fit a sequence of “stumps” (decision trees with a single node and two leaves) and weight their contribution to the final vote by how accurate their predictions are.

- After each iteration, we re-weight the dataset to assign greater importance to data points which were misclassified by the previous weak learner, so that those data points get “special attention” during iteration

Adaptive Boosting



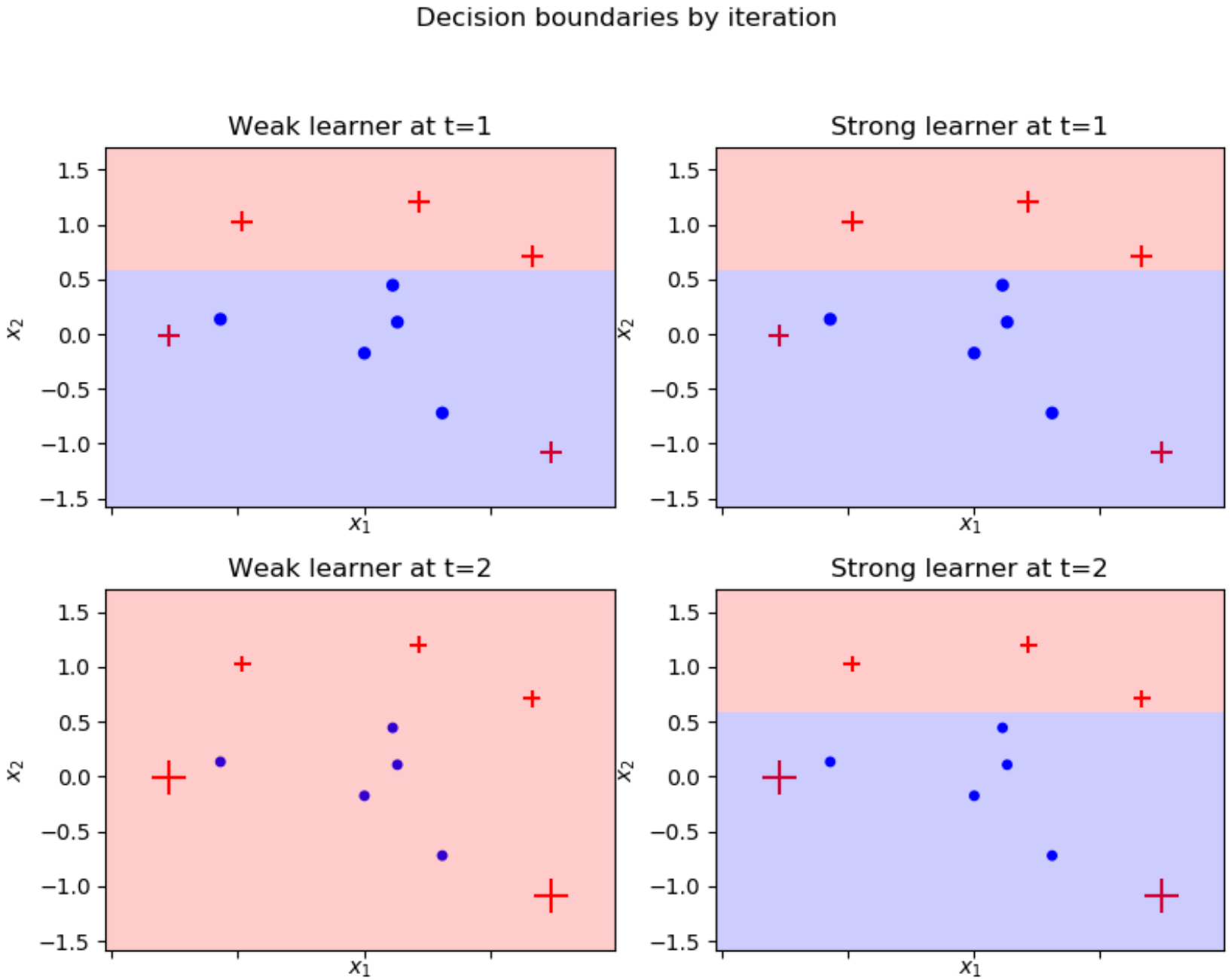
Adaptive Boosting

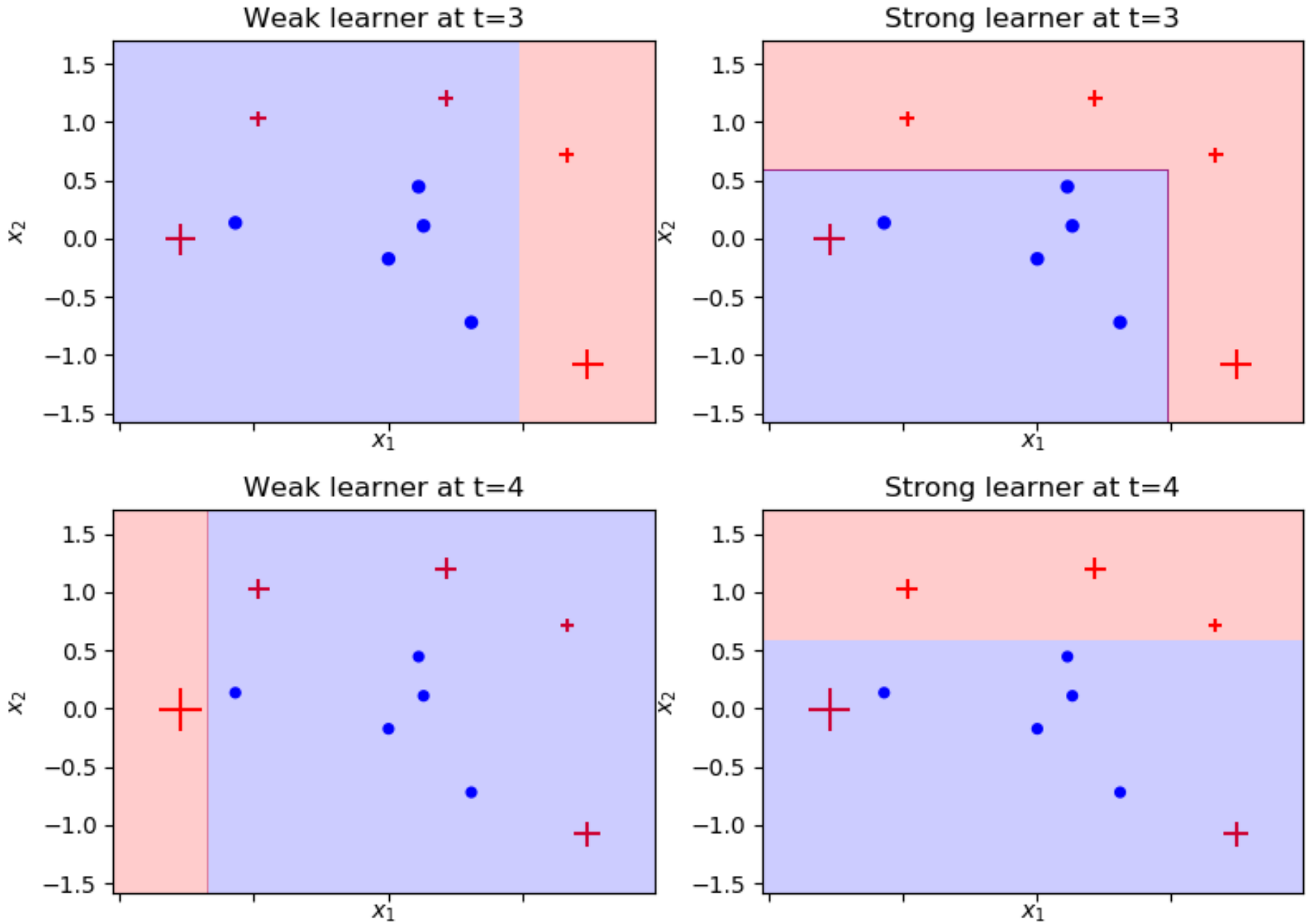
A) Initialize sample weights uniformly as $w_i^1 = \frac{1}{n}$.

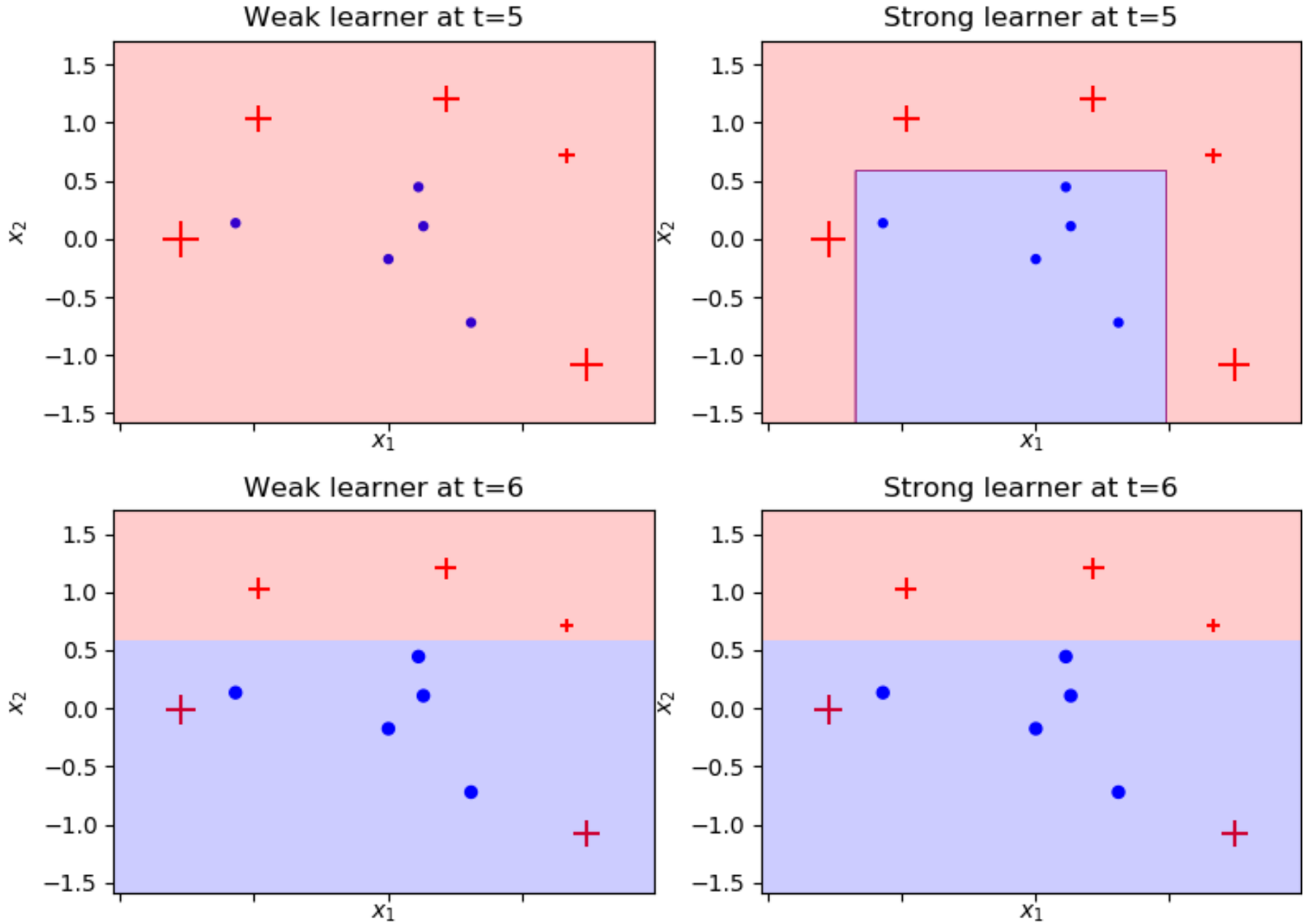
B) For each iteration t:

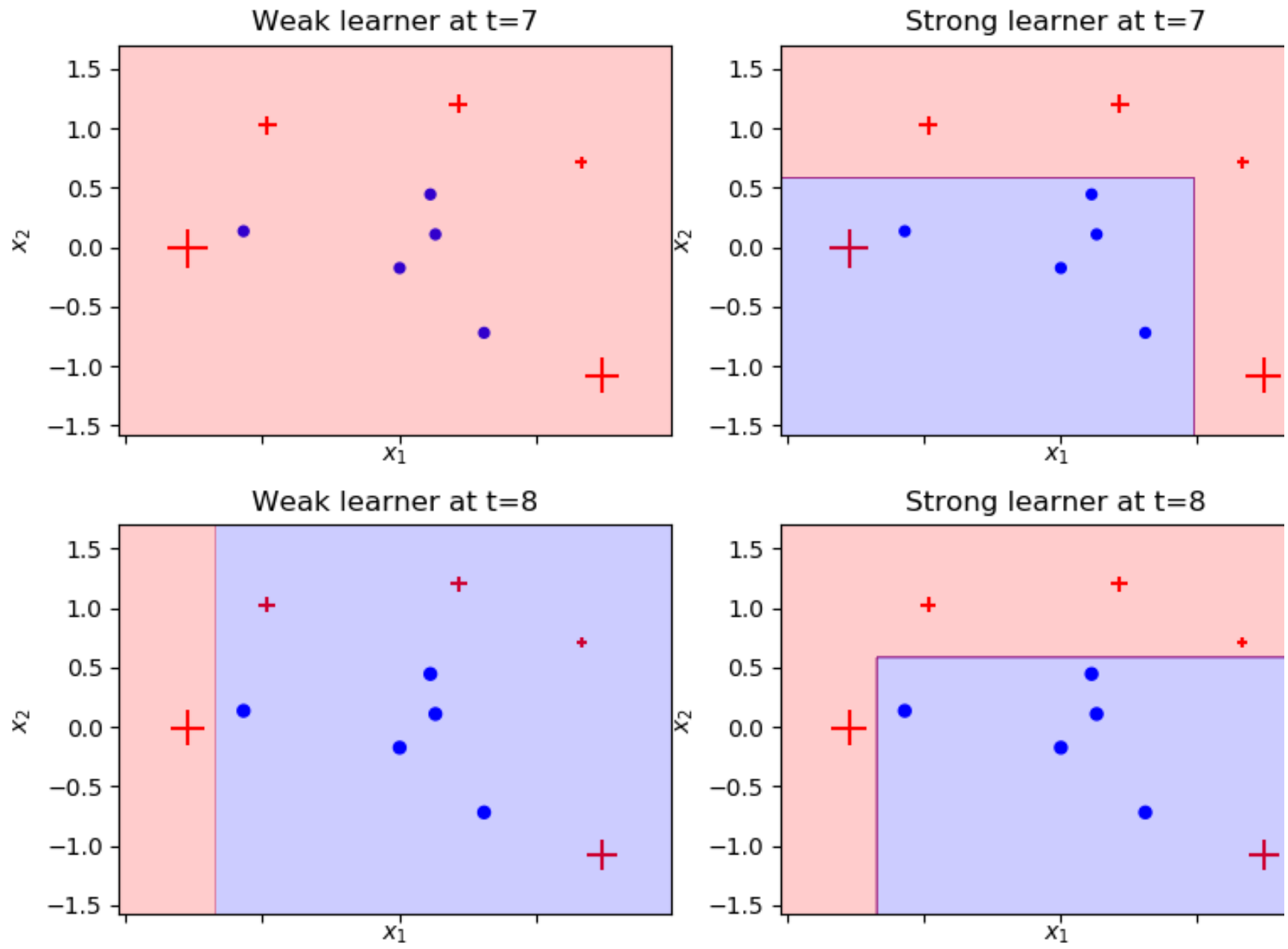
1. Find weak learner $h_t(x)$ which minimizes $\epsilon_t = \sum_{i=1}^{n} \mathbf{1}[h_t(x_i) \neq y_i]\, w_i^{(t)}$.
2. We set a weight for our weak learner based on its accuracy: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
3. Increase weights of misclassified observations: $w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\alpha^t y_i h_t(x_i)}$.
4. Renormalize weights, so that $\sum_{i=1}^{n} w_i^{(t+1)} = 1$.

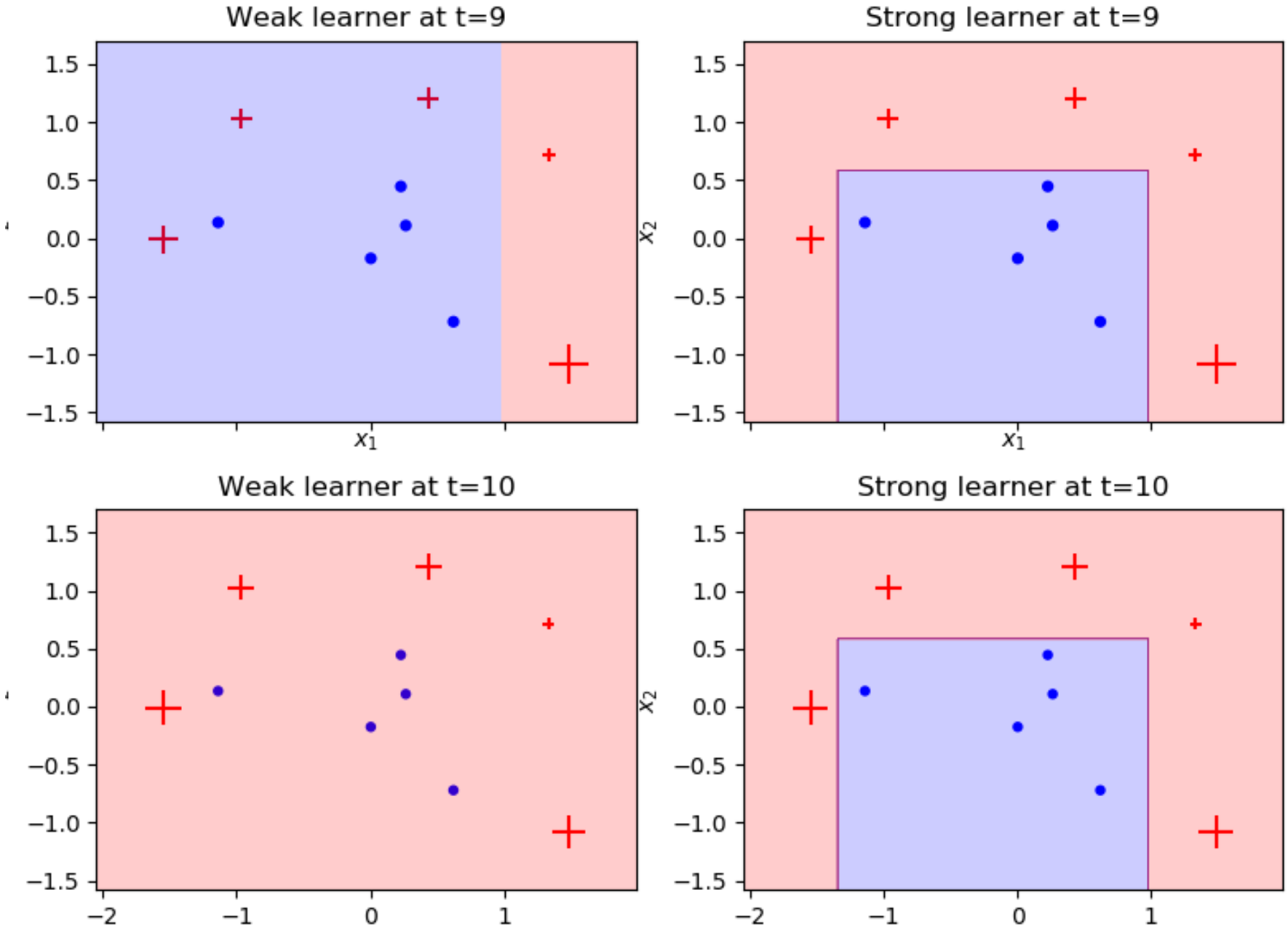
C) Make final prediction as weighted majority vote of weak learner predictions:
$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.


Decision boundaries by iteration
Weak learner at t=1
Strong learner at t=1
Weak learner at t=2
Strong learner at t=2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=3
Strong learner at t=3
Weak learner at t=4
Strong learner at t=4
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=5
Strong learner at t=5
Weak learner at t=6
Strong learner at t=6
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=7
Strong learner at t=7
Weak learner at t=8
Strong learner at t=8
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
x_1
x_2


Weak learner at t=9
Strong learner at t=9
Weak learner at t=10
Strong learner at t=10
x_1
x_2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1

How does it compare to Random Forest?

Property	Random Forest	AdaBoost
Depth	Unlimited (a full tree)	Stump (single node w/ 2 leaves)
Trees grown	Independently	Sequentially
Votes	Equal	Weighted