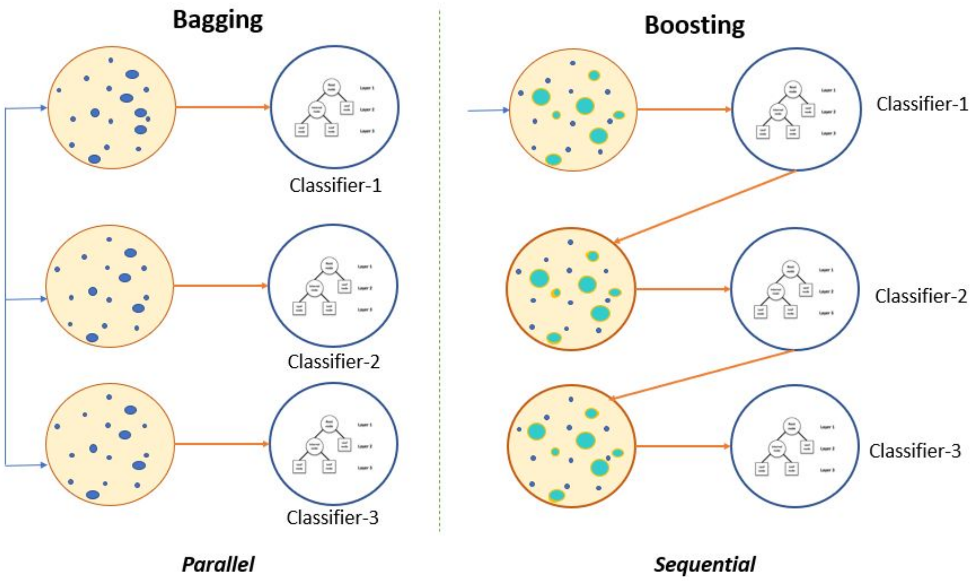
Random Forest for Classification and Regression

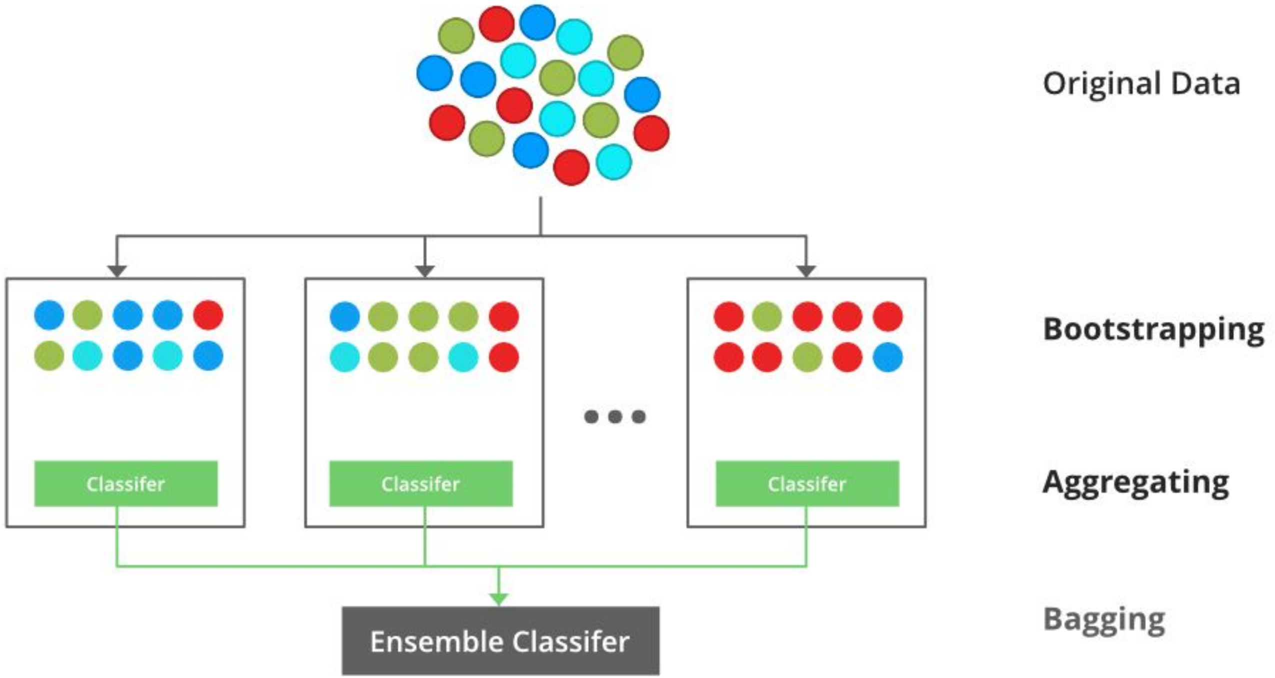
Jose Luis Paniagua Jaramillo
jlpaniagua@uao.edu.co

Ensemble Methods

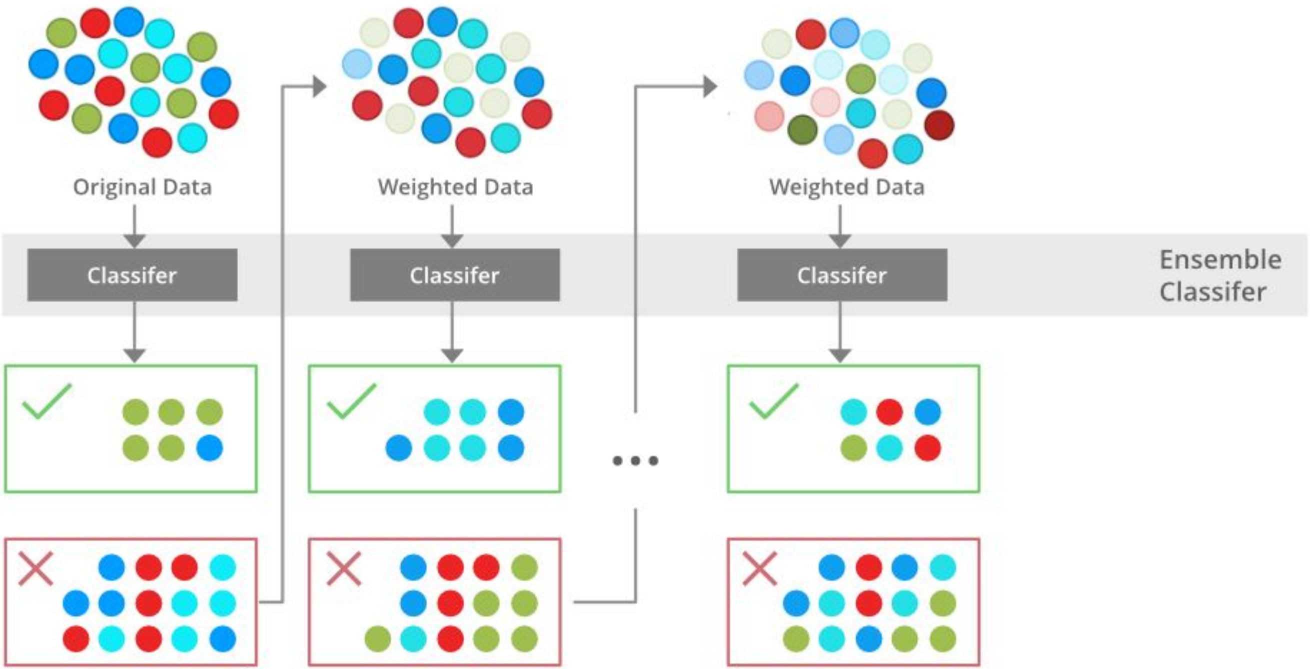
Ensembles are methods that combine multiple machine learning models to create more powerful models



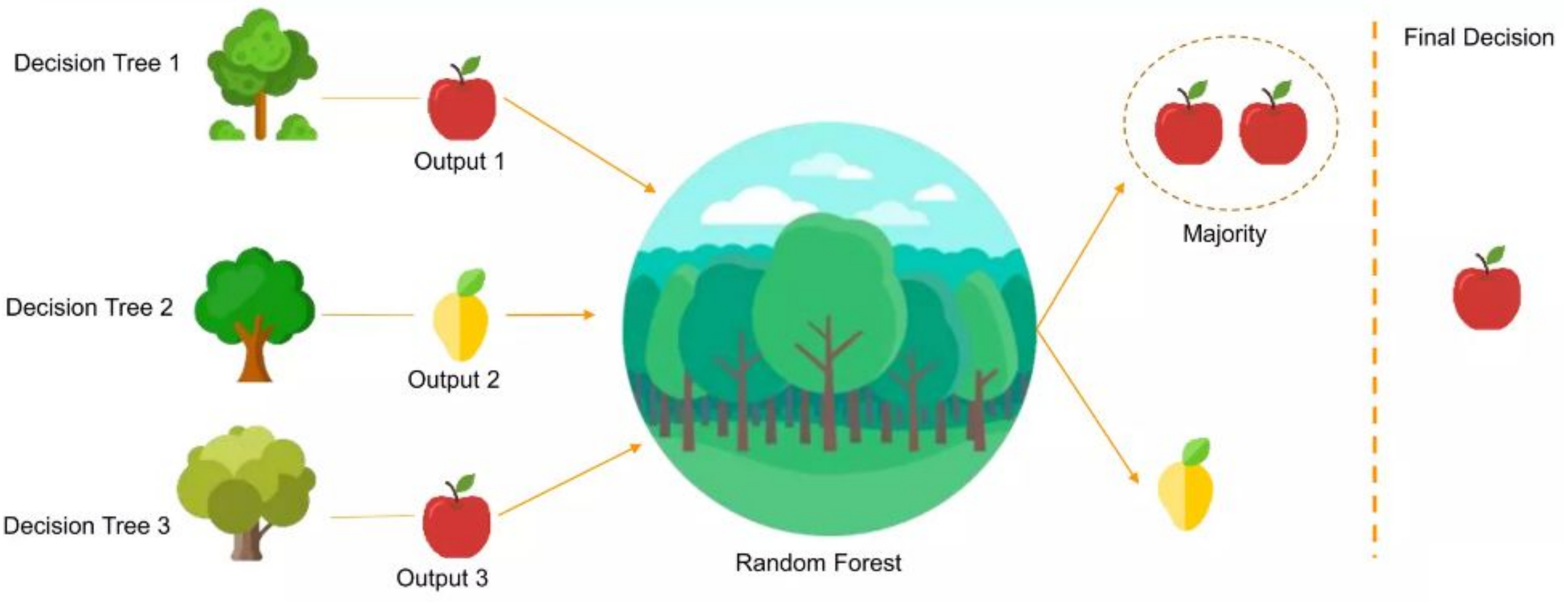
Ensemble Methods: Bagging



Ensemble Methods: Boosting



Random Forest



RF is a bagging or a boosting method?

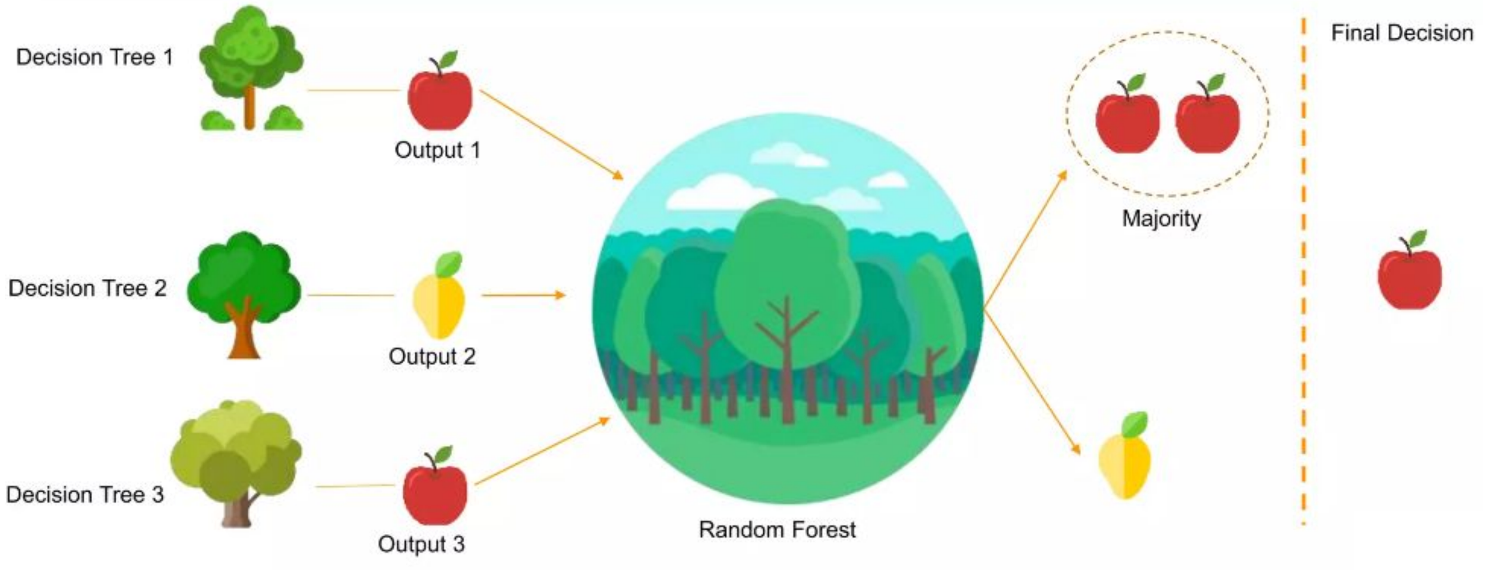
Why Random Forest?

No Overfitting

High Accuracy

Estimates
Missing Data

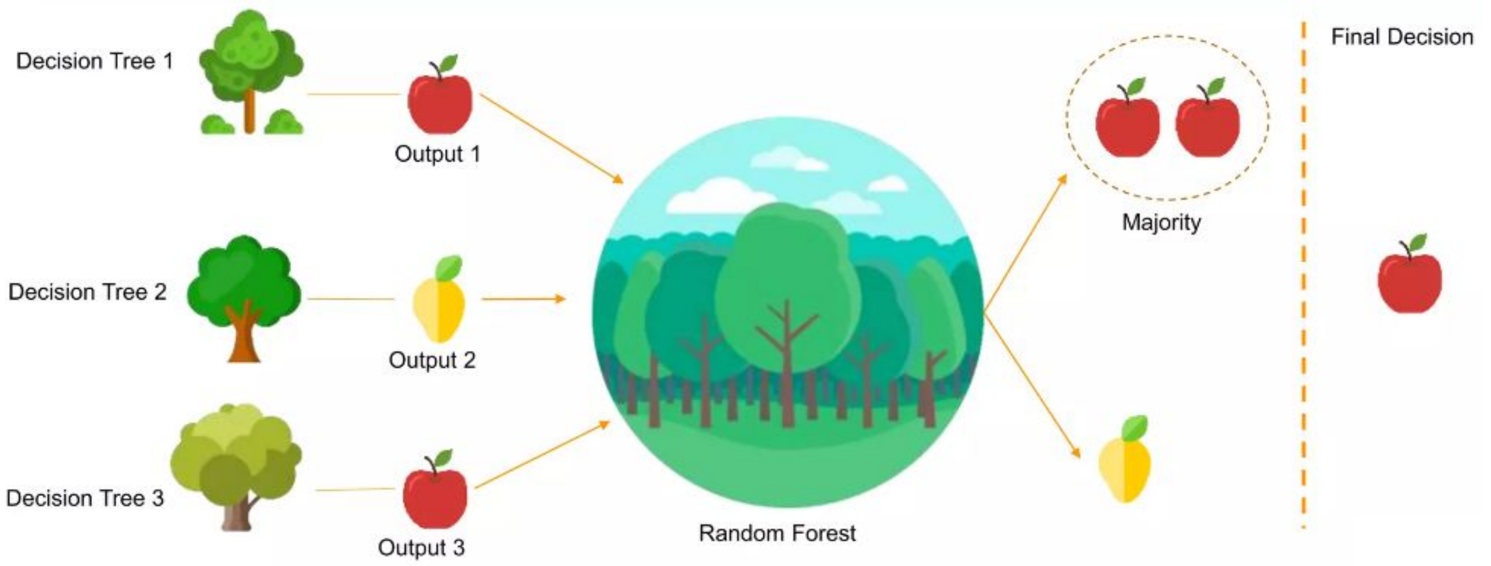
Random Forest



A RF is a collection of DTs, where each tree is slightly different from the others

- Each tree might do a good job of predicting , but **will likely overfit on part of the data**
- If **we build many trees** , all of which work well and **overfit in different ways**, we can **reduce the amount of overfitting by averaging their results.**

Random Forest



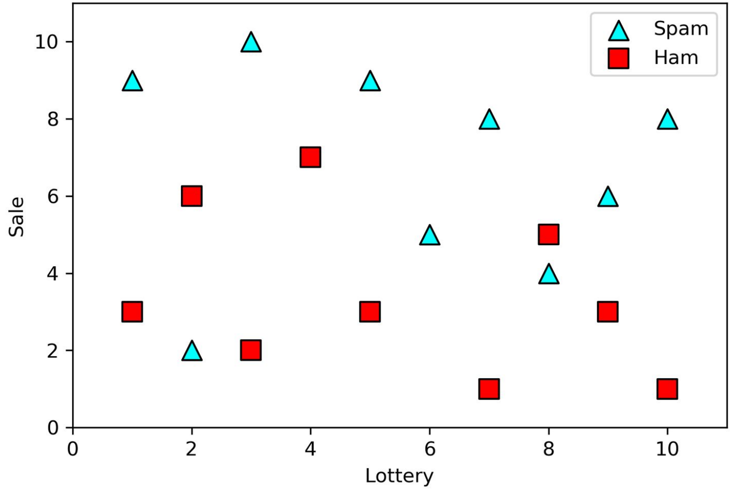
- We need to build many decision trees
  - Each tree should do and acceptable job of predicting the target
  - Should also be different from the other trees

- RFs get their name from injecting randomness into the tree building to ensure each tree is different.

Random Forest

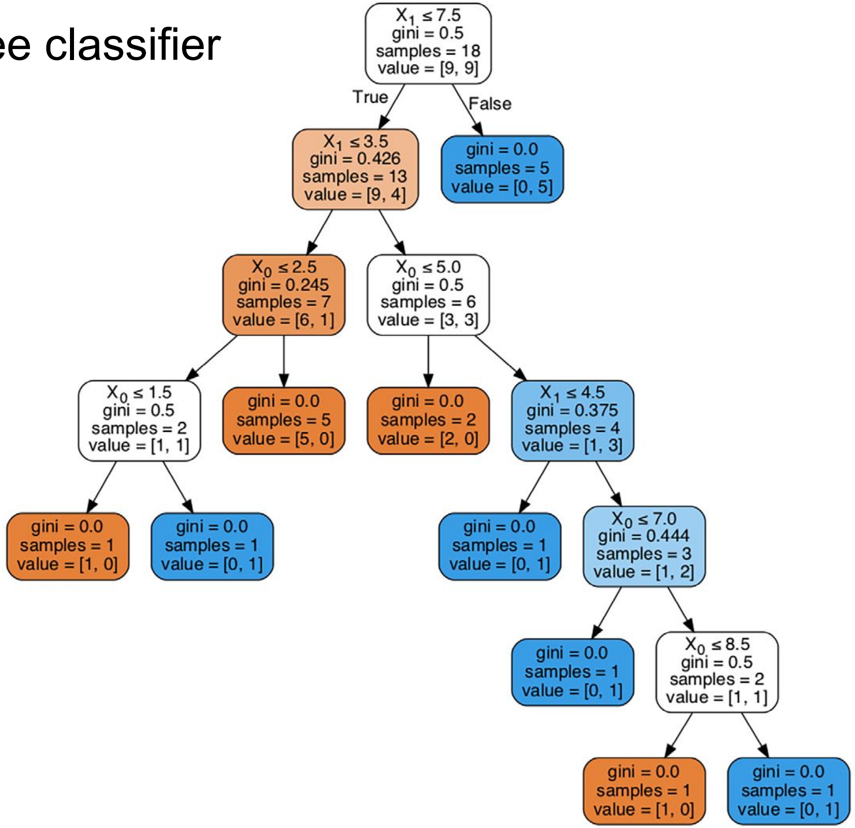
Example: spam and ham emails dataset

Lottery	Sale	Spam
7	8	1
3	2	0
8	4	1
2	6	0
6	5	1
9	6	1
8	5	0
7	1	0
1	9	1
4	7	0
1	3	0
3	10	1
2	2	1
9	3	0
5	3	0
10	1	0
5	9	1
10	8	1

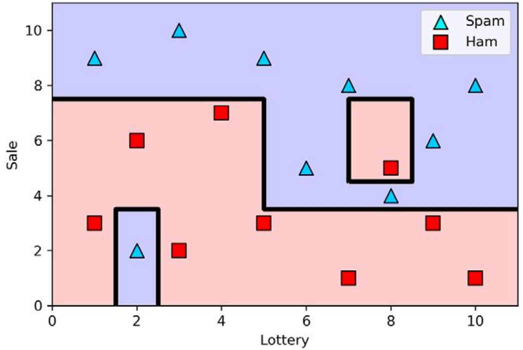


Random Forest

Decision tree classifier



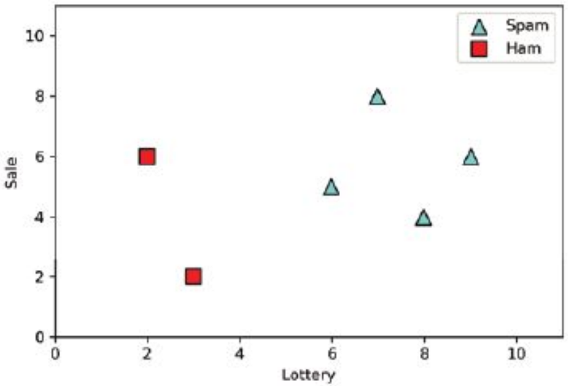
Overfitting?



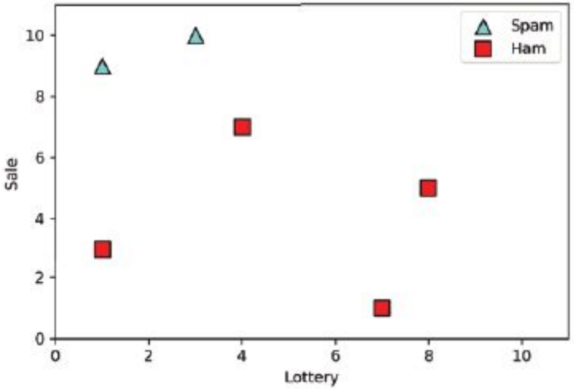
Random Forest

1. let's consider three subsets of 6 data points each

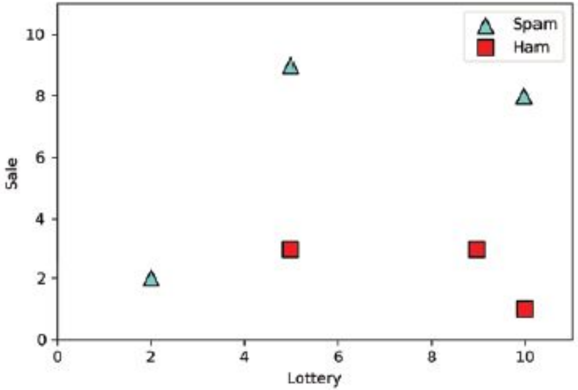
Subset 1



Subset 2

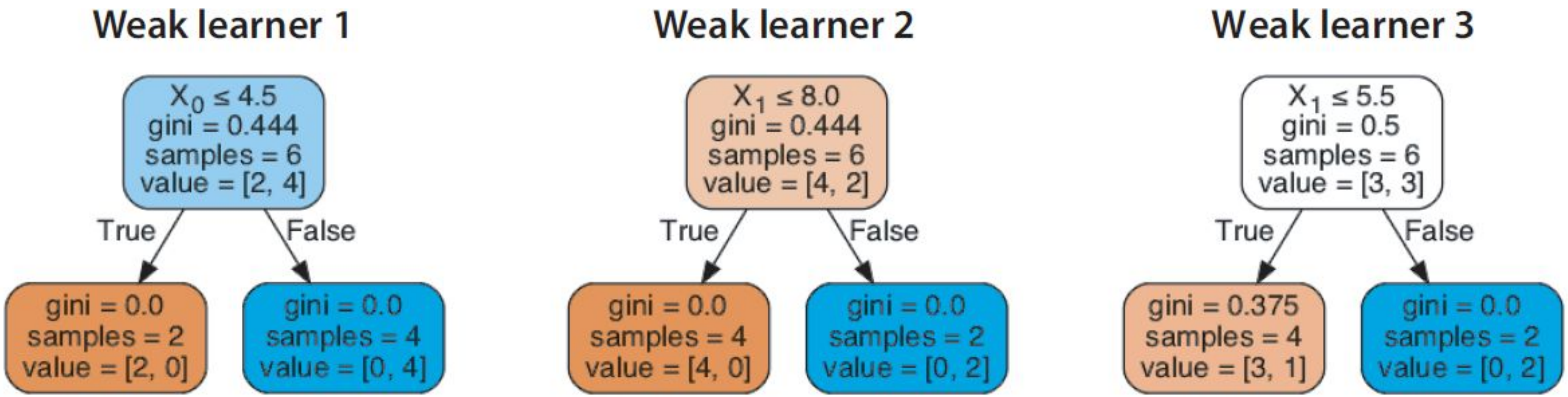


Subset 3



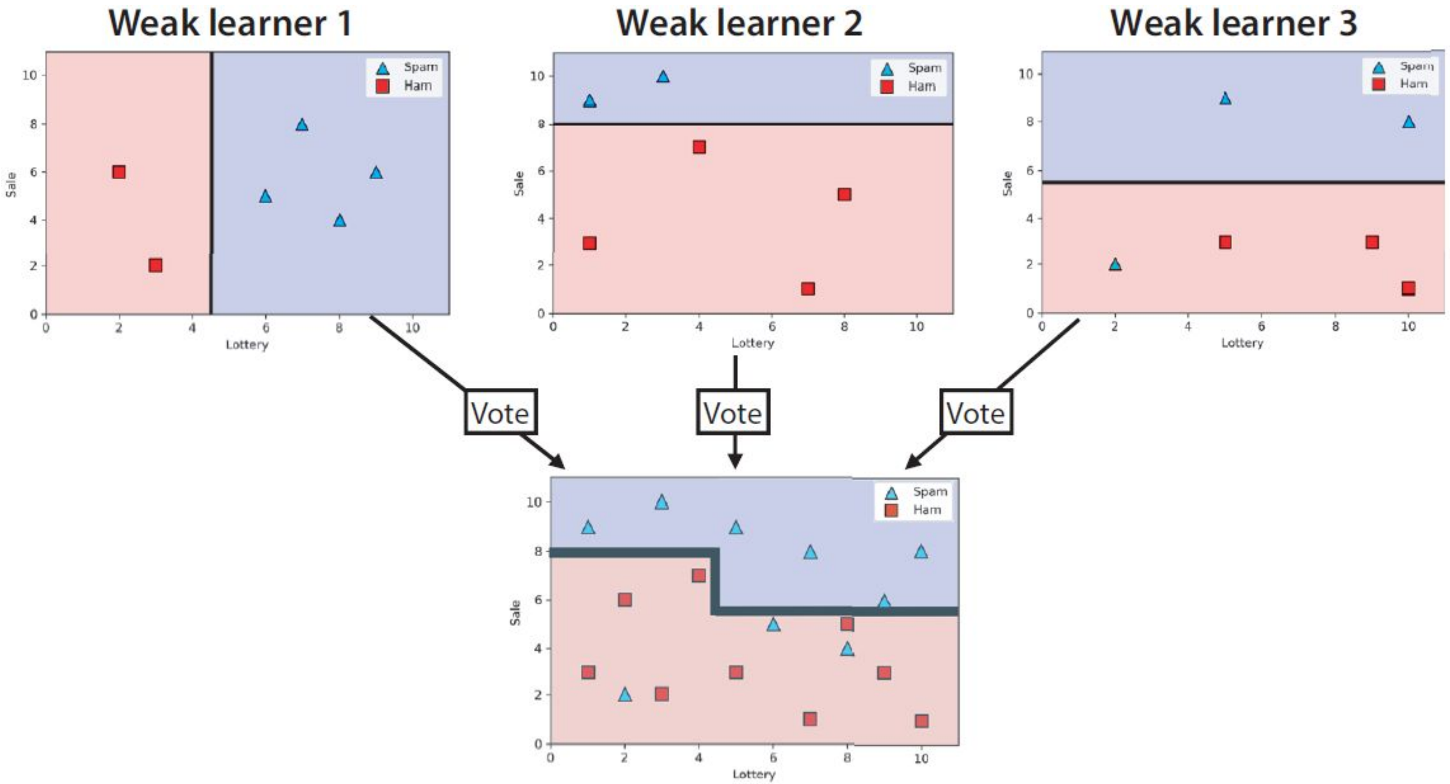
Random Forest

2. we proceed to build our three weak learners.

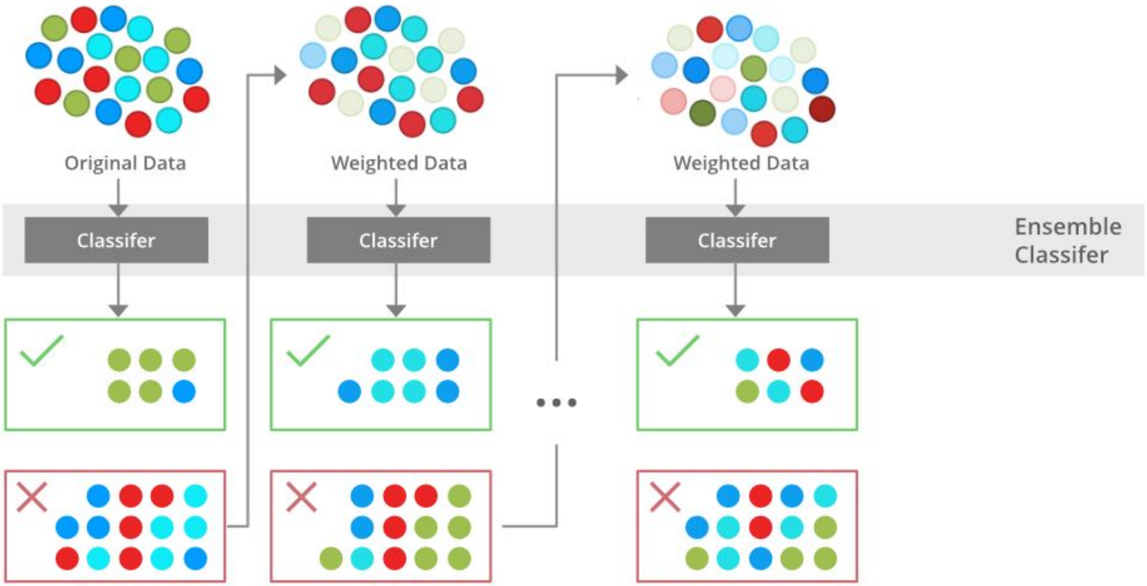


Random Forest

3. We combine these into a strong learner by voting.



Ensemble Methods: Boosting



- Boosting is similar to bagging in that we join several weak learners to build a strong learner.
- The difference is that we don't select the weak learners at random. Instead, each learner is built by focusing on the weaknesses of the previous learners.

AdaBoost (Adaptive Boosting)

- AdaBoost uses the complete training dataset to train the weak learners

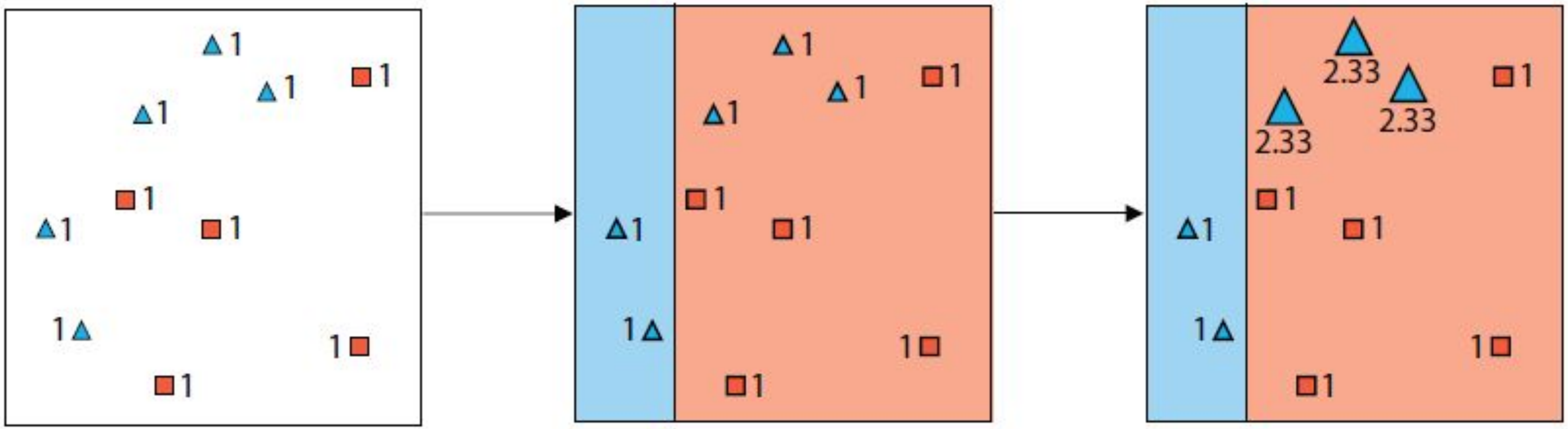
- The training examples are reweighted in each iteration to build a strong classifier that learns from the mistakes of the previous weak learners in the ensemble.

AdaBoost (Adaptive Boosting)

- AdaBoost uses the complete training dataset to train the weak learners.

- The training examples are reweighted in each iteration to build a strong classifier that learns from the mistakes of the previous weak learners in the ensemble.

AdaBoost (Adaptive Boosting)



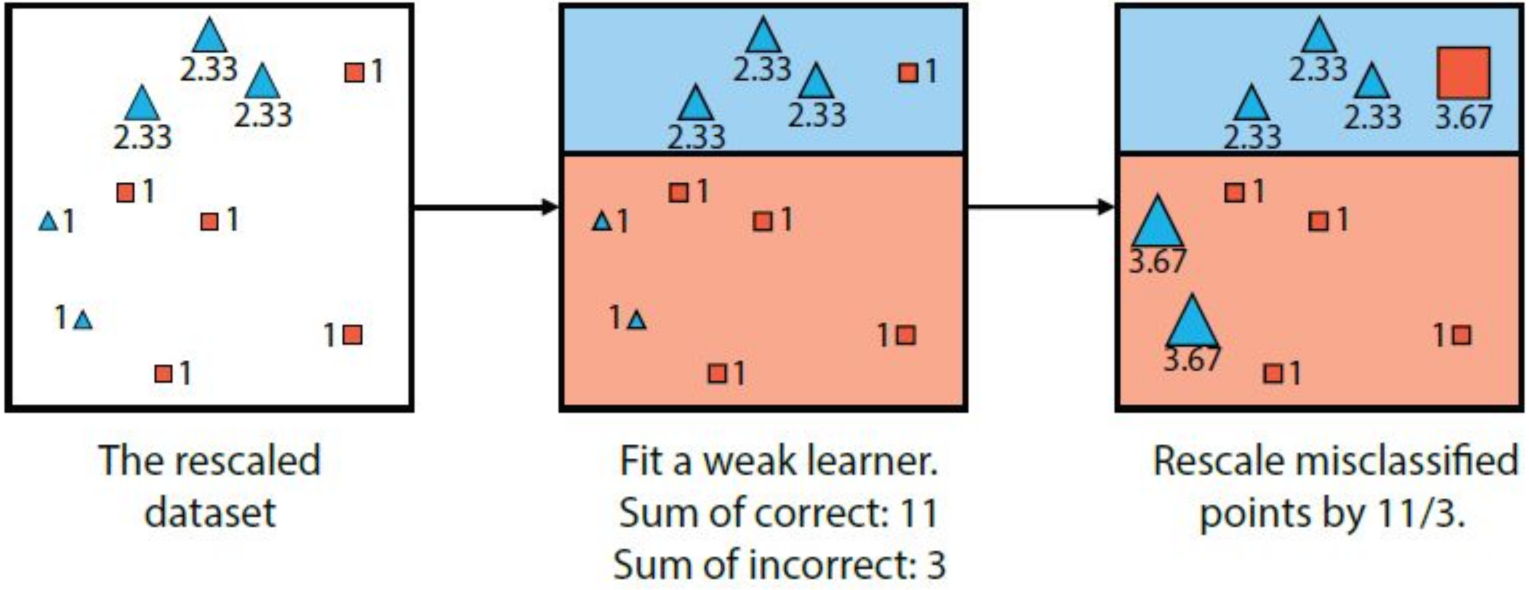
Add a weight of 1 to every point.

Fit a weak learner. Correct: 7 Incorrect: 3

Rescale misclassified points by 7/3.

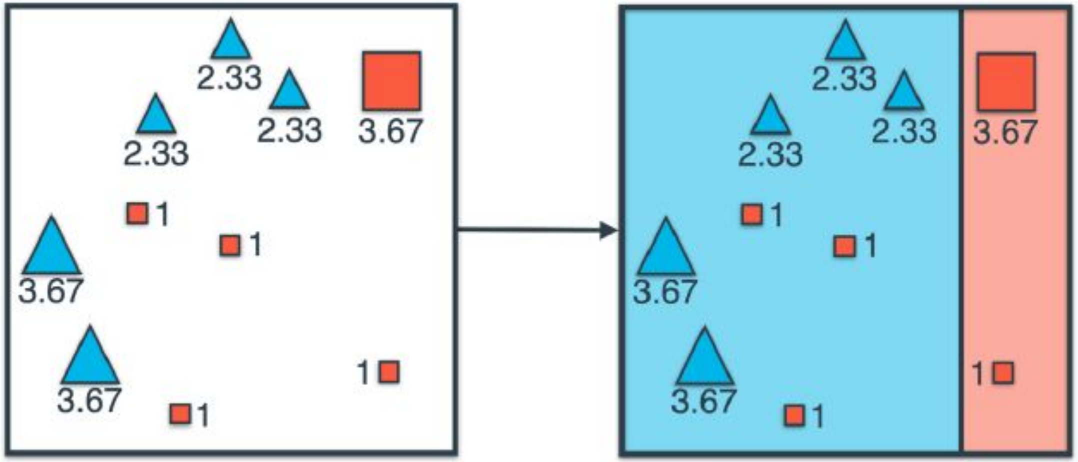
rescaling factor: $$\frac{\sum W_{classified}}{\sum W_{misclassified}}$$

AdaBoost (Adaptive Boosting)



the rescaling factor is 11/3 = 3.67

AdaBoost (Adaptive Boosting)



The rescaled dataset

Fit a weak learner
Sum of correct: 19
Sum of incorrect: 3

AdaBoost (Adaptive Boosting)

- The combination of weak learners in AdaBoost is similar to Random Forest, but with some differences

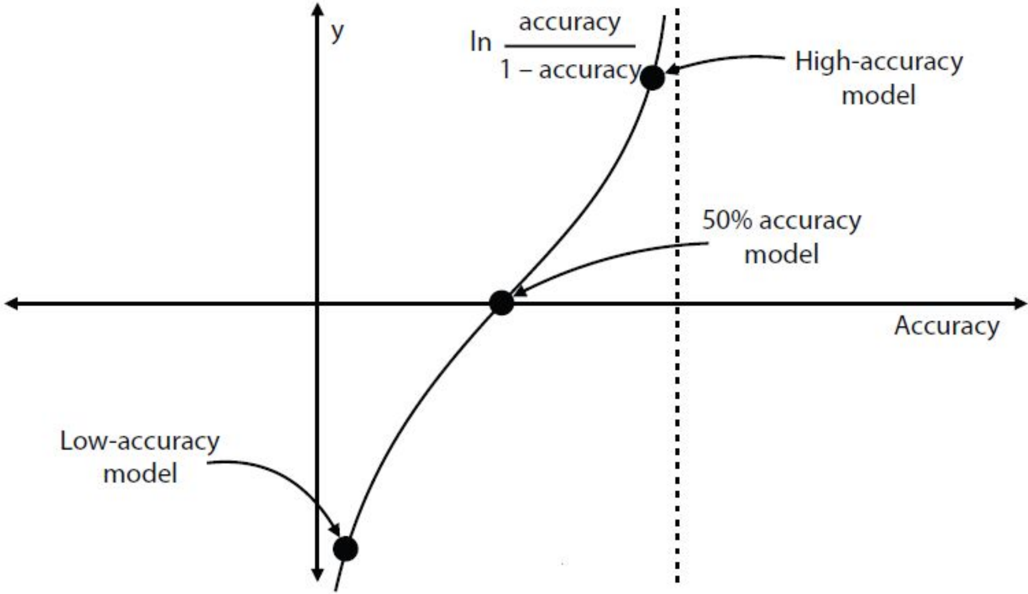
- The score of a weak learner is a number that has the following properties:

  1. Is positive when the accuracy of the learner is greater than 0.5
  2. Is 0 when the accuracy of the model is 0.5
  3. Is negative when the accuracy of the learner is smaller than 0.5
  4. Is a large positive number when the accuracy of the learner is close to 1
  5. Is a large negative number when the accuracy of the learner is close to 0

AdaBoost (Adaptive Boosting)

We are looking for a function that satisfies properties 1–5 before

$$\text{log-odds}(x) = ln \frac{x}{1-x}$$

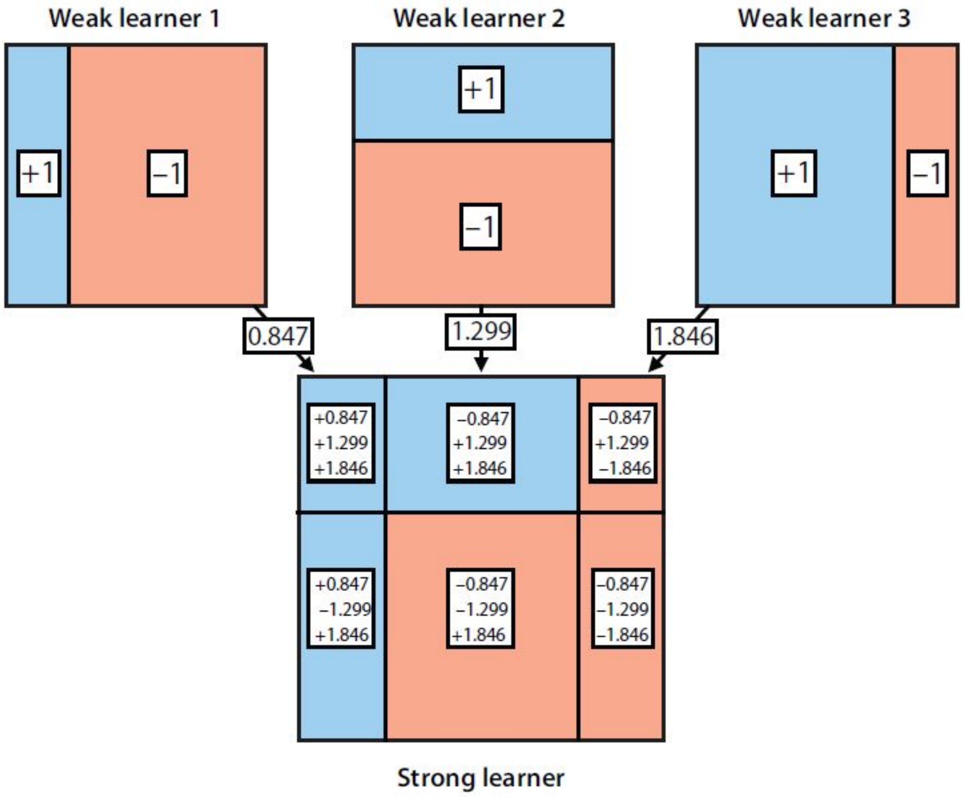


AdaBoost (Adaptive Boosting)

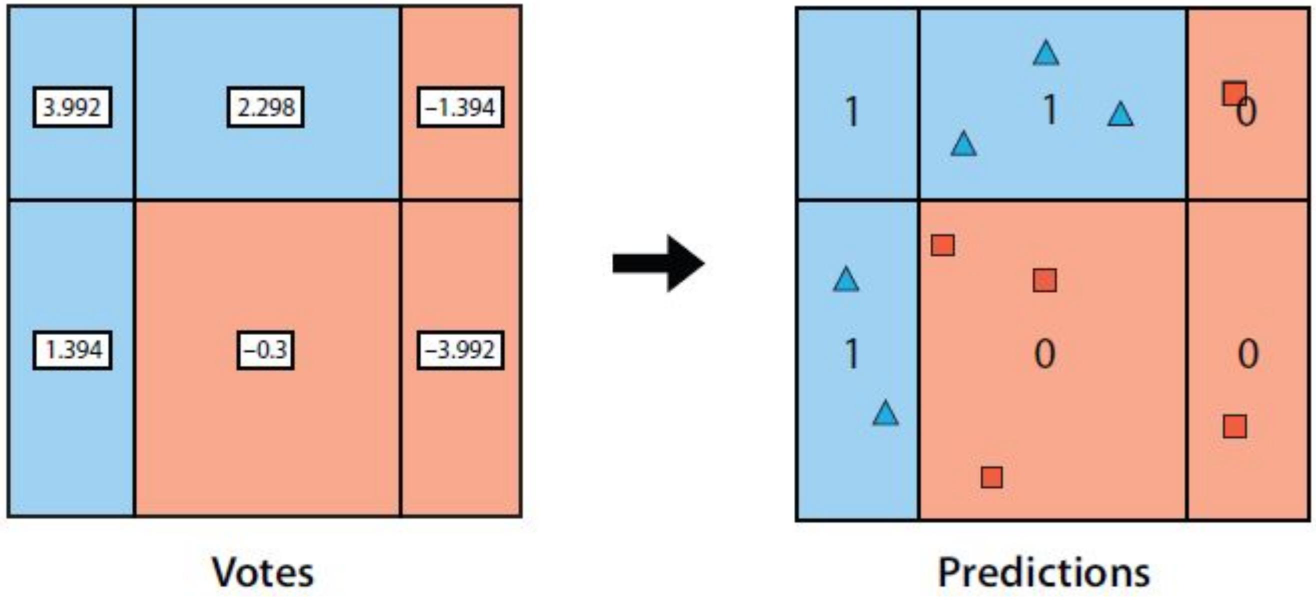
- Weak learner 1
  - Accuracy: 7/10
  - Score: ln (7/3) = 0.847
- Weak learner 2
  - Accuracy: 11/14
  - Score: ln (11/3) = 1.299
- Weak learner 3
  - Accuracy: 19/22
  - Score: ln (19/3) = 1.846

The prediction that the strong learner makes is obtained by the weighted vote of the weak classifiers, where each classifier's vote is its score.

AdaBoost (Adaptive Boosting)



AdaBoost (Adaptive Boosting)



Gradient Boosted Decision Trees

- Gradient boosting works by building trees in a serial manner, where each tree tries to correct the mistakes of the previous one.
- Gradient boosted trees often use very shallow trees, of depth one to five which makes the model smaller in terms of memory and makes predictions faster.
- Each tree can only provide good predictions on part of the data, and so more and more trees are added to iteratively improve performance.
- Gradient boosted trees are frequently the winning entries in machine learning competitions, and are widely used in industry.

Gradient Boosted Decision Trees

- An important parameter of gradientboosting is the *learning_rate*, which controls how strongly each tree tries to correct the mistakes of the previous trees.
- Another is the number of trees, which is commonly set to five.
- Also, the depth of the weak learners can be greater than one.

Gradient Boosted Decision Trees

Feature (age)	Label (engagement)
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

For the first weak learner the average value of the labels of this dataset is 4

Gradient Boosted Decision Trees

The next step is to calculate the residual, which is the difference between the label and the prediction made by this first weak learner

Feature (age)	Label (engagement)	Prediction from weak learner 1	Residual	Prediction from weak learner 2
10	7	4	3	3
20	5	4	2	2
30	7	4	3	2
40	1	4	–3	–2.667
50	2	4	–2	–2.667
60	1	4	–3	–2.667
70	5	4	1	0.5
80	4	4	0	0.5

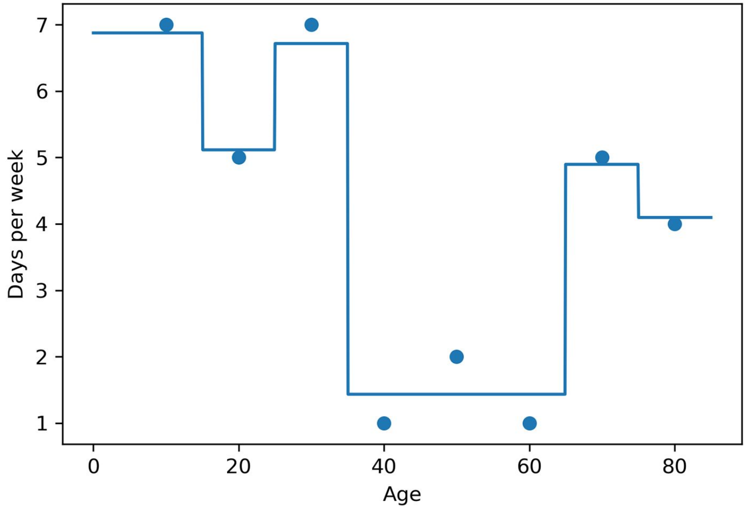
Gradient Boosted Decision Trees

To calculate the prediction from the first two weak learners, we first multiply the prediction of the second weak learner by the learning rate.

Label	Prediction from weak learner 1	Prediction from weak learner 2	Prediction from weak learner 2 times the learning rate	Prediction from weak learners 1 and 2	Residual
7	4	3	2.4	6.4	0.6
5	4	2	1.6	5.6	–0.6
7	4	2	1.6	5.6	1.4
1	4	–2.667	–2.13	1.87	–0.87
2	4	–2.667	–2.13	1.87	0.13
1	4	–2.667	–2.13	1.87	–0.87
5	4	0.5	0.4	4.4	0.6
4	4	0.5	0.4	4.4	–0.4

Gradient Boosted Decision Trees

- The idea is to continue in this fashion, calculating new residuals and training a new weak learner to fit these residuals.
- We repeat this process for every weak learner we want to build.



- *max_depth=2,*
- *n_estimators=4*
- *learning_rate=0.8*