Ensemble Methods, Random Forests and Boosting

Data Science Dojo

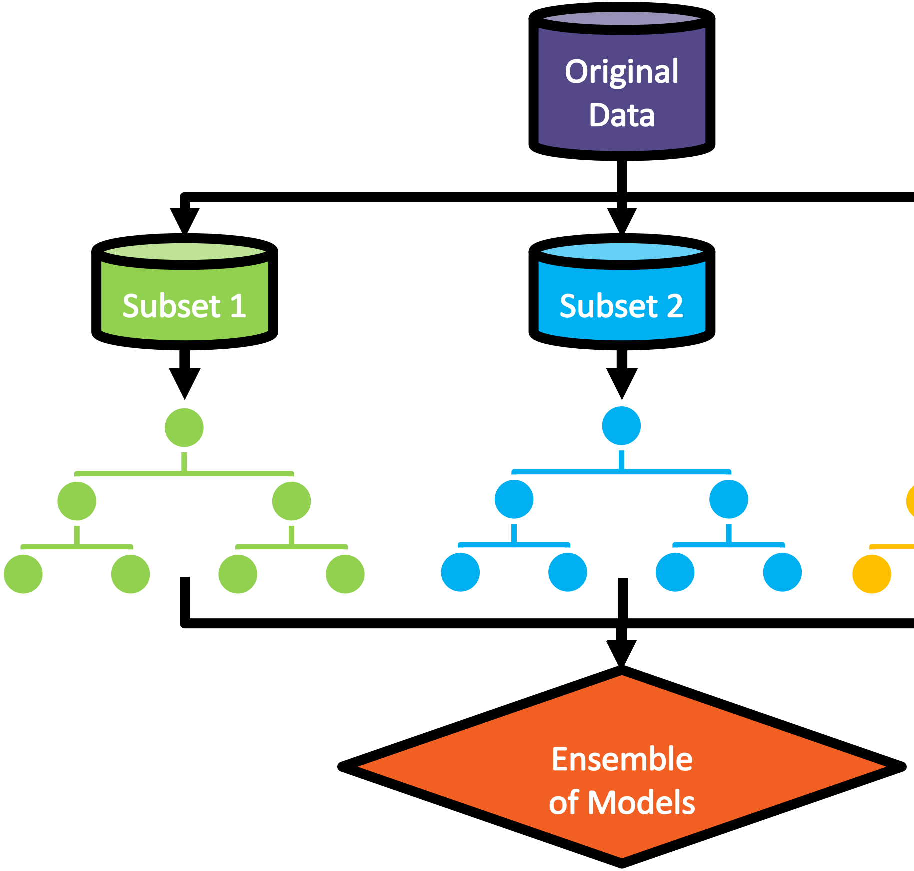
Agenda

- Overview and rationale
- Why ensemble?
  - Binomial Distribution
- Ensemble models
  - Bagging
  - Random Forests
  - Boosting
  - AdaBoost

Ensemble Methods

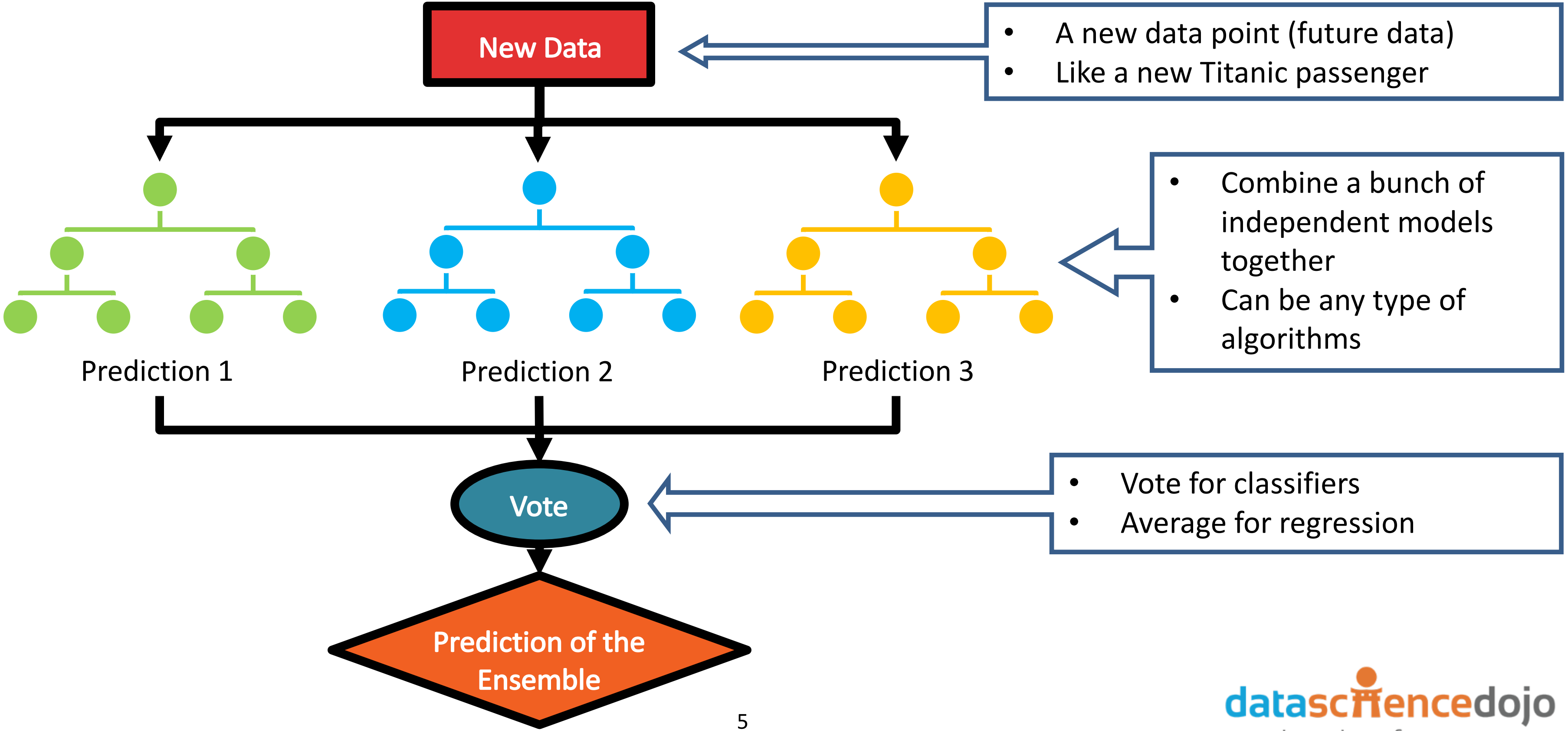
- Improve model performance by combining multiple models
- Can be used for both **classification** and **regression**

Ensemble of Decision Tree Models



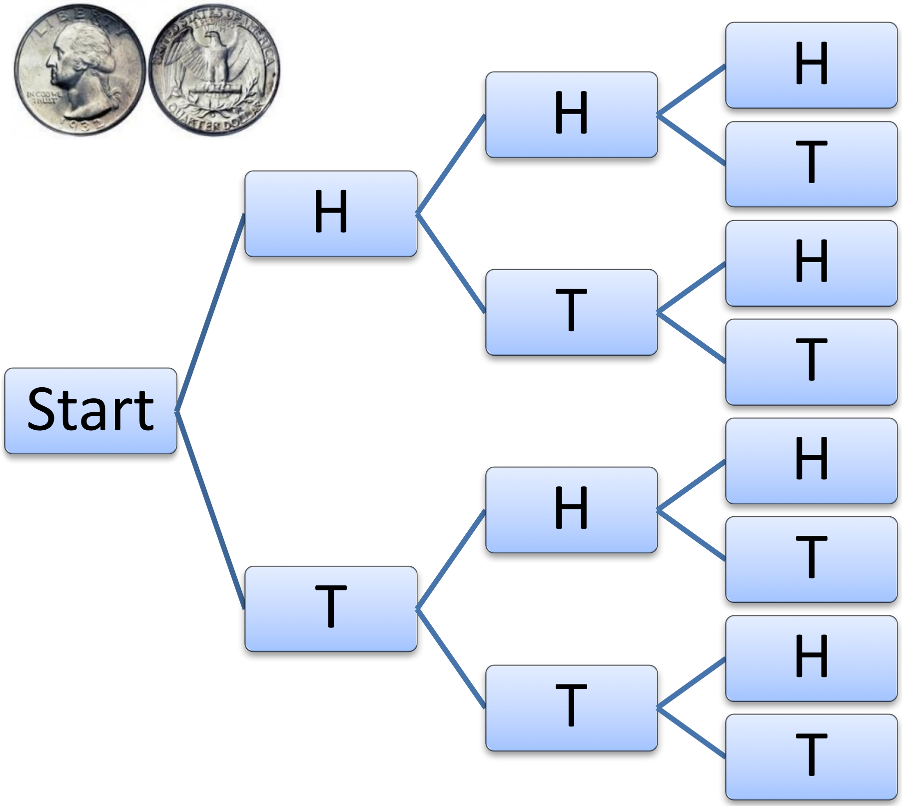
datasciencedojo
data science for everyone

Ensemble of Decision Tree Models



BINOMIAL DISTRIBUTION

Binomial Distribution



Heads	Tails
3	0
2	1
2	1
1	2
2	1
1	2
1	2
0	3

Consider…

- Flipping a coin 3 times in a row
- Each coin flip is considered **independent**
- A fair coin has a 50% rate of heads and tails

Properties of a binomial distribution:

- Well studied statistical property
- You cannot tell how each individual coin toss session will behave or their individual outcomes (such as HHH or HTH)
- However you can tell and **predict** the behavior of the aggregations of many coin toss sessions

datasciencedojo
data science for everyone

Binomial Distribution

$$f(k;n,p) = P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Exercises

Supplementary Material

Applications

- Number of life insurance holders who will claim in a given period
- Number of loan holders who will default in a certain period
- Number of false starts of a car in n attempts
- Number of faulty items in n samples from a production line
- **AND** Ensemble Methods

Why Does It Work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

Bagging

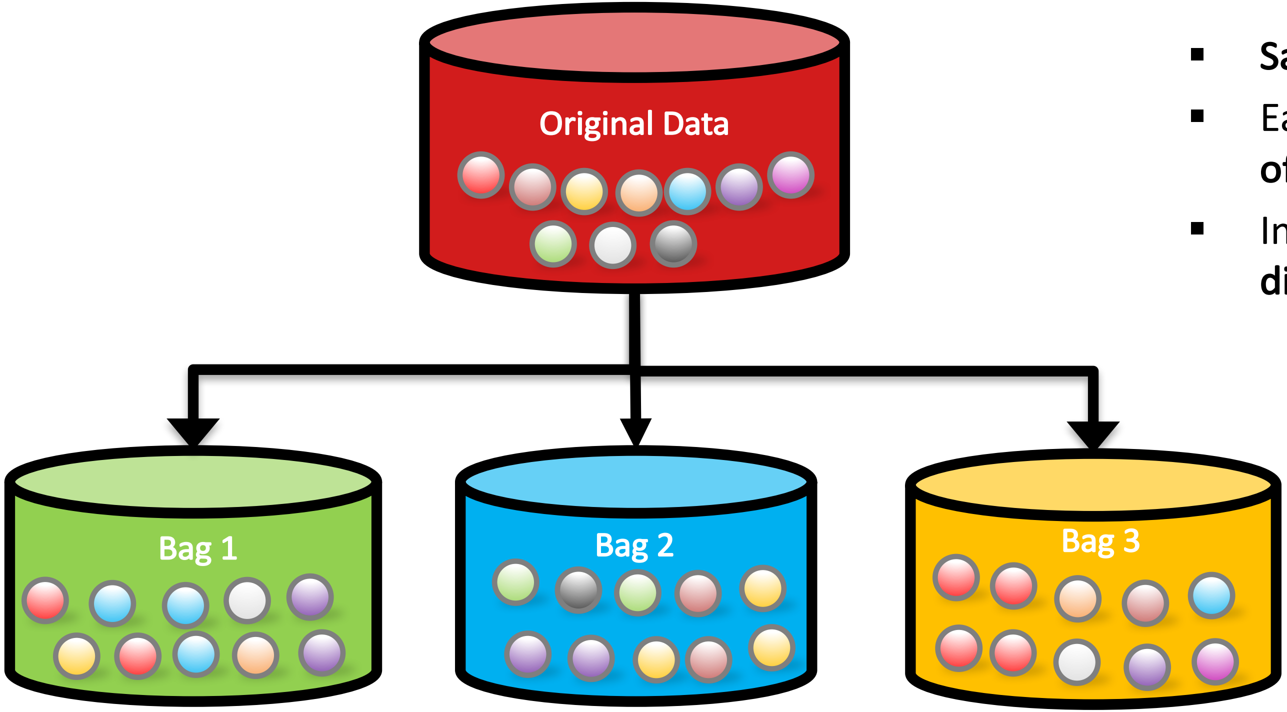
- All classifiers are created equal

Boosting

- **Not** all classifiers are created equal

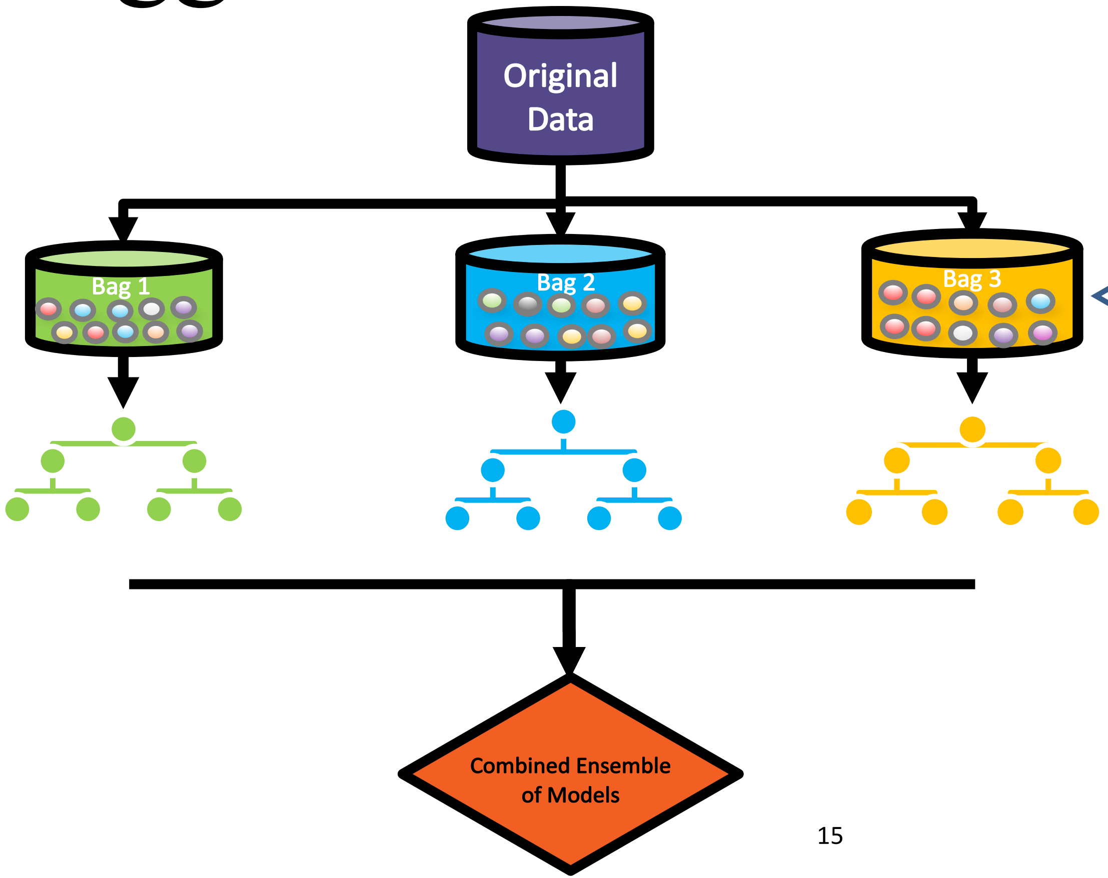
BAGGING

Bagging



- **Sampling with replacement**
- Each bag contains **variations of the original datasets**
- In turn this will result in **different** trees

Bagged Decision Forest



- Create a projection of the dataset sampled with replacement
- Each bag is randomly different, producing different trees

- Build a tree out of each bag

Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- For a training data of size n, each sample has probability $[1 - (1 – 1/n)^n]$ of being selected
- If n is large, this approximates to 1-1/e ~ 0.632

Bagging

- Reduces variance in estimate
- Prevents overfitting
- Robust to outliers

RANDOM FORESTS

What Is A Random Forest?

- An ensemble classifier using many decision tree models
- Can be used for classification or regression
- Accuracy and variable importance information is built-in

How Do Random Forests Work?

- A different subset of the training data are selected (~2/3), with replacement, to train each tree
- Remaining training data (aka out-of-bag data or simply OOB) is used to estimate error and variable importance
- Class assignment is made by the number of votes from all of the trees, and for regression, the average of the results is used

Which Features Are Used For Learning?

- A randomly selected subset of variables is used to split each node
- The number of variables used is decided by the user (mtry parameter in R)
- A smaller subset produces less correlation (lower error rate)

Rules of Thumb

- Given:
  - **N:** Total number of training data points
  - **M:** Number of features in training data
  - **m:** Number of features randomly selected for training each node

- Sample the data with replacement N times for building the training data for each tree.
- m<<M
- Classification: m = sqrt(M)
- Regression: m = M/3

Learning a Forest

- Dividing training examples into T subsets improves generalization
  - Reduces memory requirements & training time
- Train each decision tree t on subset I_t
  - Same decision tree learning as before
- Multi-core friendly (GPU implementation)

Implementation Details

- How many features and thresholds to try?
  - Just one = "extremely randomized" [Geurts *et al.* 06]
  - Few → fast training, may under-fit, may be too deep
  - Many → slower training, may over-fit
- When to stop growing the tree?
  - Maximum depth
  - Minimum entropy gain
  - Delta class distribution
  - Pruning

Random Forest: R Exercise

BOOSTING

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

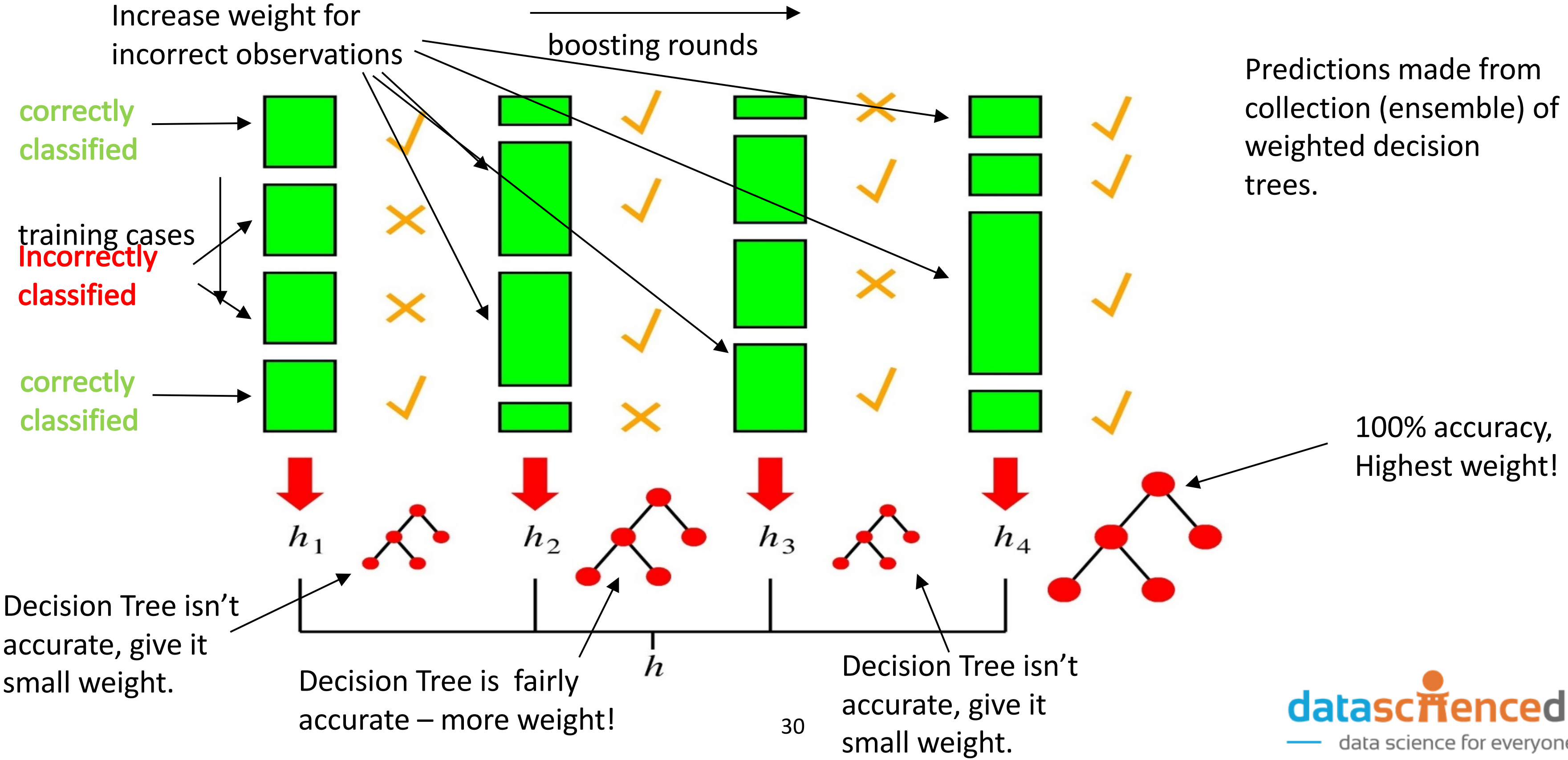
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Boosting Intuition

- We adaptively weight each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them).
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.
- We stop by monitoring a hold out set.

Boosting in a Picture



ADABOOST

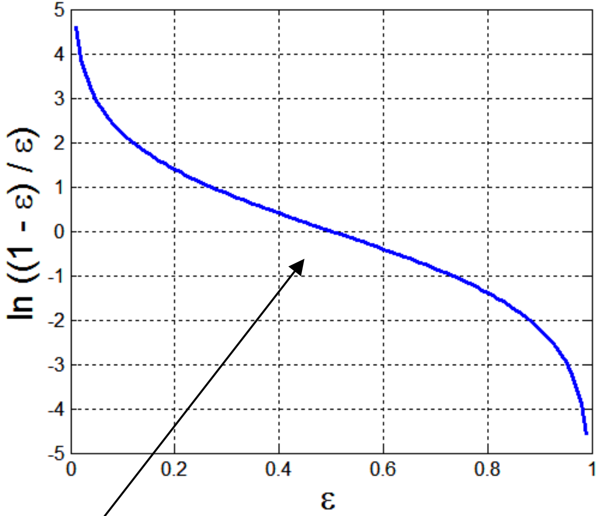
AdaBoost (Adaptive Boosting)

- Base classifiers: $C_1, C_2, \ldots, C_T$
- Error rate [Weighted loss function]:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



What's interesting about this curve?

AdaBoost

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

$$\textit{where } Z_j \textit{ is the normalization factor}$$

Ensure weights add up to 1.0

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated.

- Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

Common Misconception

A Random Forest and a Boosted Decision Tree are **not** the same

QUESTIONS