Ensemble learning

Ali Madani
Farnoosh Khodakarami

Ensemble learning (concept)

Marie Jean Antoine Nicolas de Caritat
(French mathematician; 1743–1794)

Condorcet's jury theorem in 1785:

- Each voter has a probability $p > .5$ of being correct (better than a random guess)
  - adding more voters increases the probability of making the correct decision

Ensemble learning methods

Majority

Averaging

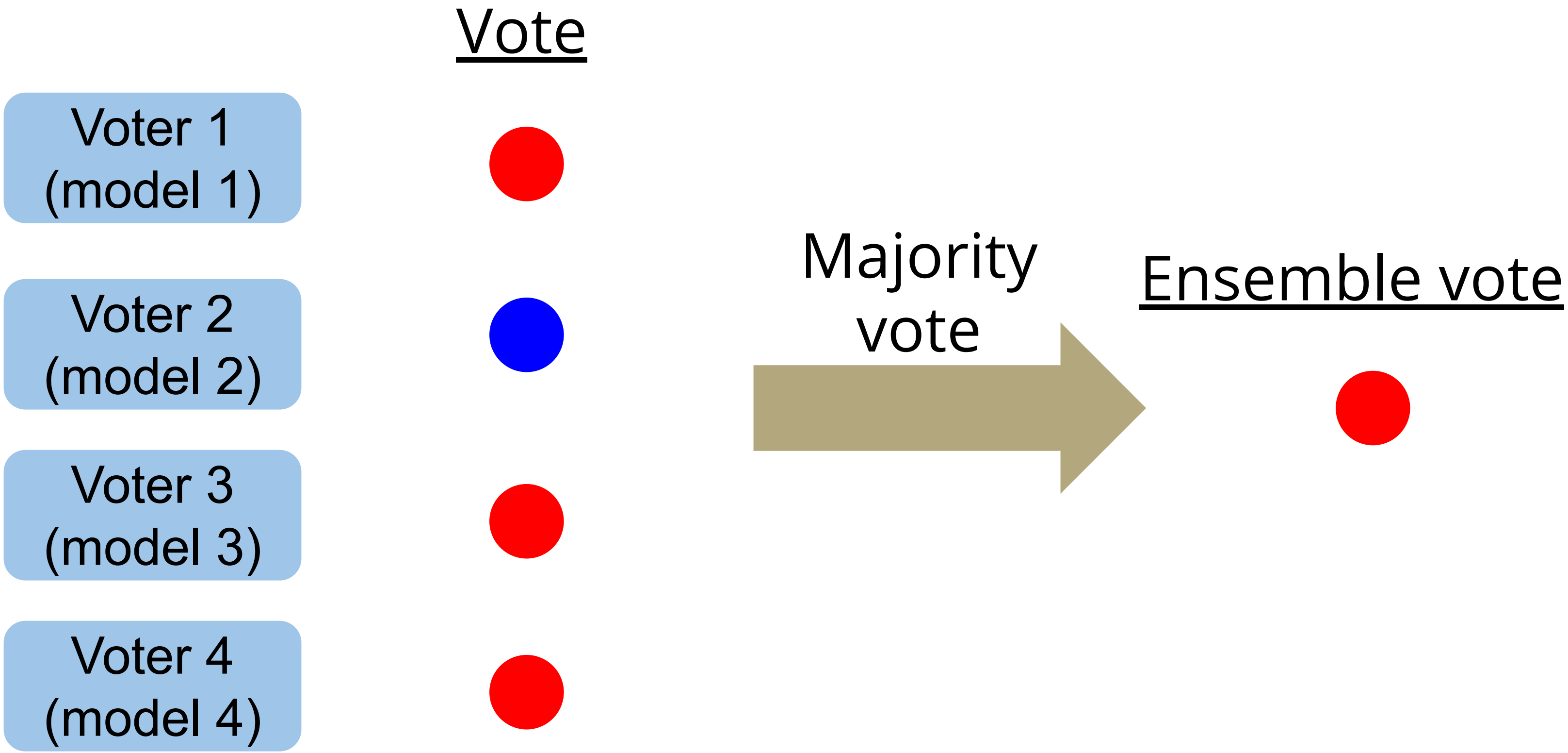
Weighted Average

Stacking

Bagging

Boosting

Majority (for classification)



Averaging (for regression or calculating class probabilities in classification)

Predicted value

model 1	1
model 2	2
model 3	2
model 4	1

Averaging

Ensemble prediction

(1+2+2+1)/4=1.5

Weighted Average

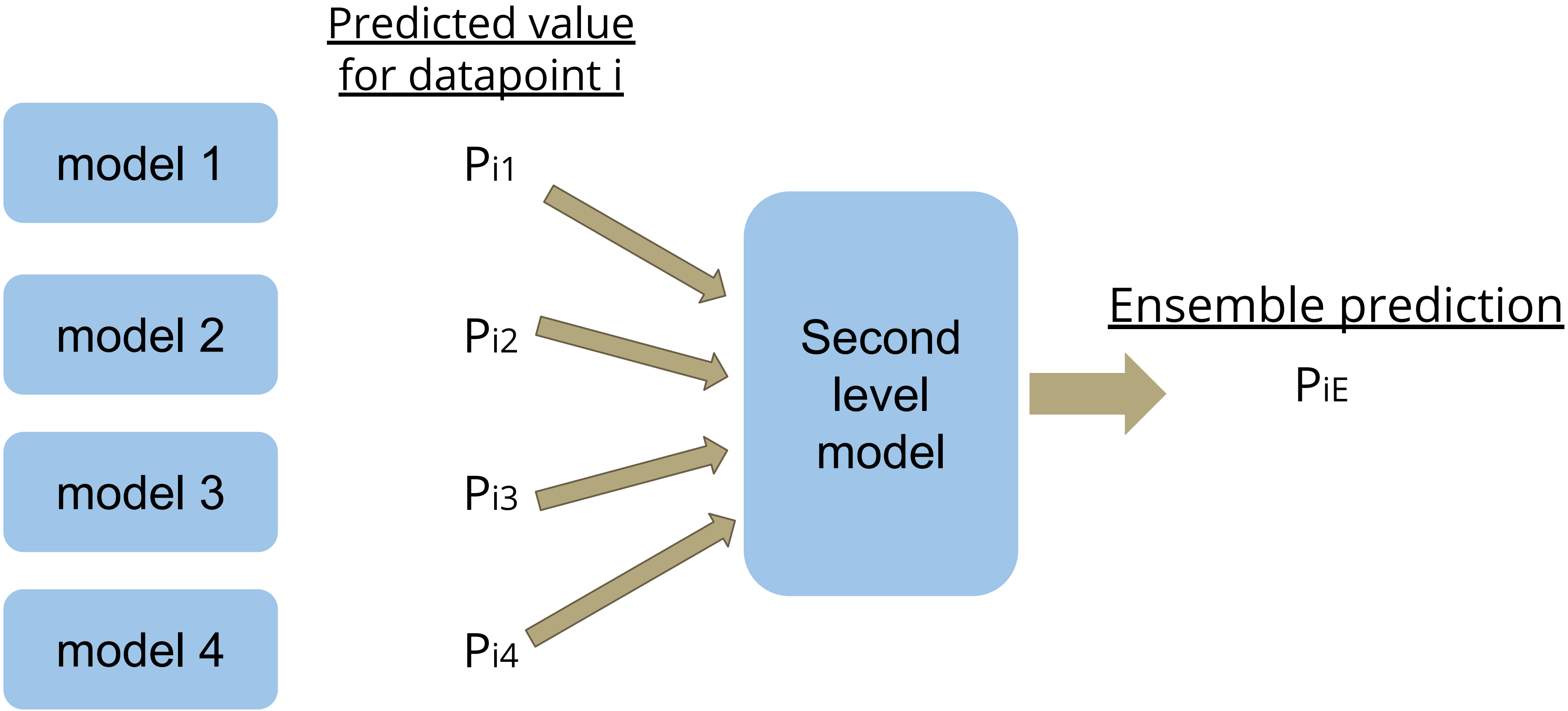
	Predicted value	Weight
model 1	1	0.1
model 2	2	0.25
model 3	2	0.35
model 4	1	0.3
		Total = 1

Weighted average

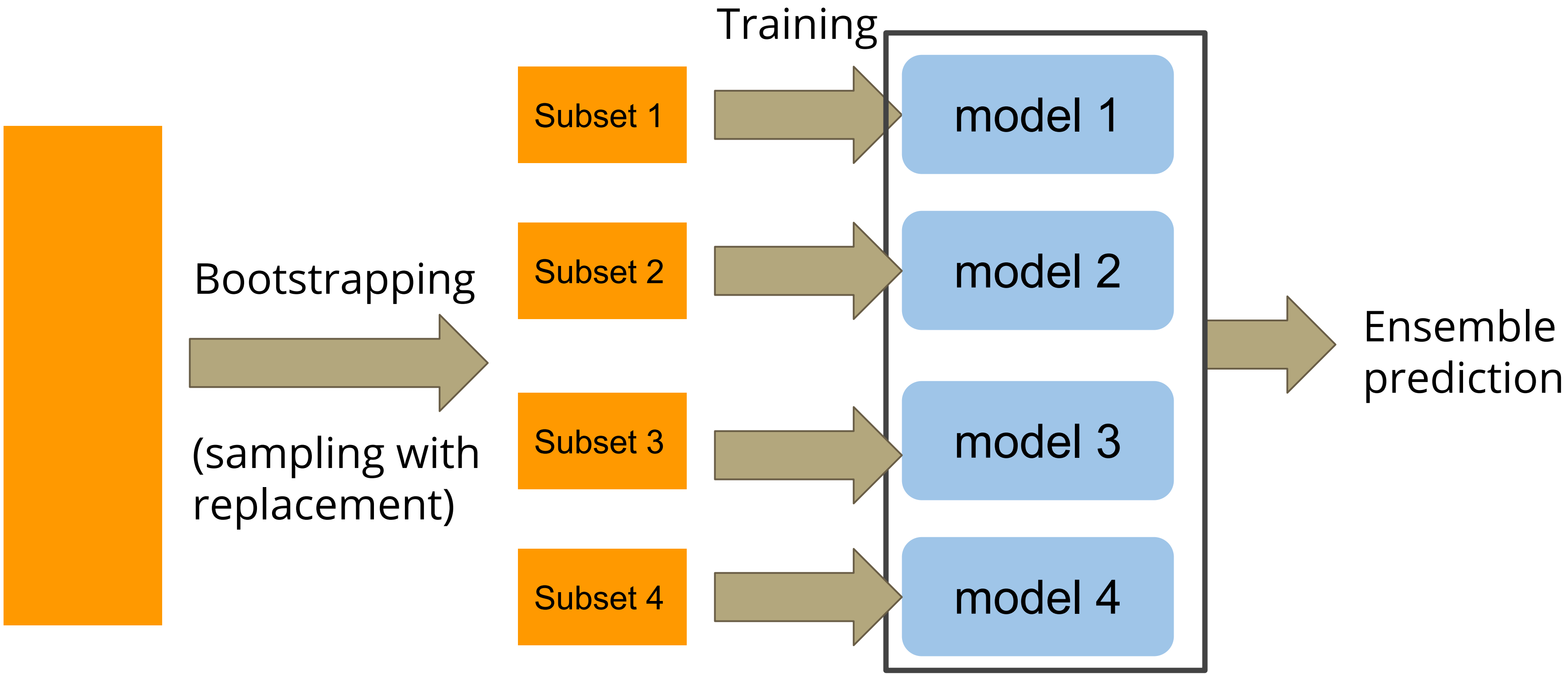
Ensemble prediction

1*0.1+2*0.25+
2*0.35+1*0.3
=1.6

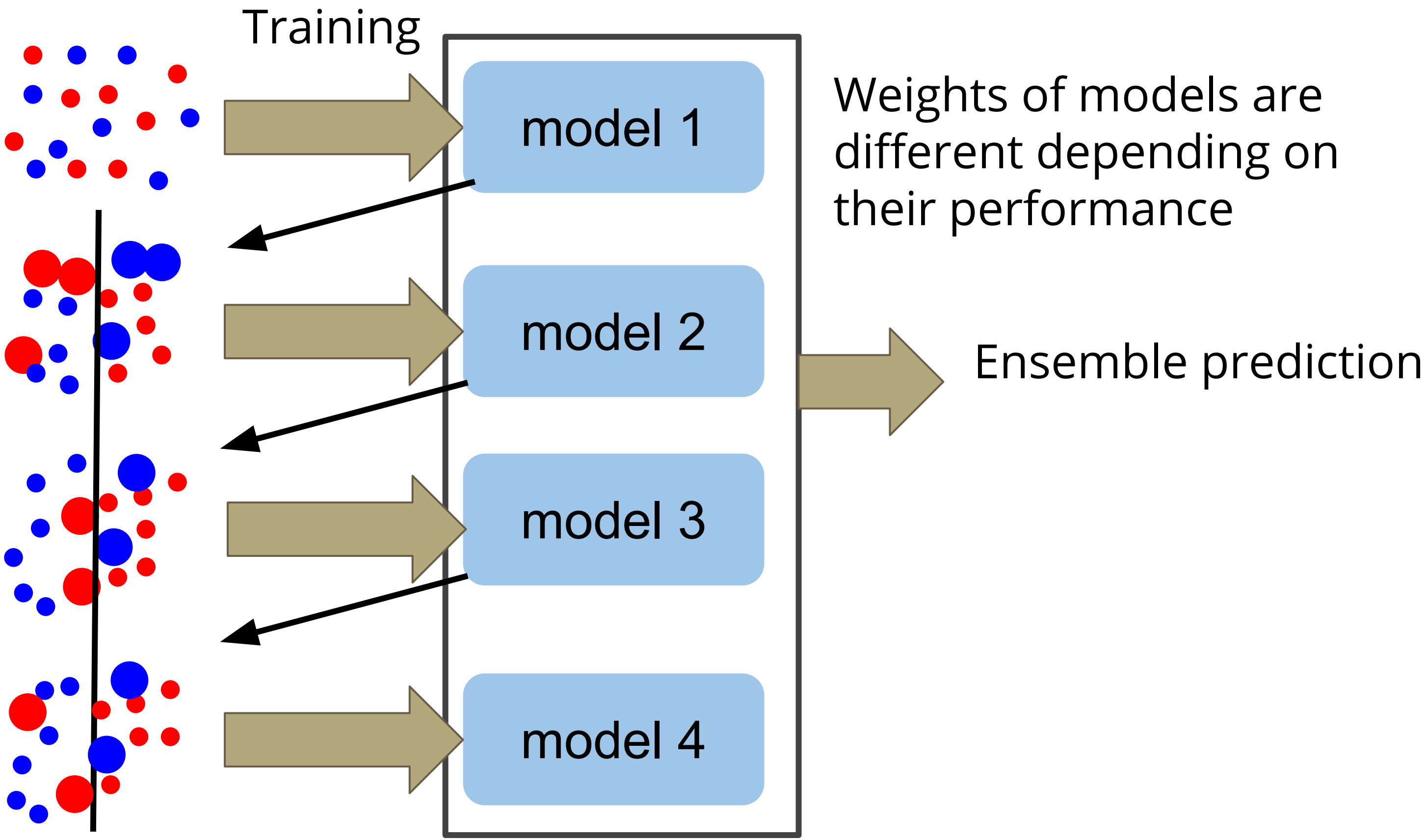
Stacking


Predicted value
for datapoint i
model 1
model 2
model 3
model 4
Pi1
Pi2
Pi3
Pi4
Second
level
model
Ensemble prediction
PiE

Bagging (Bootstrap Aggregating)



Boosting (sequential model correction)



Bagging and boosting algorithms

Bagging algorithms

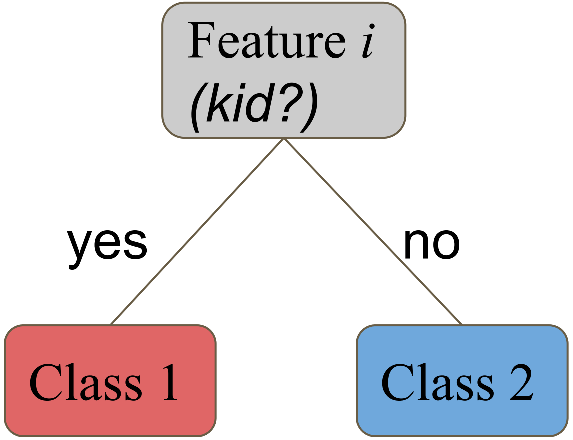
Random forest

Boosting algorithms

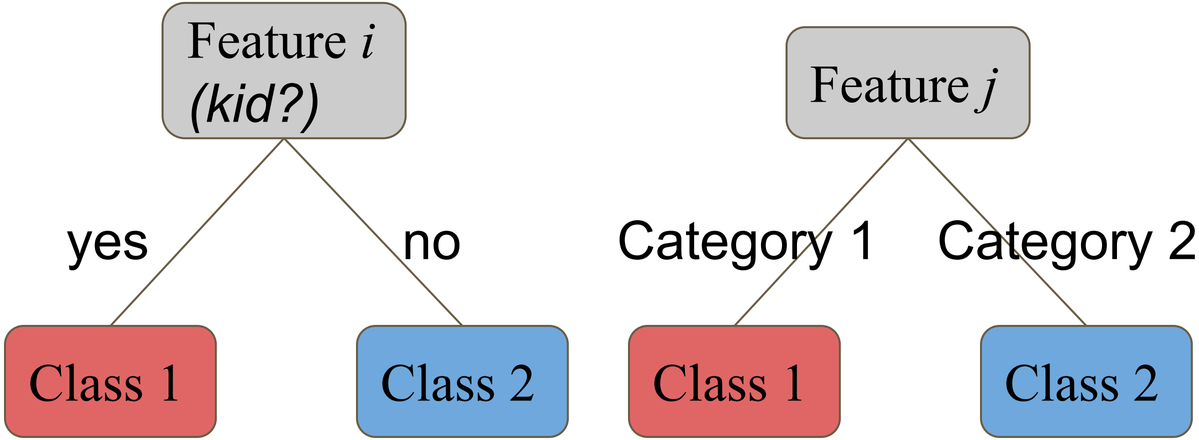
Adaboost
Gradient Boosting Method (GBM)

Decision Trees and Random Forest

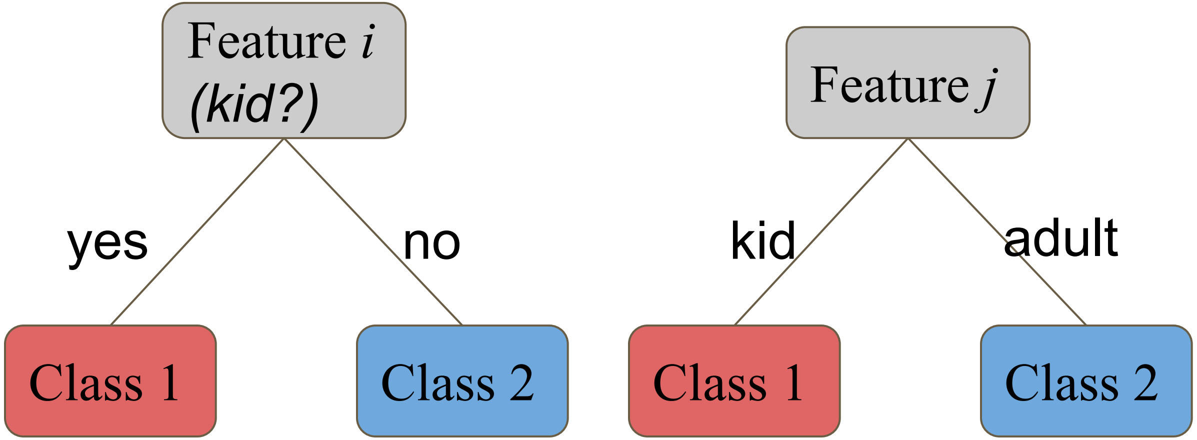
Individual decision tree for classification



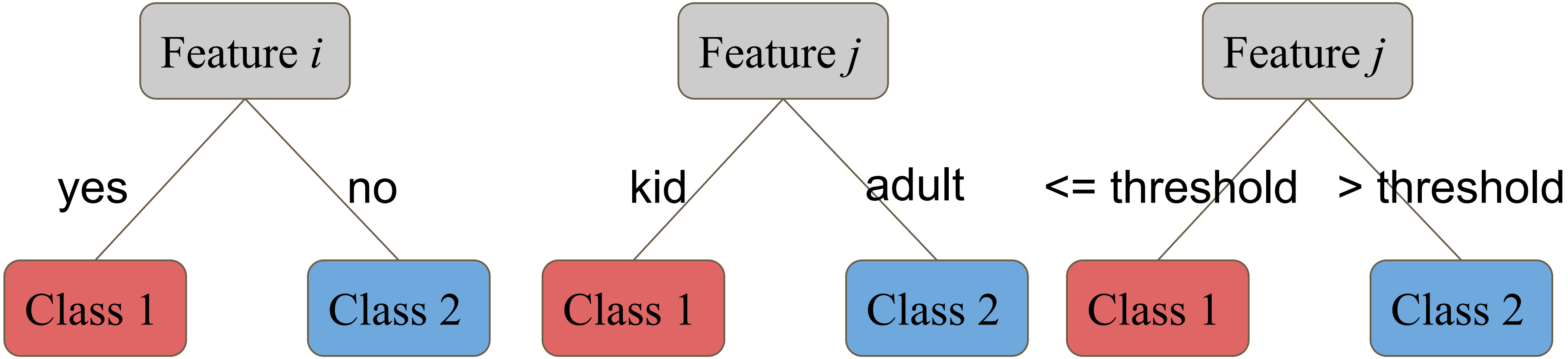
Individual decision tree for classification



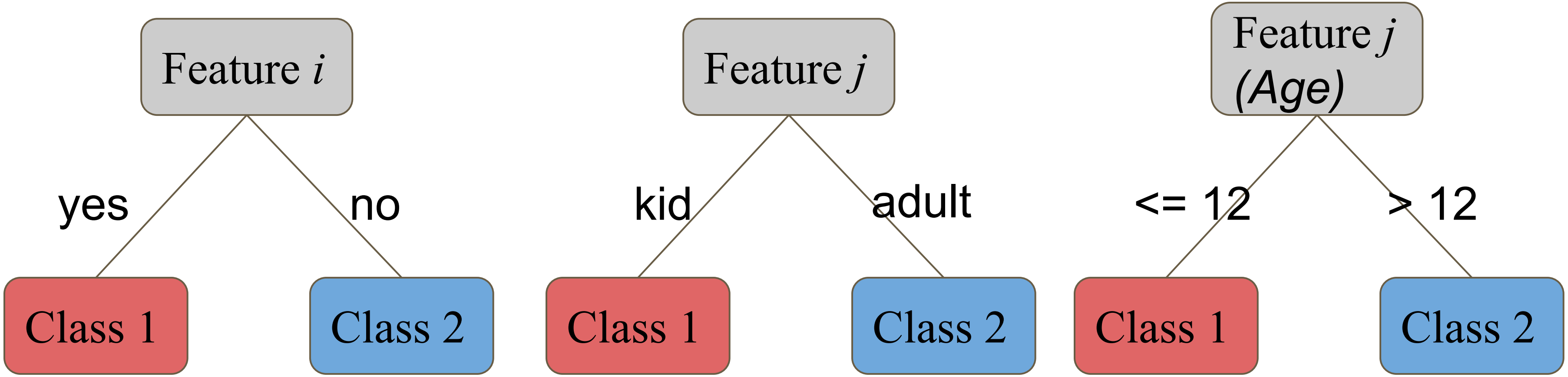
Individual decision tree for classification



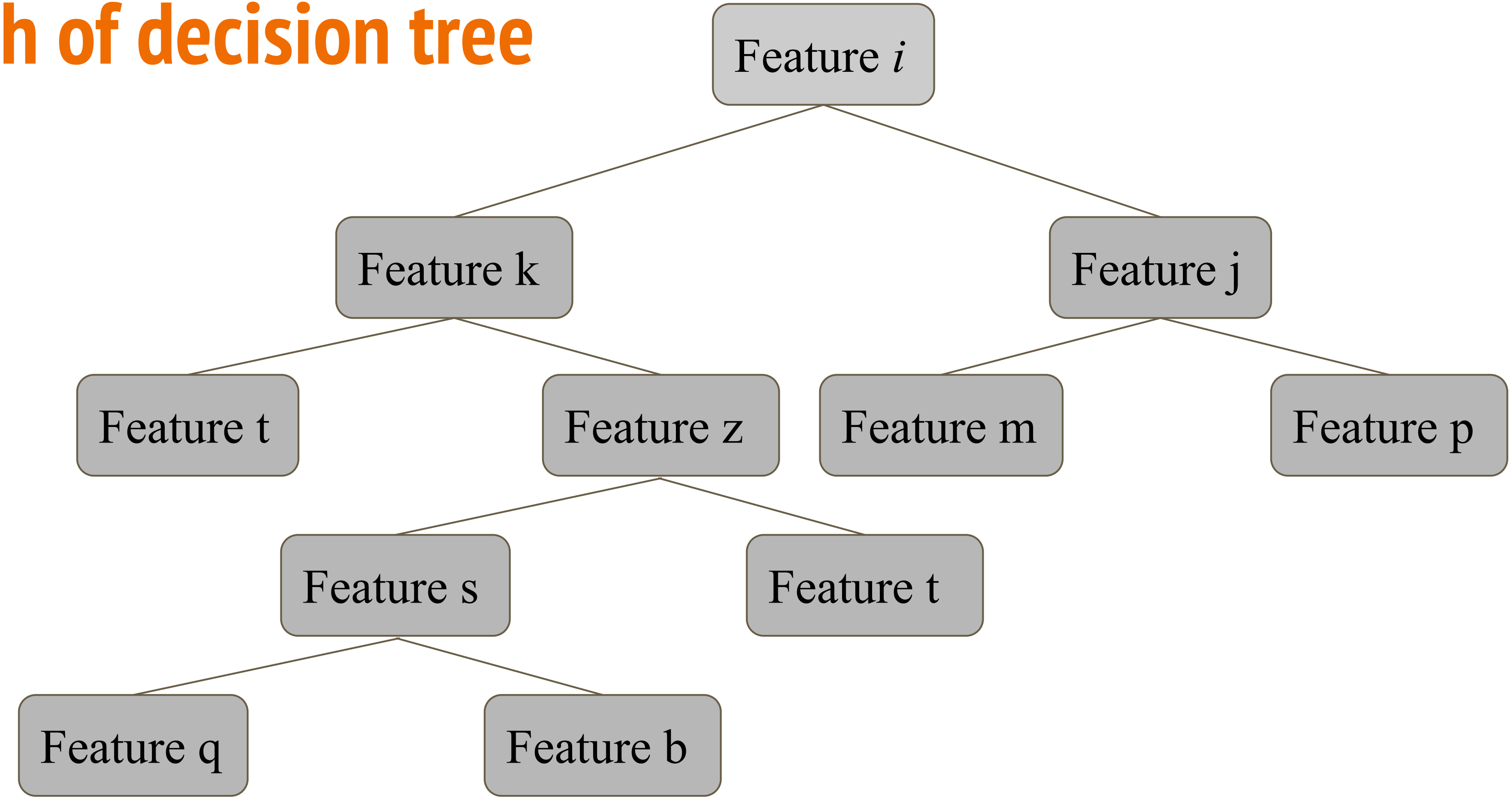
Individual decision tree for classification



Individual decision tree for classification

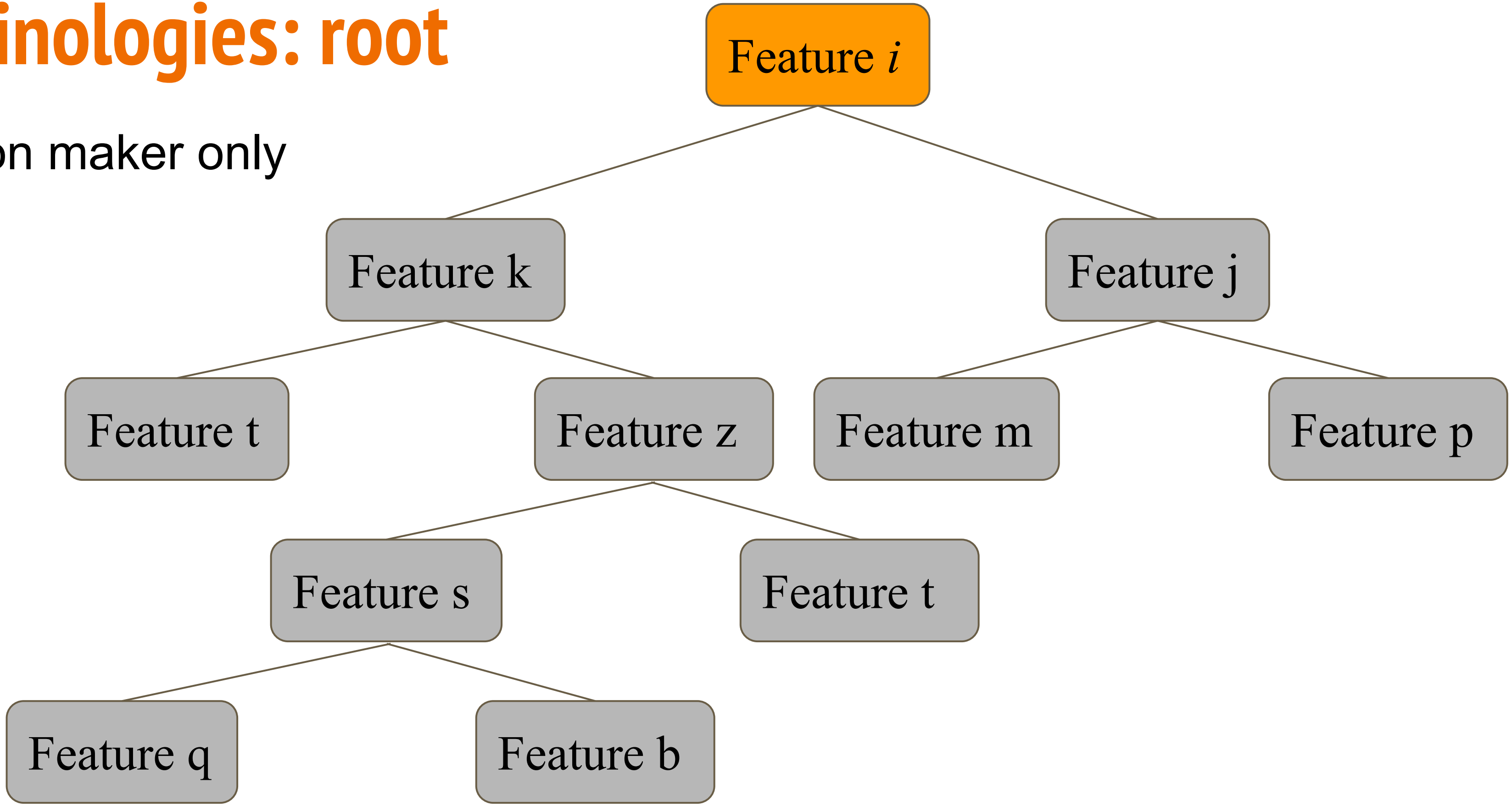


Depth of decision tree



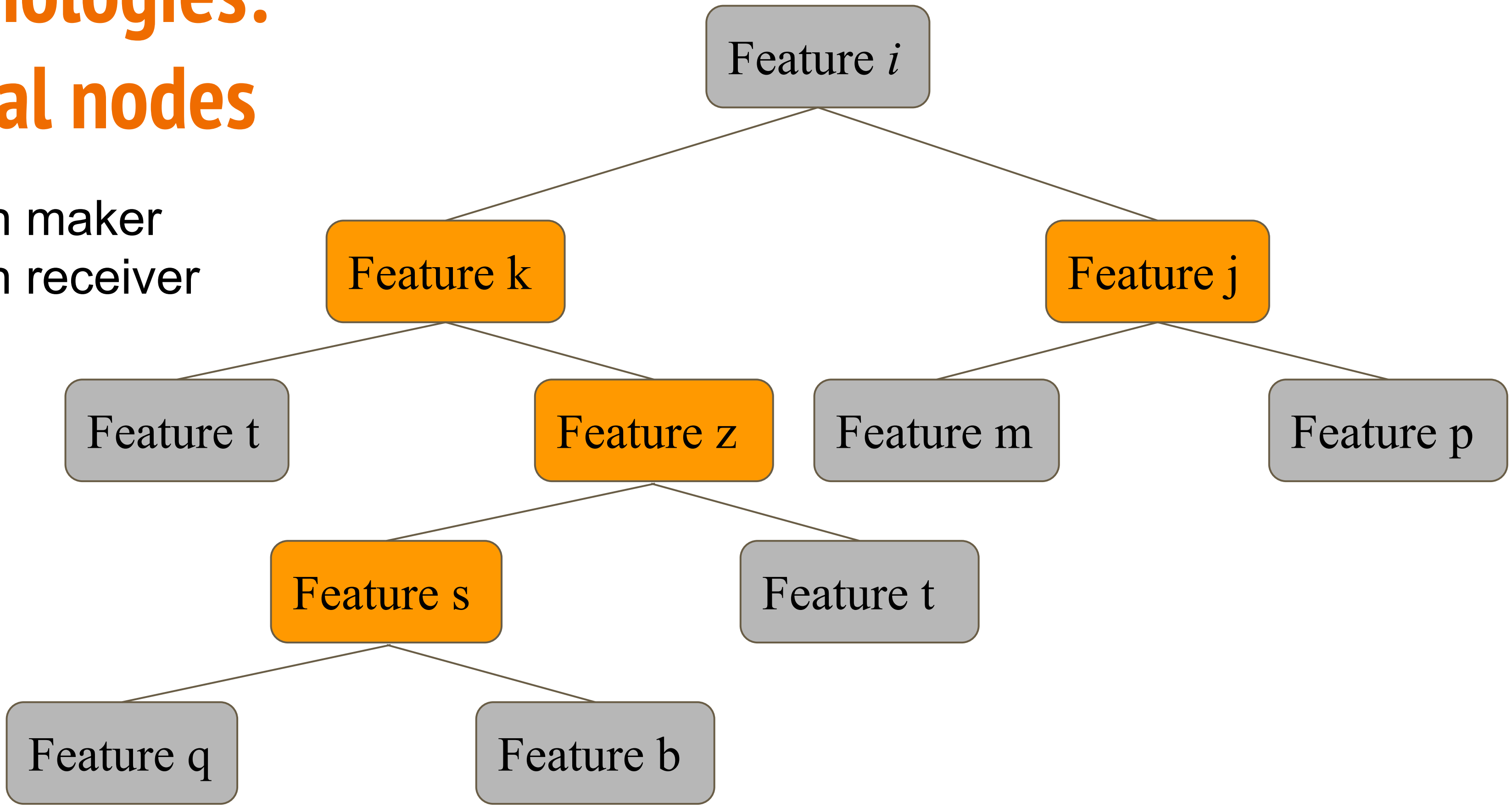
Terminologies: root

Decision maker only



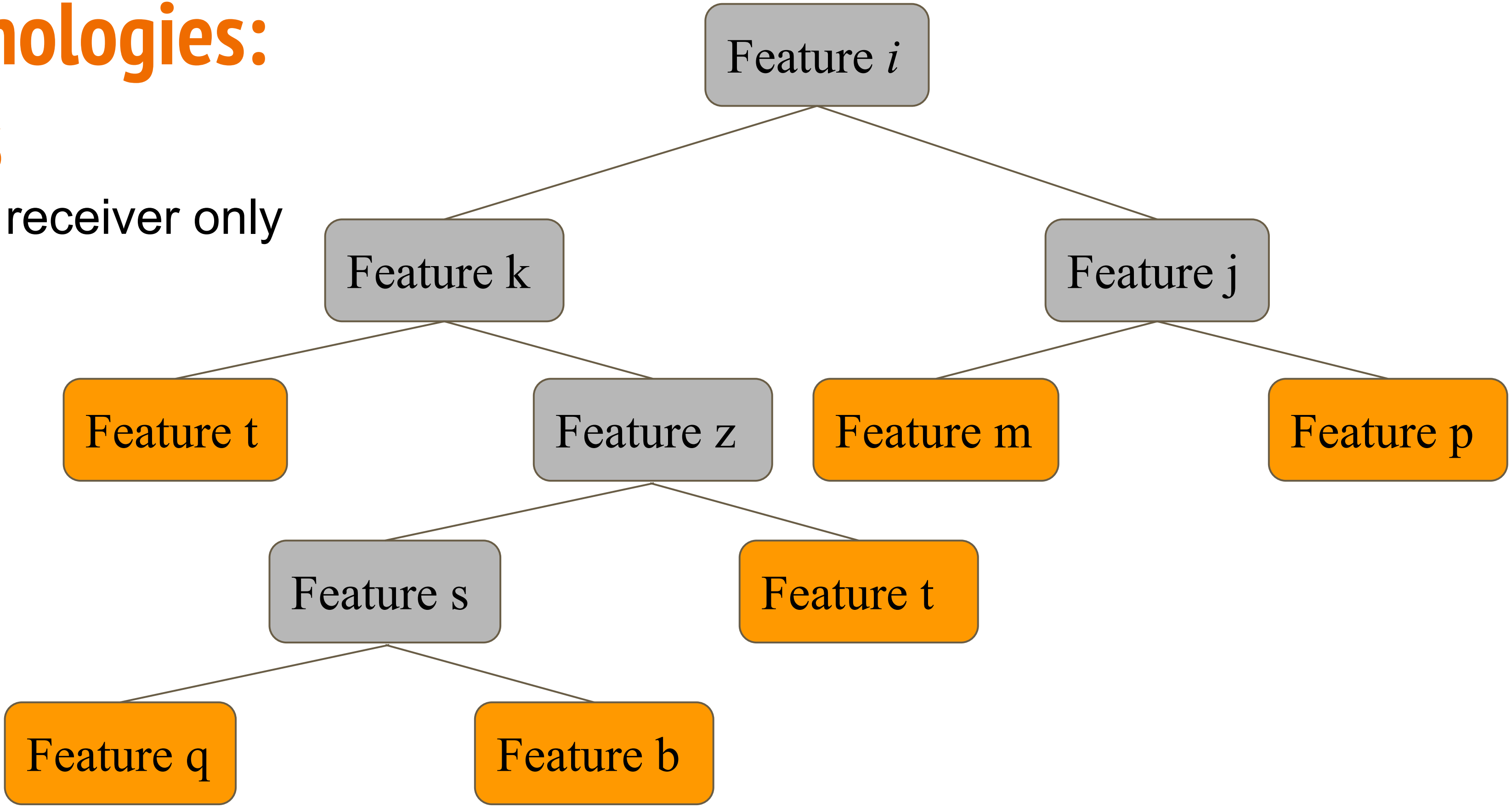
Terminologies: internal nodes

Decision maker
Decision receiver

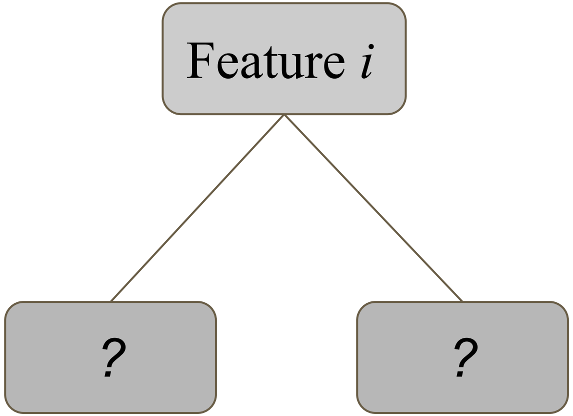


Terminologies: leaves

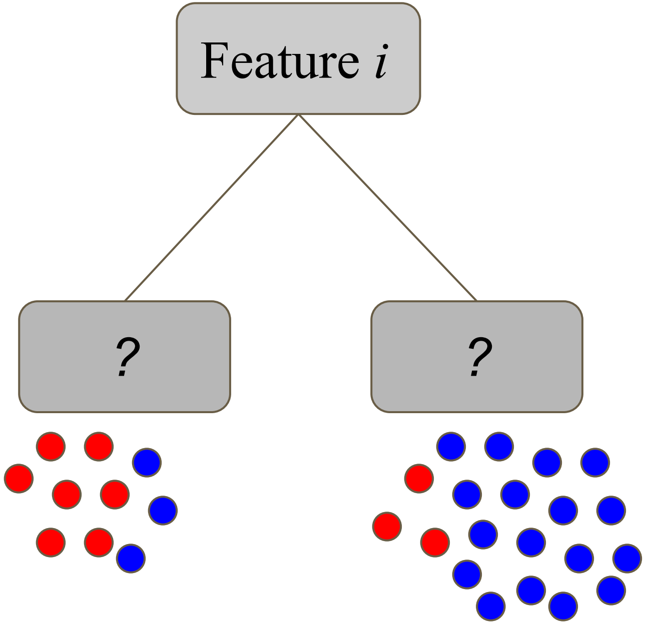
Decision receiver only



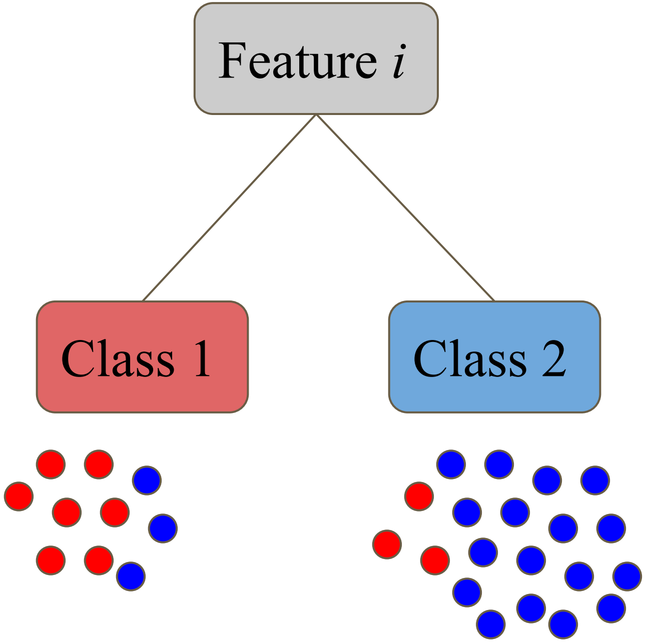
The difference between the groups are not known yet



Data points within each group determines the identities of the groups



Data points within each group determines the identities of the groups

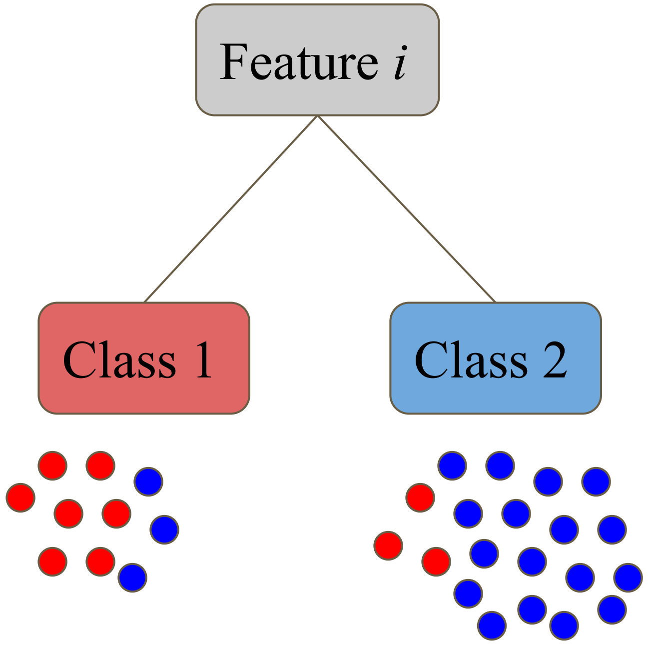


It is better to not have mixed identities (classes)

Let's assess impurity of the classes:

$$Gini = 1 - \Sigma_{i=1}^{C} P_i \qquad P_i = \frac{N_i}{\Sigma_{i=1}^{C} N_i}$$

C: total number of classes

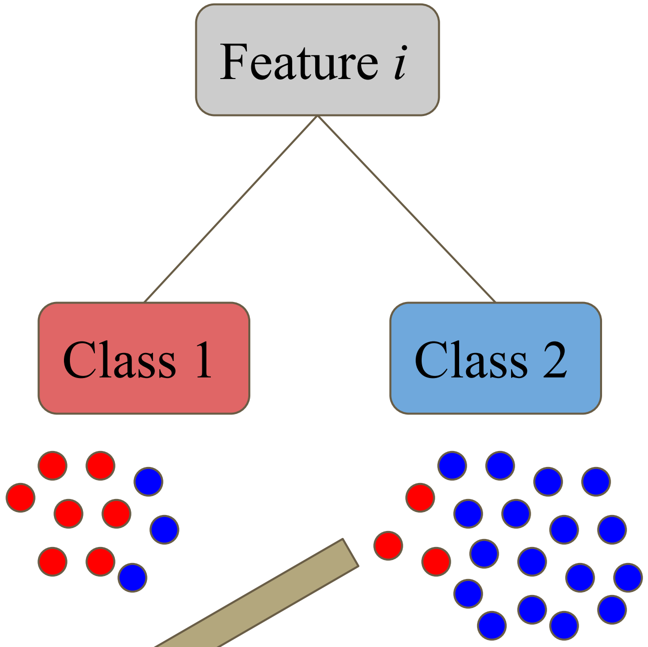


Calculating impurity (Gini) for each leaf

Let's assess impurity of the classes:

$$Gini = 1 - \Sigma_{i=1}^{C} P_i^2 \quad P_i = \frac{N_i}{\Sigma_{i=1}^{C} N_i}$$

C: total number of classes



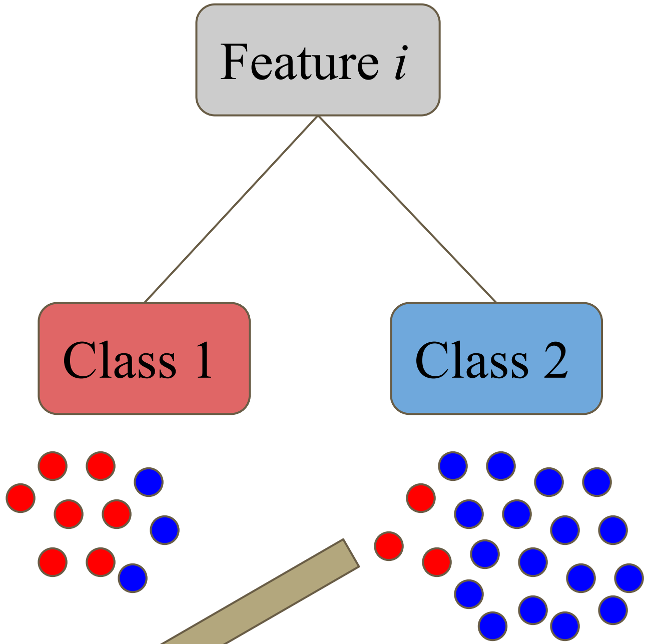
$$Gini = 1 - ((\tfrac{7}{10})^2 + (\tfrac{3}{10})^2) =$$
$$1 - (\tfrac{49}{100} + \tfrac{9}{100}) = 0.42$$

$$Gini = 1 - ((\tfrac{3}{20})^2 + (\tfrac{17}{20})^2) =$$
$$1 - (\tfrac{9}{400} + \tfrac{289}{400}) = 0.255$$

Total impurity as the weighted average of leaf impurities

$$Gini_{total} = \frac{\Sigma_{j=1}^{L} N_j * Gini_j}{\Sigma_{j=1}^{L} N_j}$$

$$Gini_{total} = \frac{10*0.42+20*0.255}{10+20} = 0.31$$



$$Gini = 1 - ((\frac{7}{10})^2 + (\frac{3}{10})^2) =$$
$$1 - (\frac{49}{100} + \frac{9}{100}) = 0.42$$

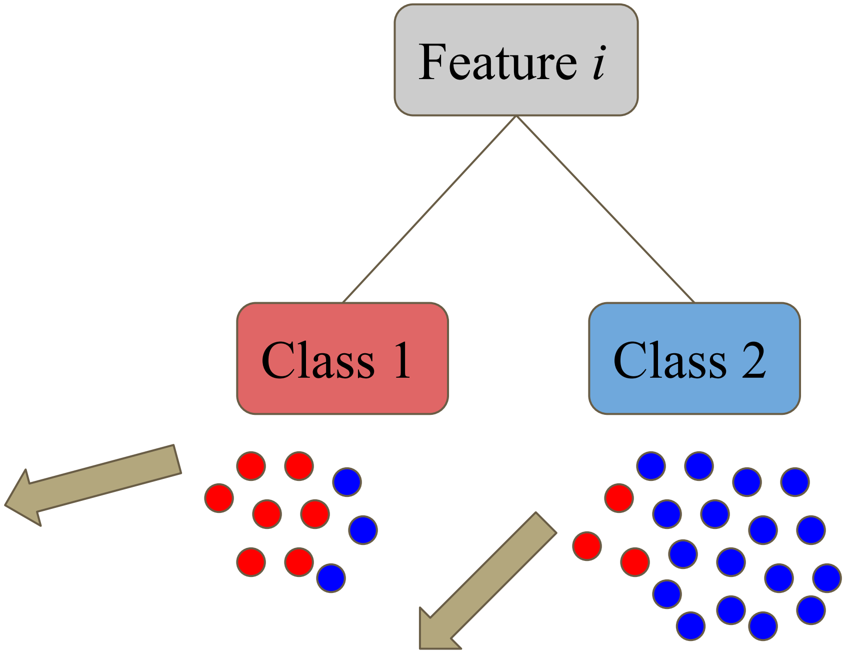
$$Gini = 1 - ((\frac{3}{20})^2 + (\frac{17}{20})^2) =$$
$$1 - (\frac{9}{400} + \frac{289}{400}) = 0.255$$

Entropy as another measure for impurity assessment

$$Entropy = -\Sigma_{i=1}^{C} P_i log_2(P_i)$$

$$P_i = \frac{N_i}{\Sigma_{i=1}^{C} N_i}$$

C: total number of classes



$$Entropy = -(\frac{7}{10} log_2 \frac{7}{10} + \frac{3}{10} log_2 \frac{3}{10})$$

$$Entropy = -(\frac{3}{20} log_2 \frac{3}{20} + \frac{17}{20} log_2 \frac{17}{20})$$

Assigning features as nodes using impurity

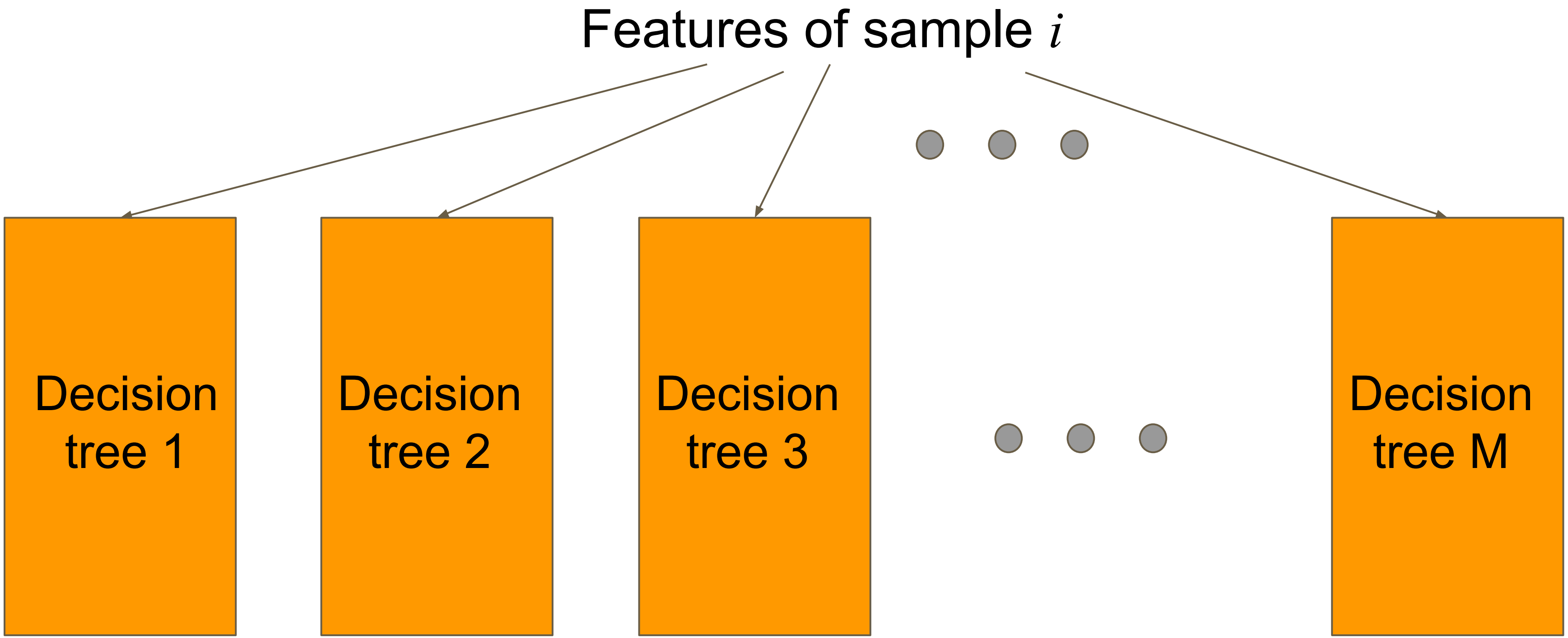
It is better to have shallower trees

- Choosing the feature with the lowest Gini as the root

- Then choosing the next features with the lowest Gini for the next internal node

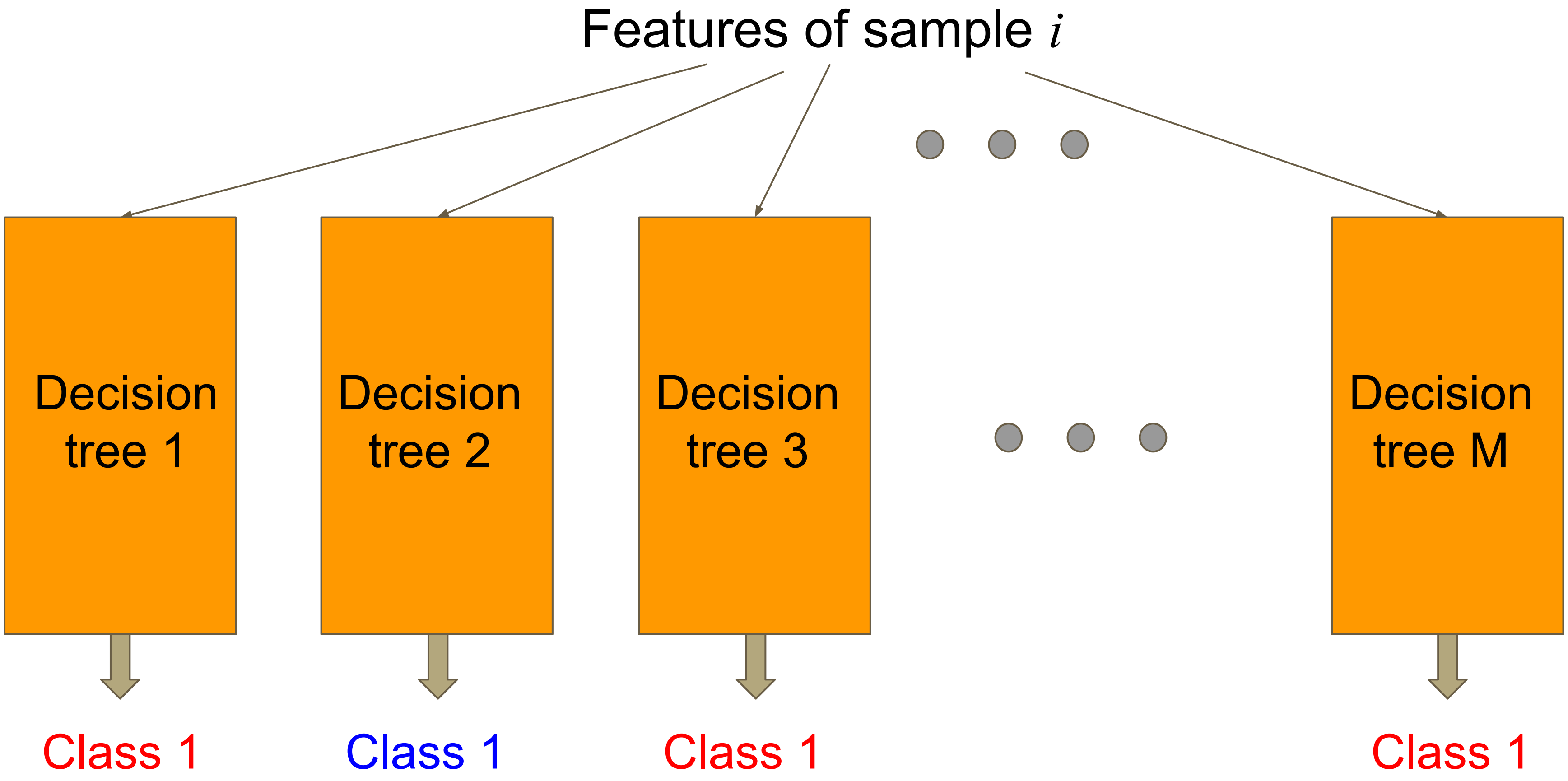
- ....

Combining decision trees to build random forests

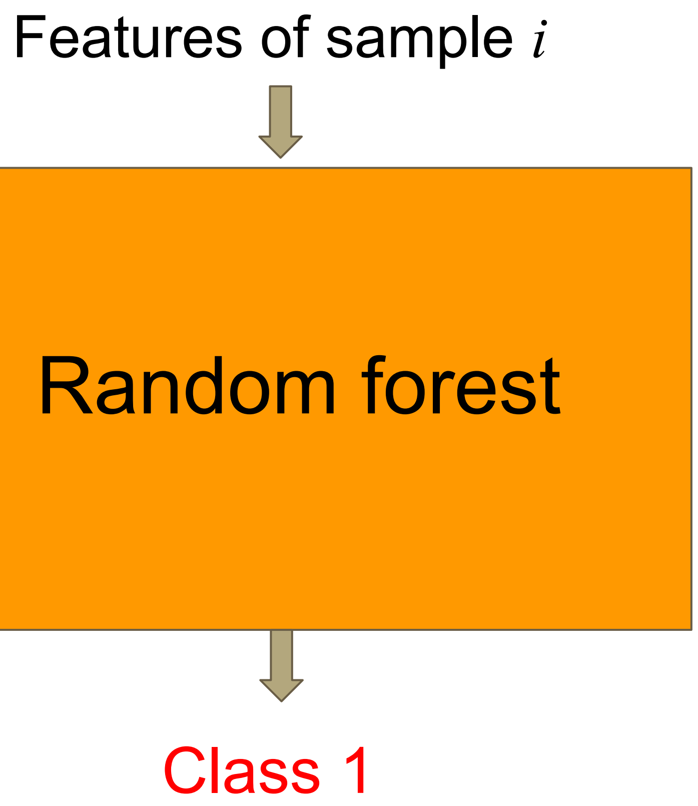


1 decision tree has high variance

Combining decision trees to build random forests



Combining decision trees to build random forests



Dataset for building a random forest model for

ID	Feature 1	Feature 2	Feature 3	...	Feature M	Class
1						1
2						1
3						2
4						1
						. . .
N-1						2
N						1

Bootstrapping (sampling with replacement)

Randomly selecting data points (IDs)

ID	Feature 1	Feature 2	Feature 3	...	Feature M	Class
1						1
2						1
3						2
4						1
						. . .
N-1						2
N						1

Random variable selection for identifying an optimal random forest

Randomly selecting columns (features) for building decision trees

ID	Feature 1	Feature 2	Feature 3	...	Feature M	Class
1						1
2						1
3						2
4						1
						. . .
N-1						2
N						1

Steps of building random forests

1) Bootstrapping (random sampling of data points with replacement)

2) Randomly selecting the features to build the decision tree

3) Repeat steps (1) and (2) to build multiple decision tree

4) Use majority vote of all the decision trees as the identified class for a given data point

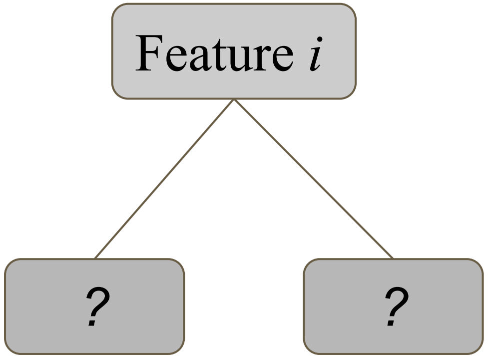
Adaboost

Important features of modeling using Adaboost

1) Using **stumps**
   a) Tree with only one node and two leaves
   b) Stumps are weak classifiers

2) Stumps are built in a sequential manner not in parallel
   a) Performance of one stump determines how the next stump is built

3) Stumps have different voting weights



This is a **stump**

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump
6) Calculate error of the selected stump as the sum of the weights of incorrectly classified samples

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump
6) Calculate error of the selected stump as the sum of the weights of incorrectly classified samples
7) Calculating voting weight of the selected stump $VW = \frac{1}{2} \frac{1-(error+\epsilon)}{error+\epsilon}$

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump
6) Calculate error of the selected stump as the sum of the weights of incorrectly classified samples
7) Calculating voting weight of the selected stump $VW = \frac{1}{2}\frac{1-(error+\epsilon)}{error+\epsilon}$
8) Increasing weights of incorrectly classified datapoints $W_{new} = W * e^{VW}$

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump
6) Calculate error of the selected stump as the sum of the weights of incorrectly classified samples
7) Calculating voting weight of the selected stump $VW = \frac{1}{2}\frac{1-(error+\epsilon)}{error+\epsilon}$
8) Increasing weights of incorrectly classified datapoints $W_{new} = W * e^{VW}$
9) Decreasing weights of correctly classified datapoints $W_{new} = W * e^{-VW}$

Steps of building an Adaboost model

1) Consider same weight for all datapoints (normalized to add up to 1)
2) Making stumps with all individual features
3) Calculate Gini index for each stump
4) Order the stumps based on their Gini indices
5) Select the best stump
6) Calculate error of the selected stump as the sum of the weights of incorrectly classified samples
7) Calculating voting weight of the selected stump $VW = \frac{1}{2} \frac{1-(error+\epsilon)}{error+\epsilon}$
8) Increasing weights of incorrectly classified datapoints $W_{new} = W * e^{VW}$
9) Decreasing weights of correctly classified datapoints $W_{new} = W * e^{-VW}$
10) Normalize the weights to add up to 1
11) Repeat steps 2 to 10 using the new sample weights

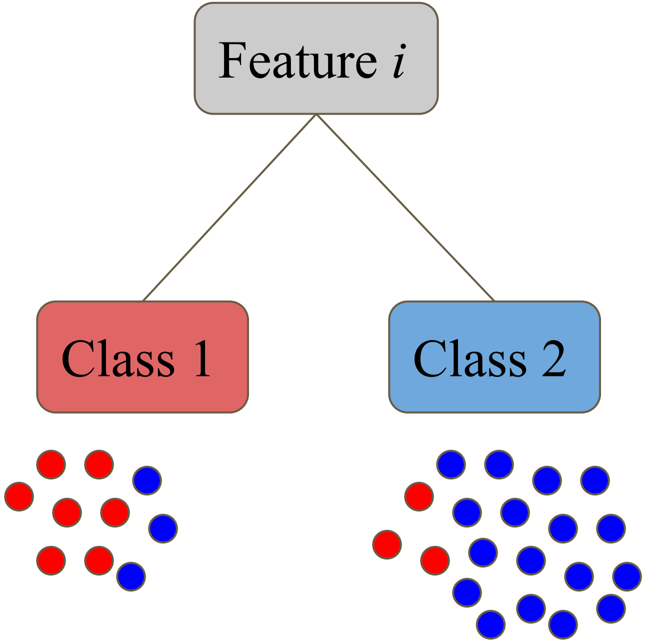
How to calculate weighted Gini index

Let's assess impurity of the classes:

$$Gini = 1 - \Sigma_{i=1}^{C} P_i$$

C: total number of classes

$$P_i = \frac{N_i}{\Sigma_{i=1}^{C} N_i} \Longrightarrow P_i = \frac{\Sigma_{j=1}^{N_i} W_j}{\Sigma_{i=1}^{C} \Sigma_{j=1}^{N_i} W_j}$$



Gradient Boosting Method (GBM)

Important features of Gradient Boosting

- Gradient Boosting Method (GBM) is used for continuous value prediction
    - Technically it is a regression model by default

- Although it is a regression model, it can be used for classification
- It starts by a single leaf (as the initial guess of all samples), then a tree is built
    - Similar to Adaboost, a tree is built relying on the error of the previous tree
    - Although the tree size is restricted, it is not necessarily a stump (like in Adaboost)
    - GBM scales the trees by the same amount
- GBM continues building trees up until
    - Specific number of trees, that we determined
    - Or additional trees does not improve the model

Extra useful information

Useful links

Installation instructions

- scikit-learn
- Anaconda distribution of Python
- IPython

Data Sets

- scikit-learn DataSet

scikit-learn: machine learning in Python :

- https://scikit-learn.org/stable/

Useful cheat sheets:

- https://www.analyticsvidhya.com/blog/2017/02/top-28-cheat-sheets-for-machine-learning-data-science-probability-sql-big-data/

Thanks