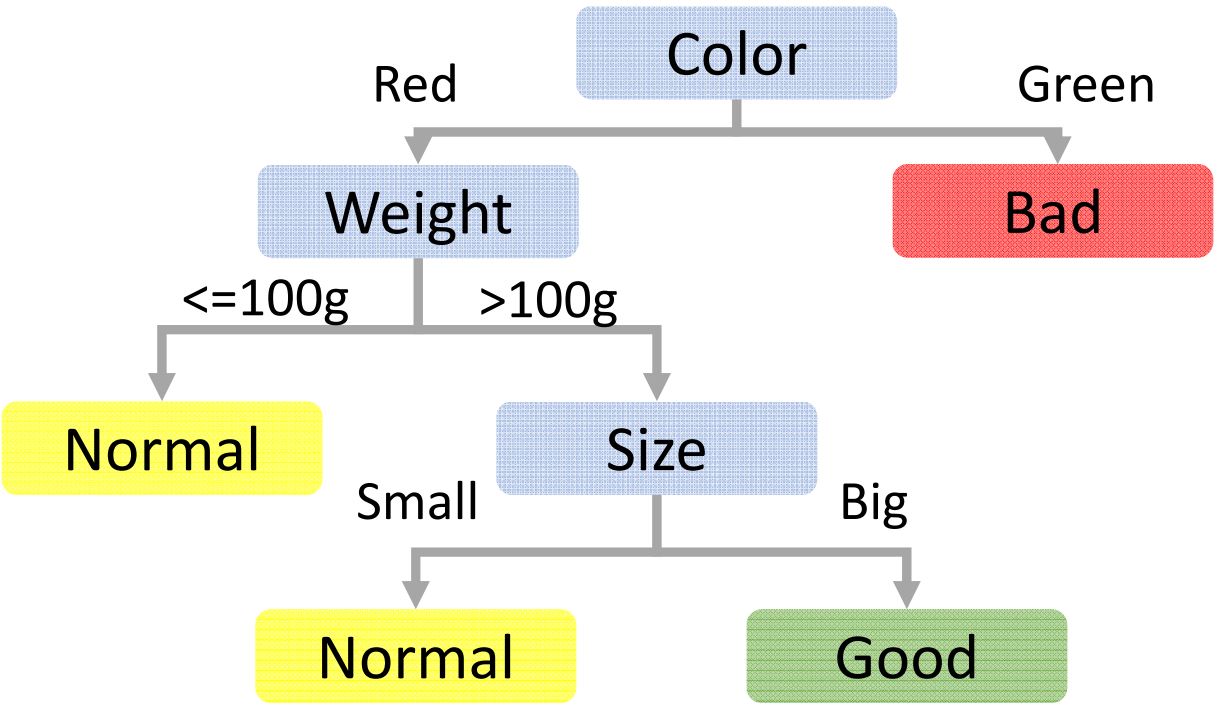
Decision Tree

Yun-Nan Chang

1 Basic

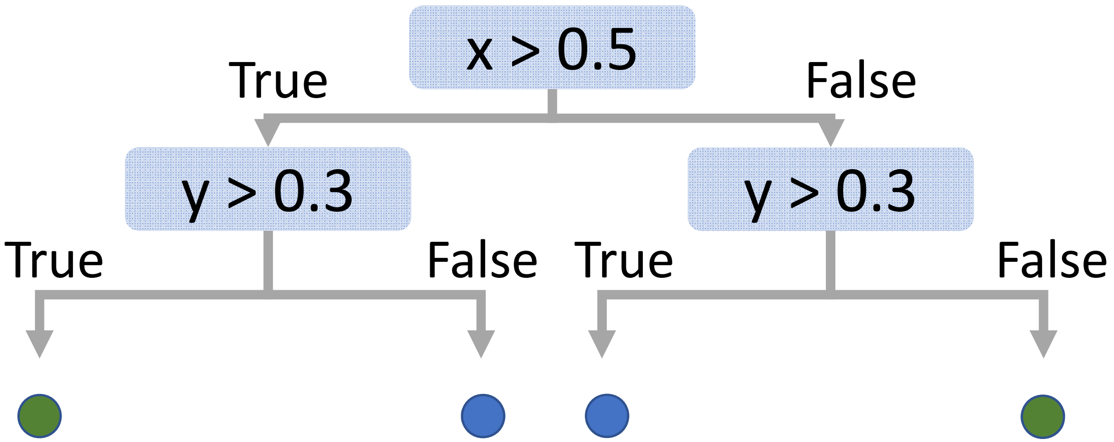
Tree Structure

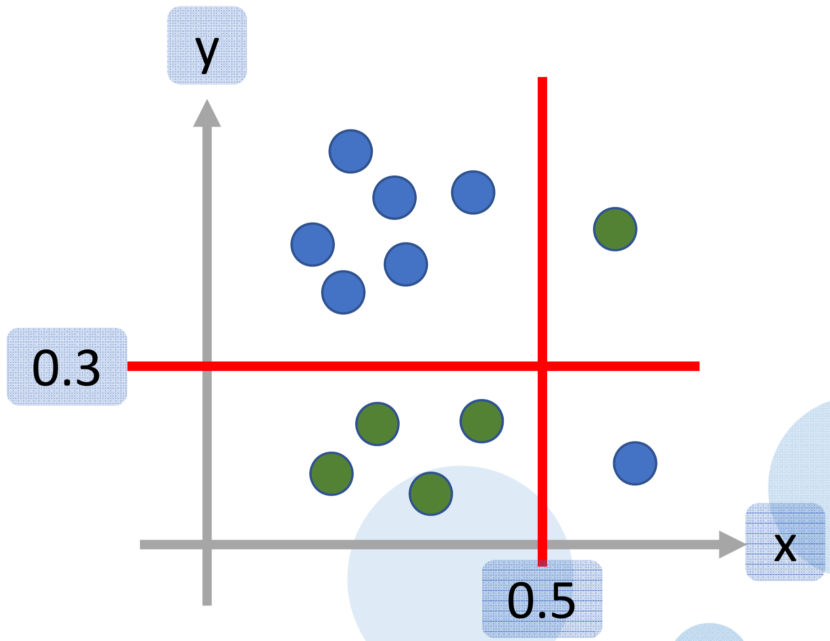
◆Prediction based on some tree structure



Tree Structure

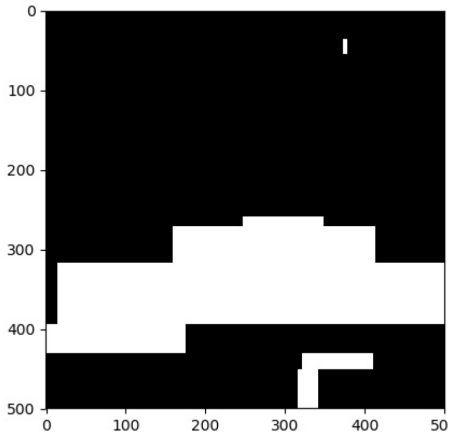
- Prediction based on some tree structure



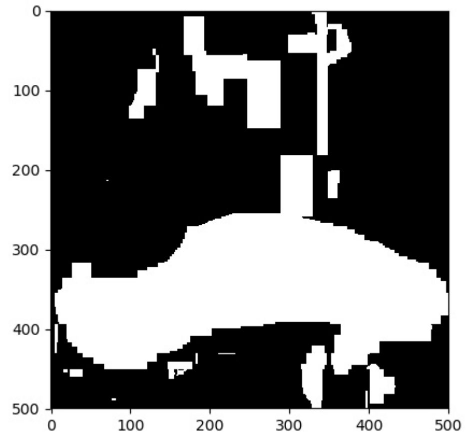


Function of NSYSU logo

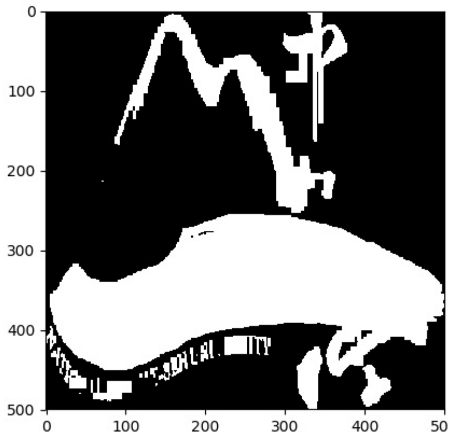
Depth = 5



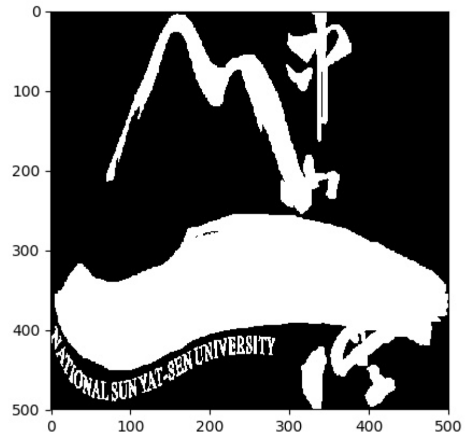
Depth = 10



Depth = 15



Depth = 30



Learning Flowchart of the decision tree

Algorithm 1 决策树学习基本算法

输入:

- 训练集 $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$;
- 属性集 $A = \{a_1, \ldots, a_d\}$.

过程: 函数 TreeGenerate(D, A)

生成结点 node;
if D 中样本全属于同一类别 C **then**
  将 node 标记为 C 类叶结点; **return**
end if
if $A = \emptyset$ **OR** D 中样本在 A 上取值相同 **then**
  将 node 标记叶结点，其类别标记为 D 中样本数最多的类; **return**
end if
从 A 中选择最优划分属性 a_*;
for a_* 的每一个值 a_*^v **do**
  为 node 生成每一个分枝; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
  if D_v 为空 **then**
    将分枝结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
  else
    以 TreeGenerate$(D_v, A \setminus \{a_*\})$ 为分枝结点
  end if
end for

输出: 以 node 为根结点的一棵决策树

1 All the samples belong to the same class.

2 No test feature left or all the samples have the same feature.

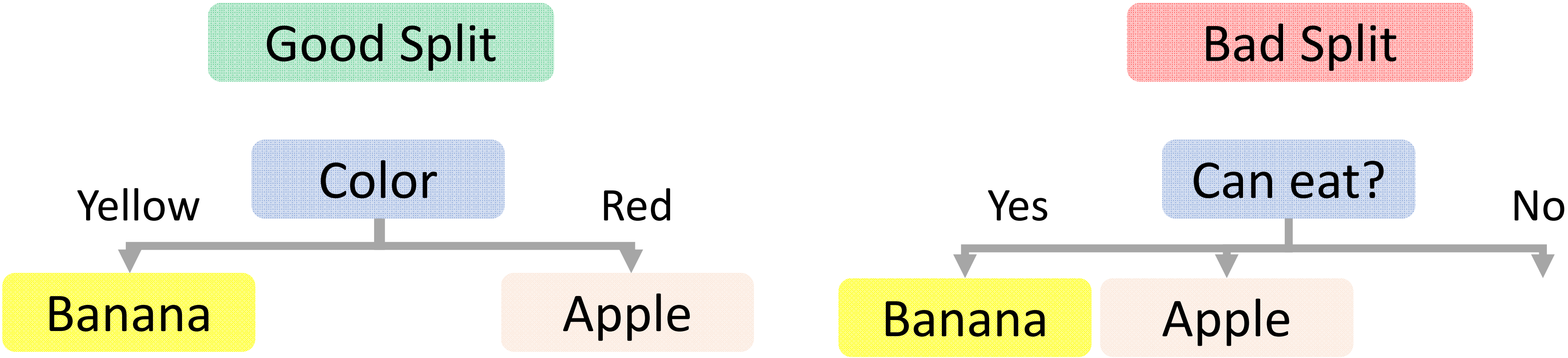
3 There is no sample in the current node.

2

Selection of split feature

Split

◇We want to split can make 'purity' more and more higher.



Entropy

- Is there any metric used to evaluate the decision model?
  - Precision of the training data?
- Classic metrics used to measure the selection of the attribute
  - Information Gain 信息增益
  - Gain Ratio 增益率
  - Gini factor 基尼指數
- Any function satisfying the following conditions can be used to measure the impurity of a split
  - $\emptyset\left(\frac{1}{2},\frac{1}{2}\right) \geq \emptyset\,(p, 1-p)$
  - $\emptyset\,(1,0) = \emptyset\,(0,1)$=0
  - $\emptyset\,(p, 1-p)$ is increasing in p on [0,1/2], decreasing on [1/2,1]
- We can choose the attribute which entropy is lowest.

$$Entropy(D) = -\sum_{k=1}^{|y|} p_k log_2 p_k$$

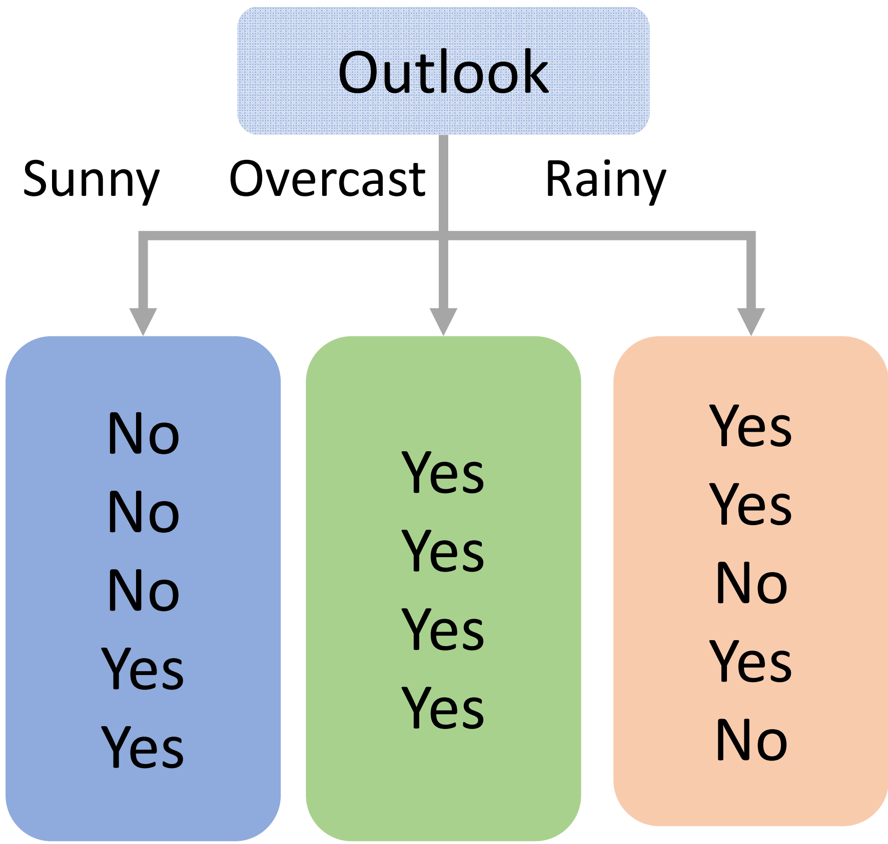
Entropy example

- Weather data
- Should I play today?

$$Entropy(D) = -\sum_{k=1}^{|y|} p_k log_2 p_k$$

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Entropy example



Outlook	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Entropy example

Outlook = Sunny

$$Entropy(D) = -\frac{2}{5} * \log\left(\frac{2}{5}\right) - \frac{3}{5} * \log\left(\frac{3}{5}\right) = 0.971$$

Outlook = Overcast

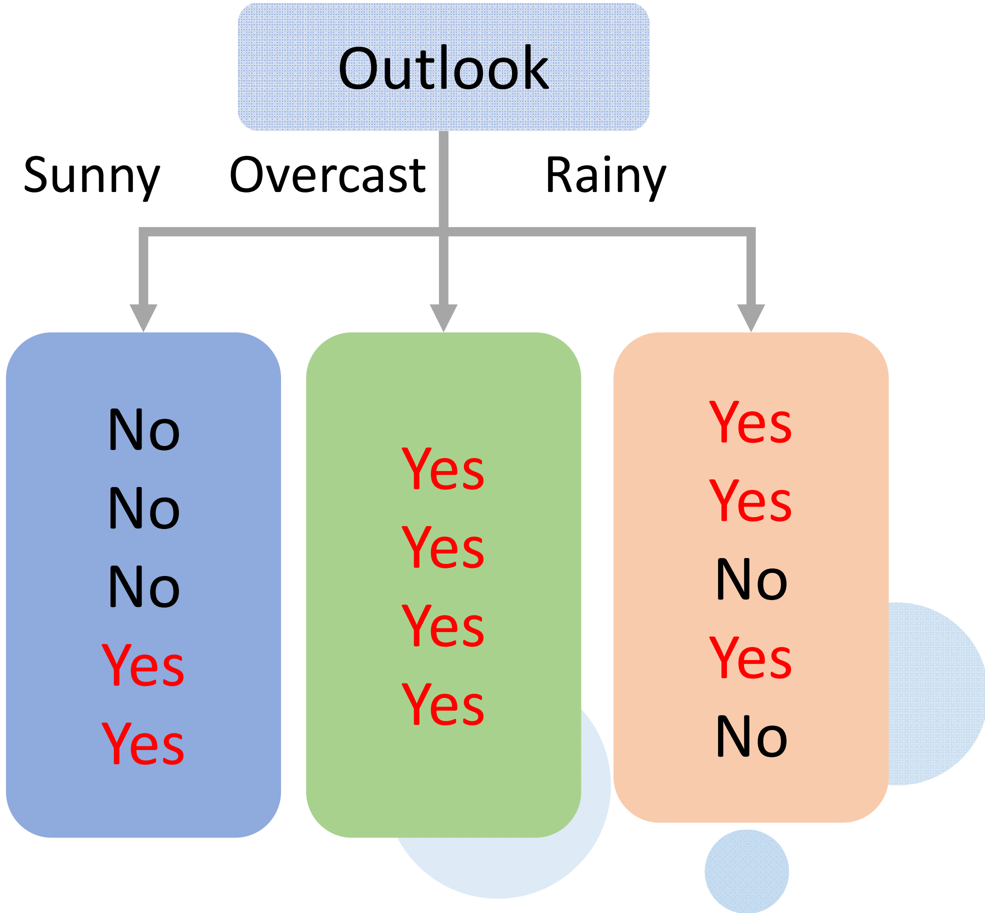
$$Entropy(D) = -\frac{4}{4} * \log\left(\frac{4}{4}\right) - \frac{0}{4} * \log\left(\frac{0}{4}\right) = 0$$

Outlook = Rainy

$$Entropy(D) = -\frac{3}{5} * \log\left(\frac{3}{5}\right) - \frac{2}{5} * \log\left(\frac{2}{5}\right) = 0.971$$

Expected info

$$0.971 * \left(\frac{5}{14}\right) + 0 * \left(\frac{4}{14}\right) + 0.971 * \left(\frac{5}{14}\right) = 0.693$$



Entropy example

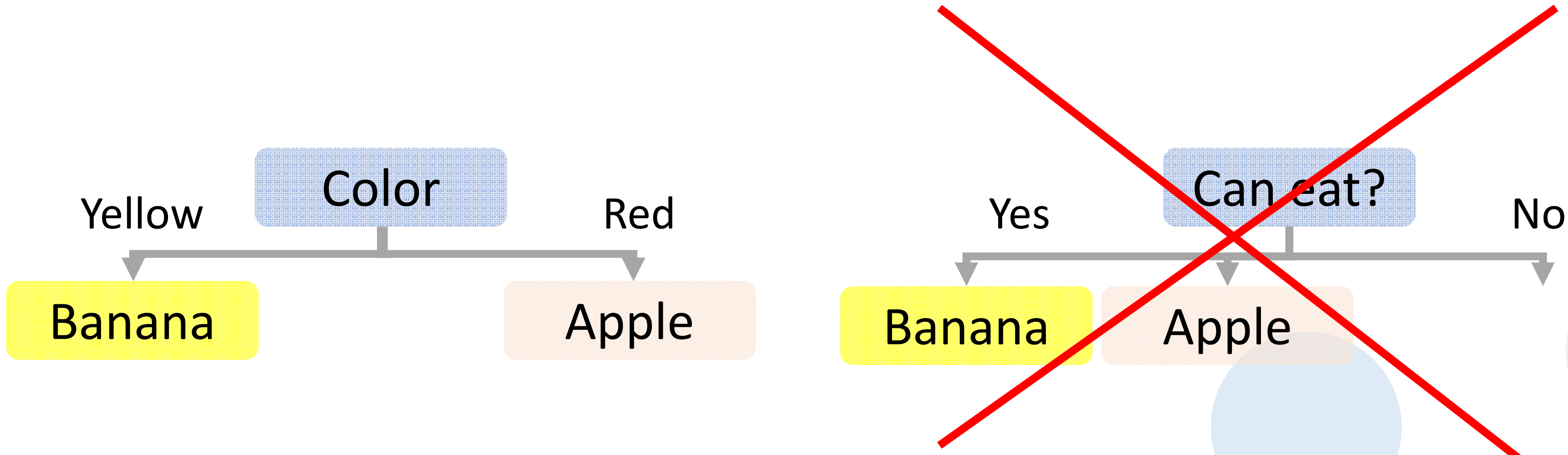
- Calculate the entropy of all attributes.

Outlook	0.693
Temperature	0.744
Humidity	0.788
Windy	0.899

Lowest

Information gain

- We want to get greatest Information Gain when we create a tree.
- In this case, if we use ‘can eat or not?’ to classify, it’s not helpful.



Information gain

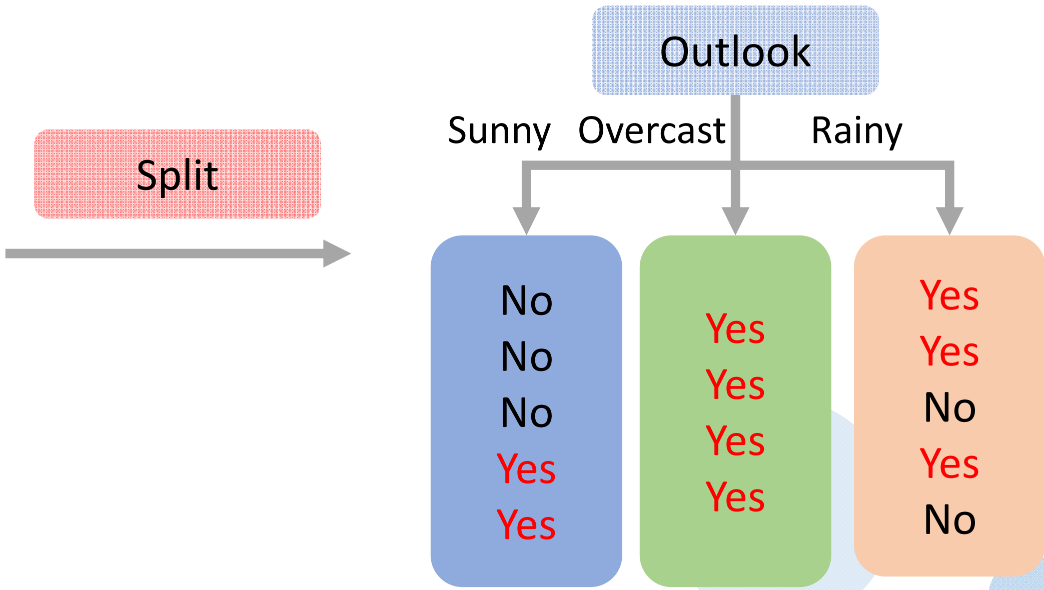
- Greater the information gain better the purity of a node

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^V|}{|\mathrm{D}|} Ent(D^V)$$

Information gain

- Information Gain = Information before split - Information after split.
- Example : Gain(Outlook) = Info(Origin) – info(Outlook)

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Information gain

- Information Gain = Information before split - Information after split.

Entropy	
Origin	0.940
Outlook	0.693
Temperature	0.744
Humidity	0.788
Windy	0.899

Information Gain	
Outlook	0.247
Temperature	0.196
Humidity	0.152
Windy	0.048

Gini factor

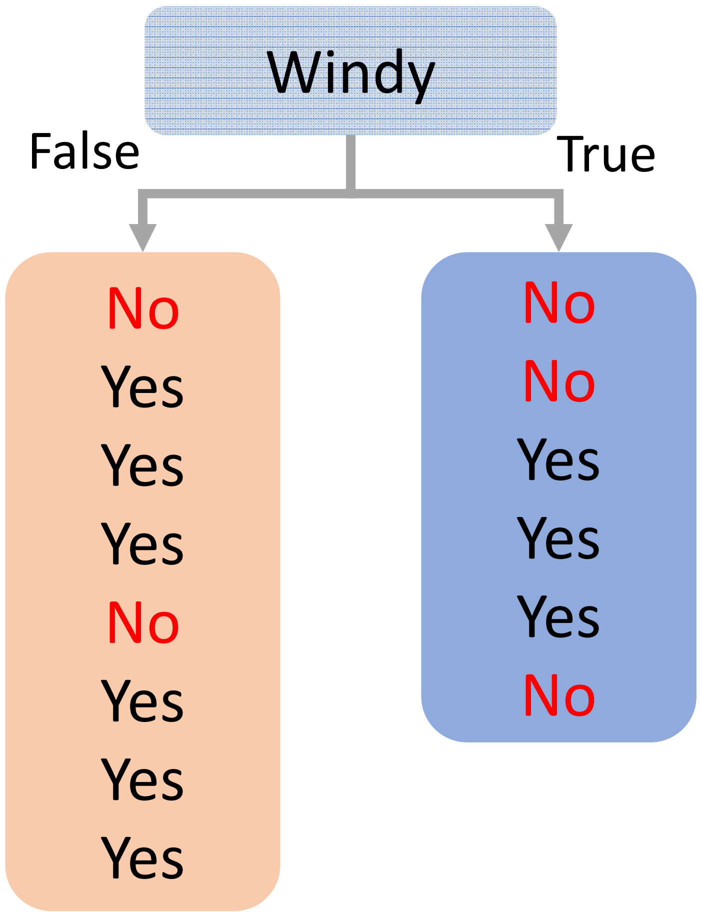
- Gini factor is smaller, dataset is purer.

- CART use Gini factor.

$$\begin{aligned}\mathrm{Gini}(D) &= \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} \\ &= 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 \ .\end{aligned}$$

$$\mathrm{Gini\_index}(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v) \ .$$

Gini factor example



Windy	Play
False	No
True	No
False	Yes
False	Yes
False	Yes
True	No
True	Yes
False	No
False	Yes
False	Yes
True	Yes
True	Yes
False	Yes
True	No

Gini factor example

Windy = False

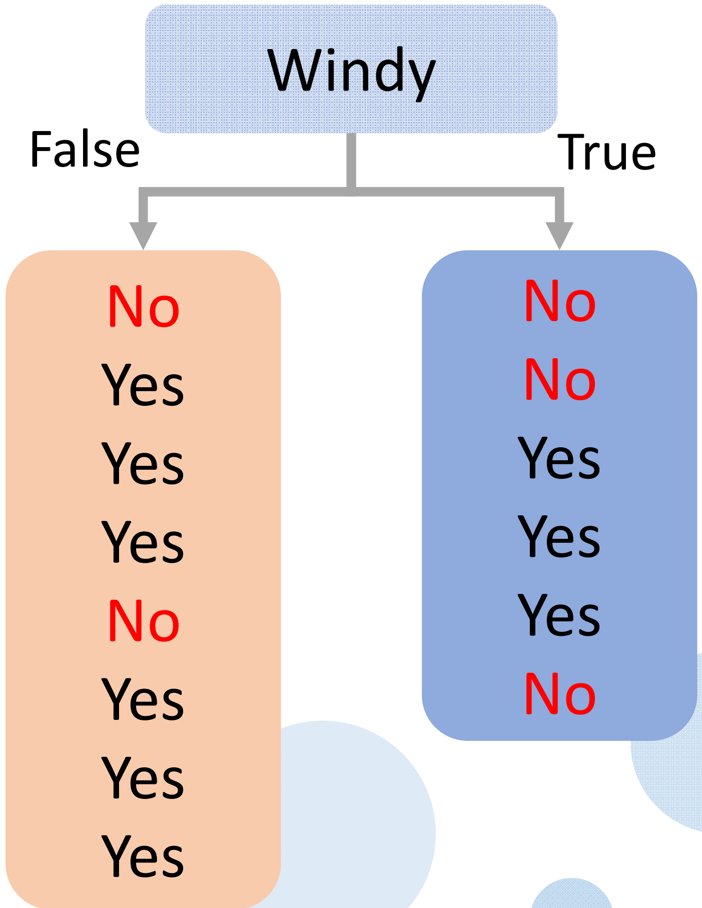
$$Gini(D) = 1 - (\frac{2^2}{8} + \frac{6^2}{8}) = 0.375$$

Windy = True

$$Gini(D) = 1 - (\frac{3^2}{6} + \frac{3^2}{6}) = 0.5$$

Expected info

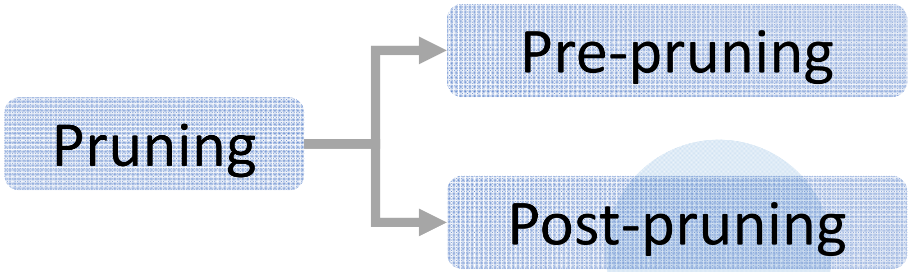
$$0.375 * \left(\frac{8}{14}\right) + 0.5 * \left(\frac{6}{14}\right) = 0.428$$



3 Pruning

Why we Pruning?

- “Pruning” is used in decision learning to tackle “overfitting”
- Reduce the number of splits through pruning in order to avoid the overfitting based on some specific attributes of the data



Validation Sets

- We can use validation sets to judges whether the generalization performance of the decision tree is improved.

sepal length	sepal width	petal length	petal width	class
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.0	3.2	4.7	1.4	*versicolor*
6.4	3.2	4.5	1.5	*versicolor*
6.9	3.1	4.9	1.5	*versicolor*
5.5	2.3	4.0	1.3	*versicolor*
6.5	2.8	4.6	1.5	*versicolor*

Train

sepal length	sepal width	petal length	petal width	class
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
7.0	3.2	4.7	1.4	*versicolor*
6.4	3.2	4.5	1.5	*versicolor*
6.9	3.1	4.9	1.5	*versicolor*

Validation

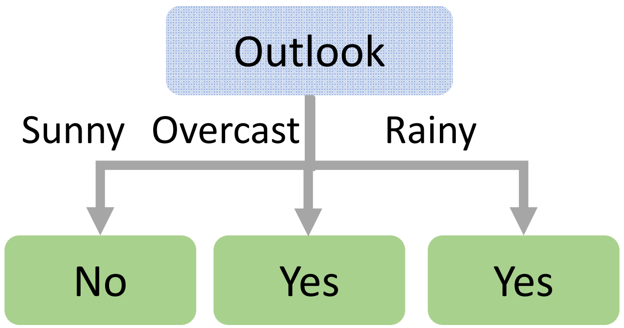
sepal length	sepal width	petal length	petal width	class
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
5.5	2.3	4.0	1.3	*versicolor*
6.5	2.8	4.6	1.5	*versicolor*

Pre-pruning

- Estimate before splitting node.
- Base on information gain.

Pre-pruning

- Estimate before splitting node.
- Base on information gain.



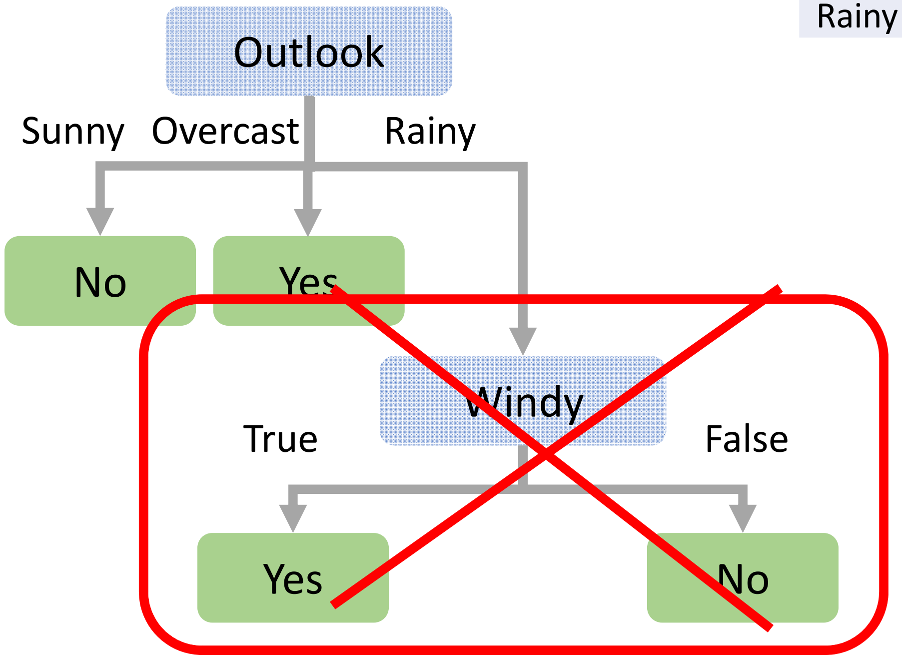
Validation set

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No

Acc : 0.83

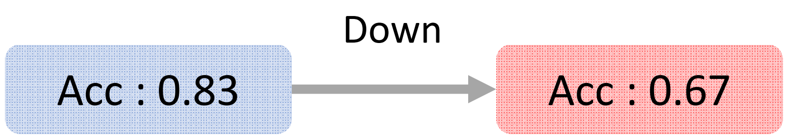
Pre-pruning

- Split new node, but validation set accuracy is worse than before.
- So we don't split this node.



Validation set

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No



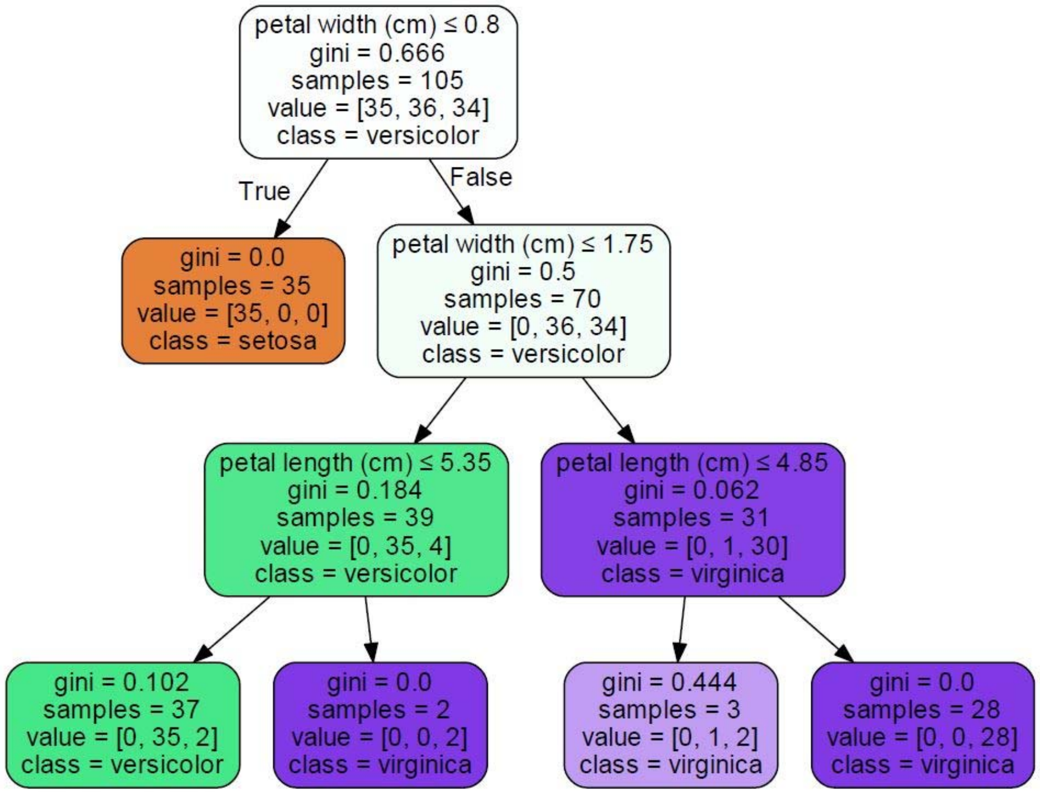
Pre-pruning pro and cons

- **Pro**
  - Reduce over-fitting risk
  - Reduce training time
- **Cons**
  - Under-fitting risk

Post-pruning

- Pruning after learning.
- If delete some nodes we can get better result, delete them.
- Bottom-up

Original tree

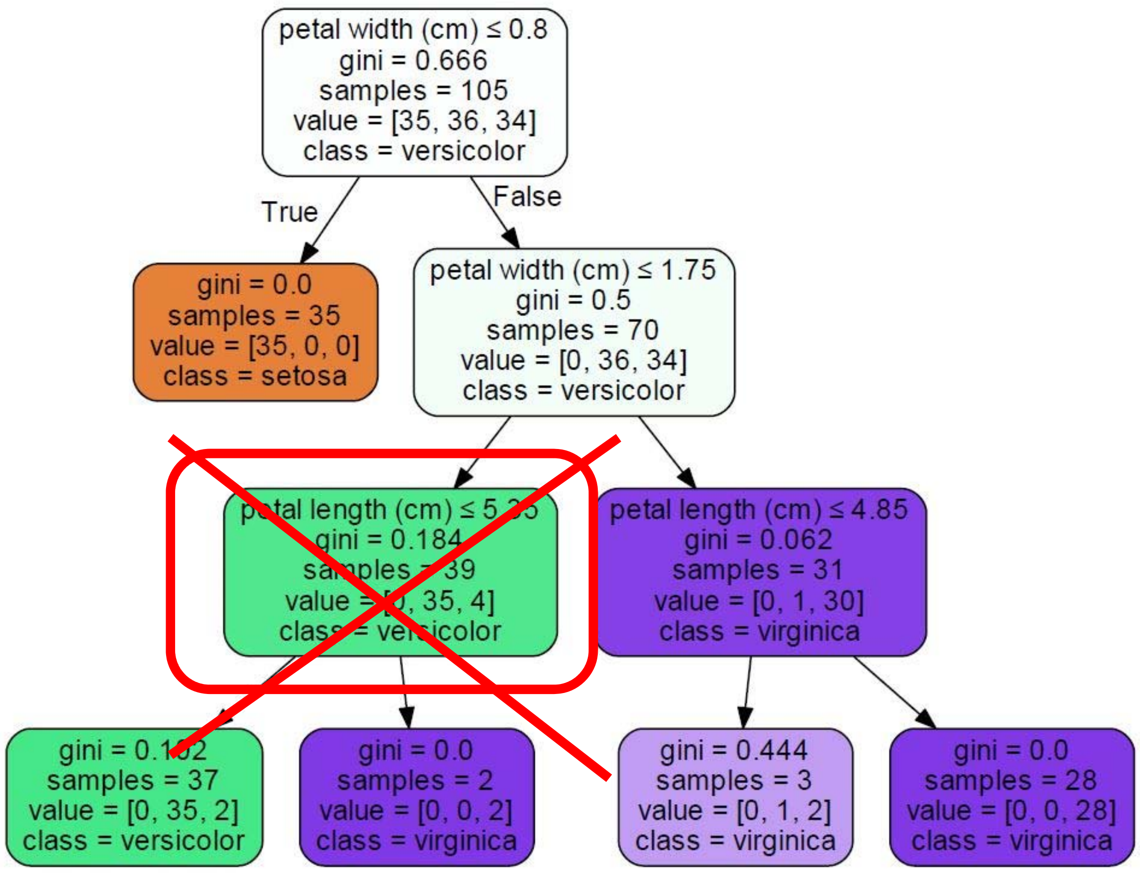


Post-pruning

Down

Acc : 0.83 → Acc : 0.67

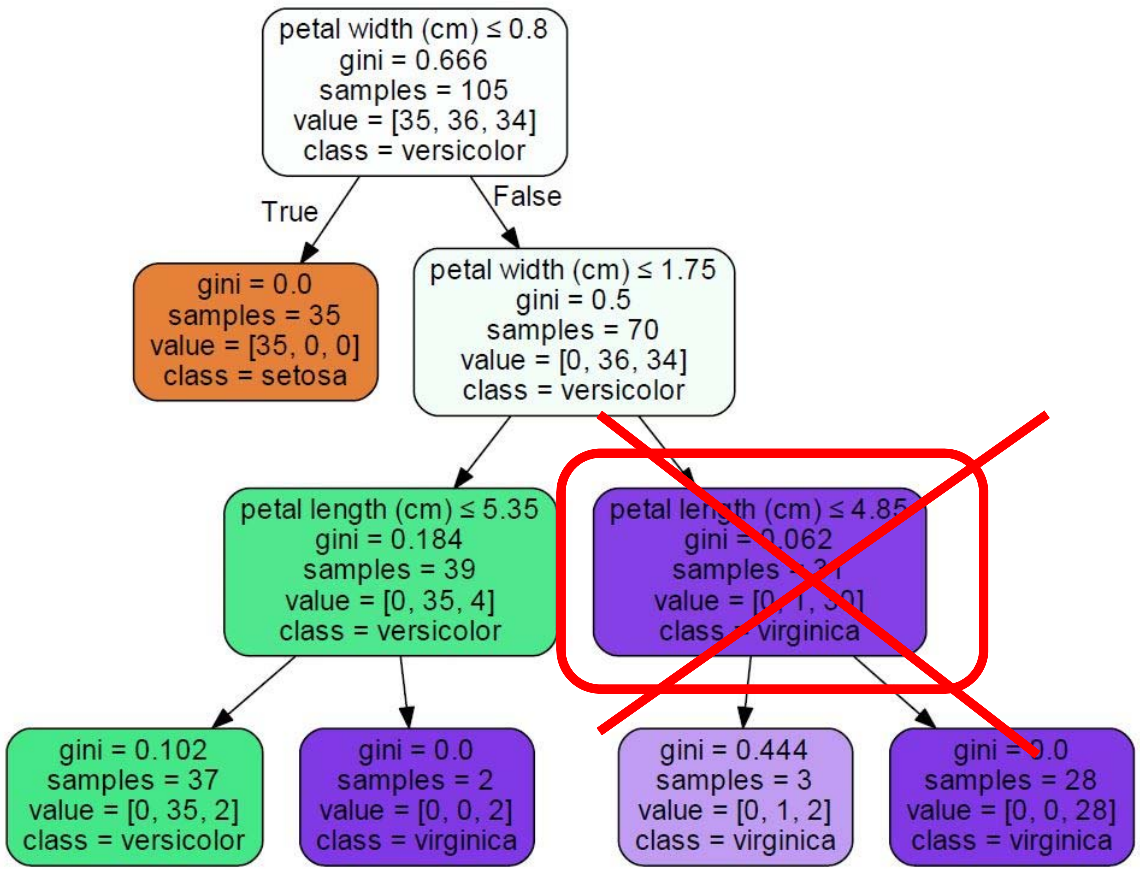
Don't remove



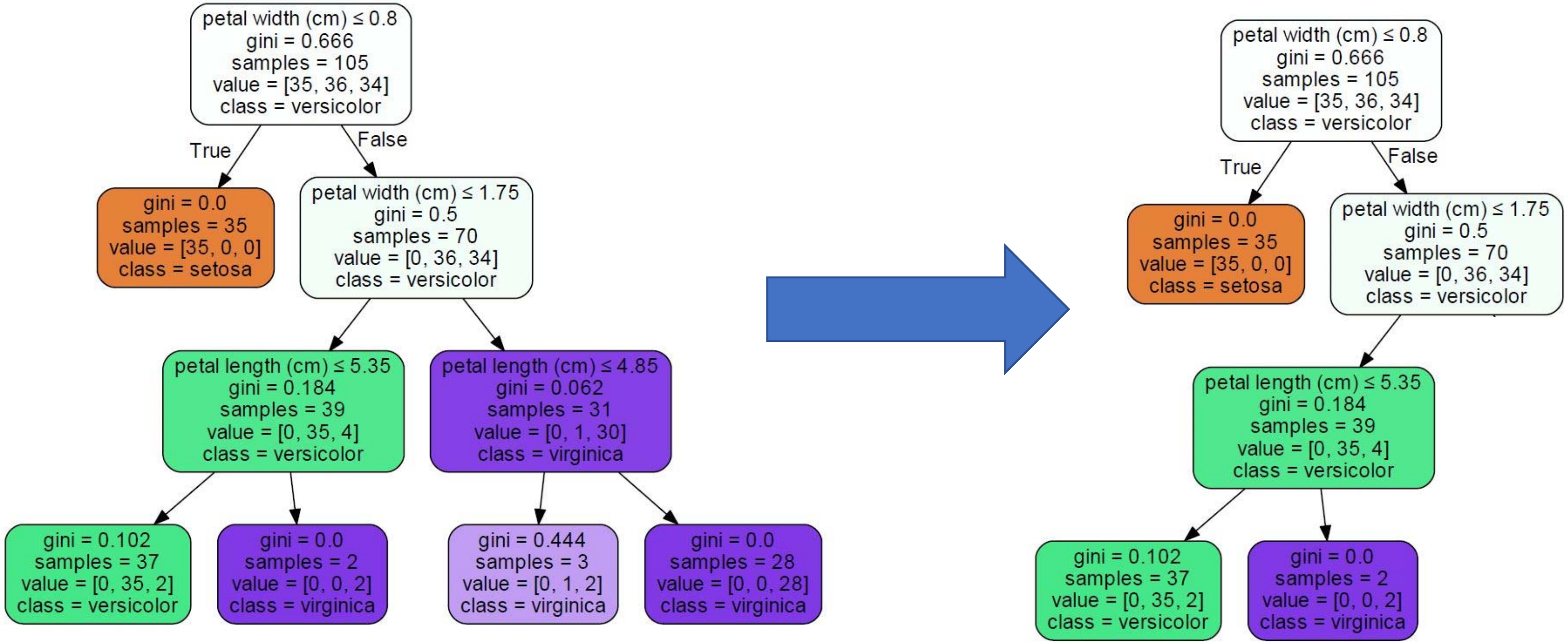
Post-pruning

Acc : 0.83 → Up → Acc : 0.85

Remove it



Post-pruning



Post-pruning pro and cons

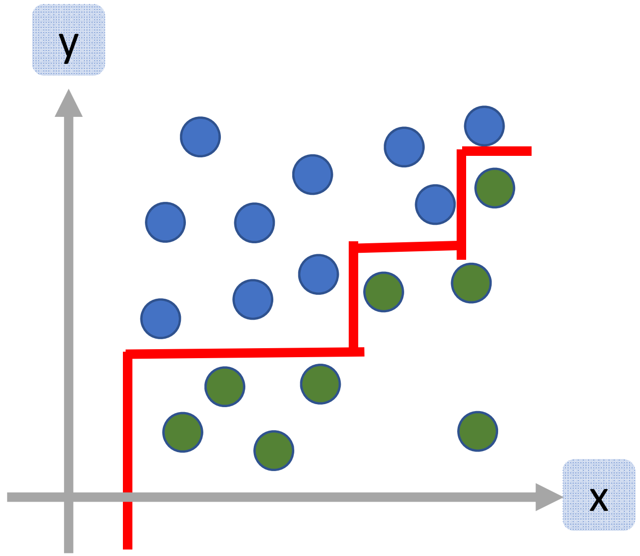
- **Pro**
  - Low risk of under-fitting
- **Cons**
  - Increase training time

4 Multivariate Trees

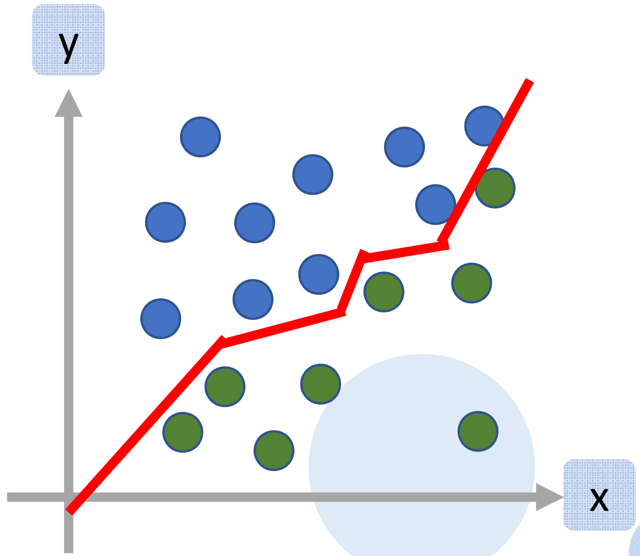
Univariate and Multivariate Trees

- Univariate decision tree classification boundary: axis parallel
- Multivariate decision tree classification boundary: Not necessarily

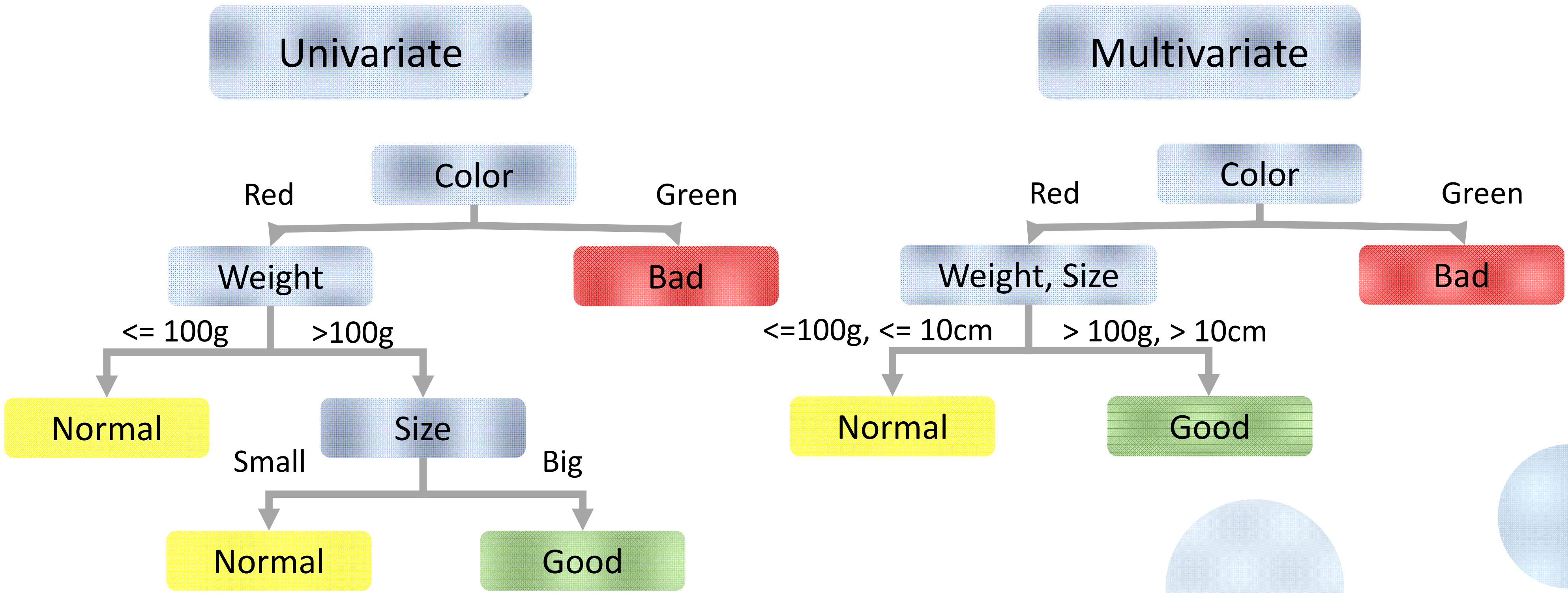
Univariate



Multivariate



Univariate and Multivariate Trees



5 Parameter

Decision tree in sklearn

- from sklearn.tree import DecisionTreeClassifier

- clf = DecisionTreeClassifier(max_depth=10)
- clf.fit(data, target)
- predict = clf.predict(test_data)

Decision tree parameters in sklearn

criterion	splitter	max_depth
min_samples_split	min_samples_leaf	min_weight_fraction_leaf
max_features	random_state	max_leaf_nodes
min_impurity_decrease	min_impurity_split	class_weight

Information Gain

- criterion='gini', default
- Supported criteria are
  - "gini" for the Gini impurity
  - "entropy" for the information gain

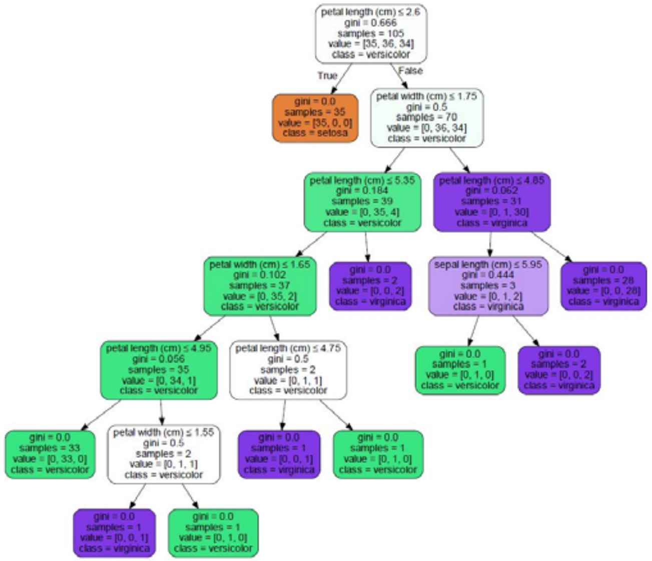
criterion

Avoid over-fitting

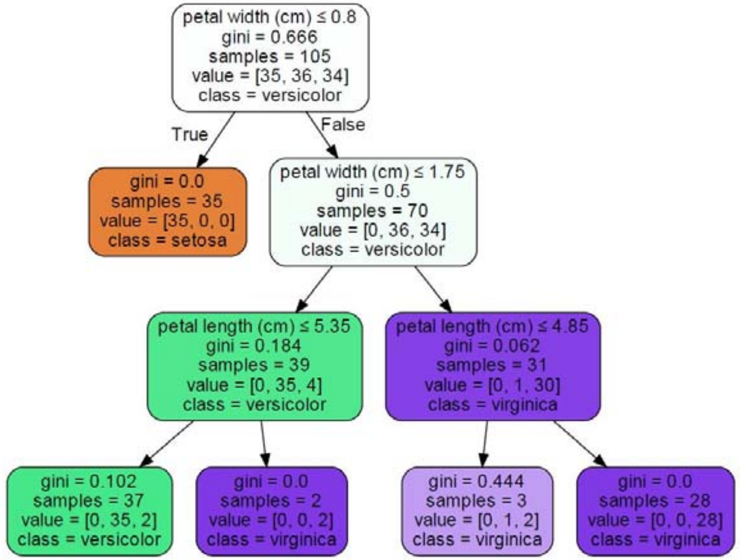
- max_depth
- min_samples_split
- max_features
- min_impurity_decrease
- min_samples_leaf
- max_leaf_nodes
- min_impurity_split

max_depth

max_depth = None



max_depth = 3



5 Ensemble learning

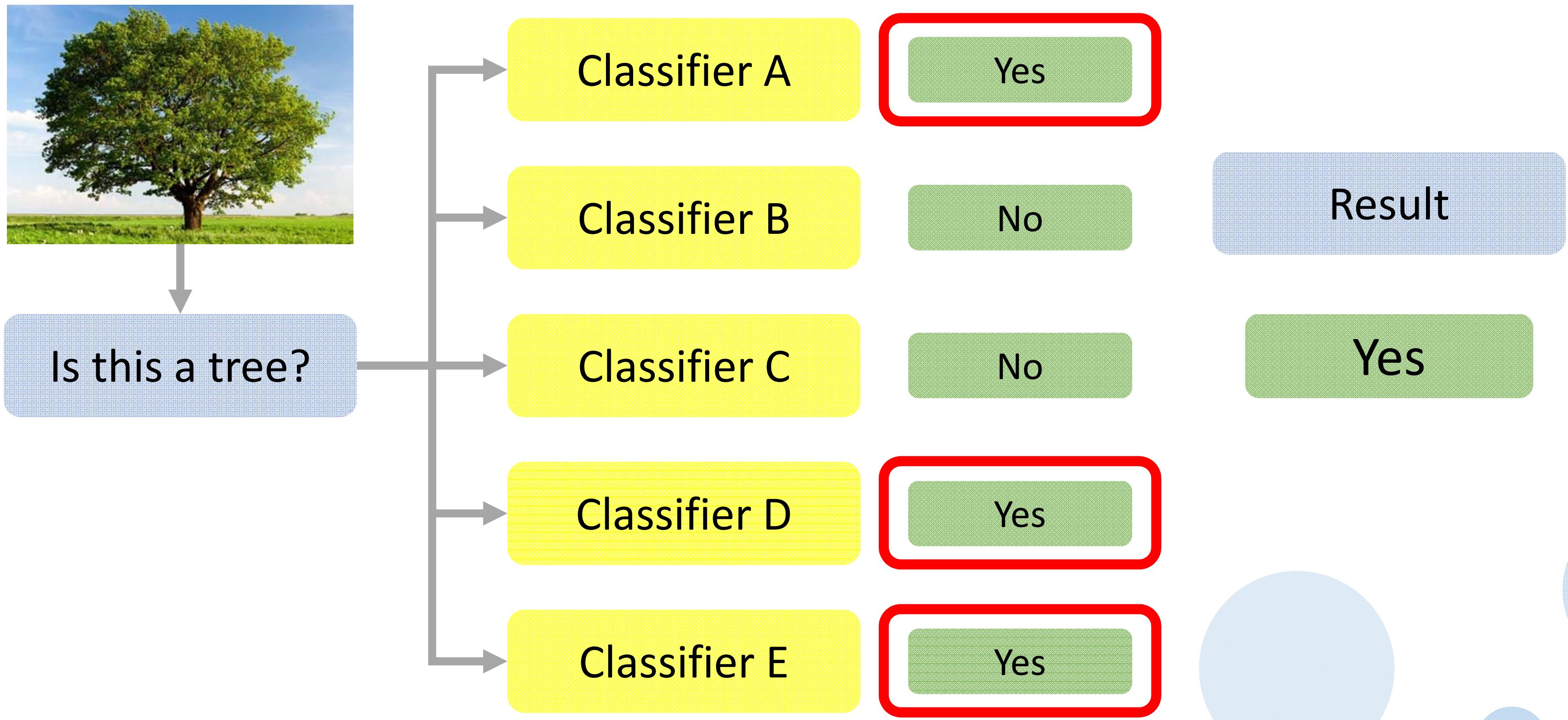
Ensemble learning

- Use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

Why use Ensemble learning?

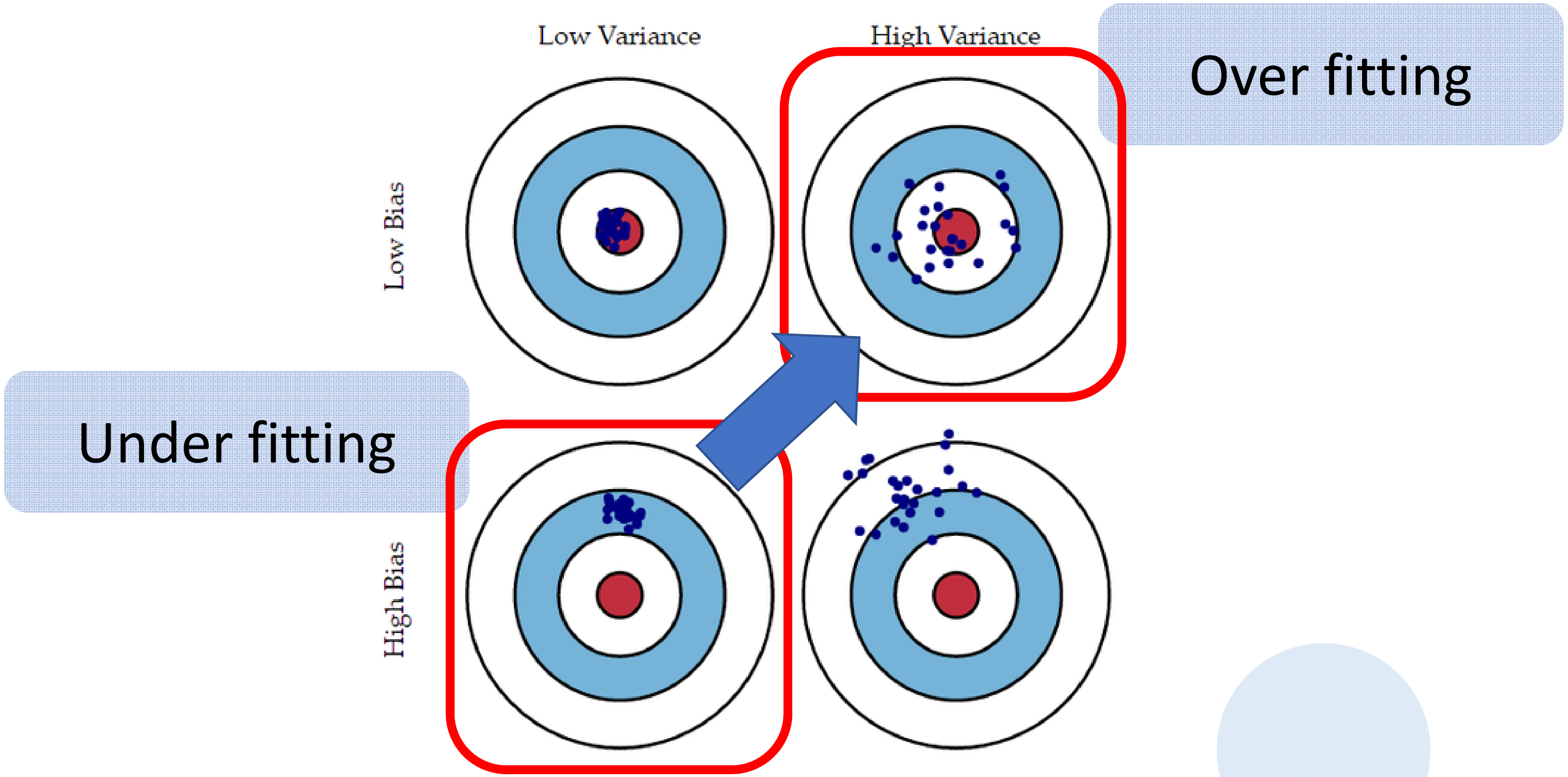
- One classifier may not get result good enough.
- Use many classifier to obtain better predictive performance.
- Many hands make light work.

Ensemble learning



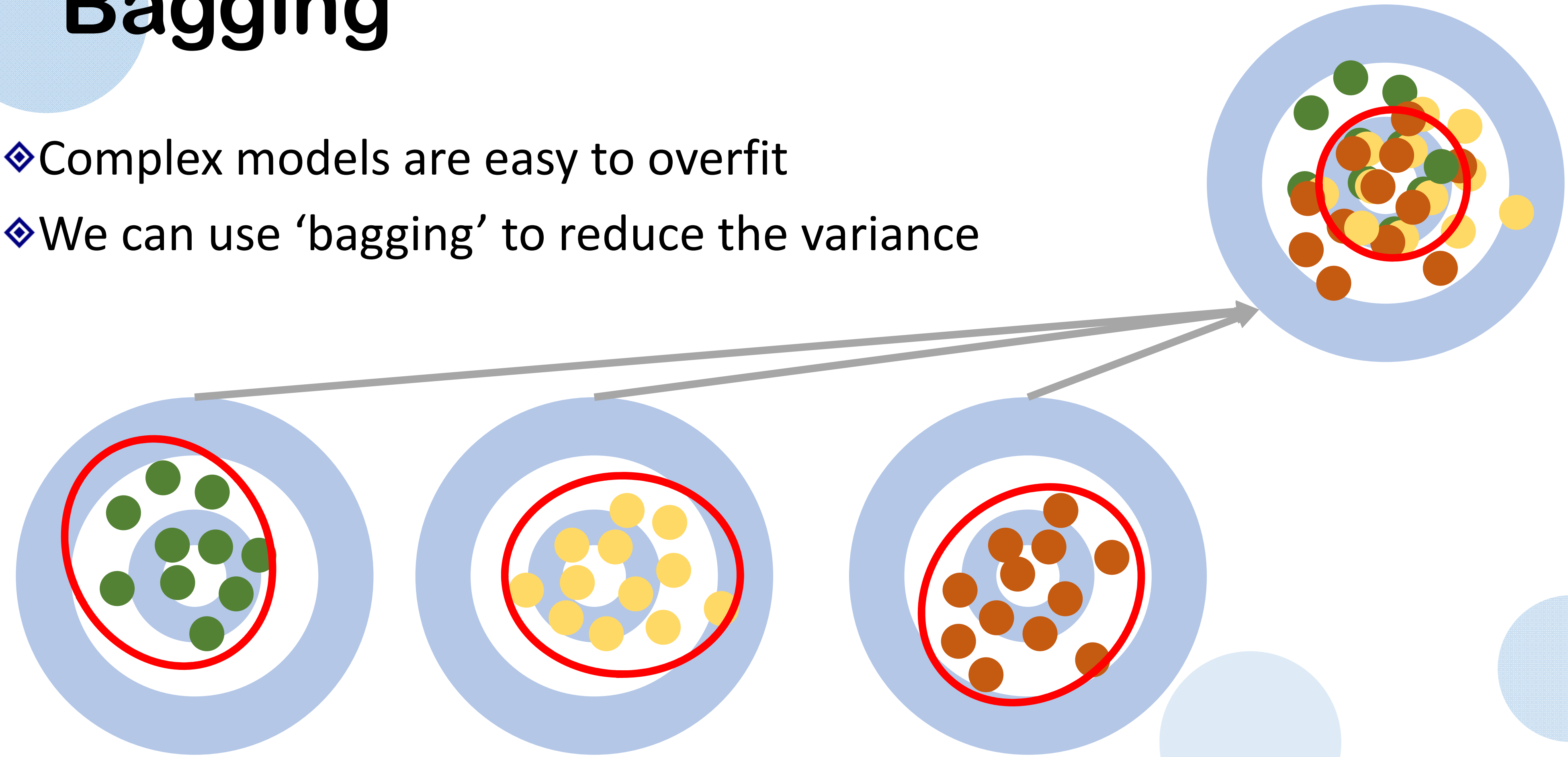
6 Bagging

Bias and Variance

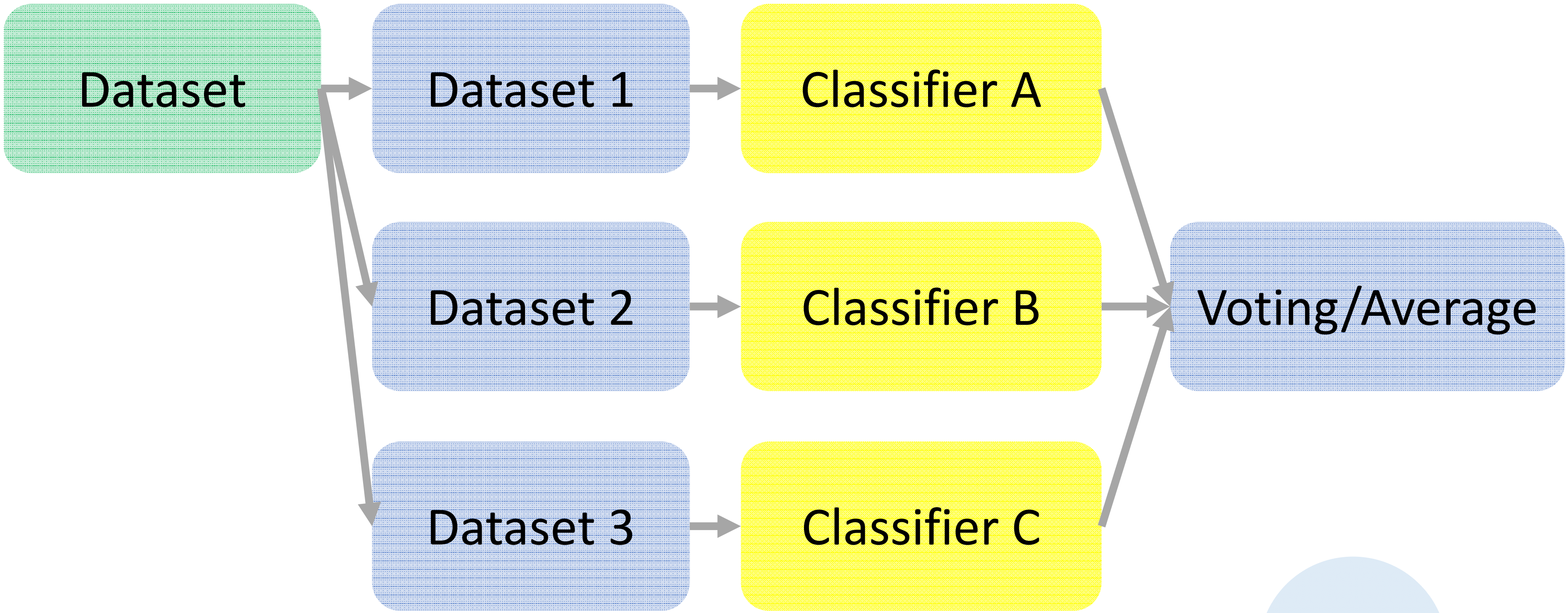


Bagging

- Complex models are easy to overfit
- We can use ‘bagging’ to reduce the variance



Bagging Train



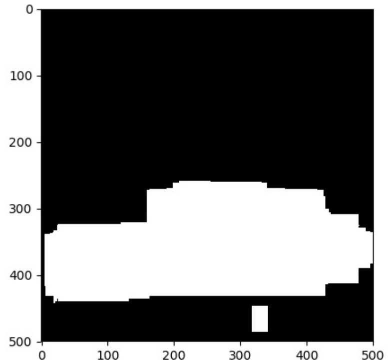
Ever dataset 1 ~ N can have same data.

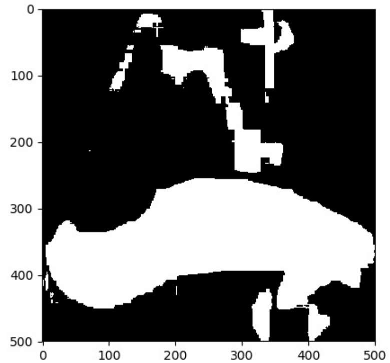
Function of NSYSU logo

Random Forest

100 trees

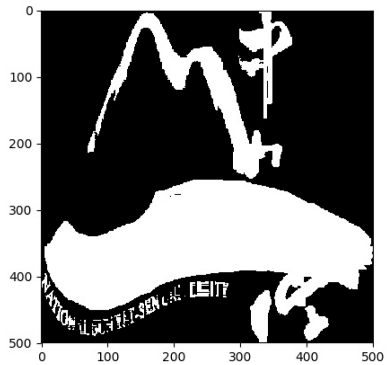
Depth = 5

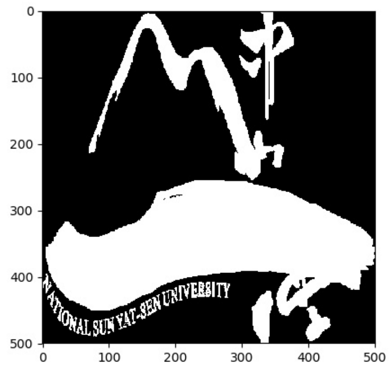




Depth = 10

Depth = 15





Depth = 20

7 Boosting

Boosting

Training data:
$\{(x^1,\hat{y}^1),\cdots,(x^n,\hat{y}^n),\cdots,(x^N,\hat{y}^N)\}$
$\hat{y}=\pm 1$ (binary classification)

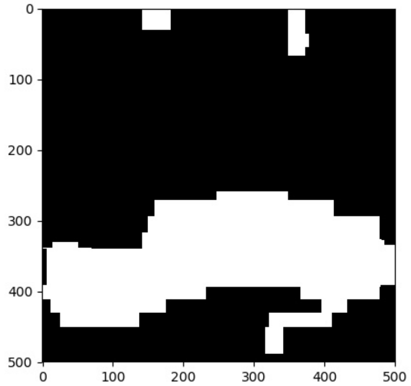
- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers
- The classifiers are learned sequentially.

Function of NSYSU logo

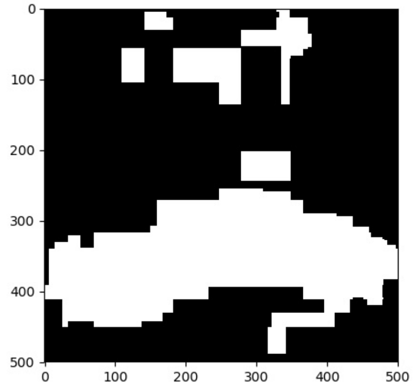
Adaboost + Decision Tree

Depth = 5

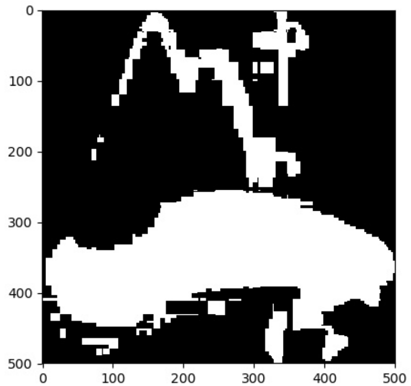
N = 3



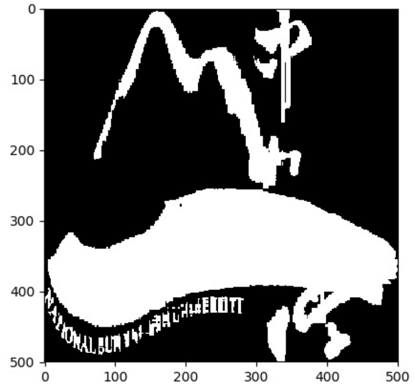
N = 5



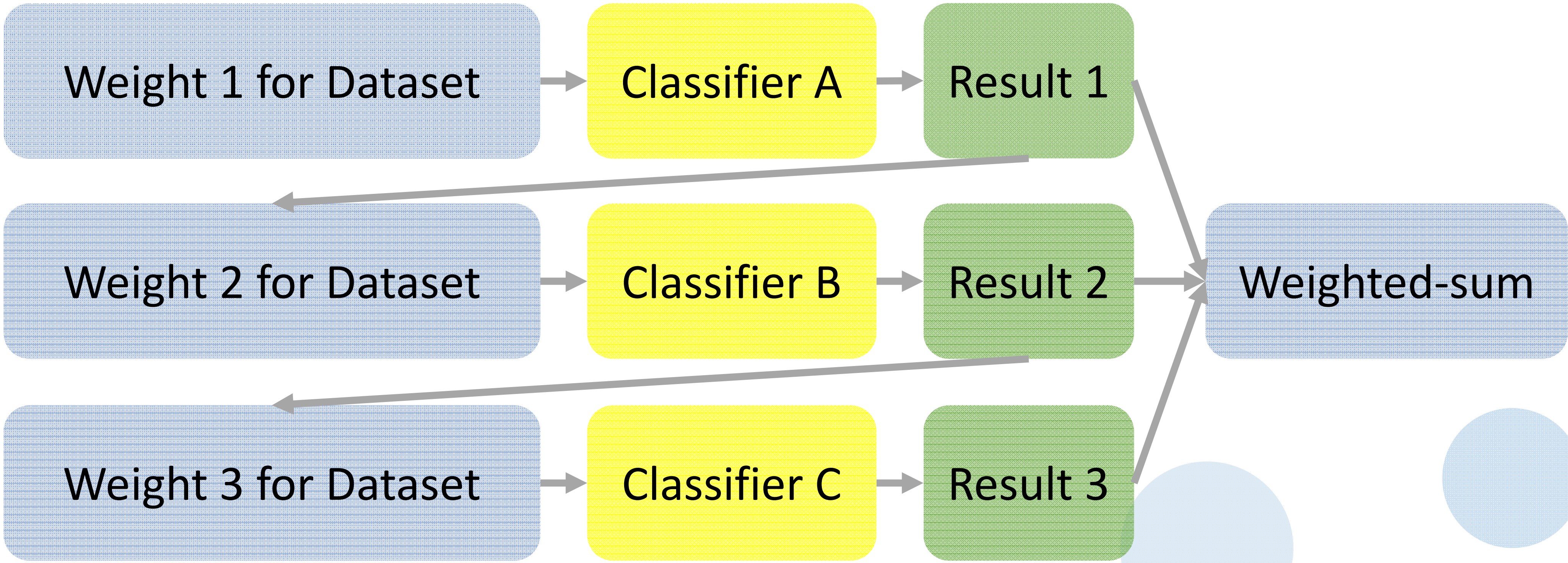
N = 20



N = 100



Boosting



Weight 1 for Dataset
Classifier A
Result 1
Weight 2 for Dataset
Classifier B
Result 2
Weighted-sum
Weight 3 for Dataset
Classifier C
Result 3

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1)$ $u^1 = \cancel{1}$ 0.4

$(x^2, \hat{y}^2, u^2)$ $u^2 = \cancel{1}$ 2.1

$(x^3, \hat{y}^3, u^3)$ $u^3 = \cancel{1}$ 0.7

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$\Downarrow$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-weighting Training Data

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ $u^1 = 1/\sqrt{3}$

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ $u^2 = \sqrt{3}$

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ $u^3 = 1/\sqrt{3}$

$(x^4, \hat{y}^4, u^4)$ $u^4 = 1$ $u^4 = 1/\sqrt{3}$

ε_1 =0.25

$f_1(x)$

0.5

ε_2 <0.5

$f_2(x)$

Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \; (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ \text{If } x^n \text{ correctly classified by } f_1 \; (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{cases}$$

increase

decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1$$

$$= \sum_n u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2$$

Re-weighting Training Data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$$

$$\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1\varepsilon_1$$

$$Z_1(1-\varepsilon_1)/d_1 = Z_1\varepsilon_1 d_1$$

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1$$