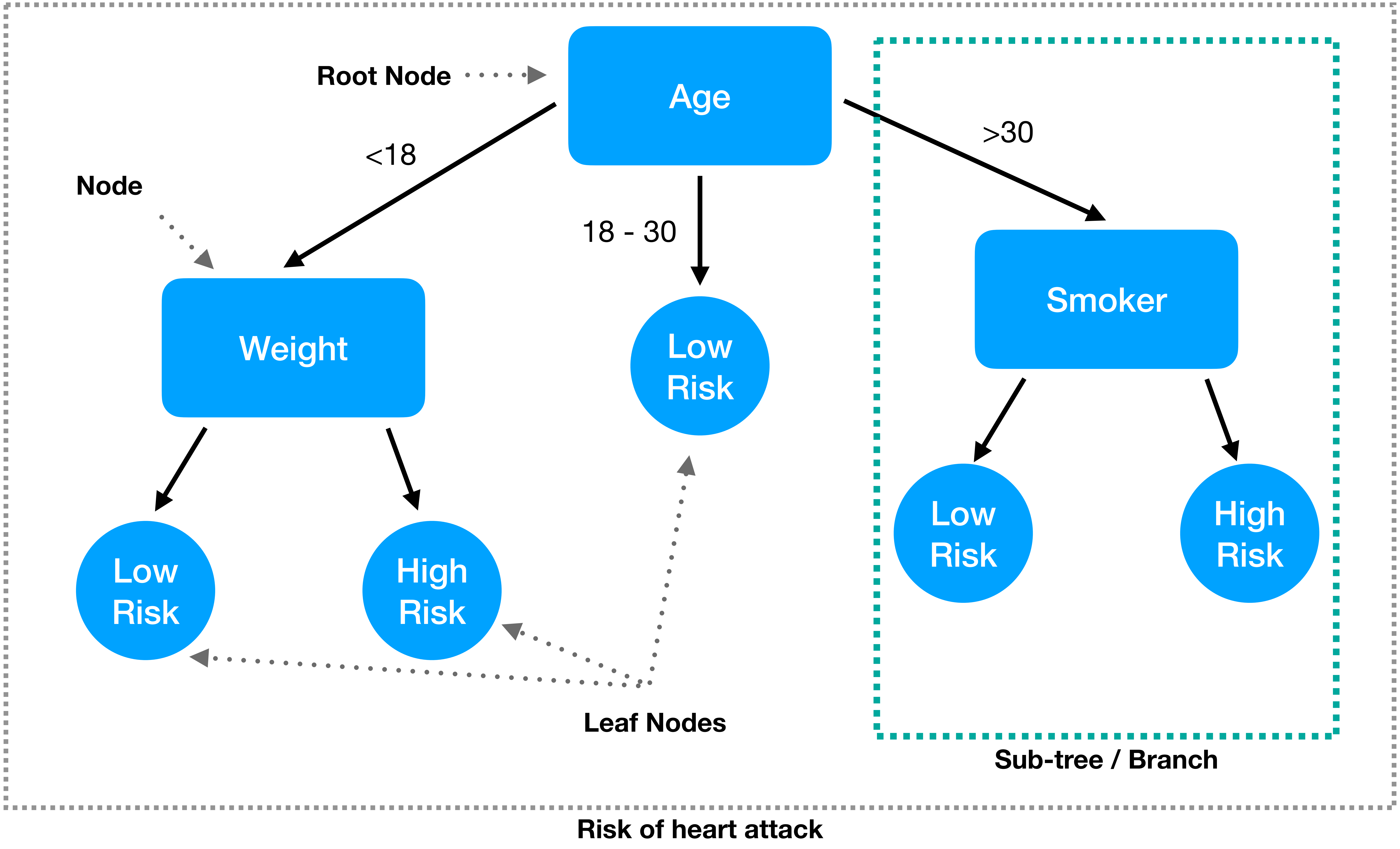
Decision Trees and Ensembles

What is a decision tree?



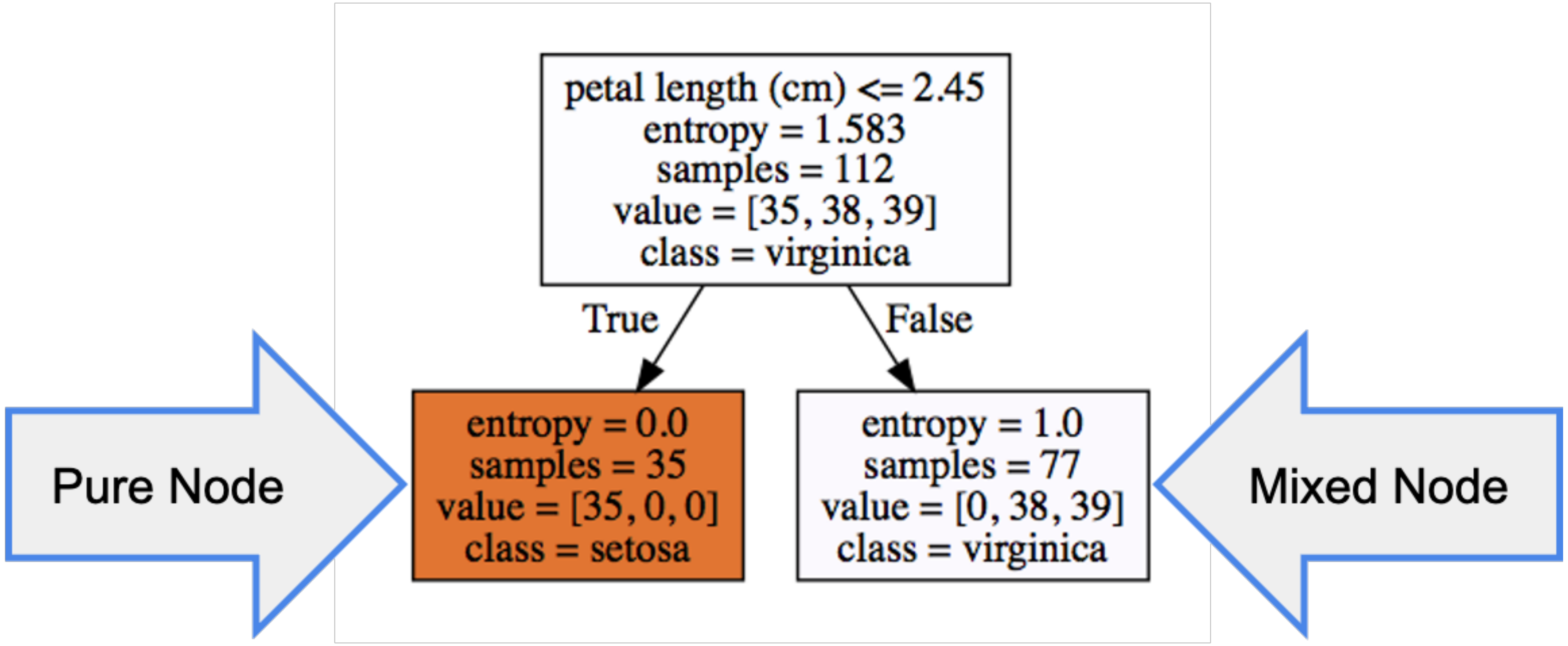
How can the algorithm perform the splitting

$$\text{Entropy}(S) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

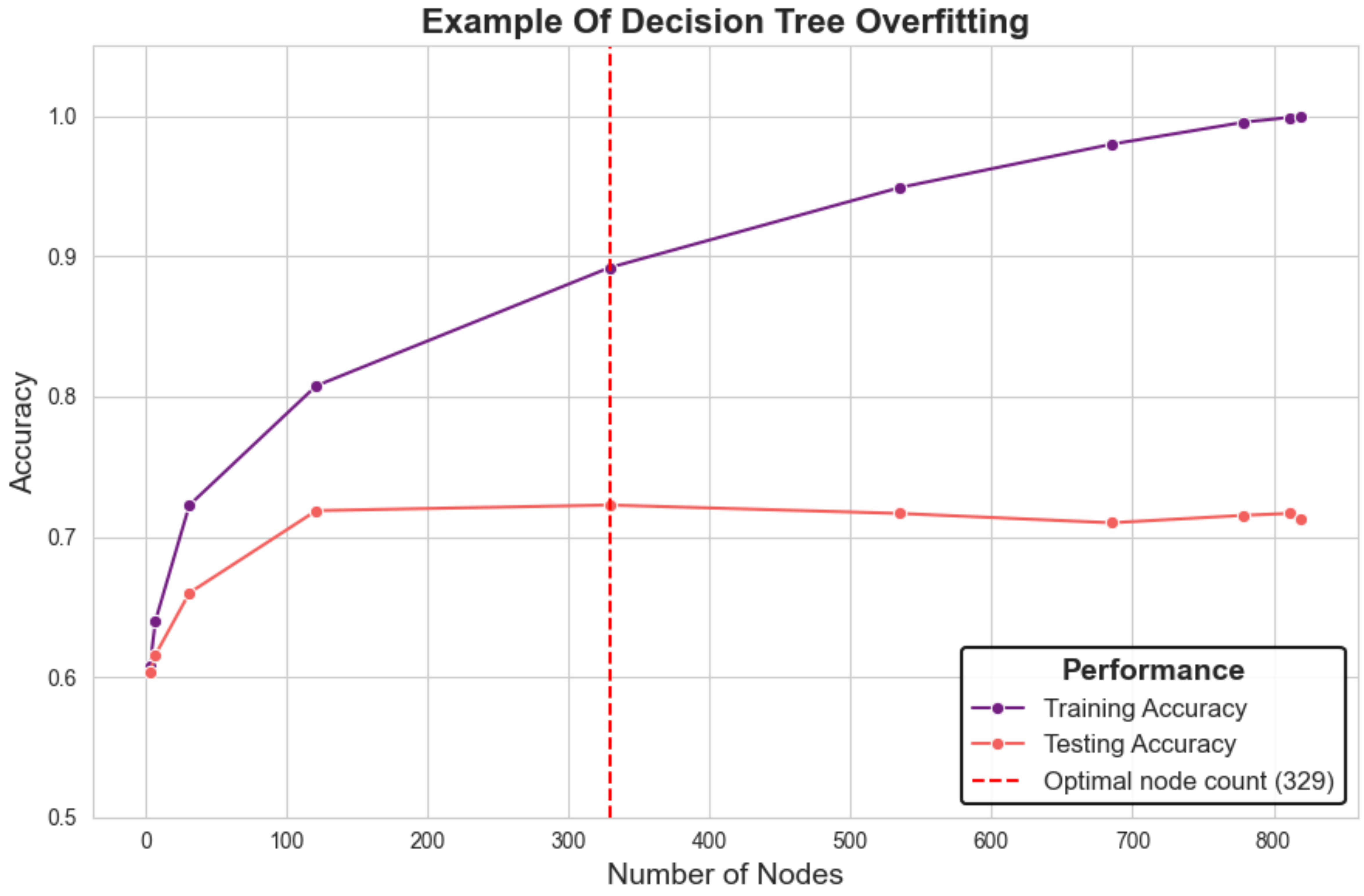
$$\text{Gini}(S) = 1 - \sum_{i=1}^{k} p_i^2$$

$$\text{Information Gain} = E_{\text{parent}} - \text{Avg}\, E_{\text{child}}$$

P_i is the probability of class i



Main problem of this algorithm



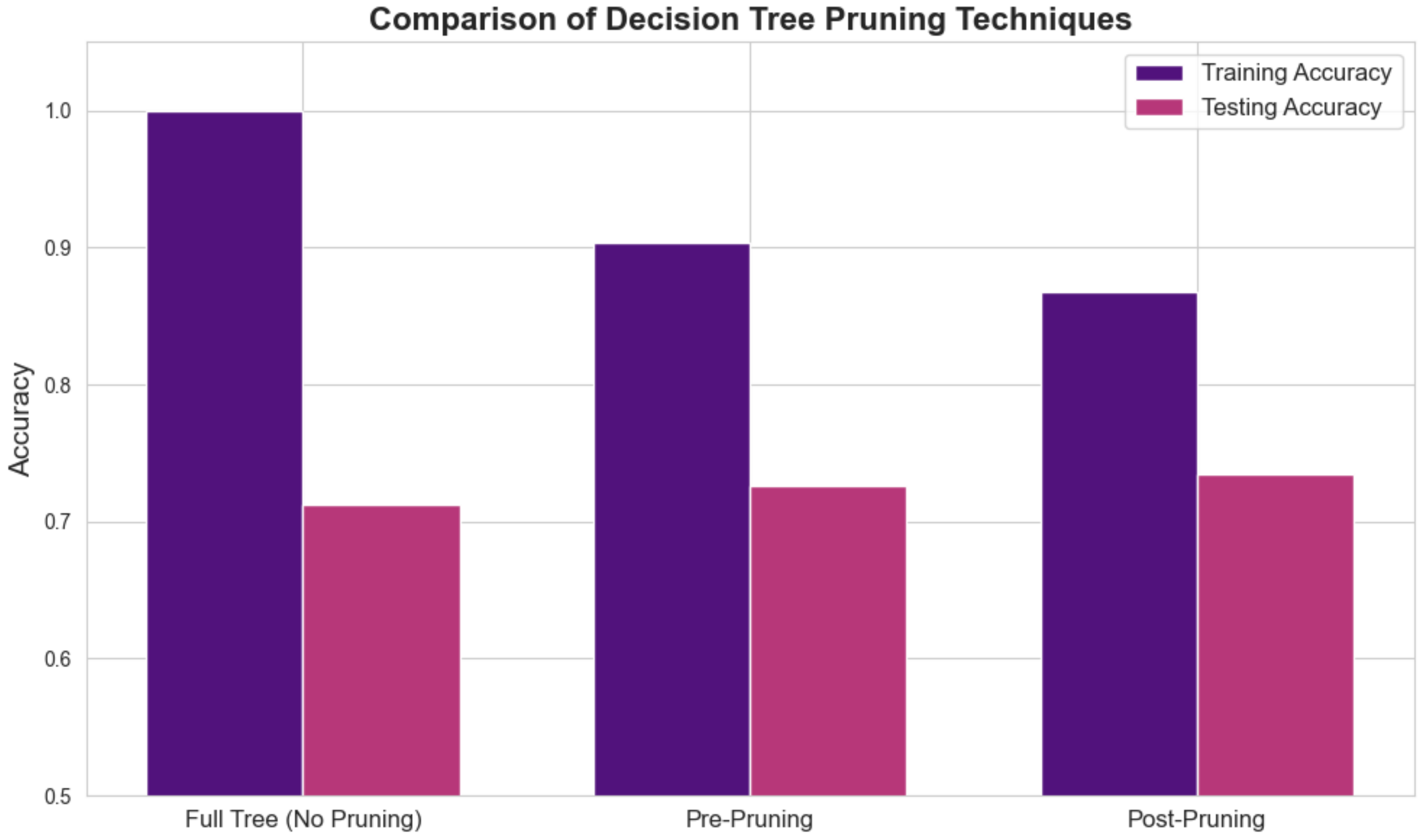
Overfitting is the main problem when using decision trees.

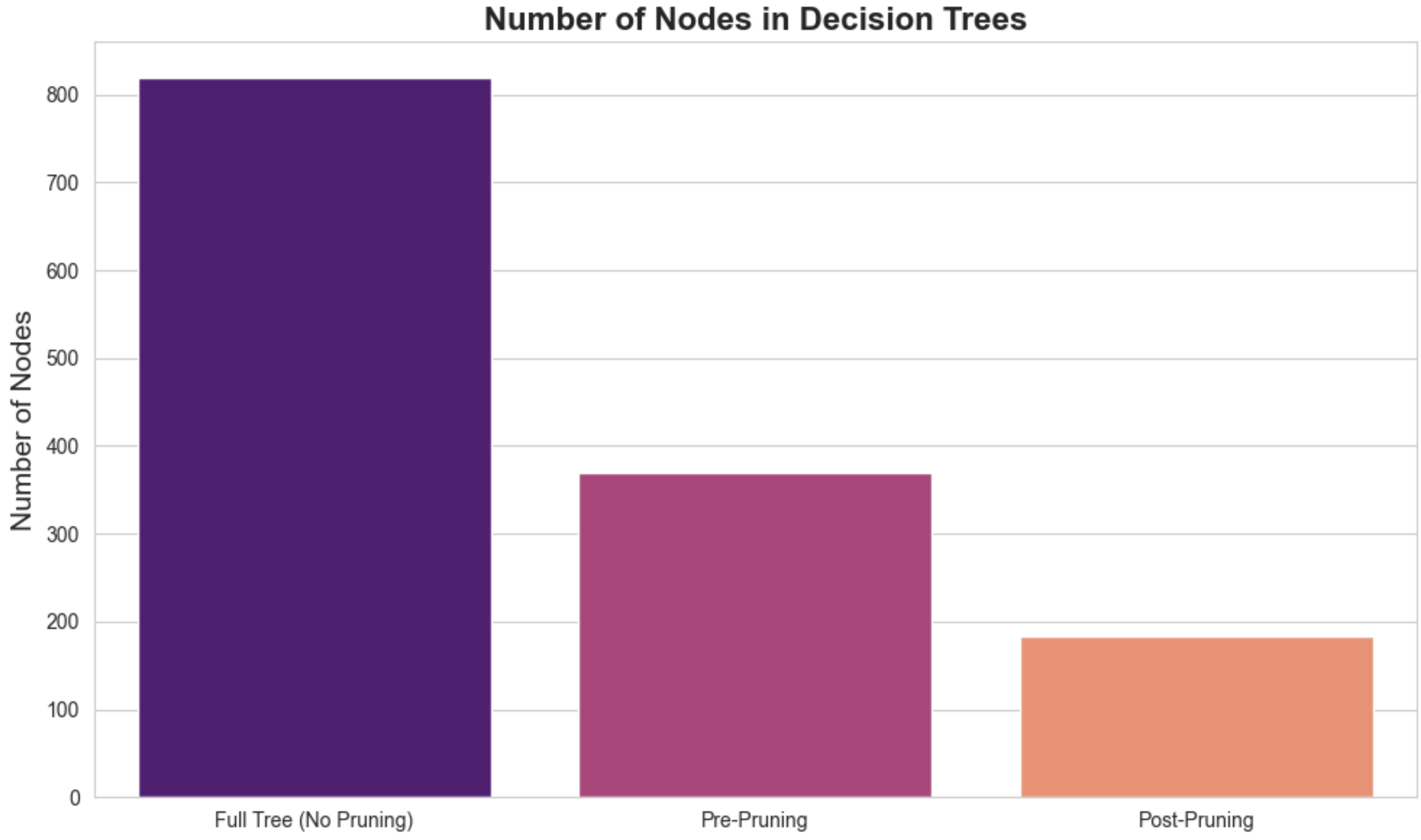
Deeper and more complex trees do **not equal better results**

Ways to mitigate the problem of overfitting

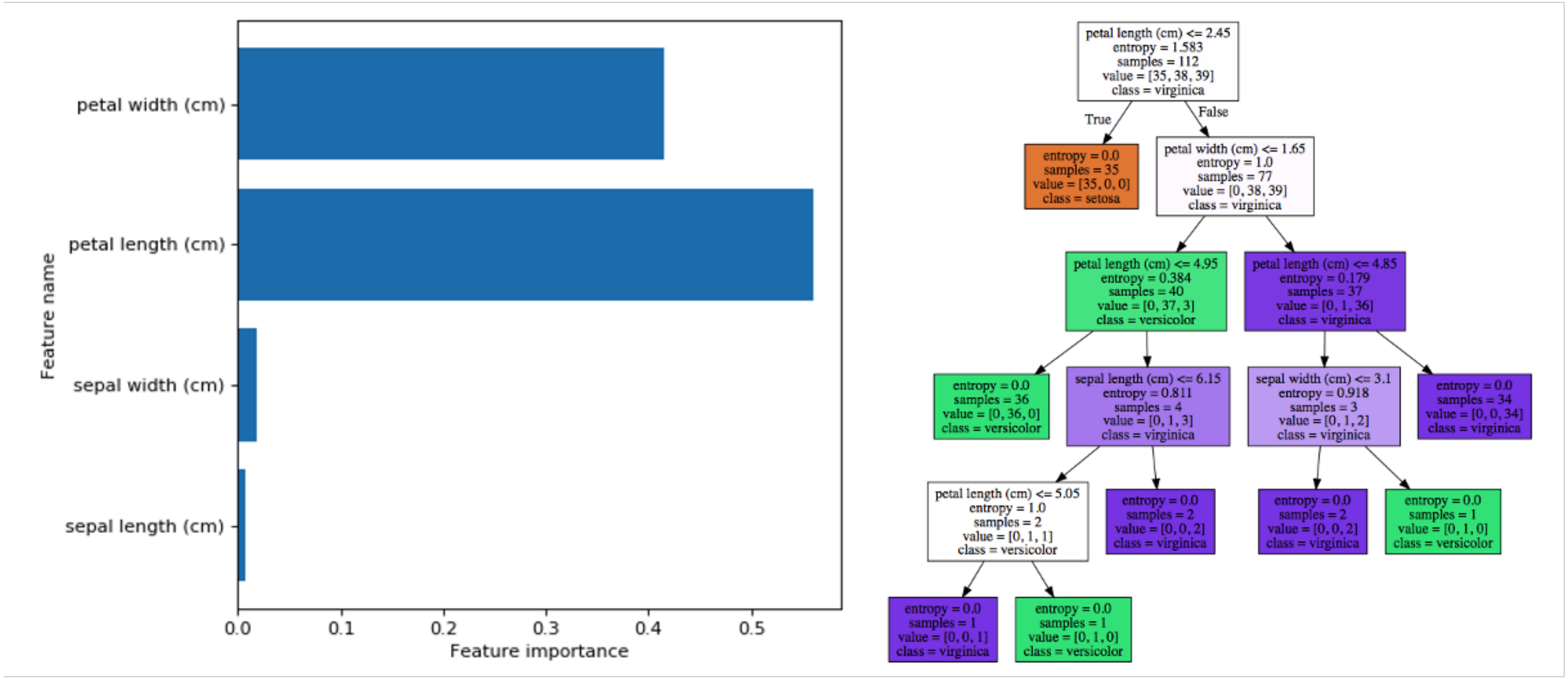
Pre-Pruning: Stops tree growth early to prevent overfitting, though it may halt promising splits.

Post-Pruning: Allows the full tree to grow and then prunes back subtrees that do not improve performance, effectively reducing overfitting.





Feature importance



Algorithms to build a decision tree

Algorithms	ID3	C4.5	C5.0	CART (used in scikit-learn)
Type of data	Categorical	Continuous and Categorical	Continuous, Categorical, Dates, Times, Timestamps	Continuous and Categorical
Speed	Low	Faster than ID3	Highest	Average
Missing values	Can’t deal with	Can deal with	Can deal with	Can deal with
Splitting	Use information entropy and information gain	Use split info and gain ratio	Same as C4.5	Use Gini index

Example of ID3 algorithm work

Outlook	Humidity	Wind	Play
sunny	high	weak	no
sunny	high	strong	no
overcast	high	weak	yes
rainy	high	weak	yes
rainy	normal	weak	yes
rainy	normal	strong	no
overcast	normal	strong	yes
sunny	high	weak	no
sunny	normal	weak	yes
rainy	normal	weak	yes
sunny	normal	strong	yes
overcast	high	strong	yes
overcast	normal	weak	yes
rainy	high	strong	no

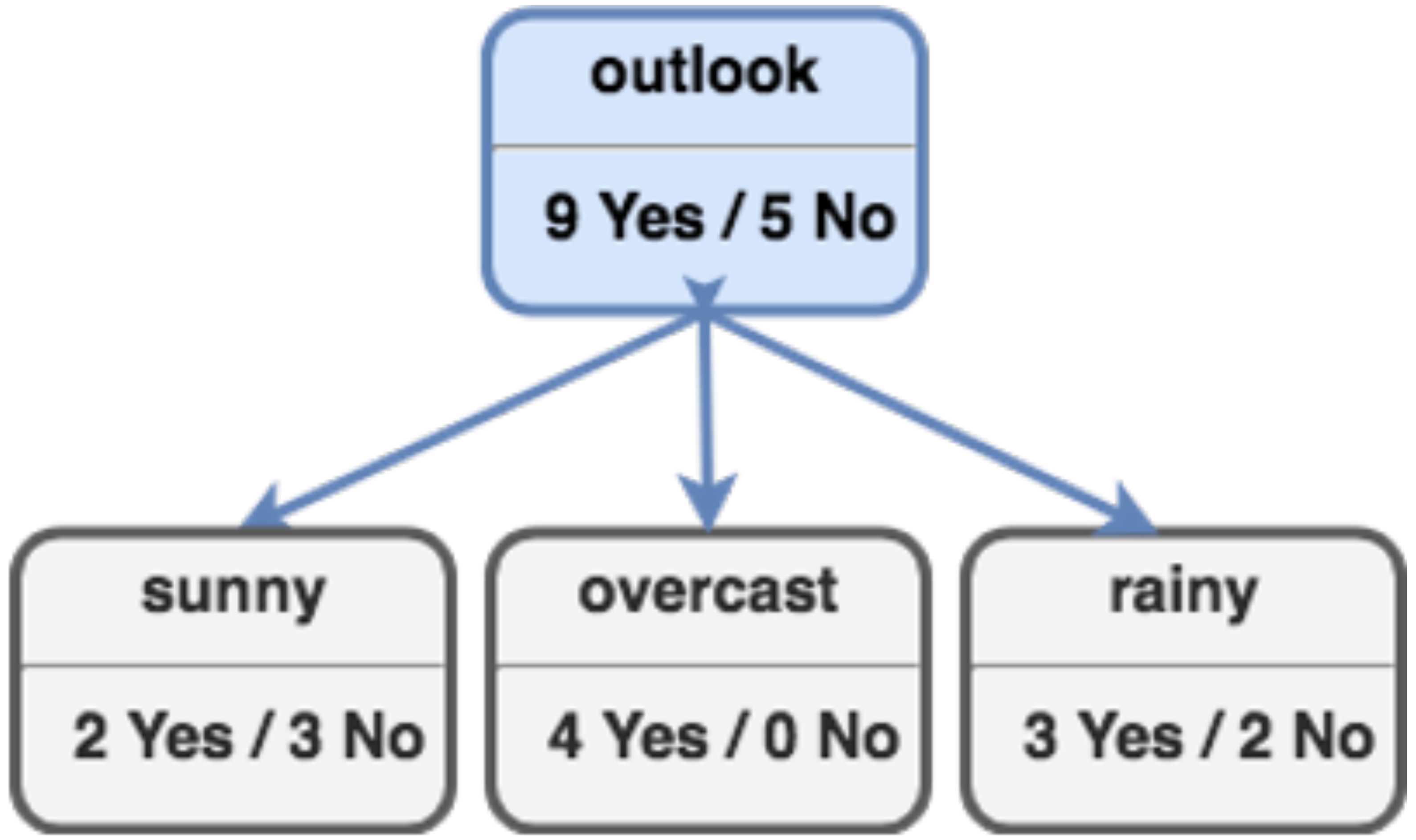
Outlook	Humidity	Wind	Play
rainy	high	no	?

Example of ID3 algorithm work

Need to choose feature to split:

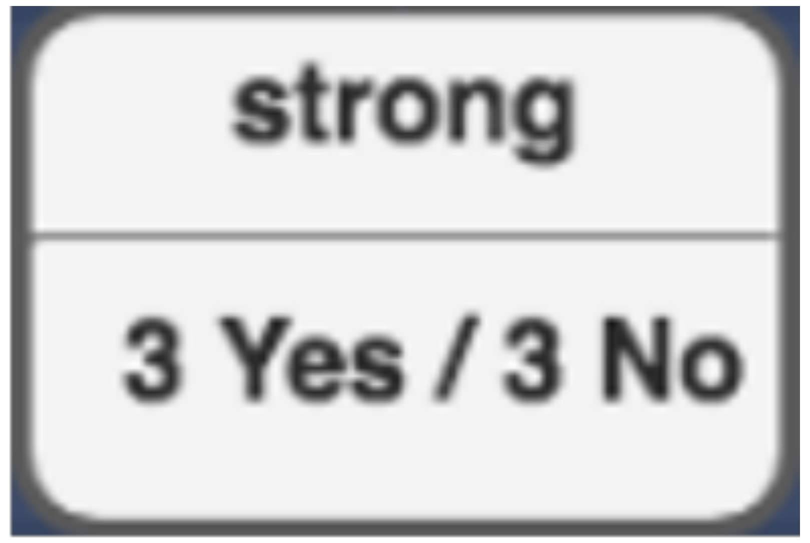
outlook | humidity | wind

Which one is the best?



Example of ID3 algorithm work | Certainty

Completely certain => entropy = 0



Completely uncertain => entropy = 1

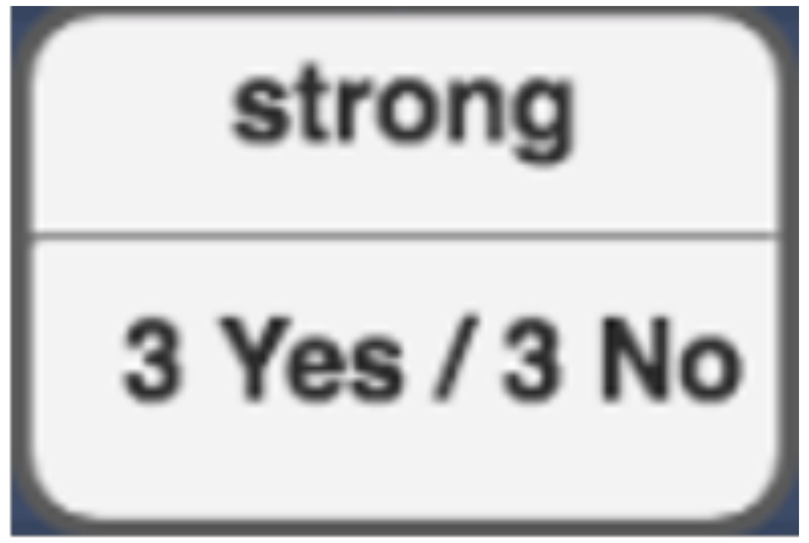
Expected symmetric measurement of uncertainty (entropy)
e.g. Value 1 for both “4 Yes/0 No” and “0 Yes/4 No”

Example of ID3 algorithm work | Entropy for binary classification

S - set of examples

$$H(S) = -\,p_{yes} \cdot \log_2 p_{yes} - p_{no} \cdot \log_2 p_{no}$$

$$H(S) = -\frac{4}{4} \cdot \log_2 \frac{4}{4} - \frac{0}{4} \cdot \log_2 \frac{0}{4} = 0 - 0 = 0$$



$$H(S) = -\frac{3}{6} \cdot \log_2 \frac{3}{6} - \frac{3}{6} \cdot \log_2 \frac{3}{6} = -\frac{1}{2} \cdot (-1) - \frac{1}{2} \cdot (-1) = 1$$

Example of ID3 algorithm work | Information Gain

$$Gain(S, A) = H(S) - \sum_{v \in V} \frac{|S_v|}{|S|} H(S_v)$$

A - attribute(feature)

S - set of examples

V - values of A

S_v - subset, where X_a =v

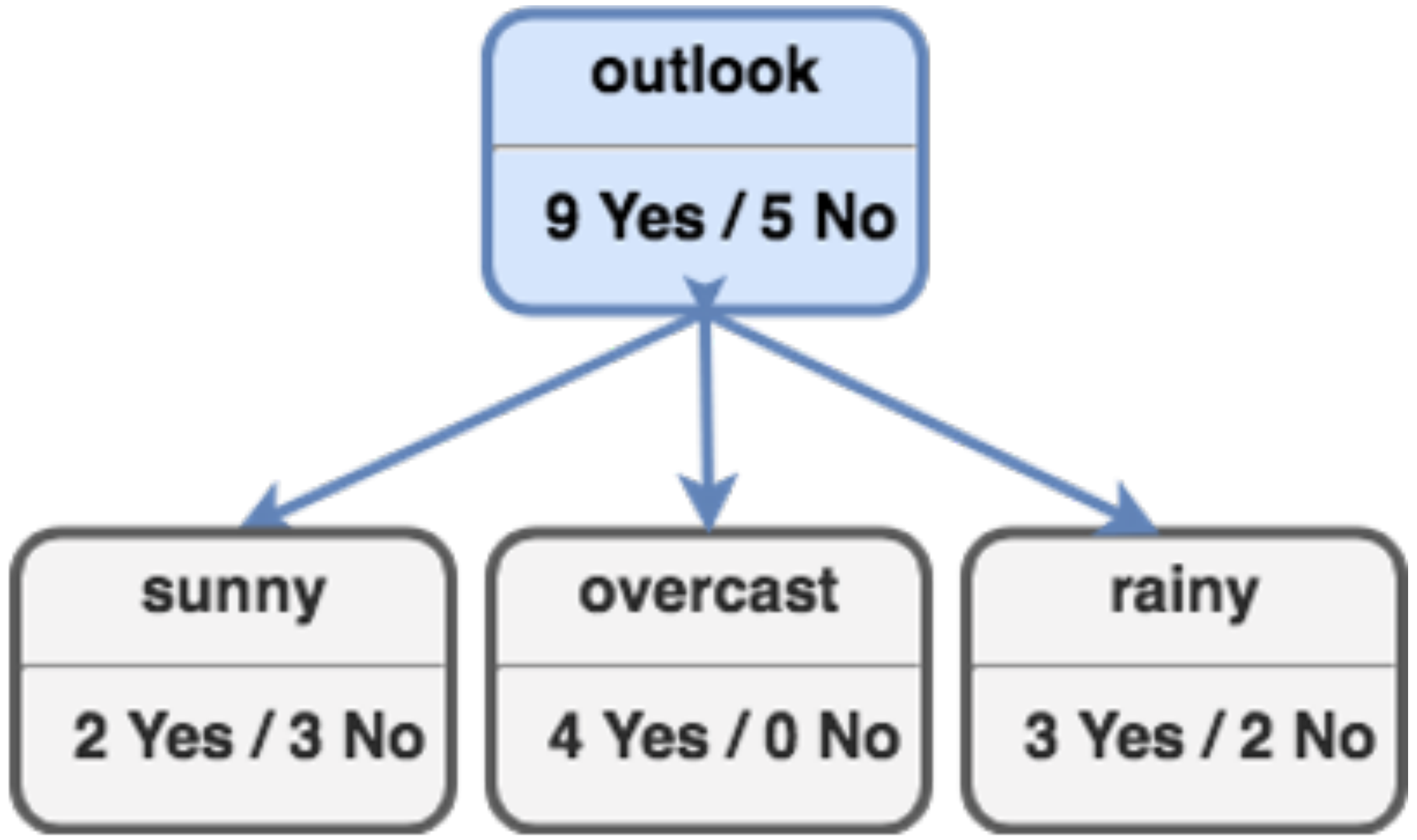
Information gain = loss of entropy

Example of ID3 algorithm work | Gain calculation

$$H(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

$$H(S_{\text{weak}}) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.81$$

$$H(S_{\text{strong}}) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$$

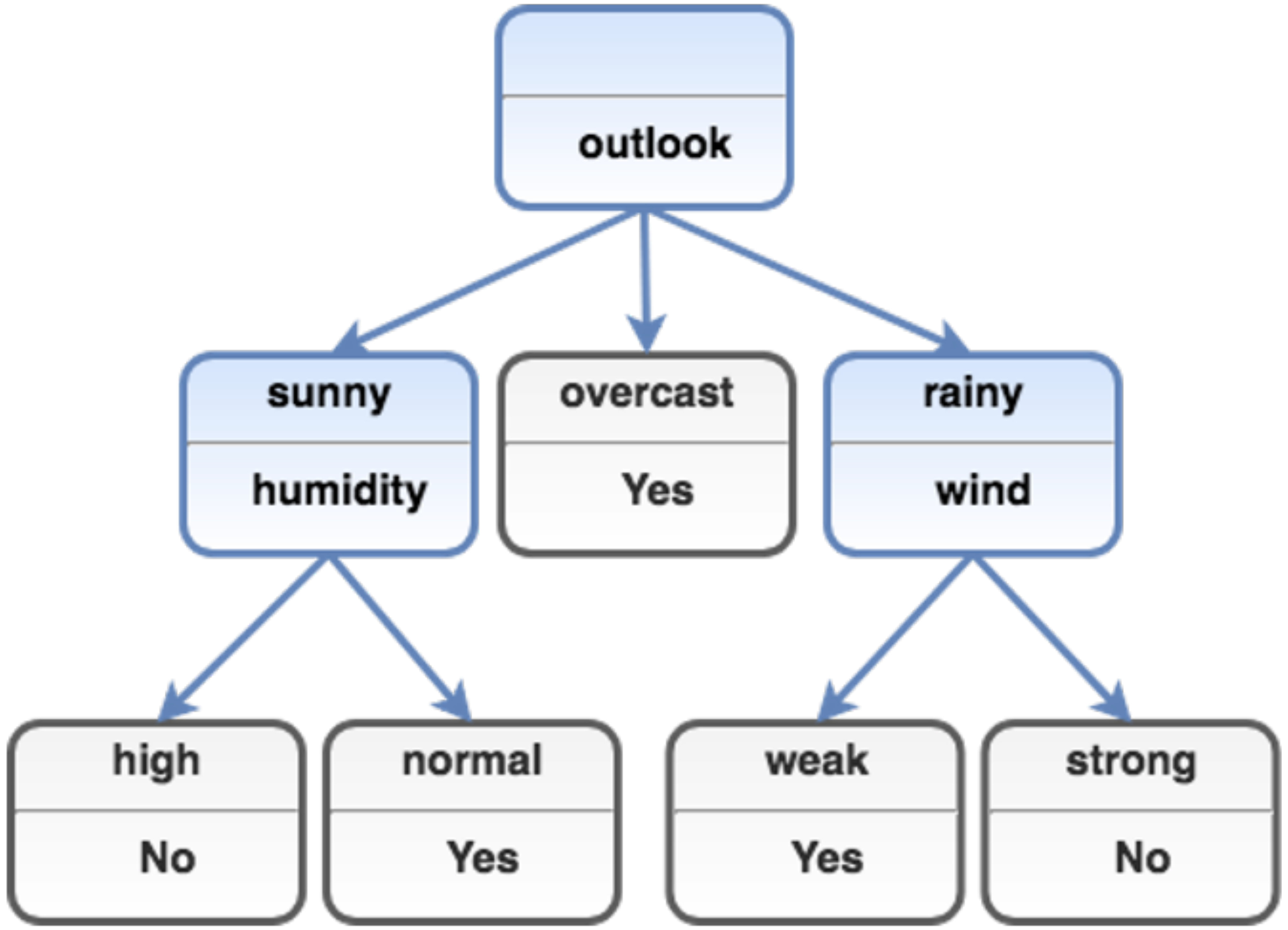


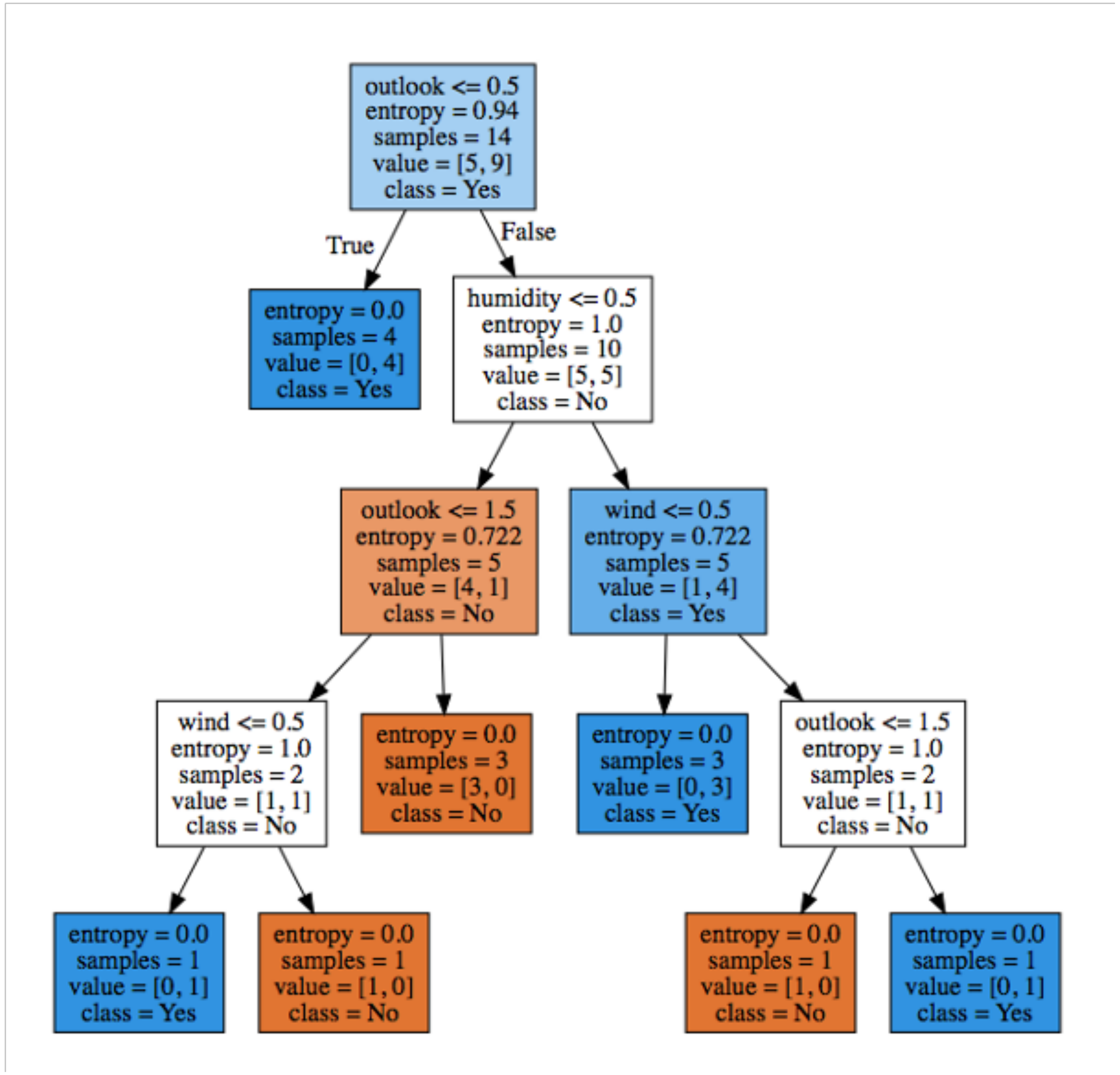
$$Gain(S, wind) = 0.94 - \frac{8}{14}\cdot 0.81 - \frac{6}{14}\cdot 1 = 0.049$$

$$Gain(S, outlook) = 0.94 - \frac{5}{14}\cdot 0.971 - \frac{4}{14}\cdot 0 - \frac{5}{14}\cdot 0 = 0.247$$

Feature 'outlook' has the best information gain.

Example of ID3 algorithm work | Full tree

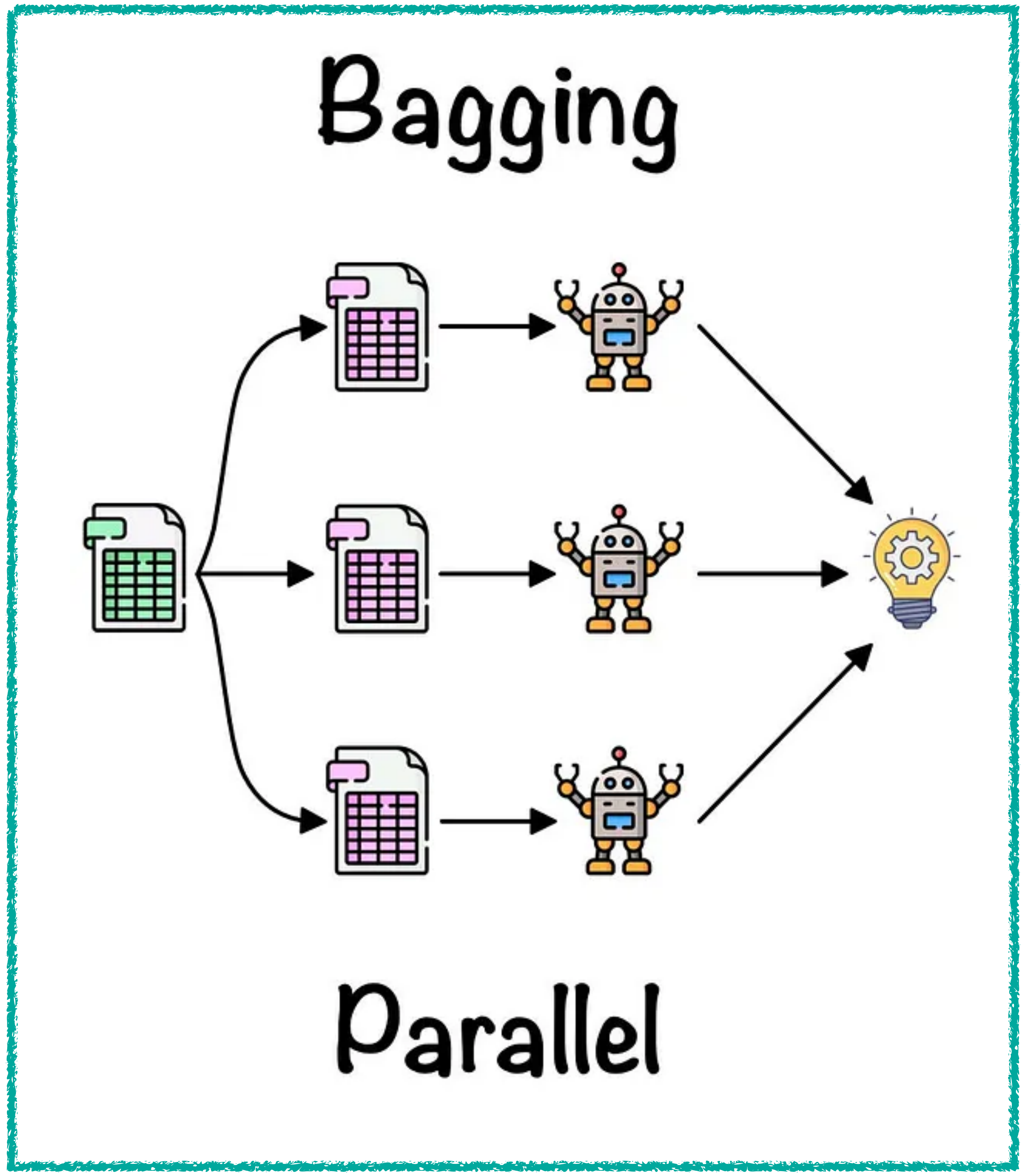


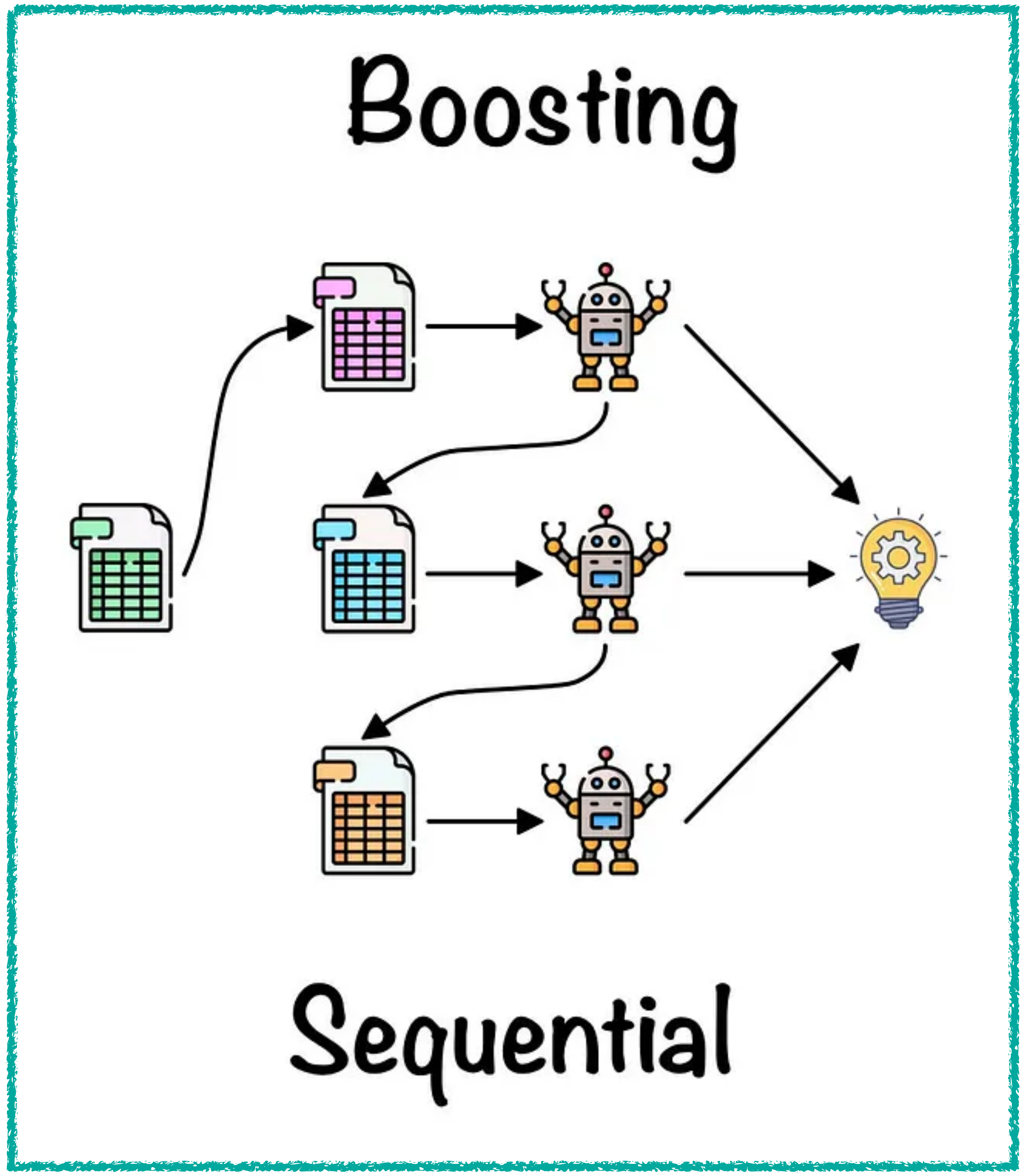


Decision Trees Sklearn Implementation (Classification)

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(

#     criterion : ”gini”, “entropy” (default=”gini”)

    random_state=0, # (default None) The best split may vary due to features are randomly permuted at each split.

#     max_depth : int or None, (default=None)

#     max_leaf_nodes : int or None, optional (default=None)

).fit(X_train, y_train)


print("train accuracy= {:.3%}".format(clf.score (X_train, y_train)))

print("test accuracy= {:.3%}".format(clf.score (X_test, y_test)))
```

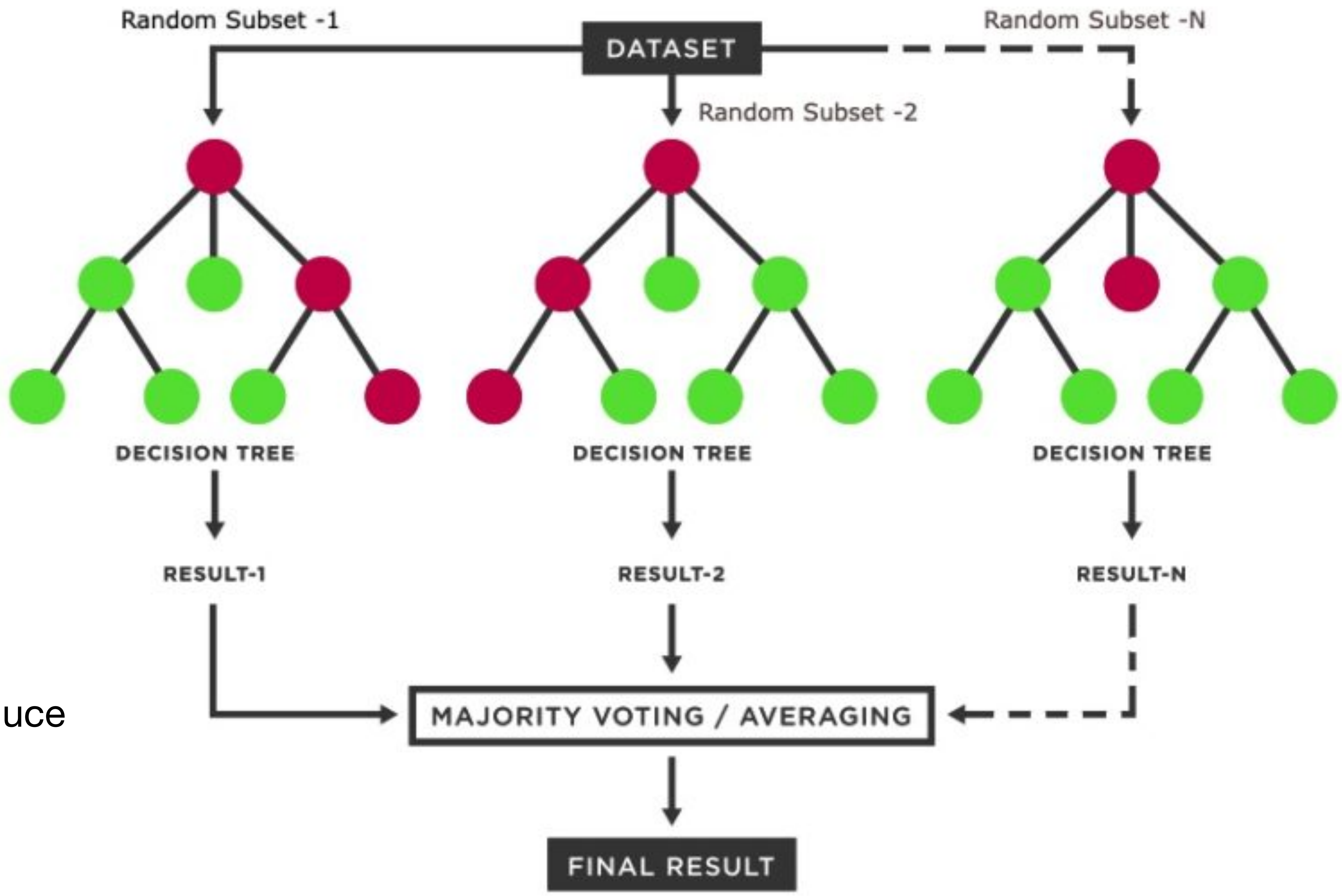
Ensembles | Boosting and Bagging





Ensembles | Random Forest

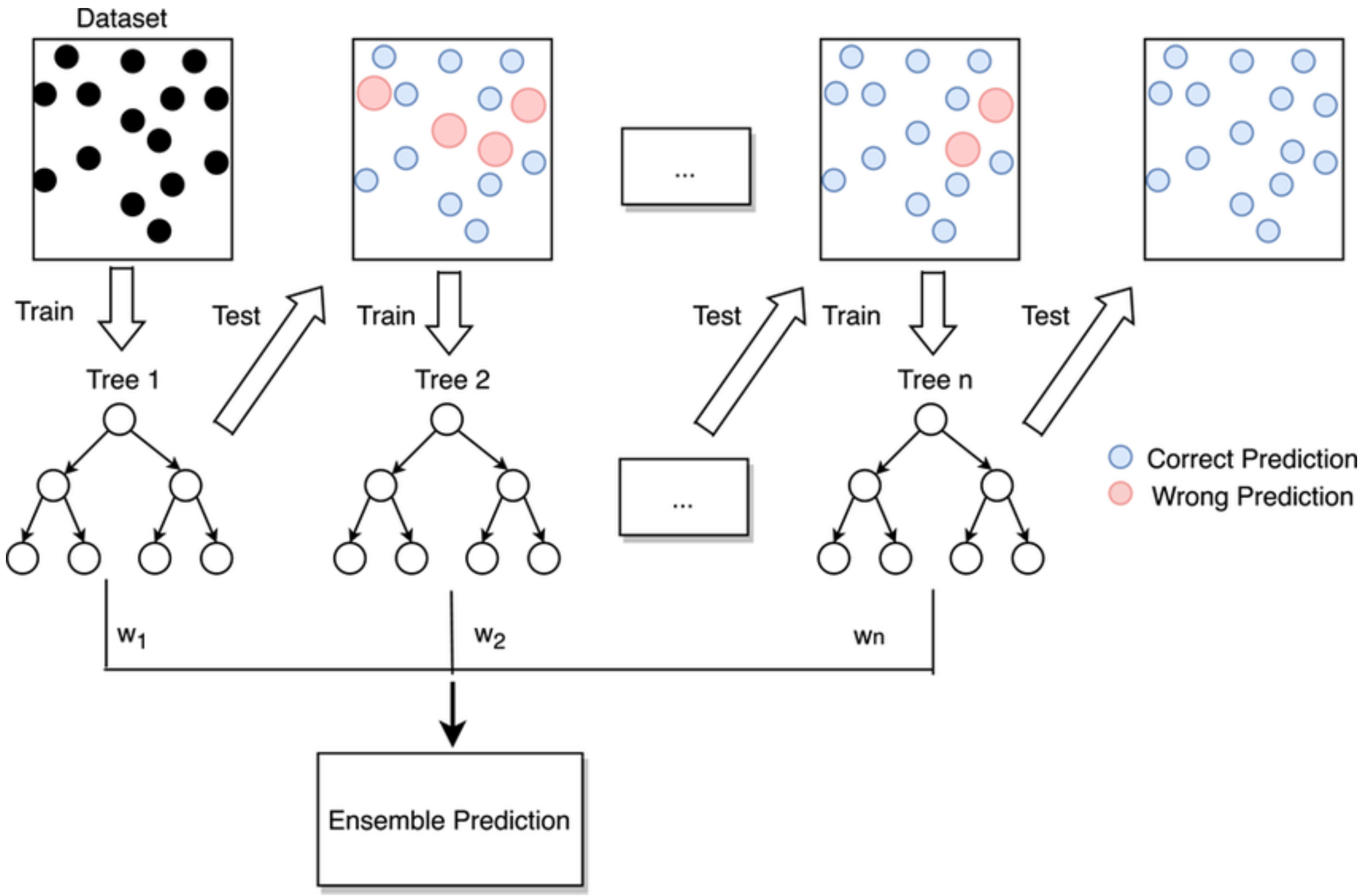
- An ensemble of trees, not just one tree.
- Widely used, very good results on many problems.
- sklearn.ensemble module:
    - ➡ **Classification**: RandomForestClassifier
    - ➡ **Regression**: RandomForestRegressor
- One decision tree → Prone to overfitting.
- Many decision trees → More stable, better generalization
- Ensemble of trees should be diverse: introduce random variation into tree-building.



Random Forest Sklearn Implementation (Classification)

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(

          # n_estimators: default = 10

          # max_features:  default  “auto” => sqrt(n_features),  None => n_features

 # max_depth: default = None,

 # n_jobs=None,

   random_state= 0

      ).fit(X_train, y_train)
```

Ensembles | Gradient Boosting



- An ensemble of trees, built sequentially, not in parallel.
- Widely used, often achieves top results on many problems.
- sklearn.ensemble module:
    - ➡ **Classification: GradientBoostingClassifier**
    - ➡ **Regression: GradientBoostingRegressor**
- One decision tree → High bias, underfitting.
- Many decision trees → Gradually reduce residuals and improve accuracy.
- Each tree corrects errors (residuals) of the previous trees.
- Ensemble of trees should learn slowly: control with **learning_rate** to prevent overfitting.
- Smaller learning_rate → Better generalization, but requires more trees.

Gradient Boosting Sklearn Implementation (Classification)

```
clf = GradientBoostingClassifier(

learning_rate=0.01, # (default=0.1) larger value ->  more complex trees

max_depth=3, # (default=3)

# n_estimators= 10,  (default=100)

).fit(X_train, y_train)
```

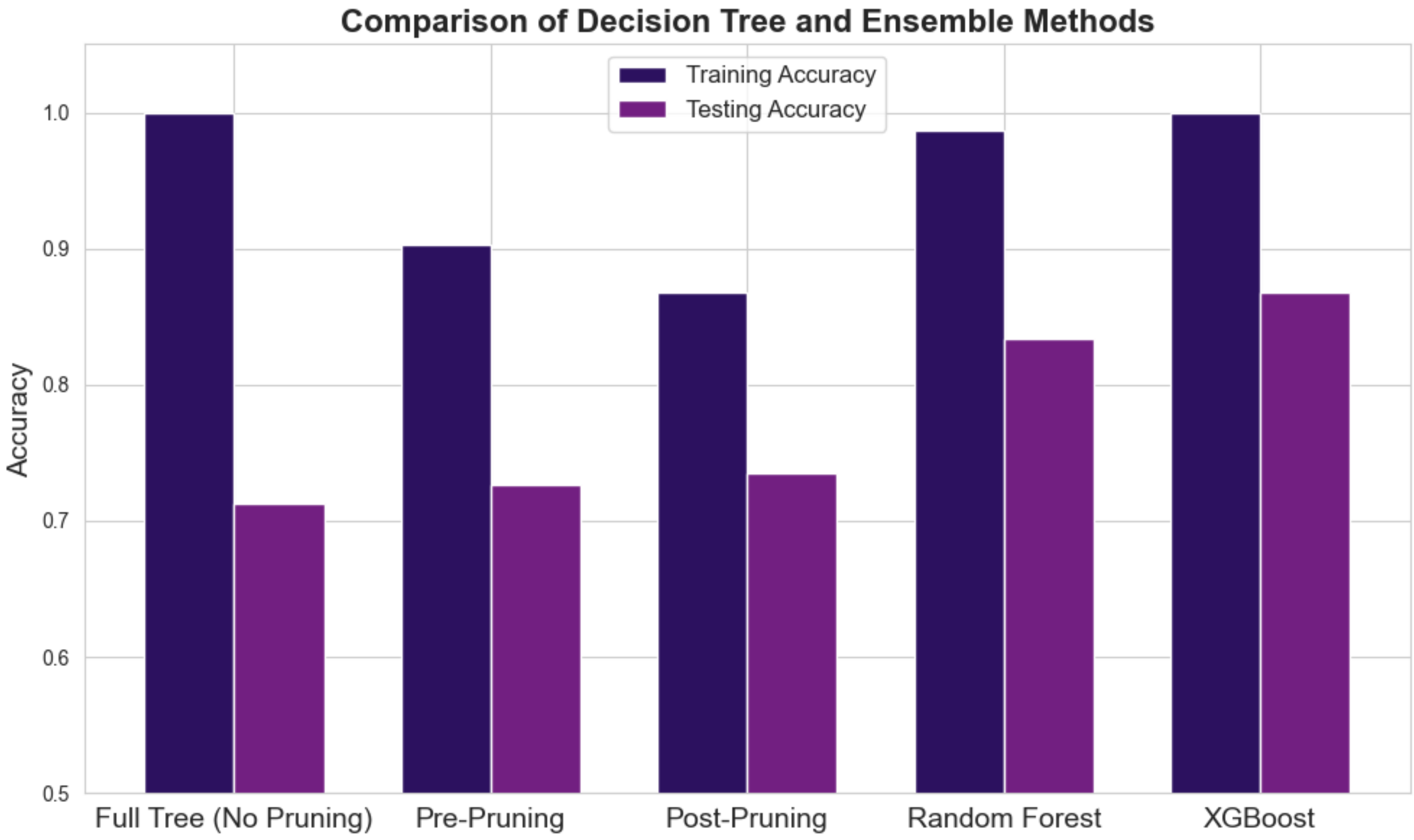
XGBoost - powerful Gradient Boosting algorithm

Extremely fast gradient boosting modifications that allows to apply L1 and L2 regularizations

+ reg_alpha : L1 regularization
reg_lambda : L2 regularization

```
from xgboost import XGBClassifier

clf = XGBClassifier() .fit(X_train, y_train)
```

Comparison of the results of a simple Decision Tree with ensembles



Comparison of the results of a simple Decision Tree with ensembles

Decision Tree

Pros:

- Easily visualized and interpreted
- No feature normalization needed
- Works well for mixture feature types

Cons:

- Cannot capture complex relation between features
- Not good choice for high-dimensional data comparing with linear models

Random Forest

Pros:

- Great performance
- No feature normalization needed
- Works well for mixture feature types

Cons:

- Difficult to interpret
- Not good choice for high-dimensional data comparing with linear models

XGBoost

Pros:

- Often achieves state-of-the-art results in classification and regression tasks.
- Prevents overfitting using L1 and L2 regularization.

Cons:

- Although faster than traditional gradient boosting, still slower than Random Forest for very large datasets.
- In most cases, it is difficult to achieve optimal results without hyperparameter optimization

Homework

Use load_breast_cancer and classify with:

- Decision Trees
- Random Forest
- GBDT
- XGBoost