Decision Tree

Lê Anh Cường

TDTU

Outline

- What kind of machine learning type for Decision Tree classifier
- Decion Tree for classification
- How to build Decision Tree
  - Entropy
  - Information Gain and Feature selection

Decision Tree: example

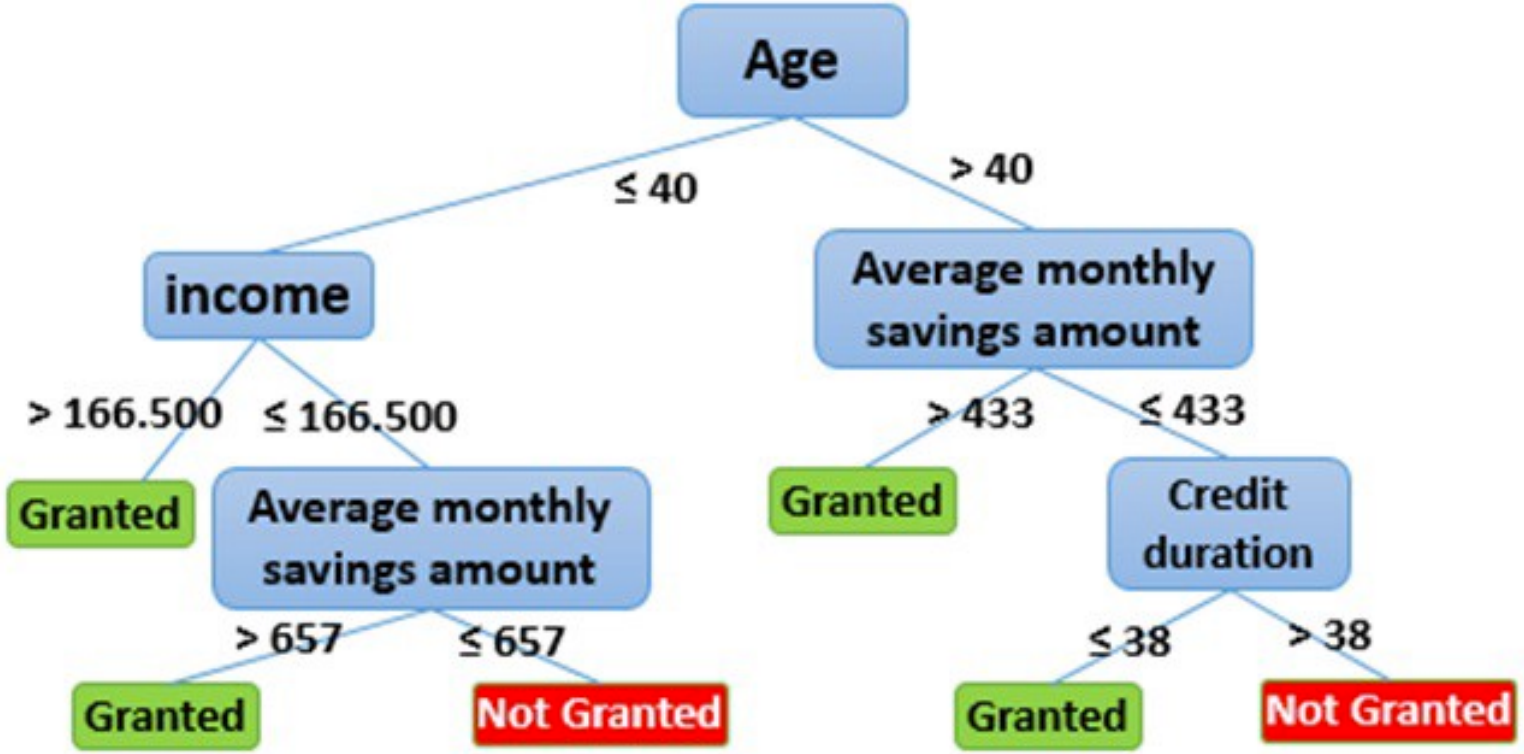
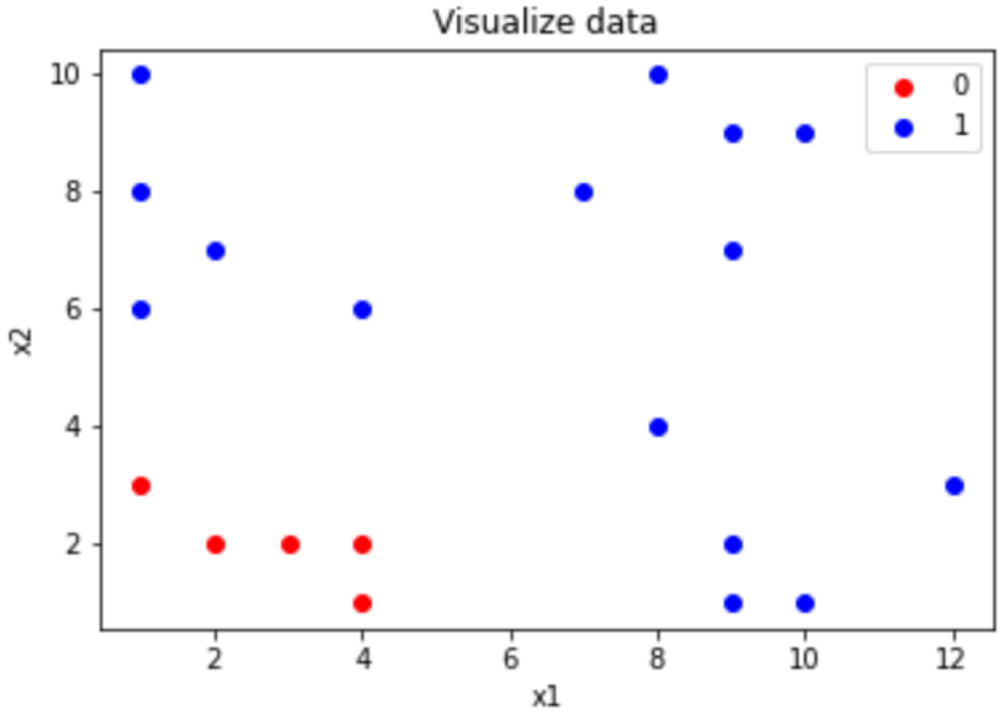
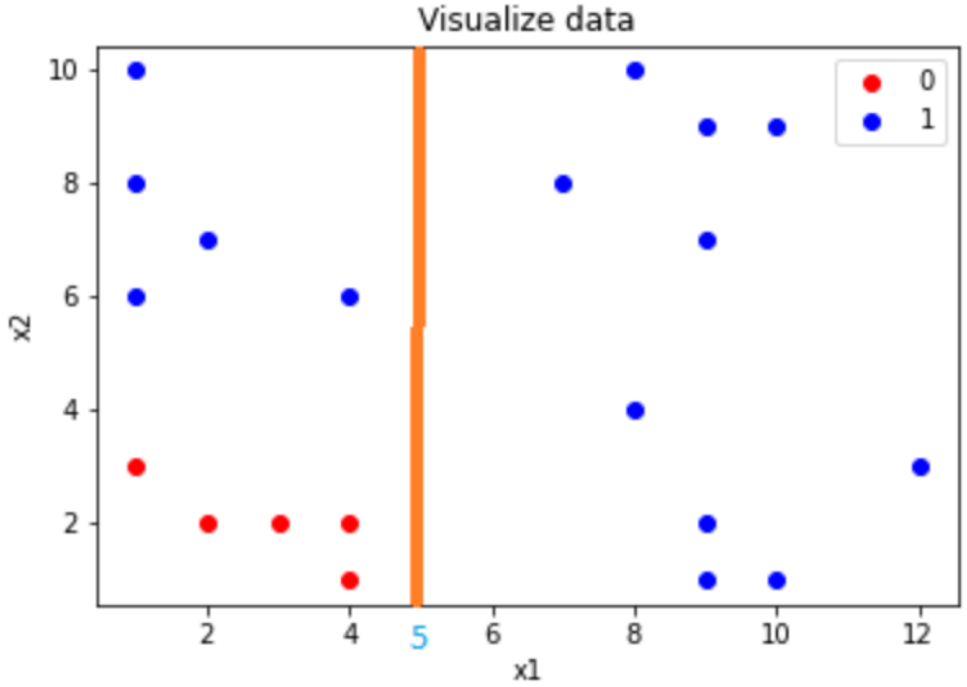
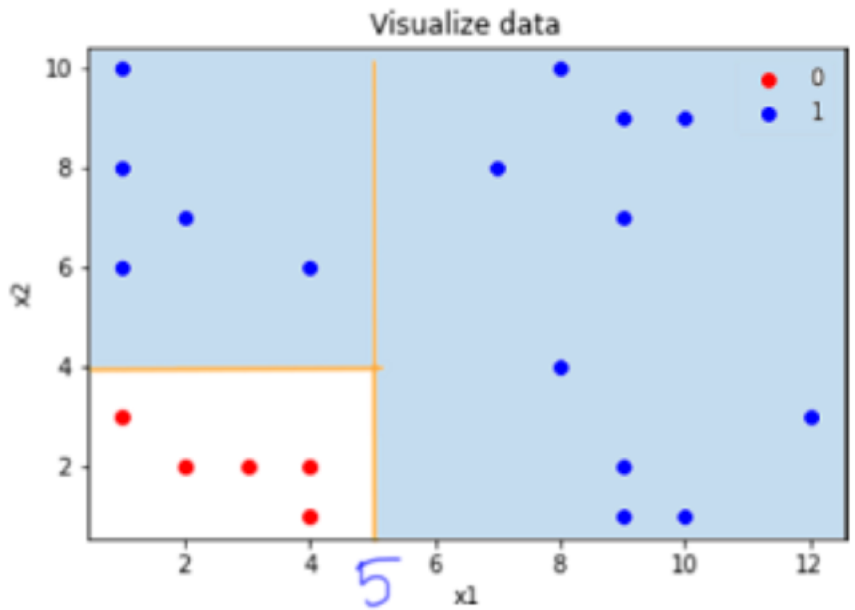


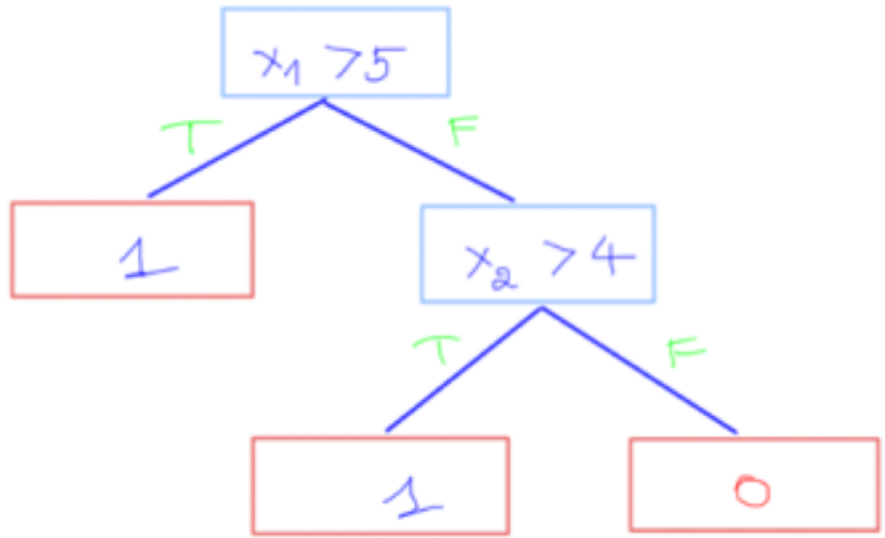
Fig1. Loan approval example

DT: Segmentation

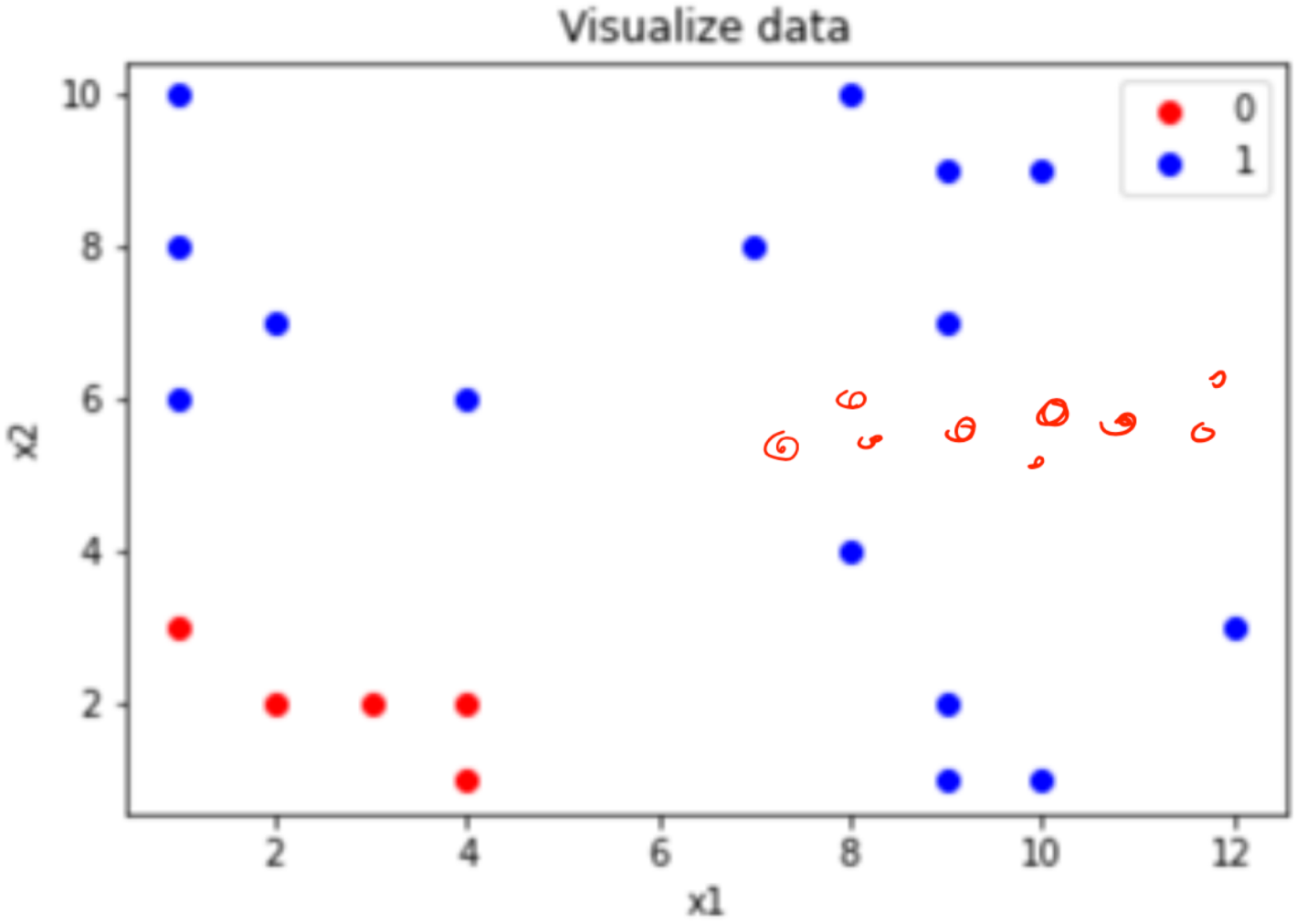




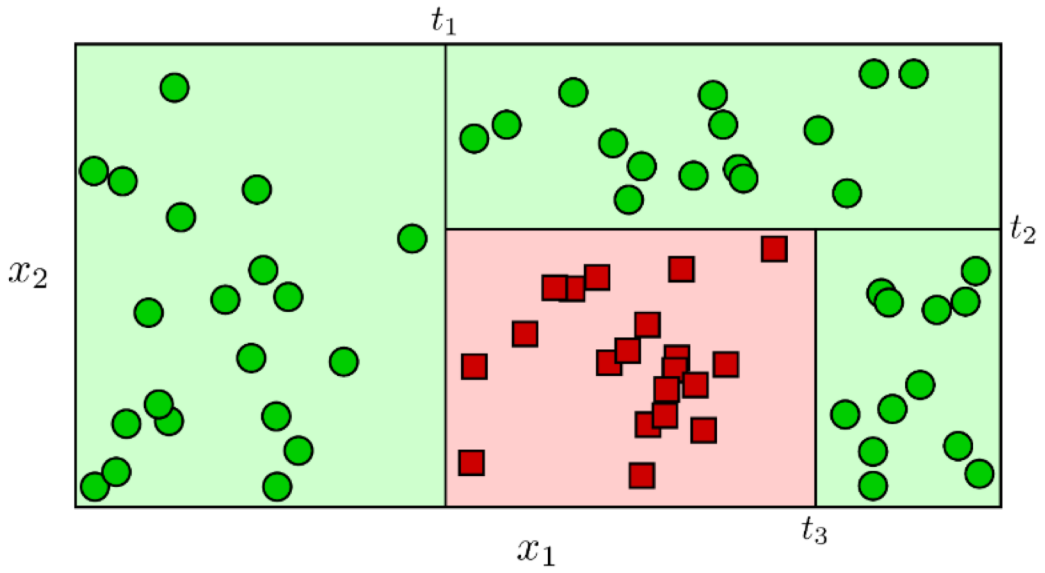

Visualize data
0
1
x2
x1
10
8
6
4
2
2
4
6
8
10
12
5


$x_1 > 5$
T
F
1
$x_2 > 4$
T
F
1
0

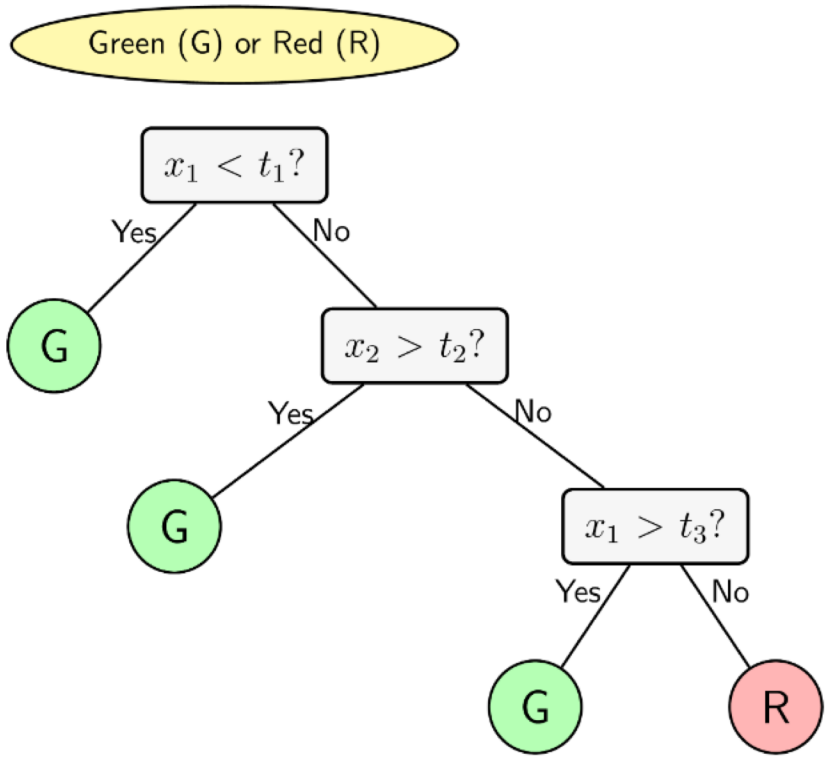
How?



Decision Tree: example



(a)



(b)

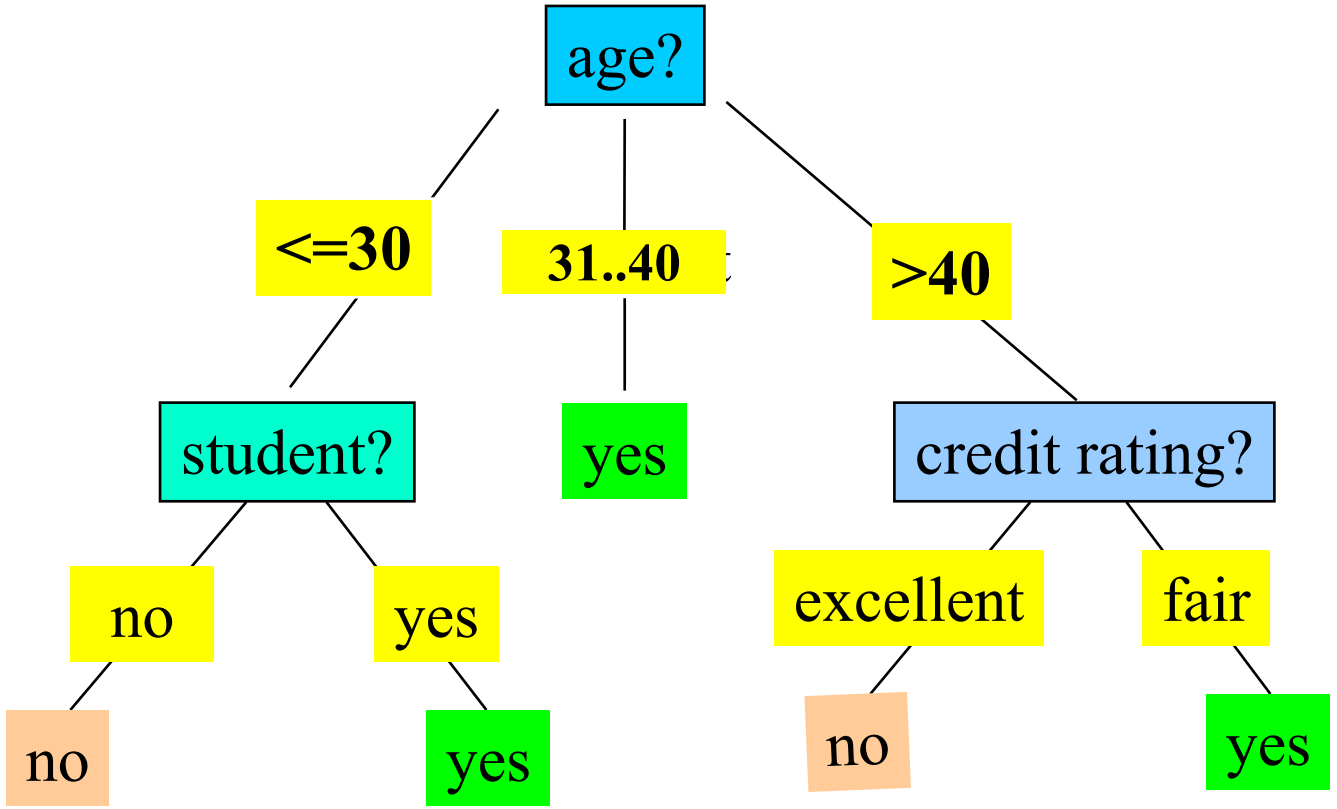
Machine Learning Type of Decision Tree

- **Logical models** use a logical expression to divide the instance space into segments and hence construct grouping models.
- A **logical expression** is an expression that returns a Boolean value, i.e., a True or False outcome.
- Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve.
- There are mainly two kinds of logical models: **Tree models** and **Rule models**.

Machine Learning Type of Decision Tree

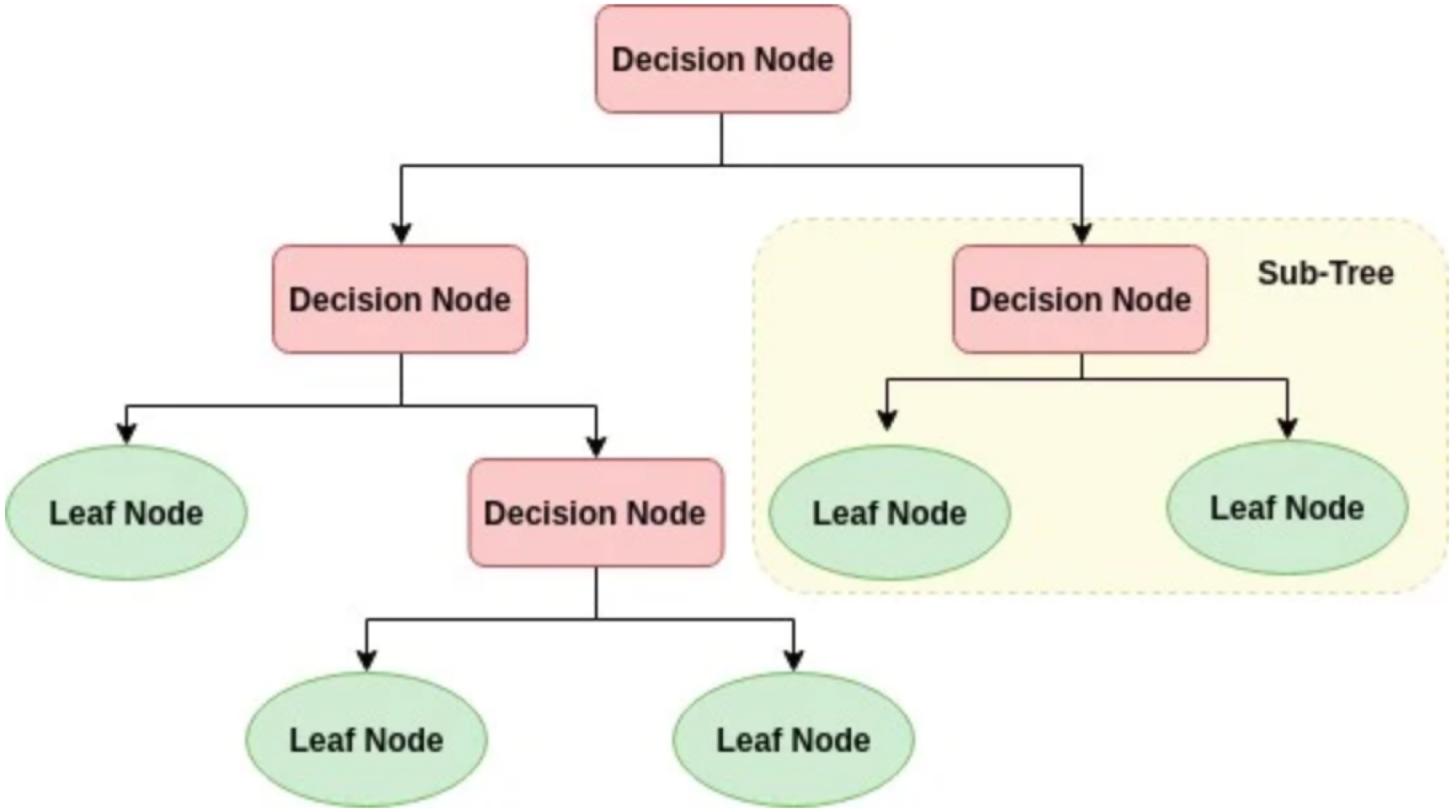
- Rule models consist of a collection of implications or IF-THEN rules.
- For tree-based models, the ‘if-part’ defines a segment and the ‘then-part’ defines the behaviour of the model for this segment. Rule models follow the same reasoning.
- Tree models can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure.

How does DT work?



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree: General Architecture



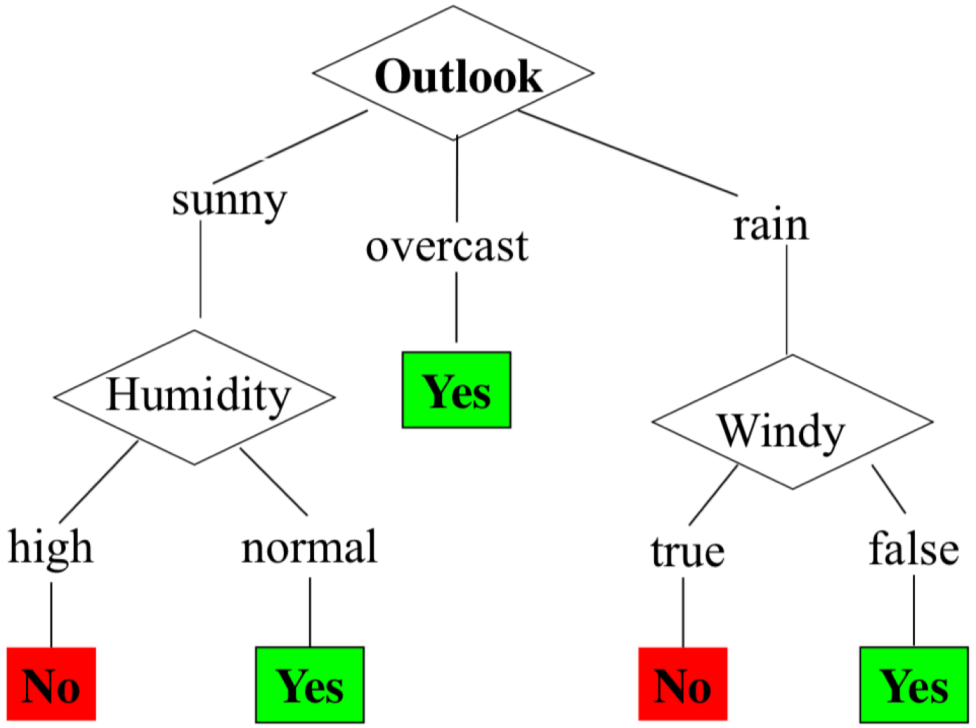
How to build Decision Tree for Classification?

Problem example

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

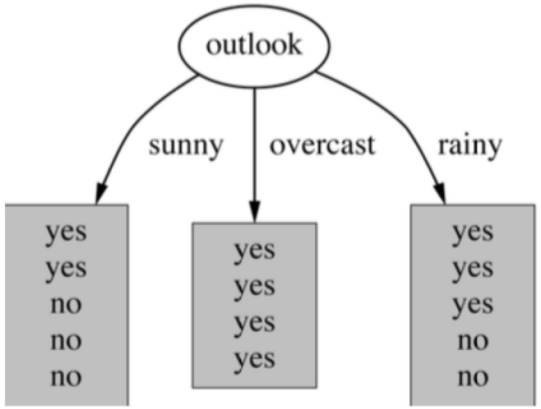
Building Decision Trees

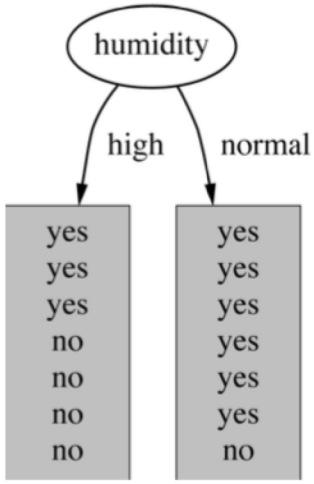
Attribute: Outlook, Temp, Humidity, Windy

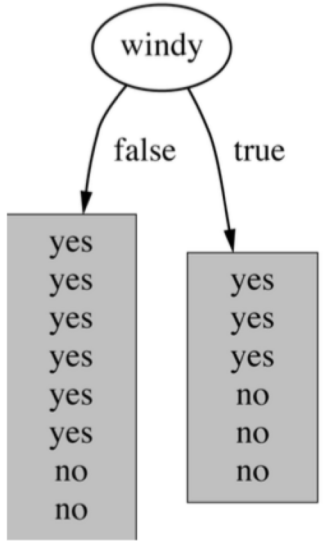


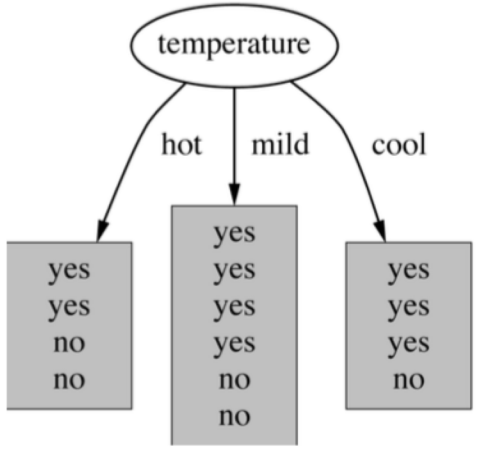
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny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

Which attribute for Segmentation?

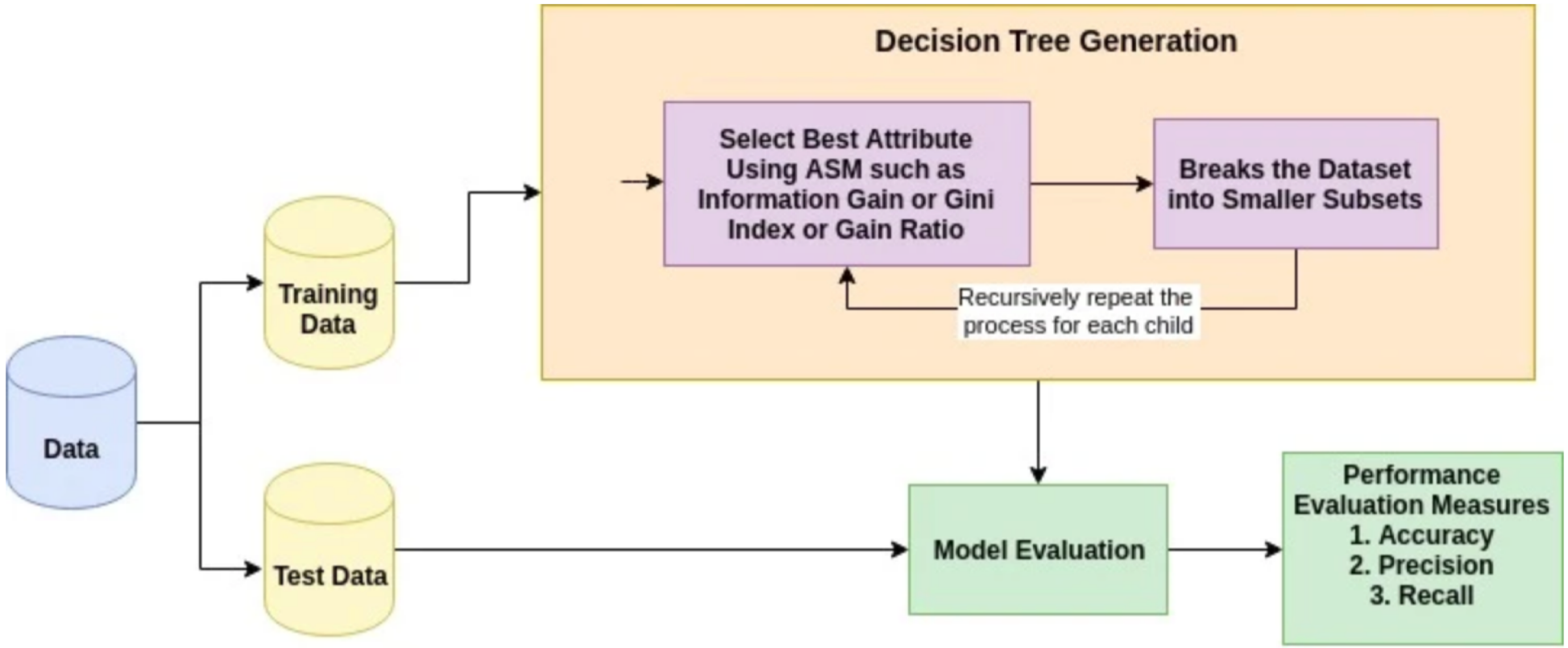








The Algorithm for building DT

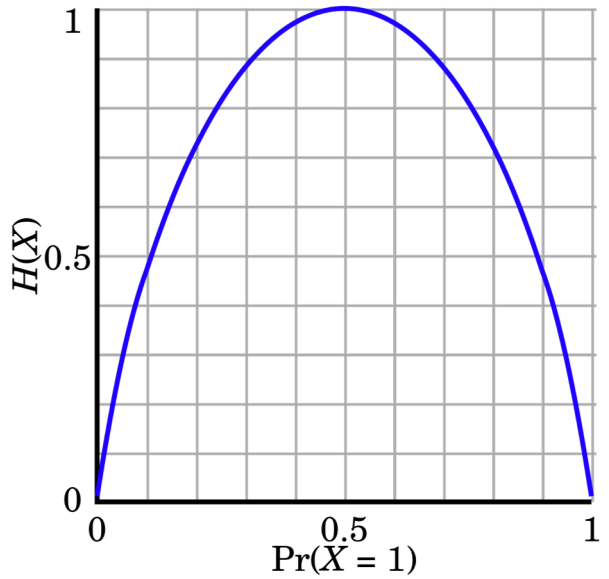


ID3 Algorithm for learning/building Decision Tree

Entropy

Entropy is a **measure** of the unpredictability of the state, or equivalently, of its average **information** content.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$



p=0.7, then

$$\begin{aligned}
\mathrm{H}(X) &= -p \log_2(p) - q \log_2(q) \\
&= -0.7 \log_2(0.7) - 0.3 \log_2(0.3) \\
&\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\
&= 0.8816 < 1
\end{aligned}$$

$$\begin{aligned}
\mathrm{H}(X) &= -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i) \\
&= -\sum_{i=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} \\
&= -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1) = 1
\end{aligned}$$

Attribute Selection Measures

ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

$$\text{Info}(D) = -\sum_{i=1}^{m} pi \log_2 pi$$

Where, Pi is the probability that an arbitrary tuple in D belongs to class Ci.

$$\text{Info}_A(D) = \sum_{j=1}^{V} \frac{|Dj|}{|D|} \quad \text{X Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

Attribute Selection Measures

$$\text{Info}_A(D) = \sum_{j=1}^{V} \frac{|Dj|}{|D|} \quad \text{X Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Where,

- Info(D) is the average amount of information needed to identify the class label of a tuple in D.
- |Dj|/|D| acts as the weight of the jth partition.
- InfoA(D) is the expected informa-tion required to classify a tuple from D based on the partitioning by A.

The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N().

Attribute: ‘Outlook’

- “Outlook” = “Sunny”:

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,\text{bits}$$

- “Outlook” = “Overcast”:

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\,\text{bits}$$

chú ý : log(0) không xác định nhưng 0*log(0) là 0

- “Outlook” = “Rainy”:

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,\text{bits}$$

- thông tin của thuộc tính outlook:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693\,\text{bits}$$

Information Gain

$$\text{gain(" Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

Attribute: ‘Humidity’

- “Humidity” = “High”:

$$\text{info}([3,4]) = \text{entropy}(3/7,4/7) = -3/7\log(3/7) - 4/7\log(4/7) = 0.985\,\text{bits}$$

- “Humidity” = “Normal”:

$$\text{info}([6,1]) = \text{entropy}(6/7,1/7) = -6/7\log(6/7) - 1/7\log(1/7) = 0.592\,\text{bits}$$

- thông tin của thuộc tính humidity

$$\text{info}([3,4],[6,1]) = (7/14)\times 0.985 + (7/14)\times 0.592 = 0.788\,\text{bits}$$

- độ lợi thông tin của thuộc tính humidity

$$\text{info}([9,5]) - \text{info}([3,4],[6,1]) = 0.940 - 0.788 = 0.152$$

IG of all attributes

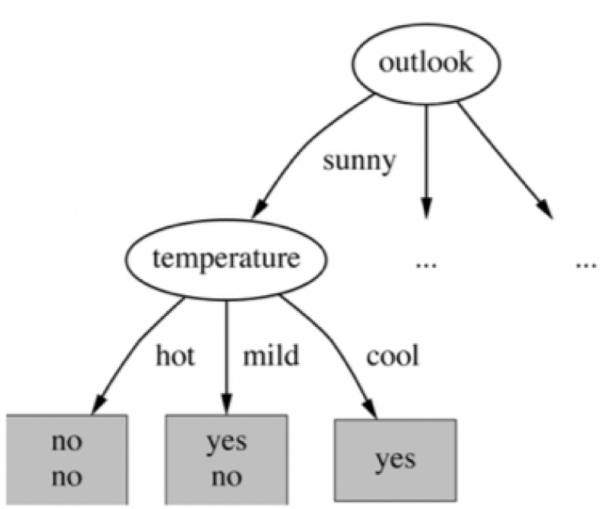
$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$

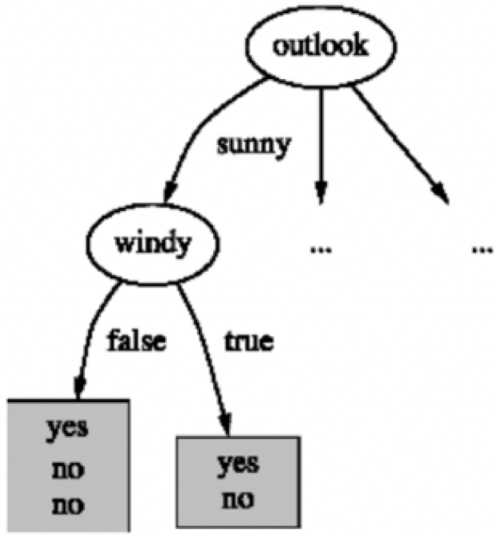
$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$

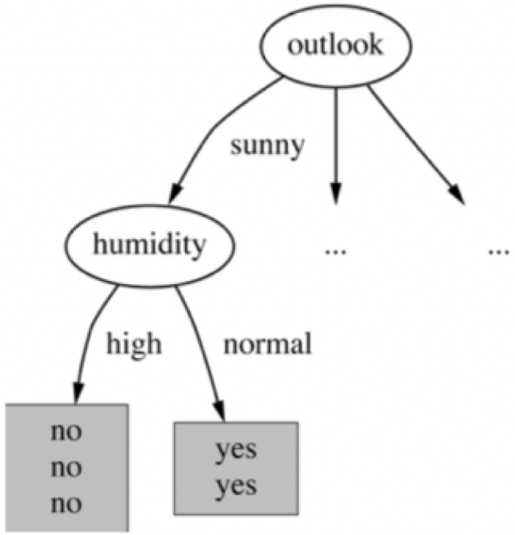
$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$

$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

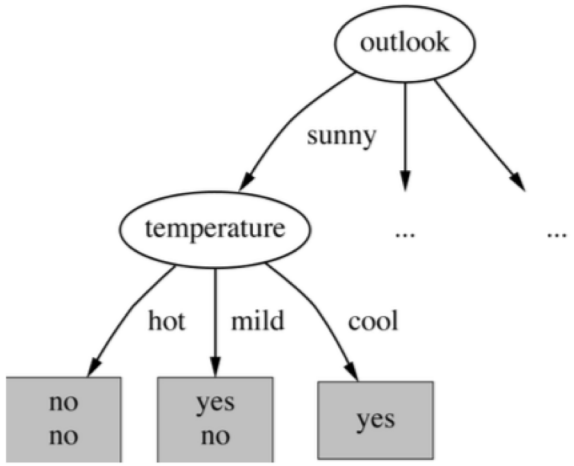
What is next?

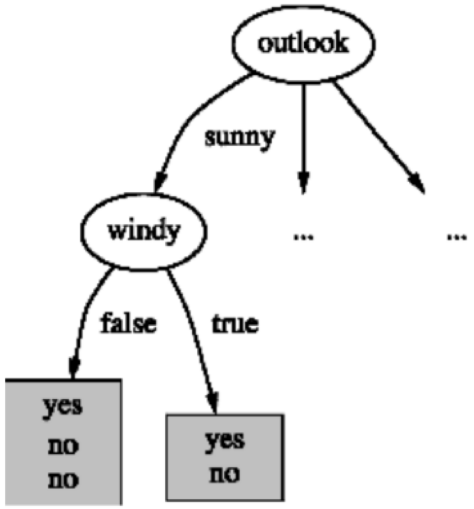

outlook
sunny
temperature
...
...
hot
mild
cool
no
no
yes
no
yes

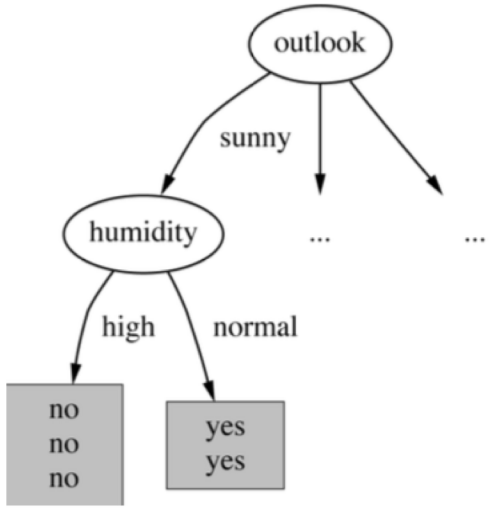

outlook
sunny
windy
...
...
false
true
yes
no
no
yes
no


outlook
sunny
humidity
...
...
high
normal
no
no
no
yes
yes

Continue for the remain data





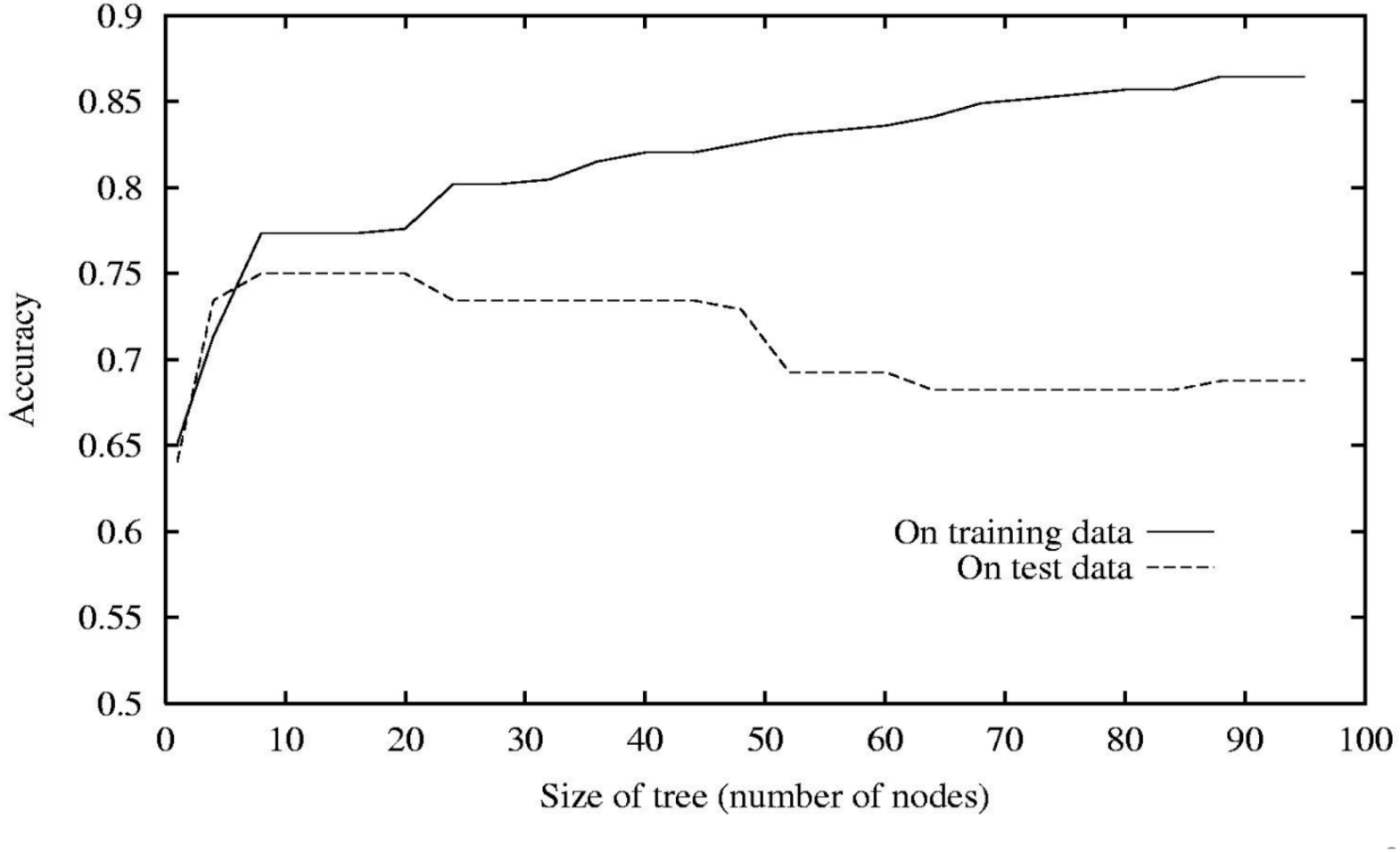


$$\text{gain}(\text{"Humidity"}) = 0.971\,\text{bits}$$

$$\text{gain}(\text{"Temperature"}) = 0.571\,\text{bits}$$

$$\text{gain}(\text{"Windy"}) = 0.020\,\text{bits}$$

Overfitting in Decision Trees



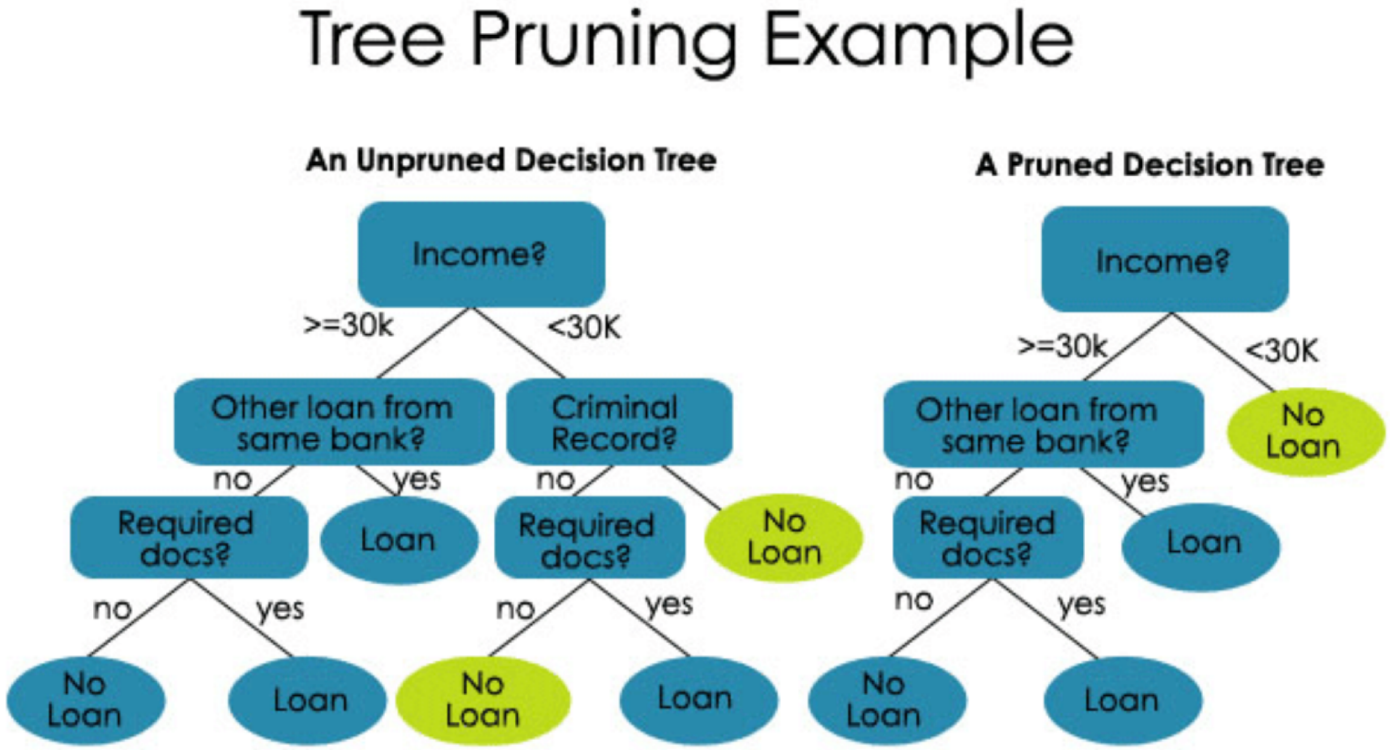
Avoiding Overfitting in Decision Trees

- How can we avoid overfitting?
  - Stop growing when data split is not statistically significant
  - Acquire more training data
  - Remove irrelevant attributes (manual process – not always possible)
  - Grow full tree, then post-prune
- How to select “best” tree:
  - Measure performance over training data
  - Measure performance over separate validation data set
  - Add complexity penalty to performance measure (heuristic: simpler is better)

Avoiding Overfitting in Decision Trees

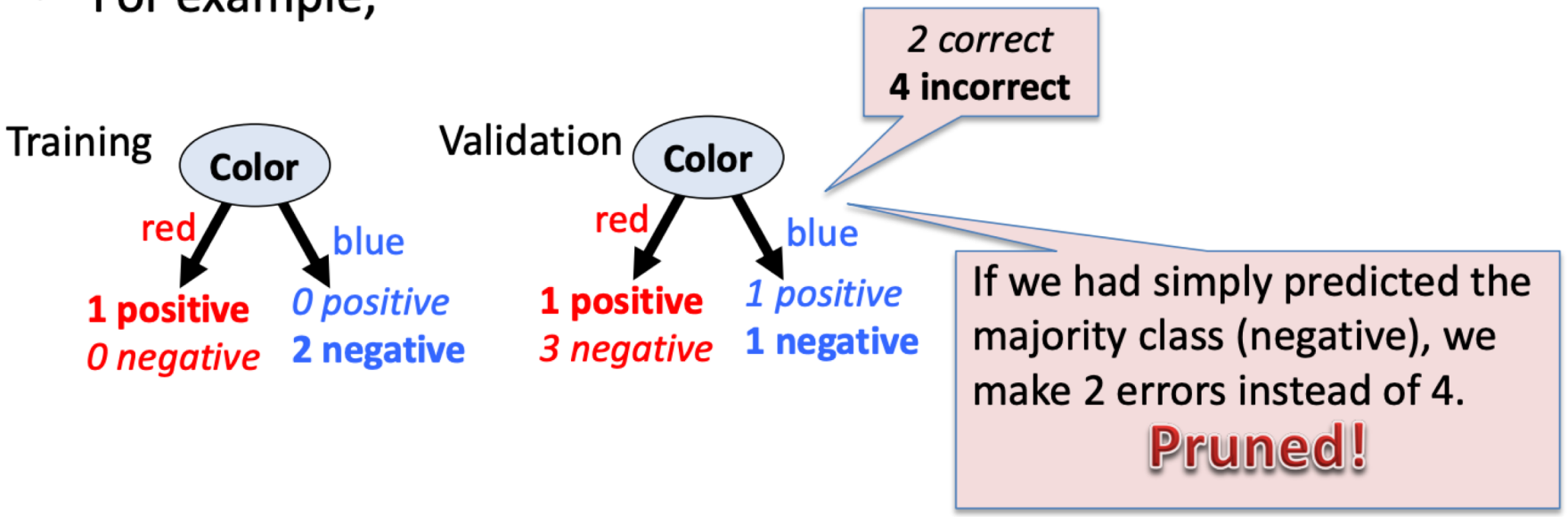
- Limiting Max-Depth
- Pruning

It reduces the size of the decision tree which might slightly increase the training error but drastically decrease the testing error. For this, it become more adaptable.



Pruning Decision Trees

- Pruning of the decision tree is accomplished by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
- For example,



Gain Ratio

- Information gain is biased for the attribute with many outcomes. It means it prefers the attribute with a large number of distinct values. For instance, consider an attribute with a unique identifier such as customer_ID has zero info(D) because of pure partition. This maximizes the information gain and creates useless partitioning.
- C4.5, an improvement of ID3, uses an extension to information gain known as the gain ratio. Gain ratio handles the issue of bias by normalizing the information gain using Split Info.

Gain Ratio

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Where,

- |Dj|/|D| acts as the weight of the jth partition.
- v is the number of discrete values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$\text{Gini}(D) = 1 - \sum_{i=1}^{m} P_i^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.
The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is:

$$\text{Gini}_A(D) = \frac{|D1|}{|D|}\text{Gini}(D_1) + \frac{|D2|}{|D|}\text{Gini}(D_2)$$

Gini index

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point.

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

The attribute with minimum Gini index is chosen as the splitting attribute.

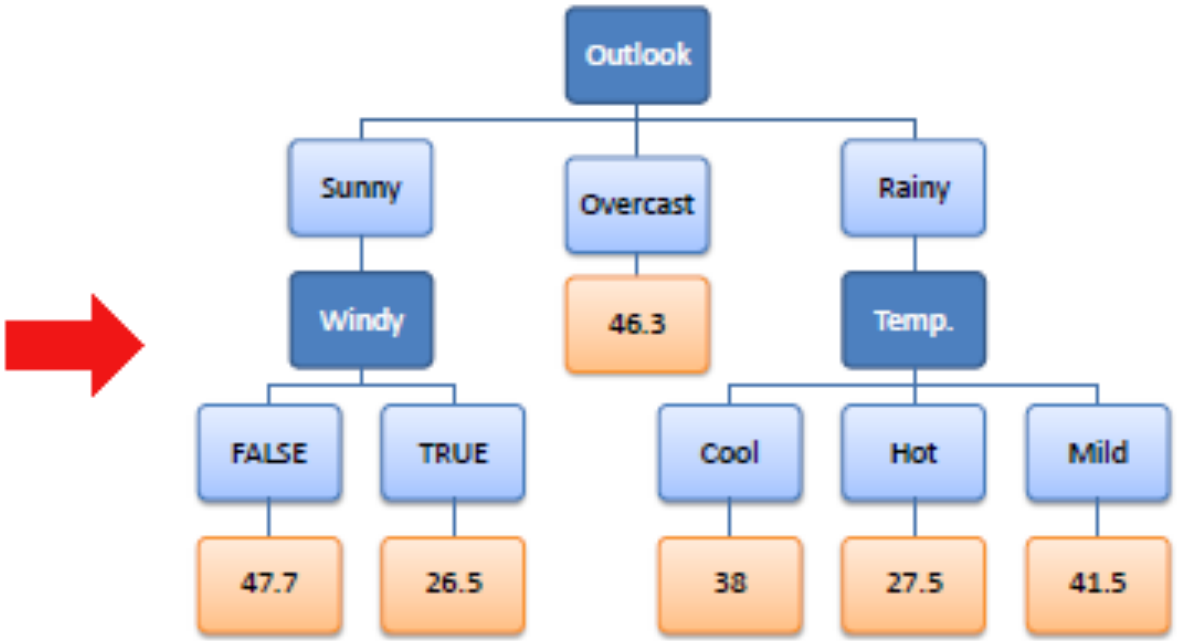
DT using sklearn

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

estimator = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
estimator.fit(X_train, y_train)
```

https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html

DT for Regression

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



https://www.saedsayad.com/decision_tree_reg.htm

DT Regressor

- The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*

Standard deviation for **one** attribute:

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$\boldsymbol{Count} = n = 14$$

$$\boldsymbol{Average} = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$\boldsymbol{Standard\ Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}} = 9.32$$

$$\boldsymbol{Coeffeicient\ of\ Variation} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

Standard deviation for **two** attributes (target and predictor):

$$S(T,X) = \sum_{c \in X} P(c)S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)*S(Sunny) + **P**(Overcast)*S(Overcast) + **P**(Rainy)*S(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Step 1: The standard deviation of the target is calculated.

Standard deviation (Hours Played) = 9.32

Step 2: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

$$SDR(T,X) = S(T) - S(T,X)$$

SDR(Hours , Outlook) = **S**(Hours) – **S**(Hours, Outlook)

= 9.32 – 7.66 = 1.66

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR= 0.48		

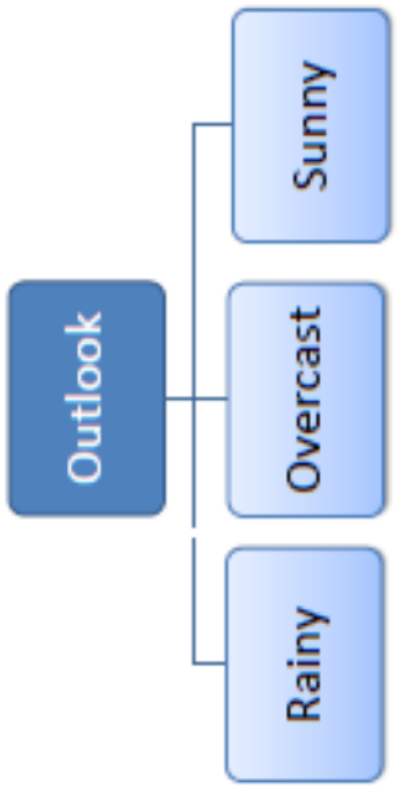
		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

Step 3: The attribute with the largest standard deviation reduction is chosen for the decision node.

★		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

Step 4a: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.



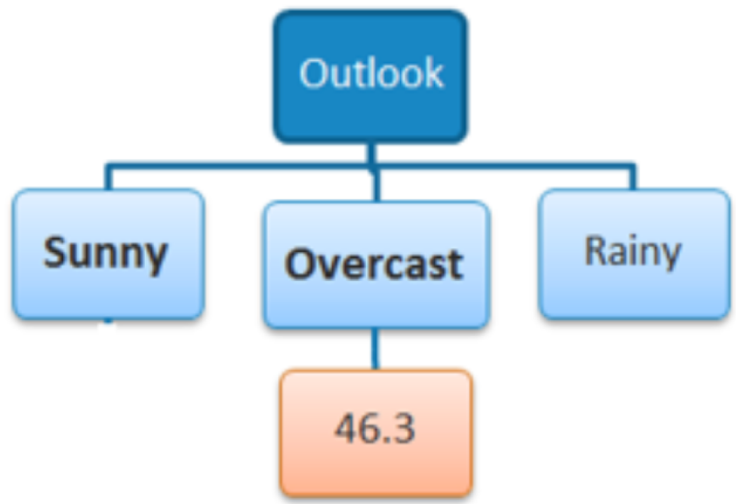
Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

In practice, we need some termination criteria. For example, when coefficient of deviation (**CV**) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (**n**) remain in the branch (e.g., 3).

Step 4b: "Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.

Outlook - Overcast

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



Step 4c: However, the "Sunny" branch has an CV (28%) more than the threshold (10%) which needs further splitting. We select "Temp" as the best best node after "Outlook" because it has the largest SDR.

Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	**45**
Cool	Normal	FALSE	**52**
Cool	Normal	TRUE	**23**
Mild	Normal	FALSE	**46**
Mild	High	TRUE	**30**
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

SDR = 10.87-((2/5)*14.5 + (3/5)*7.32) = **0.678**

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

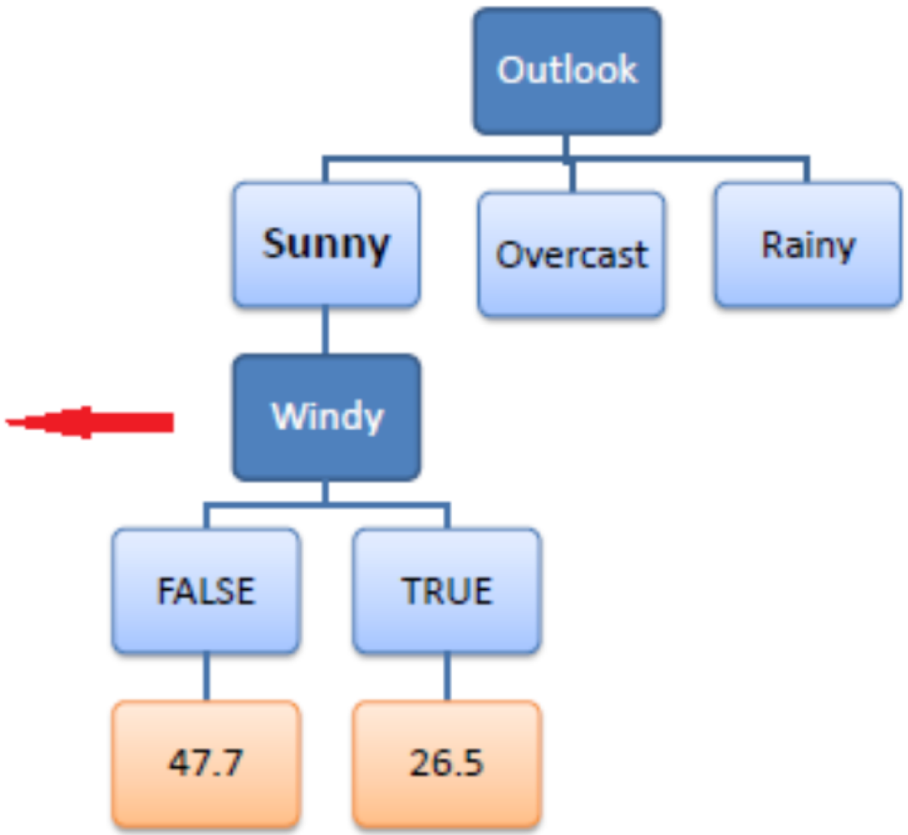
SDR = 10.87-((2/5)*7.5 + (3/5)*12.5) = **0.370**

		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

SDR = 10.87-((3/5)*3.09 + (2/5)*3.5) = **7.62**

Because the number of data points for both branches (FALSE and TRUE) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.

Temp.	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30



Step 4d: Moreover, the "rainy" branch has an CV (22%) which is more than the threshold (10%). This branch needs further splitting. We select "Temp" as the best best node because it has the largest SDR.

Outlook - Rainy

Temp	Humidity	Windy	Hours Played
Hot	High	FALSE	**25**
Hot	High	TRUE	**30**
Mild	High	FALSE	**35**
Cool	Normal	FALSE	**38**
Mild	Normal	TRUE	**48**
			S = 7.78
			AVG = 35.2
			CV = 22%

		Hours Played (StDev)	Count
Temp	Cool	0	1
	Hot	2.5	2
	Mild	6.5	2

SDR = 7.78 - ((1/5)*0+(2/5)*2.5 + (2/5)*6.5) = **4.18**

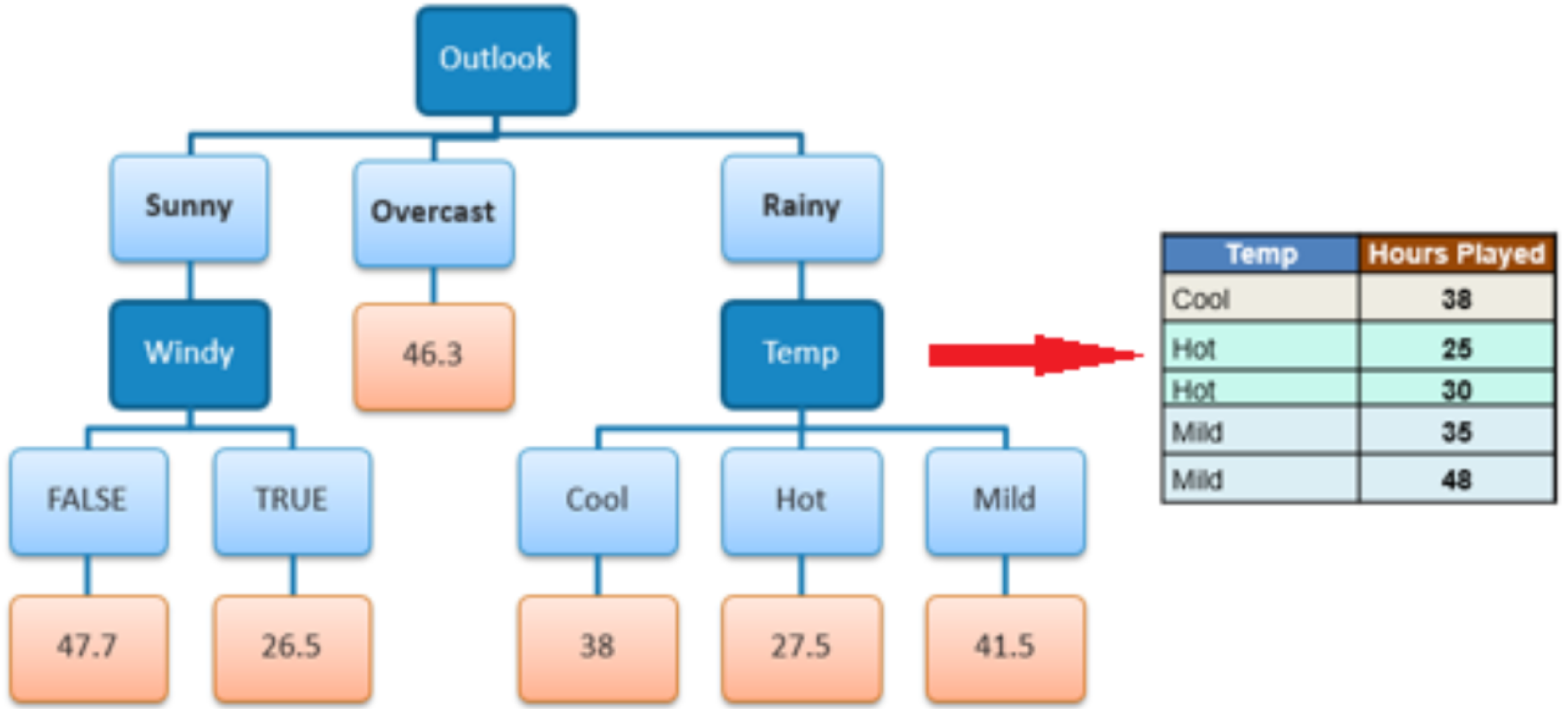
		Hours Played (StDev)	Count
Humidity	High	4.1	3
	Normal	5.0	2

SDR = 7.78 - ((3/5)*4.1 + (2/5)*5.0) = **3.32**

		Hours Played (StDev)	Count
Windy	False	5.6	3
	True	9.0	2

SDR = 7.78 - ((3/5)*5.6 + (2/5)*9.0) = **0.82**

Because the number of data points for all three branches (Cool, Hot and Mild) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.



Temp	Hours Played
Cool	38
Hot	25
Hot	30
Mild	35
Mild	48

When the number of instances is more than one at a *leaf node* we calculate the *average* as the final value for the target.

sklearn.tree.DecisionTreeRegressor

Examples

```
>>> from sklearn.datasets import load_diabetes
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.tree import DecisionTreeRegressor
>>> X, y = load_diabetes(return_X_y=True)
>>> regressor = DecisionTreeRegressor(random_state=0)
>>> cross_val_score(regressor, X, y, cv=10)
...
...
array([-0.39..., -0.46...,  0.02...,  0.06..., -0.50...,
       0.16...,  0.11..., -0.73..., -0.30..., -0.00...])
```