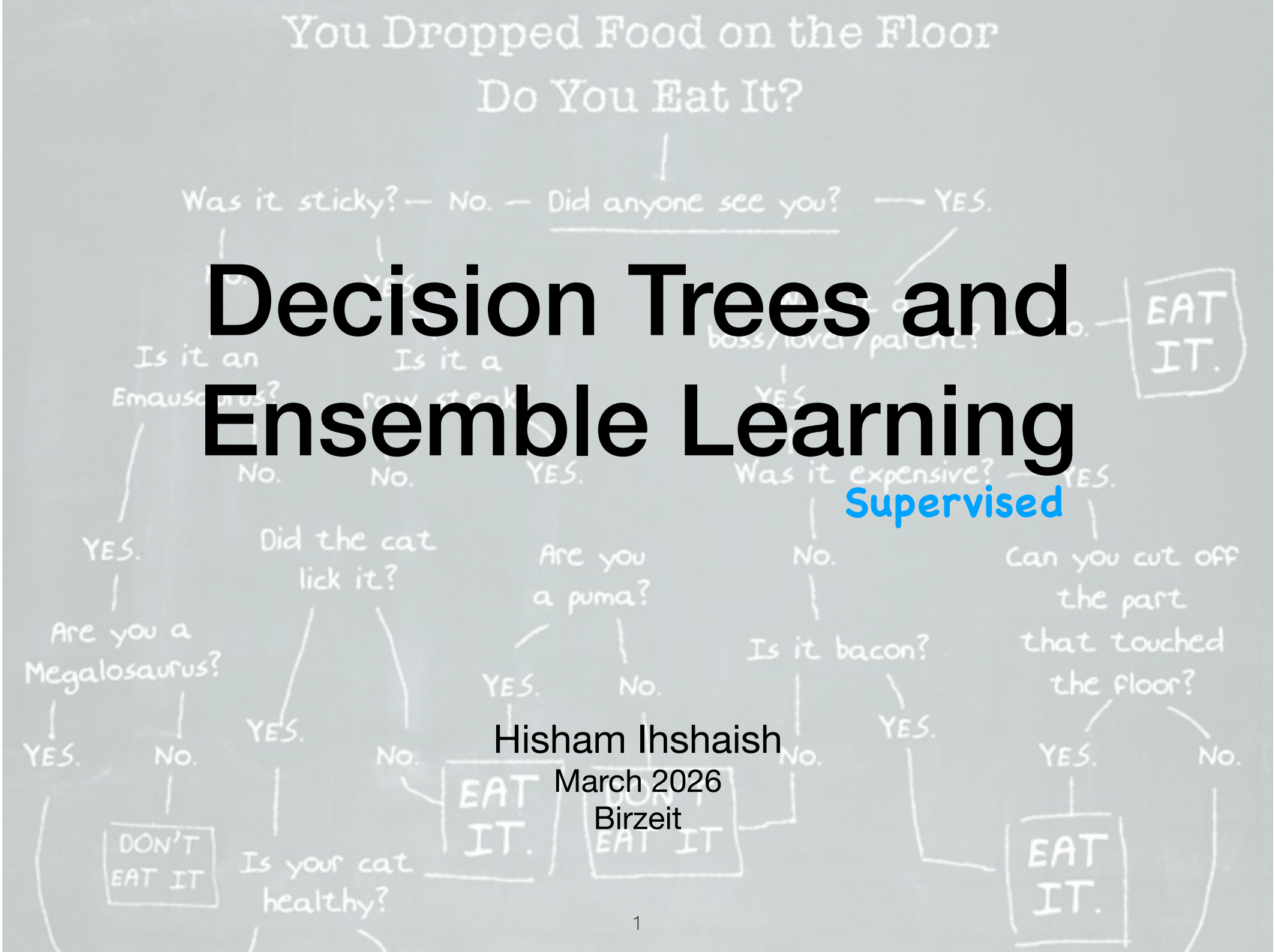


Decision Trees and Ensemble Learning

Hisham Ihshaish

March 2026

Birzeit

Agenda

- Decision Trees Induction
  - Context
  - Estimation/“design”
  - Properties
- Ensemble Learning
  - Bagging (mainly - I’m hoping to cover more next week)
- Summary
- Tutorial

Decision Trees

- Simple (or maybe the simplest), yet **can be** powerful, or amongst most powerful learners, for classification (mainly) and regression problems.

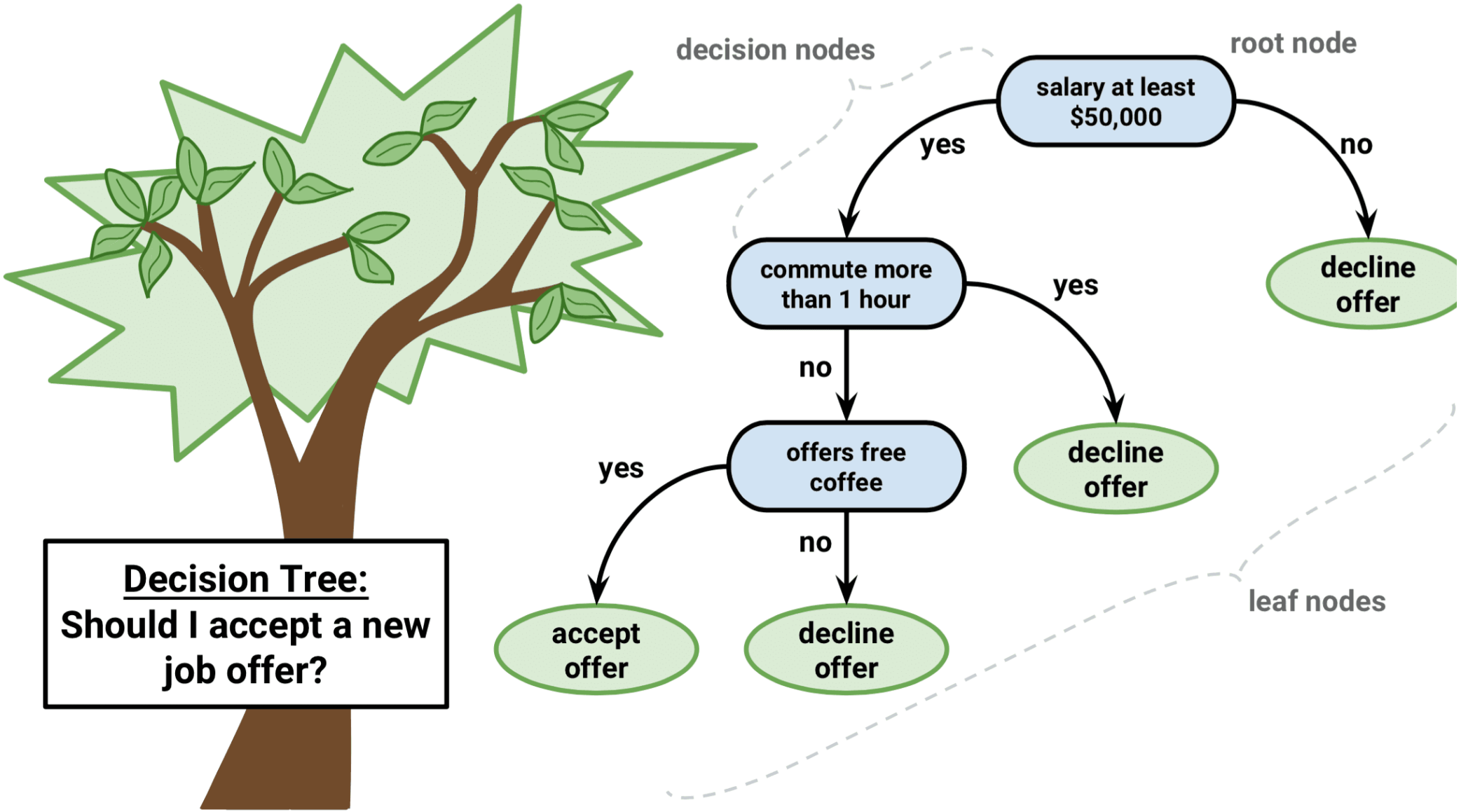
Decision Trees

- A very popular model for both **regression** and **classification** tasks — regardless of linearity (huh!)
- ~~Non-parametric.~~

Decision Trees

- Random Forests (*a variant* - later on in the lecture) is one of the most used models for predictive analytics in practice - because they’re good ;)

Decision Trees

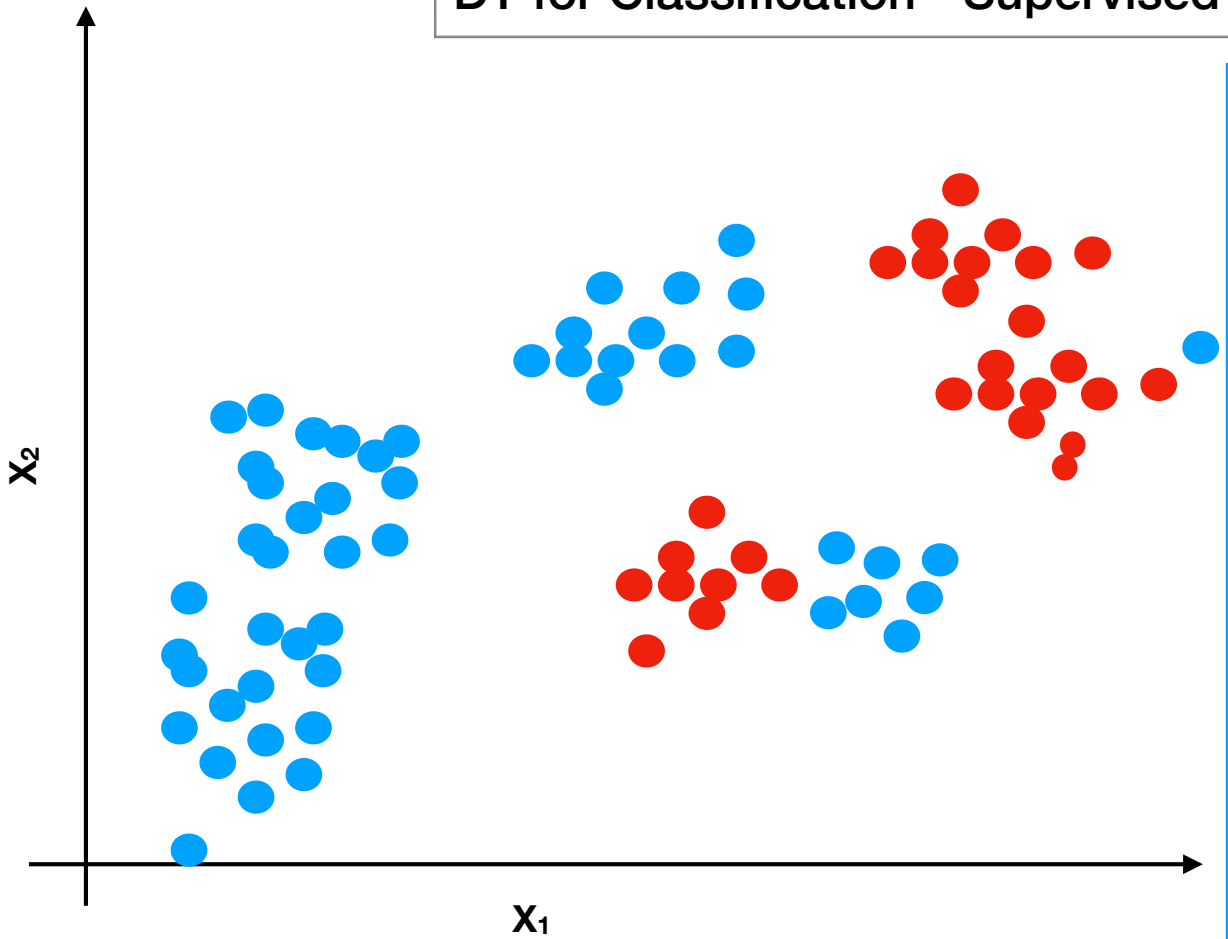


- Decision Trees (simplified) is the task of learning a hierarchy of **if/else** questions, leading to a decision (leaf).

Image credit: www.packtpub.com

DT for Classification

DT for Classification - Supervised Learning



➡ Data:

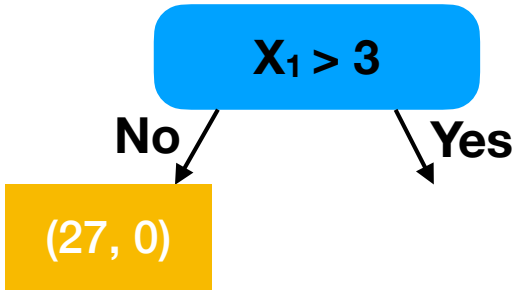
$((x_{i1},x_{i2}), y_i)$, i =1, 2, …,n

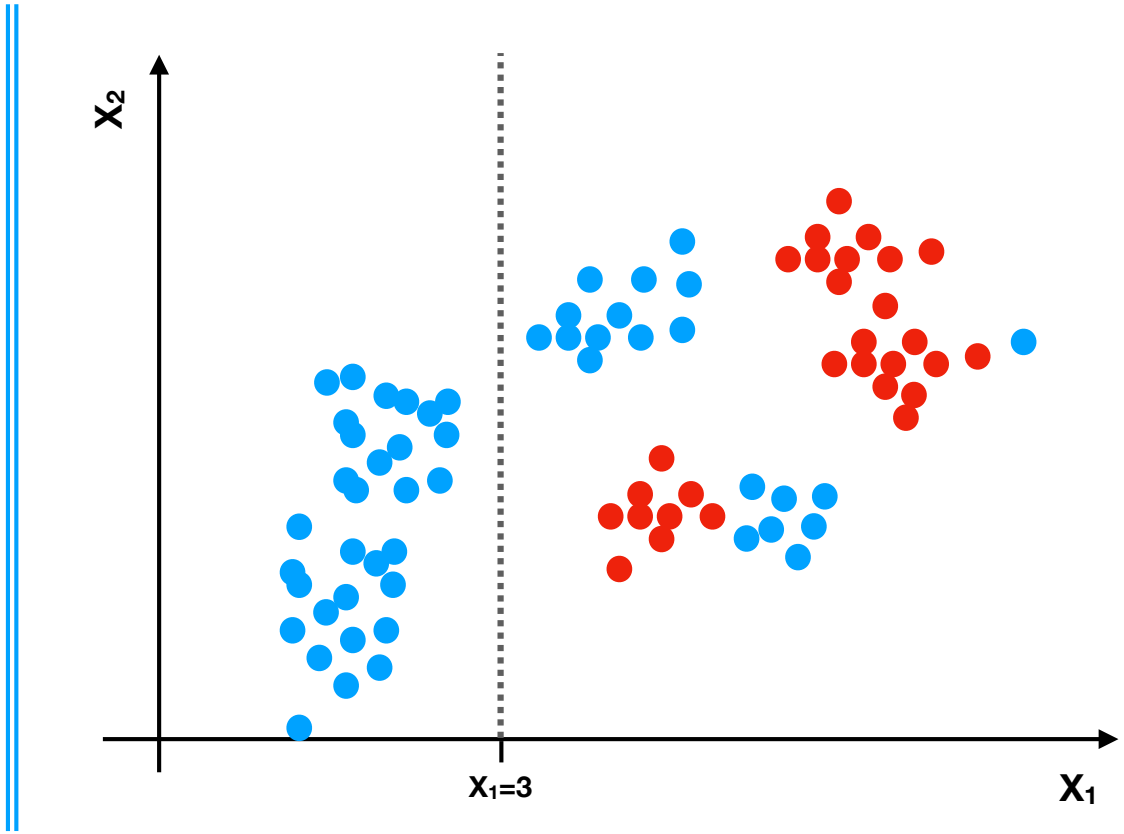
➡ Task:

E (*Y*: colour | *X*: x_1,x_2)

such that ***error*** is minimised at each Leaf.

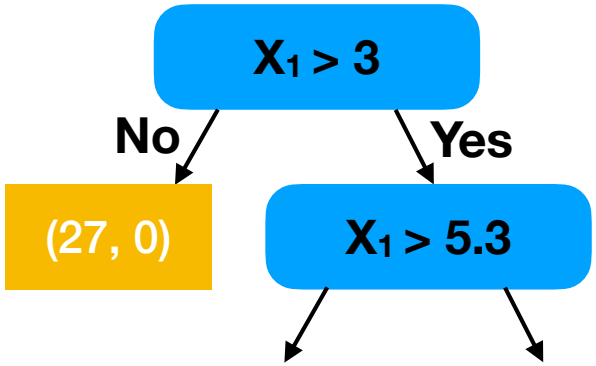
DT for Classification

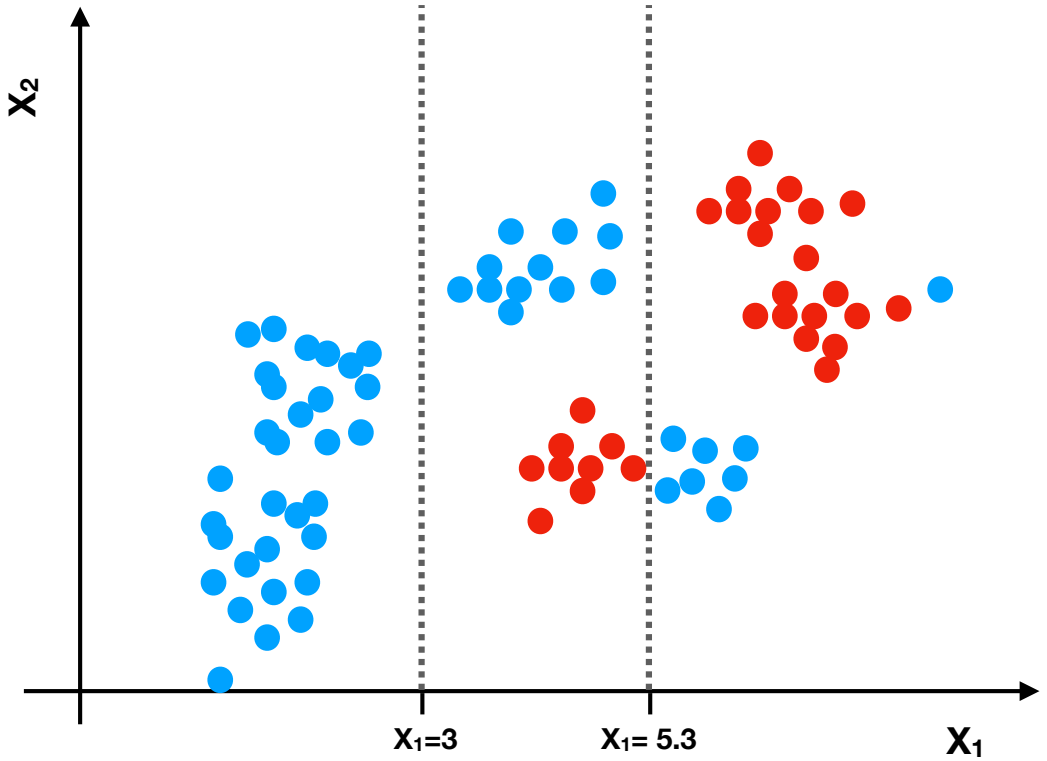




- Depth = 1
- Decision Attribute: X_1

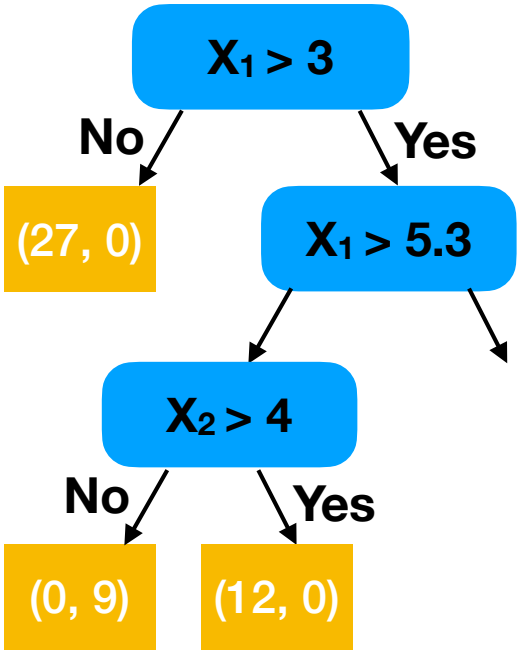
DT for Classification

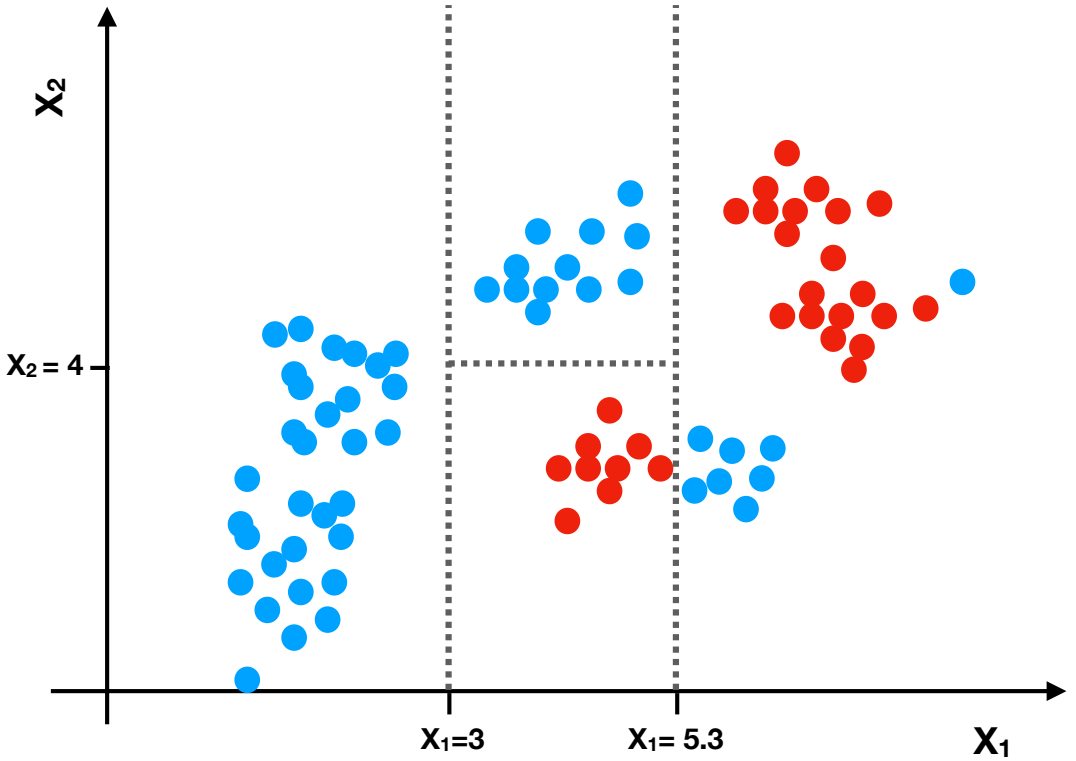




- Depth = 2
- Decision Attribute: X_1

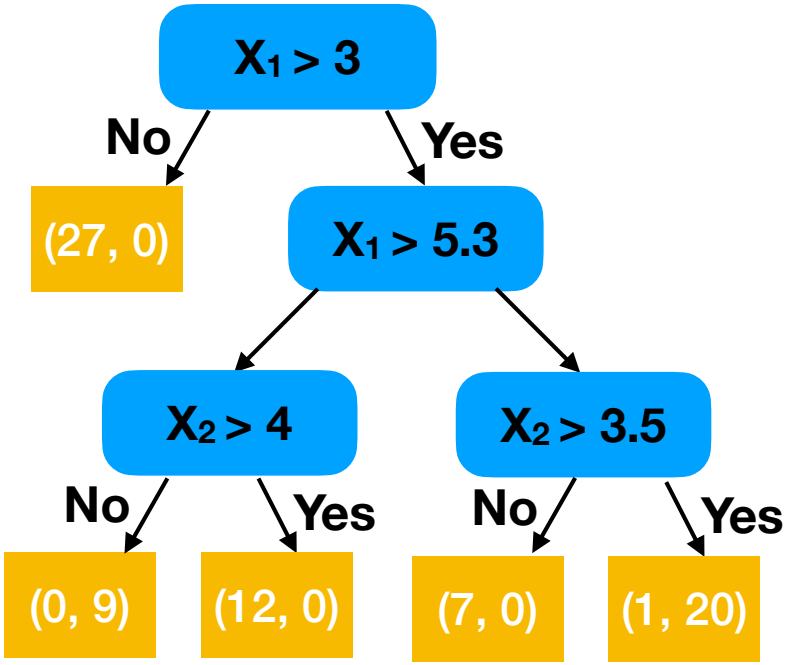
DT for Classification

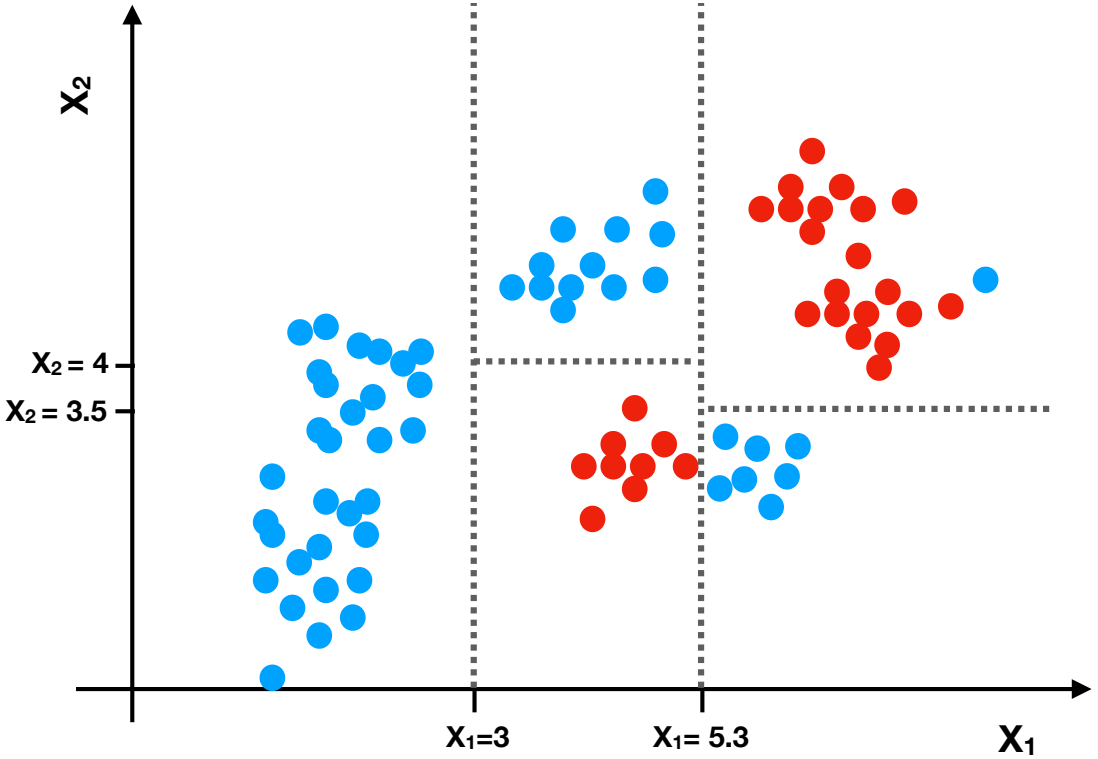




- Depth = 3
- Decision Attribute: X_2

DT for Classification



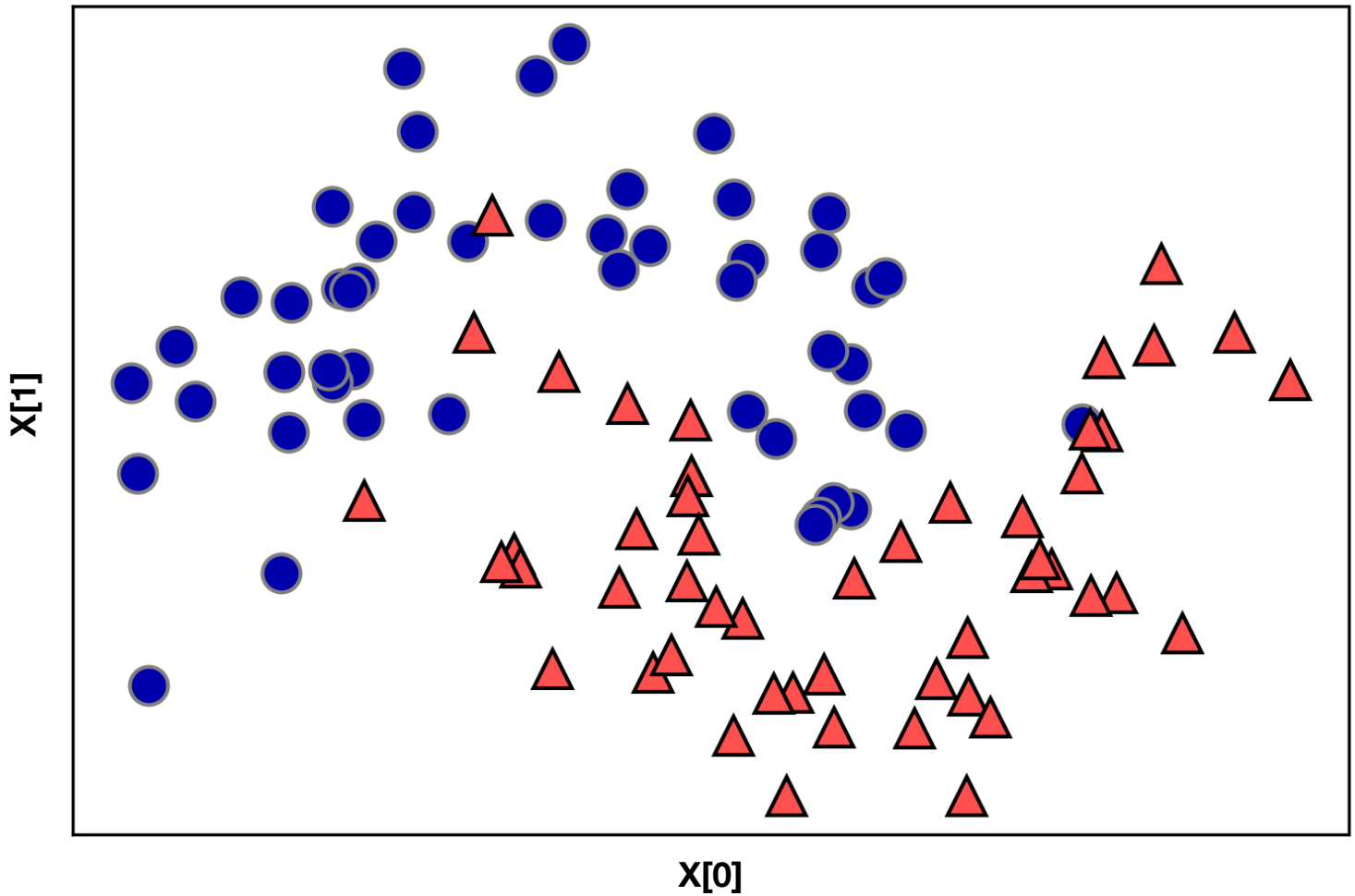


- Depth = 3
- Decision Attribute: X_2

‘two_moons’ toy dataset

- Sklearn dataset, two half moon shapes of a sample (default n =100). Labeled (50%, 50%)
- Task: classify a shape (0: not moon, 1:moon)

```
>> from sklearn.datasets import make_moons
>> moons =make_moons(n_samples=150, random_state=0)
```

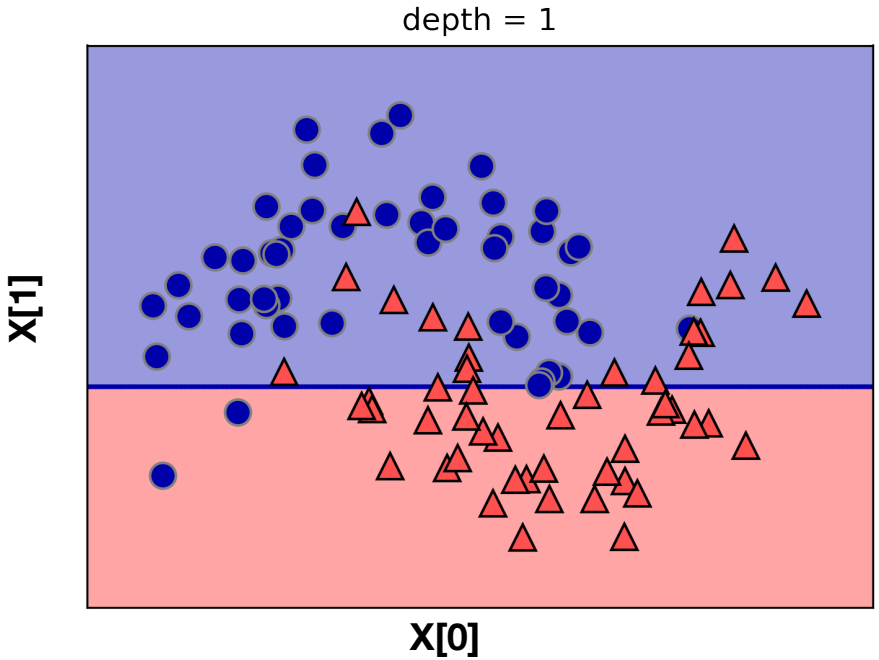


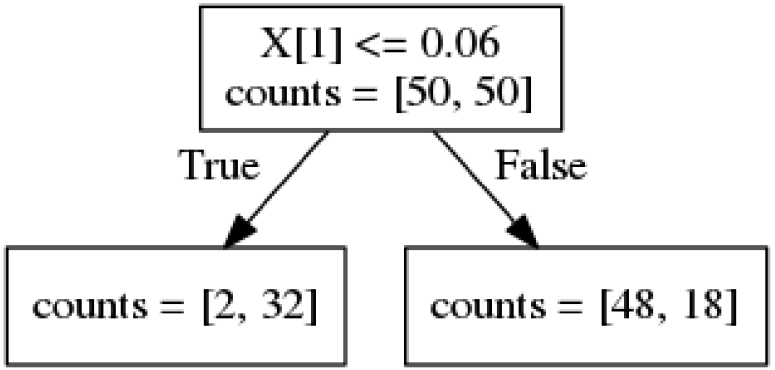
credit: A Muller, mglearn

‘two_moons’ toy dataset

- Sklearn dataset, two half moon shapes of a sample (default n =100). Labeled (50%, 50%)
- Task: classify a shape (0: not moon, 1:moon)

```
>> from sklearn.datasets import make_moons

>> moons =make_moons(n_samples=150, random_state=0)
```



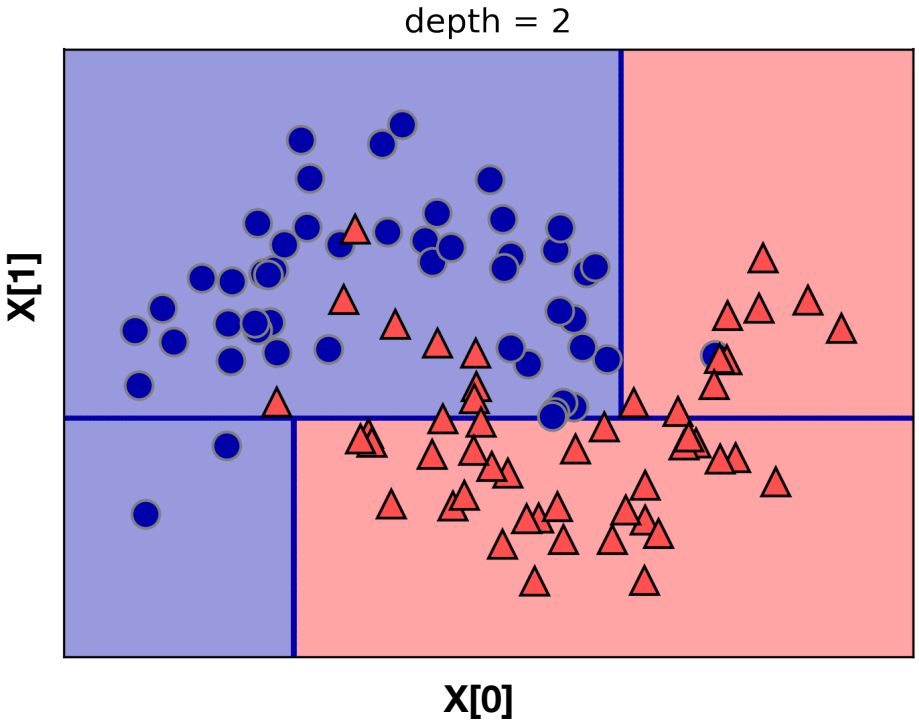


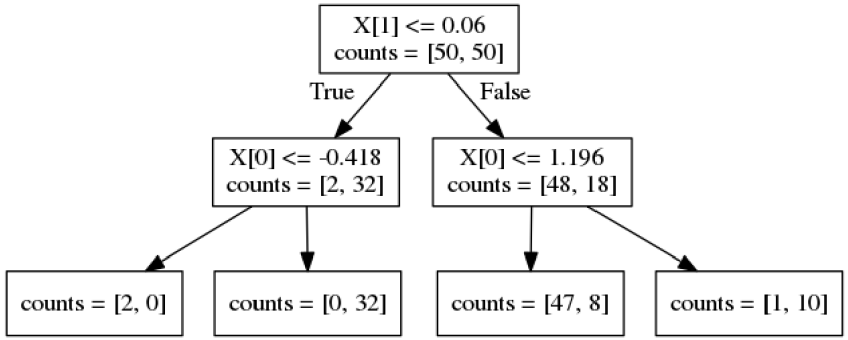
credit: A Muller, mglearn

‘two_moons’ toy dataset

- Sklearn dataset, two half moon shapes of a sample (default n =100). Labeled (50%, 50%)
- Task: classify a shape (0: not moon, 1:moon)

```
>> from sklearn.datasets import make_moons

>> moons =make_moons(n_samples=150, random_state=0)
```



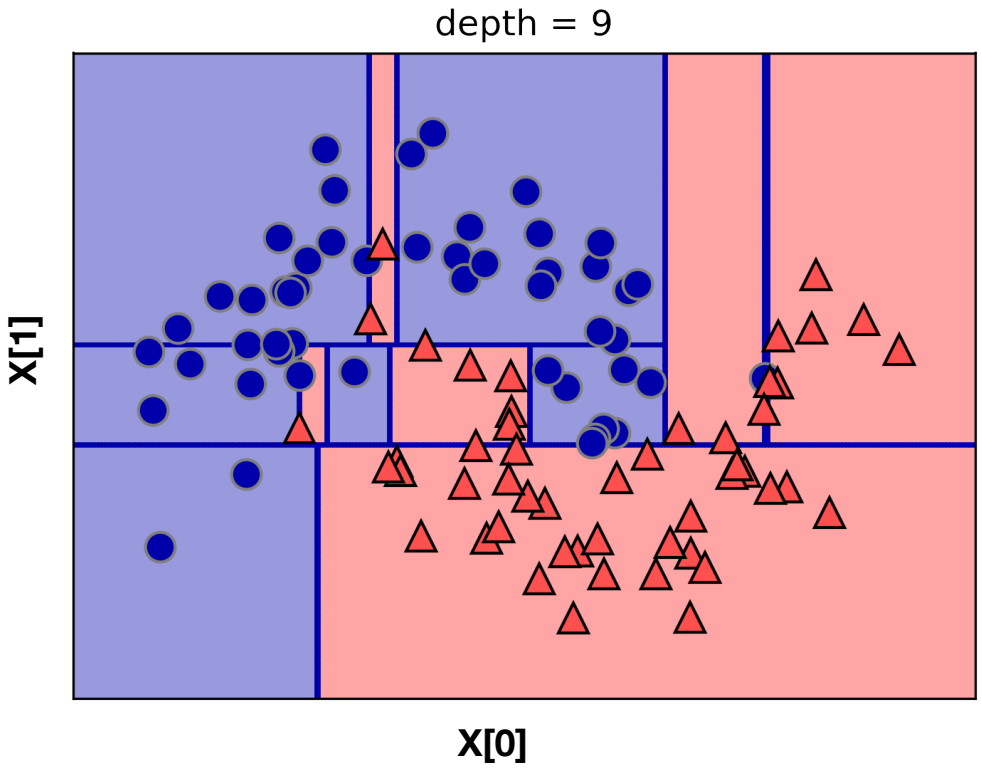


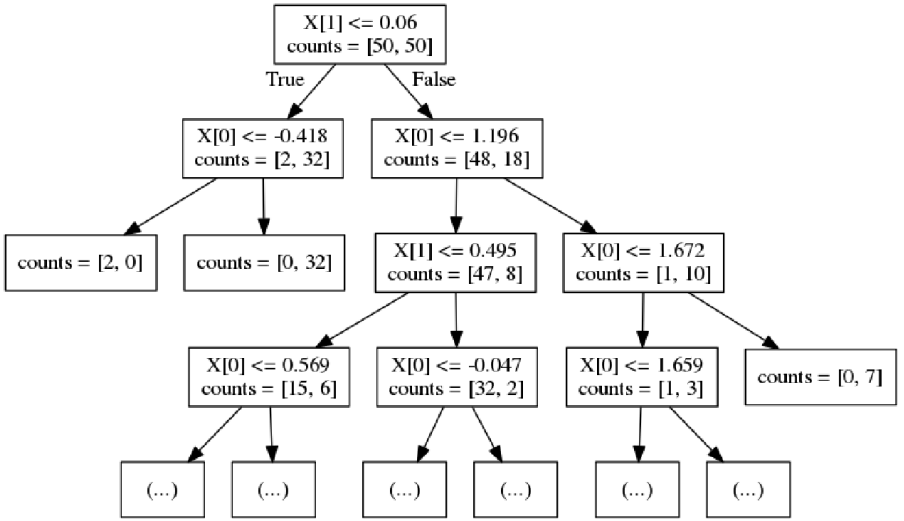
credit: A Muller, mglearn

‘two_moons’ toy dataset

- Sklearn dataset, two half moon shapes of a sample (default n =100). Labeled (50%, 50%)
- Task: classify a shape (0: not moon, 1:moon)

```
>> from sklearn.datasets import make_moons

>> moons =make_moons(n_samples=150, random_state=0)
```





credit: A Muller, mglearn

Top-down Induction of decision trees

The **widely used** technique is a top-down, greedy search approach:

1. Choose the **best** feature x* for the root of the tree.
2. Separate training set **S** into subsets {$\mathbf{S}_1$, $\mathbf{S}_2$, .., $\mathbf{S}_k$} where each subset $\mathbf{S}_i$ contains examples having the same value for x*.
3. Recursively apply the algorithm on each new subset until all examples have the same class label.

Choosing (decision) Features

- **ID3** (Iterative Dichotomiser 3) and **C4.5** by Ross Quinlan (1986) based on Information Gain.

- **CART** by Leo Breiman: **Gini Impurity** — measure of the heterogeneoity (or "impurity") of the nodes. If all data points at one node belong to the same class then this node is considered "pure". So by minimising the Gini Impurity the decision tree finds the features they separate the data best.

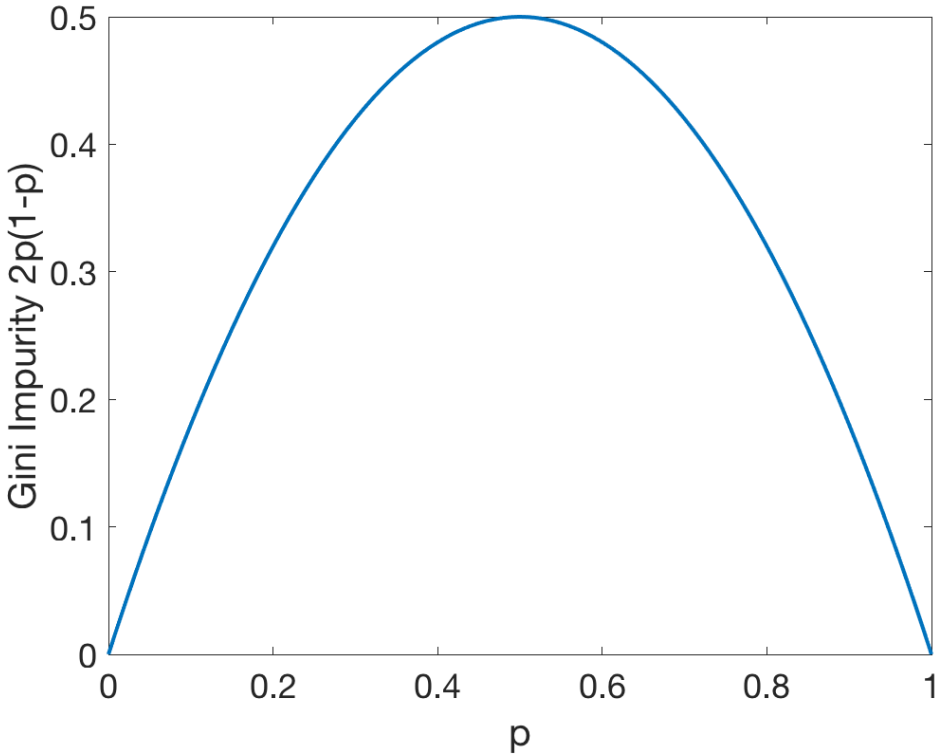
Impurity Function(s)

1 Gini Impurity → CART

Let k be the given classes in a dataset,

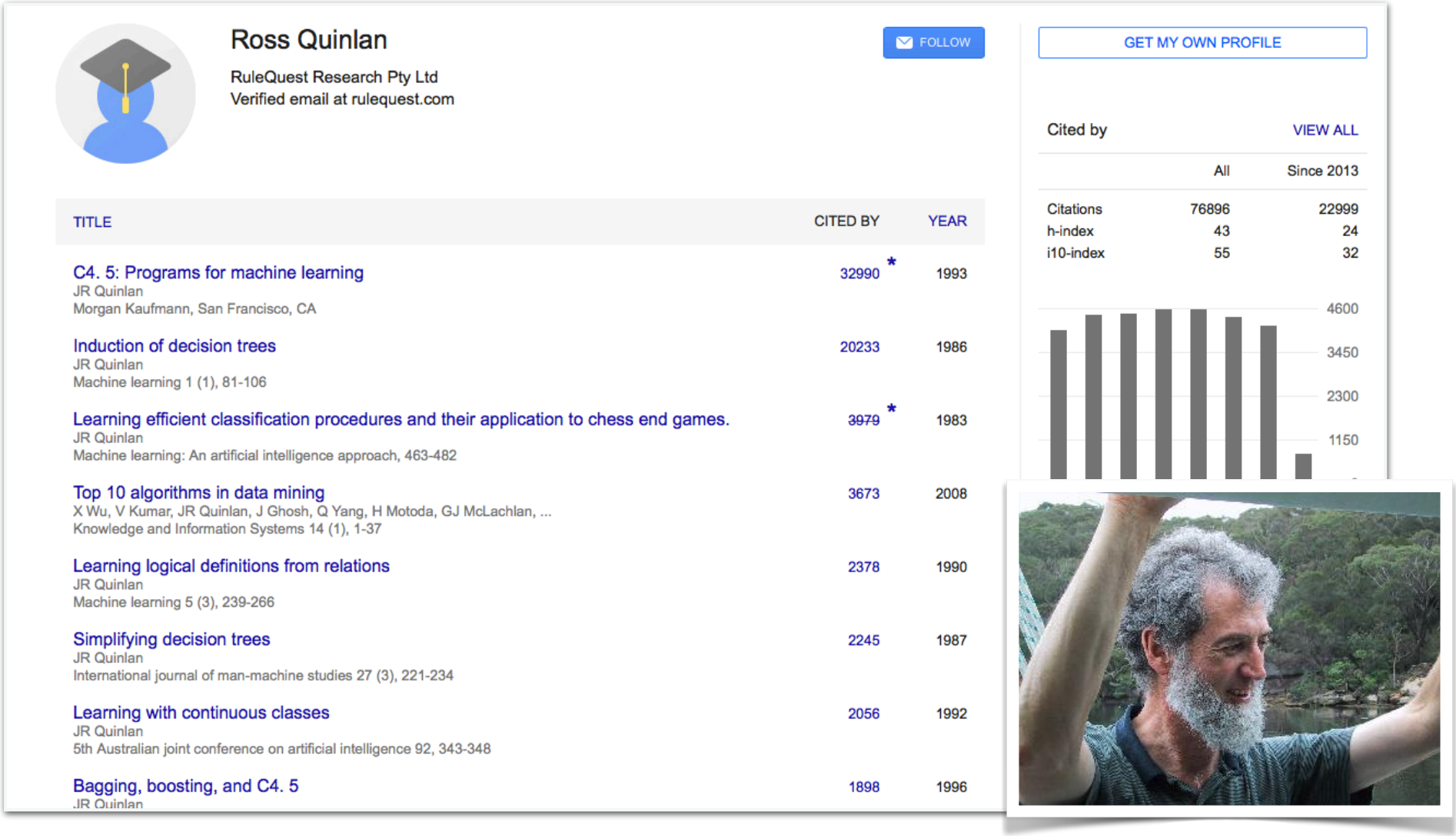
$$I_G(p) = \sum_{i=1}^{k} p_i(1 - p_i)$$

$$= 1 - \sum_{i=1}^{k} p_i^2$$



	Count		Probability		Gini Impurity
	n_1	n_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Feature A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Feature B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Feature C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

2 Entropy → ID3



Ross Quinlan

RuleQuest Research Pty Ltd
Verified email at rulequest.com

FOLLOW

GET MY OWN PROFILE

TITLE	CITED BY	YEAR
C4. 5: Programs for machine learning JR Quinlan Morgan Kaufmann, San Francisco, CA	32990 *	1993
Induction of decision trees JR Quinlan Machine learning 1 (1), 81-106	20233	1986
Learning efficient classification procedures and their application to chess end games. JR Quinlan Machine learning: An artificial intelligence approach, 463-482	3979 *	1983
Top 10 algorithms in data mining X Wu, V Kumar, JR Quinlan, J Ghosh, Q Yang, H Motoda, GJ McLachlan, ... Knowledge and Information Systems 14 (1), 1-37	3673	2008
Learning logical definitions from relations JR Quinlan Machine learning 5 (3), 239-266	2378	1990
Simplifying decision trees JR Quinlan International journal of man-machine studies 27 (3), 221-234	2245	1987
Learning with continuous classes JR Quinlan 5th Australian joint conference on artificial intelligence 92, 343-348	2056	1992
Bagging, boosting, and C4. 5 JR Quinlan	1898	1996

Cited by VIEW ALL

	All	Since 2013
Citations	76896	22999
h-index	43	24
i10-index	55	32

J.R. Quinlan, *Induction of Decision Trees, Machine Learning* 1: 81—106, 1986.
Available: http://hunch.net/~coms-4771/quinlan.pdf

2 Entropy → ID3

- Let *X* be a random variable with the following probability distribution

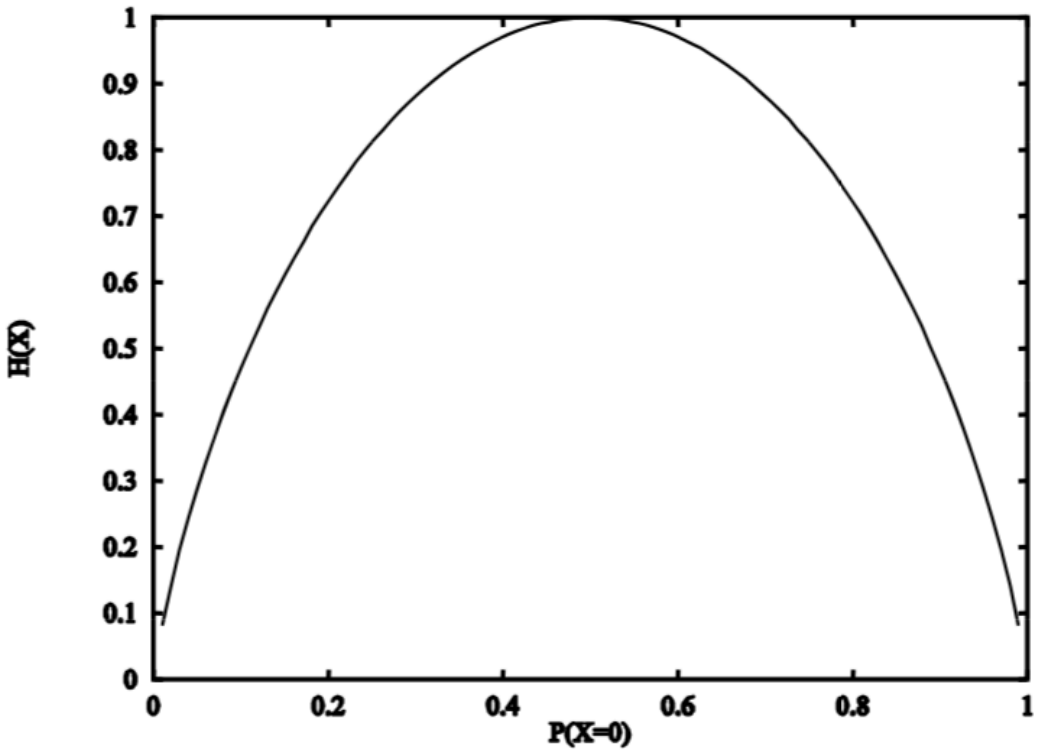
P(X = 0)	*P(X = 1)*
0.2	0.8

- The entropy of ***X***, denoted *H(**X**)*, is defined as

$$H(X) = -\sum_{x} P_X(x) \log_2 P_X(x)$$

- **Entropy** measures the uncertainty of a random variable
- The larger the entropy, the more uncertain we are about the value of *X*
- If P(X=0)=0 (or 1), there is no uncertainty about the value of X, entropy = 0
- If P(X=0)=P(X=1)=0.5, the uncertainty is maximized, entropy = 1

Entropy of random variable X



Entropy is applied as a term of the heterogeneity of a split/sample; If the sample is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one.

If P(X=0)=0 (or 1), there is no uncertainty about the value of X, entropy = 0

If P(X=0)=P(X=1)=0.5, the uncertainty is maximized, entropy = 1

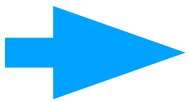
Mutual Information

The mutual information measure of two random variables *X* and *Y* is defined as:

$$I(X,Y) = H(Y) - H(Y|X)$$

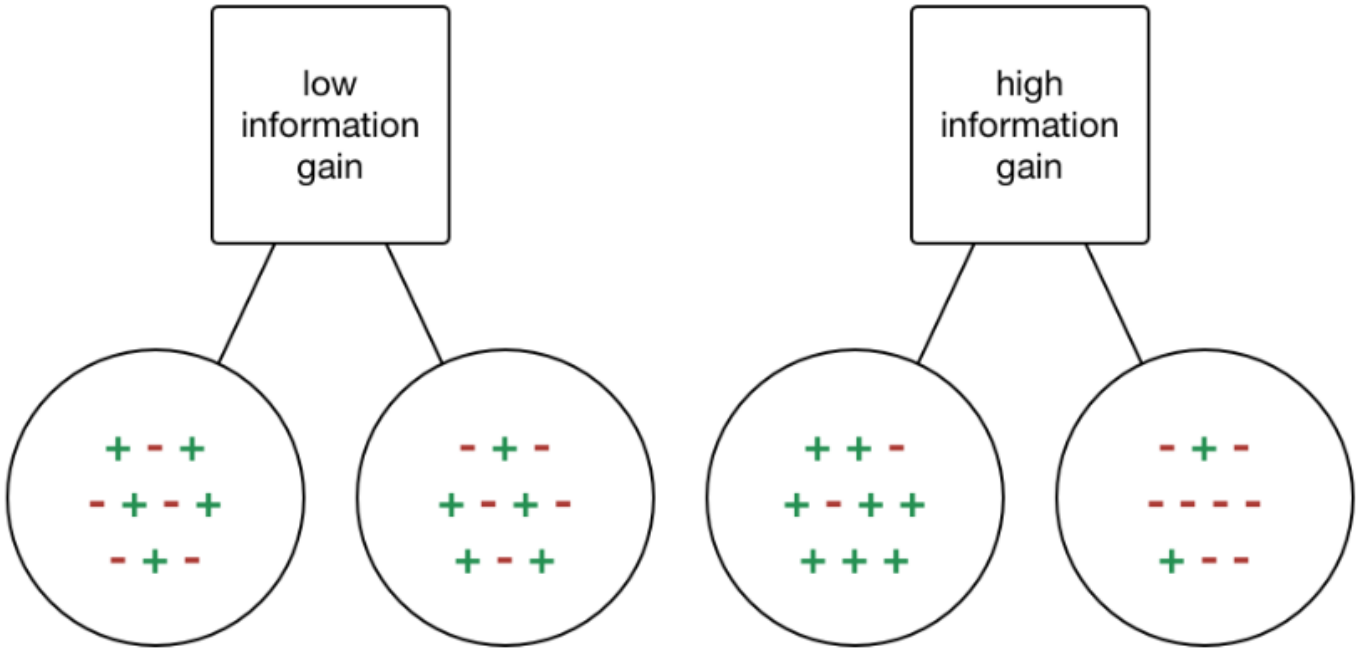
the amount of information we learn about Y by knowing the value of X (and vice versa – it is symmetric).

Information Gain (IG) = $\underset{j}{\operatorname{argmax}}\ H(Y) - H(Y|X_j)$



$\underset{j}{\operatorname{argmin}}\ H(Y|X_j)$

Information Gain

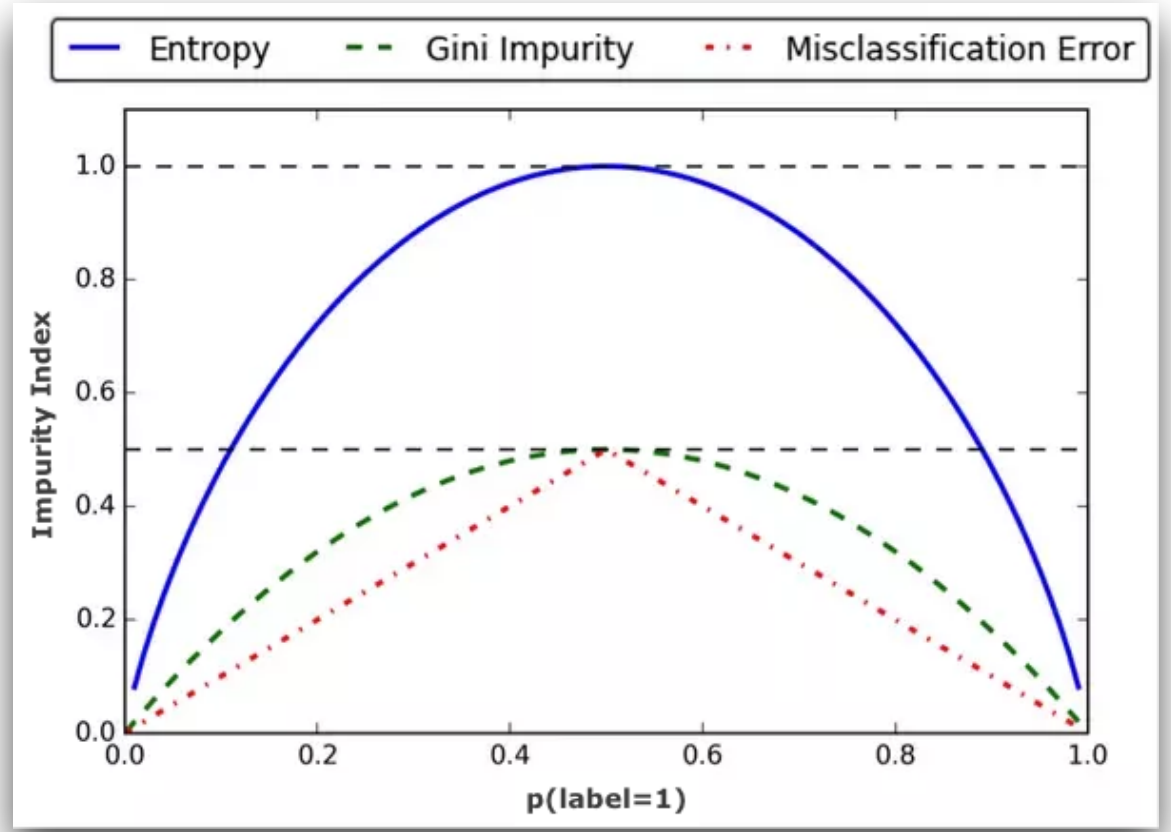


Information Gain (IG) = $\underset{j}{\operatorname{argmax}}\ H(Y) - H(Y|X_j)$

$\underset{j}{\operatorname{argmin}}\ H(Y|X_j)$

Gini and Entropy

- Gini : $I_G(S) = 1 - \sum_{j=1}^{c} p_j^2$
- Entropy : $H(S) = -\sum_{j=1}^{c} p_j \log p_j$



Example: Play Golf dataset - available on Kaggle

Attributes | **class**

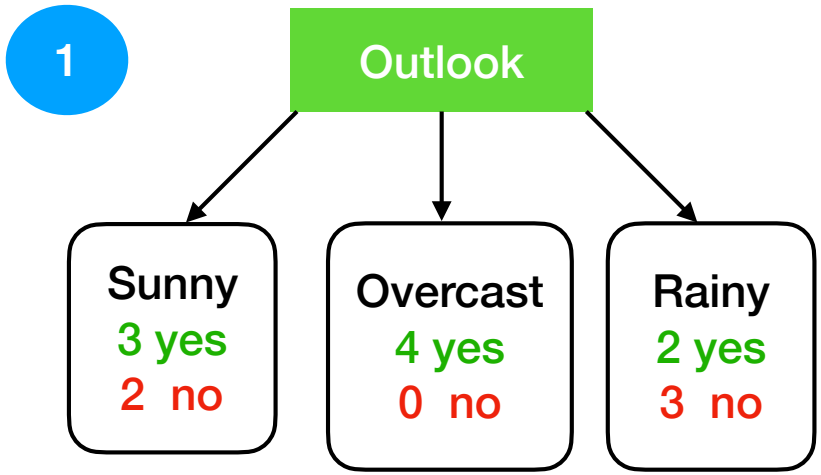
	Outlook	Temperature	Humidity	Windy	Play Golf
1	Rainy	Hot	High	FALSE	No
2	Rainy	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Sunny	Mild	High	FALSE	Yes
5	Sunny	Cool	Normal	FALSE	Yes
6	Sunny	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Rainy	Mild	High	FALSE	No
9	Rainy	Cool	Normal	FALSE	Yes
10	Sunny	Mild	Normal	FALSE	Yes
11	Rainy	Mild	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
14	Sunny	Mild	High	TRUE	No

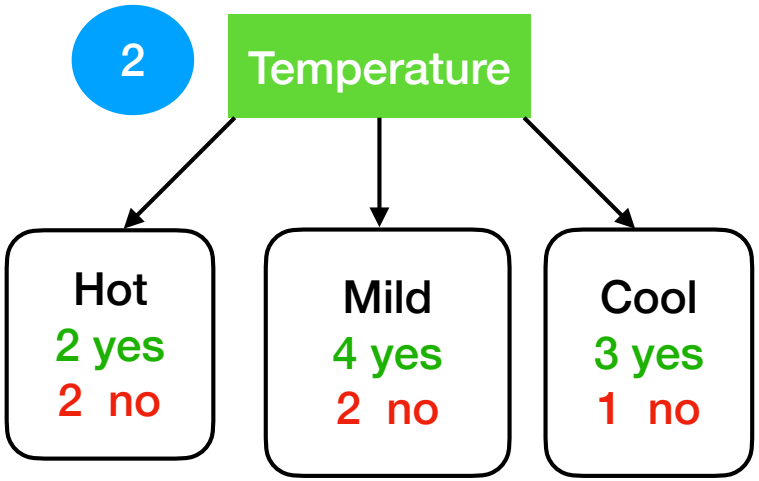
- 4 features:
  - **outlook**: *rainy, overcast, sunny*
  - **temperature**: *cool, mild, hot*
  - **humidity**: *normal, high*
  - **windy**: *false, true*
- Possible outcomes (play golf?):
  - **false**
  - **true**

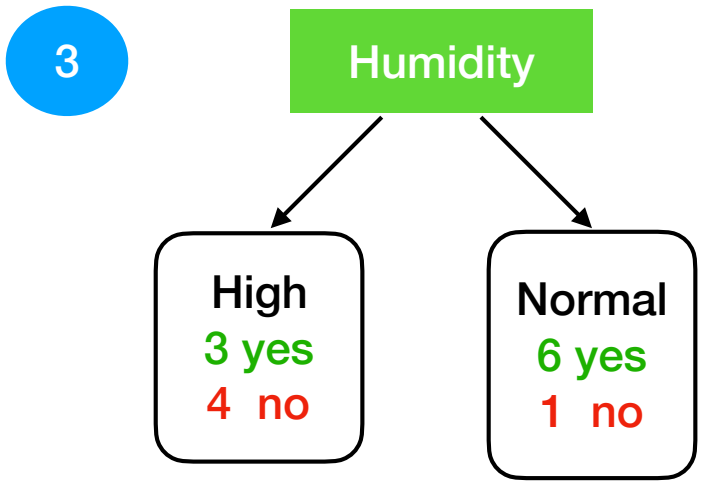
Frequency Table

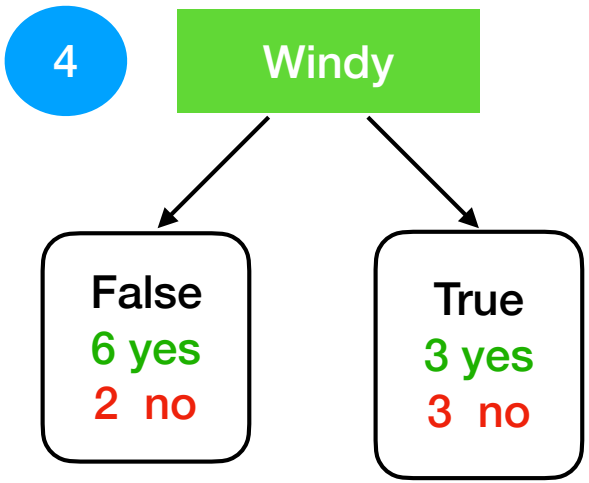
Play golf	
yes	no
9	5

Potential Splits on X (attributes)









Play not play Tree!

Let S be the set of training samples with c possible classes, thus $S = \{S_1, S_2, ..., S_n\}$

$$\text{Entropy: } H(S) = -\sum_{i=1}^{C} p_i \cdot \log(p_i) = -\sum_{i=1}^{C} \frac{|S_i|}{|S|} \cdot \log(\frac{|S_i|}{|S|})$$

Play golf
yes | no
9 | 5

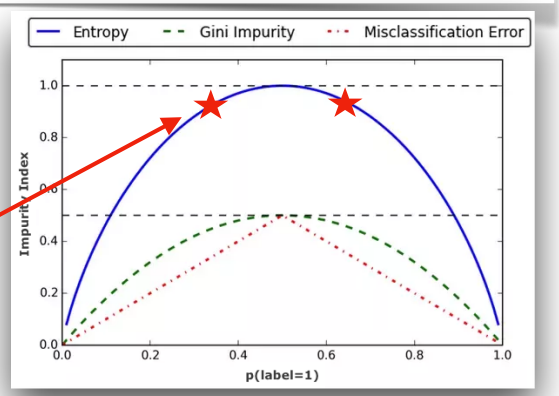
-> H(S) = 0.94

```
from math import log

def entropy(*probs):

  try:
    total = sum(probs)
    return sum([-p / total * log(p / total, 2) for p in probs])
  except:
    return 0

print(entropy(6, 5), entropy(1, 2), entropy(2, 2), entropy (9,5), entropy (5,0))
```

```
0.9940302114769565 0.9182958340544896 1.0 0.9402859586706309 0
```

Entropy [my data = entropy (9,5)]:

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



Play not play Tree!

Let S be the set of training samples with c possible classes, thus $S = \{S_1, S_2, ..., S_n\}$

{number of observations of class 1 (i) over the total number of observations}

$$\text{Entropy: } H(S) = -\sum_{i=1}^{C} p_i \cdot \log(p_i) = -\sum_{i=1}^{C} \frac{|S_i|}{|S|} \cdot \log(\frac{|S_i|}{|S|})$$

Play golf
yes | no
9 | 5

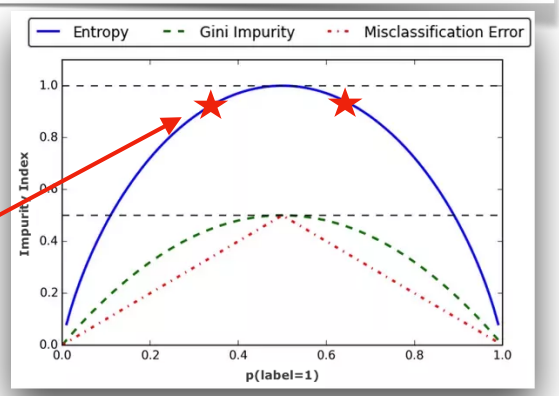
-> H(S) = 0.94

```
from math import log

def entropy(*probs):

  try:
    total = sum(probs)
    return sum([-p / total * log(p / total, 2) for p in probs])
  except:
    return 0

print(entropy(6, 5), entropy(1, 2), entropy(2, 2), entropy (9,5), entropy (5,0))
```

```
0.9940302114769565 0.9182958340544896 1.0 0.9402859586706309 0
```

Entropy [my data = entropy (9,5)]:

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



Information Gain $G(X) = H(S) - H(S, X)$

Certainty gain on attribute X

Entropy before split

Entropy after split on attribute X

		Play golf	
		yes	no
Outlook	sunny	3	2
	overcast	4	0
	rainy	2	3
		9	5

entropy(3, 2), 0, entropy(2, 3)

$H(\text{sunny}) = 0.97$
$H(\text{overcast}) = 0$
$H(\text{rainy}) = 0.97$

$$H(S, \text{outlook}) = P(\text{sunny}) \cdot H(\text{sunny}) + P(\text{overcast}) \cdot H(\text{overcast}) + P(\text{rainy}) \cdot H(\text{rainy})$$
$$= \frac{5}{14} \cdot 0.97 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.97 = 0.69$$

Information Gain $G(\text{outlook}) = H(S) - H(S, \text{outlook}) = 0.94 - 0.69 = 0.25$

Potential Splits on X (attributes)

		Play golf	
		yes	no
outlook	sunny	3	2
	overcast	4	0
	rainy	2	3

Info. gain = 0.25

		Play golf	
		yes	no
temperature	hot	2	2
	mild	4	2
	cool	3	1

Info gain = 0.03

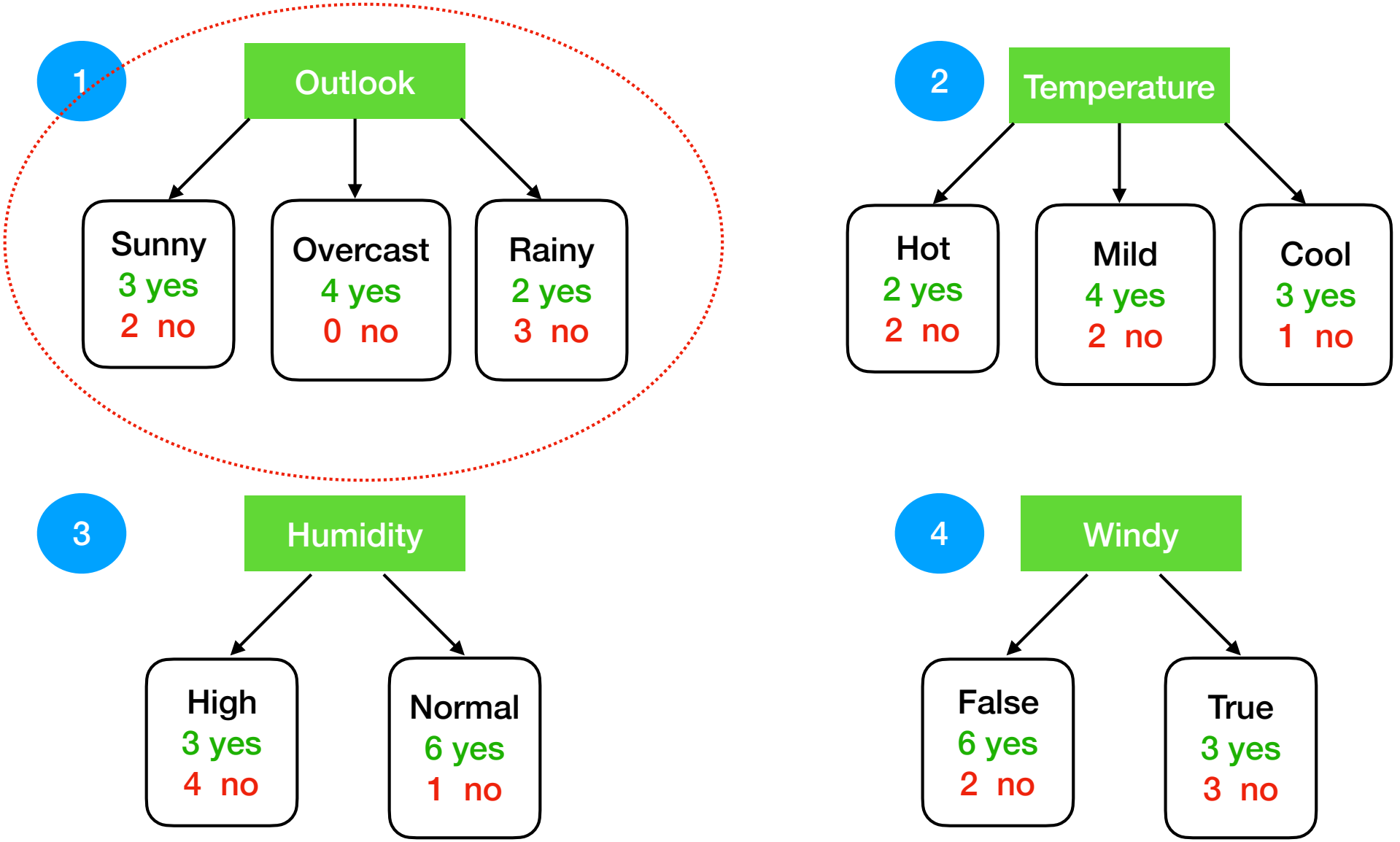
		Play golf	
		yes	no
humidity	high	3	4
	normal	6	1

Info. gain = 0.15

		Play golf	
		yes	no
windy	false	6	2
	true	3	3

Info gain = 0.05

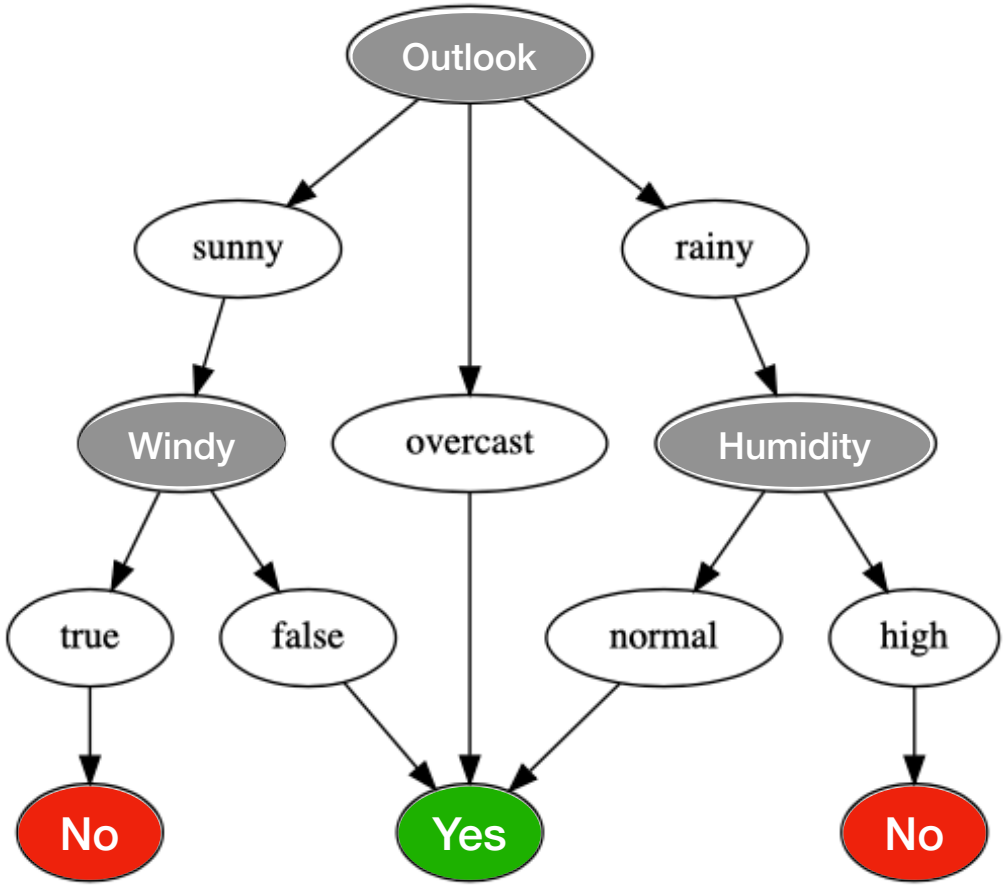
Potential Splits on X (attributes)



Our tree then is:

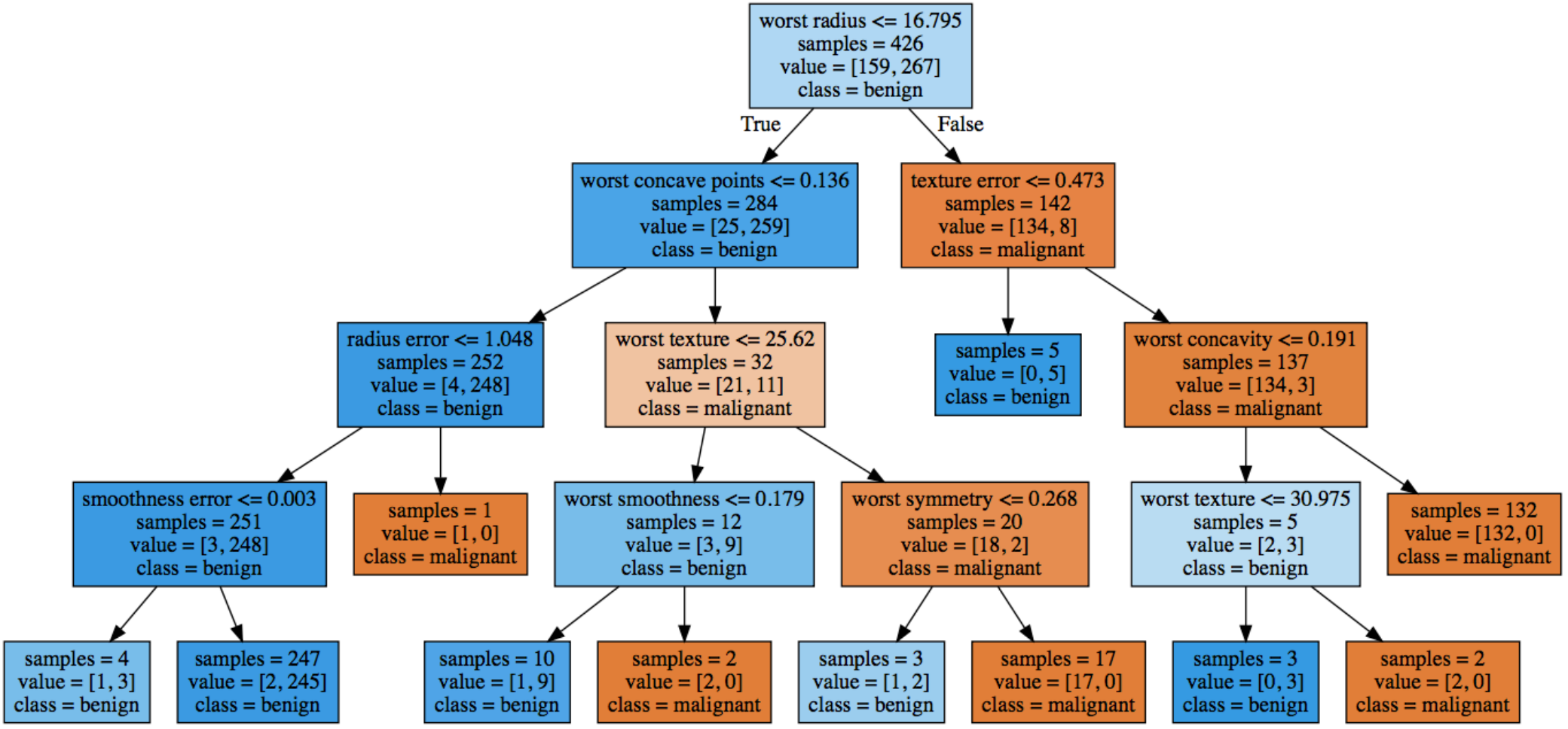
Temp. is out

	Outlook	Temperature	Humidity	Windy	Play Golf
1	Rainy	Hot	High	FALSE	No
2	Rainy	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Sunny	Mild	High	FALSE	Yes
5	Sunny	Cool	Normal	FALSE	Yes
6	Sunny	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Rainy	Mild	High	FALSE	No
9	Rainy	Cool	Normal	FALSE	Yes
10	Sunny	Mild	Normal	FALSE	Yes
11	Rainy	Mild	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
14	Sunny	Mild	High	TRUE	No



Feature Importance

```
>> load_breast_cancer()
```



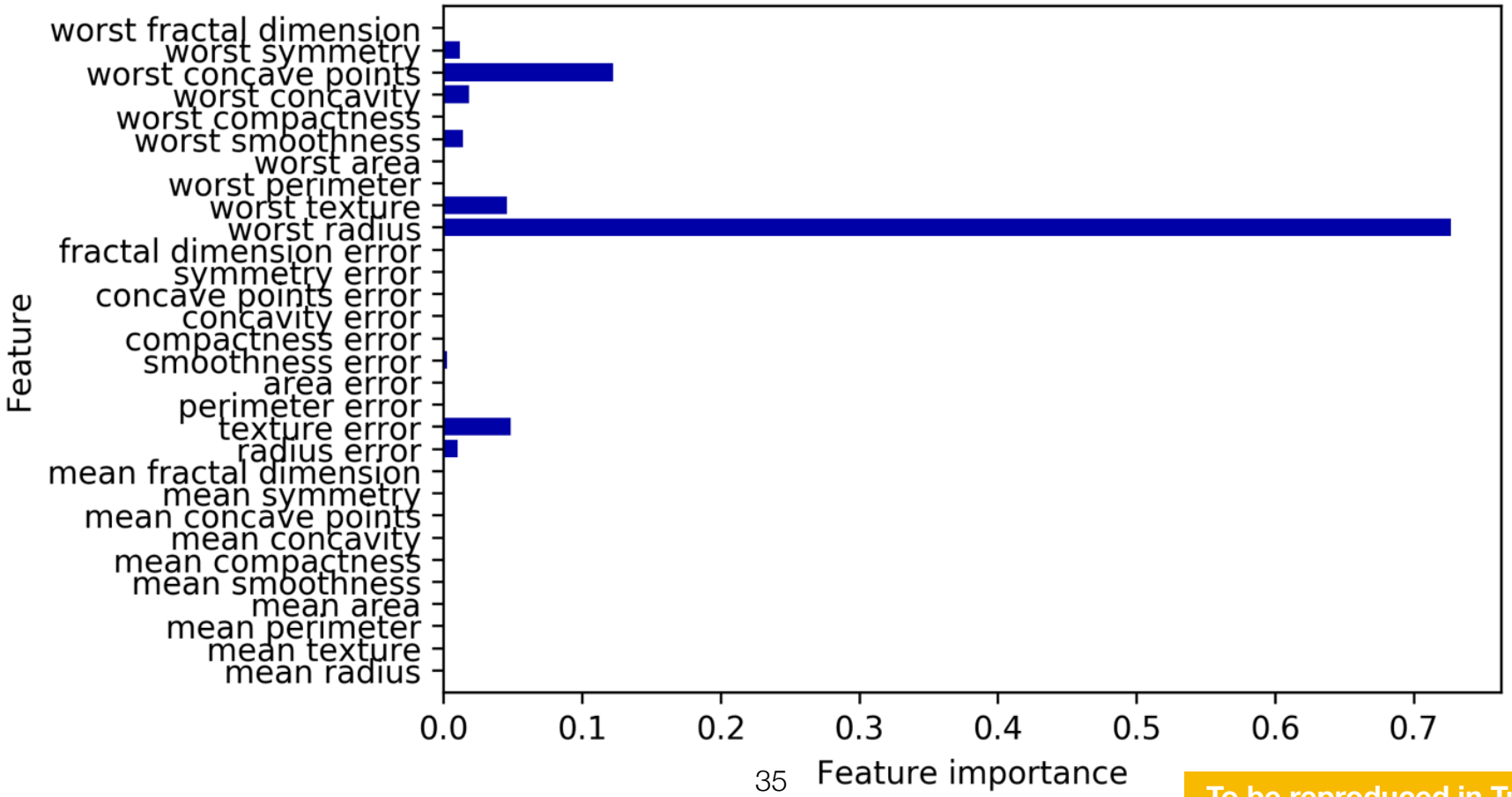
To be reproduced in Tutorial tasks

Feature Importance

```
>> load_breast_cancer()
```

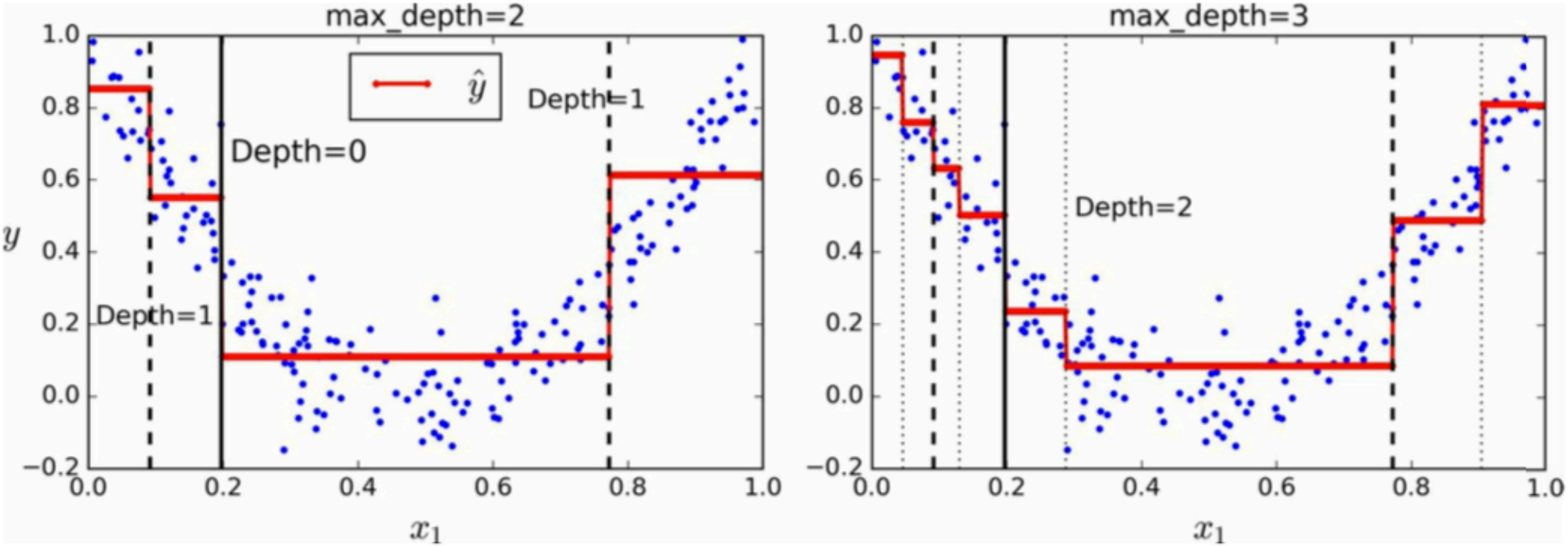
```
print("Feature importances:\n{}".format(tree.feature_importances_))
```

```
Feature importances:
[ 0.     0.     0.     0.     0.     0.     0.     0.     0.     0.     0.01
  0.048  0.     0.     0.002  0.     0.     0.     0.     0.     0.727
  0.046  0.     0.     0.014  0.     0.018  0.122  0.012  0.     ]
```



To be reproduced in Tutorial tasks

DT for Regression



Loss Function $$\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

$$\underset{X}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Figure from: A Geron, Hands on ML - chapter 6

Decision Trees' Summary

➡ very popular:

- Easy to understand (highly **interpretable**)

- Easy to implement
- Easy to use
- Computationally cheap (well!)
- Suitable for linearly (& non linearly) separable classes
- Does regression as well.
- Completely invariant to scaling of the data

But it is horrible!

Today's Puzzle

DT complexity

Decision Tree Depth

Consider a/an (insane) DT where each split (decision node) would peel off one training example, the absolute maximum depth would be ***?***, where *N* is the number of training samples.

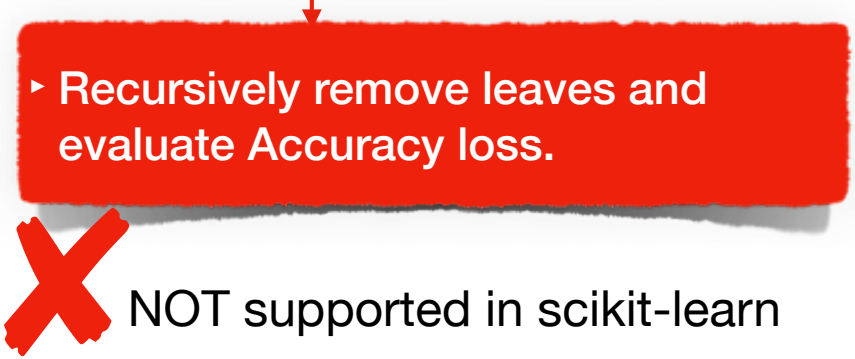
- Training Error = ?
- Generalisation (test) error = ?

Decision Trees Main Issue

OVERFITTING

Pruning - Pre and Post

- A common technique to avoid overfitting in Decision Trees is to prune the tree; either whilst it's being constructed (pre-pruning) or after it's been constructed (post-pruning).



NOT supported in scikit-learn

- Limiting maximum depth of the tree
- Limiting maximum number of leaves
- Requiring a minimum number of samples in a node to allow split

supported in scikit-learn

Decision Trees, review

Pros

- Popular - highly interpretable.
- Model-free (don't assume an underlying distribution).
- Fast (well, super fast!)
- Suitable for both regression and classification problems.

Cons

Prediction "accuracy" isn't that great - inherently high variance

Agenda

- Decision Trees Induction
  - Context
  - Estimation/“design”
  - Properties
- Ensemble Learning
  - Bagging (mainly - I’m hoping to cover more next week)
- Summary
- Tutorial