COMP47750/COMP47990

Decision Trees

Part I
Fundamentals

Pádraig Cunningham
Original slides by Derek Greene

School of Computer Science

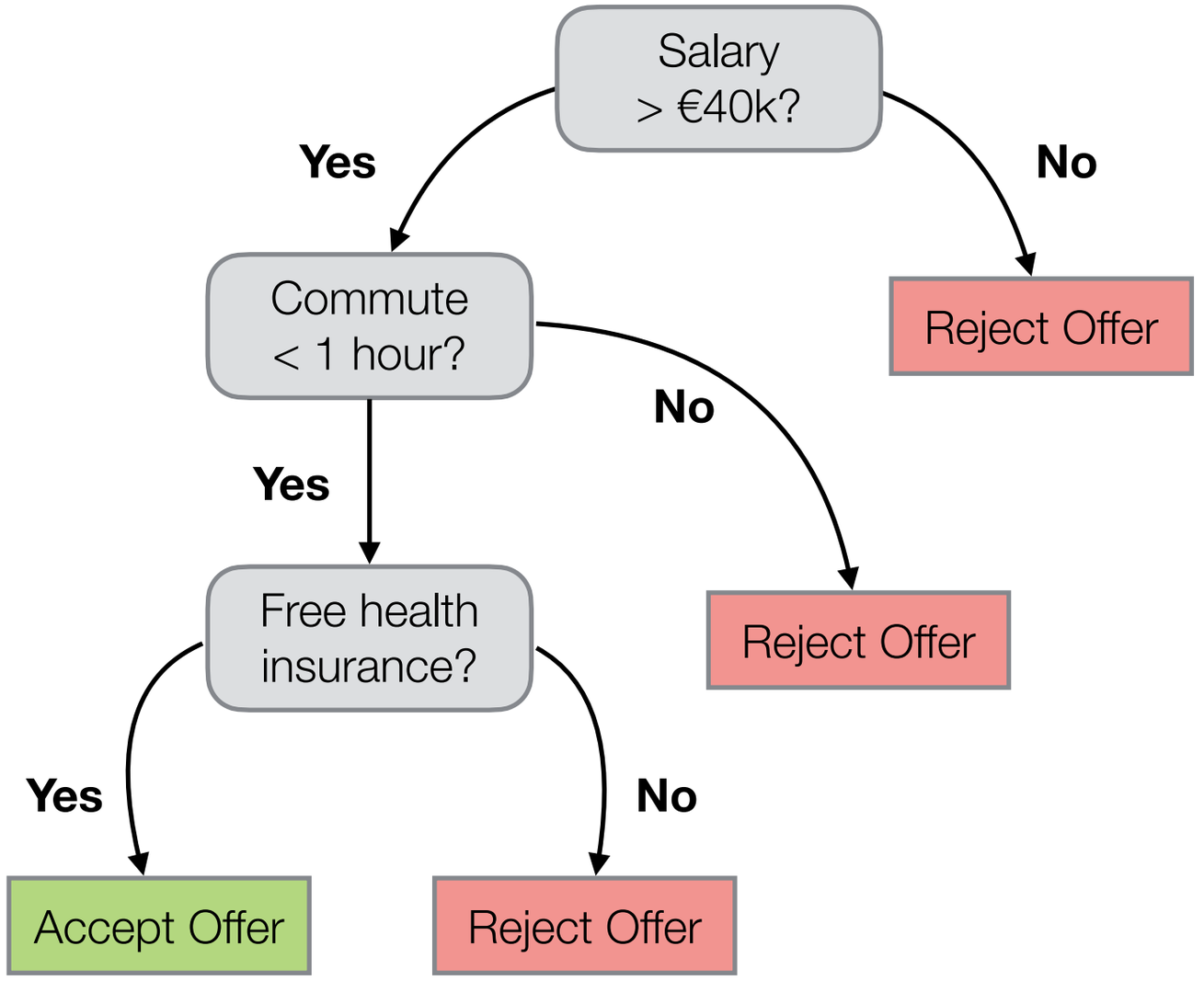
Overview

- What is a Decision Tree?
- Feature Selection
  - Good v Bad features
  - Information Theory - Entropy
- How to build a Decision Tree?
  - ID3 top-down algorithm
  - Information Gain
- Decision Trees in scikit learn

Decision Tree Learning

- Create a classification model to predict class labels by learning decision rules inferred from training data.

Decision:
Should I accept the job offer?



Decision Tree Learning

- **Basic Idea:** Build a tree model that splits the training set into subsets in which all examples have the same class.
- Splitting is done using rules inferred from the training set.
- Each rule is based on a feature, and the split corresponds to the values it can take:
  - e.g. insured = {true, false}
  - e.g. wind = {weak, strong}
  - e.g. income = {low, average, high}
  - e.g. height < 6ft, height ≥ 6ft
- If necessary, each subset can be split again using another feature, and so on until all examples have the same class.
- Once the tree is built, we can use it to quickly classify new input examples - this is an eager learning strategy.

Example: Apples v Pears

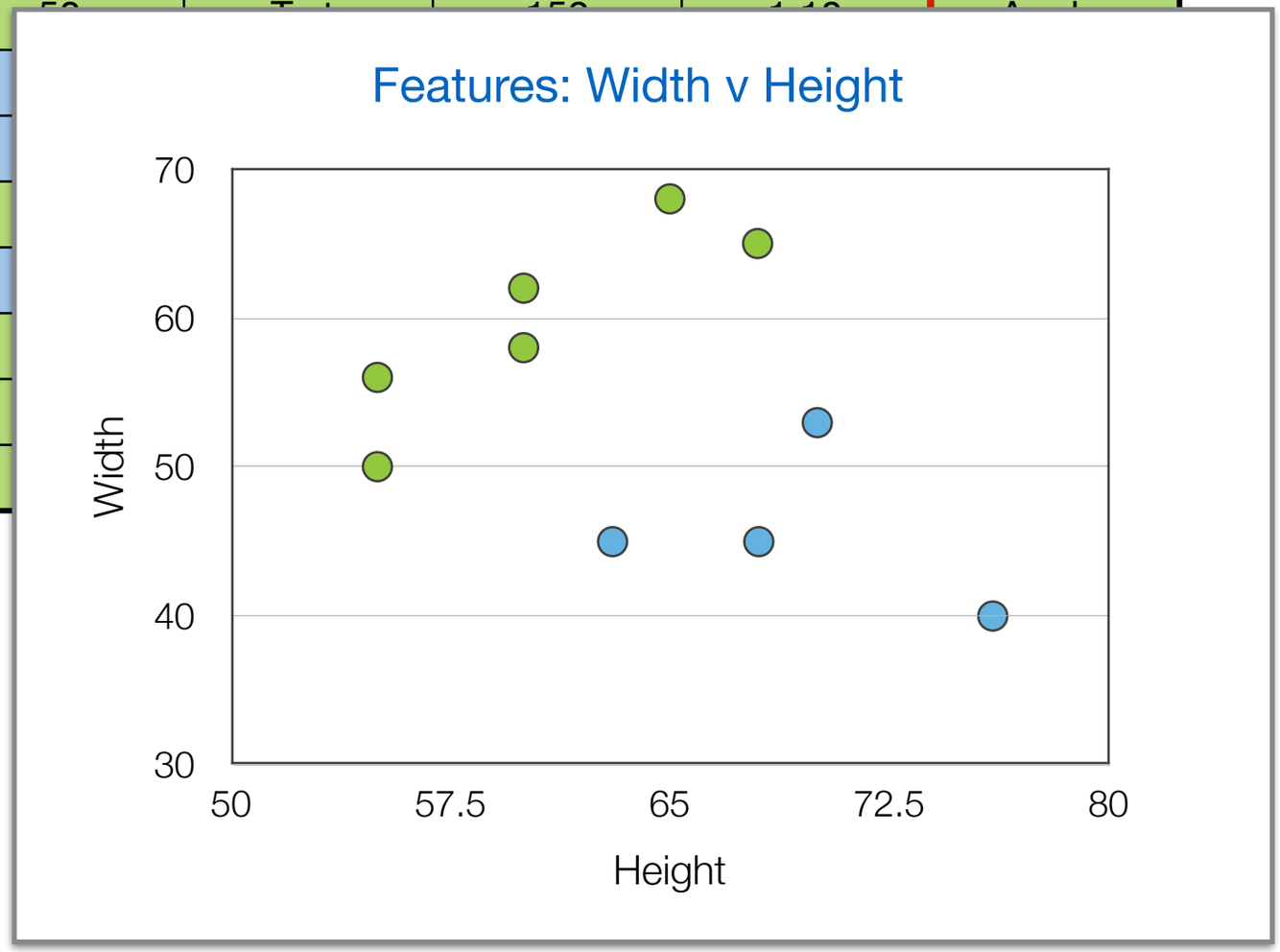
- 10 training examples where each has a class label (“apple” or “pear”), and each is described with 6 features.

Example	*Colour*	*Height*	*Width*	*Taste*	*Weight*	*H/W*	*Class*
1	210	60	62	Sweet	186	0.97	Apple
2	220	70	53	Sweet	180	1.32	Pear
3	215	55	50	Tart	152	1.10	Apple
4	180	76	40	Sweet	152	1.90	Pear
5	220	68	45	Sweet	153	1.51	Pear
6	160	65	68	Sour	221	0.96	Apple
7	215	63	45	Sweet	140	1.40	Pear
8	180	55	56	Sweet	154	0.98	Apple
9	220	68	65	Tart	221	1.05	Apple
10	190	60	58	Sour	175	1.03	Apple

Example: Apples v Pears

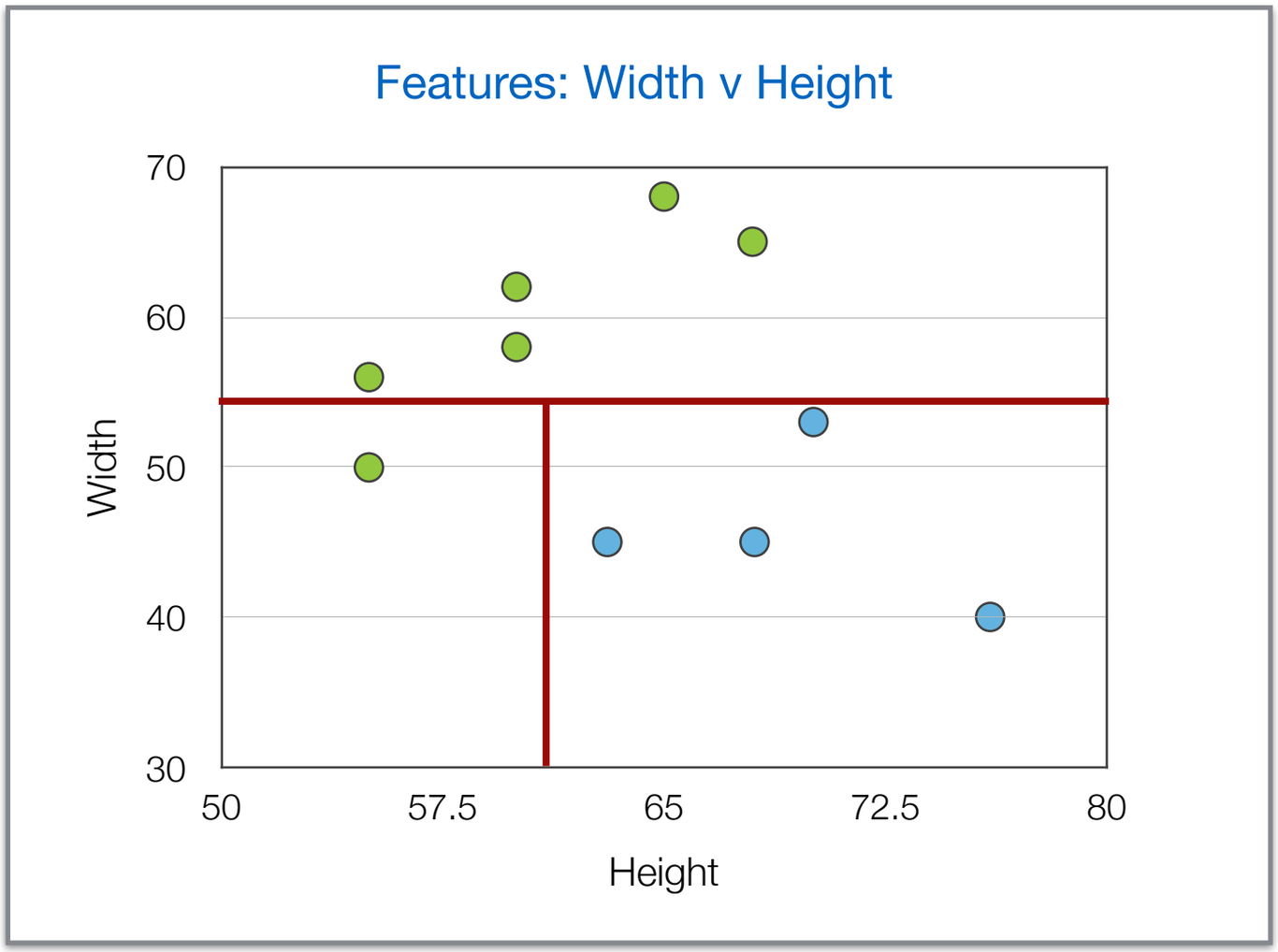
- 10 training examples where each has a class label (“apple” or “pear”), and each is described with 6 features.

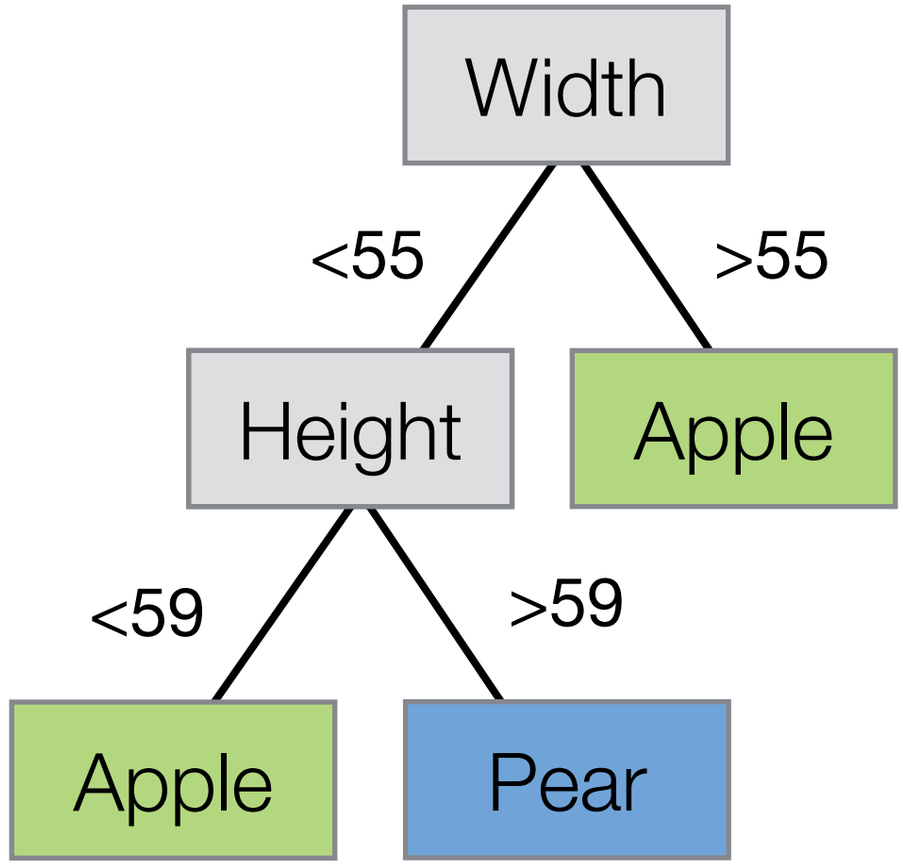
Example	*Colour*	*Height*	*Width*	*Taste*	*Weight*	*H/W*	*Class*
1	210	60	62	Sweet	186	0.97	Apple
2	220	70	53	Sweet	180	1.32	Pear
3	215	55					
4	180	76					
5	220	68					
6	160	65					
7	215	63					
8	180	55					
9	220	68					
10	190	60					



Example: Apples v Pears

- Simple decision tree for classifying Apples v Pears using only 2 features: {Height, Weight}



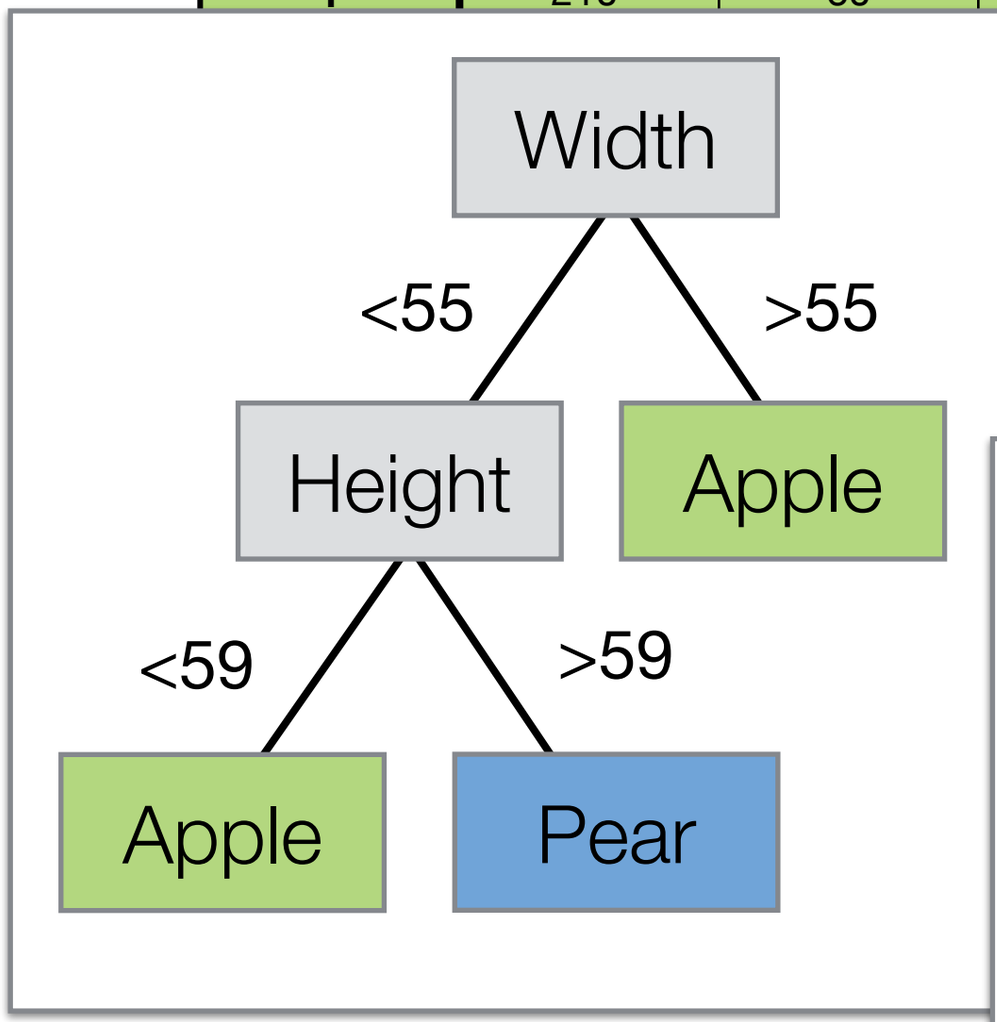


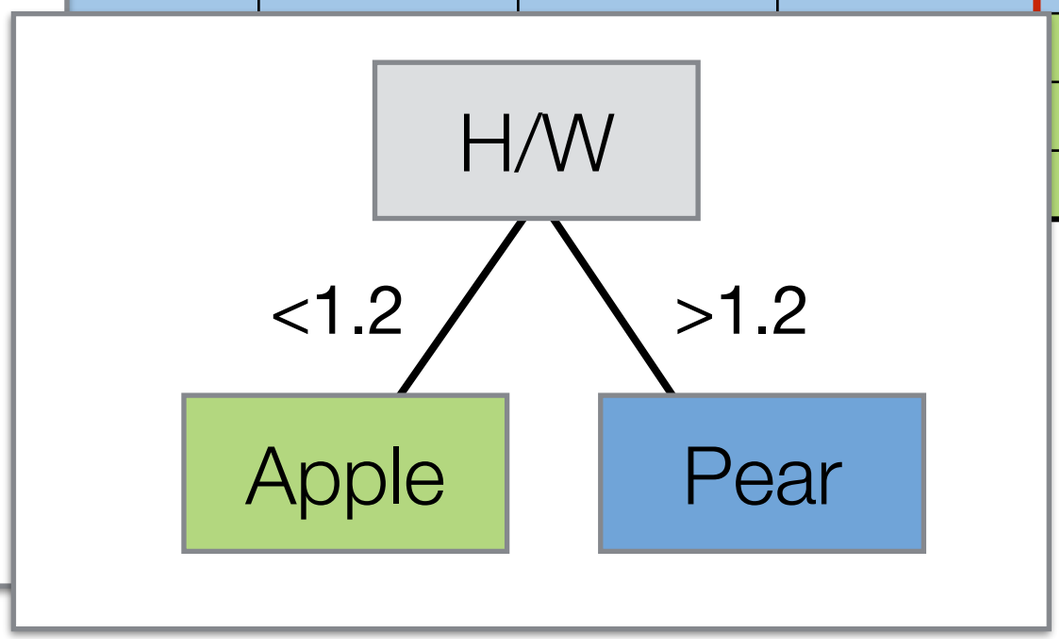
Just 2 features can split the data based on these decision rules.

Example: Apples v Pears

- 10 training examples such that: each has a class label (“apple” or “pear”), and each is described with 6 features.

Example	*Colour*	*Height*	*Width*	*Taste*	*Weight*	*H/W*	*Class*
1	210	60	62	Sweet	186	0.97	Apple
			53	Sweet	180	1.32	Pear
			50	Tart	152	1.10	Apple
			40	Sweet	152	1.90	Pear
			45	Sweet	153	1.51	Pear
			68	Sour	221	0.96	Apple
			45	Sweet	140	1.40	Pear
							Apple
							Apple
							Apple



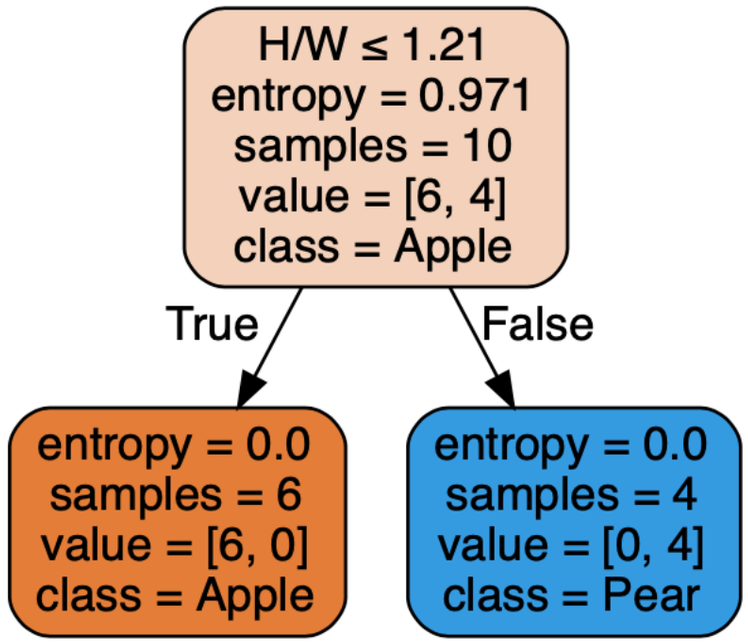


Decision Trees with scikit-learn

Code Notebook:
04 DTrees

- Apples / Pears data

	Greeness	Height	Width	Taste	Weight	H/W	Class
0	210	60	62	Sweet	186	0.97	Apple
1	220	70	53	Sweet	180	1.32	Pear
2	215	55	50	Tart	152	1.10	Apple
3	180	76	40	Sweet	152	1.90	Pear
4	220	68	45	Sweet	153	1.51	Pear




```
from sklearn.tree import DecisionTreeClassifier,
apears = pd.read_csv('ApplesPears.csv')
y = apears.pop('Class').values
apears.pop('Taste')
ap_features = apears.columns
X = apears.values
tree = DecisionTreeClassifier(criterion='entropy')
ap_tree = tree.fit(X, y)
```

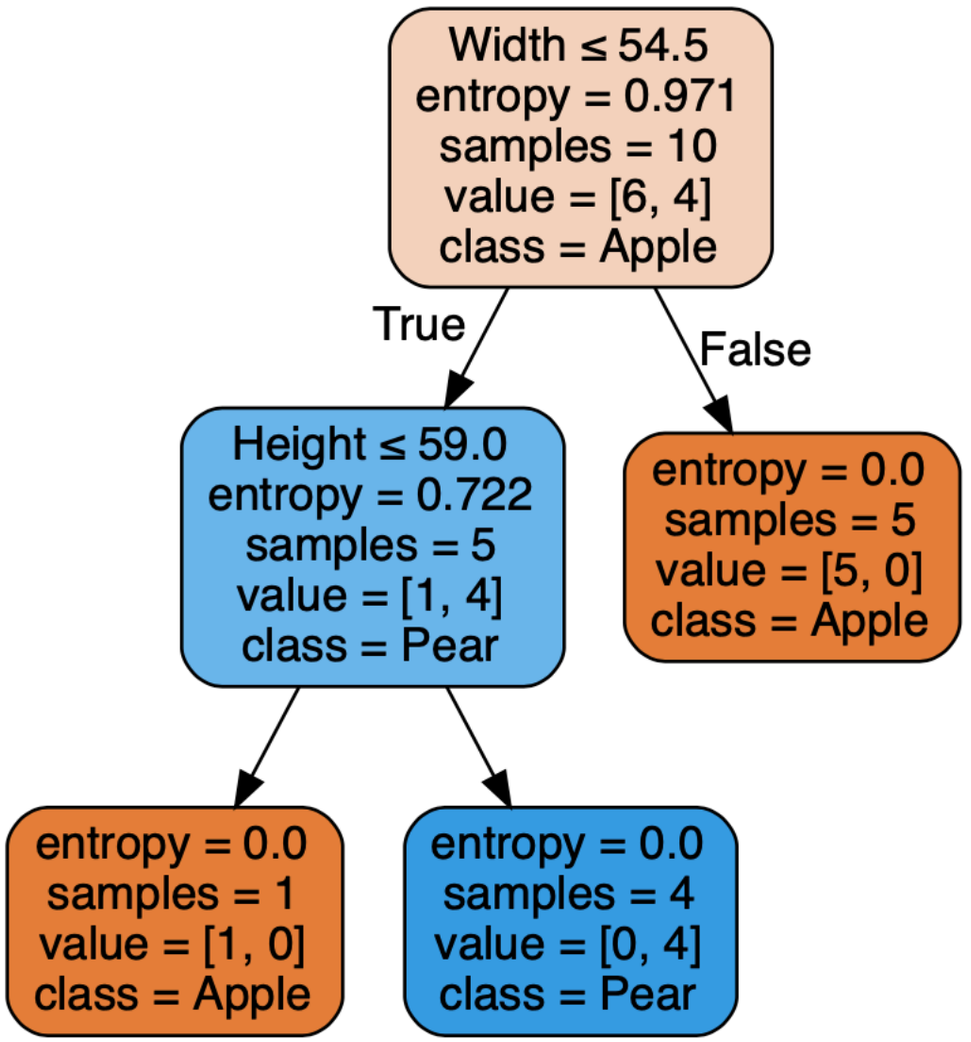
Can't deal with category features

Apple / Pear Decision Tree

- Remove the ‘H/W’ feature (to make it harder)

```
apears.pop('H/W')
ap_features = apears.columns
X = apears.values

ap2_tree = tree.fit(X, y)
```



Aside on guessing games

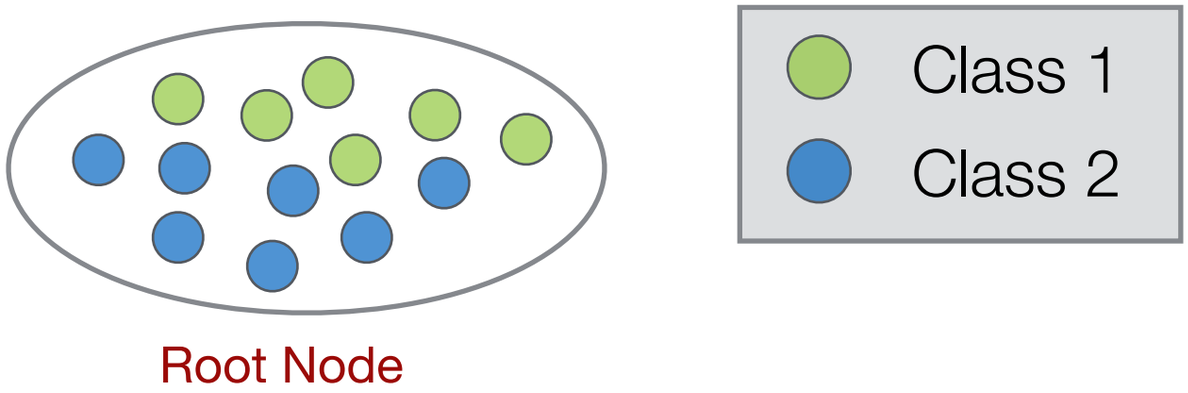
- 20Q

20Q: 10,000 ‘classes’ - ask discriminating questions - balanced tree

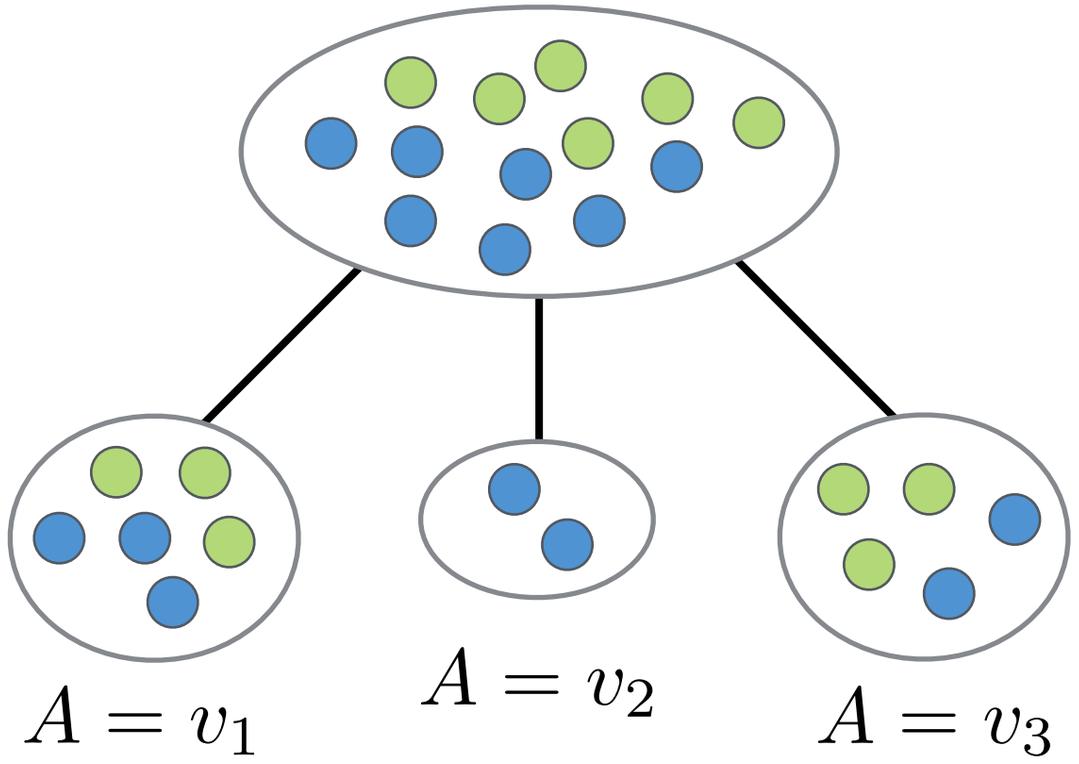
D-Trees: 2 - 10 classes - ask discriminating questions - **organise** samples

Decision Tree Learning

1. Initially all examples in the training set are placed at the root node of the tree.

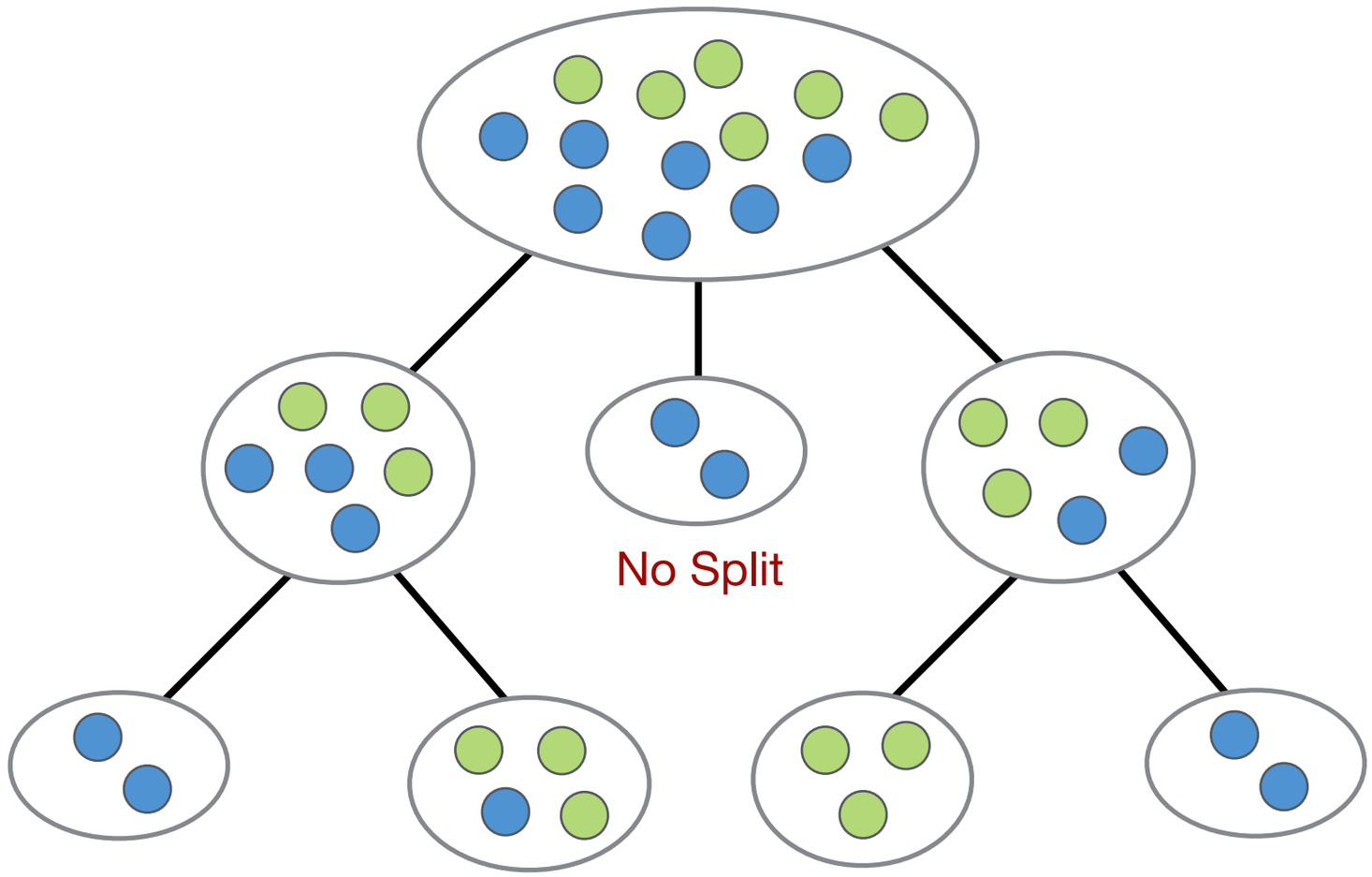


2. One of the available features (A) is now used to split the examples at the root node into two or more child nodes containing subsets of examples.



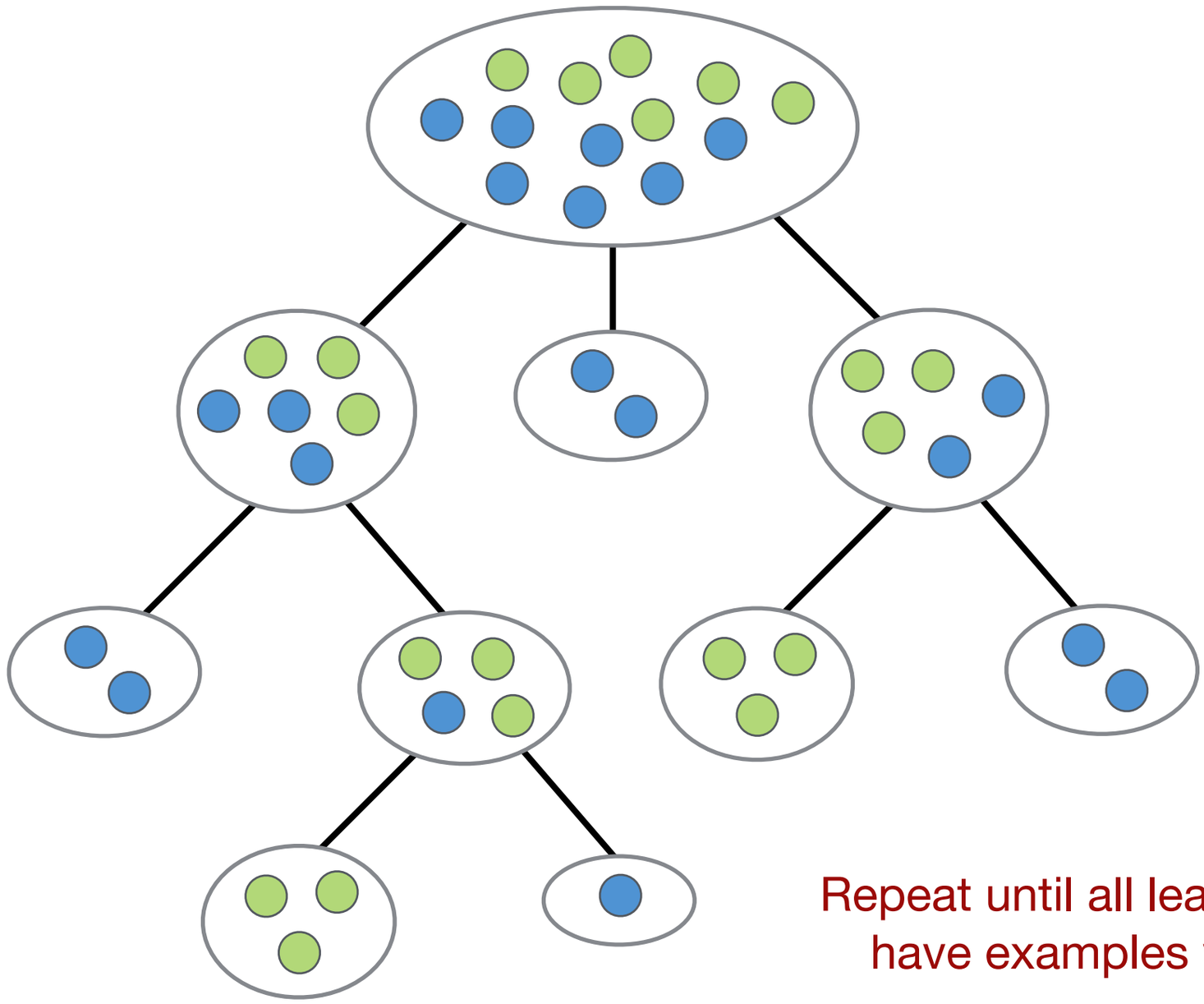
Decision Tree Learning

3. The same process is now applied to each child node, except for any child node at which all examples have the same class.



Decision Tree Learning

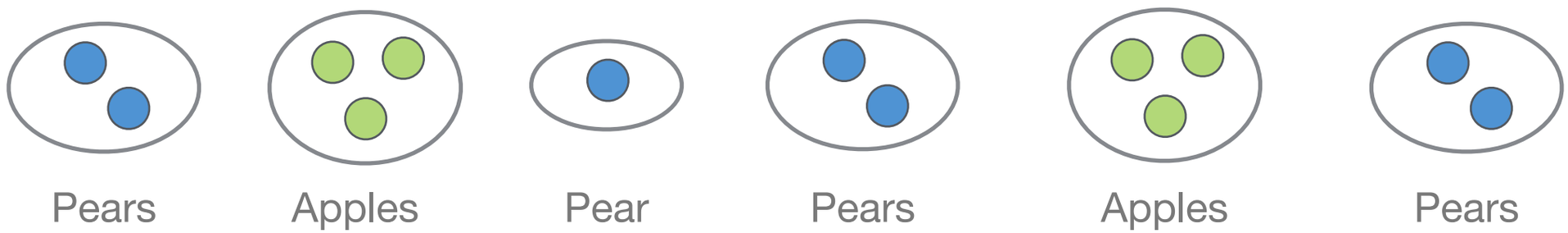
4. This continues until the training set has been divided into subsets in which all the examples have the same class.



Repeat until all leaf nodes in the tree have examples with same class

Node Purity

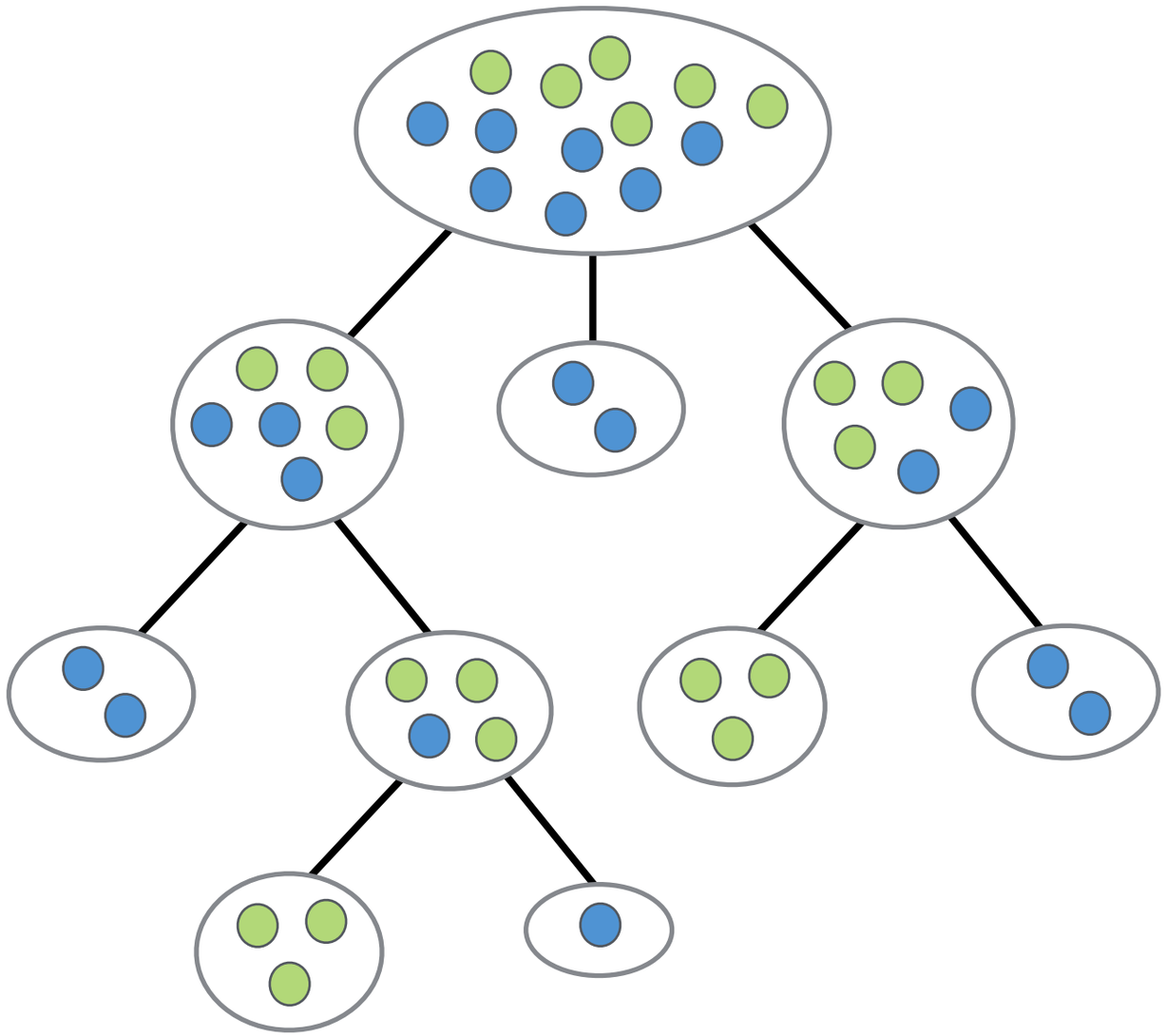
- A tree node is pure if all examples at that node have the same class label.

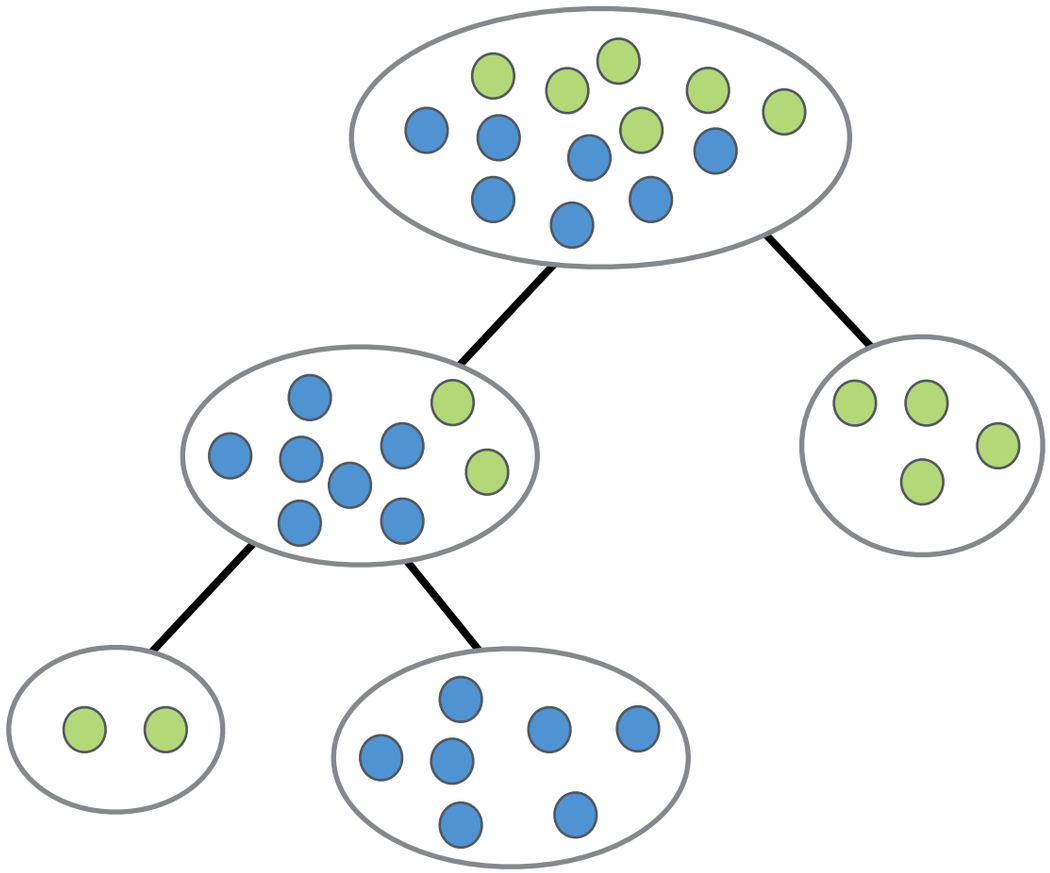


- A decision tree in which all the leaf nodes are pure can always be constructed provided there are no clashes in the data.
  - i.e. examples having the same “description” in terms of features, but with different class labels.
- Most decision tree algorithms use some measure of node (im)purity to choose features to split when building the tree. This measure guides the learning process.

What makes a good tree?

Loads to choose from…





We prefer simpler trees…

Ockham's Razor

Prefer the simplest solution

Prefer the answer the requires the fewest assumptions

Decision Trees Example

Q. "Will a customer wait for a restaurant table?"

Russell & Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

Binary classification task (WillWait = {Yes,No}), with examples described by 10 descriptive features:

Feature	Description
Alternate	Is a suitable alternative restaurant nearby? (Yes/No)
Bar	Does the restaurant have a comfortable bar area to wait in? (Yes/No)
Fri/Sat	True on Fridays and Saturdays, False otherwise.
Hungry	Is the customer hungry? (Yes/No)
Patrons	How many people are in the restaurant: {None, Some, Full}?
Price	Restaurant's price range: {€, €€, €€€}
Raining	Is it raining outside? (Yes/No)
Reservation	Has the customer made a reservation? (Yes/No)
Type	Type of restaurant: {French, Italian, Thai, Burger}
WaitEstimate	Length of wait estimated by the host: {0-10, 10-30, 30-60, > 60 minutes}

Decision Trees Example

Q. "Will a customer wait for a restaurant table?"

Russell & Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

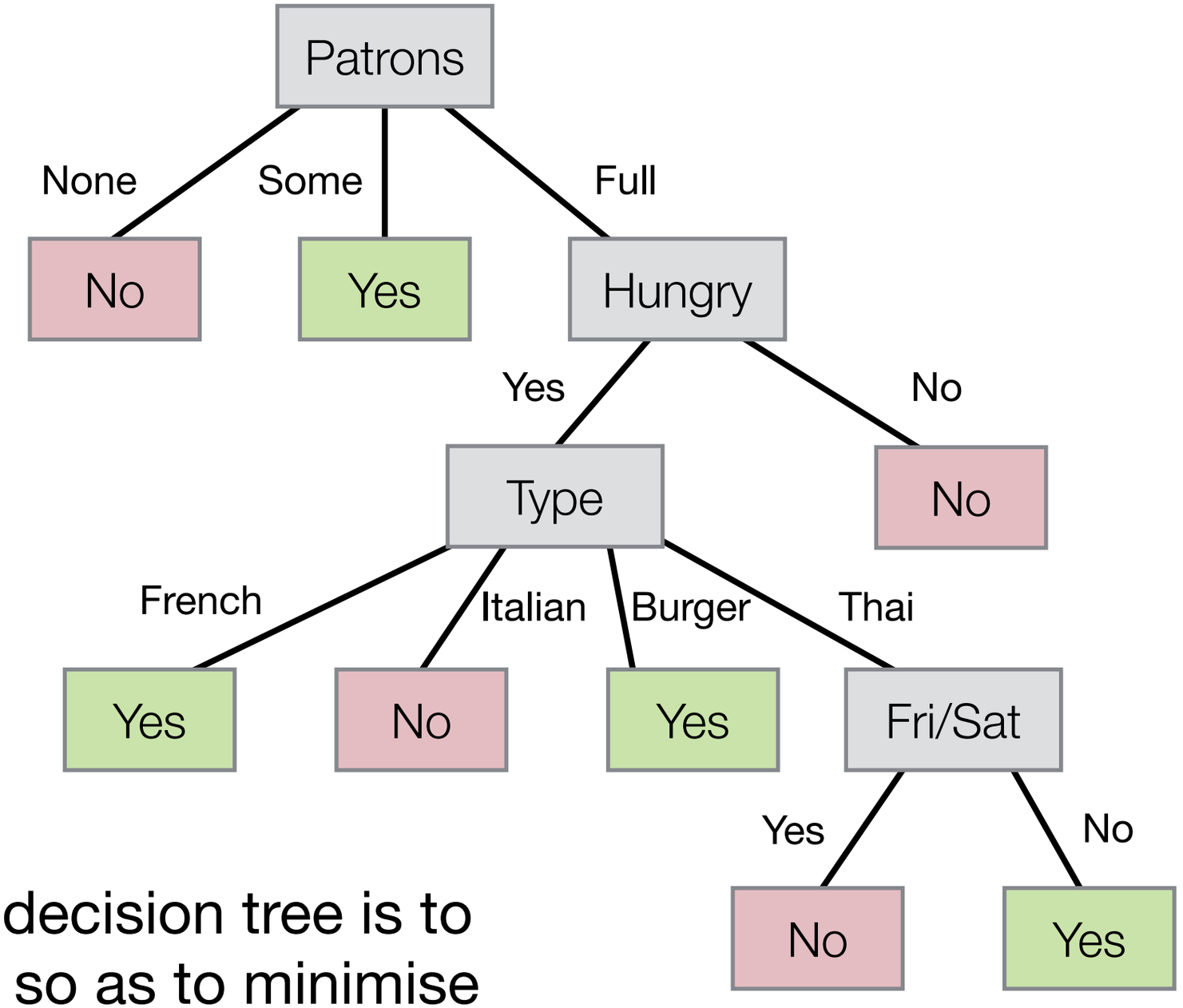
Binary classification task (WillWait = {Yes,No}), with examples described by 10 descriptive features:

Example	Alternate	Bar	Fri/Sat	Hungry	Patrons	Price	Raining	Reservation	Type	WaitEst	WillWait?
1	Yes	No	No	Yes	Some	€€€	No	Yes	French	0-10	Yes
2	Yes	No	No	Yes	Full	€	No	No	Thai	30-60	No
3	No	Yes	No	No	Some	€	No	No	Burger	0-10	Yes
4	Yes	No	Yes	Yes	Full	€	No	No	Thai	10-30	Yes
5	Yes	No	Yes	No	Full	€€€	No	Yes	French	>60	No
6	No	Yes	No	Yes	Some	€€	Yes	Yes	Italian	0-10	Yes
7	No	Yes	No	No	None	€	Yes	No	Burger	0-10	No
8	No	No	No	Yes	Some	€€	Yes	Yes	Thai	0-10	Yes
9	No	Yes	Yes	No	Full	€	Yes	No	Burger	>60	No
10	Yes	Yes	Yes	Yes	Full	€€€	No	Yes	Italian	10-30	No
11	No	No	No	No	None	€	No	No	Thai	0-10	No
12	Yes	Yes	Yes	Yes	Full	€	No	No	Burger	30-60	Yes

➡ How do we build a "good" decision tree for this data set?

Decision Trees - Objective

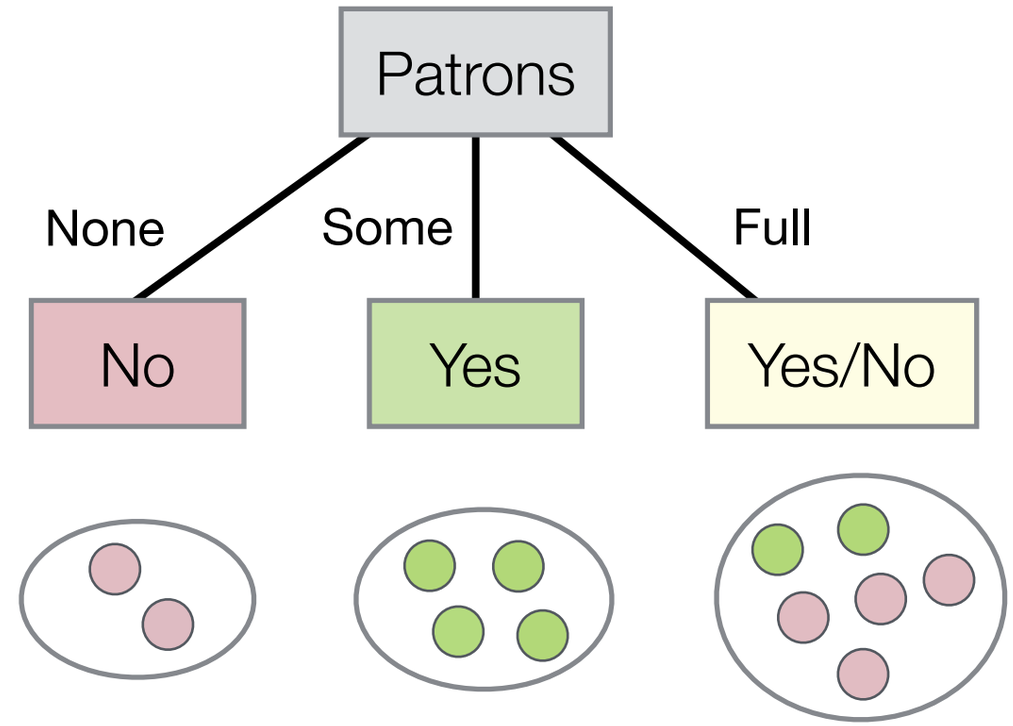
A “good” decision tree will classify all examples correctly using as few tree nodes as possible.



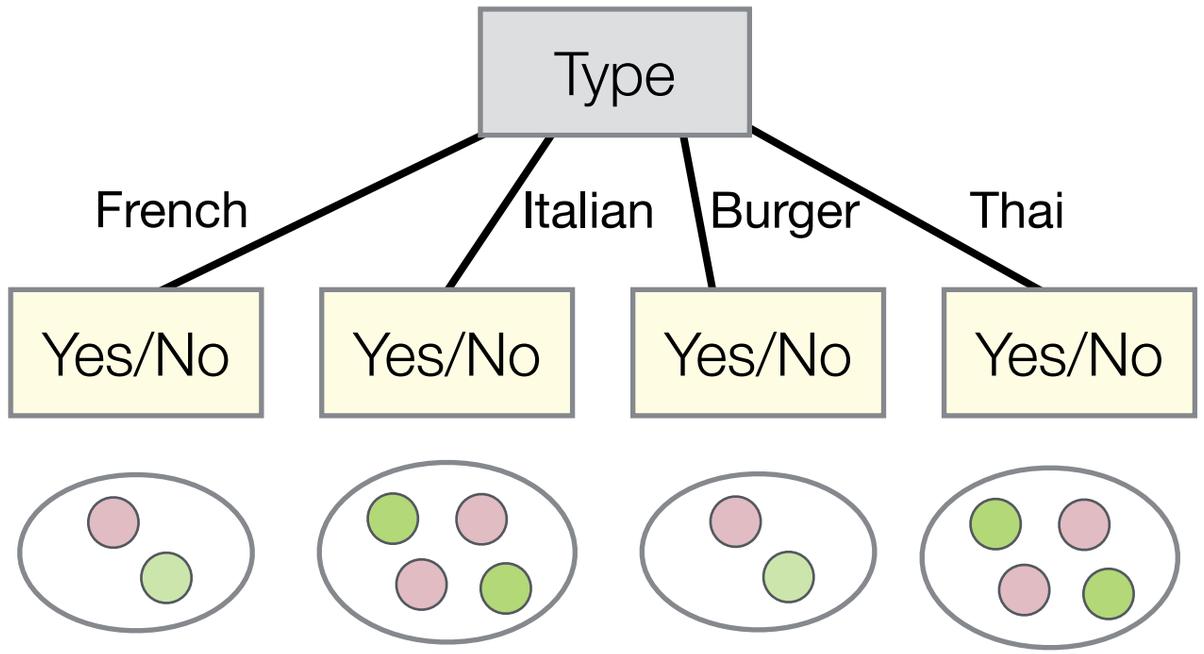
The goal in building a decision tree is to choose good features so as to minimise the depth of the tree.

Good v Bad Features

- Good Features:
  A perfect feature divides examples into categories of one class
  ⇒ high purity



- Bad Features:
  A poor choice of feature produces categories of mixed classes
  ⇒ high impurity



Feature Selection

- Goal: Find good features which divide examples into categories of a single class.
- Feature selection algorithms have been developed which use impurity as an objective to guide feature selection (e.g. Entropy, Gini impurity).

- **Common selection strategy in decision trees:**
  - For each feature, some measure of impurity is applied to the current set of tree nodes.
  - The feature that maximises the reduction in impurity is selected as the next most useful feature.

Coin & Die game

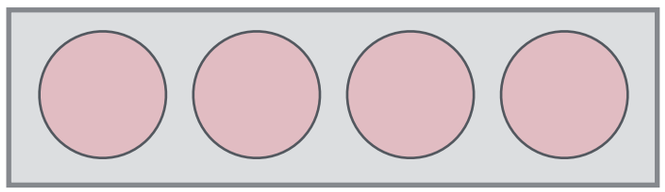
- You win if you throw heads and a six.

- Multiple tries
  - Coin: **W L W W L W L W W L L W L W W L W L** …
  - Die: **L L L L W L L L W L L L L L L W L L** …

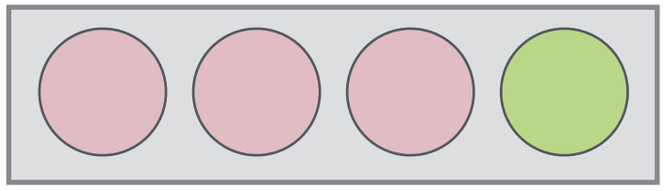
Higher entropy

Entropy

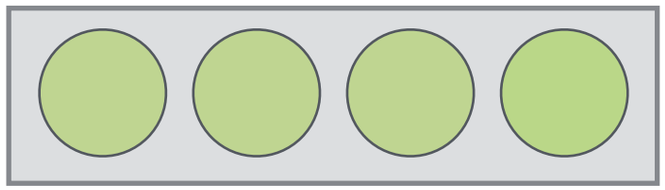
- In information theory, entropy is a measure of the uncertainty around a source of information. Entropy is low for predictable sources, higher for more unpredictable "random" sources.



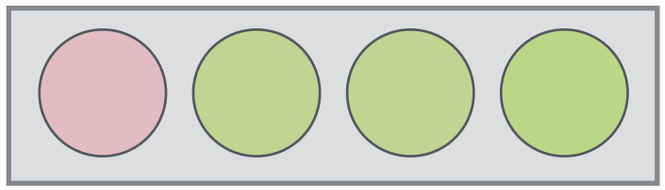
"Definitely No"



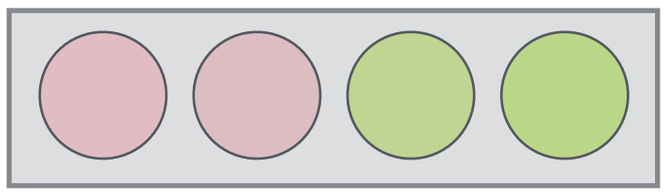
"Probably No"



"Definitely Yes"



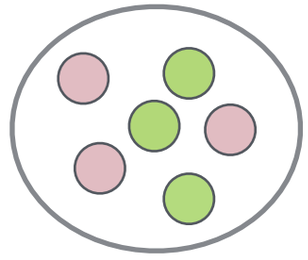
"Probably Yes"



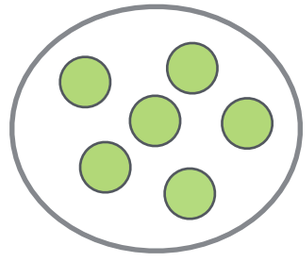
"Completely unsure?"

Entropy

- For decision trees, entropy provides a measure of impurity for each node in a tree - i.e. how uncertain we are about the decision for a given set of examples.



Node has high uncertainty
→ High entropy



Node has low uncertainty
→ Low entropy

- **Calculating entropy:**

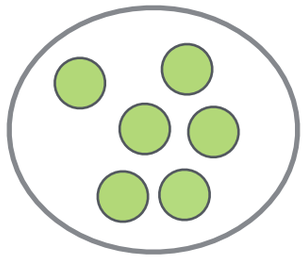
Entropy of a set of examples S with class labels $\{C_1, \ldots, C_n\}$:

$$H(S) = \sum_{j=1}^{n} - p_j log_2(p_j)$$

where p_j is the relative frequency (probability) of class C_j

Entropy Examples

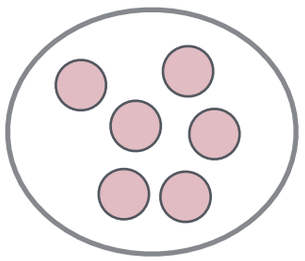
- The lowest possible entropy (i.e. zero) occurs when all examples have the same class label.
- The highest entropy occurs when we are most uncertain.



$$p_1 = 6/6 = 1.0 \qquad p_2 = 0/6 = 0.0$$

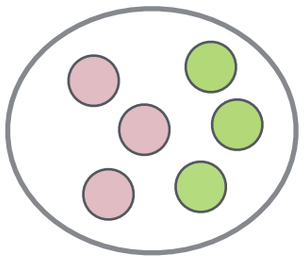
NB: Define $\log_2(0)=0$

$$H(S) = -((1 \times \log_2(1)) + (0 \times \log_2(0))) = -(0 + 0) = 0$$



$$p_1 = 0/6 = 0.0 \qquad p_2 = 6/6 = 1.0$$

$$H(S) = -((0 \times \log_2(0)) + (1 \times \log_2(1))) = -(0 + 0) = 0$$



$$p_1 = 3/6 = 0.5 \qquad p_2 = 3/6 = 0.5$$

$$H(S) = -((0.5 \times \log_2(0.5)) + (0.5 \times \log_2(0.5))) = -(-0.5 - 0.5) = 1$$

COMP47750

Decision Trees

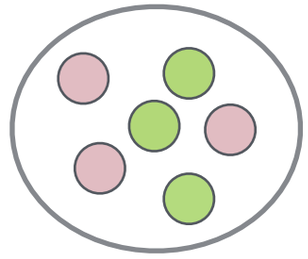
Part II
Algorithms

Pádraig Cunningham
Original slides by Derek Greene

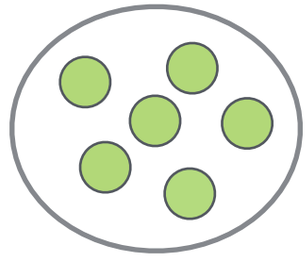
School of Computer Science

Entropy

- For decision trees, entropy provides a measure of impurity for each node in a tree - i.e. how uncertain we are about the decision for a given set of examples.



Node has high uncertainty
→ High entropy



Node has low uncertainty
→ Low entropy

- **Calculating entropy:**

Entropy of a set of examples S with class labels $\{C_1, \ldots, C_n\}$:

$$H(S) = \sum_{j=1}^{n} - p_j log_2(p_j)$$

where p_j is the relative frequency (probability) of class C_j

Top-Down Induction of Decision Trees

- ID3 Algorithm: Popular algorithm which repeatedly builds a decision tree from the top down (Quinlan, 1986).
- Start with an empty tree and set of all training examples *S*.

ID3(S):

- IF all examples in S belong to the same class C THEN
  - Return new leaf node and label it with class C.
- ELSE
  - Select a feature A based on some feature selection criterion.
  - Generate a new tree node with A as the test feature.
  - FOR EACH value v_i of A:
    - Let $S_i \subset S$ contain all examples with $A = v_i$.
    - Build subtree by applying ID3(S_i)

Criterion: Information Gain

- Information Gain (IG): Popular information theoretic approach for selecting features in decision trees, based on entropy.
- Measures a feature's overall impact on entropy when used to split a set of training examples into two or more subsets.
  - How much information do we learn by splitting on the feature?
  - How much is the reduction in entropy?

- **Definition:**

IG for feature A that splits a set of examples S into $\{S_1, \ldots, S_m\}$:

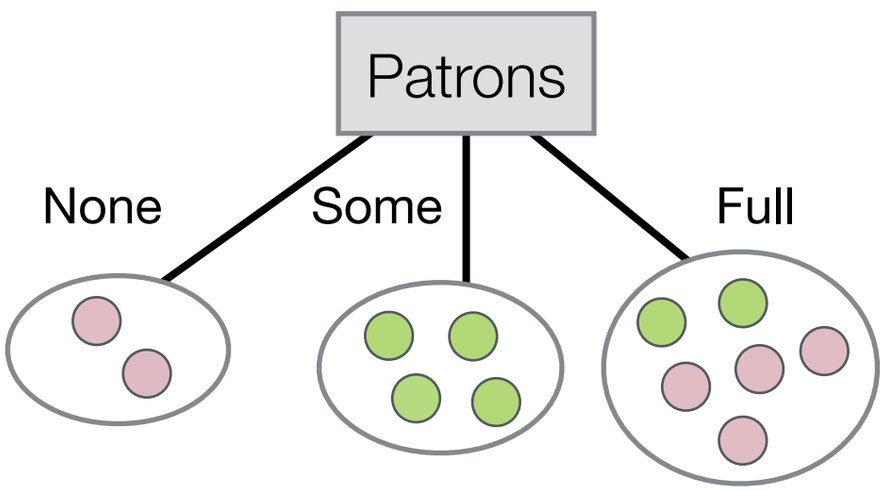
$$IG(S, A) = \text{(original entropy) - (entropy after split)}$$

$$IG(S, A) = H(S) - \sum_{i=1}^{m} \frac{|S_i|}{|S|} H(S_i)$$

Each subset is weighted in proportion to its size

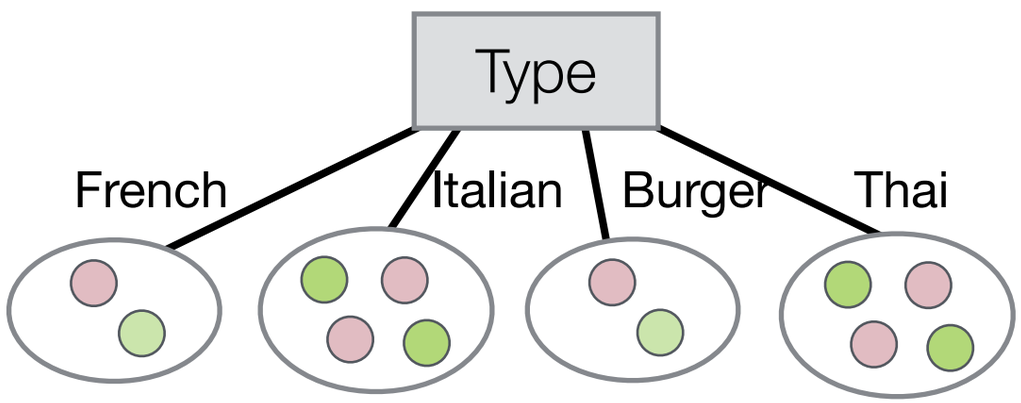
Information Gain Example

- Previous example: Initial training set has 6 Yes, 6 No examples.
- Which feature should we select to split at the root node?



$$IG = H([\frac{6}{12}, \frac{6}{12}]) - \left(\frac{2}{12} H([0,1]) + \frac{4}{12} H([1,0]) + \frac{6}{12} H([\frac{2}{6}, \frac{4}{6}]) \right)$$

$$IG(\text{Patrons}) = 1 - 0.459 = 0.541$$



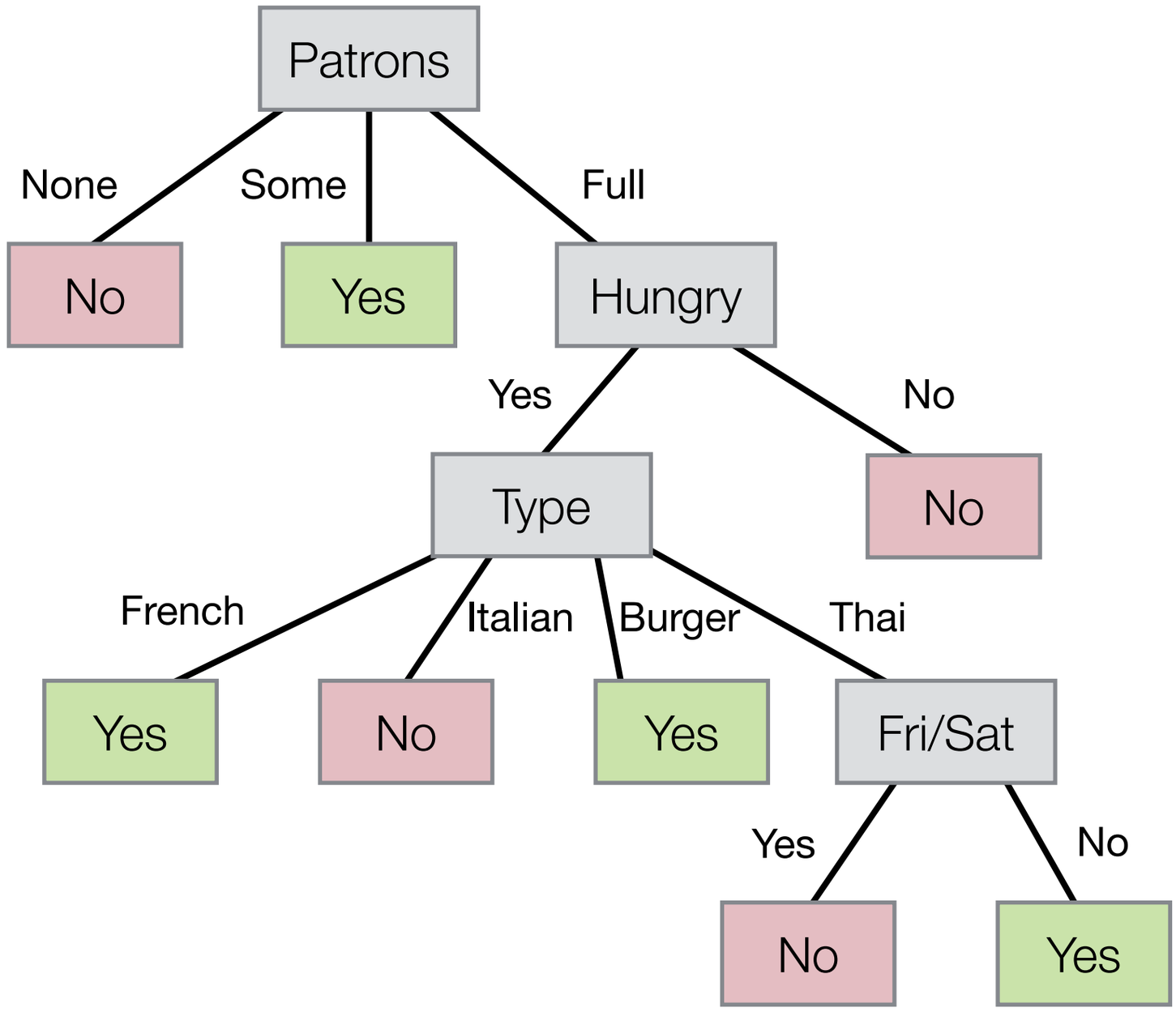
$$IG = H([\frac{6}{12}, \frac{6}{12}]) - \left(\frac{2}{12} H([\frac{1}{2}, \frac{1}{2}]) + \frac{4}{12} H([\frac{2}{4}, \frac{2}{4}]) + \frac{2}{12} H([\frac{1}{2}, \frac{1}{2}]) + \frac{4}{12} H([\frac{2}{4}, \frac{2}{4}]) \right)$$

$$IG(\text{Type}) = 1 - 1 = 0$$

➡ Feature “Patrons” has higher IG, so a better choice for splitting.

Information Gain Example

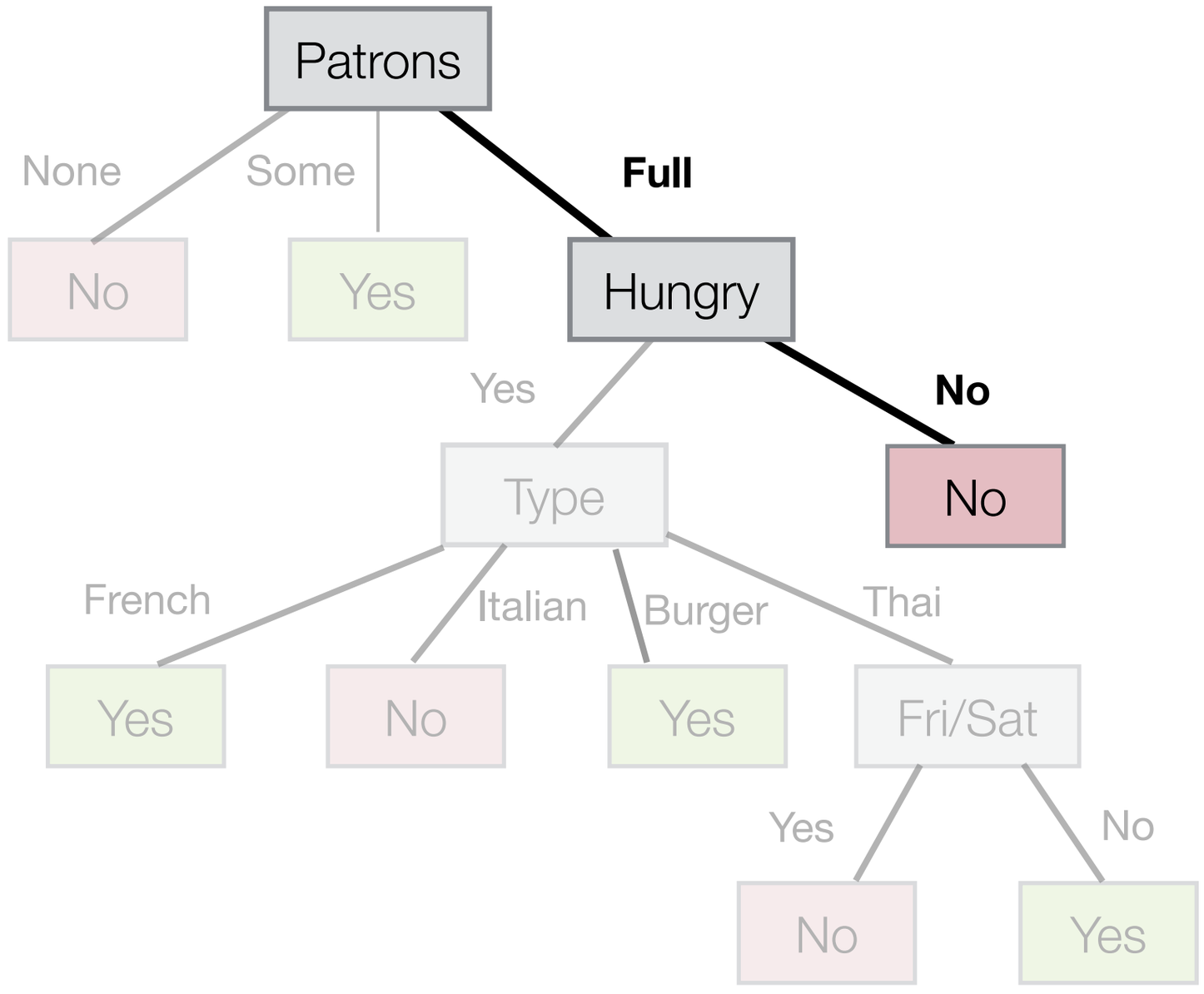
- ID3 repeats the feature selection + splitting process until all examples have the same class, or no features are left to split.



Classifying New Inputs

- Once we have constructed a suitable tree from the training set, we can use the rules in the tree to quickly classify new input examples.

Feature	Query *q1*
Alternate	Yes
Bar	No
Fri/Sat	Yes
Hungry	**No**
Patrons	**Full**
Price	€€
Raining	Yes
Reservation	No
Type	French
WaitEstimate	10-30

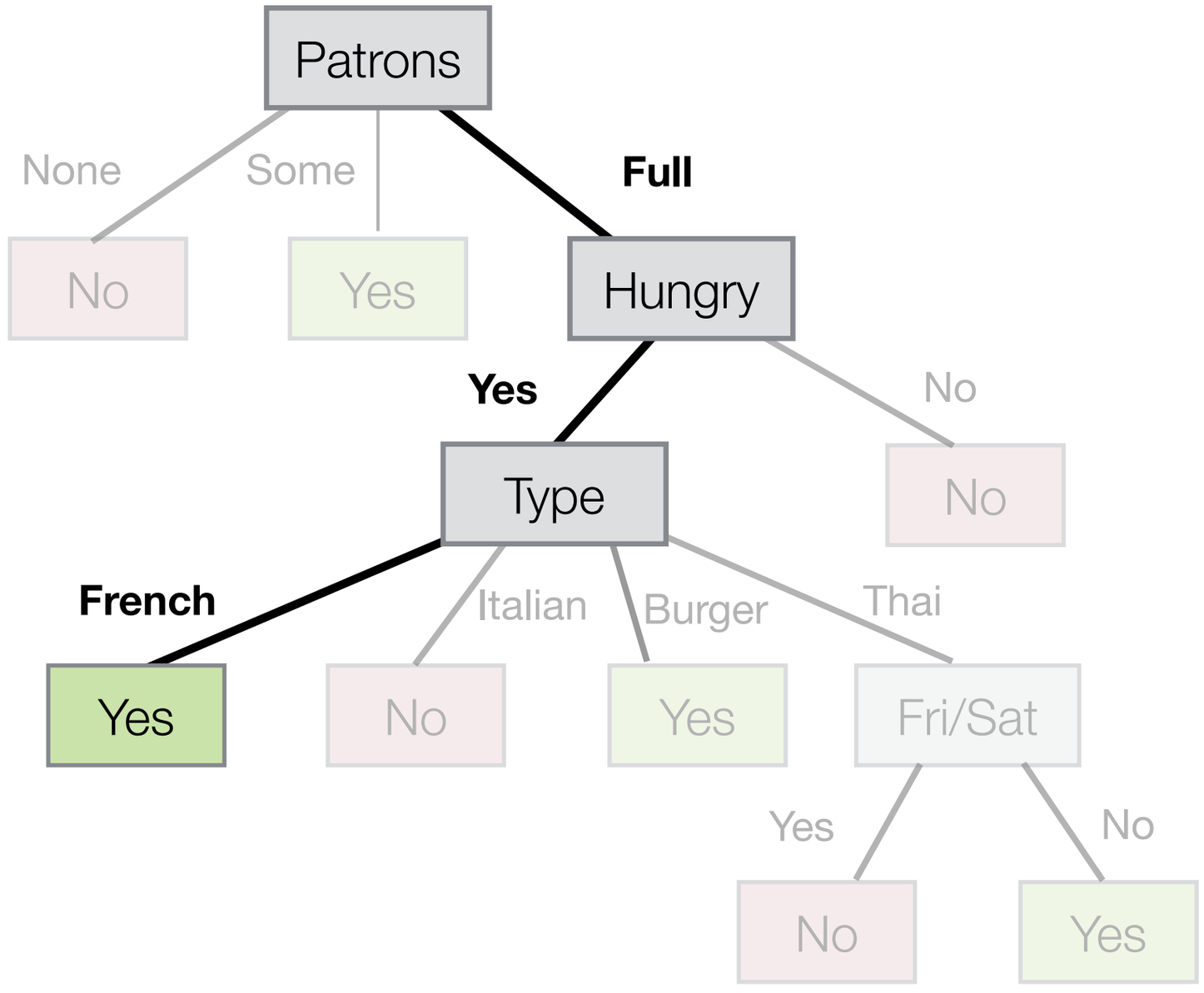


⇒ Based on tree, the decision for *q1* is "No"

Classifying New Inputs

- Once we have constructed a suitable tree from the training set, we can use the rules in the tree to quickly classify new input examples.

Feature	Query *q2*
Alternate	Yes
Bar	No
Fri/Sat	Yes
Hungry	**Yes**
Patrons	**Full**
Price	€€
Raining	Yes
Reservation	No
Type	**French**
WaitEstimate	10-30

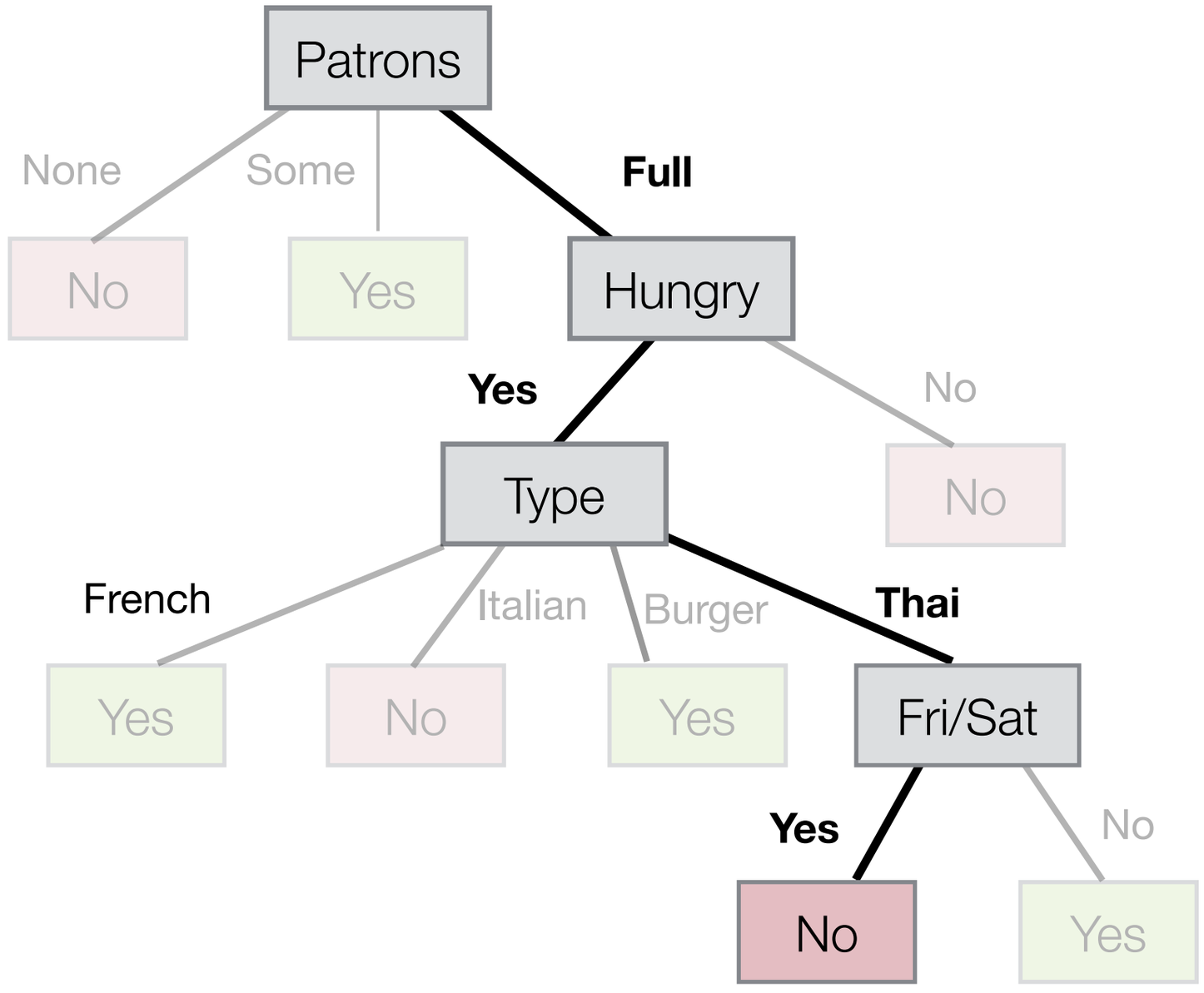


⇒ Based on tree, the decision for *q2* is "Yes"

Classifying New Inputs

- Once we have constructed a suitable tree from the training set, we can use the rules in the tree to quickly classify new input examples.

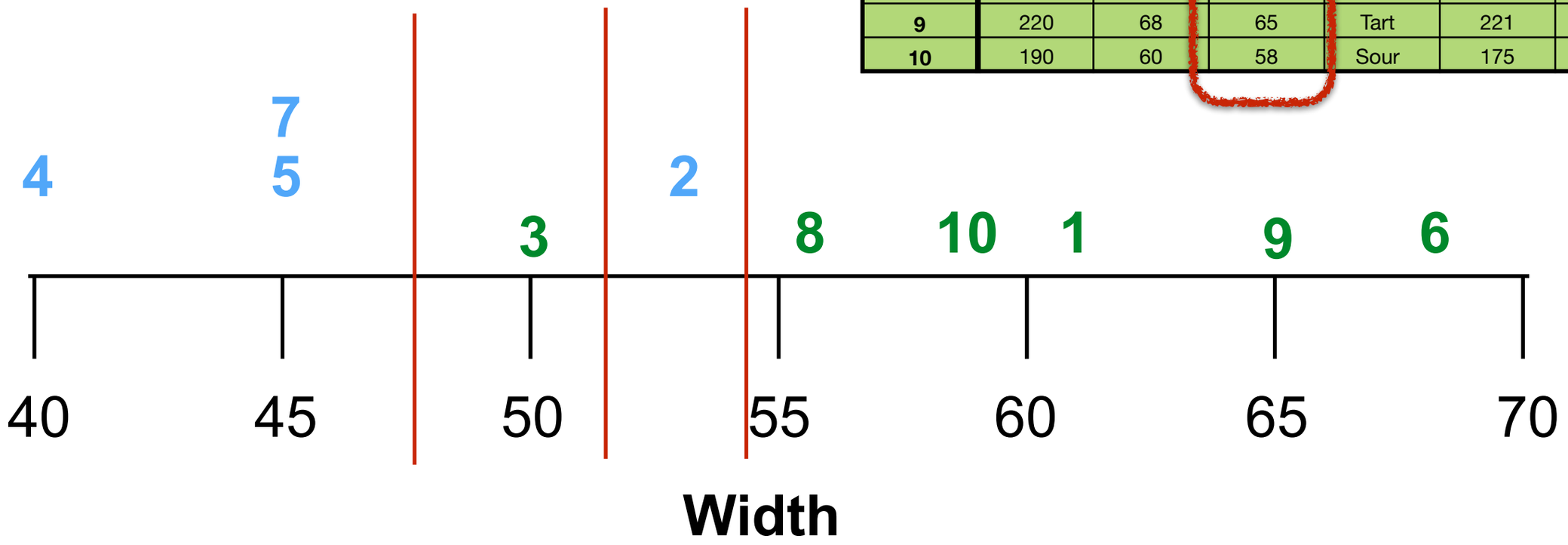
Feature	Query *q3*
Alternate	Yes
Bar	No
Fri/Sat	**Yes**
Hungry	**Yes**
Patrons	**Full**
Price	€€
Raining	Yes
Reservation	No
Type	**Thai**
WaitEstimate	10-30



⇒ Based on tree, the decision for *q3* is "No"

Entropy Numeric Features

Example	Colour	Height	Width	Taste	Weight	H/W	Class
1	210	60	62	Sweet	186	0.97	Apple
2	220	70	53	Sweet	180	1.32	Pear
3	215	55	50	Tart	152	1.10	Apple
4	180	76	40	Sweet	152	1.90	Pear
5	220	68	45	Sweet	153	1.51	Pear
6	160	65	68	Sour	221	0.96	Apple
7	215	63	45	Sweet	140	1.40	Pear
8	180	55	56	Sweet	154	0.98	Apple
9	220	68	65	Tart	221	1.05	Apple
10	190	60	58	Sour	175	1.03	Apple



Candidate split points: points where counts left and right change

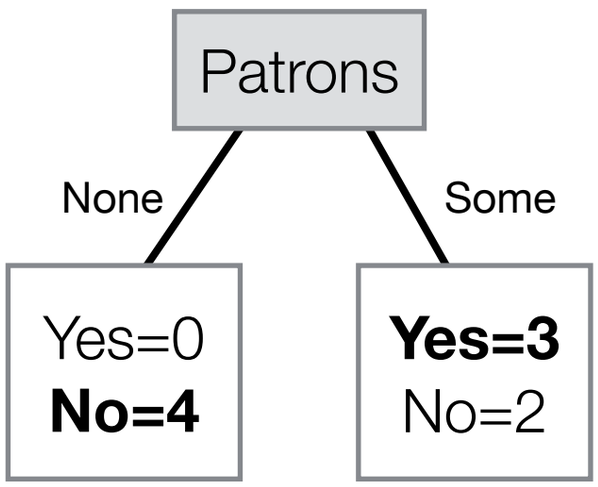
Find candidate split points, calculate I-Gain for all and select the highest

Handling Inconsistent Data

- Inconsistent training data occurs when two identical examples from the training set have different labels:

Example	Alternate	Bar	Fri/Sat	Hungry	Patrons	Price	Raining	Reservation	Type	WaitEst	WillWait?
1	Yes	No	No	Yes	Full	€	Yes	No	Thai	30-60	Yes
2	Yes	No	No	Yes	Full	€	Yes	No	Thai	30-60	No

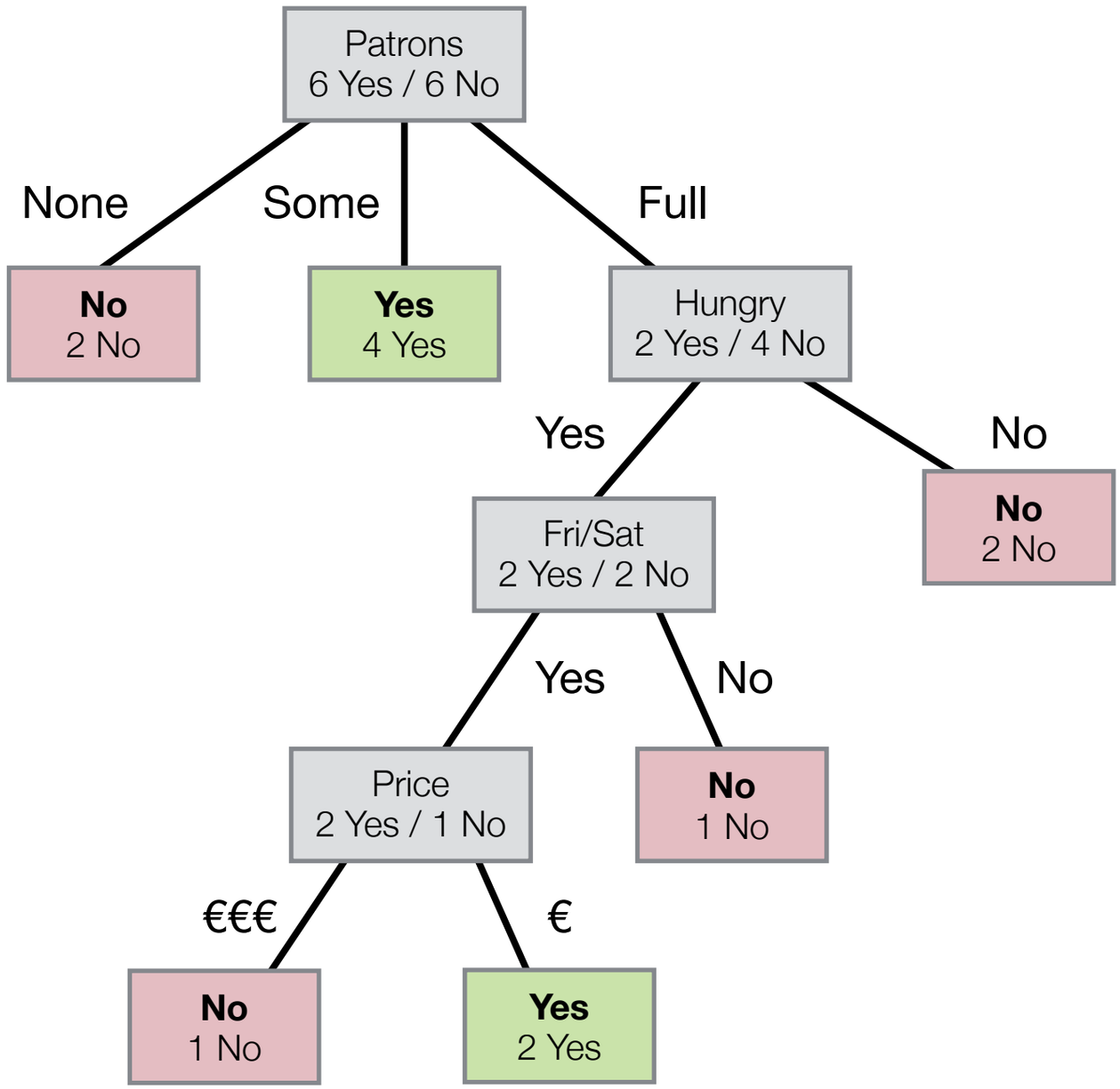
- When building a tree, this can result in cases where leaf nodes are not “pure” and no features are left to split.



- Simple solution:
  - For the label, take the majority vote at the leaf node.
  - If there is a tie, randomly choose one of the class labels.

Managing Overfitting

- A tree that perfectly models the training data
  - May not generalise well…



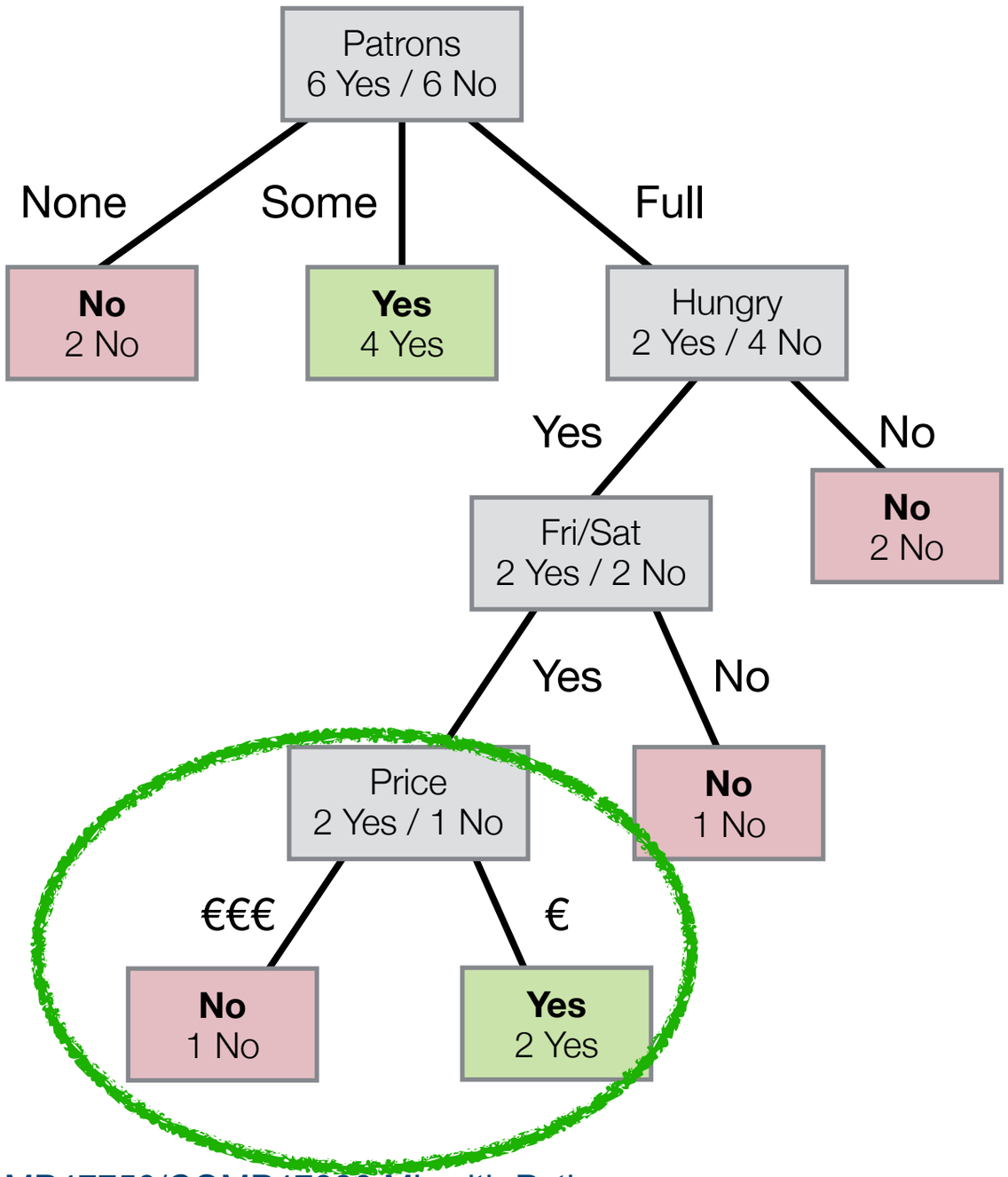
Scikit-learn pruning mechanisms

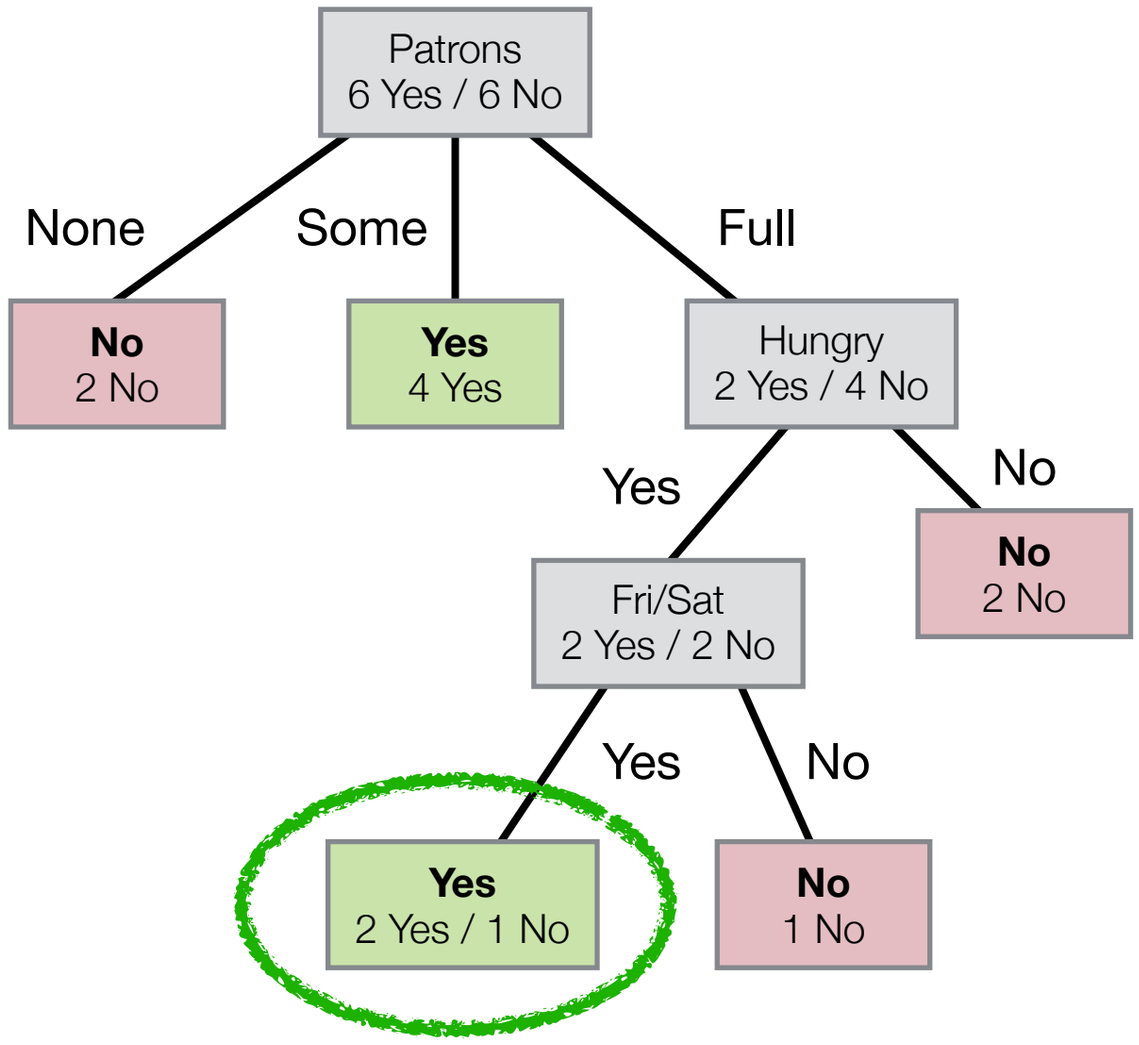
- max_depth
- max_leaf_nodes
- min_samples_leaf
- …

Reduced model capacity

Managing Overfitting

- Set max_depth to 3





Reduced model capacity:
May improve generalisation

scikit-learn - the problem with category data

- Category data must be converted to numbers
  - LabelEncoder does this
    - but not as we would want

```
label_encoder = LabelEncoder()
labelE = df.apply(label_encoder.fit_transform)
labelE
```

	Pet	Transp	Area
0	cat	bike	urban
1	dog	car	urban
2	cat	car	rural
3	ferret	bike	urban

	Pet	Transp	Area
0	0	0	0
1	1	1	0
2	0	1	1
3	2	0	0

OneHot Encoding

- Two options:
  - Use sklearn **`OneHotEncoder`**
    - Creates a transformation object
    - Can be applied to other data
  - Use pandas **`get_dummies`** function

```
df = pd.get_dummies(df,drop_first=False)
```

	Pet	Transp	Area
0	cat	bike	urban
1	dog	car	urban
2	cat	car	rural
3	ferret	bike	urban

	Pet_cat	Pet_dog	Pet_ferret	Transp_bike	Transp_car	Area_urban	Area_rural
0	1	0	0	1	0	1	0
1	0	1	0	0	1	1	0
2	1	0	0	0	1	0	1
3	0	0	1	1	0	1	0

Pandas get_dummies

- A column for each category is not necessary

```
df = pd.get_dummies(df)
```

	Pet_cat	Pet_dog	Pet_ferret	Transport_bike	Transport_car	Area_urban	Area_rural
0	1	0	0	1	0	1	0
1	0	1	0	0	1	1	0
2	1	0	0	0	1	0	1
3	0	0	1	1	0	1	0

```
df = pd.get_dummies(df,drop_first=True)
```

	Pet_dog	Pet_ferret	Transport_car	Area_rural
0	0	0	0	0
1	1	0	1	0
2	0	0	1	1
3	0	1	0	0

One-Hot Encoding

- sklearn OneHotEncoder produces a numpy array

```
from sklearn.preprocessing import OneHotEncoder
onehot_encoder = OneHotEncoder(sparse=False)
dfOH = onehot_encoder.fit_transform(df)
dfOH
Out[14]:
array([[1., 0., 0., 1., 0., 0., 1.],
       [0., 1., 0., 0., 1., 0., 1.],
       [1., 0., 0., 0., 1., 1., 0.],
       [0., 0., 1., 1., 0., 0., 1.]])
In [15]:
onehot_encoder.get_feature_names_out()
Out[15]:
array(['Pet_cat', 'Pet_dog', 'Pet_ferret', 'Transport_bike',
       'Transport_car', 'Area_rural', 'Area_urban'], dtype=object)
```

Important: we have a handle for a OneHotEncoder object that can be applied to other data

Restaurant data

	Alternate	Bar	Fri/Sat	Hungry	Patrons	Price	Raining	Reserv	Type	WaitEst	WillWait?
No											
1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No

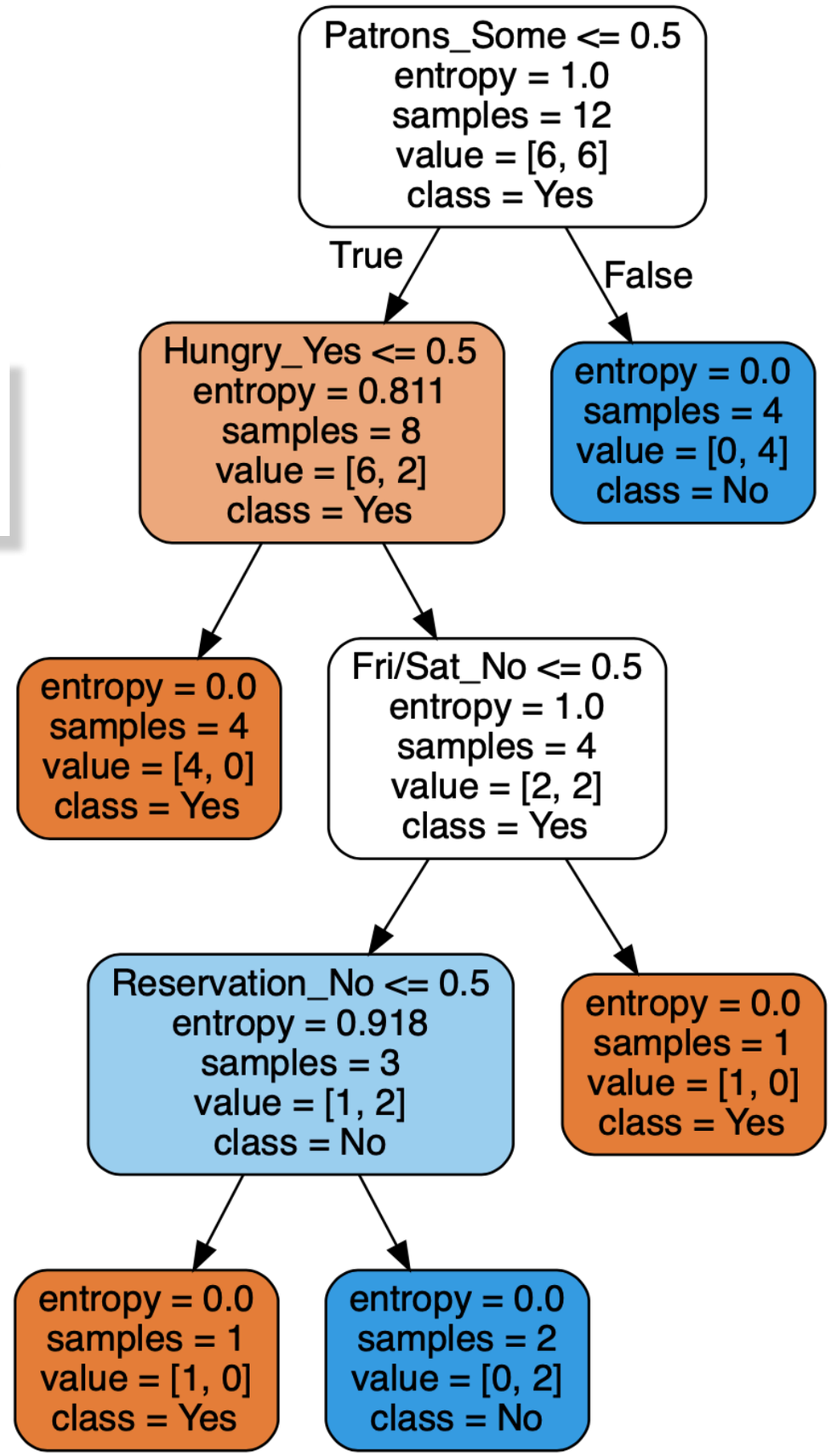
```
onehot_encoder = OneHotEncoder(sparse=False)
restOH = onehot_encoder.fit(restaurant)
restOH_data = restOH.transform(restaurant)

restOH.get_feature_names(restaurant.columns)
Out[21]:
array(['Alternate_No', 'Alternate_Yes', 'Bar_No', 'Bar_Yes', 'Fri/Sat_No',
       'Fri/Sat_Yes', 'Hungry_No', 'Hungry_Yes', 'Patrons_Full',
       'Patrons_None', 'Patrons_Some', 'Price_$', 'Price_$$', 'Price_$$$',
       'Raining_No', 'Raining_Yes', 'Reservation_No', 'Reservation_Yes',
       'Type_Burger', 'Type_French', 'Type_Italian', 'Type_Thai',
       'WaitEst_0-10', 'WaitEst_10-30', 'WaitEst_30-60', 'WaitEst_>60'],
      dtype=object)
```

Restaurant data

```
rtree = DecisionTreeClassifier(
                        criterion='entropy')
rtreeOH = rtree.fit(restOH_data,y)
```

Sadly: these OneHotEncoded trees are really hard to read.



Summary

- A decision tree is an eager learning algorithm where the model is induced from the data in the form of decision rules.
- Many different trees can correctly model same training data.
- We want the simplest tree possible that can generalise to new unseen examples.
- Information Gain helps us select features to use when building simple decision trees.
- The ID3 algorithm for decision trees does not handle numeric data. The extended C4.5 algorithm can handle this type of feature.

Scikit-learn does not handle category data, **one-hot encoding** solves the problem but at the cost of interpretability.

References

- Russell & Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.
- Quinlan, J. R. 1986. “Induction of Decision Trees”. Machine Learning 1, 1 (Mar. 1986), 81-106.
- Mitchell, Tom M. Machine Learning. McGraw-Hill, 1997. pp. 55–58.
- Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.