Decision trees

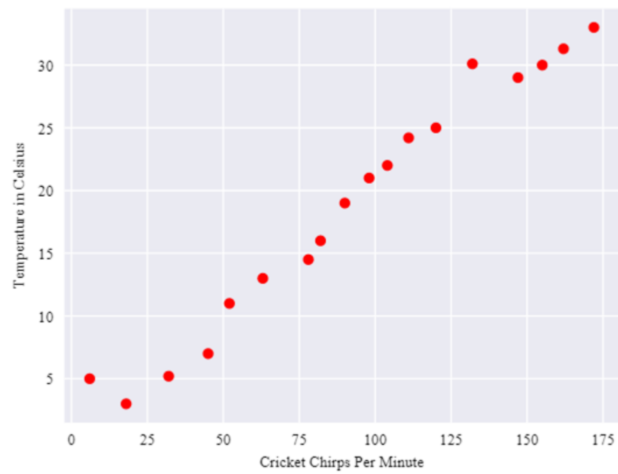
Lecture 02

By Marina Barsky

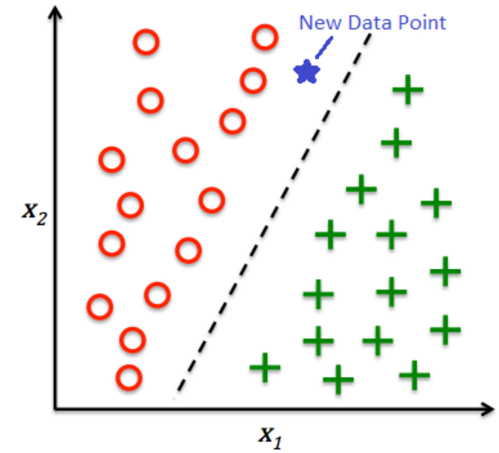
- akinator

Types of learning tasks

Supervised learning

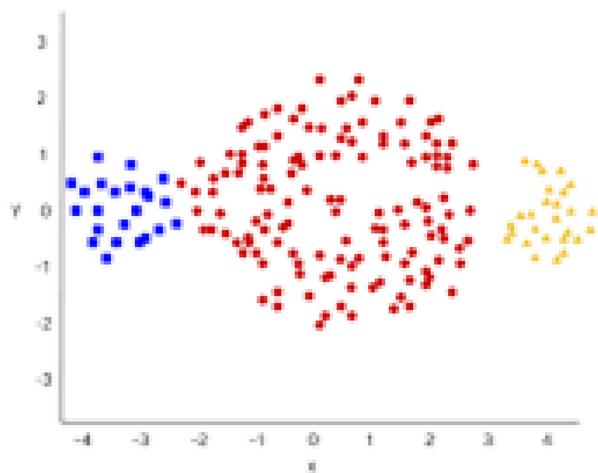


Prediction



Classification

Unsupervised learning



Clustering

TransactionId	Items
1	{A,C,D}
2	{B,C,D}
3	{A,B,C,D}
4	{B,D}
5	{A,B,C,D}

Associations

Decision trees

- Decision support tool
- **Input**: a situation or an object described by a set of attributes
- **Output**: decision

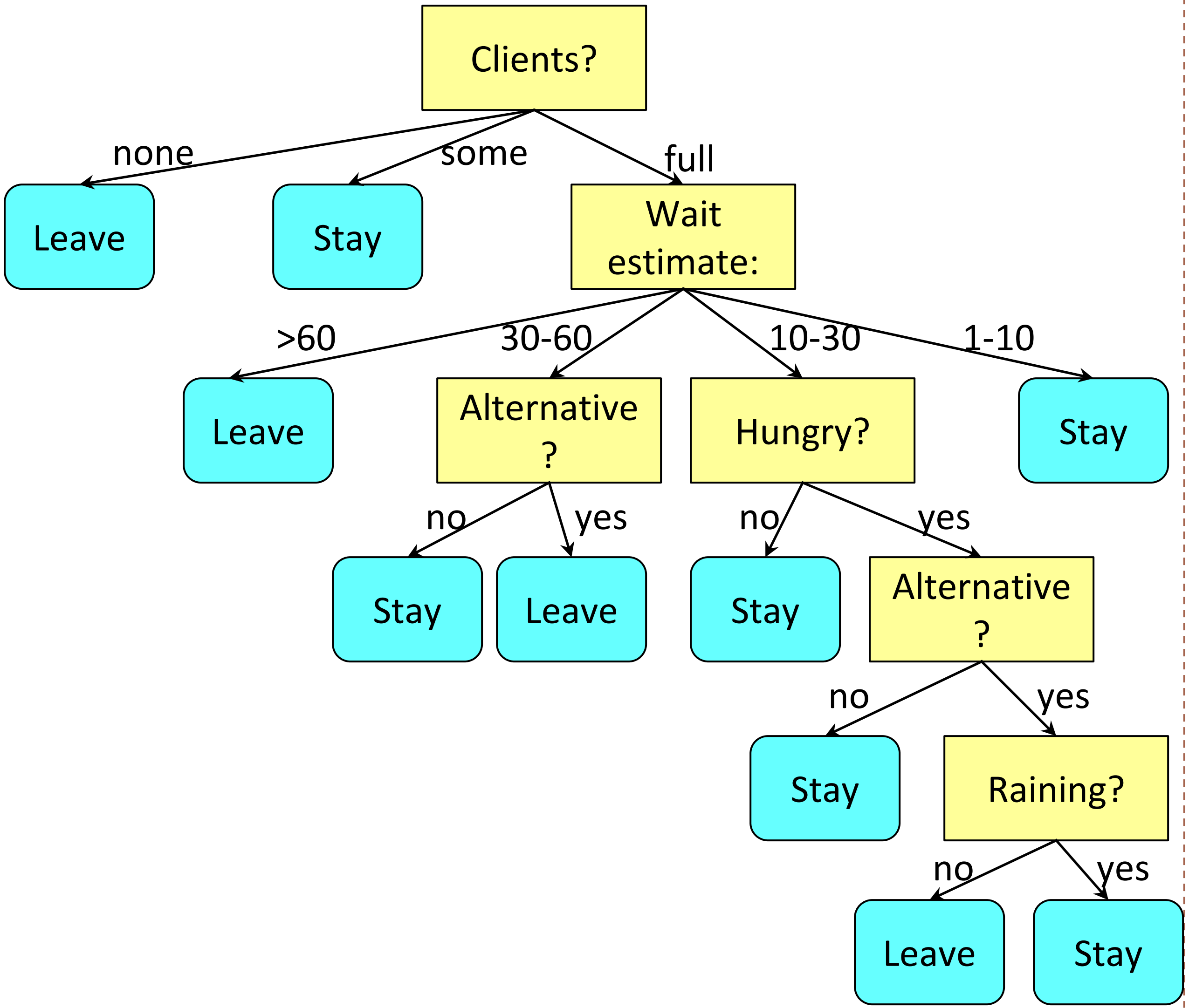
Decision tree – example

- Situation: restaurant
- Question: to leave or to stay?

Decision tree – select features

- Situation: restaurant
- Question: to leave or to stay?
- Set of important attributes:
  - Alternative restaurants: yes, no
  - Am I really hungry?: yes, no
  - Clients?: none, some, full
  - Is it raining?: yes, no
  - Wait estimate: 0-10, 10-30, 30-60, >60 min

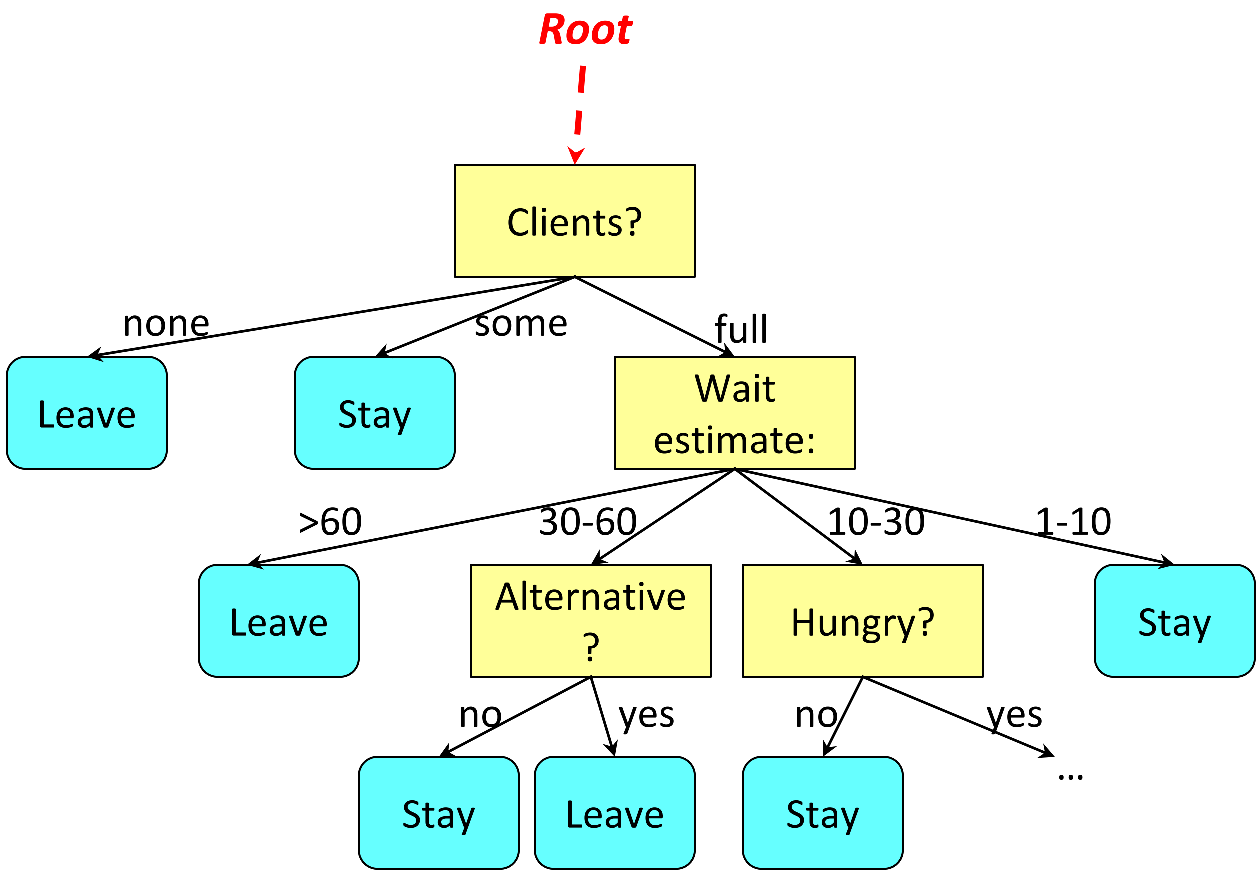
Decision tree – mental model



- We build mental models for such situations

Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Applications of decision trees

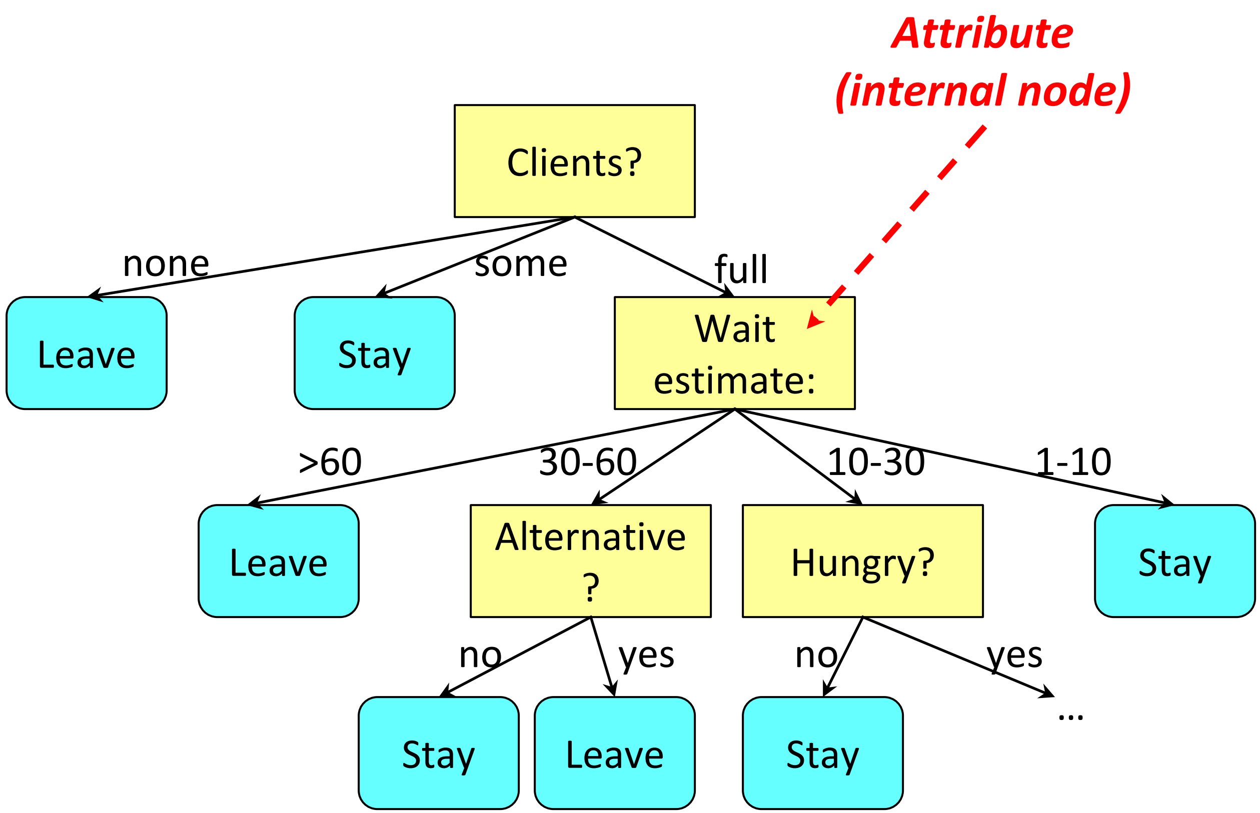
Decision tree – structure



Decision trees

- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Applications of decision trees

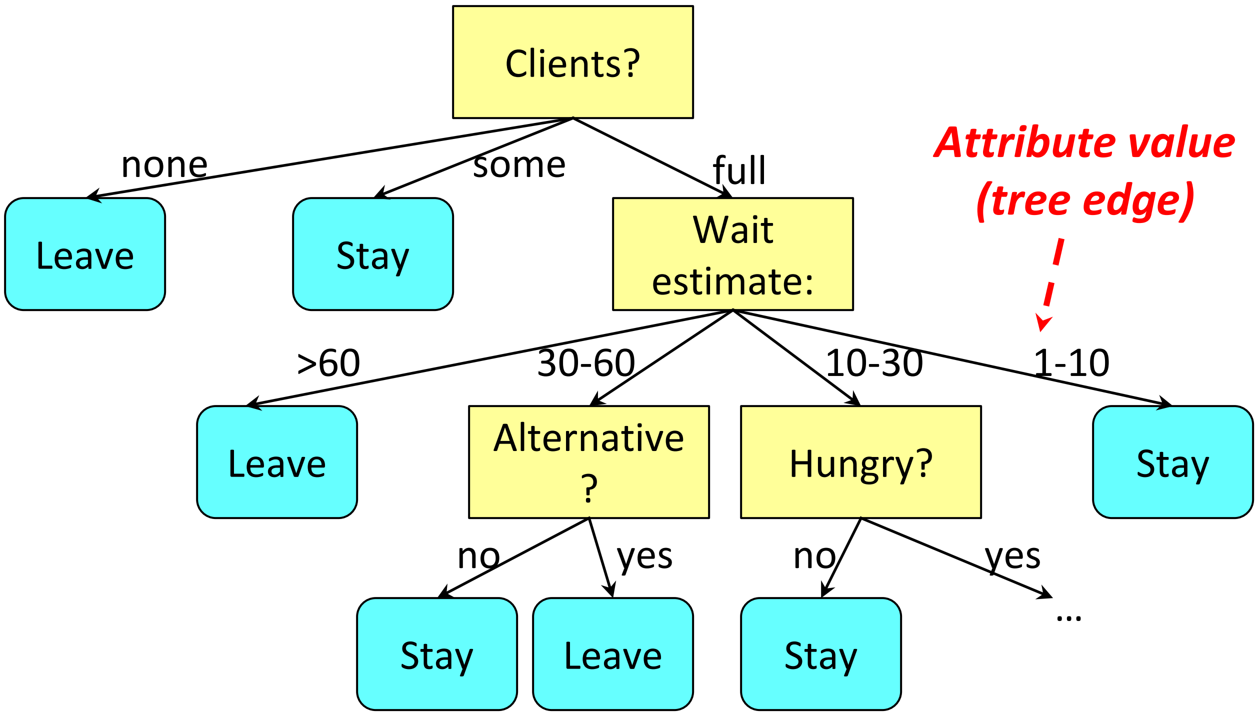
Decision tree – structure



Decision trees

- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Applications of decision trees

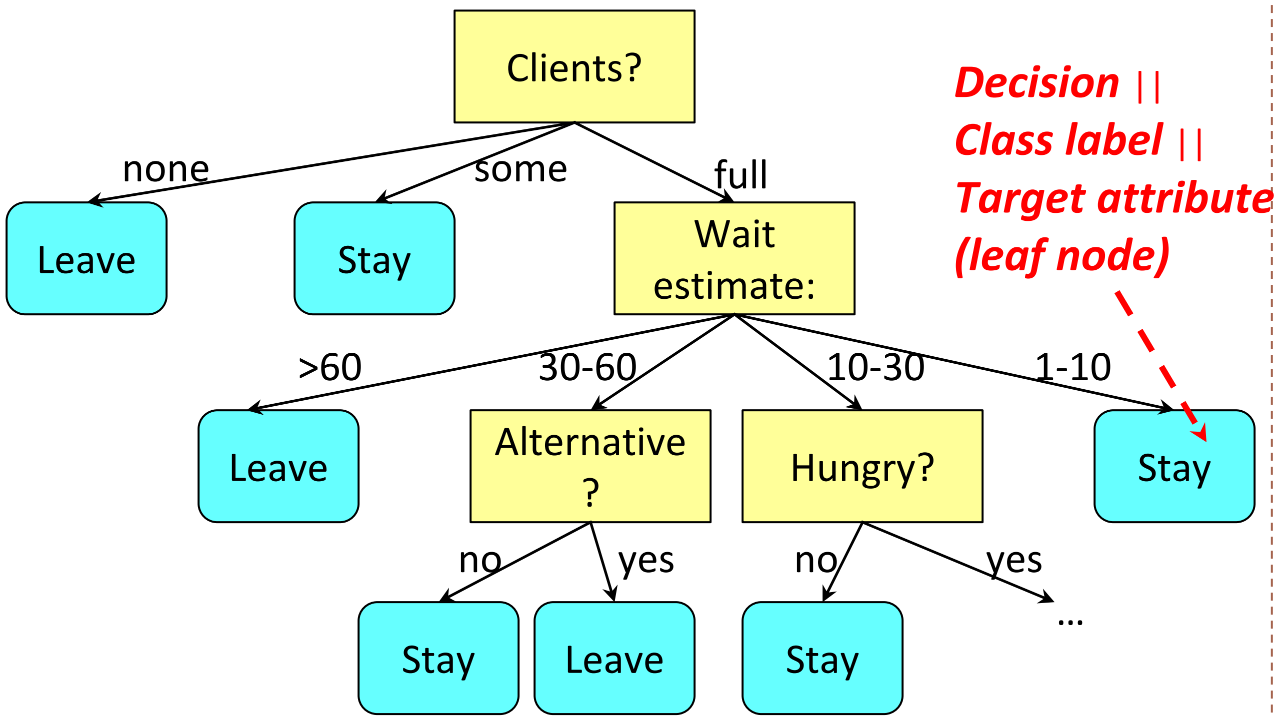
Decision tree – structure



Decision trees

- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Applications of decision trees

Decision tree – structure



Decision trees as a Machine Learning task

- ML is looking for *hidden* patterns, structures, models
- **Task**: *generate a decision tree model from tabular data for which the decision (class label) is known*
- Teach computer to generate the model *automatically,* and then use the model to make an autonomous decision or to assist us with the decision

Decision tree induction

- Top-down recursive divide-and-conquer algorithm
  - First: an attribute is selected for root node and an outgoing edge (a branch) is created for each possible attribute value
  - Then: the instances are split into subsets (one for each branch extending from the node) based on the value of the selected attribute
  - Finally: the same procedure is repeated recursively for each branch, using only instances that reached that branch
- Process stops if all instances in the node have the same class label

Example: Weather dataset

Outlook	Temp, C	Play
Sunny	30	Yes
Overcast	15	No
Sunny	16	Yes
Rainy	27	Yes
Overcast	25	Yes
Overcast	17	No
Rainy	17	No
Rainy	35	Yes

Weather $\xrightarrow{?}$ Play (Yes, No)

Categorizing numeric attributes

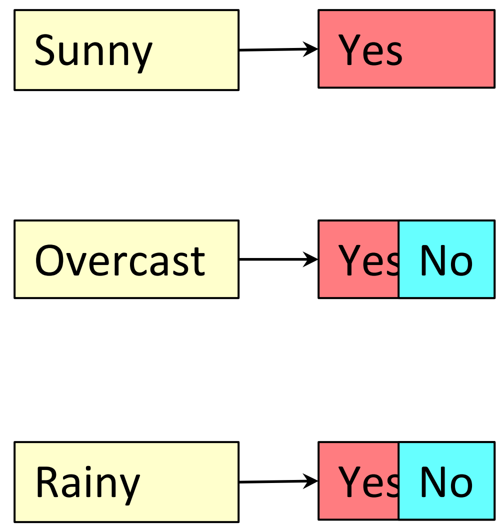
Temp
30
15
16
27
25
17
17
35

→

Temp
Hot
Chilly
Chilly
Warm
Warm
Chilly
Chilly
Hot

Decision tree from the weather dataset

Outlook	Temp	Play
Sunny	Hot	Yes
Overcast	Chilly	No
Sunny	Chilly	Yes
Rainy	Warm	Yes
Overcast	Warm	Yes
Overcast	Chilly	No
Rainy	Chilly	No
Rainy	Hot	Yes



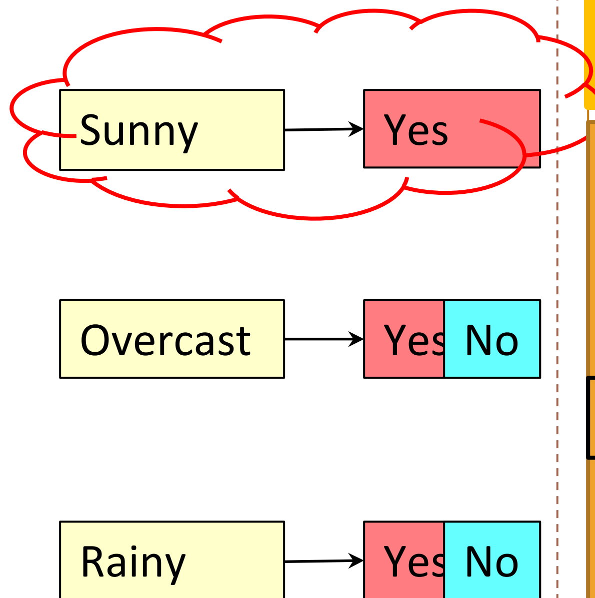
Weather $\xrightarrow{?}$ Play (Yes, No)

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Applications of decision trees

Observations about outlook

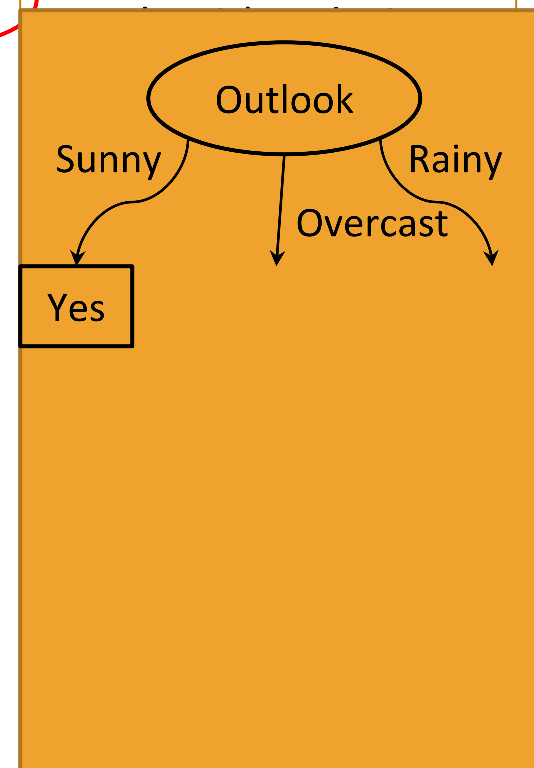
Decision tree from the weather dataset

Outlook	Temp	Play
Sunny	Hot	Yes
Overcast	Chilly	No
Sunny	Chilly	Yes
Rainy	Warm	Yes
Overcast	Warm	Yes
Overcast	Chilly	No
Rainy	Chilly	No
Rainy	Hot	Yes



Weather $\xrightarrow{?}$ Play (Yes, No)

- Decision trees
- Supervised learning
- Tree induction algorithm



Observations about outlook: if it is sunny, always play

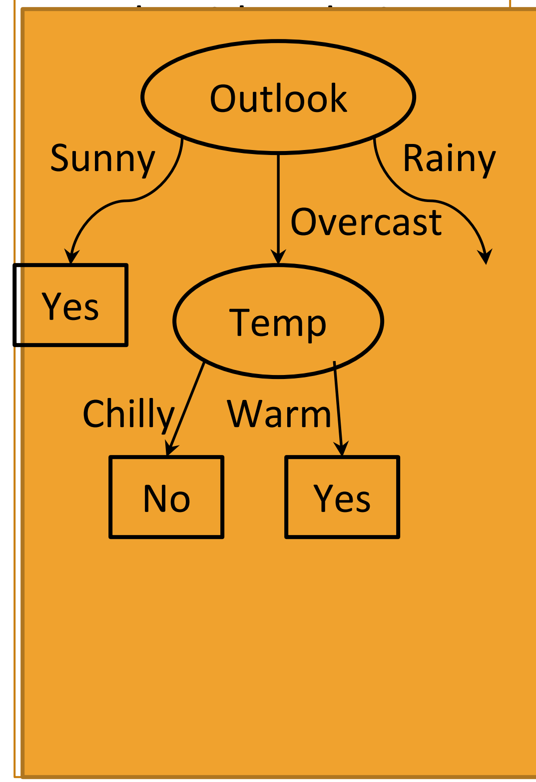
Decision tree from the weather dataset

- Decision trees
- Supervised learning
- Tree induction algorithm

Outlook	**Temp**	**Play**
Sunny	Hot	Yes
Overcast	Chilly	No
Sunny	Chilly	Yes
Rainy	Warm	Yes
Overcast	Warm	Yes
Overcast	Chilly	No
Rainy	Chilly	No
Rainy	Hot	Yes

Overcast, Chilly → No

Overcast, Warm → Yes



Weather $\xrightarrow{\quad ? \quad}$ Play (Yes, No)

- Adding temperature to outlook = overcast to arrive to a decision

Decision tree from the weather dataset

Outlook	Temp	Play
Sunny	Hot	Yes
Overcast	Chilly	No
Sunny	Chilly	Yes
Rainy	Warm	Yes
Overcast	Warm	Yes
Overcast	Chilly	No
Rainy	Chilly	No
Rainy	Hot	Yes

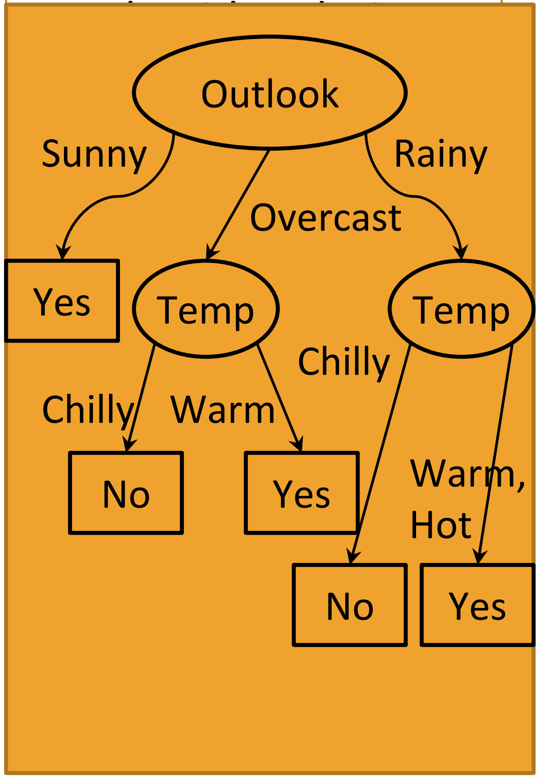
Rainy, Warm → Yes

Rainy, Chilly → No

Rainy, Hot → Yes

Weather $\xrightarrow{?}$ Play (Yes, No)

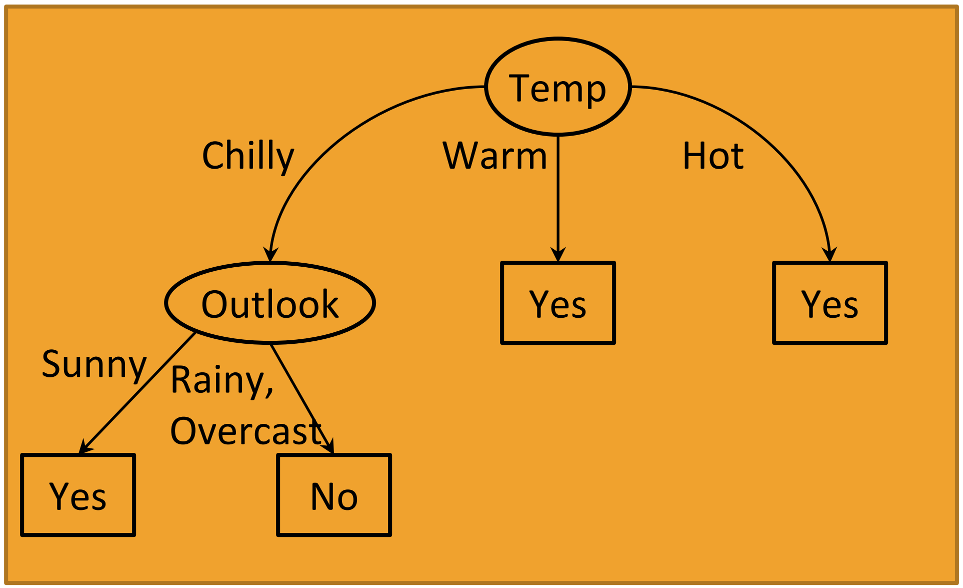
- Decision trees
- Supervised learning
- Tree induction algorithm

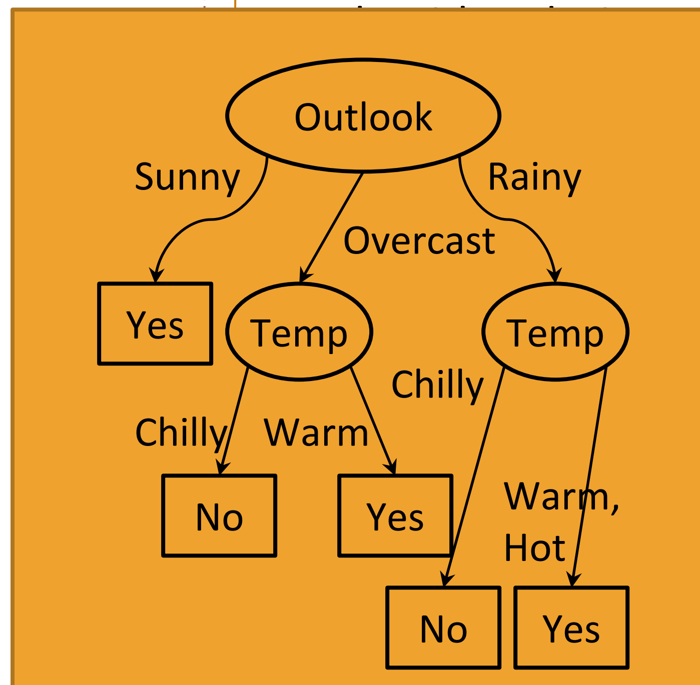


- Adding temperature to outlook = rainy to arrive to a decision

Variations

- There are many different possible trees which fit the same data



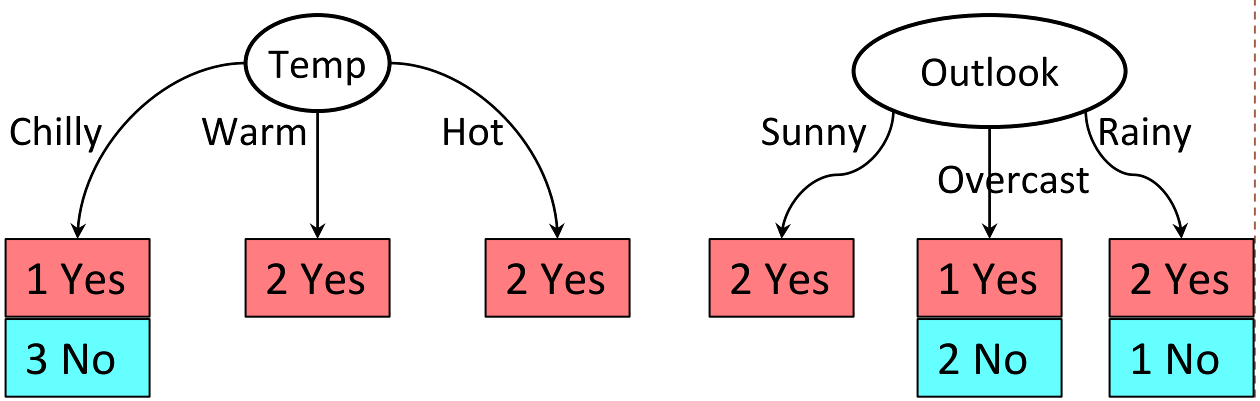


Which tree is better?

- Decision trees
- Supervised learning
- Tree induction algorithm

Design issues

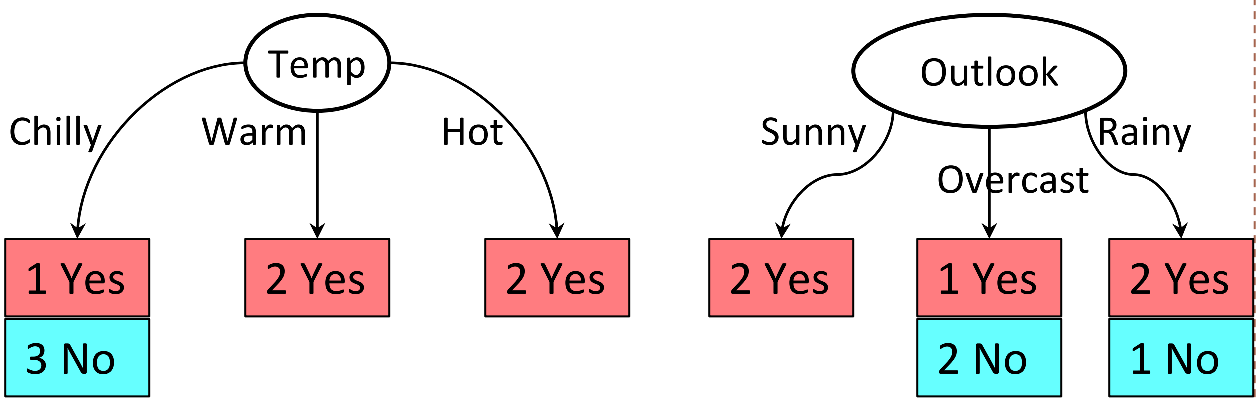
- What attribute to select at each step for splitting?



- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
- Applications of decision trees

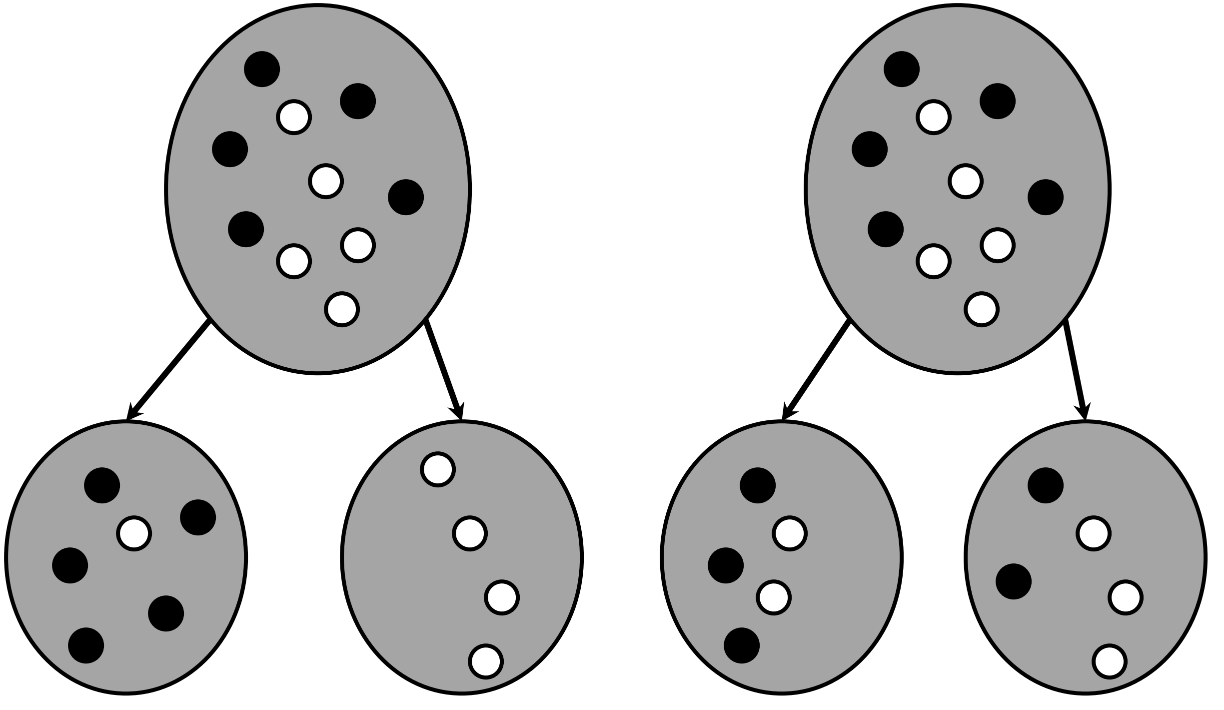
Best splitting attribute: intuition

- Select attribute which divides records into most class-homogenous groups – into nodes with the highest possible *purity*



Purity

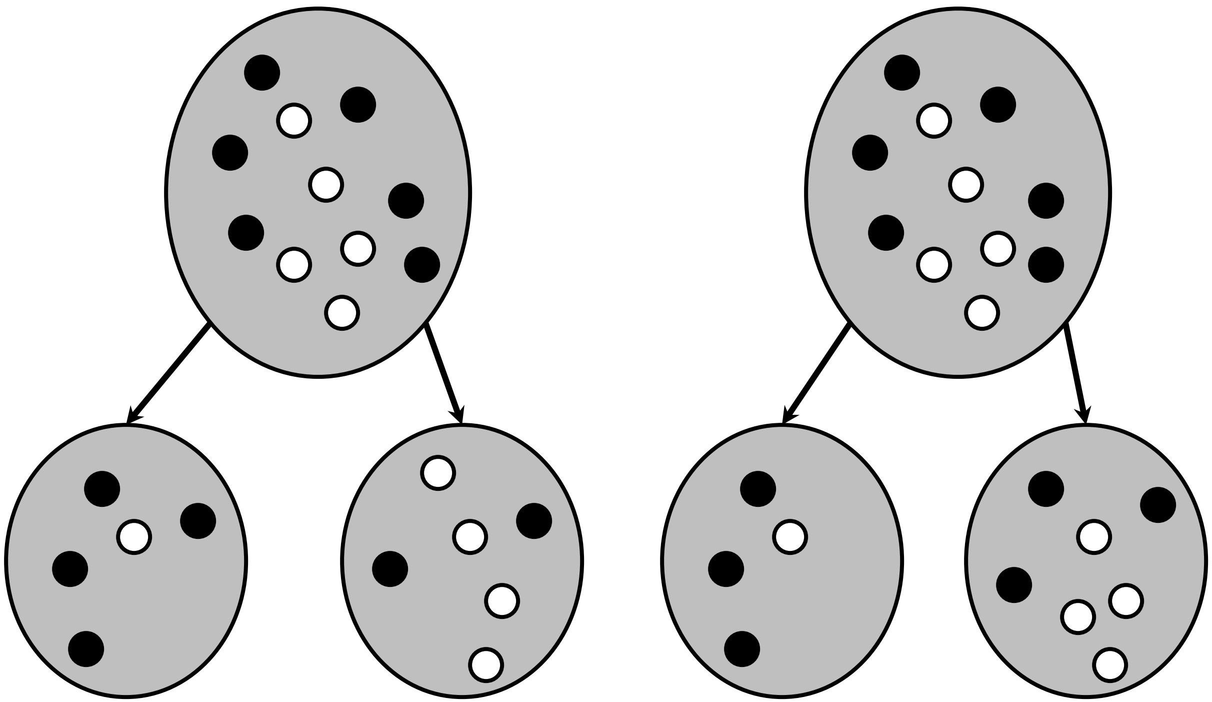
- Which split is better?



- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
- Applications of decision trees

Purity

- And now?

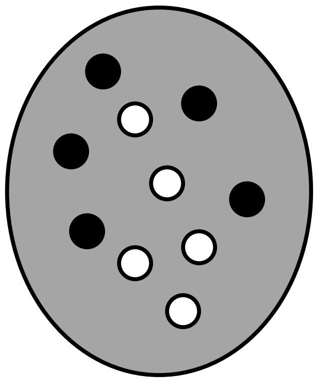


- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - **Best splitting attribute**
- Applications of decision trees

We need a measure of node purity

Purity measure: *GINI* score

- The probability that two items chosen at random are in the same class: the sum of squares of the proportions of the classes

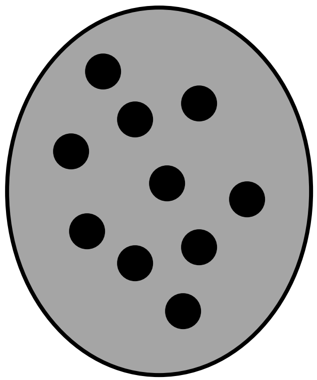


A node with evenly mixed classes has GINI:
$0.5^2+0.5^2=0.5$

The chance of picking the same class twice by random selection is: the probability of picking 2 white dots twice (0.5^2) or picking 2 black dots twice (0.5^2).

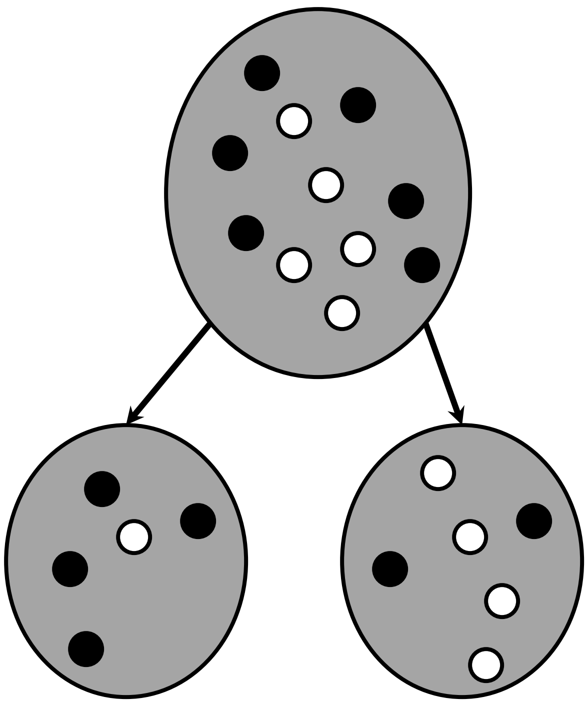
Purity measure: *GINI* score

- The probability that two items chosen at random are in the same class: the sum of squares of the proportions of the classes



A node with one homogenous class has GINI: 1.0
(The chance of picking the same class twice is 100%)

Best split with GINI score

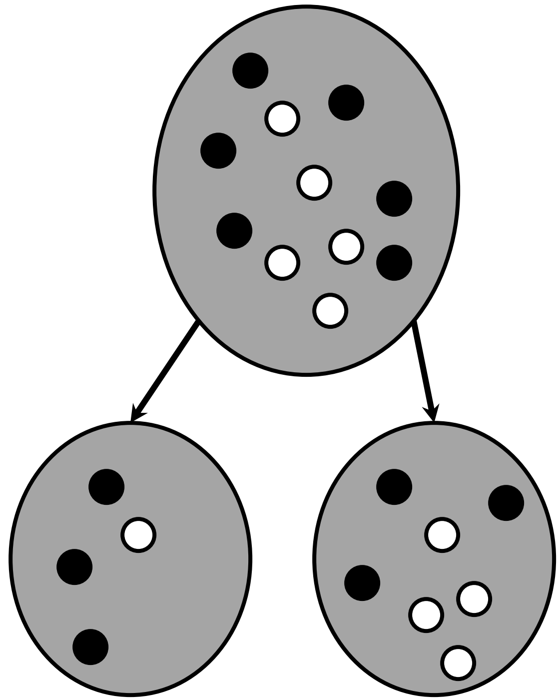


GINI(1,4)=1/5^2+4/5^2=0.04+0.64=0.68

GINI(2,4)=2/6^2+4/6^2=0.11+0.44=0.55

We take a *weighted average*:
5/11*0.68 + 6/11*0.55=0.31+0.3=0.61

Best split with GINI score



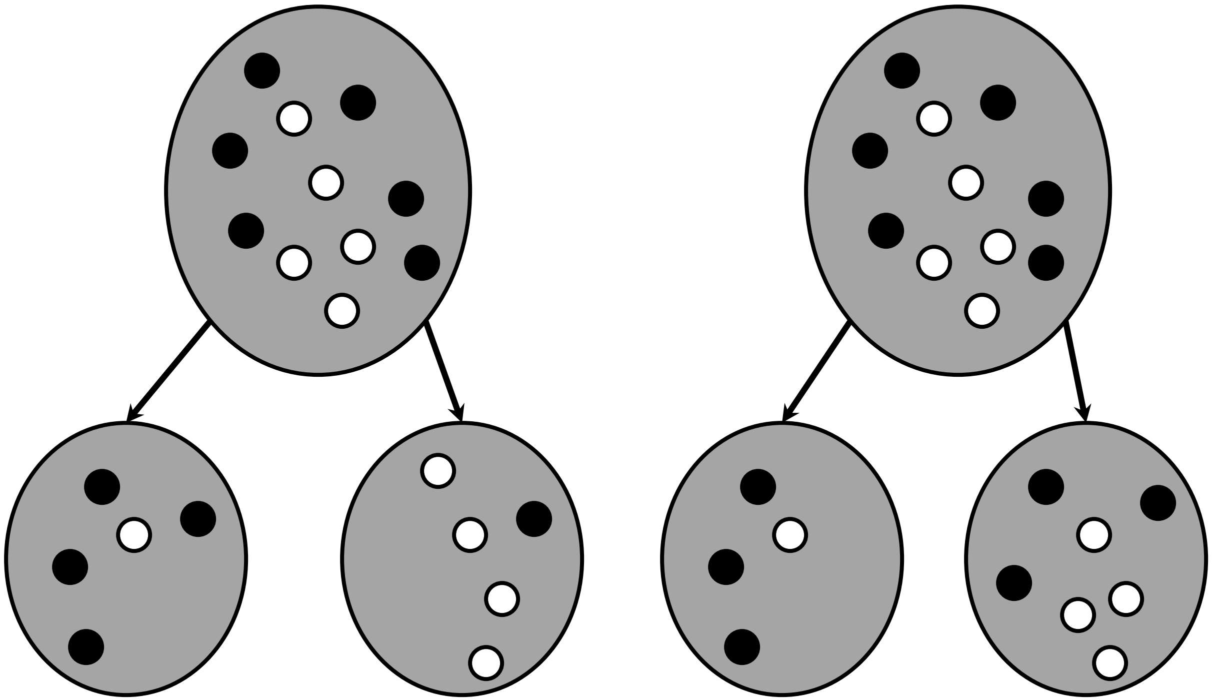
GINI(3,1)=3/4^2+1/4^2=0.56+0.06=0.62

GINI(3,4)=3/7^2+4/7^2=0.18+0.33=0.51

We take a *weighted average*:
4/11*0.62 + 7/11*0.51=0.23+0.32=0.55

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
- Applications of decision trees

Comparing average GINI scores



GINI = 0.61

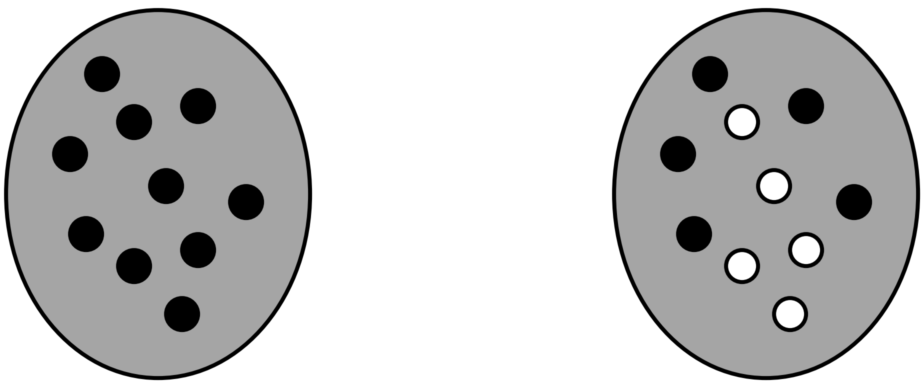
GINI = 0.55

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
    - Best splitting attribute
- Applications of decision trees

The larger the GINI score, the better

Purity measure: *Entropy*

- In information theory *entropy* is a measure of how disorganized the information is



A node with one homogenous class has entropy: 0 (very organized)

A node with evenly mixed population has the largest entropy: 1.0 (most disorganized)

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Best splitting attribute
- Applications of decision trees

The smaller the entropy, the better

Bits

- We are watching a sequence of independent random samples of X
- We see that X has four possible values

P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

- So we might see: BAACBADCDADDDA…
- We transmit data over a binary serial link.
- We can encode each symbol with two bits (e.g. A=00, B=01, C=10, D = 11)

0100001001001110110011111100…

Fewer Bits

- Someone tells us that the probabilities are not equal

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8

- It's possible…

…to invent an encoding for your transmission that only uses 1.75 bits on average per symbol.

A	0
B	10
C	110
D	111

Here is one.

General Case

- Suppose X can have one of *m* values…

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

- What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution? It's

$$entropy(p_1,\ldots,p_m) = -p_1 \log_2 p_1 - \ldots p_m \log_2 p_m$$

- Well, Shannon got to this formula by setting down several desirable properties for uncertainty, and then finding it.

Tree node entropy

- Suppose class attribute X in a given tree node occurs in the following proportions

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	....	$P(X=V_m) = p_m$

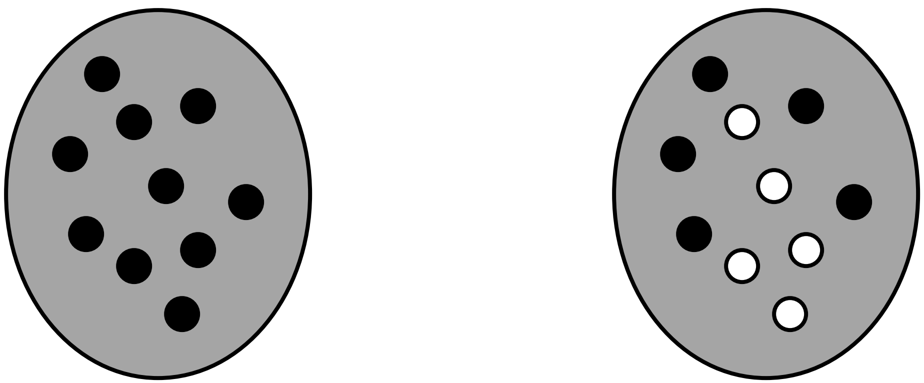
- By finding entropy of the node, we evaluate how many bits are needed to encode this node

$$entropy(p_1,\ldots,p_m) = -p_1 \log_2 p_1 - \ldots p_m \log_2 p_m$$

- **The smaller the number of bits to encode the entire tree, the better: the *minimum description length* (MDL) principle**

Computing entropy of a node

- Compute entropy of a node

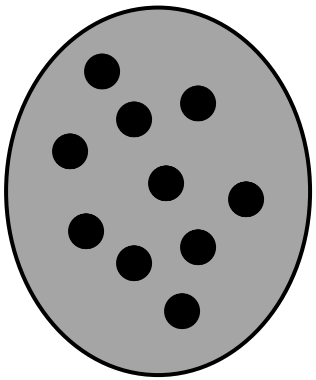


Entropy(10,0)=
-10/10*log (10/10)-0*log (0)
=0

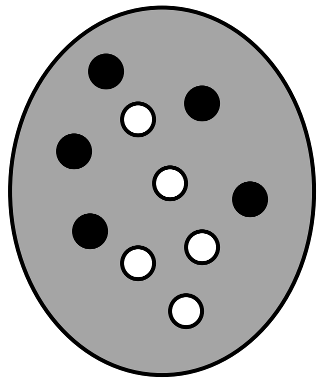
=0 in this formula

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
- Best splitting attribute
- Applications of decision trees

Computing entropy of a node



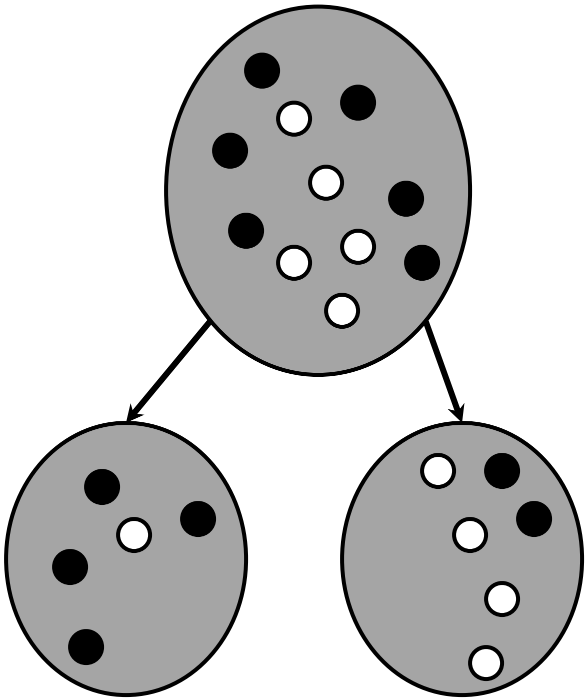
Entropy(10,0)=0



Entropy(5,5)=
-5/10*log 5/10 -5/10*log(5/10)
=1

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
- Applications of decision trees

Best split with Entropy reduction

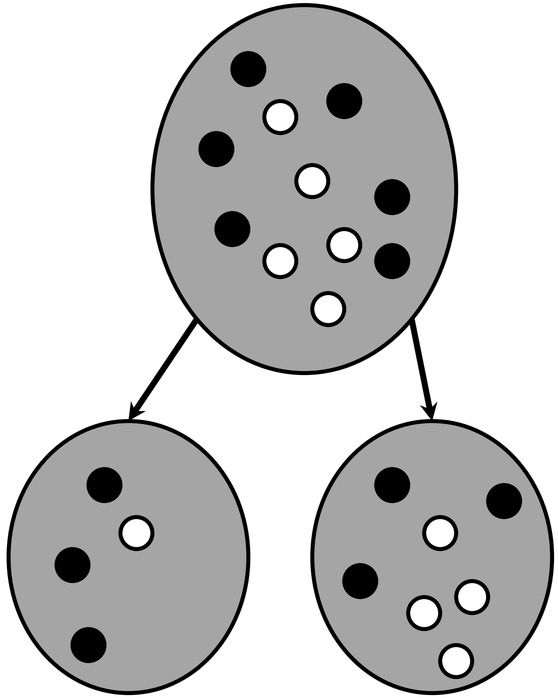


Entropy(4,1)=-4/5 log 4/5-1/5 log 1/5= 0.26+0.46=0.72

Entropy(2,4)=-2/6 log 2/6 - 4/6 log 4/6=0.53+0.39=0.92

We take a *weighted average*:
5/11*0.72 + 6/11*0.92=0.33+0.5=0.83

Best split with Entropy reduction

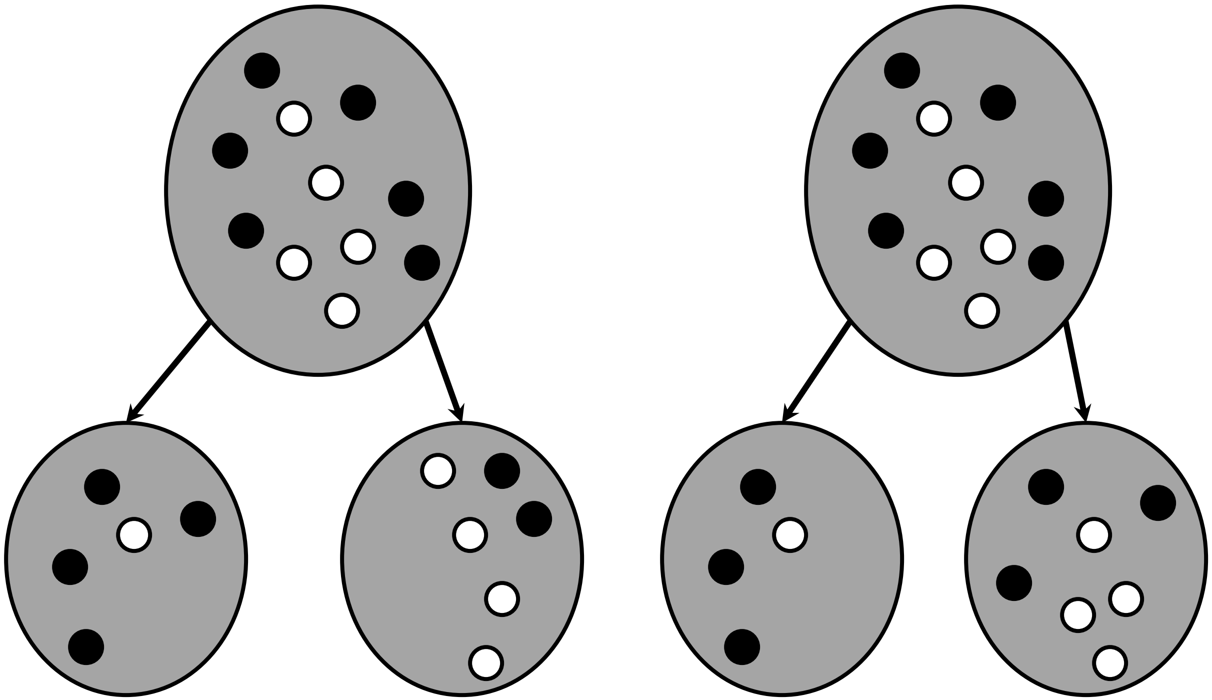


Entropy(3,1)=-3/4 log 3/4-1/4 log 1/4= 0.31+0.5=0.81

Entropy(3,4)=-3/7 log 3/7 - 4/7 log 4/7=0.52+0.46=0.98

We take a *weighted average*:
4/11*0.81 + 7/11*0.98=0.295+0.63=0.92

Comparing average entropies



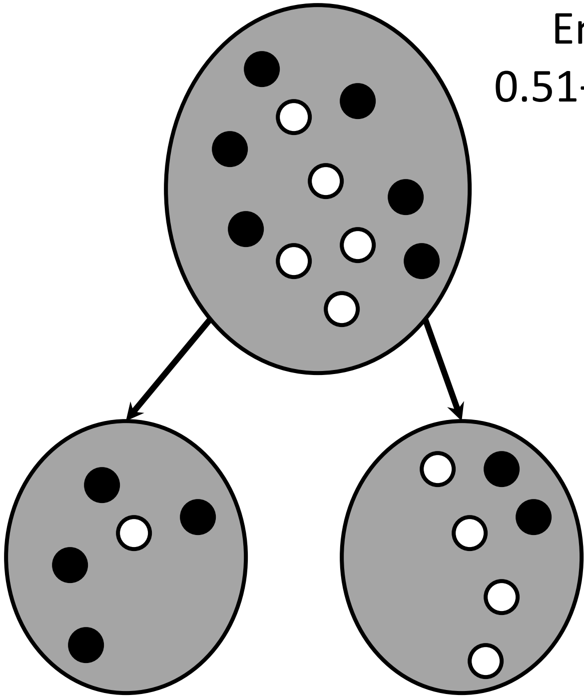
Entropy = 0.83

Entropy=0.92

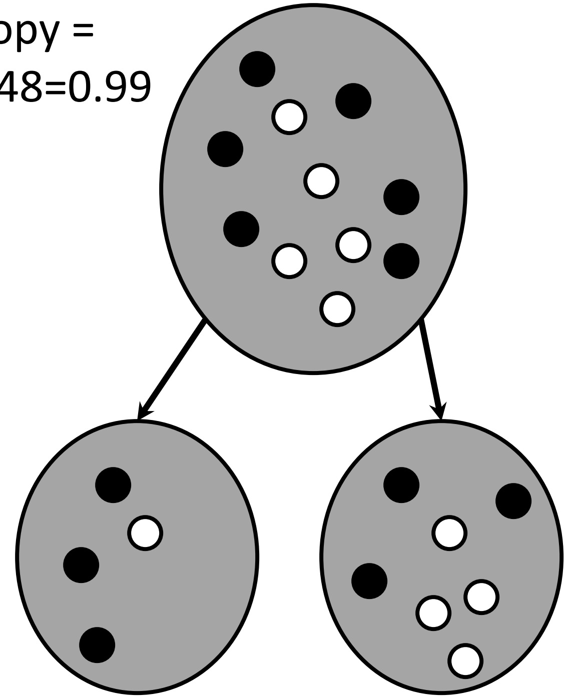
- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - **Best splitting attribute**
- Applications of decision trees

The smaller the entropy, the better

Entropy reduction or *information gain*



Entropy = 0.51+0.48=0.99



Entropy = 0.83

Entropy=0.92

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - **Best splitting attribute**
- Applications of decision trees

In this case, it might be better not to split at all, since the information gain is small

To split or not to split?

- Not to split: when the node consists of elements of the same class
- Not to split: when the node consists of elements which have the same attribute values, except the class attribute
- Not to split: when there is no information gain (no reduction in entropy). **Not to split when information gain is insignificant**

Full tree induction algorithm

- **Step 1.** Compute entropy of the instances in the current set with respect to class label (in the beginning – on the entire dataset).
- **Step 2.** For each attribute, compute information gain and select the attribute which gives maximum information gain.
- **Step 3.** Create a node with the selected attribute and create branch for each possible attribute value. Split instances into subsets according to this value.
- **Step 4.** For each subset:
  - **If** no split is possible, create leaf node and mark it with the majority class
  - **Else** go to Step 1

Example: tax cheating dataset

ID	Refund	Marital status	Taxable income	Cheat
1	Yes	Single	125 K	No
2	No	Married	100 K	No
3	No	Single	70 K	No
4	Yes	Married	120 K	No
5	No	Divorced	95 K	Yes
6	No	Married	60 K	No
7	Yes	Divorced	220 K	No
8	No	Single	85 K	Yes
9	No	Married	105 K	No
10	No	Single	110 K	Yes

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
  - When to stop splitting
- Applications of decision trees

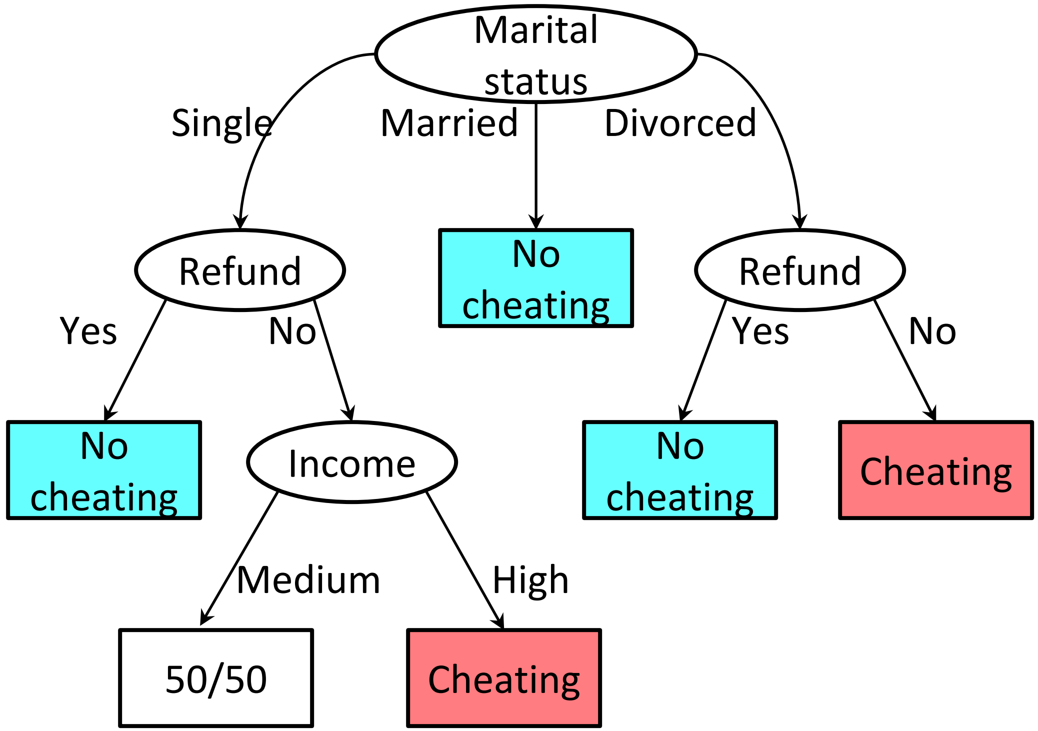
https://hope.simons-rock.edu/~mbarsky/research2018/decision_trees/tree111.html

Categorizing numeric features

>=100K → high
<100K → medium

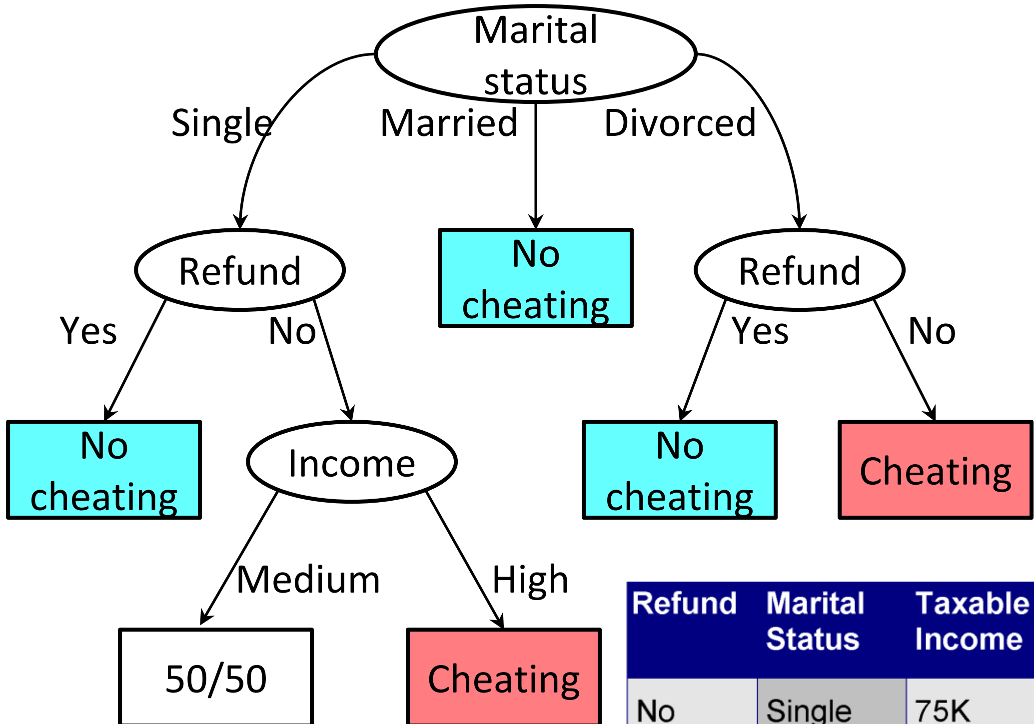
ID	Refund	Marital status	Taxable income	Cheat
1	Yes	Single	high	No
2	No	Married	high	No
3	No	Single	medium	No
4	Yes	Married	high	No
5	No	Divorced	medium	Yes
6	No	Married	medium	No
7	Yes	Divorced	high	No
8	No	Single	medium	Yes
9	No	Married	high	No
10	No	Single	high	Yes

Decision tree for tax cheating dataset



- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
  - When to stop splitting
- Applications of decision trees

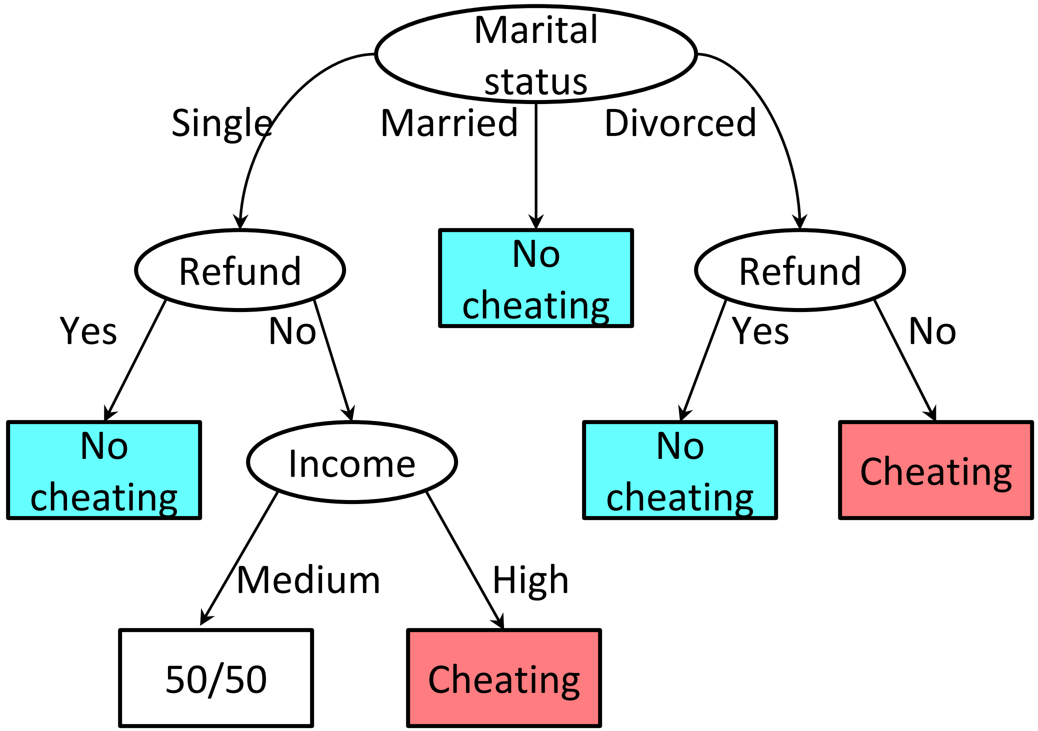
Classify new records



Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
  - When to stop splitting
- **Applications of decision trees**

Identify the most important features



The most important features are at the top of the tree

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
  - When to stop splitting
- Applications of decision trees

When to use decision tree classifier

Use decision trees

- The factors of the decision are not less important than the classification accuracy
- Attributes with nominal values (not numeric) and with low cardinality*
- Categorical class labels with low cardinality*
- There is a set of objective rules underlying the data

Use something else

- Continuous numeric attributes, ordinal attributes
- Hierarchical relationships between classes
- High-cardinality attributes
- Numeric value prediction

- Decision trees
- Supervised learning
- Tree induction algorithm
- Algorithm design issues
  - Best splitting attribute
  - When to stop splitting
- Applications of decision trees

*cardinality - the number of possible distinct values