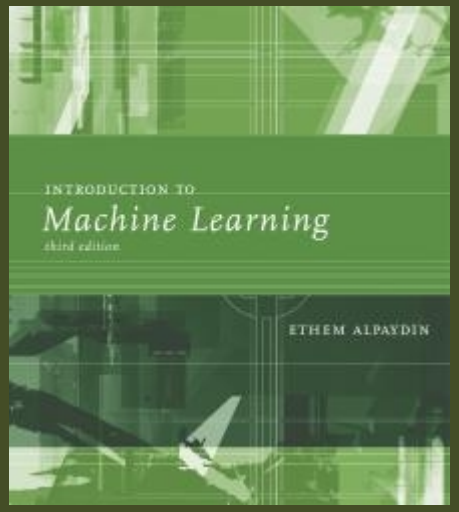
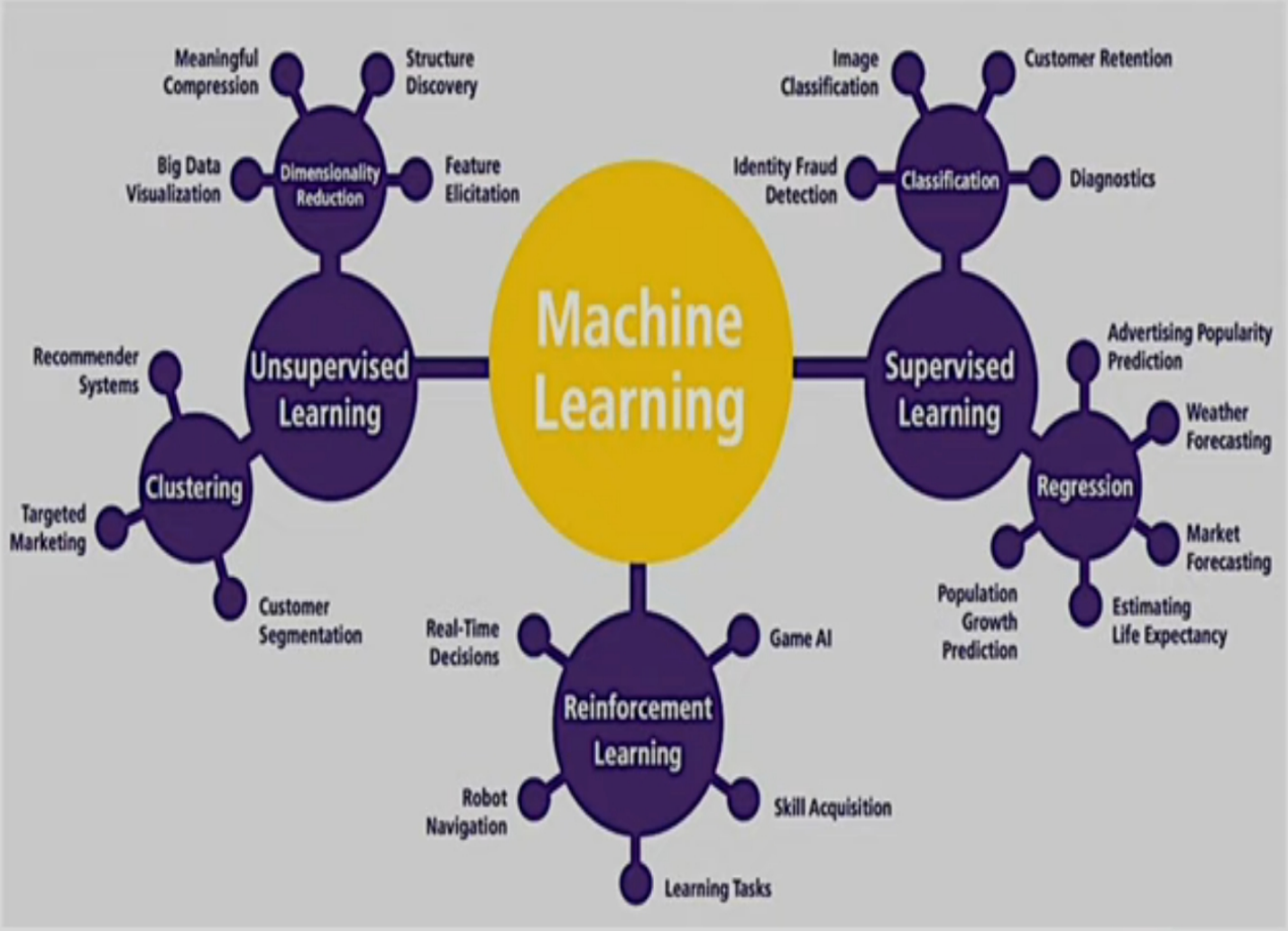


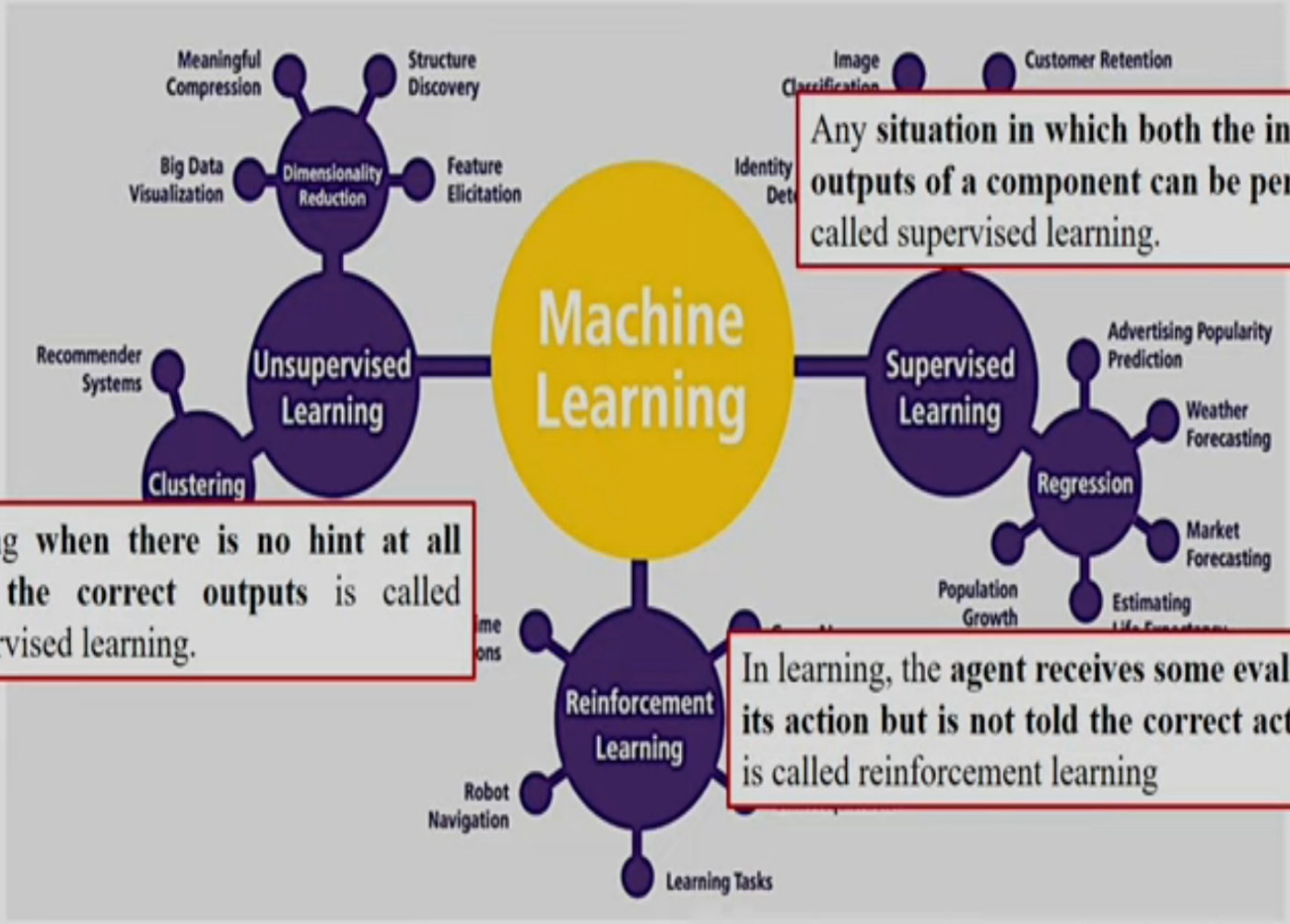
Lecture Slides for

DECISION TREE

Machine Learning



Machine Learning



Any **situation in which both the inputs and outputs of a component can be perceived** is called supervised learning.

Learning **when there is no hint at all about the correct outputs** is called unsupervised learning.

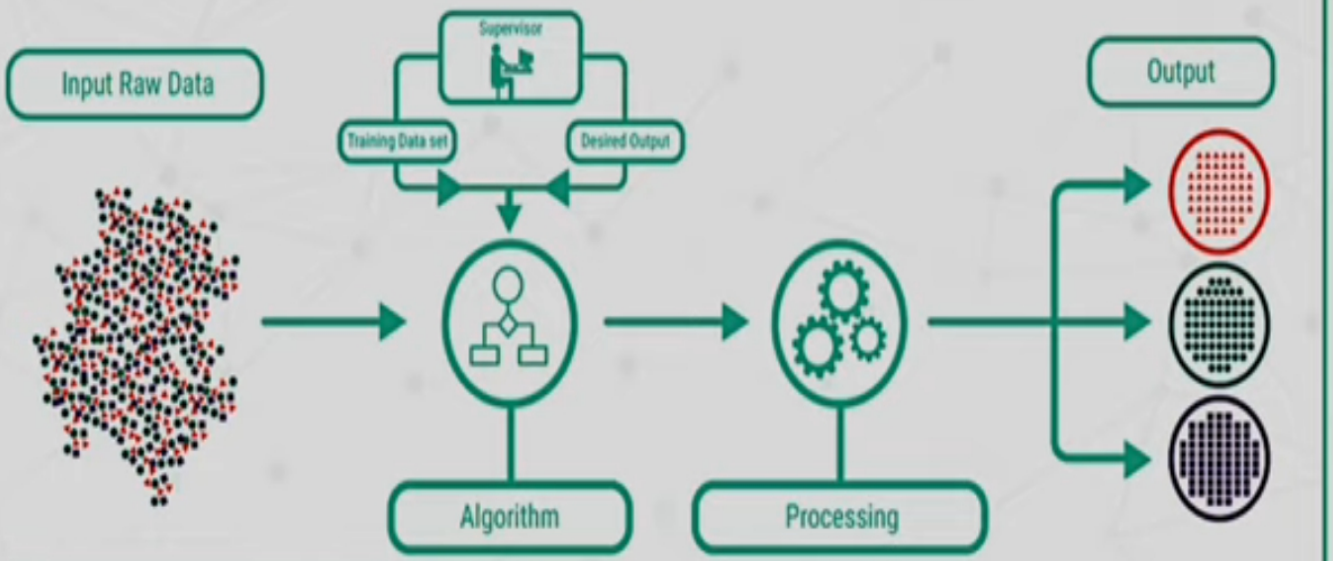
In learning, the **agent receives some evaluation of its action but is not told the correct action**. This is called reinforcement learning

Image Source: DHL, Artificial Intelligence in Logistics, 2018.

Supervised Learning

The set of (training/learning) data consists of a set of input data and correct responses (labels) corresponding to every piece of data.

Supervised learning: a type of machine learning that learns from training data with labels as learning targets. It is the most widely used type of machine learning



The algorithm has to generalize such that it is able to correctly (or with low error margin) respond to all possible inputs.

Image Source: Data Demystified — Machine Learning: http://towardsdatascience.com

Learn to predict output when **given an input vector**

Machine Learning

A **computer program** is said to **learn** from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its **performance** at tasks in T, as measured by P, **improves with experience** E.

– Tom Mitchell

A **computer system learns from data**, which **represent some "past experiences" of an application domain**. Our focus: learn a target function that can be used to predict the values of a discrete class attribute.

Inductive Learning

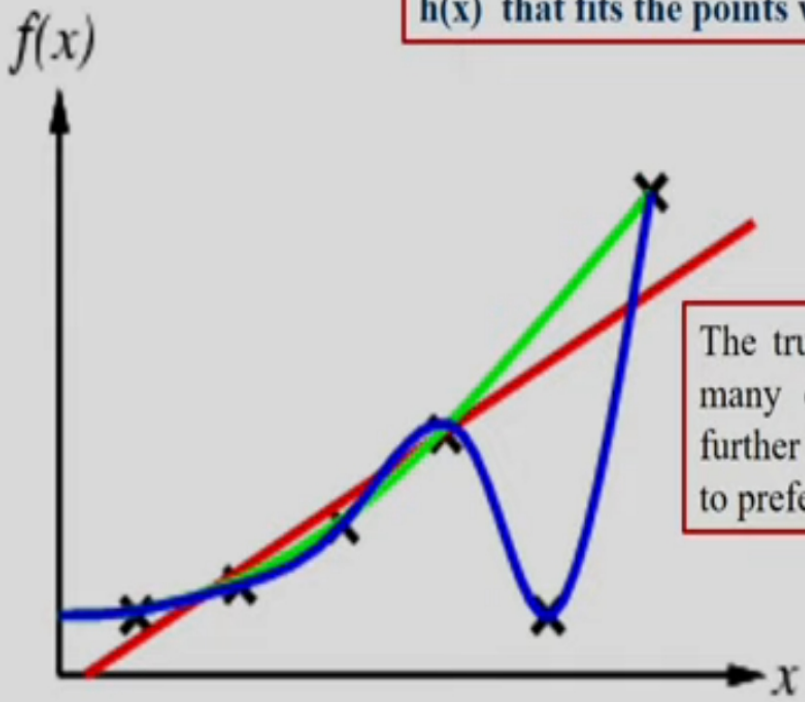
- In supervised learning, the learning element is given the correct (or approximately correct) value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback.
  - An example is a pair (x, f(x)), where x is the input and f(x) is the output of the function applied to x.

Pure Inductive Inference

Given a collection of examples of f, return a function h that approximates f. The function h is called a hypothesis.

Inductive Learning

From plane geometry: Examples are (x,y) points in the plane, where y = f(x). **The task is to find a function h(x) that fits the points well.**

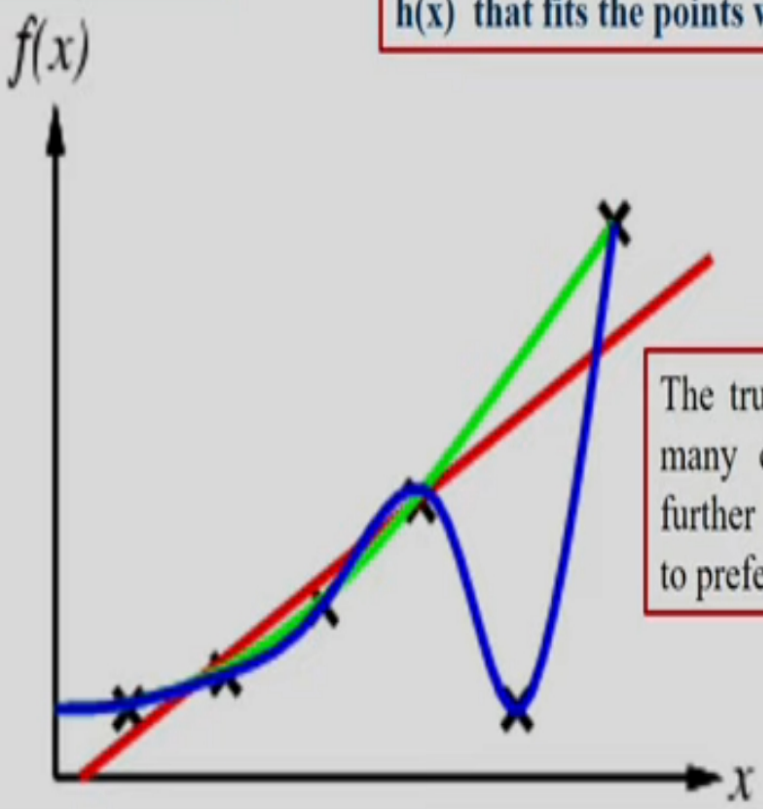


The true *f* is unknown, so there are many choices for *h*, but without further knowledge, we have no way to prefer one over the other.

Inductive Learning

Three hypotheses for functions from which these examples could be drawn

From plane geometry: Examples are (x,y) points in the plane, where y = f(x). **The task is to find a function h(x) that fits the points well.**

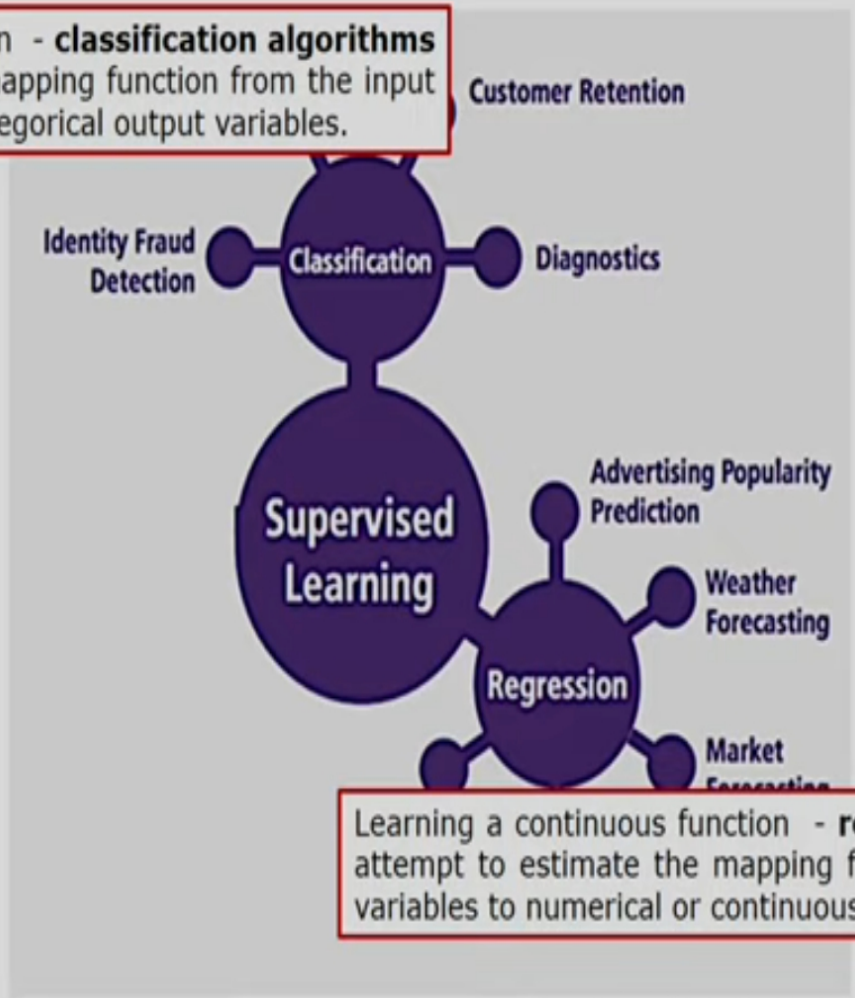


The true *f* is unknown, so there are many choices for *h*, but without further knowledge, we have no way to prefer one over the other.

Any preference for one hypothesis over another, beyond mere consistency with the examples, is called a bias. There are always a large number of possible consistent hypotheses; learning algorithms exhibit bias.

Supervised Learning

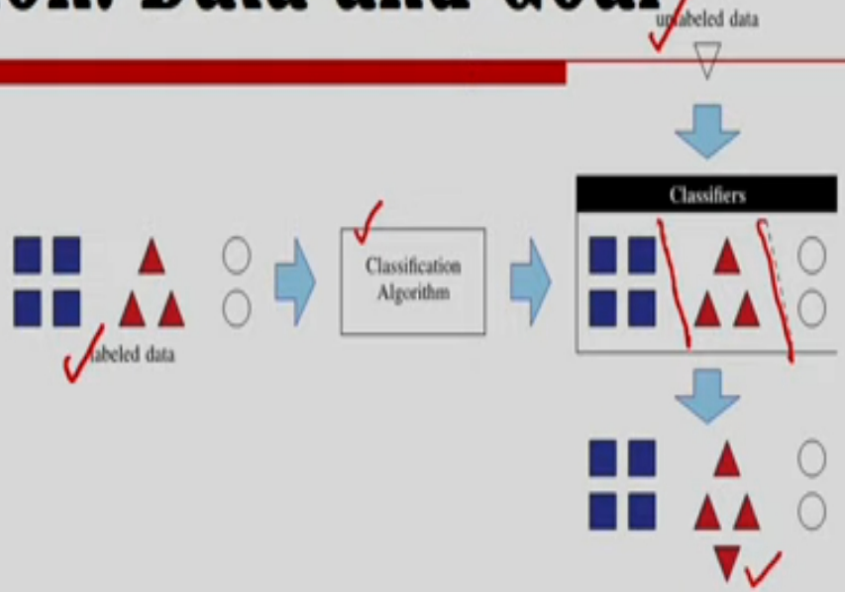
Learning a discrete function - **classification algorithms** attempt to estimate the mapping function from the input variables to discrete or categorical output variables.



Learning a continuous function - **regression algorithms** attempt to estimate the mapping function from the input variables to numerical or continuous output variables.

Image Source: DHL, Artificial Intelligence in Logistics, 2018

Classification: Data and Goal



- Data: A **set of data records** (also called examples, instances or cases) described by
  - k attributes: $A_1, A_2, \ldots A_k$.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a **classification model** from the data that can be used to predict the classes of new (future, or test) cases/instances.

Classification:

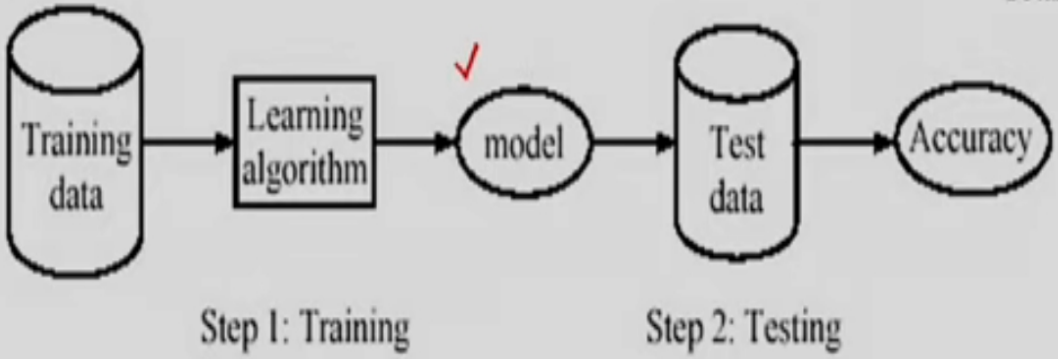
1. Learning (training): Learn a model using the training data

Model construction: describing a set of predetermined classes; Each tuple/sample is assumed to belong to a predefined class, as determined by the class label. The set of tuples used for model construction is the training set.

2. Testing: Test the model using unseen test data to assess the model accuracy.

Model usage: for classifying future or unknown objects. If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$



Fundamental Assumption

Assumption: The **distribution of training examples is identical to the distribution of test examples** (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Learning Decision Trees

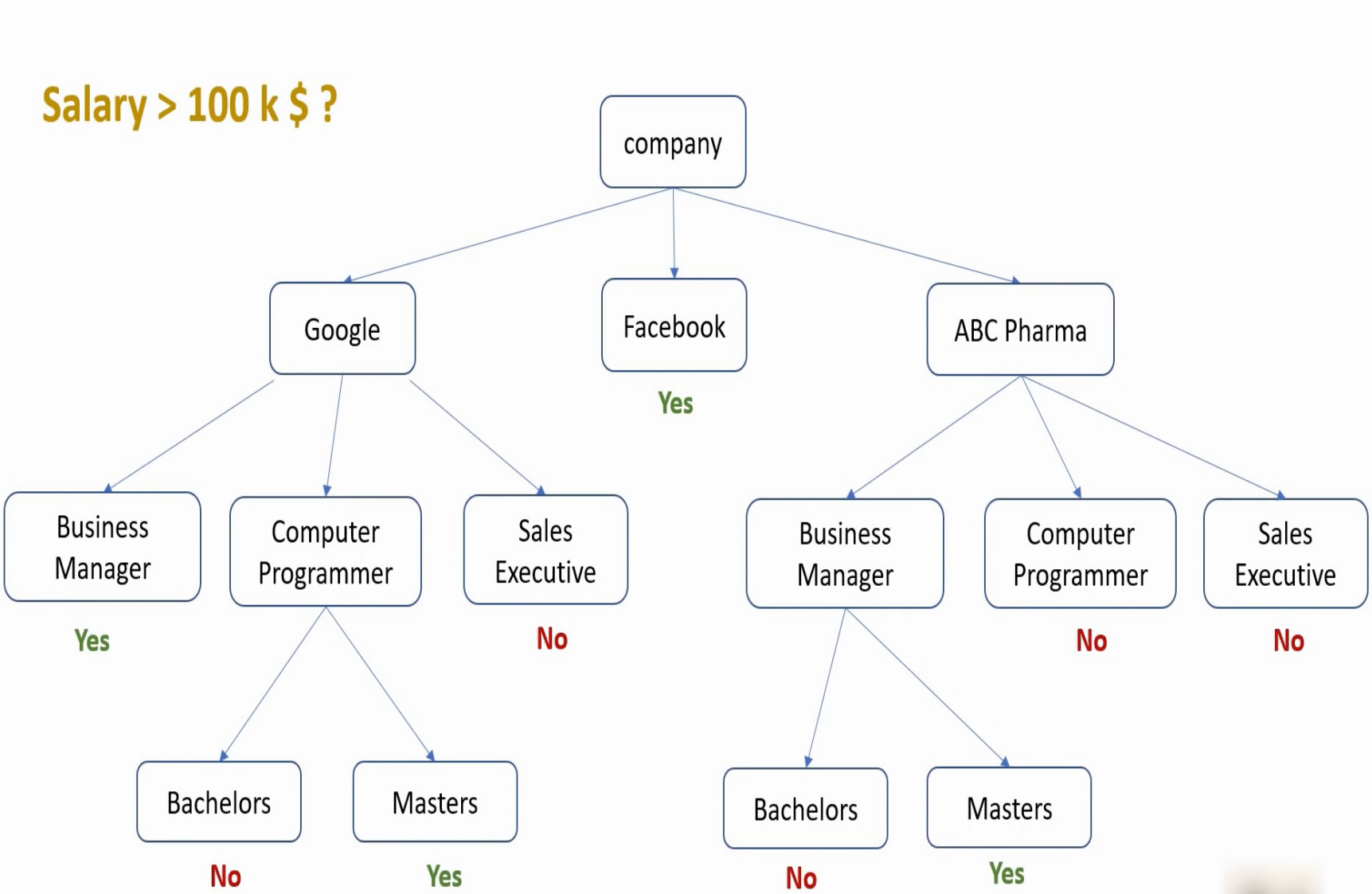
- Decision tree learning is **one of the most widely used techniques for classification**.
  - Its classification accuracy is competitive with other methods;
  - It is very efficient.
  - It serves as a good introduction to the area of inductive learning.

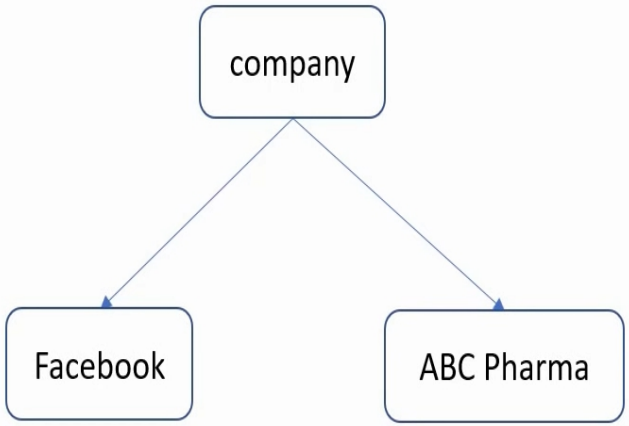
Decision tree learning uses a **decision tree (as a predictive model)** to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves)

Decision Tree as Performance Element

- A decision tree takes as input an object or situation described by a set of properties, and outputs a yes/no "decision." Decision trees represent Boolean functions.
  - output values are true or false
  - conceptually the simplest case, but still quite powerful
- Each internal node in the tree corresponds to a test of the value of one of the properties, and the branches from the node are labelled with the possible values of the test.
  - a sequence of test is performed, testing the value of one of the attributes in each step
  - when a leaf node is reached, its value is returned
  - good correspondence to human decision-making

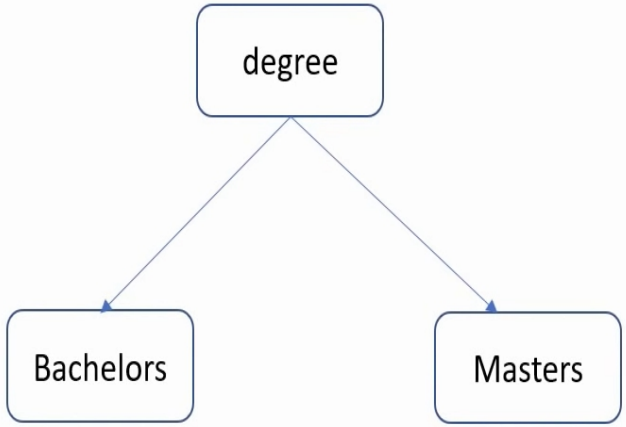
Company	Job	Degree	Salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0
google	computer programmer	masters	1
abc pharma	sales executive	masters	0
abc pharma	computer programmer	bachelors	0
abc pharma	business manager	bachelors	0
abc pharma	business manager	masters	1
facebook	sales executive	bachelors	1
facebook	sales executive	masters	1
facebook	business manager	bachelors	1
facebook	business manager	masters	1
facebook	computer programmer	bachelors	1
facebook	computer programmer	masters	1


Salary > 100 k $?
company
Google
Facebook
ABC Pharma
Yes
Business Manager
Computer Programmer
Sales Executive
Yes
No
Business Manager
Computer Programmer
Sales Executive
No
No
Bachelors
Masters
No
Yes
Bachelors
Masters
No
Yes



facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors
facebook	computer programmer	masters

abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters



google	sales executive	bachelors
google	business manager	bachelors
google	computer programmer	bachelors
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
facebook	sales executive	bachelors
facebook	business manager	bachelors
facebook	computer programmer	bachelors

google	sales executive	masters
google	business manager	masters
google	computer programmer	masters
abc pharma	sales executive	masters
abc pharma	business manager	masters
facebook	sales executive	masters
facebook	business manager	masters
facebook	computer programmer	masters

Learning decision trees

Decide whether to wait for a table at a restaurant.

Aim is to **learn a definition for the goal predicate WillWait**, where the definition is expressed as a decision tree.

Setting this up as a learning problem; we decide what properties or attributes are available to describe examples in the domain:

1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range ($, $$, $$$)
7. **Raining**: is it raining outside?
8. **Reservation**: have we made a reservation?
9. **Type**: kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate**: estimated waiting time (0-10, 10-30, 30-60, >60)

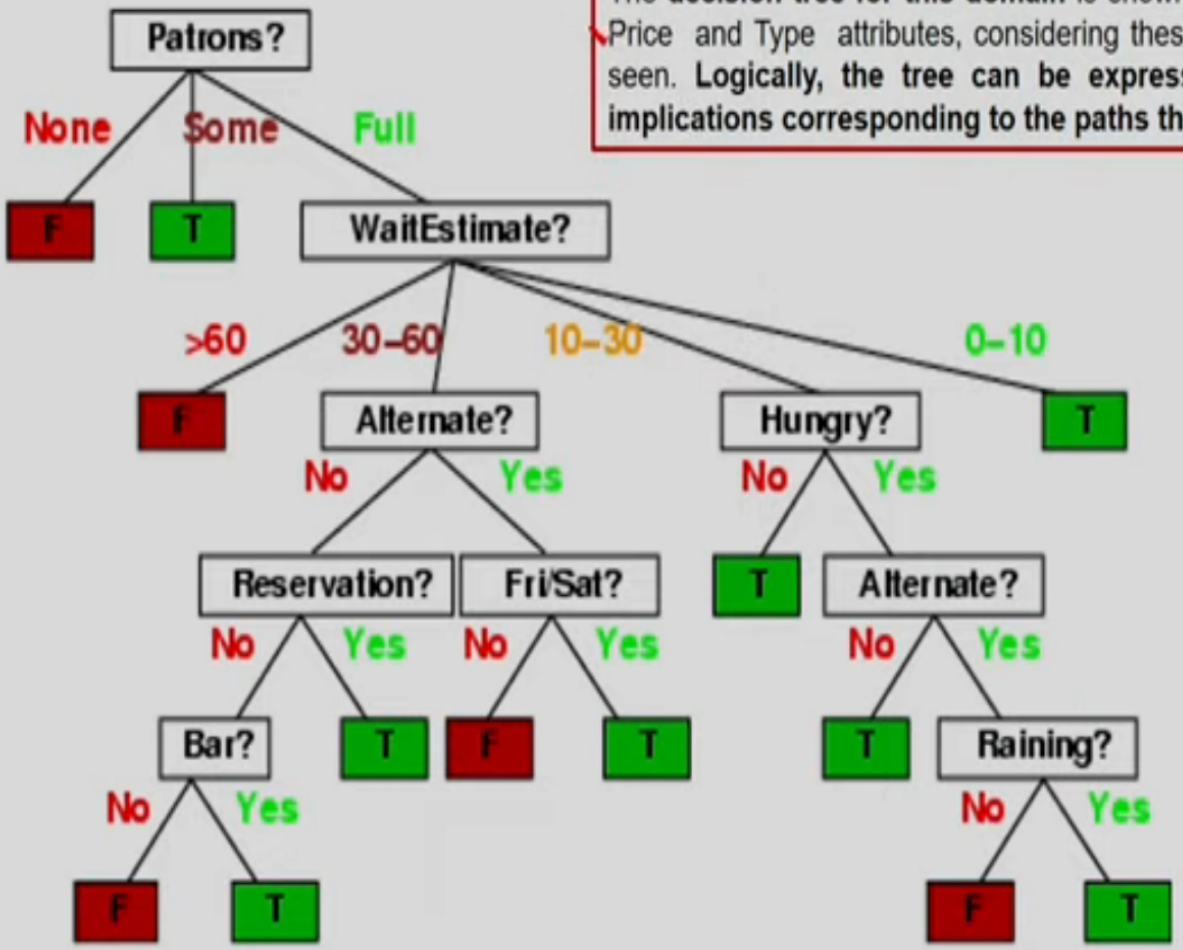
Attribute-based Representations

- Examples described by attribute values
  - Boolean, discrete, continuous
  - E.g., situations where I will/won't wait for a table:
- Classification of examples is positive (T) or negative (F)

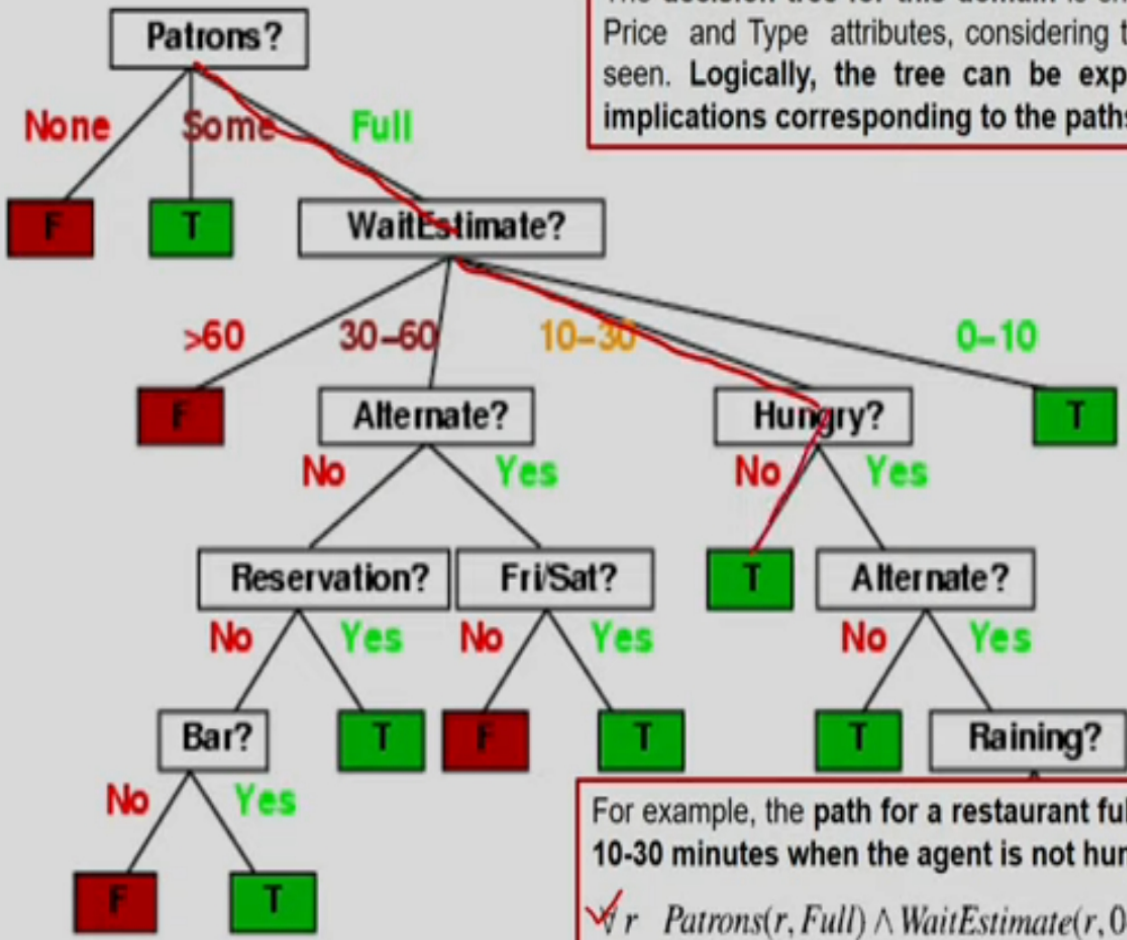
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree

The **decision tree for this domain** is shown. Notice that the tree does not use the Price and Type attributes, considering these to be irrelevant given the data it has seen. **Logically, the tree can be expressed as a conjunction of individual implications corresponding to the paths through the tree ending in Yes nodes.**



Decision Tree



The **decision tree for this domain** is shown. Notice that the tree does not use the Price and Type attributes, considering these to be irrelevant given the data it has seen. **Logically, the tree can be expressed as a conjunction of individual implications corresponding to the paths through the tree ending in Yes nodes.**

For example, the **path for a restaurant full of patrons, with an estimated wait of 10-30 minutes when the agent is not hungry** is expressed by the logical sentence

$$\forall r \quad Patrons(r, Full) \wedge WaitEstimate(r, 0\text{-}10) \wedge Hungry(r, N) \Rightarrow WillWait(r)$$

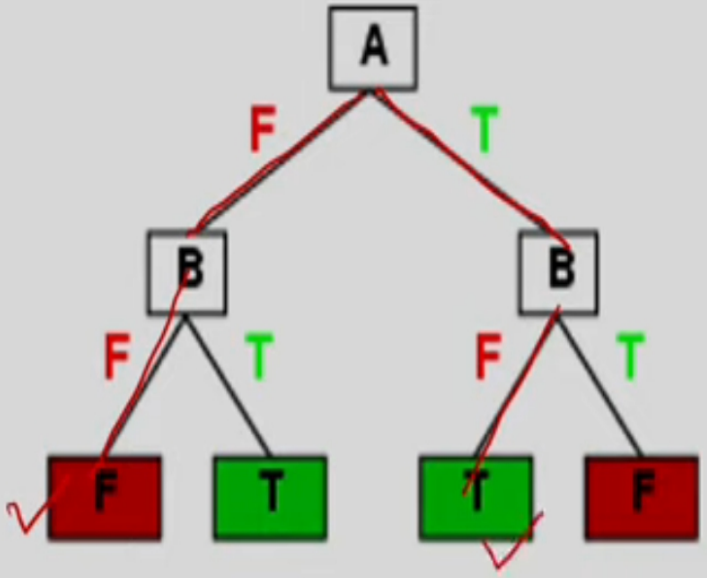
Expressiveness

- Decision Trees can be expressed as implication sentences
- In principle, they can **express propositional logic sentences**
  - Each row in the truth table of a sentence can be represented as a path in the tree
  - Often there are more efficient trees
- Some functions require exponentially large decision trees
  - Parity function, Majority function.

Expressiveness

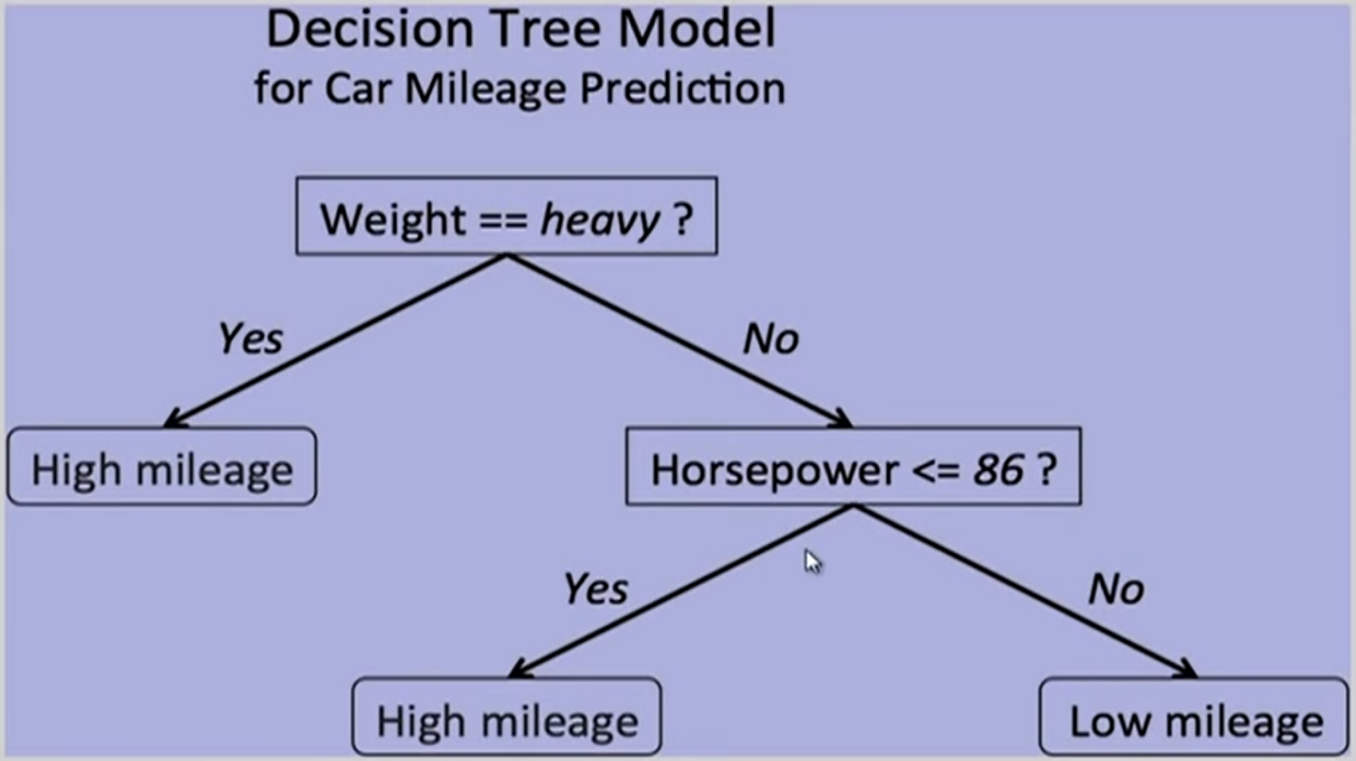
Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf.

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples.

Decision Tree Example 3

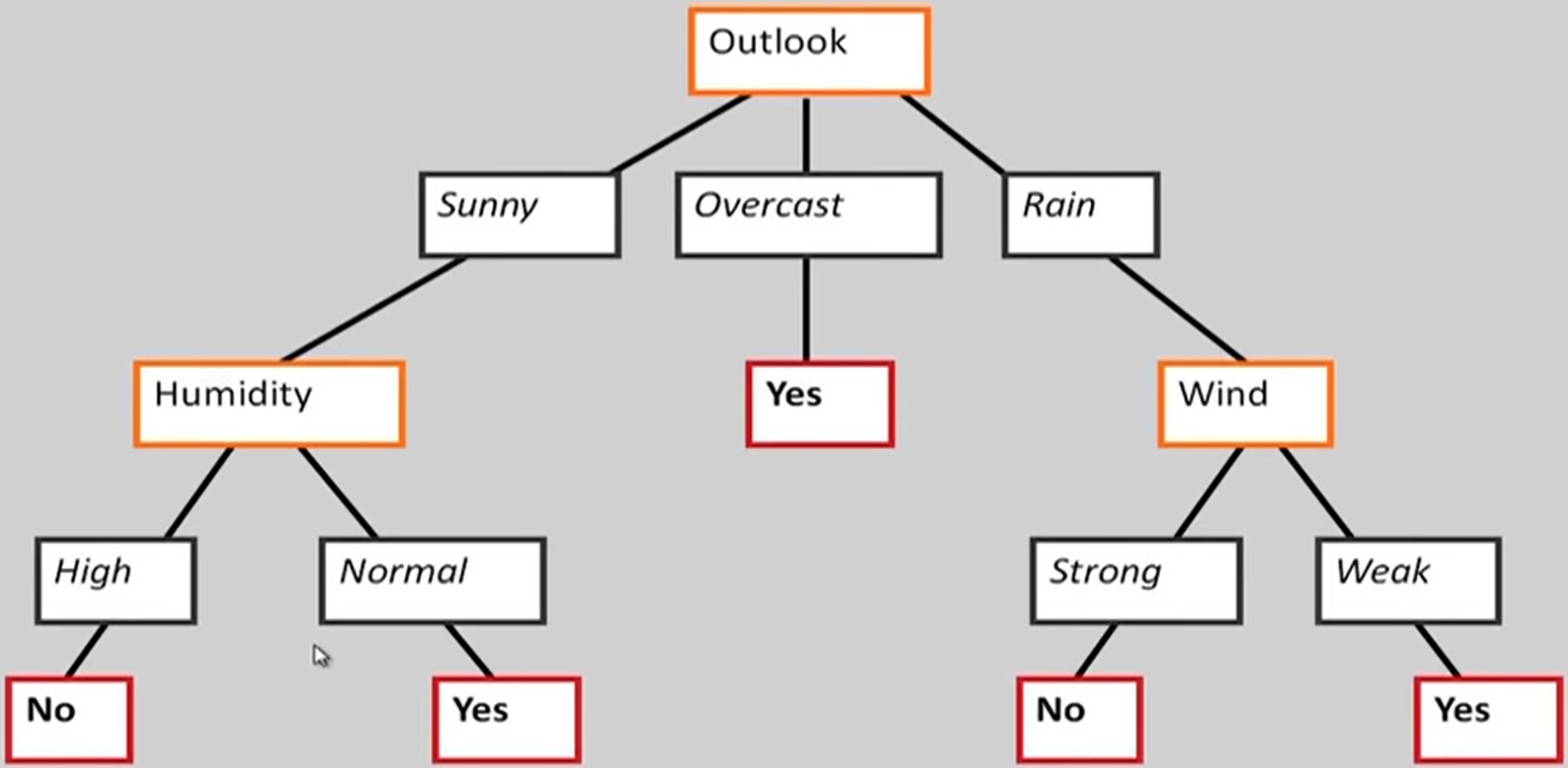


Example from Book(Tom Mitchell)

Decision Tree for PlayTennis

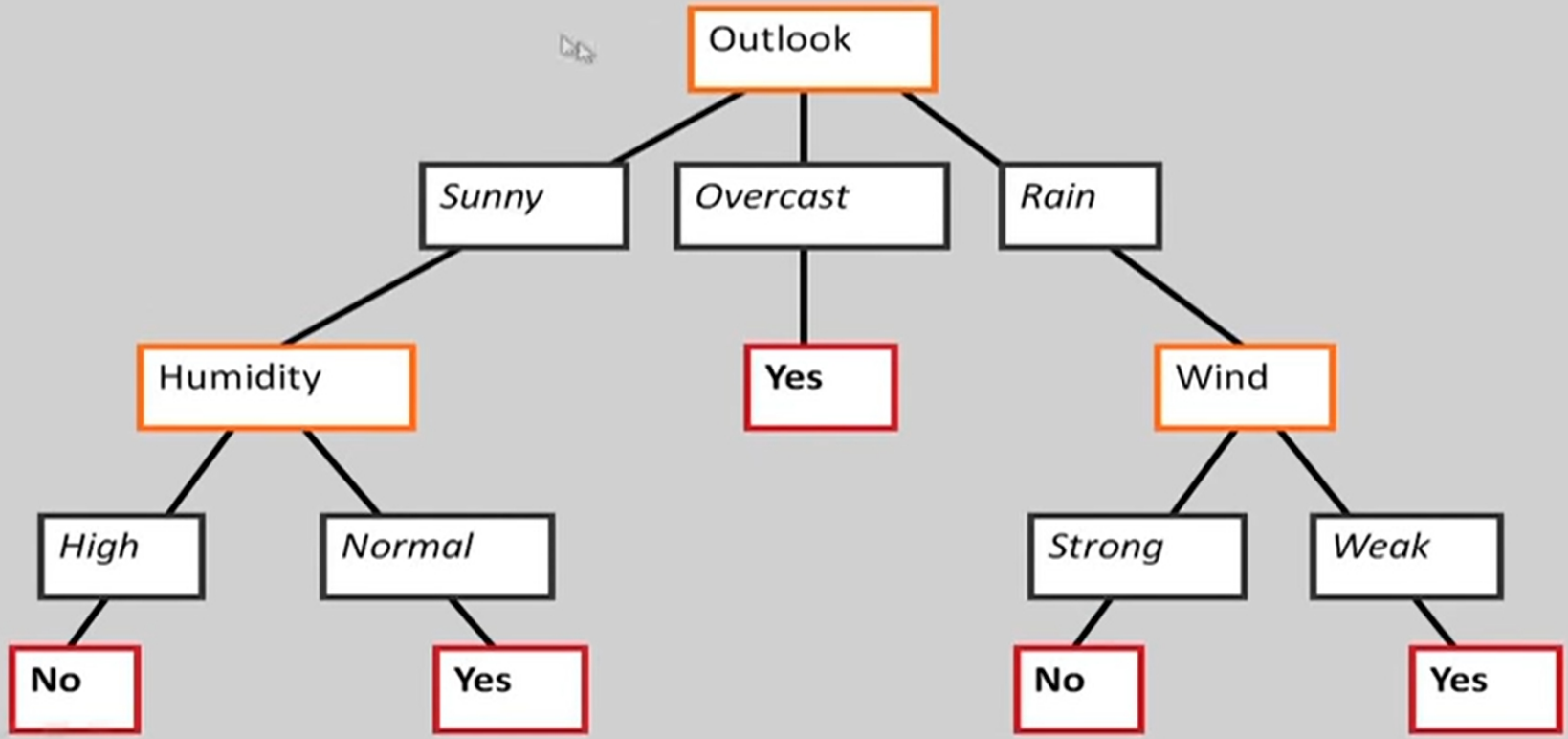
- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*
- Target concept - Play Tennis: *Yes, No*

Decision Tree for PlayTennis

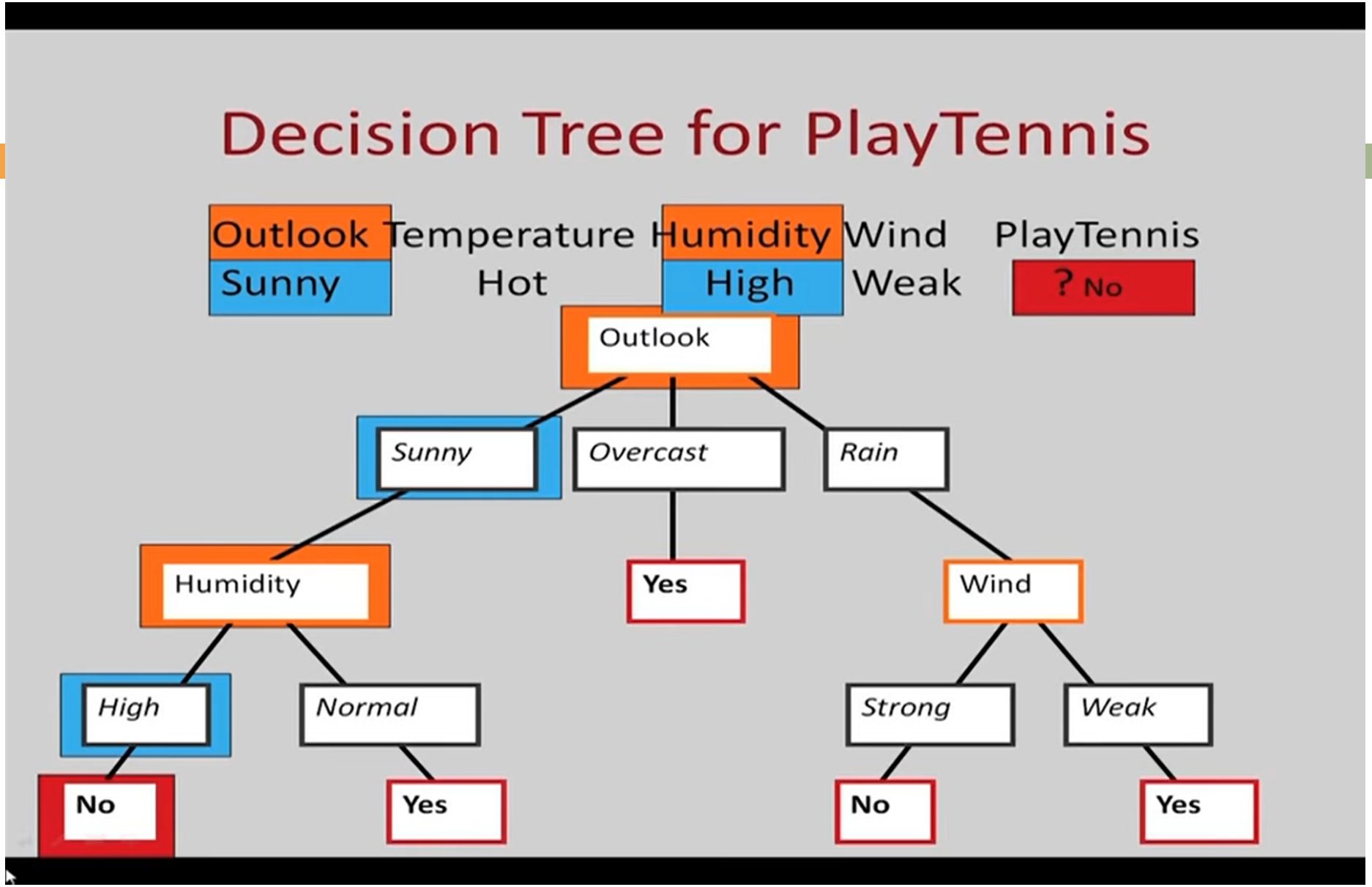


Decision Tree for PlayTennis

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	?

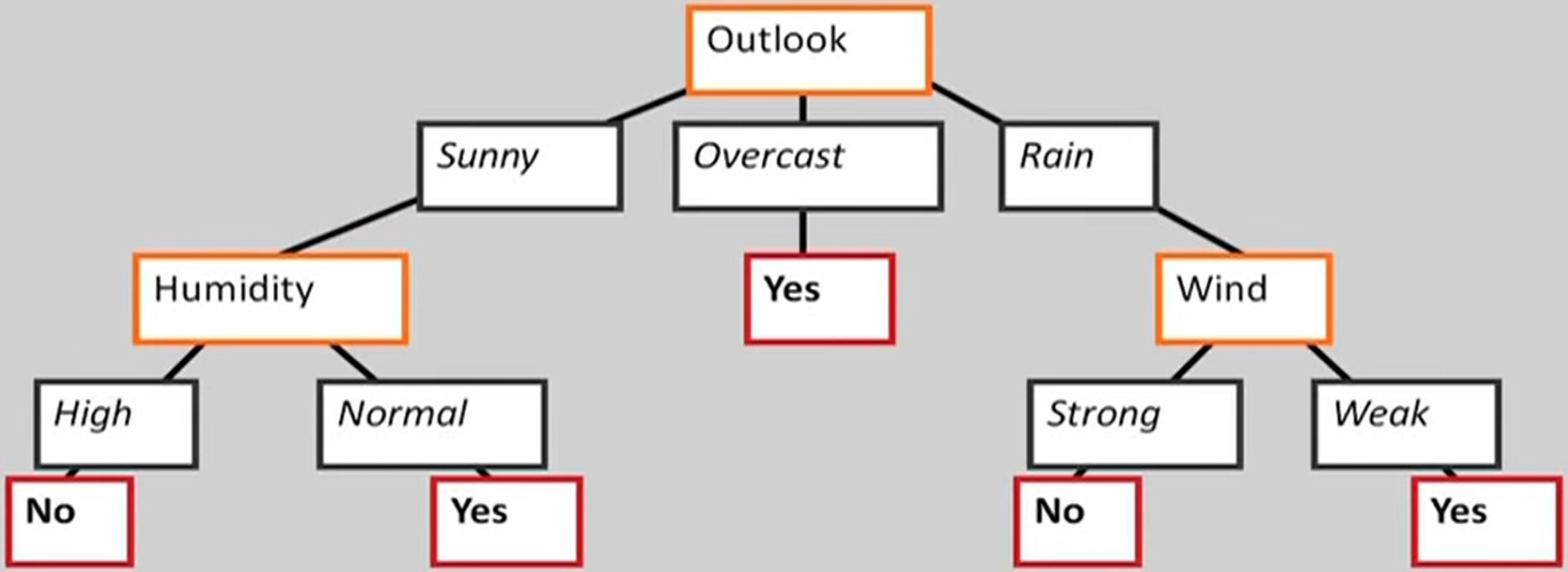


Decision Tree for PlayTennis



Decision Tree

decision trees represent disjunctions of conjunctions



(Outlook=Sunny ∧ Humidity=Normal)
∨ (Outlook=Overcast)
∨ (Outlook=Rain ∧ Wind=Weak)

Hw: Draw decision tree of following functions

Decision Tree for Boolean Functions

(a) $A \wedge \neg B$
(b) $A \vee [B \wedge C]$
(c) A XOR B
(d) $[A \wedge B] \vee [C \wedge D]$

Learning Decision Trees

- Problem: Find a decision tree that agrees with the training set
- Trivial solution: construct a tree with one branch for each sample of the training set
  - works perfectly for the samples in the training set
  - may not work well for new samples (generalization)
  - results in relatively large trees
- Better solution: find a concise tree that still agrees with all samples
  - corresponds to the simplest hypothesis that is consistent with the training set

Constructing Decision Trees

- In general, constructing the smallest possible decision tree is an intractable problem
- Algorithms exist for constructing reasonably small trees
- Basic idea: test the most important attribute first
  - Attribute that makes the most difference for the classification of an example.
    - Can be determined through information theory
  - Hopefully will yield the correct classification with few tests.

Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the smallest tree that is consistent with the data (Bias)

- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data

Searching for a good tree

- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values
- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

Top-Down Induction of Decision Trees ID3

1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split
- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

Decision Tree Learning

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

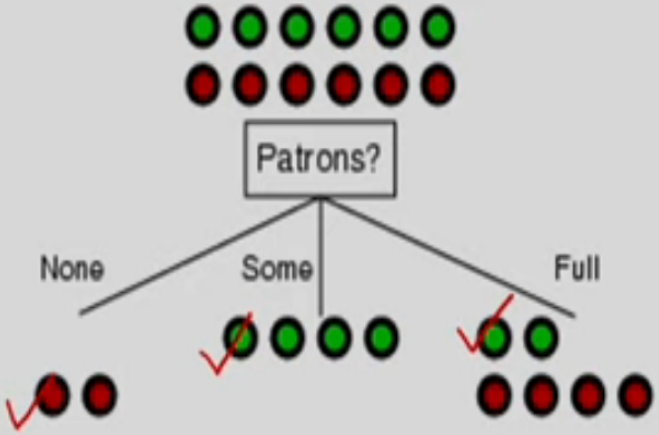
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

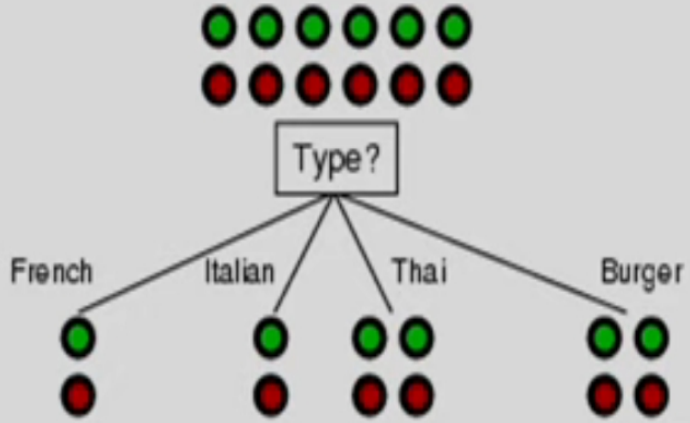
Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

Splitting the examples by testing on attributes. Patrons is a good attribute to test first!

Type is a poor attribute, because it leaves us with four possible outcomes, each of which has the same number of positive and negative answers.

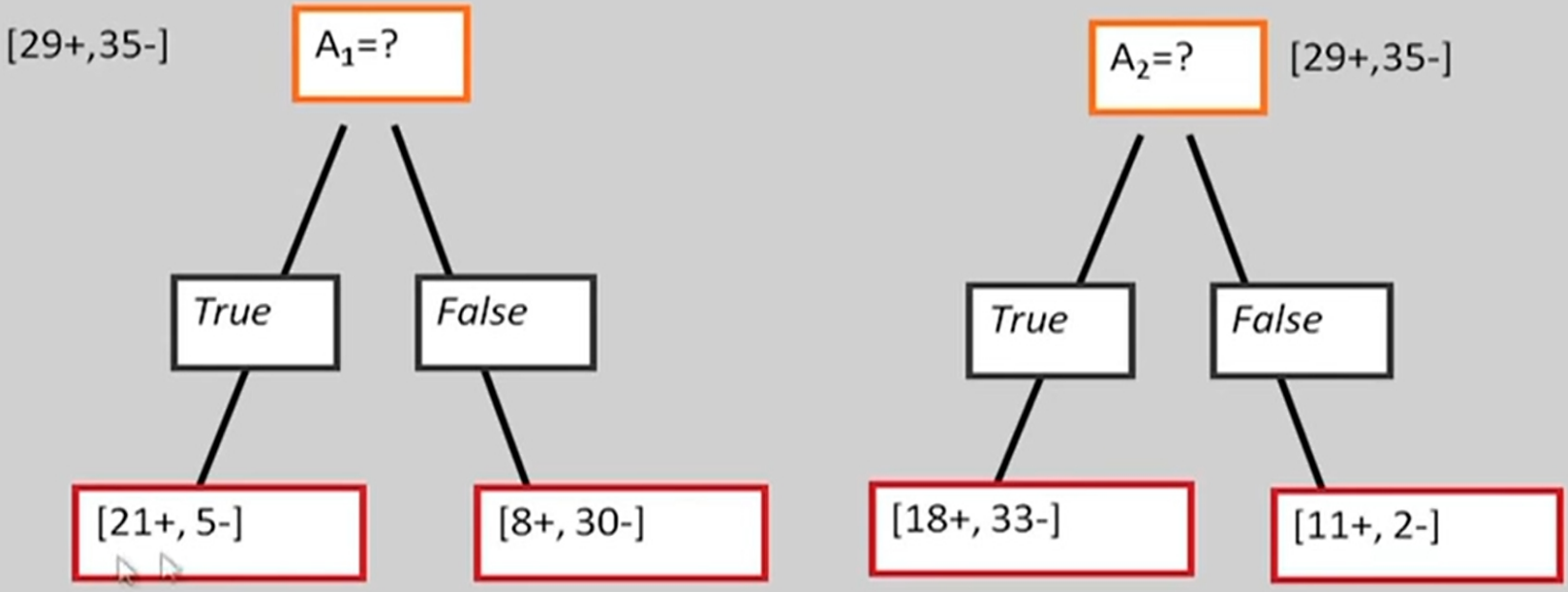




- *Patrons?* is a better choice

Patrons is a fairly important attribute, because if the value is None or Some, then we are left with example sets for which we can answer definitively (No and Yes, respectively). If the value is Full, we will need additional tests.

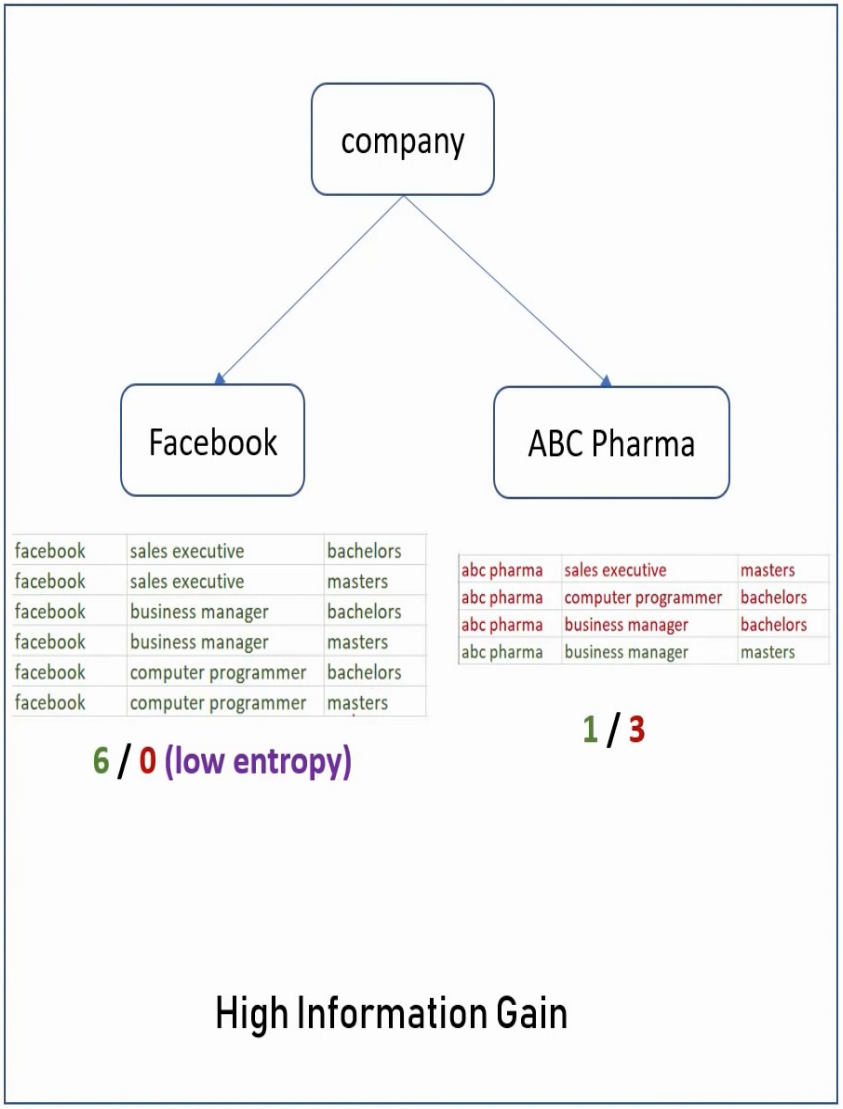
Which Attribute is "best"?

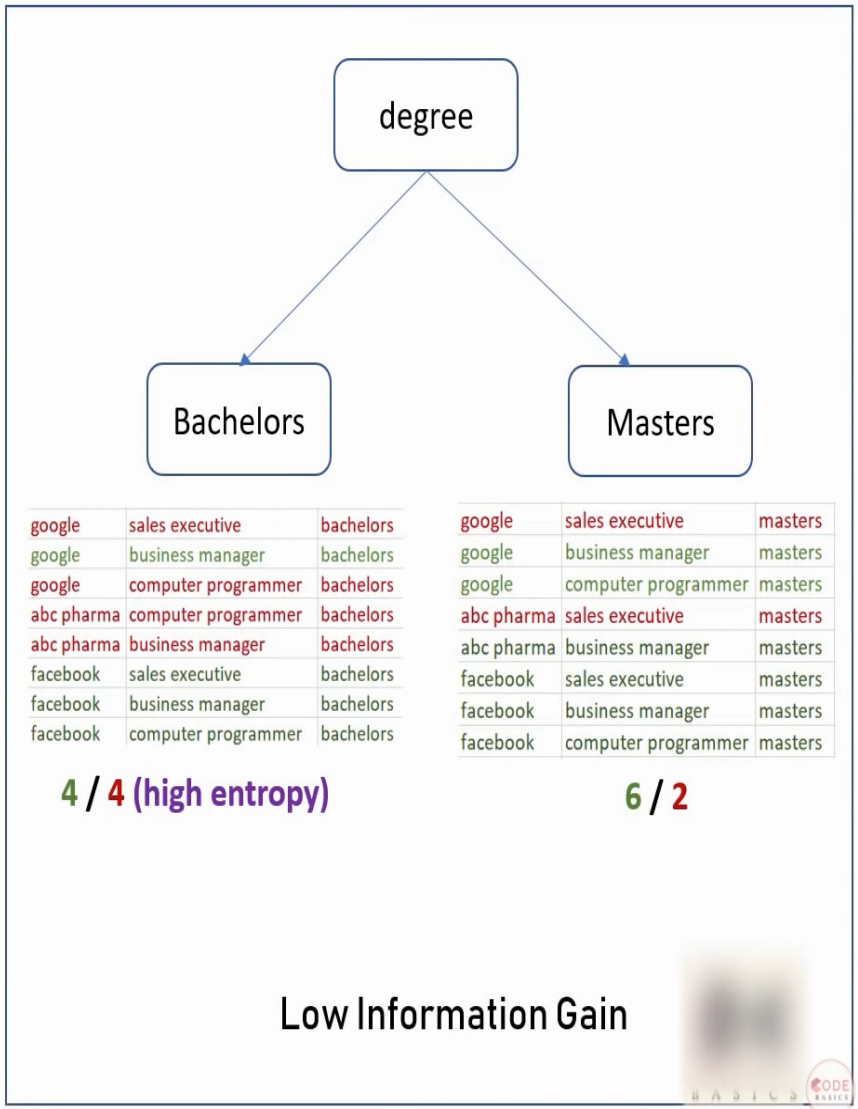


To choose which attribute is best, we use entropy and information gain.

Principled Criterion

- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)


company
Facebook
ABC Pharma
facebook sales executive bachelors
facebook sales executive masters
facebook business manager bachelors
facebook business manager masters
facebook computer programmer bachelors
facebook computer programmer masters
6 / 0 (low entropy)
abc pharma sales executive masters
abc pharma computer programmer bachelors
abc pharma business manager bachelors
abc pharma business manager masters
1 / 3
High Information Gain

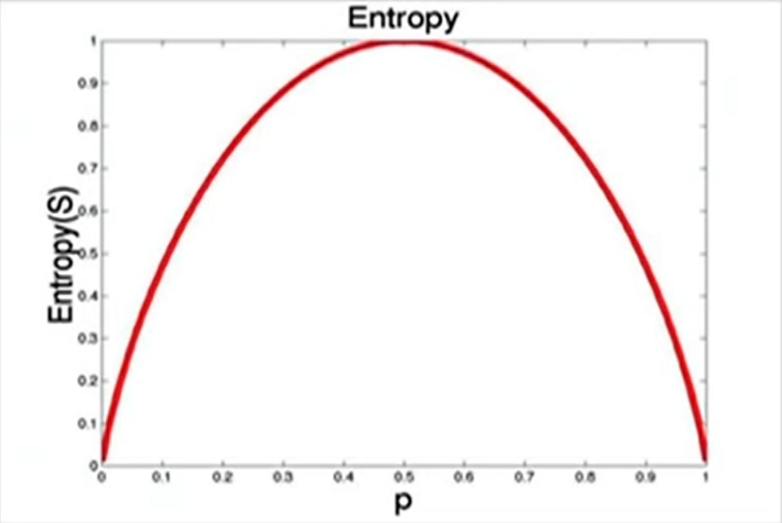

degree
Bachelors
Masters
google sales executive bachelors
google business manager bachelors
google computer programmer bachelors
abc pharma computer programmer bachelors
abc pharma business manager bachelors
facebook sales executive bachelors
facebook business manager bachelors
facebook computer programmer bachelors
4 / 4 (high entropy)
google sales executive masters
google business manager masters
google computer programmer masters
abc pharma sales executive masters
abc pharma business manager masters
facebook sales executive masters
facebook business manager masters
facebook computer programmer masters
6 / 2
Low Information Gain

Entropy

- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns $(-\log_2 p)$ bits to message having probability p
- S is a sample of training examples
  - p_+ is the proportion of positive examples in S
  - p_- is the proportion of negative examples in S
- Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

$$Entropy(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$

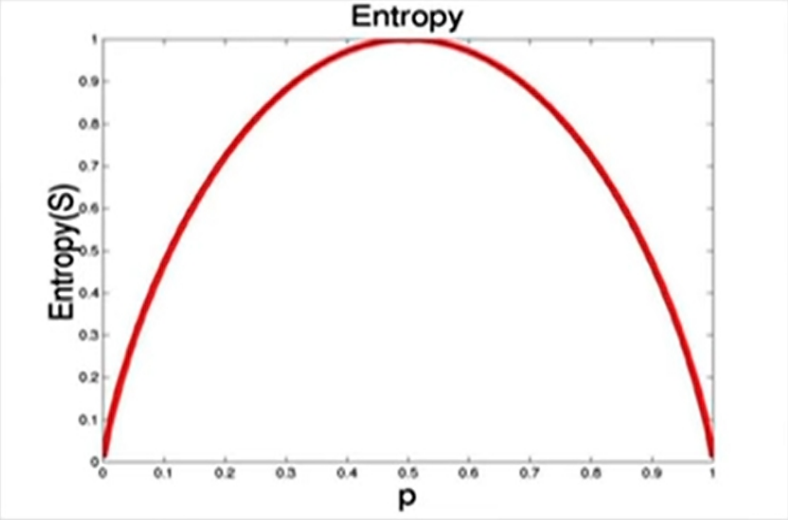
Entropy



- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy



- The entropy is 0 if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy

- An estimate of the probabilities of the possible answers before any of the attributes have been tested is given by the proportions of positive and negative examples in the training set.

- Suppose the training set contains p positive examples and n negative examples. Then an estimate of the information contained in a correct answer is

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Entropy

- For tossing of a *fair* coin the average information content is given as

$$I\left(\frac{1}{2},\frac{1}{2}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1 \text{ bit.}$$

If the coin is loaded to give 99% heads we get

$$I\left(\frac{1}{100},\frac{99}{100}\right) = -\frac{1}{100}\log_2\frac{1}{100} - \frac{99}{100}\log_2\frac{99}{100} = 0.08 \text{ bit.}$$

- As the probability of heads goes to 1, the information of the actual answer goes to 0.

Entropy

- The entropy is the average amount of information for a set of examples D

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1$$

$\Pr(c_j)$ is the probability of class c_j in data set D; We use entropy as a measure of impurity or disorder of data set D. (Or, a measure of information in a tree)

Information Gain

- How much information we still need after the attribute test?
- A chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values.

A random example has the ith value for the attribute with probability $(p_i+n_i)/(p+n)$.

$$\text{remainder}(A)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right)$$

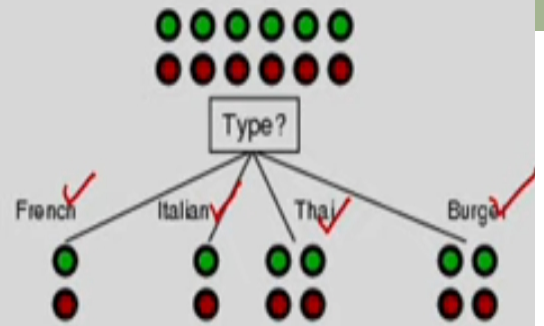
- **Information Gain** (IG) or reduction in entropy from the attribute test:

$$IG(A)=I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-\text{remainder}(A)$$

Information gain

For the training set,

$$I\left(\frac{6}{12},\frac{6}{12}\right)=1$$



Consider the attributes *Patrons* and *Type*; and others too.

$$IG(Patrons)=1-\left[\frac{2}{12}I(0,1)+\frac{4}{12}I(1,0)+\frac{6}{12}I\left(\frac{2}{6},\frac{4}{6}\right)\right]=.0541\text{ bits}$$

$$IG(Type)=1-\left[\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)+\frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)\right]=0\text{ bits}$$

Patrons has the highest IG of all attributes and so chosen by the DTL algorithm as the root.

Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$\text{Gain}(S,A)=\text{Entropy}(S) - \sum_{v \in \text{values}(A)} |S_v|/|S| \ \text{Entropy}(S_v)$$

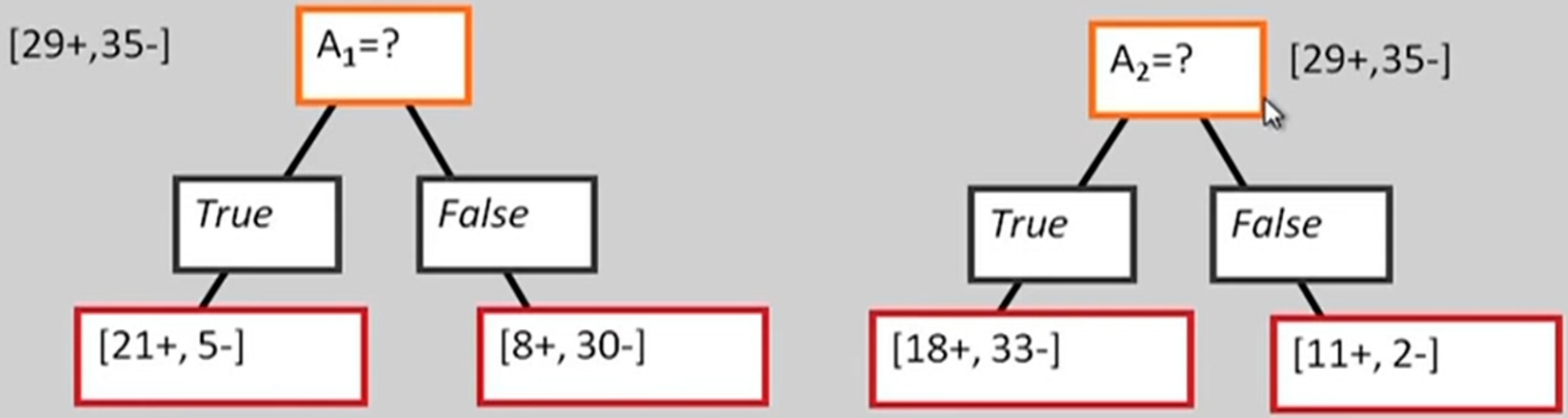
Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64$
= 0.99

Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$Gain(S,A)=Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

$$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$



Here Gain is higest for A1 among (A1, A2), so we will split A1.

Information Gain

Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

To choose which attribute is best, we use entropy and information gain.

Using Information Theory

- Implement Choose-Attribute in the DTL algorithm

We need a formal measure of "fairly good" and "really useless". The measure should have its maximum value when the attribute is perfect and its minimum value when the attribute is of no use at all . One suitable measure is the expected amount of information provided by the attribute, where we use the term in the mathematical sense from Information Theory.

- Information Content?

To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.

Information theory provides a mathematical basis for measuring the information content.

Let E be an event that occurs with probability $P(E)$.

If we are told that E has occurred, then we received

$$I(E) = \log_2 \frac{1}{P(E)}$$

Result of a fair coin flip provides 1 bit of information

bits of information.

Principled Criterion

- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

Entropy

- Interested in the information content of a source; rather than the information of any particular message. Information content is the average information per message. **Information content is also called entropy.**
  - Suppose we have an information source S which emits symbols from an alphabet $\{s_1, \ldots, s_k\}$ with probabilities $\{p_1, \ldots, p_k\}$ and each emission is independent of the others.
  - **Entropy is the average amount of information** we get when we observe a symbol emitted by S.

Entropy depends only on the probability distribution and not on the alphabet used by S.

$$I(S) = \sum_i p_i \log \frac{1}{p_i}$$

Example from Book(Tom Mitchell)

Decision Tree for PlayTennis

- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*
- Target concept - Play Tennis: *Yes, No*

ENTROPY AND INFORMATION GAIN

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ENTROPY AND INFORMATION GAIN

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable p.
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

ENTROPY AND INFORMATION GAIN

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *p*.
  - *S* is a collection of training examples
  - p_+ the proportion of positive examples in *S*
  - p_- the proportion of negative examples in *S*

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Examples

Entropy ([14+, 0–]) = – 14/14 log_2 (14/14) – 0 log_2 (0) = 0

Entropy ([9+, 5–]) = – 9/14 log_2 (9/14) – 5/14 log_2 (5/14) = 0.94

Entropy ([7+, 7–]) = – 7/14 log_2 (7/14) – 7/14 log_2 (7/14) = 1/2 + 1/2 = 1

ENTROPY AND INFORMATION GAIN

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

- Given entropy as a measure of the impurity in a collection of training examples, the ***information gain,*** is simply the expected reduction in entropy caused by partitioning the examples according to an attribute.
- More precisely, the information gain, ***Gain(S, A)*** of ***an*** attribute **A,** relative to a collection of examples ***S,*** is defined as,

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- where ***Values(A)*** is the set of all possible values for attribute A, and S_v, is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$)

ENTROPY AND INFORMATION GAIN

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$ → 9-NO, 5-yes

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v$$

$$= Entropy(S) - (8/14)Entropy(S_{Weak}) - (6/14)Entropy(S_{Strong})$$

$$= 0.940 - (8/14)0.811 - (6/14)1.00$$

$$= 0.048$$

The attribute having highest information gain is selected as the root node or for splitting

ENTROPY AND INFORMATION GAIN

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

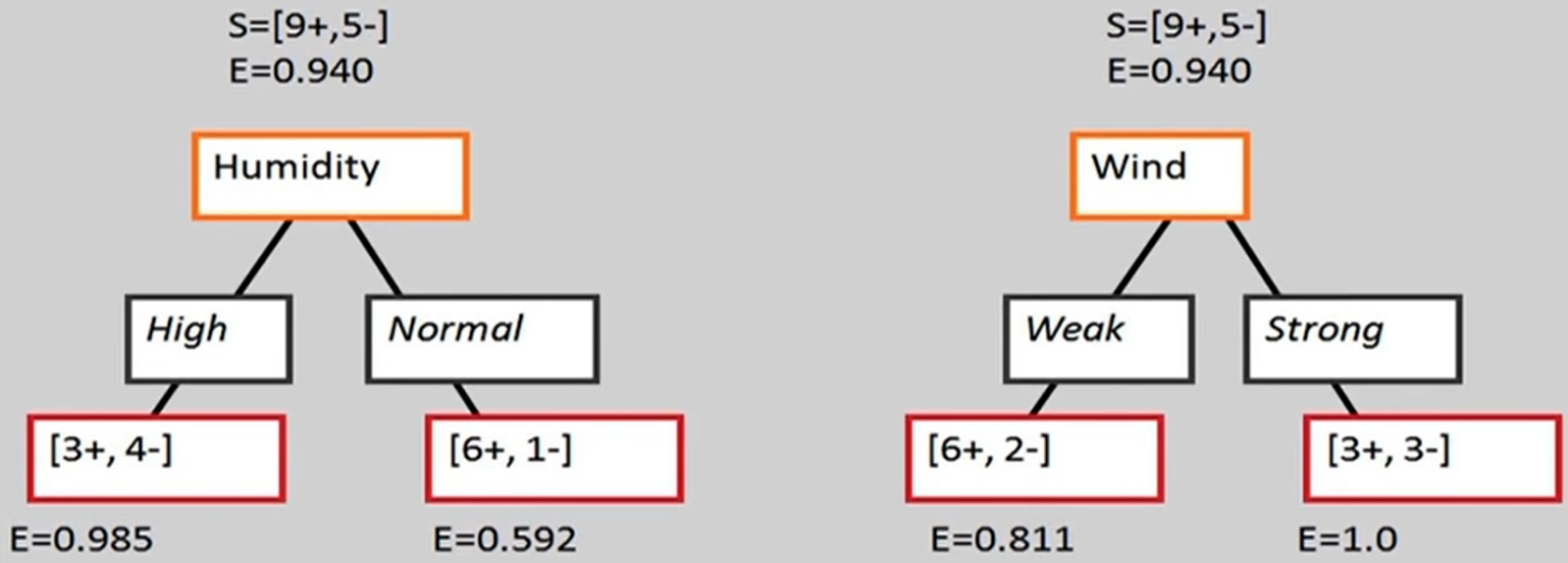
$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v$$

$$= Entropy(S) - (8/14)Entropy(S_{Weak}) - (6/14)Entropy(S_{Strong})$$

$$= 0.940 - (8/14)0.811 - (6/14)1.00$$

$$= 0.048$$

Selecting the Next Attribute



Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute

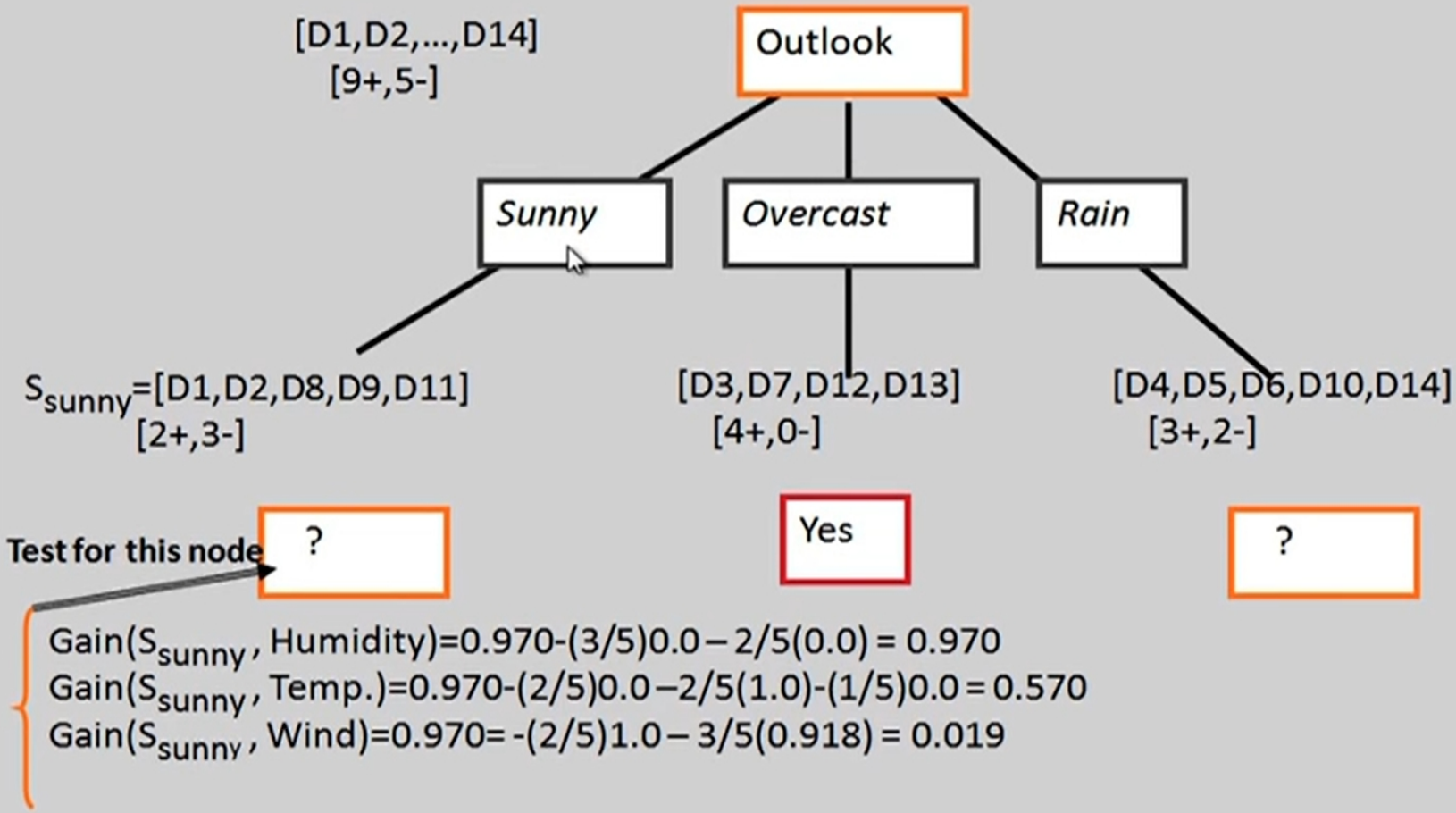
The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

Note: $0\text{Log}_2 0 = 0$

ID3 Algorithm



Continuous Attribute – Binary Split

- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals
  - Create a new boolean attribute A_c , looking for a threshold c,

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

How to choose c ?

- consider all possible splits and finds the best cut

Splitting Rule: GINI Index

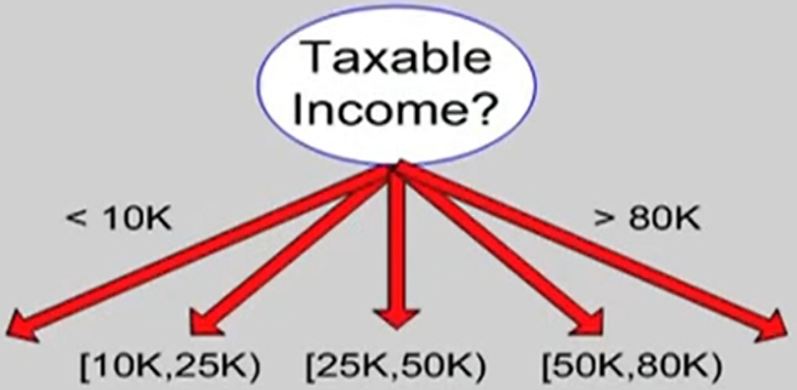
- GINI Index
  - Measure of node impurity

$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

Splitting Based on Continuous Attributes

(i) Binary split



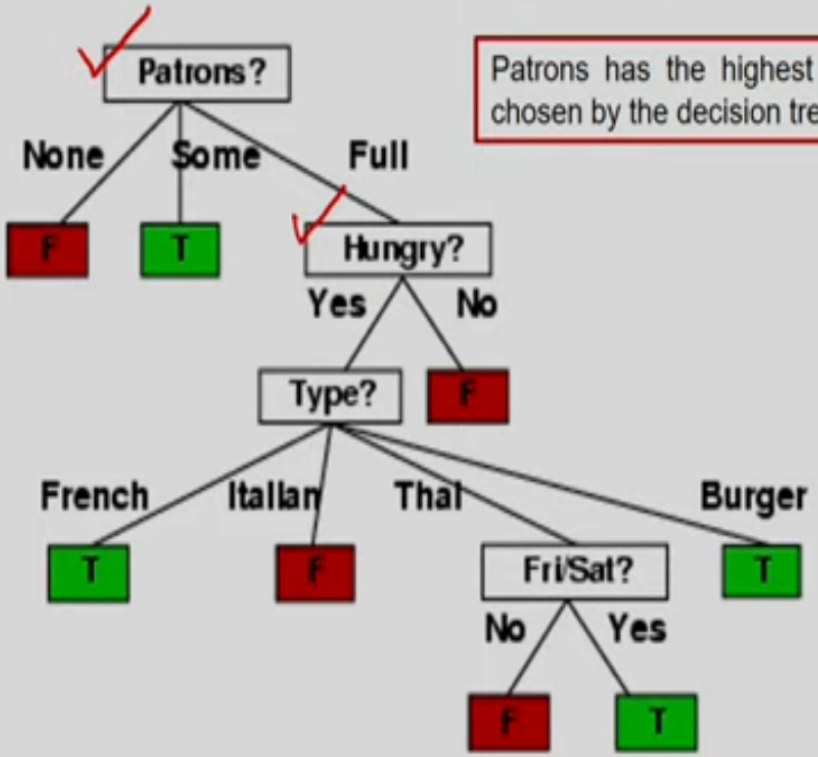
(ii) Multi-way split

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

The Decision Tree Induced

- Decision tree learned from the 12 example-training set.



Patrons has the highest gain of any of the attributes and chosen by the decision tree learning algorithm as the root.

- Substantially simpler than "true" tree - a more complex hypothesis isn't justified by small amount of data.

Performance of Decision Tree Learning

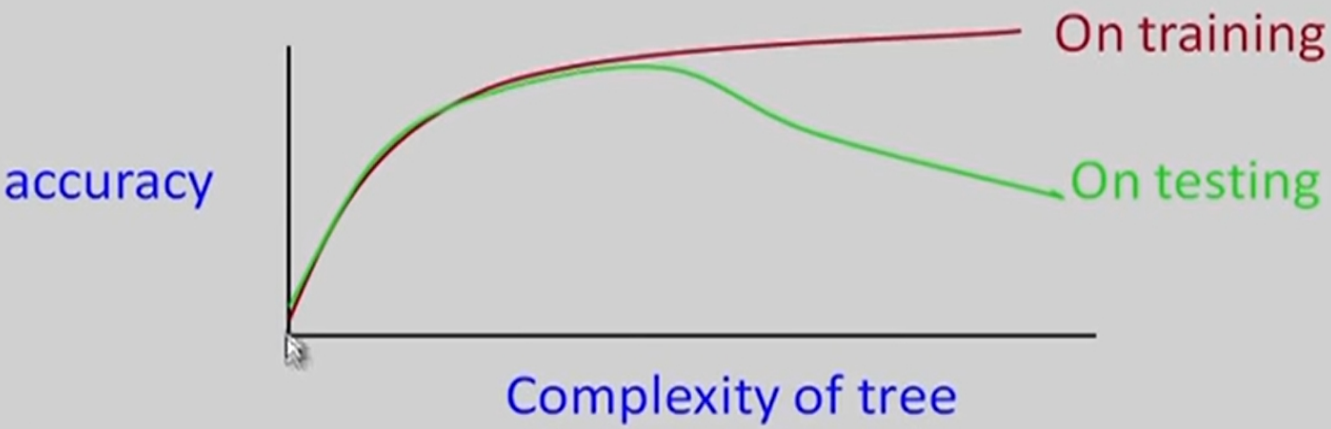
- Quality of Predictions
  - Predictions for the classification of unknown examples that agree with the correct result are obviously better.
  - Can be measured easily.
  - It can be assessed in advance by splitting the available examples into a training set and a test set
    - learn the training set, and assess the performance via the test set
- Size of the tree
  - A smaller tree (especially depth-wise) is a more concise representation

Noise and Overfitting

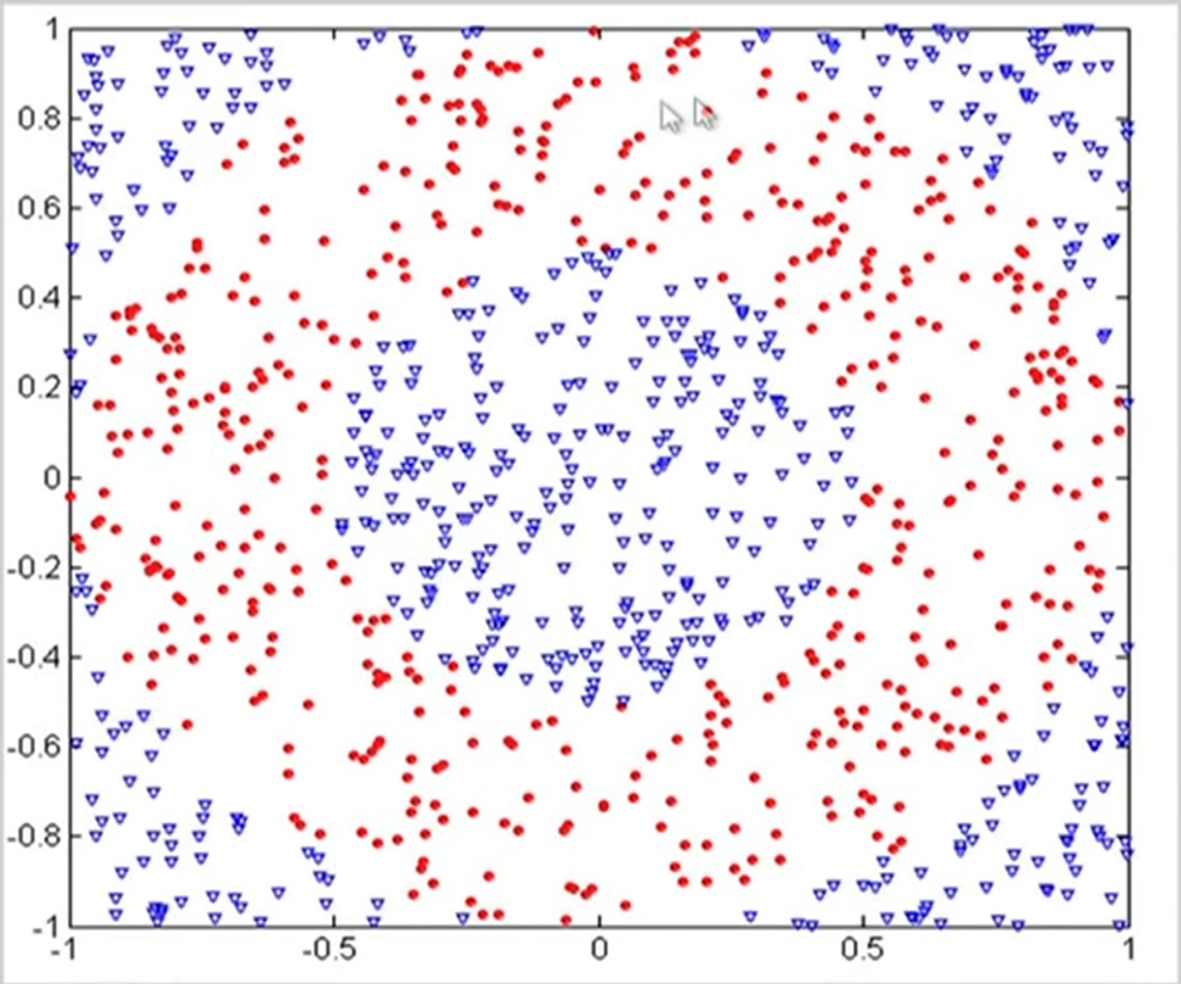
- The presence of irrelevant attributes ("noise") may lead to more degrees of freedom in the decision tree
  - The hypothesis space is unnecessarily large
- *Overfitting* makes use of irrelevant attributes to distinguish between samples that have no meaningful differences
  - E.g. using the day of the week when rolling dice
  - Overfitting is a general problem for all learning algorithms
- *Decision Tree Pruning* identifies attributes that are likely to be irrelevant
  - Very low information gain

Overfitting

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data
  - May be based on insufficient data
- A hypothesis *h* is said to overfit the training data if there is another hypothesis, h', such that *h* has smaller error than *h*' on the training data but *h* has larger error on the test data than *h*'.



Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

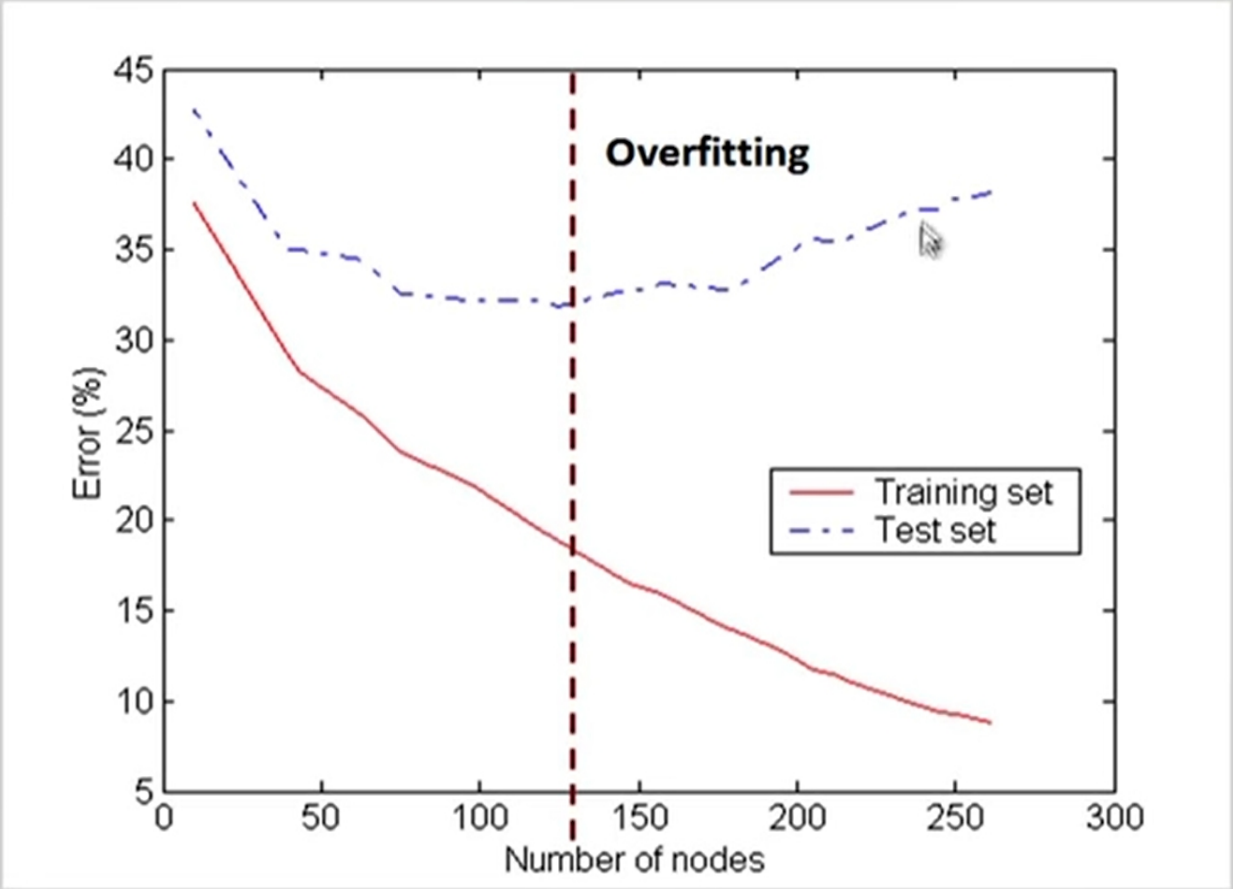
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes

- Methods for evaluating subtrees to prune:
  - Minimum description length (MDL):

  Minimize: size(tree) + size(misclassifications(tree))

  - Cross-validation

Pre-Pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Reduced-error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "grow" set and "validation" set.
  - Build a complete tree for the "grow" data
  - Until accuracy on validation set decreases, do:

    For each non-leaf node in the tree

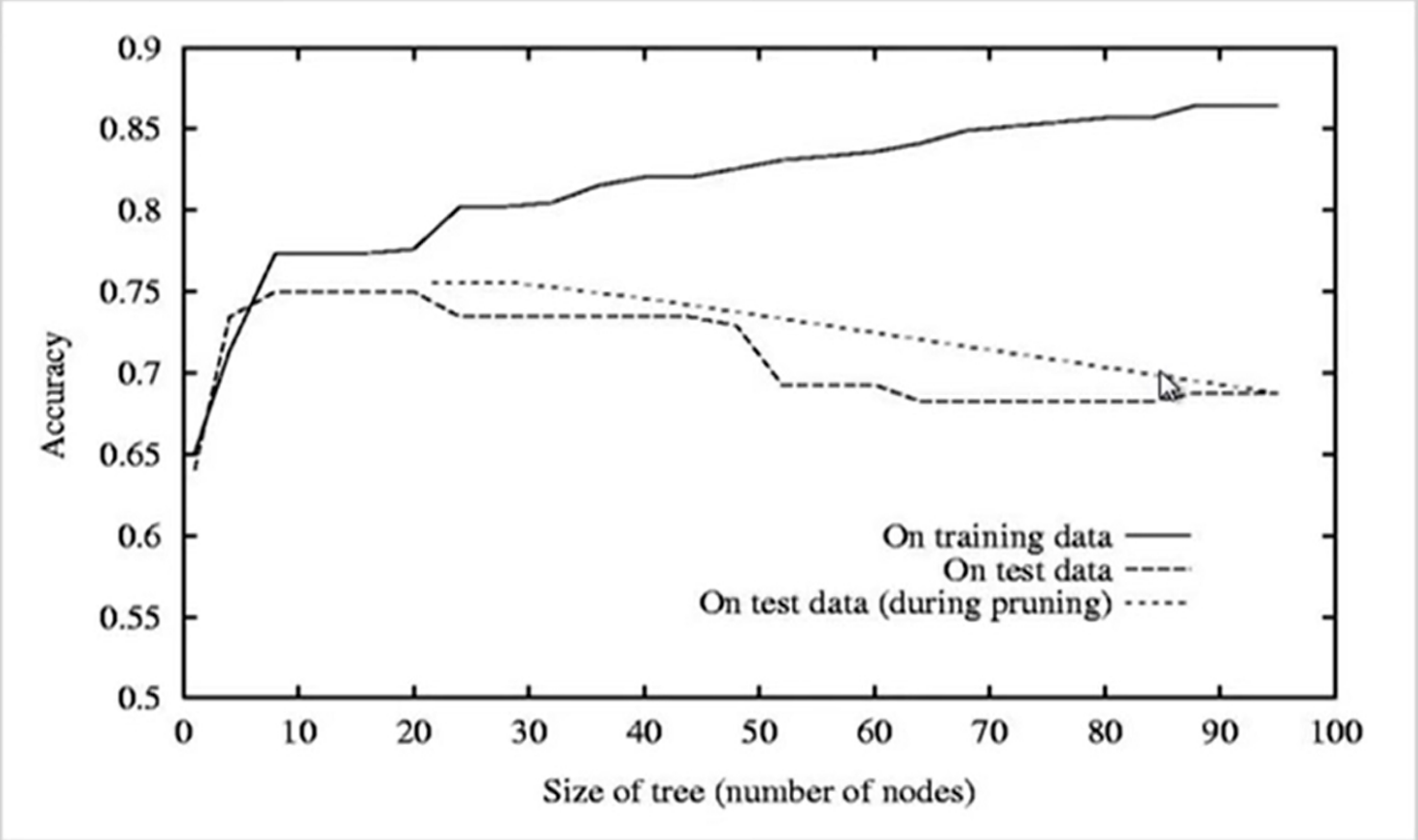
    Temporarily prune the tree below; replace it by majority vote

    Test the accuracy of the hypothesis on the validation set

    Permanently prune the node with the greatest increase in accuracy on the validation test.
- Problem: Uses less data to construct the tree
- Sometimes done at the rules level

General Strategy: Overfit and Simplify

Reduced Error Pruning



Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than C or f
- Underfitting: H less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors:
  - Complexity of H, *c* (H),
  - Training set size, *N*,
  - Generalization error, *E* on new data

- As *N* *increases*-, *E* decreases
- As *c* (H) *increases*-, first *E* *decreases* and then *E*- *increases*
- As *c* (H)- *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

Notes on Overfitting

- **overfitting** happens when a model is capturing idiosyncrasies of the data rather than generalities.
  - Often caused by too many parameters relative to the amount of training data.
  - E.g. an order-*N* polynomial can intersect any *N+1* data points

Dealing with Overfitting

- Use more data
- Use a tuning set
- **Regularization**
- Be a Bayesian

M= degree of polynomial

Regularization

- In a linear regression model overfitting is characterized by large weights.

	M = 0	M = 1	M = 3	M = 9
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Penalize large weights in Linear Regression

- Introduce a penalty term in the loss function.

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}\{t_n - y(x_n, \vec{w})\}^2$$

Regularized Regression

1. (L2-Regularization or Ridge Regression)

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \frac{\lambda}{2}\|\vec{w}\|^2$$

1. L1-Regularization

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \lambda|\vec{w}|_1$$

Broadening the Applicability

- Extend decision tree induction to a wider variety of problems; a number of issues must be addressed:

1. Missing data

   In many domains, not all the attribute values will be known for every example. The values may not have been recorded, or they may be too expensive to obtain.

2. Multivalued Attributes

   When an attribute has a large number of possible values, the information gain measure gives an inappropriate indication of the attribute's usefulness.

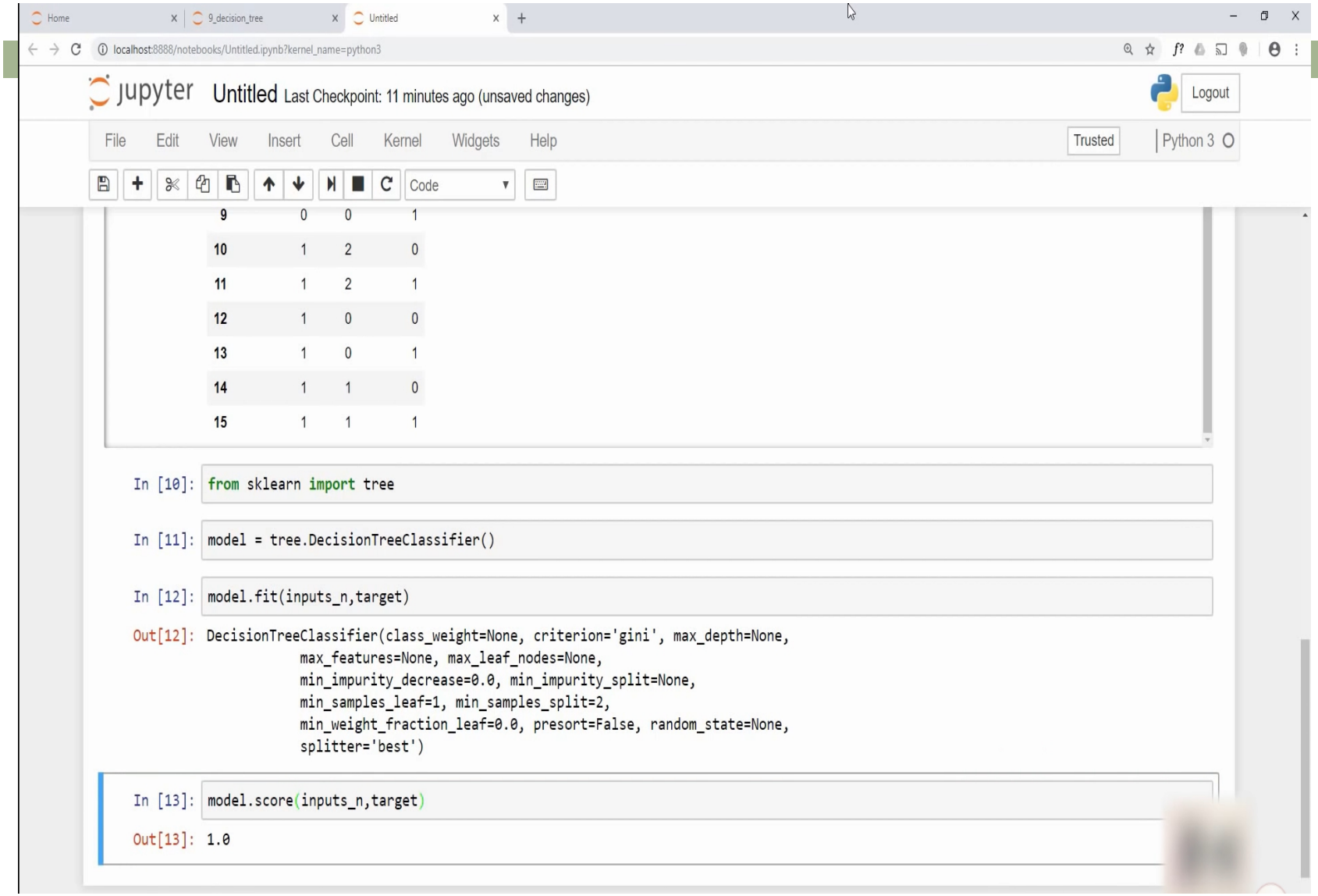
3. Continuous-valued Attributes

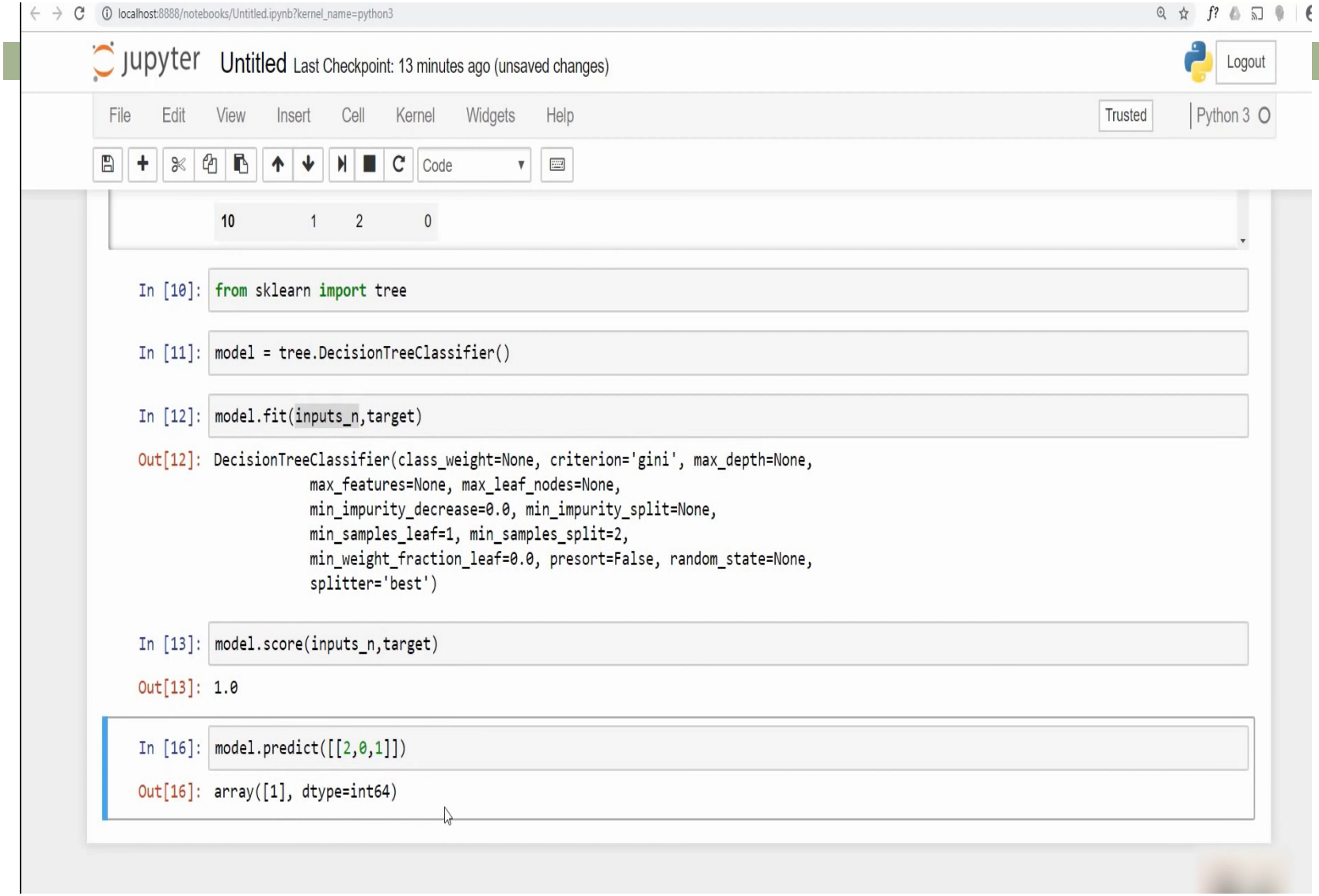
   Certain attributes (such as Height; Weight) have a large or infinite set of possible values. They are therefore not well-suited for decision-tree learning in raw form.

A decision-tree learning system for real-world applications must be able to handle all of these problems. **Handling continuous-valued variables is especially important** - physical and financial processes provide numerical data. Several commercial packages been built that meet these criteria, and they have been used to develop several hundred fielded systems.

Company	Job	Degree	Salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0
google	computer programmer	masters	1
abc pharma	sales executive	masters	0
abc pharma	computer programmer	bachelors	0
abc pharma	business manager	bachelors	0
abc pharma	business manager	masters	1
facebook	sales executive	bachelors	1
facebook	sales executive	masters	1
facebook	business manager	bachelors	1
facebook	business manager	masters	1
facebook	computer programmer	bachelors	1
facebook	computer programmer	masters	1

Google-facebook salary prediction




localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Code
10 1 2 0
In [10]: from sklearn import tree
In [11]: model = tree.DecisionTreeClassifier()
In [12]: model.fit(inputs_n,target)
Out[12]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=None,
splitter='best')
In [13]: model.score(inputs_n,target)
Out[13]: 1.0
In [16]: model.predict([[2,0,1]])
Out[16]: array([1], dtype=int64)