THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 04: Decision Trees and the Curious Case of Oxide Growth

Sergei V. Kalinin

Learning programming:

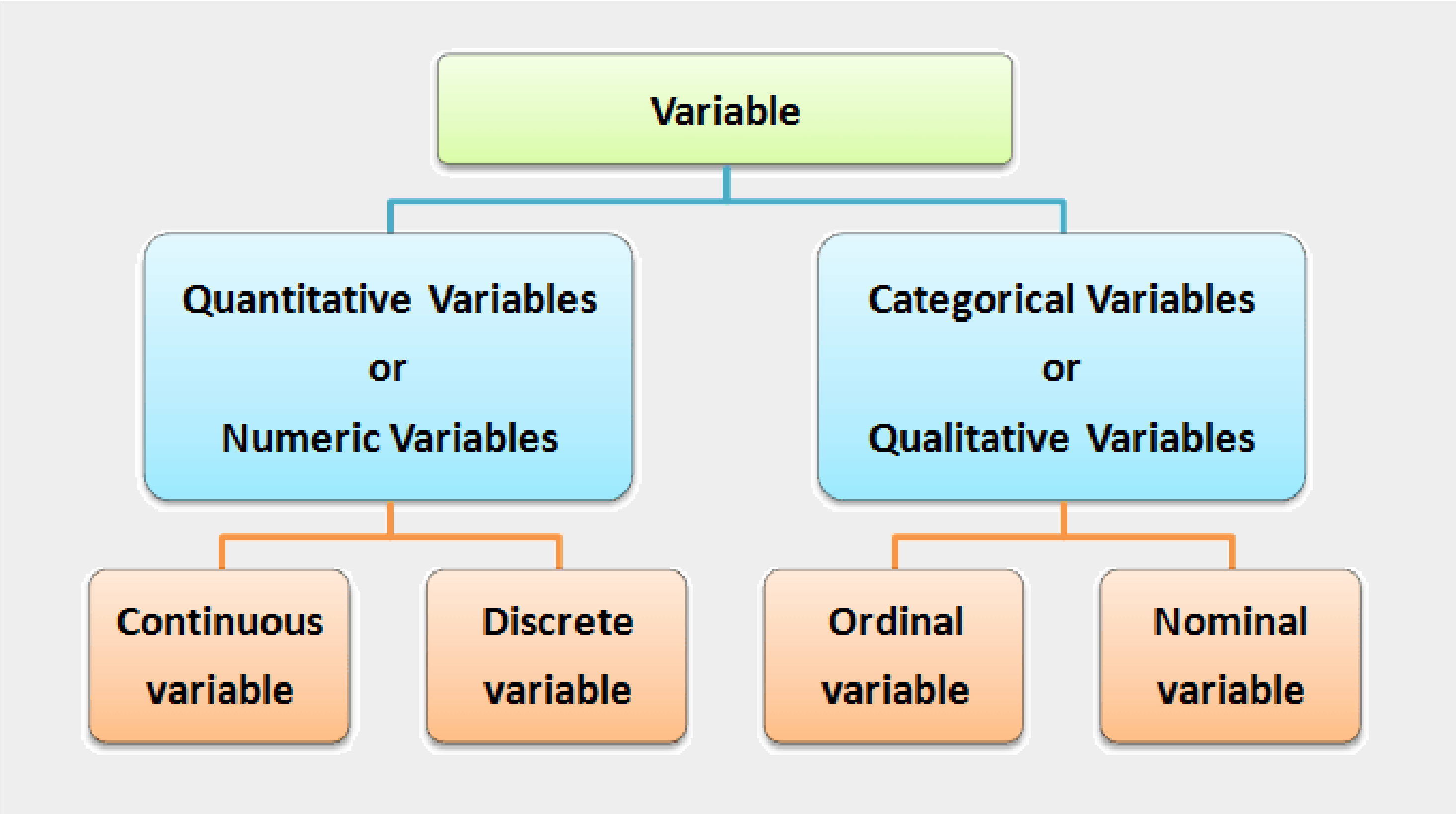
1. Learn to program per se
2. Writing code used in project
3. Making code that others will use
4. Using codes written by others
5. Being a part of the team developing code
6. Leading the team developing code

We have got the data… Now what?

- Classification is the process of identifying and grouping objects or ideas into predetermined categories.
- In data management, classification enables the separation and sorting of data according to set requirements for various business or personal objectives.
- In machine learning (ML), classification is used in predictive modeling to assign input data with a class label.

https://www.techopedia.com/definition/13779/classification

Type of variables



Nominal variable: no order

Colors: blue, red, yellow, white

Dog breed: Labrador, German shepherd, poodle

Ordinal variable: there is order

Letter grades: A, B, C, …

Severity of the software bug: mild, serious, critical

https://prinsli.com/categorical-variables/

Classification Workflow

1. Selecting features and collecting labeled training examples
2. Choosing a performance metric
3. Choosing a learning algorithm and training a model
    - Can we understand how it works?
    - Does it have any hyperparameters
    - How well does it generalize to new data?
    - How expensive is it?
4. Evaluating the performance of the model
5. Changing the settings of the algorithm and tuning the model.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

There are many classifiers:

Choosing an appropriate classification algorithm for a particular problem task requires practice and experience; each algorithm has its own quirks and is based on certain assumptions.

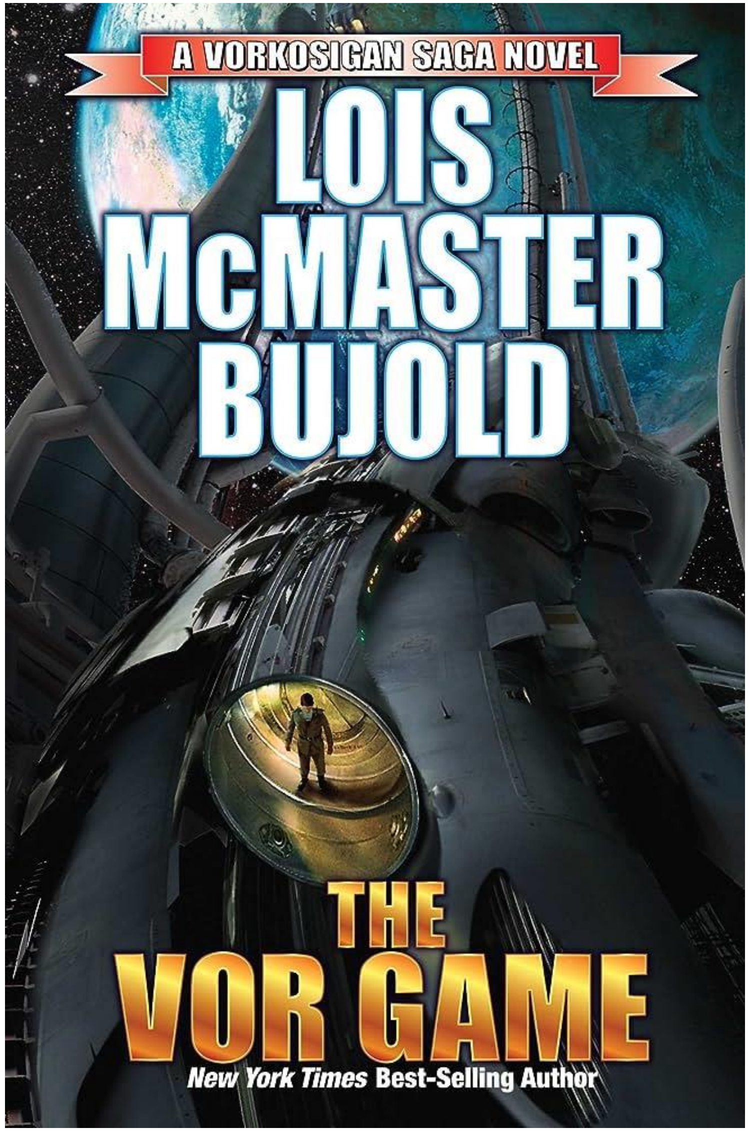
To paraphrase the **no free lunch theorem** by David H. Wolpert, no single classifier works best across all possible scenarios (*The Lack of A Priori Distinctions Between Learning Algorithms*, *Wolpert, David H*, *Neural Computation 8.7* (1996): 1341-1390).

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn


Hello, I am Akinator
Think about a real or fictional character.
I will try to guess who it is
akinator®
Inactive
Sensitive content
Download on the App Store
PLAY
GET IT ON Google Play
824 people are playing right now.
631563588 games played 30599 today.
Famous sportsmen Akinator can guess

Playing a classifier game:

1. Is your character a pokemon: No
2. Does your character have legs: Yes
3. Is your character a girl: No
4. Is your character famous youtuber: No
5. Is your character real: No
6. Does your character have human head: Yes
7. Is your character from Japanese anime: No
8. Wear a mask: No
9. Originally from video game: No
10. Animated: No
11. Played in superhero movie: No
12. From a book: Yes
13. Have been in movie: No
14. Have powers: No
15. Popular tv show: No
16. Have phone: yes
17. Adult: Yes
18. Killed people: Yes
19. Live in future: Yes
20. Been in space: Yes
21. Wife dead: No
22. Linked with metal suit: No
23. Short: Yes
24. Father famous: Yes



Answer: Miles Vorkosigan

Tennis Player Example

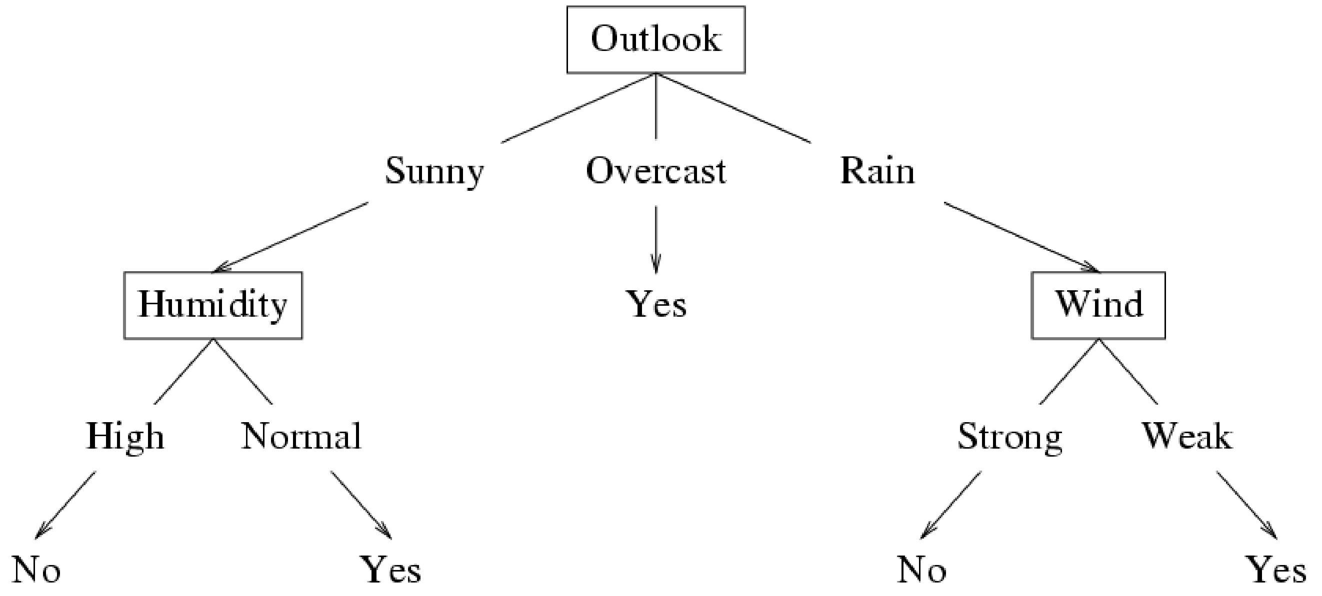
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.

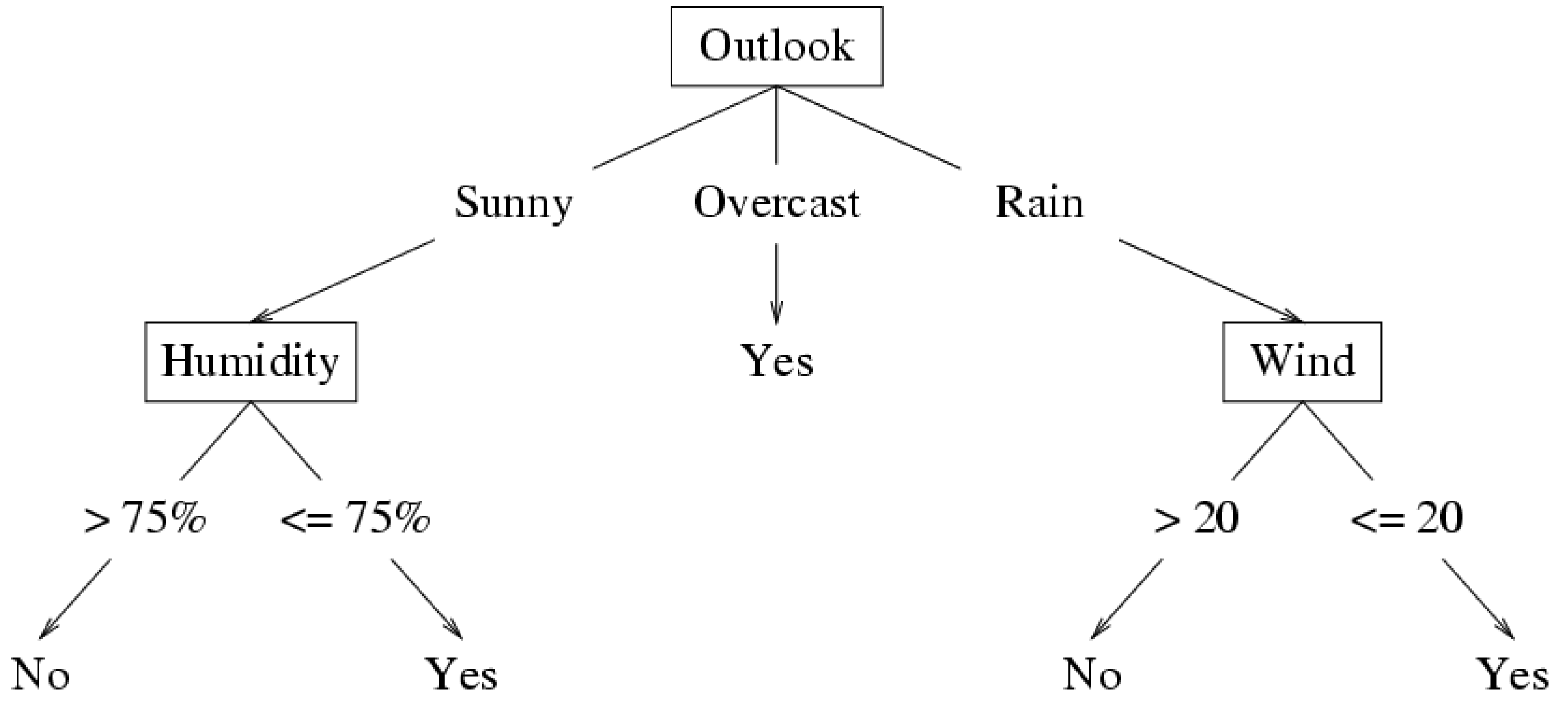


Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

What if we have continuous variables?

If the features are continuous, internal nodes may test the value of a feature against a threshold.

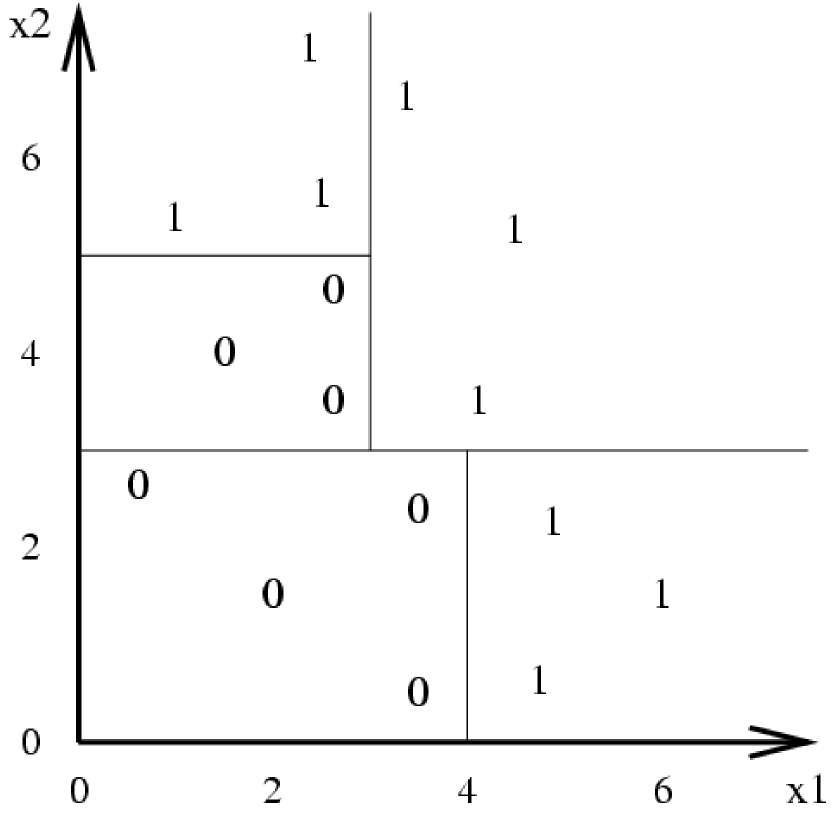


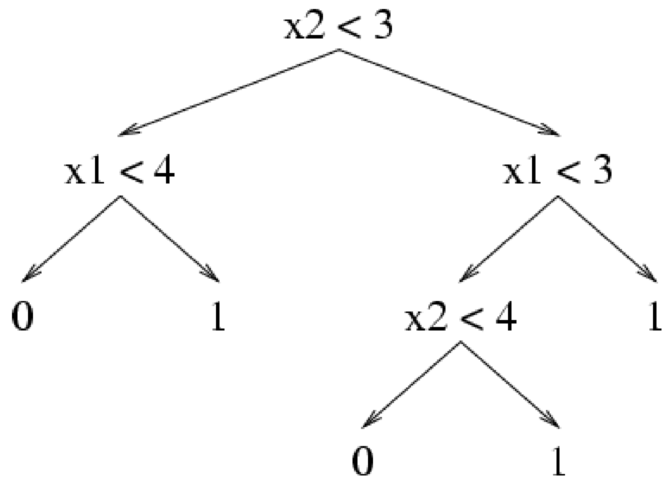
Also: regression trees

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Decision Tree Boundaries

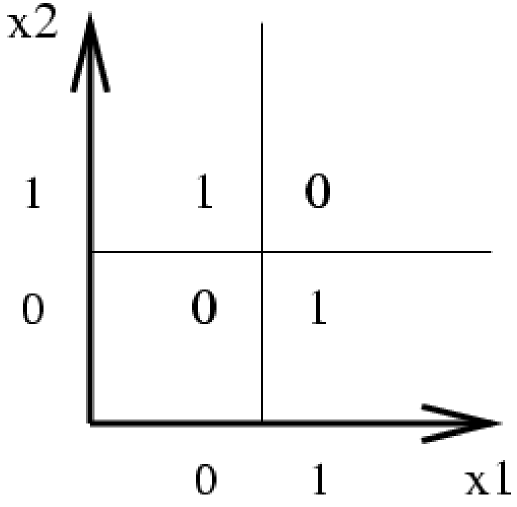
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

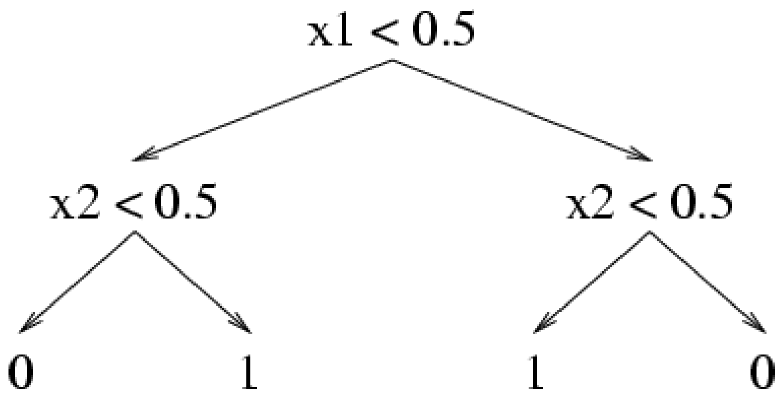




Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Can Represent Any Boolean Function





The tree will in the worst case require exponentially many nodes, however.

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Classification and Regression Tree (CART)

- Create a set of questions that consists of all possible questions about the measured variables

- Select a splitting criterion (likelihood).
  - Initialization: create a tree with one node containing all the training data.
  - Splitting: find the best question for splitting each terminal node. Split the one terminal node that results in the greatest increase in the likelihood.
  - Stopping: if each leaf node contains data samples from the same class, or some pre-set threshold is not satisfied, stop. Otherwise, continue splitting.
  - Pruning: use an independent test set or cross-validation to prune the tree.

- There are ways to estimate and incorporate priors into the decision tree (though these methods somewhat predate Bayesian methods).

- Computational complexity is very low for both evaluation and training.

How do we split?

Information gain:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- f is the feature to perform the split
- D_p and D_j are the dataset of the parent and jth child node
- I is our **impurity** measure
- N_p is the total number of training examples at the parent node
- N_j is the number of examples in the jth child node

Binary split:

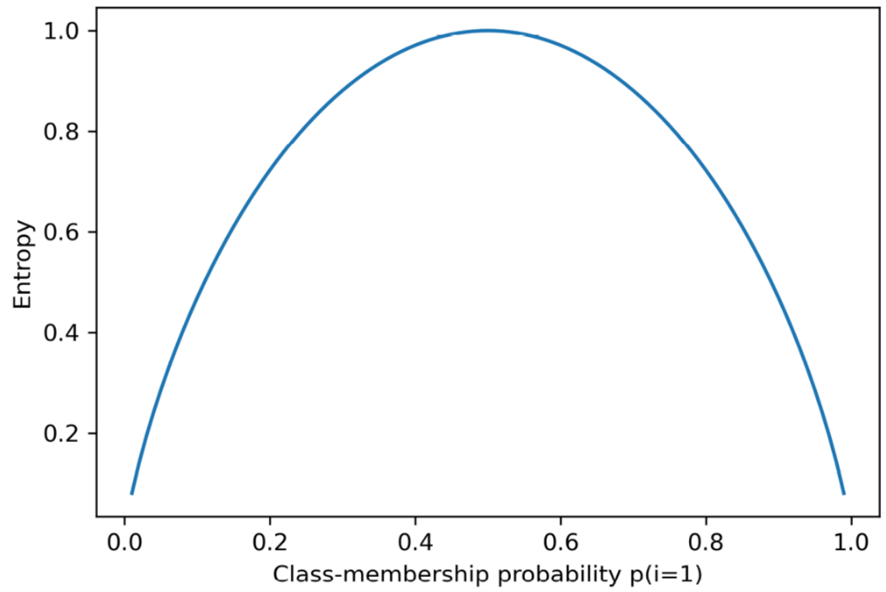
$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

What are impurity measures?

Entropy:

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$



- $p(i|t)$ is the proportion of the examples that belong to class i for a particular node, t.
- The entropy is 0 if all examples at a node belong to the same class, and entropy is maximal if we have a uniform class distribution.
- For binary class setting, the entropy is 0 if $p(i=1|t) = 1$ or $p(i=0|t) = 0$. If the classes are distributed uniformly with $p(i=1|t) = 0.5$ and $p(i=0|t) = 0.5$, the entropy is 1.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

What are impurity measures?

Gini impurity:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\,(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

Classification error:

$$I_E(t) = 1 - \max\{p(i|t)\}$$

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Computing Information Gain

- Let's begin with the root node of the DT and compute *IG* of each feature
- Consider feature "wind" $\in$ {weak,strong} and its *IG* w.r.t. the root node

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

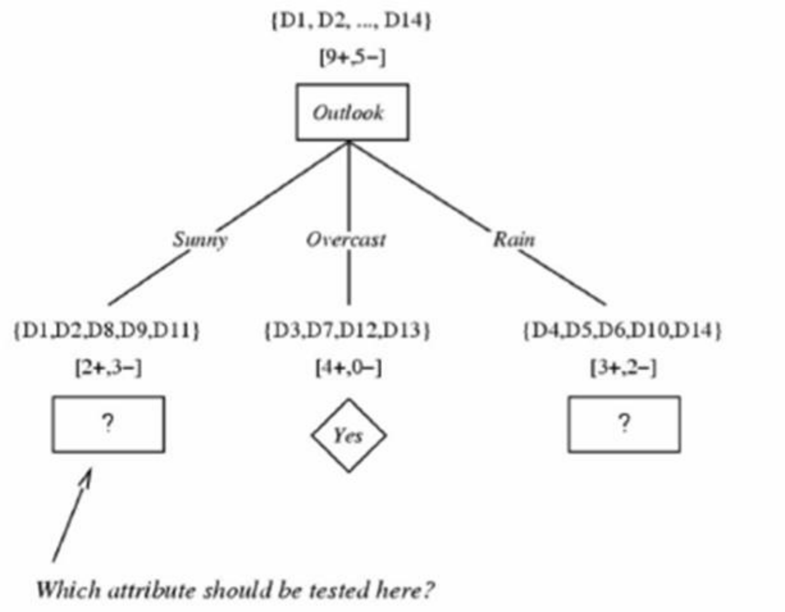
- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \Longrightarrow H(S_{weak}) = 0.811$
- $S_{strong} = [3+, 3-] \Longrightarrow H(S_{strong}) = 1$

$$
\begin{aligned}
IG(S, \text{wind}) &= H(S) - \frac{|S_{weak}|}{|S|} H(S_{weak}) - \frac{|S_{strong}|}{|S|} H(S_{strong}) \\
&= 0.94 - 8/14 * 0.811 - 6/14 * 1 \\
&= 0.048
\end{aligned}
$$

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

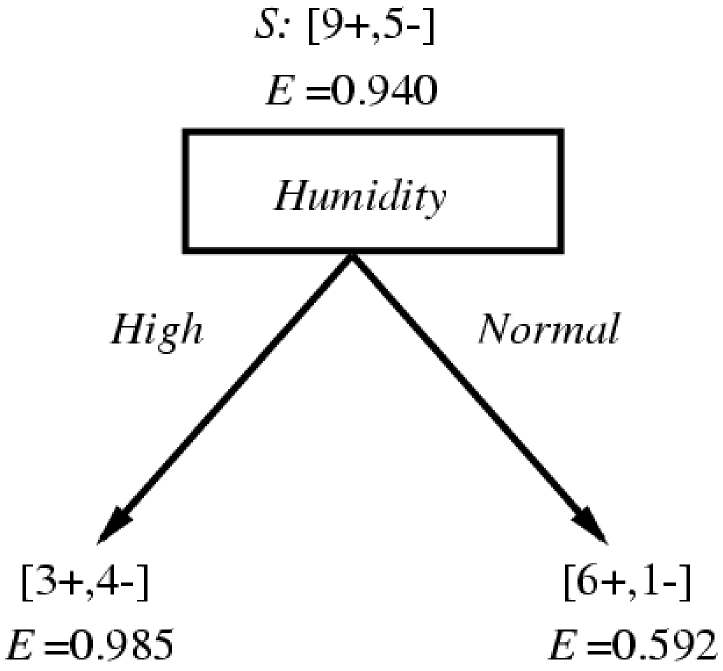
Choosing the Most Informative Feature

- At the root node, the information gains are:
  - $IG(S, \text{wind}) = 0.048$ (we already saw)
  - $IG(S, \text{outlook}) = 0.246$
  - $IG(S, \text{humidity}) = 0.151$
  - $IG(S, \text{temperature}) = 0.029$
- "outlook" has the maximum $IG \implies$ chosen as the root node
- Growing the tree:
  - Iteratively select the feature with the highest information gain for each child of the previous node



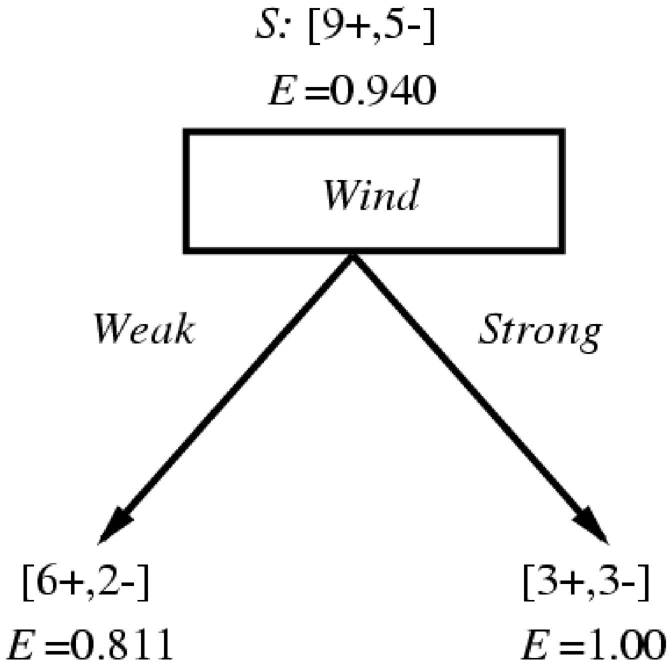
Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Selecting the Next Attribute



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

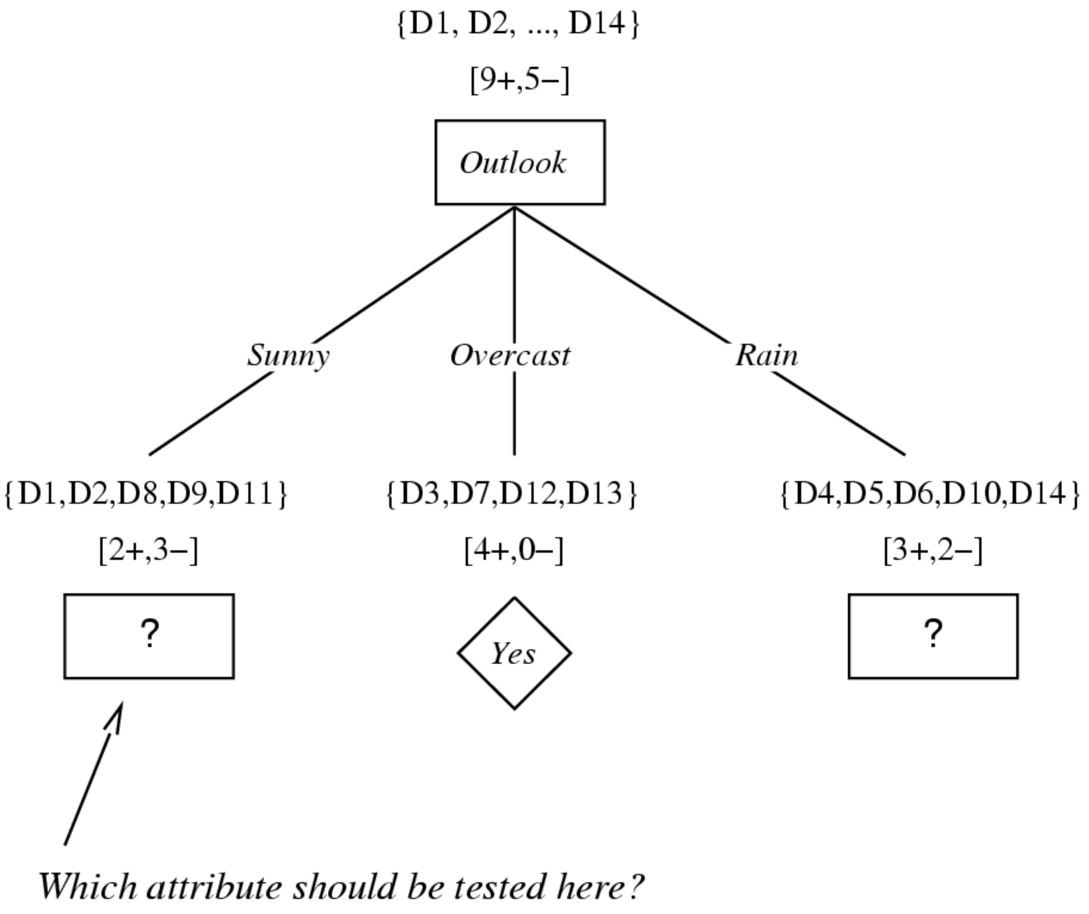


Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

And so on…



$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}\ , Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}\ , Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}\ , Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Decision Tree are adaptable:

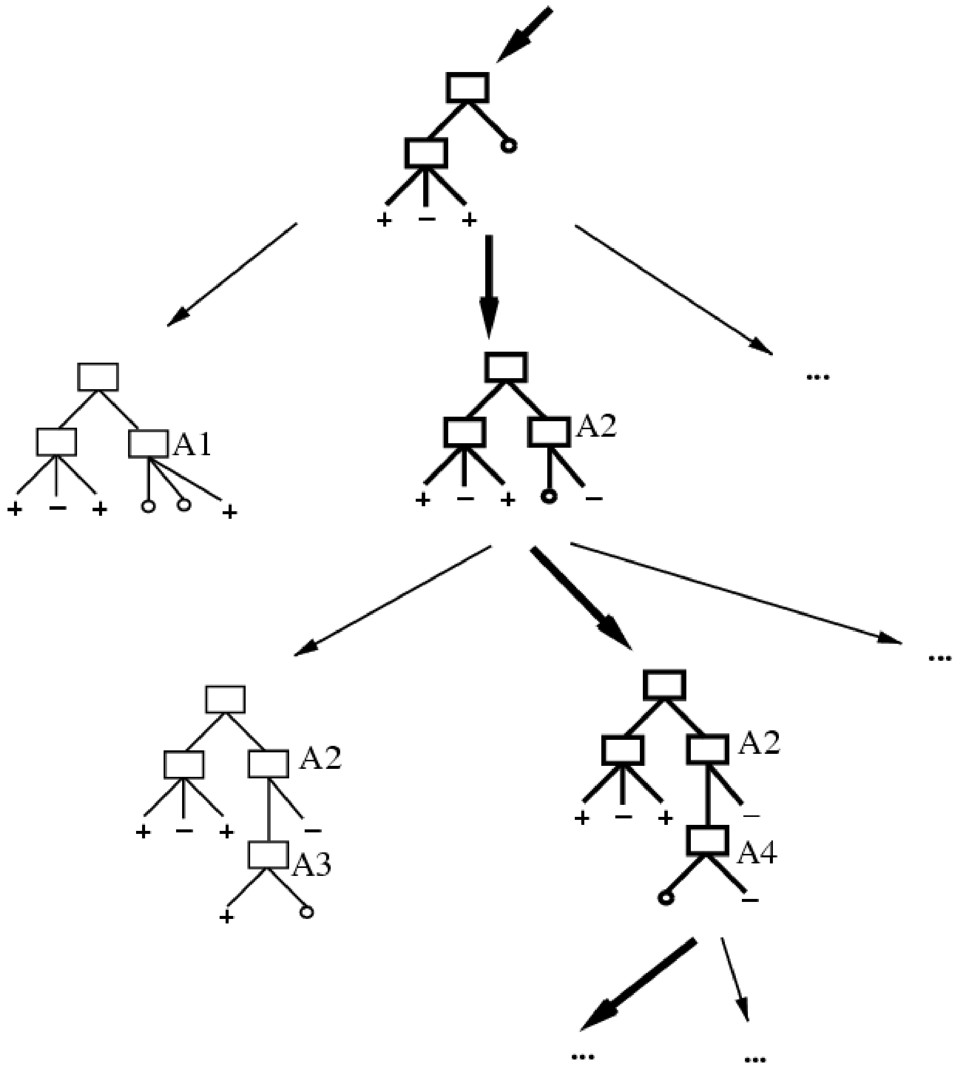
- Features with multiple discrete values
  - Multi-way splits
  - Test for one value versus the rest
  - Group values into disjoint sets

- Real-valued features
  - Use thresholds

- Regression
  - Splits based on mean squared error metric

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Hypothesis Space Search



You do not get the globally optimal tree!

Search space is exponential.

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Solution: Decision Tree Forest

1. Draw a random **bootstrap** sample of size *n* (randomly choose *n* examples from the training dataset with replacement).

2. Grow a decision tree from the bootstrap sample. At each node:
   a. Randomly select *d* features without replacement.
   b. Split the node using the feature that provides the best split according to the objective function, for instance, maximizing the information gain.

3. Repeat *steps 1-2* *k* times.

4. Aggregate the prediction by each tree to assign the class label by **majority vote**.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Overfitting

Consider error of hypothesis h over

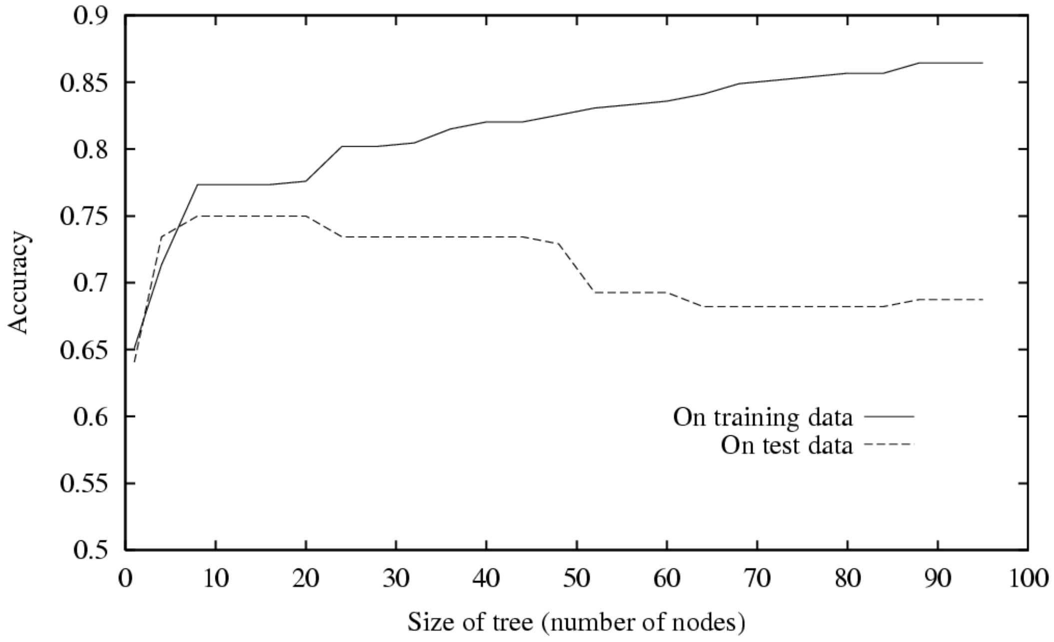
- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$



- Prune tree to reduce error on validation set

Adapted from Greg Grudic, Decision Trees (Notes borrowed from Thomas G. Dietterich and Tom Mitchell)

Conclusion:

- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap

- It's possible to get in trouble with overfitting

- They do classification: predict a categorical output from categorical and/or real inputs