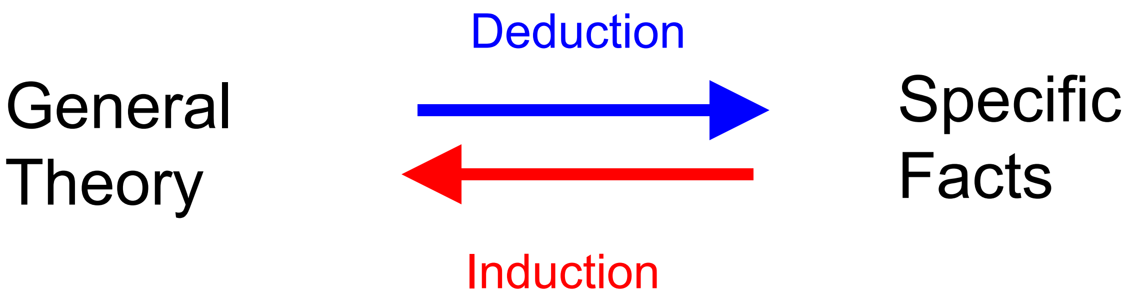
Machine Learning

- What is Machine Learning?
  - Programs that get better with *experience* given a *task* and some *performance measure.*
    - Learning to classify news articles
    - Learning to recognize spoken words
    - Learning to play board games
    - Learning to navigate (e.g. self-driving cars)
- Usually involves some sort of <u>inductive reasoning</u> step.

Inductive Reasoning

- Deductive reasoning (rule based reasoning)
  - From the general to the specific
- Inductive reasoning
  - From the specific to the general



Note: not to be confused with mathematical induction!

Example

- Facts: every time you see a swan you notice that the swan is white.
- Inductive step: you infer that all swans are white.

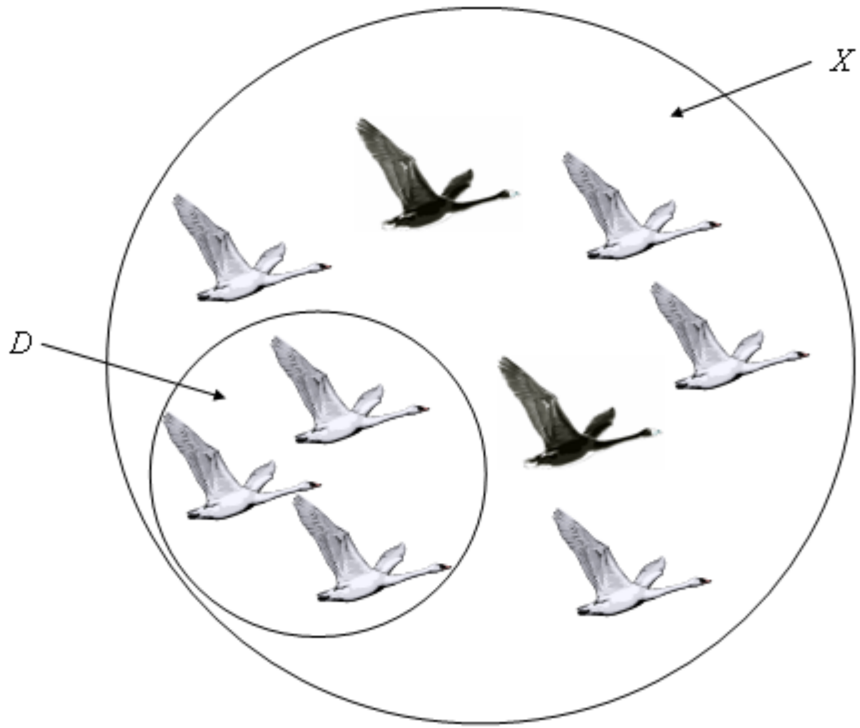
Observed Swans are white. $\xrightarrow{\text{Induction}}$ All Swans are white.

Inference is the act or process of drawing a conclusion based solely on what one already knows.

Observation

- Deduction is “truth preserving”
  - If the rules employed in the deductive reasoning process are sound, then, what holds in the theory will hold for the deduced facts.
- Induction is NOT “truth preserving”
  - It is more of a statistical argument
  - The more swans you see that are white, the more probable it is that all swans are white. But this does not exclude the existence of black swans.

Observation



D ≡ observations
X ≡ universe of all swans

Different Styles of Machine Learning

- <u>Supervised</u> Learning
  - The learning needs explicit examples of the concept to be learned (e.g. white swans, playing tennis, *etc*)
- <u>Unsupervised</u> Learning
  - The learner discovers autonomously any structure in a domain that might represent an interesting concept

Knowledge - Representing what has been learned

- Symbolic Learners (transparent models)
  - If-then-else rules
  - Decision trees
  - Association rules
- Sub-Symbolic Learners (non-transparent models)
  - (Deep) Neural Networks
  - Clustering (Self-Organizing Maps, k-Means)
  - Support Vector Machines

Decision Trees

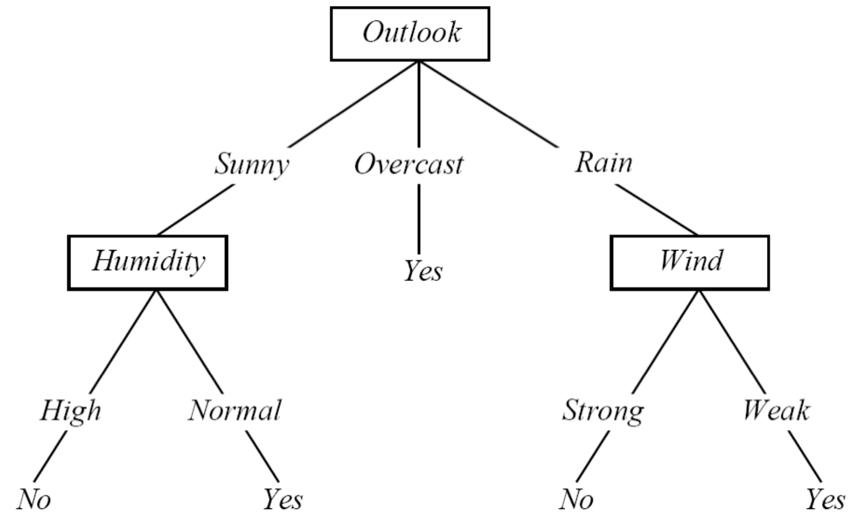
- Learn from labeled observations - supervised learning
- Represent the knowledge learned in form of a tree

Example: learning when to play tennis.

- Examples/observations are days with their observed characteristics and whether we played tennis or not

Play Tennis Example

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

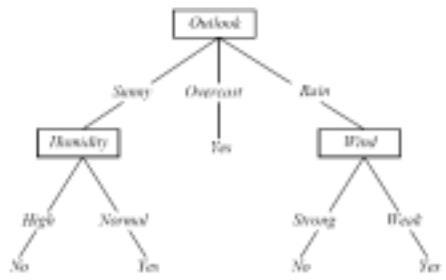


Decision Tree Learning

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Facts or Observations

Induction

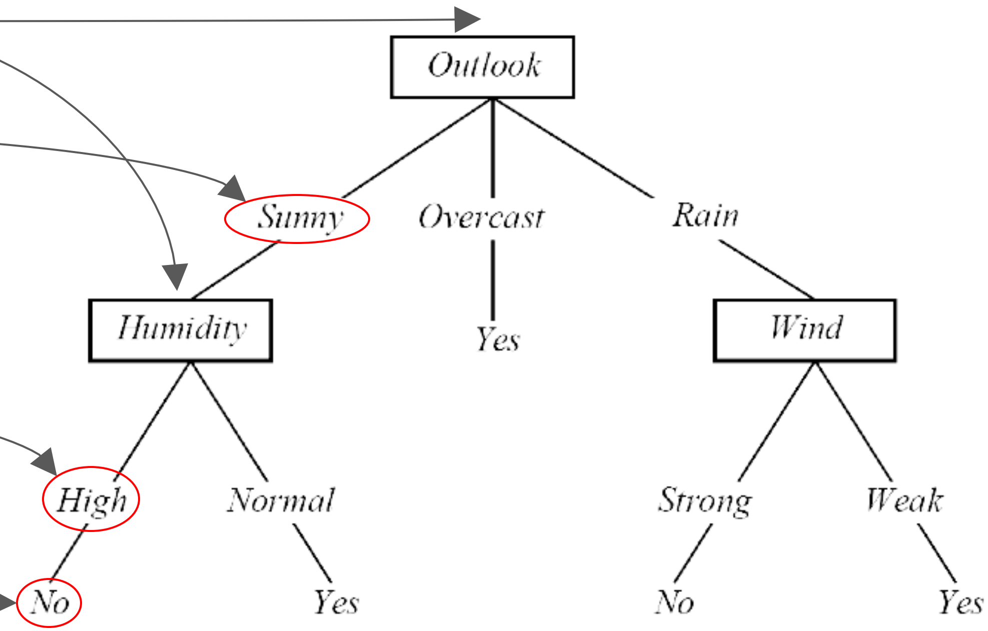


Theory

Interpreting a DT

DT ≡ Decision Tree

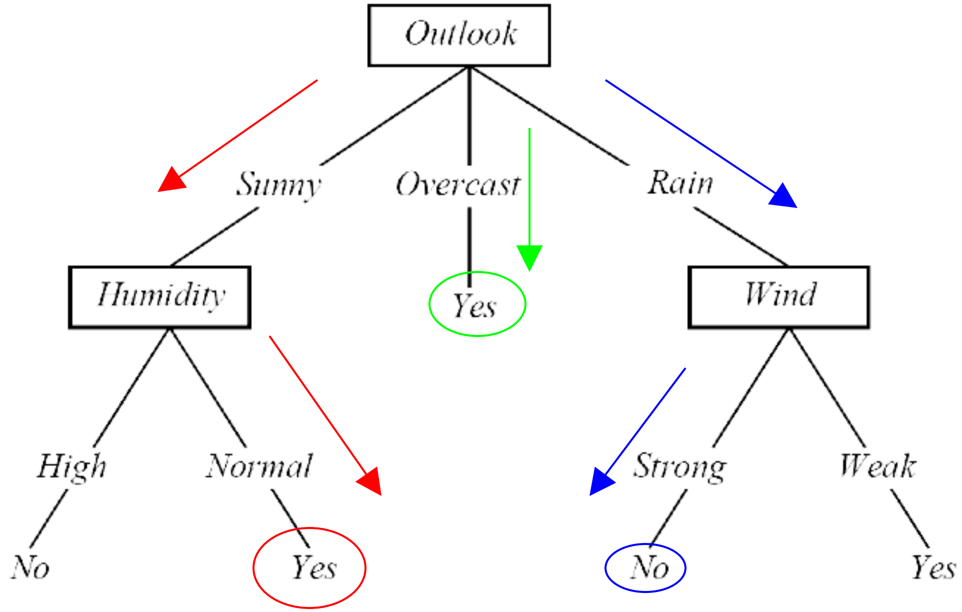
Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



- ➔ A DT uses the <u>features</u> of an observation table as nodes and the <u>feature values</u> as links.
- ➔ <u>All</u> feature values of a particular feature need to be represented as links.
- ➔ The target feature is special - its values show up as <u>leaf nodes</u> in the DT.

Interpreting a DT

Each path from the root of the DT to a leaf can be interpreted as a decision rule.



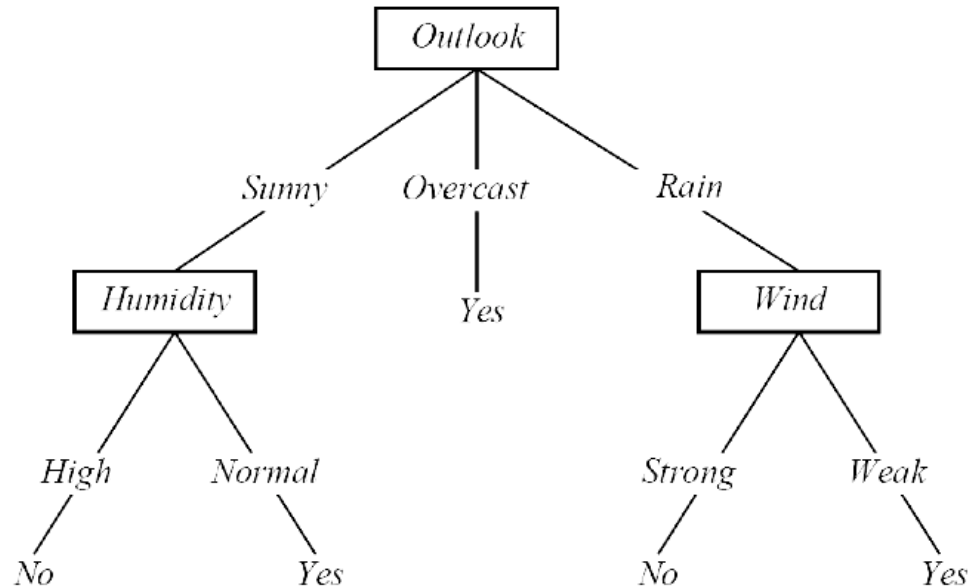
IF *Outlook* = *Sunny* AND *Humidity* = *Normal* THEN *Playtennis* = *Yes*

IF *Outlook* = *Overcast* THEN *Playtennis* =*Yes*

IF *Outlook* = *Rain* AND *Wind* = *Strong* THEN *Playtennis* = *No*

DT: Explanation & Prediction

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Explanation: the DT summarizes (explains) all the observations in the table perfectly ⇒ 100% Accuracy

Prediction: once we have a DT (or model) we can use it to make predictions on observations that are not in the original training table, consider:

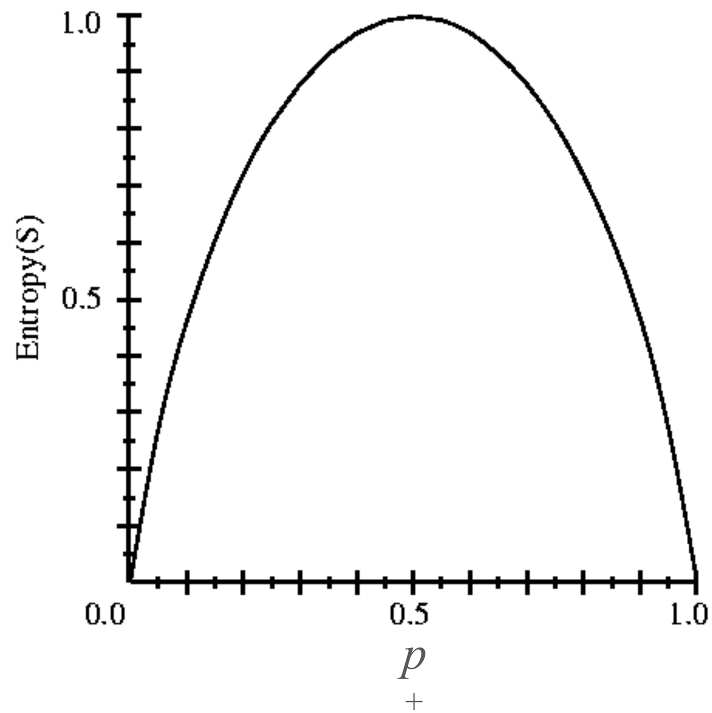
Outlook = Sunny, Temperature = Mild, Humidity = Normal, Windy = False, Playtennis = ?

Constructing DTs

- How do we choose the attributes and the order in which they appear in a DT?
  - Recursive partitioning of the original data table
  - Heuristic - each generated partition has to be “less random” (entropy reduction) than previously generated partitions

Entropy

- 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 measures the impurity (randomness) of S



S

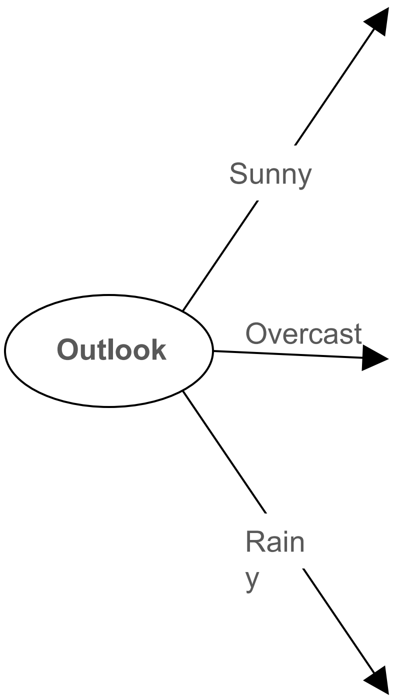
Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$Entropy(S) = Entropy([9+,5-]) = .94$

$$Entropy(S) \equiv - p^+ \log_2 p^+ - p^- \log_2 p^-$$

$$\text{AvgEntropy}(S,A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v) \quad \text{(weighted average)}$$

Partitioning the Data Set



Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

E = .97

Outlook	Temperature	Humidity	Windy	PlayTennis
Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

E = 0

Outlook	Temperature	Humidity	Windy	PlayTennis
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No

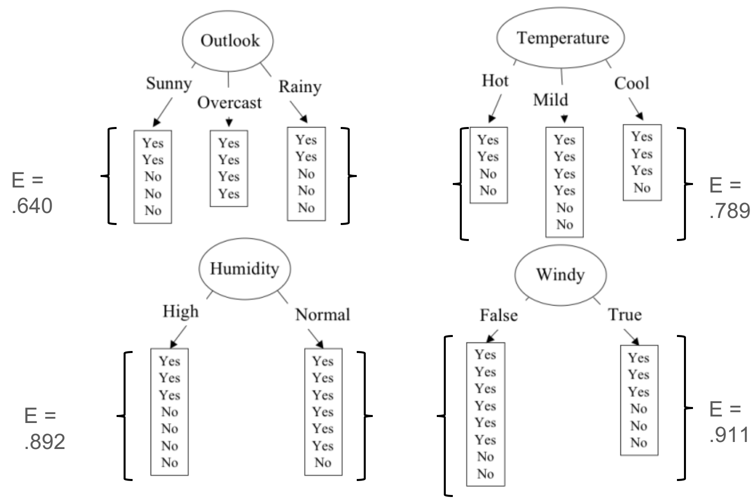
E = .97

Average Entropy = .64

(weighted .69)

Partitioning in Action

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Recursive Partitioning

Partition(*Examples*, *TargetAttribute*, *Attributes*)

Examples are the training examples. TargetAttribute is a binary (+/-) *categorical dependent variable and Attributes is the list of independent variables which are available for testing at this point. This function returns a decision tree.*

- Create a *Root* node for the tree.
- If all *Examples* are positive then return *Root* as a leaf node with label = +.
- Else if all *Examples* are negative then return *Root* as a leaf node with label = -.
- Else if *Attributes* is empty then return *Root* as a leaf node with label = most common value of TargetAttribute in Examples.
- Otherwise
  - A := the attribute from *Attributes* that reduces entropy the most on the *Examples*.
  - *Root* := A
  - For each $v \in values(A)$
    - Add a new branch below the *Root* node with value $A = v$
    - Let $Examples_v$ be the subset of *Examples* where $A = v$
    - If $Examples_v$ is empty then add new leaf node to branch with label = most common value of *TargetAttribute* in *Examples*.
    - Else add new subtree to branch Partition($Examples_v$, *TargetAttribute*, $Attributes - \{A\}$)
- Return Root

Based on material from the book: "Machine Learning", Tom M. Mitchell. McGraw-Hill, 1997.

Recursive Partitioning

Our data set:

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Recursive Partitioning

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Outlook

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No

Recursive Partitioning

Outlook

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No

Recursive Partitioning

Outlook

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No

Humidity

Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No

Recursive Partitioning

Outlook

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No

Humidity

Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Sunny	Mild	High	False	No

Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Windy

Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes

Rainy	Cool	Normal	True	No
Rainy	Mild	High	True	No

Machine Learning in Python - Scikit-Learn

- We will be using the Scikit-Learn module to build decision trees.
    - Scikit-learn or sklearn for short provides all kinds of models
        - Neural networks
        - Support vector machines
        - Clustering algorithms
        - Linear regression
        - *etc*
- We will be using the treeviz module to visualize decision trees.
    - A simple ASCII based tree visualizer

SKlearn Decision Tree Basics

Training data needs to be structured into a *feature matrix* and a *target vector*.

In the feature matrix one row for each observations.

In the target vector one entry for each observation.

NOTE: rows and vector entries have to be consistent!