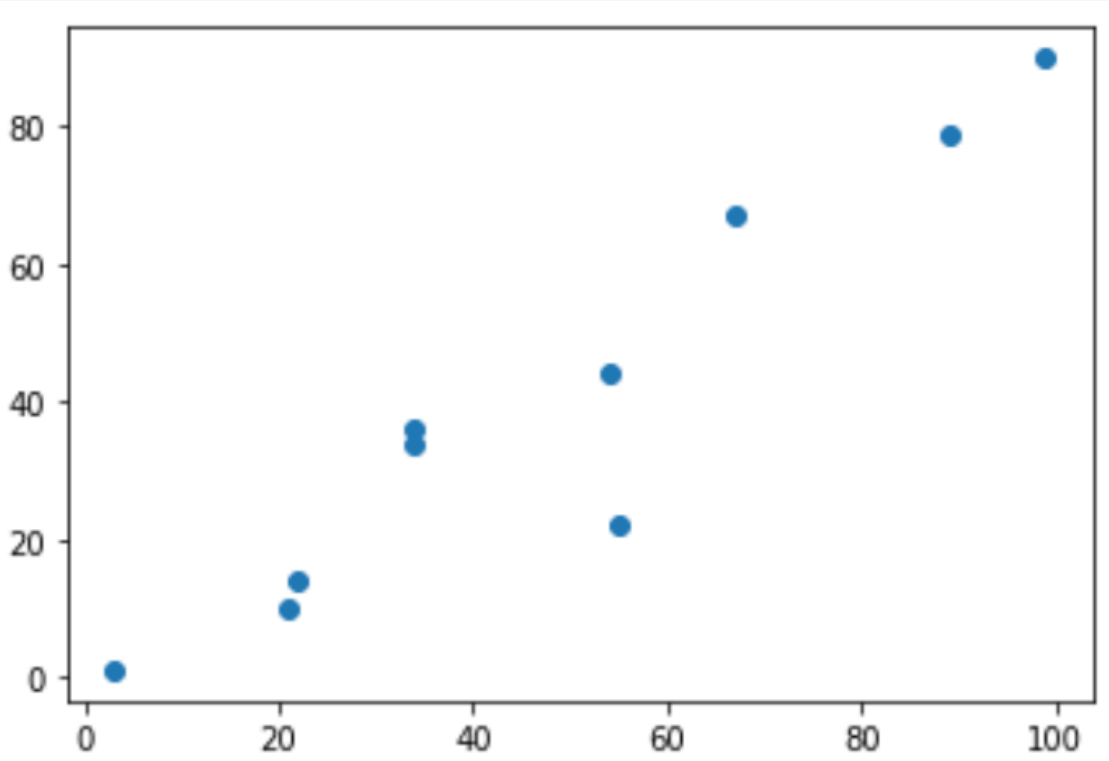
Lab 6 – Machine Learning Algorithms

Contents

Linear Regression

- Data Preparation

```
#Import the required modules, numpy for calculation, and Matplotlib for drawing
import numpy as np
import matplotlib.pyplot as plt
#This code is for jupyter Notebook only
%matplotlib inline

# define data, and change list to array
x = [3,21,22,34,54,34,55,67,89,99]
x = np.array(x)
y = [1,10,14,34,44,36,22,67,79,90]
y = np.array(y)

#Show the effect of a scatter plot
plt.scatter(x,y)
```



Linear Regression

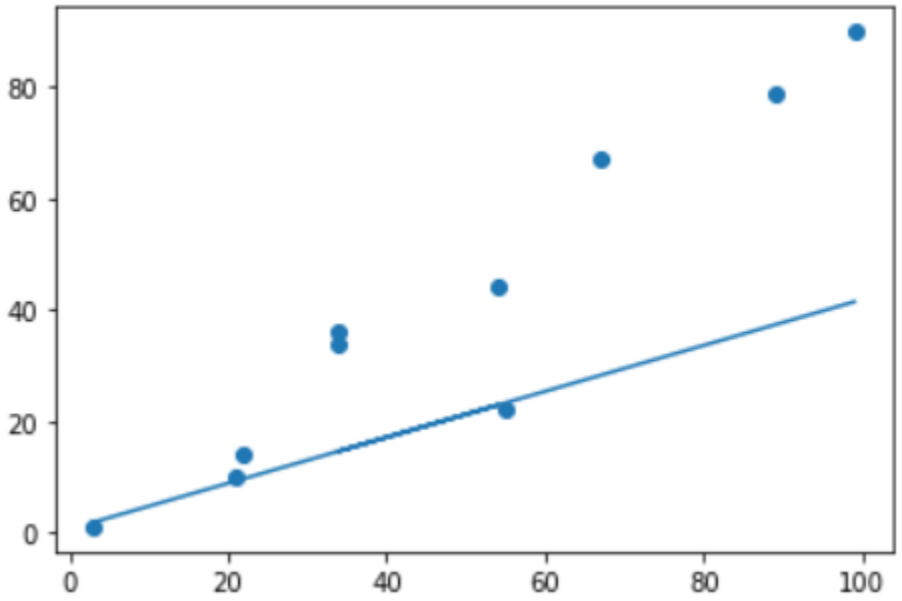
```python
#The basic linear regression model is wx+ b, and since this is a two-dimensional space, the model is ax+ b

def model(a, b, x):
    return a*x + b

#The most commonly used loss function of linear regression model is the loss function of mean variance difference
def loss_function(a, b, x, y):
    num = len(x)
    prediction=model(a,b,x)
    return (0.5/num) * (np.square(prediction-y)).sum()

#The optimization function mainly USES partial derivatives to update two parameters a and b
def optimize(a,b,x,y):
    num = len(x)
    prediction = model(a,b,x)
    #Update the values of A and B by finding the partial derivatives of the loss function on a and b
    da = (1.0/num) * ((prediction -y)*x).sum()
    db = (1.0/num) * ((prediction -y).sum())
    a = a - Lr*da
    b = b - Lr*db
    return a, b

#iterated function, return a and b
def iterate(a,b,x,y,times):
    for i in range(times):
        a,b = optimize(a,b,x,y)
    return a,b
```

Linear Regression

- Start the Iteration

  Step1: Initialization and Iterative optimization model

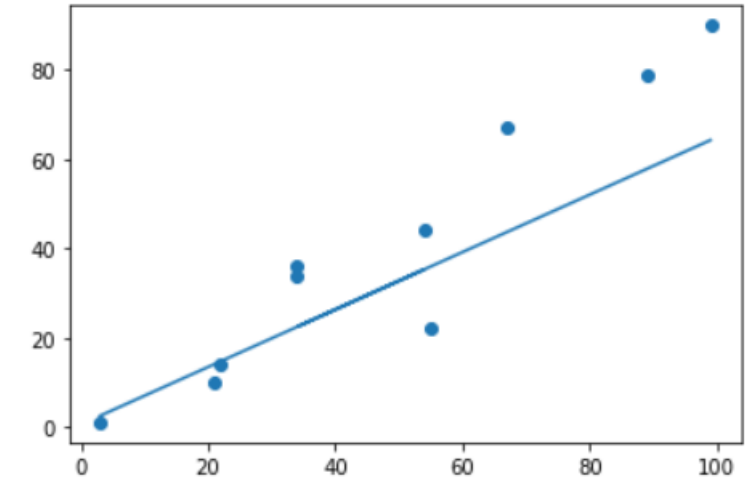
```
#Initialize parameters and display
a = np.random.rand(1)
print(a)
b = np.random.rand(1)
print(b)
Lr = 1e-4

#For the first iteration, the parameter values, losses,

#and visualization after the iteration are displayed
a,b = iterate(a,b,x,y,1)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Linear Regression

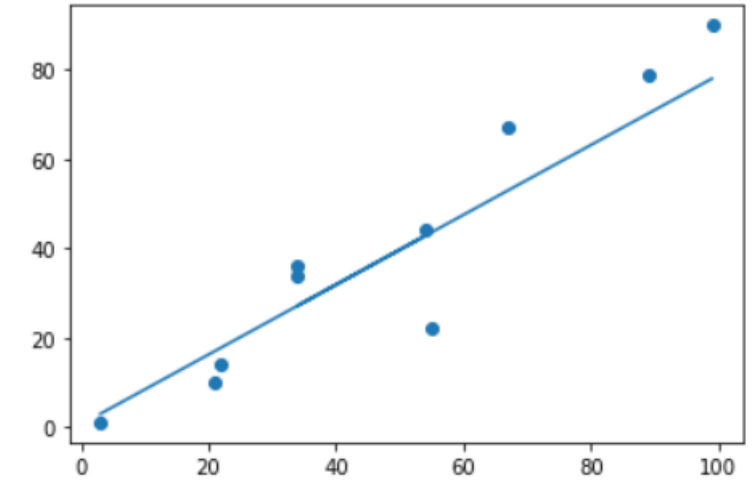
Step2: In the second iteration, the parameter values, loss values and visualization effects after the iteration are displayed

```
a,b = iterate(a,b,x,y,2)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Step 3: The third iteration shows the parameter values, loss values and visualization after iteration

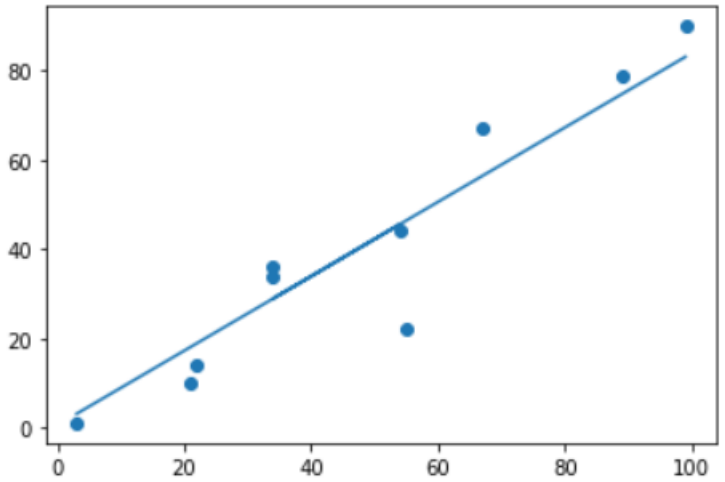
```
a,b = iterate(a,b,x,y,3)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Linear Regression

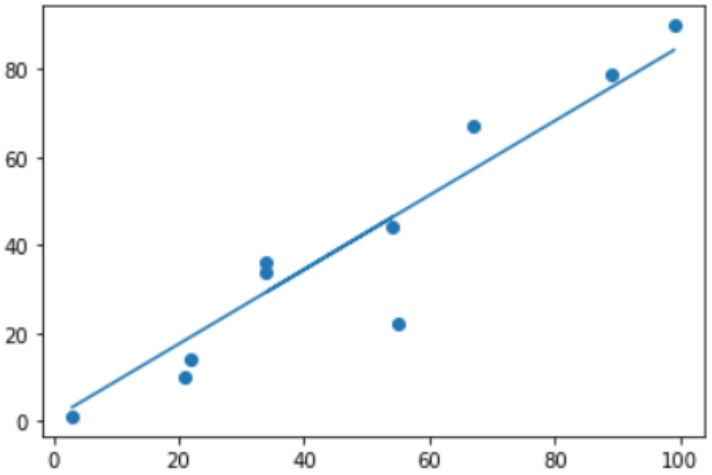
Step 4: In the fourth iteration, parameter values, loss values and visualization effects are displayed

```
a,b = iterate(a,b,x,y,4)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Step 5: The fifth iteration shows the parameter value, loss value and visualization effect after iteration

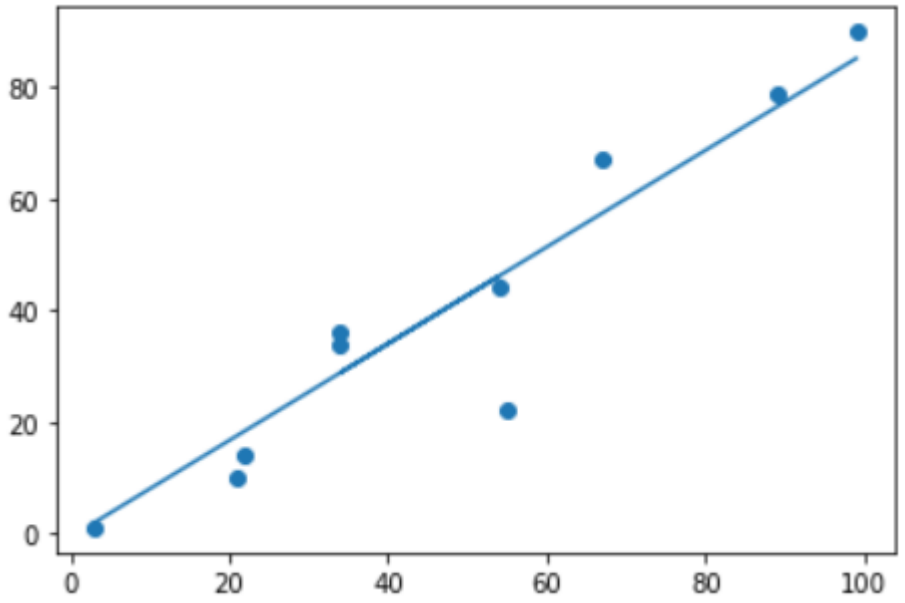
```
a,b = iterate(a,b,x,y,5)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Linear Regression

Step 6: The 10000th iteration, showing the parameter values, losses and visualization after iteration

```
a,b = iterate(a,b,x,y,10000)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```



Contents

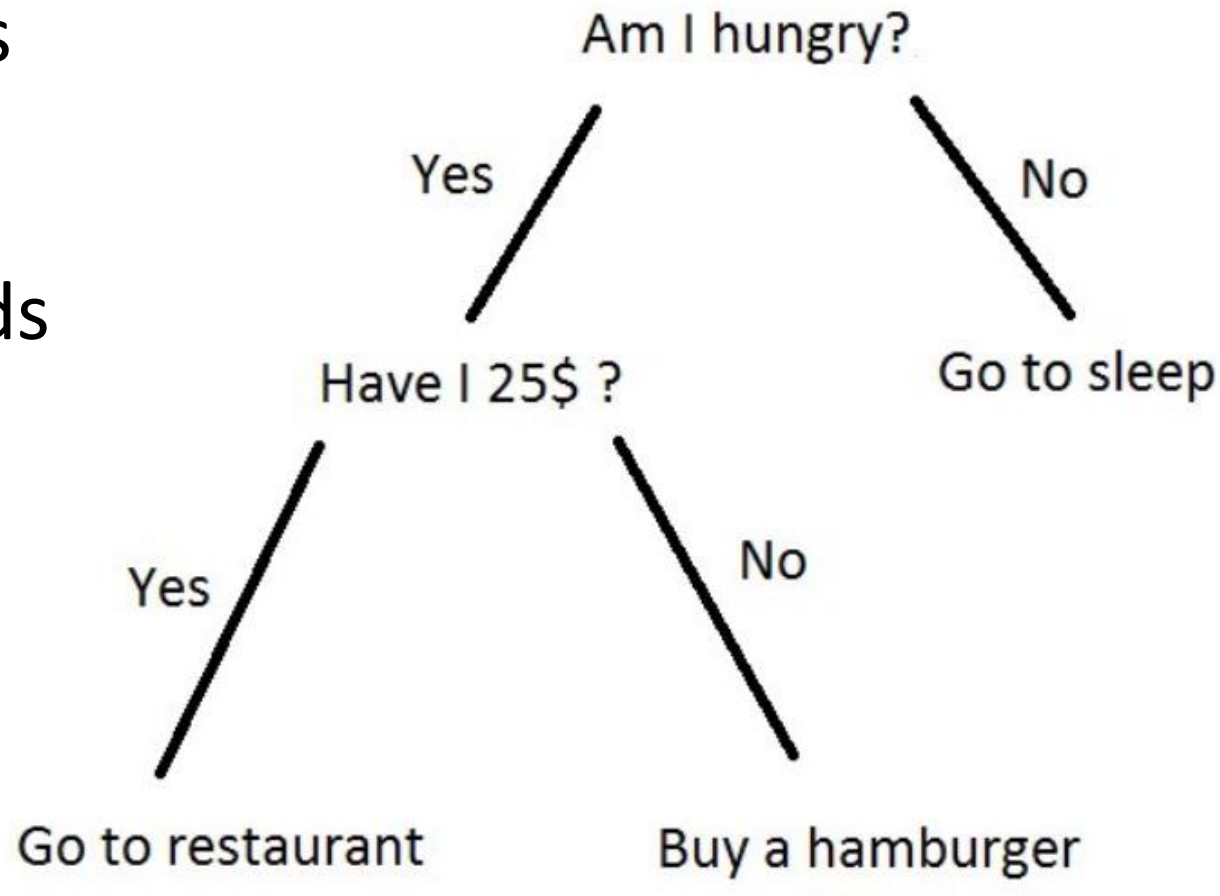
1. Linear Regression
2. **Decision Trees**

Decision Trees

- The idea is to partition input space into a disjoint set of regions and to use a very simple predictor for each region.
- For classification simply predict the most frequent class in the region

Decision Trees Representation

- Each internal node tests an attribute.
- Each branch corresponds to attribute value.
- Each leaf node make a prediction.



Decision Trees

The strengths of decision tree methods are:

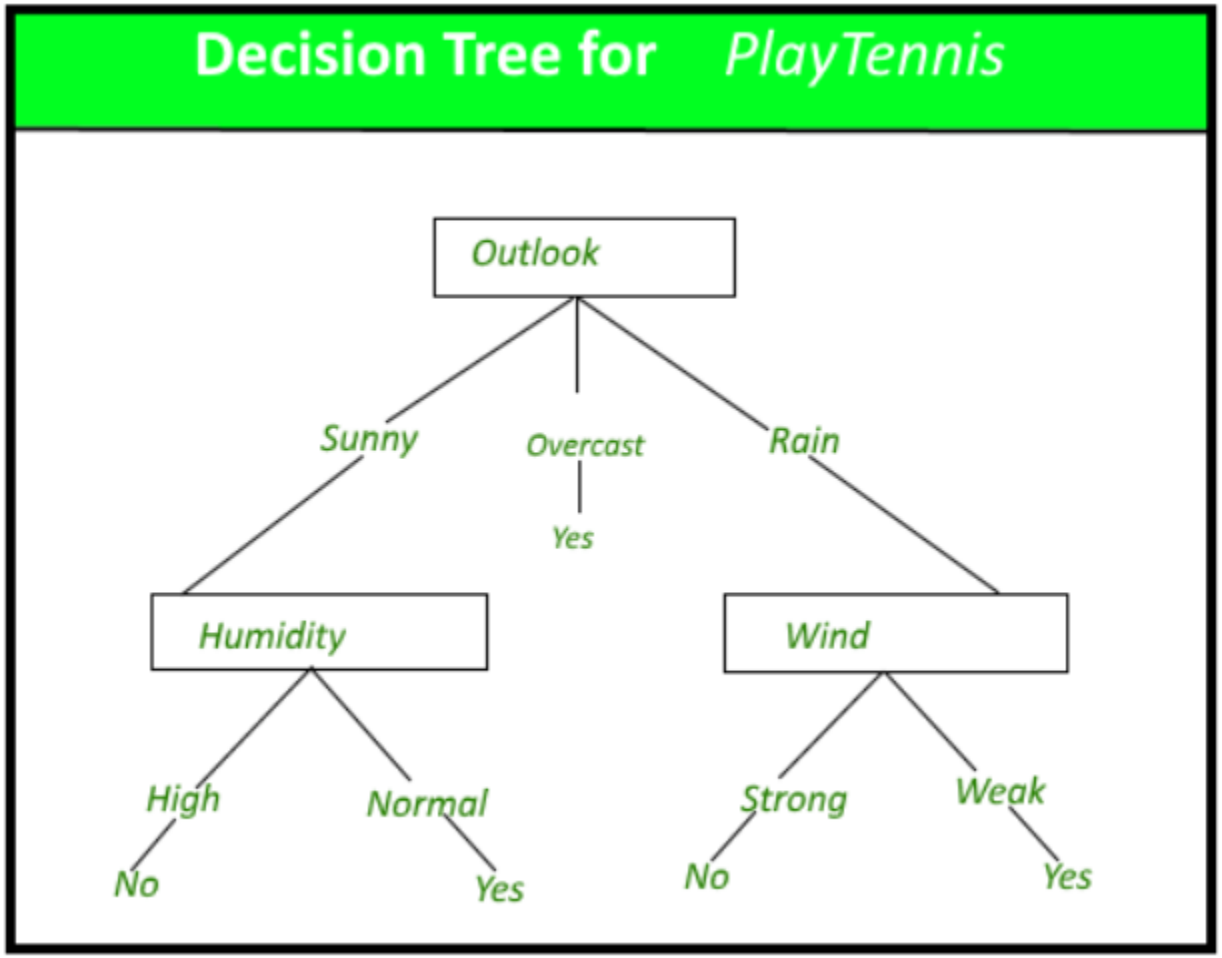
- Decision trees are able to generate **understandable rules**.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to **handle both continuous** and **categorical variables**.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Decision Trees

The weaknesses of decision tree methods are:

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems **with many classes** and relatively **small number of training examples**.
- Decision tree can be **computationally expensive to train**.

Play tennis training data



Play tennis training data

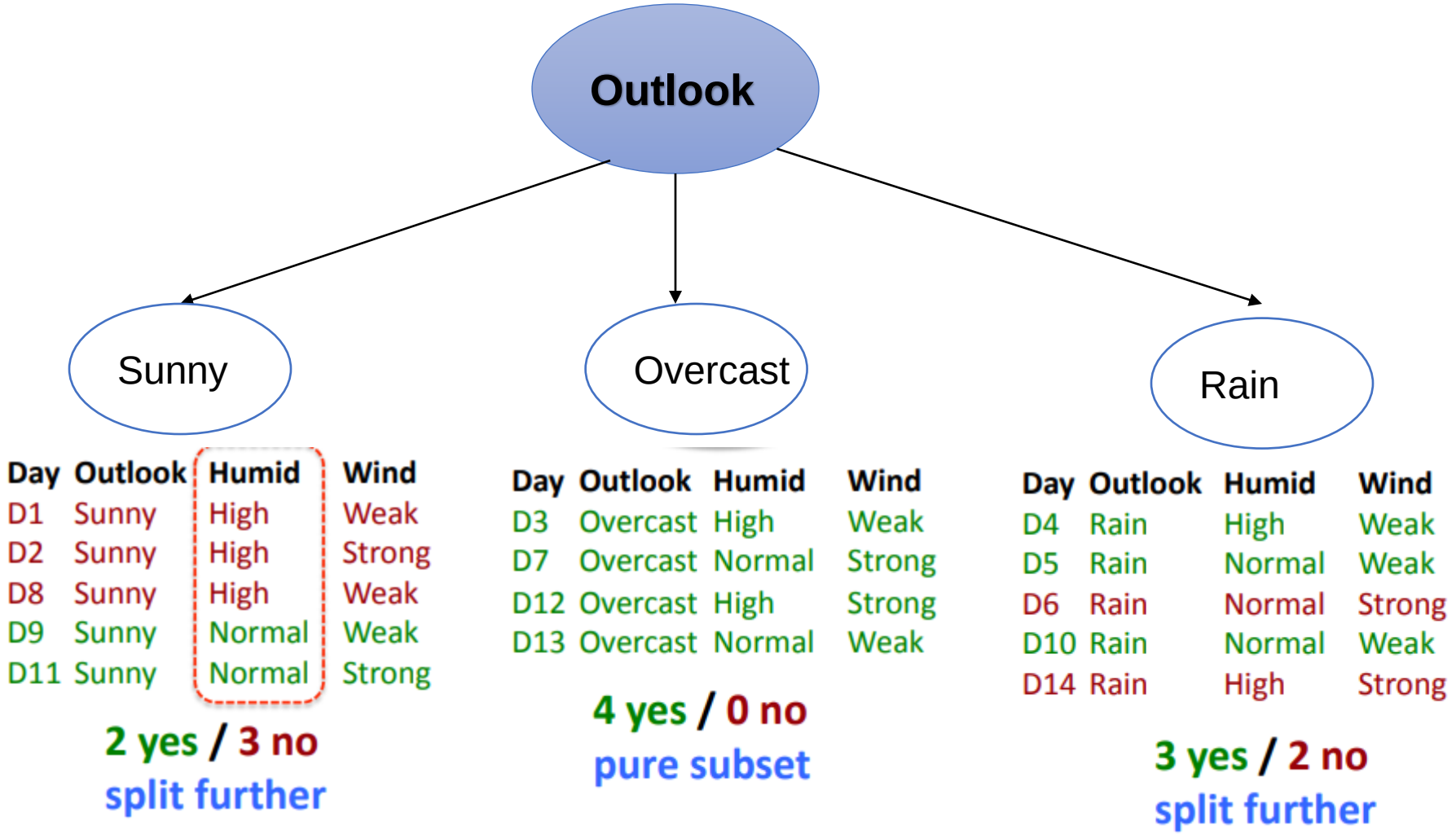
- Hard to guess.
- Divide & Conquer:
  - split into subsets
  - are they pure? (all yes or all no)
  - if yes: stop.
  - If no: repeat.
- See which subset new data falls into

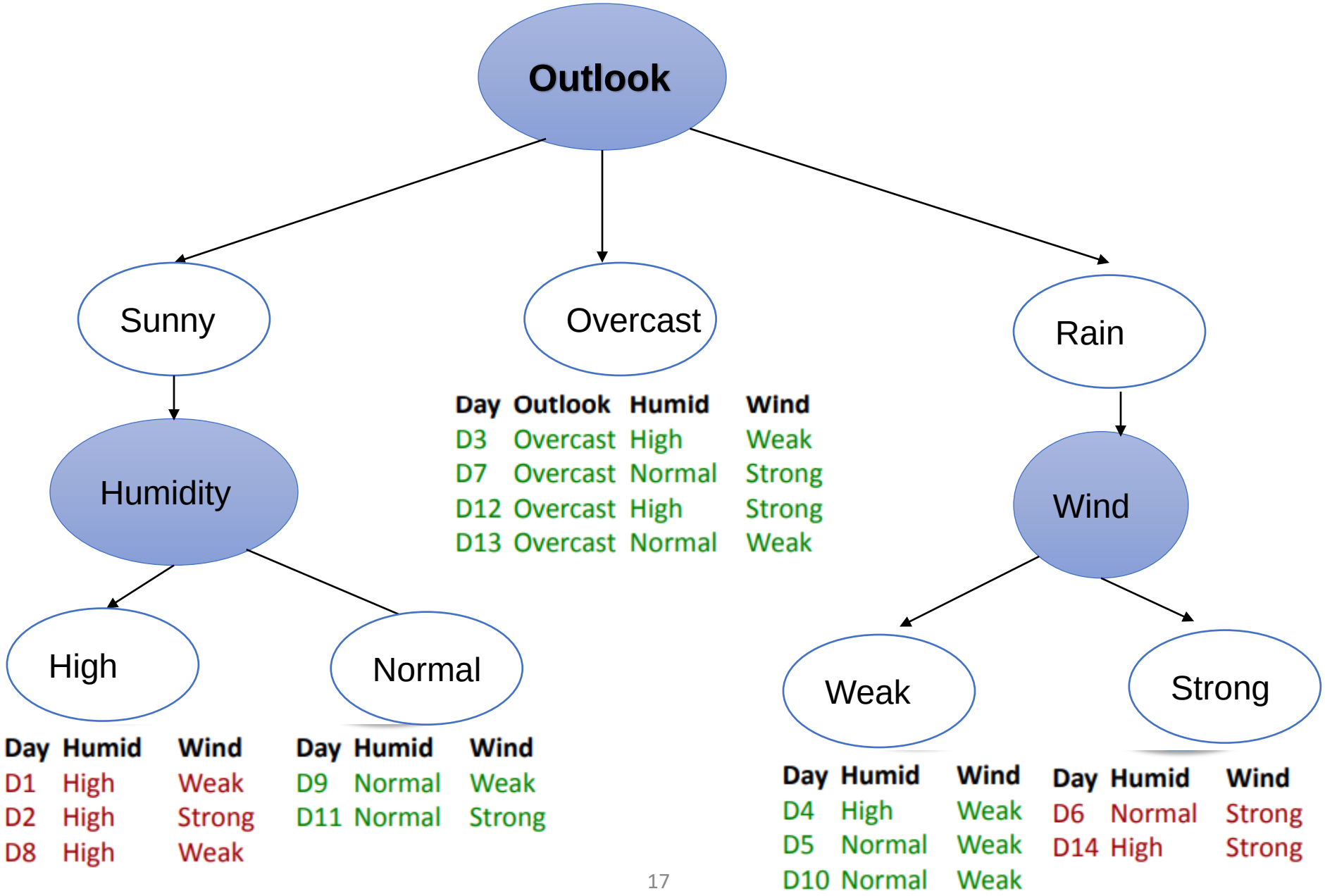
Training examples: **9 yes / 5 no**

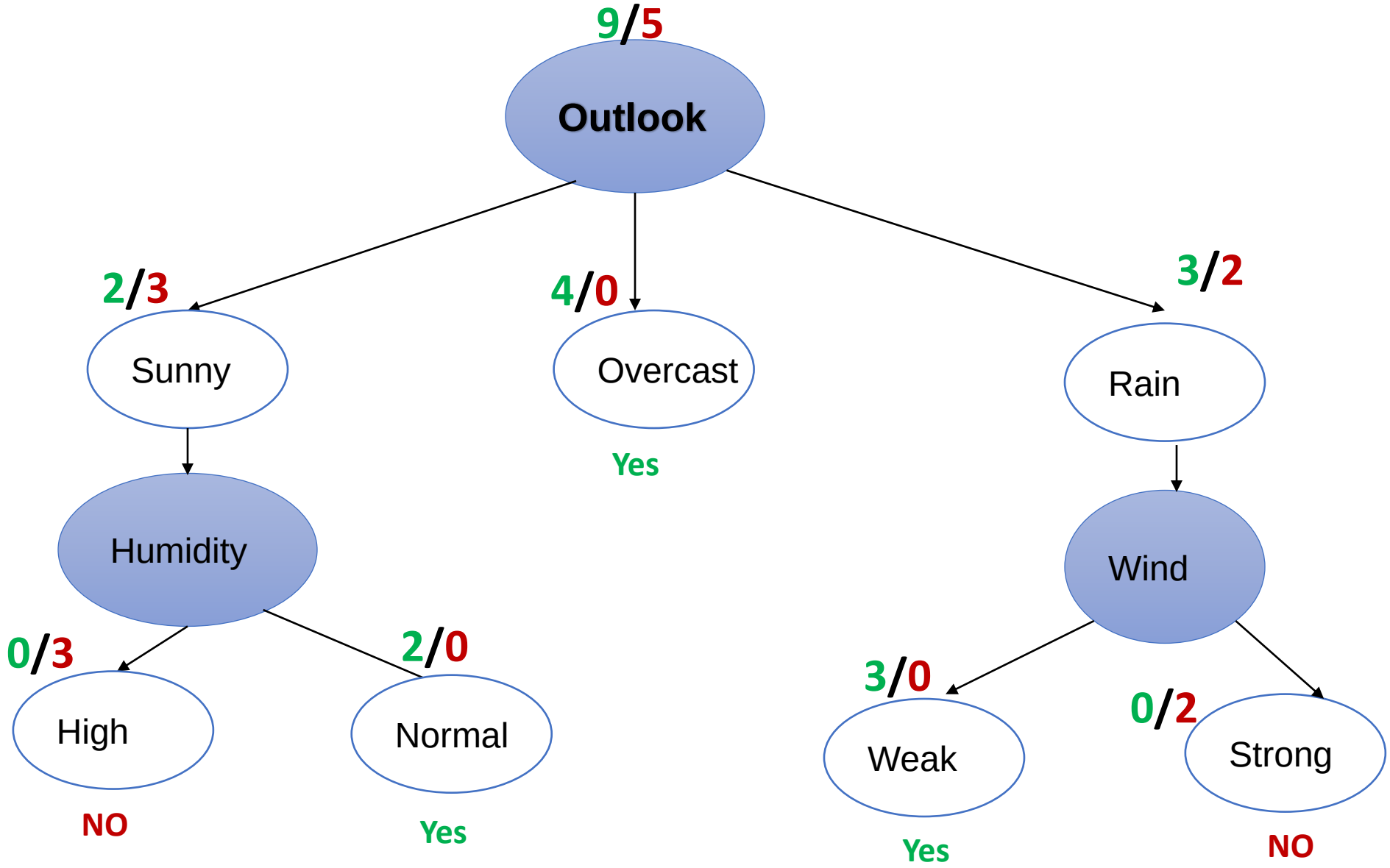
Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes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

New Data

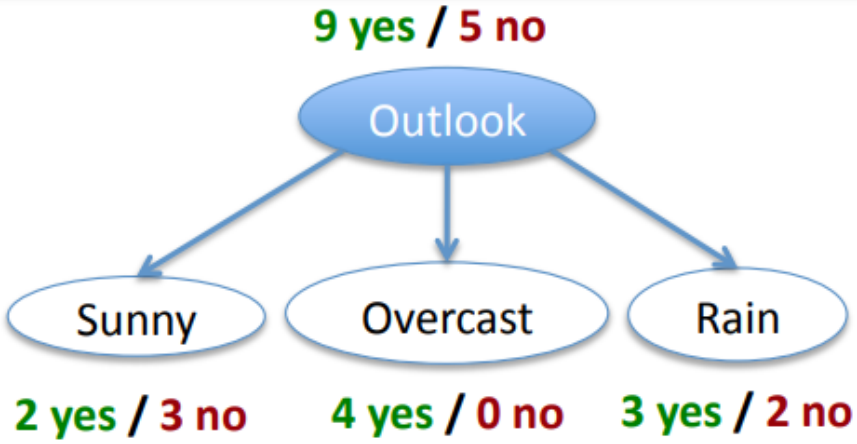
D15	Rain	High	weak	?

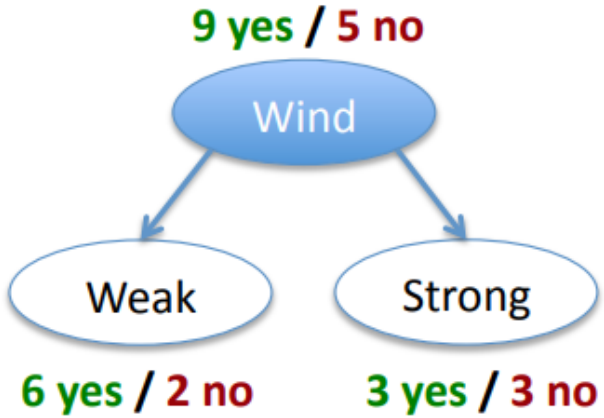

Outlook
Sunny
Overcast
Rain
Day Outlook Humid Wind
D1 Sunny High Weak
D2 Sunny High Strong
D8 Sunny High Weak
D9 Sunny Normal Weak
D11 Sunny Normal Strong
2 yes / 3 no
split further
Day Outlook Humid Wind
D3 Overcast High Weak
D7 Overcast Normal Strong
D12 Overcast High Strong
D13 Overcast Normal Weak
4 yes / 0 no
pure subset
Day Outlook Humid Wind
D4 Rain High Weak
D5 Rain Normal Weak
D6 Rain Normal Strong
D10 Rain Normal Weak
D14 Rain High Strong
3 yes / 2 no
split further


Outlook
Sunny
Overcast
Rain
Humidity
Wind
High
Normal
Weak
Strong
Day Outlook Humid Wind
D3 Overcast High Weak
D7 Overcast Normal Strong
D12 Overcast High Strong
D13 Overcast Normal Weak
Day Humid Wind
D1 High Weak
D2 High Strong
D8 High Weak
Day Humid Wind
D9 Normal Weak
D11 Normal Strong
Day Humid Wind
D4 High Weak
D5 Normal Weak
D10 Normal Weak
Day Humid Wind
D6 Normal Strong
D14 High Strong


9/5
Outlook
2/3
Sunny
4/0
Overcast
3/2
Rain
Yes
Humidity
Wind
0/3
High
2/0
Normal
3/0
Weak
0/2
Strong
NO
Yes
Yes
NO

Which attribute to split on





- Want to measure "purity" of the split
  - more certain about Yes/No after the split
    - pure set (4 yes / 0 no) => completely certain (100%)
    - impure (3 yes / 3 no) => completely uncertain (50%)
  - can't use P("yes" | set):
    - must be symmetric: 4 yes / 0 no as pure as 0 yes / 4 no

How Does a Decision Tree Select Splits?

- The ID3 decision makes use of two concepts when creating a tree from top-down:
  - **Entropy**
  - **Information Gain** (as referred to as just **gain**)

- Using these two concepts, the nodes to be created and the attributes to split on can be determined.

- **Gain** measures how well a given attribute separates training examples into targeted classes. The one with **the highest information** (information being the most useful for classification) is selected.

- In order to define gain, we first borrow an idea from information theory called entropy. **Entropy** measures the amount of information in an attribute.

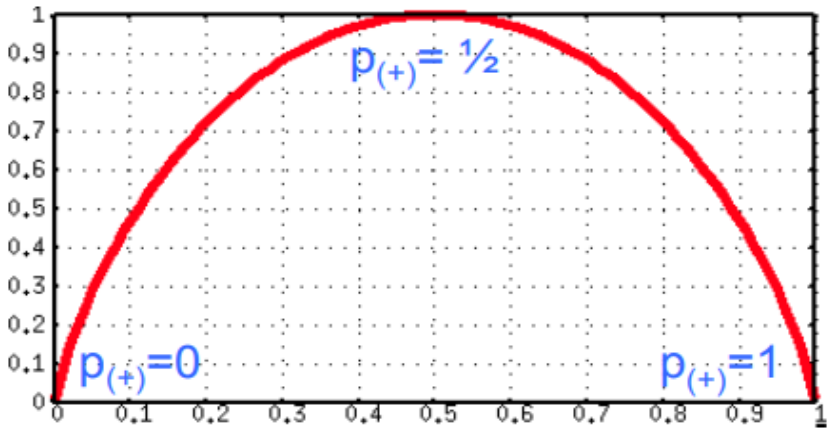
Entropy

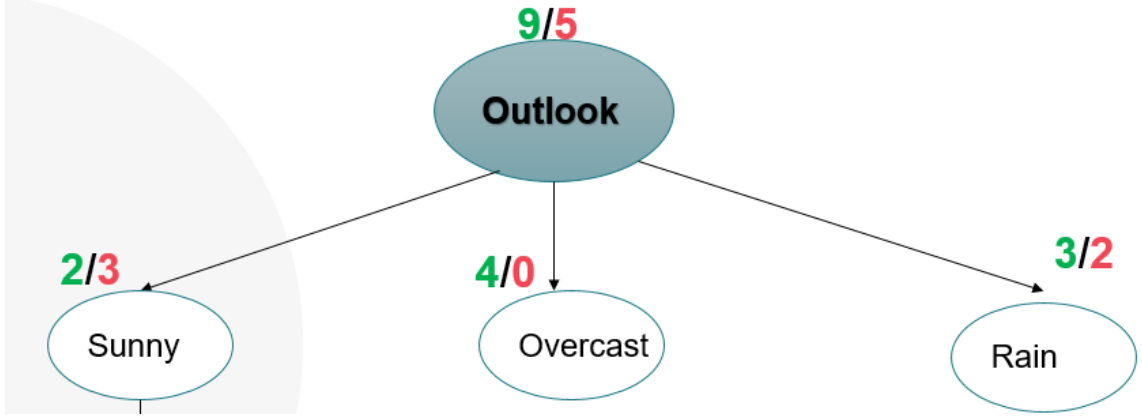
- Entropy: $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$ bits
  - S … subset of training examples
  - $p_{(+)}$ / $p_{(-)}$ … % of positive / negative examples in S
- Interpretation: assume item X belongs to S
  - how many bits need to tell if X positive or negative
- impure (3 yes / 3 no):

$$H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1 \text{ bits}$$

- pure set (4 yes / 0 no):

$$H(S) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0 \text{ bits}$$





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

- H(Outlook) = $-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$
- H(Sunny) = $-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$
- H(Overcast) = $-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$
- H(Rain) = $-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$

Information Gain

- Information Gain (also known as just Gain) uses the entropy in order to determine what attribute is best used to create a split with.

- The column with the higher Gain will be used as the node of the decision tree.

Information Gain

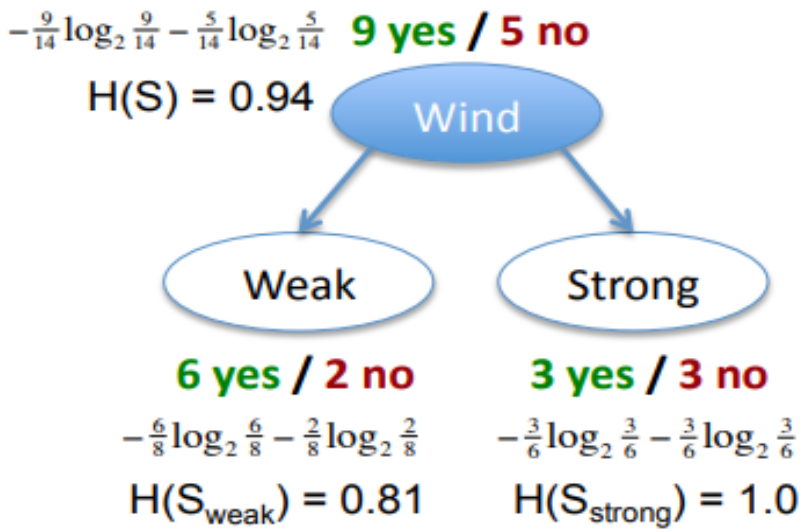
- Want many items in pure sets.
- Expected drop in entropy after split:

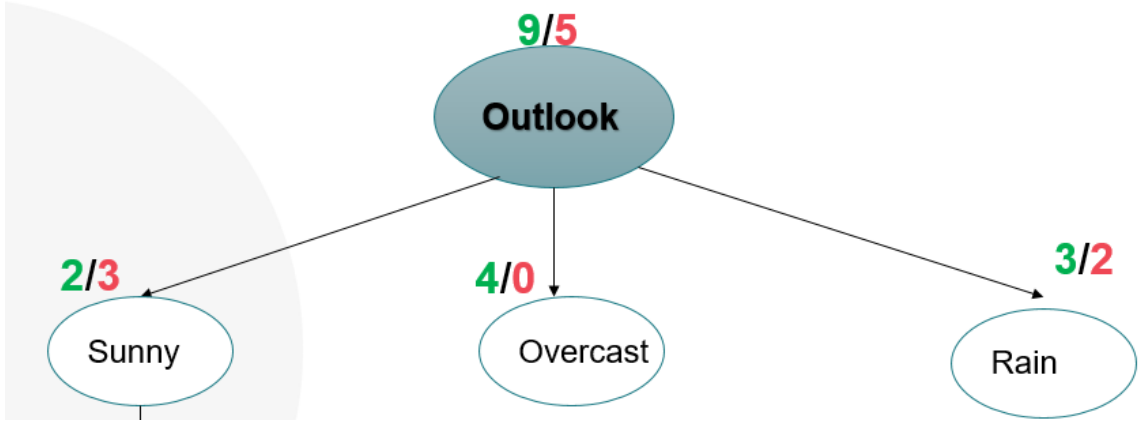
V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

Gain (S, Wind)
= H(S) – $^8/_{14}$ H(S_{weak}) – $^6/_{14}$ H(S_{strong})
= 0.94 – $^8/_{14}$ * 0.81 – $^6/_{14}$ * 1.0
= 0.049

Wind Example





$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

- H(Outlook) = $-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$
- Gain(Outlook) = H(Outlook) – $\sum_{v \in Outlook} \frac{S_v}{S} H(Sv)$
- Gain(Outlook) = H(Outlook) – ($\frac{5}{14}$ H(Sunny) + $\frac{4}{14}$ H(Overcast) + $\frac{5}{14}$ H(Rain))

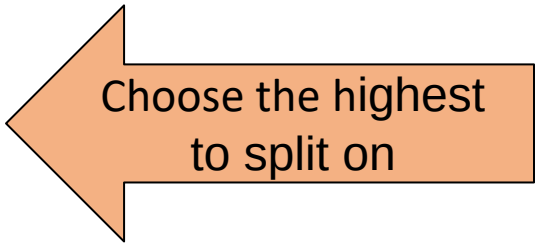
Similarly,

Note: Highest gain is always selected.

Gain(Humidity)=0.151

Gain(Outlook)=0.246

Gain(Wind)=0.048

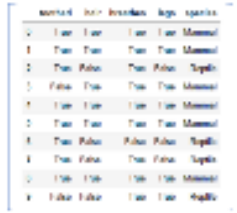


ID3 Algorithm

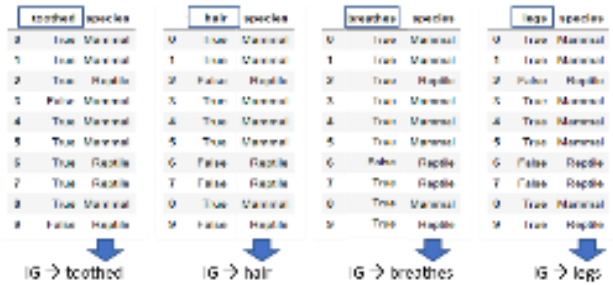
- Split (node, {examples}):
  1. A ← the best attribute for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. If examples perfectly classified: STOP
     else: iterate over new child nodes
     Split (child_node, {subset of examples})

1. Create a root node

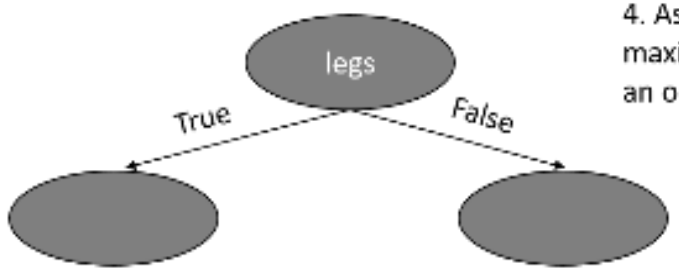
Entropy



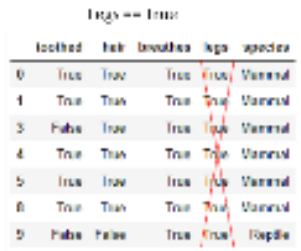
2. Calculate the entropy of the whole (sub) dataset

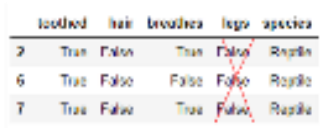


3. Calculate the Information gain of each single feature and pick that feature with the larges Information gain



4. Assign the (root) node the label of the feature with the maximum information gain. Grow for each feature value an outgoing branch and add unlabelled nodes at the end





5. Split the dataset along the values of the maximum information gain feature and remove this feature from the dataset

First sub tree

Second sub tree

6. For each of the sub_datasets, repeat steps 2 to 5 until a stopping criteria is satisfied →Here the recursion kicks in

Thanks