K6312 Information Mining & Analysis

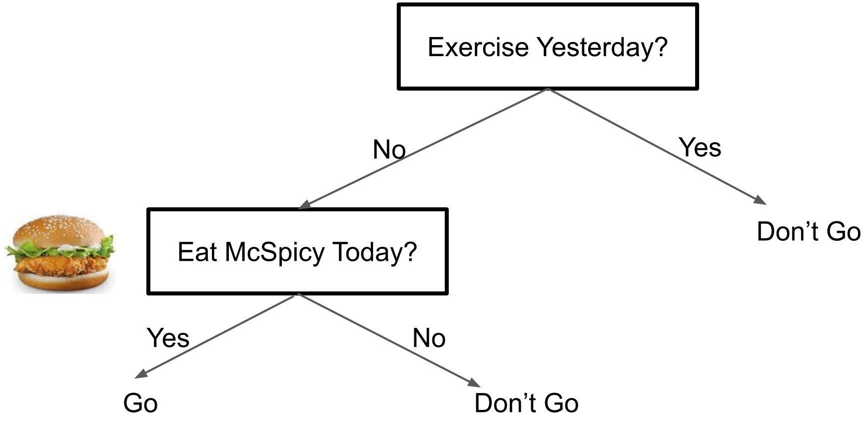
Chen Zhenghua & Zhao Rui

Decision Tree

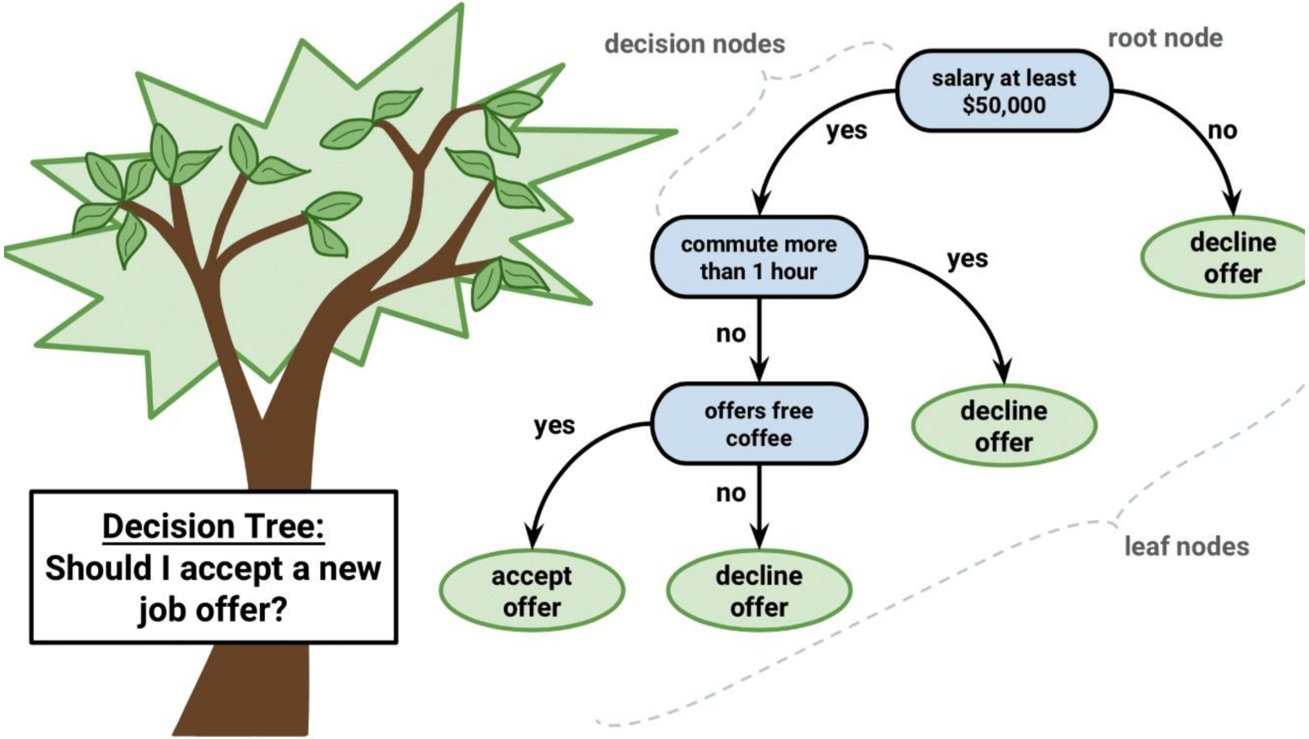
What is Decision Tree

- Decision Tree: a decision support tool that uses a tree-like model/graph of decisions and their possible consequences.

Should I do exercise today?



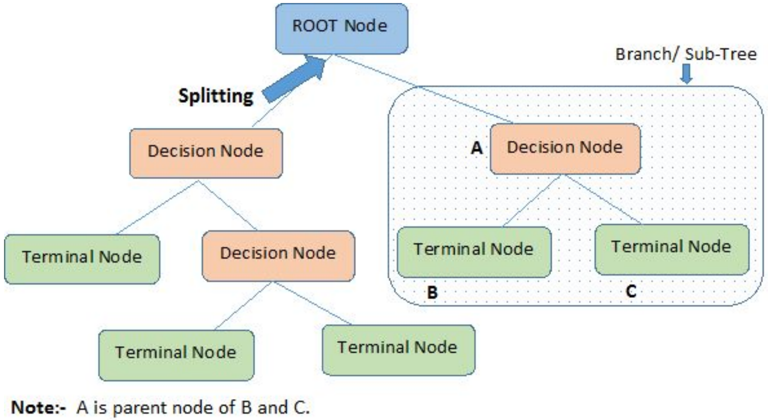
1. Top-down approach
2. Divide and Conquer



Source:
https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Terminology

- **Root Node:** represent entire population or sample, which will be further divided into multiple subsets.
- **Splitting:** a process of dividing a node into two or more sub-nodes.
- **Decision Node:** a sub-node that can be split into further sub-nodes.
- **Leaf/Terminal Node:** nodes do not split.
- **Branch/Sub-Tree:** a sub section of entire tree.
- **Pruning:** remove nodes (opposite of Splitting)
- **Parent and child Node:** a node which is divided into sub-nodes is called parent node of sub-nodes where sub-nodes are the child of parent node.



Source:
https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134

Function Approximation

Function Approximation

- Problem Setting:
  - Set of possible instances $\mathbb{X}$ (input space, feature space)
  - Unknown target function $f : \mathbb{X} \rightarrow \mathbb{Y}$
  - Set of function hypotheses $H = \{h | h : \mathbb{X} \rightarrow \mathbb{Y}\}$(hypothesis space)
- Input
  - Training examples $\{< \mathbf{x}^i, y^i >\}$ of unknown target function f
- Output
  - Hypothesis $h \in H$ that best approximates target function f

For Supervised Learning

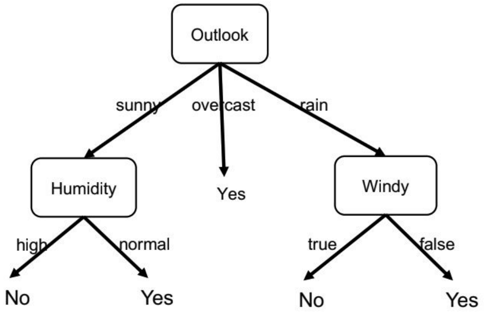
How to have chicken rice in Can 2 ASAP

- Problem Setting:
  - Set of possible instances
    - Each instance will be the day of week. E.g. <monday>, <sun.>
  - Unknown target function
    - Function can be the route you take that make you arrive at Can 2 ASAP
  - Set of function hypotheses
    - Each hypothesis may be a combination of taking bus or walking along nanyang avenue. E.g. {monday: bus, the rest: walk}. It may have 2^7 possibilities.
- Output:
  - The most optimal strategy: maybe weekdays by bus, weekends by walk
- Input
  - Training examples e.g., <Mon. take bus, 10mins>, <Sun, take bus 50mins>, <Sun, walk, 20mins>, <Tues., walk, 25mins>….

Function Approximation-Decision Tree

- Problem Setting:
  - Set of possible instances $\mathbb{X}$
    - Each instance x in X is a feature vector. E.g., <humidity: low, wind: weak, outlook:rain, temp: hot>
  - Unknown target function $f : \mathbb{X} \rightarrow \mathbb{Y}$
    - A decision tree map **x** to y
  - Set of function hypotheses $H = \{h | h : \mathbb{X} \rightarrow \mathbb{Y}\}$
    - Each hypothesis is a decision tree, which sorts x to leaf and assign its corresponding y
- Input
  - Training examples $\{< \mathbf{x}^i, y^i >\}$ of unknown target function f
- Output
  - Hypothesis $h \in H$ that best approximates target function f
    - The decision tree fit the data best

Outlook	Temperature	Humidity	Windy	Play
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*



Quick Questions

- Suppose X = [x1, x2, x3, x4] is a boolean valued vector
    - How would you represent x1 AND x2 AND x4 using decision tree?
    - How would you represent x1 OR x3 using decision tree?
    - Is decision tree able to represent every boolean function over any number of boolean variables?

Decision Tree: How to Build and Use

Can a “Good Tree” be automatically built?

- We can always come up with some decision tree for a dataset
  - Pick any feature not used above, brach on its values, recursively.
  - However starting with a random feature may lead to a large and complex tree.

- In general, we prefer short trees over larger and complex ones because a simple consistent hypothesis is more likely to be true

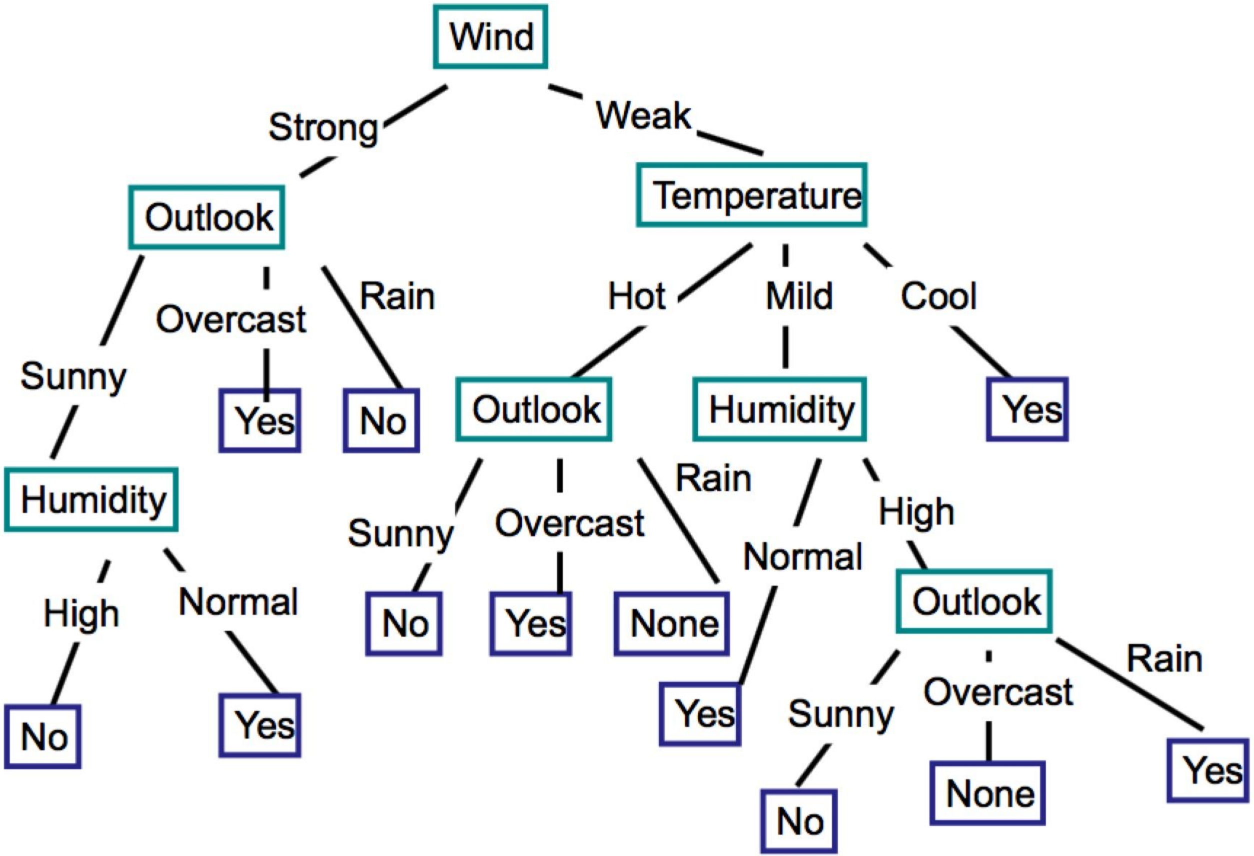
Toy Example: Play Tennis

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
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***

Features/
Attributes

Label

Poor Decision Tree



Algorithm for Building Decision Tree

node=root

Main Loop:

1. Decide the "best" attribute or feature (Say **A**) for next node;
2. For each value of **A**, create a new descendant of node;
3. Partition training examples to leaf nodes;
4. **IF** training examples are perfectly classified, **THEN STOP**; **ELSE** iterate over new leaf nodes

How to Select "Optimal" Attribute ?

Outlook	Temperature	Humidity	Windy	Play
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*

Learning Process

- Finding the smallest decision tree turns out to be intractable.
- However, there is a simple heuristics algorithm that does a good job of finding small trees.
- Inductive decision tree algorithm 3 (ID3)

ID3

- ID3 is a well-known decision tree algorithms that uses a top-down greedy search through the hypothesis space.
- ID3 was designed to handle large training sets with many attributes.
- ID3 tries to generate fairly simple trees, but is not guaranteed to produce the best one.

Which is the best attribute to split

- Imagine that we have examples for two classes P and N. How do we decide which attribute to split on?
- Let’s first take a look at some characteristic of tests
  - Let S contain 20 occurrences of P and 20 of N.
  - Imagine a Boolean test that splits the data into two subsets S1 and S2.
- Best case: S1 = 20P and S2 = 20N
- Worst case: S1 = 10P, 10N and S2 = 10P, 10N
- Intermediate case: S1 = 17P, 1N and S2 = 3P, 19N

- Why is the third case better than the second?
  - This third case is less chaos and more pure

Search Heuristic in ID3

- Central choice in ID3: Which attribute to test at each node in the tree?
    - The attribute that is most useful for classifying examples.

- Define a statistical property, called information gain, measuring how well a given attribute separates the training examples according to their target classification.

- First define a measure commonly used in information theory, called entropy, to characterize the (im)-purity of an arbitrary collection of examples

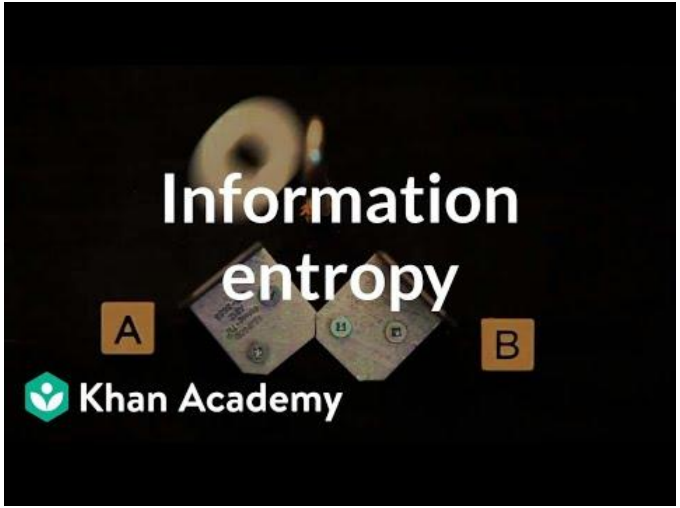
Entropy

- Entropy is the measure of the information in a set of examples.

$$Entropy = -\sum_{i=1}^{K} p_i log_2 p_i$$

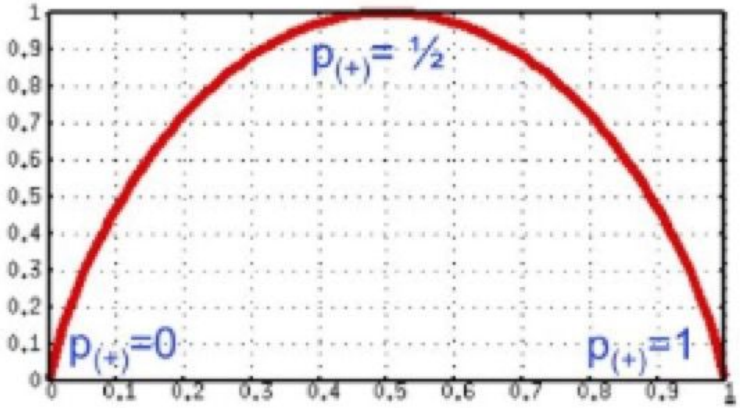
  - Where i={1,...,K}, K is the number of possible actions, pi is the proportion of each action i in the example set
  - For example: $Entropy([9*, 5+, 6-]) = -\frac{9}{20}log_2\frac{9}{20} - \frac{5}{20}log_2\frac{5}{20} - \frac{6}{20}log_2\frac{6}{20}$

- High Entropy: more information
- Low Entropy: less information



Entropy for binary case

- *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 of S

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

$$Entropy([9+, 5-]) = -\frac{9}{14} log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14}) = 0.94$$

Information Gain

- Entropy:

$$E(X) = -\sum_{i=1}^{K} p(X = X_i) log_2 p(X = X_i)$$

  - **Intuition**: uncertainty of X, information contained in X, expected information bits required to represent X.
- Conditional Entropy

$$E(X|Y) = \sum_{i=1} p(Y = Y_i) E(X|Y = Y_i)$$

  - **Intuition**: given y, how much uncertainty remains in X
- Mutual Information (Information Gain)

$$I(X,Y) = E(X) - E(X|Y) = E(Y) - E(Y|X)$$

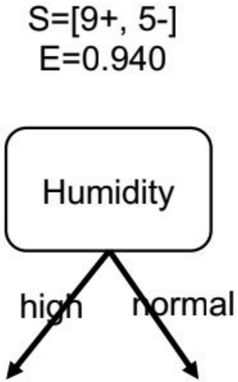
  - **Intuition**: how much knowing Y reduces uncertainty about X, and vice versa.

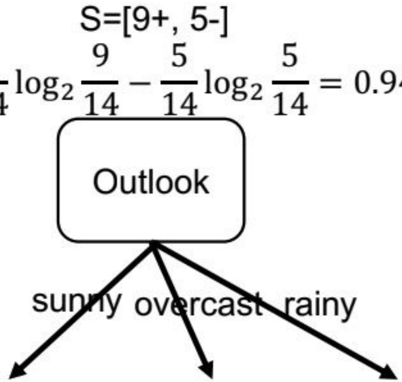
Information Gain Splitting

E(X)

E(X|Y)

I(X, Y)





$$E = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

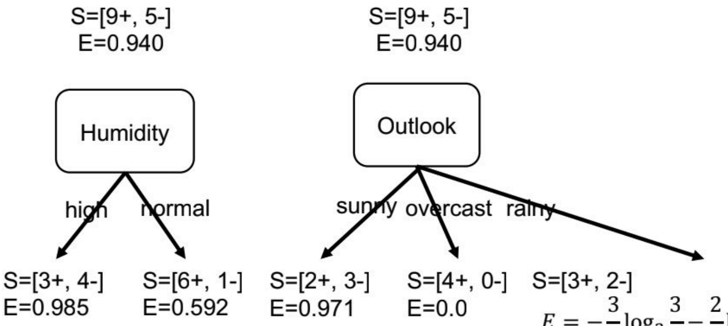
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy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

Information Gain Splitting

E(X)

E(X|Y)

I(X, Y)



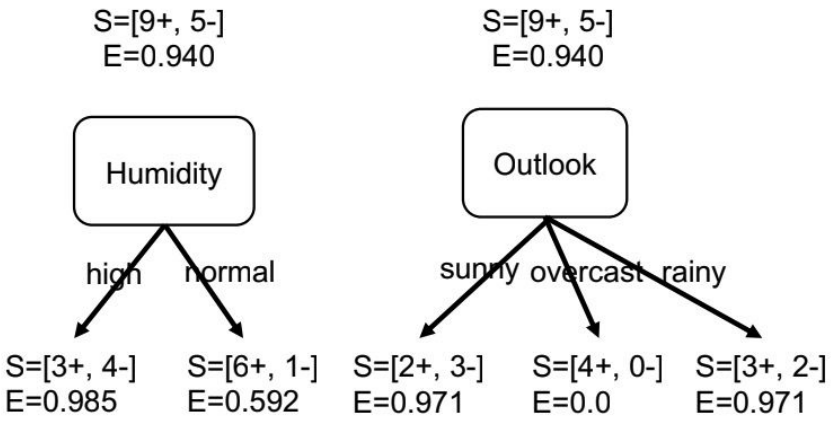
$$E = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

Outlook	Temperature	Humidity	Windy	Play
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*

Information Gain Splitting

E(X)

E(X|Y)



I(X, Y)

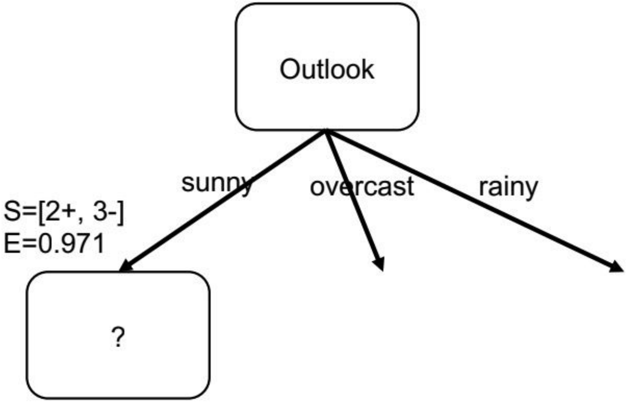
I=0.940-0.5*0.985-0.5*0.592
=0.151

I=0.940-0.357*0.971*2
=0.247

Outlook wins over Humidity

Outlook	Temperature	Humidity	Windy	Play
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*

Exercise

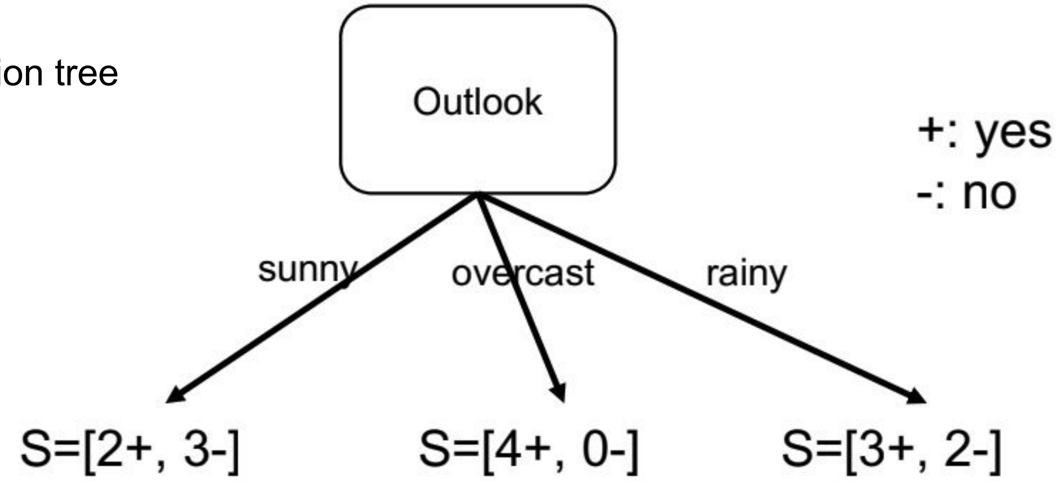


Next attributes to split ?

Outlook	Temperature	Humidity	Windy	Play
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***

Prediction with Decision Tree

Use outlook only to build the decision tree



Classify be majority vote: no yes yes

How about Continuous Attributes?

- Real-valued attributes can, in advance, be discretized into ranges, such as big, medium, small.

- Alternatively, for continuous attribute A, one can develop splitting nodes based on thresholds of the form $A<c$ that partition the examples into those with $A<c$ and those with $A>=c$.
  - The information gain of such splits are easily computed and compared to splits on discrete features in order to select the best split.

Continuous Value Attribute

Personal GPS tracking in 30 seconds window

Total distance	Average accuracy	Average speed	Average acceleration	Mode
0	10	0	0	stop
0	10	0.02	0	stop
0	14	0.04	0	stop
0	10	0	0	stop
94.721062	60	3.559097	-0.213634	stop
13.798621	10	0.446895	-0.068557	walk
0	10	0	0	walk
5.02672	10	0.201069	0	walk
29.51251	10	0.953268	0.046126	walk
18.448198	18	0.798536	-0.040226	walk
93.197034	77.5	4.434993	-0.378918	walk
63.604663	37	2.450164	0.035747	walk

- Defining a discrete attribute that partitions the continuous attribute value into a discrete set of intervals
  - Speeds in the sample: [3,0.5, 10, 4,0.3,0.1,1]
  - Speed < 3?:
    - [0.5, 0.3, 0, 2,1]
    - [3, 10, 4]

Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play (Tennis)
D1	Sunny	85	85	Weak	No
D2	Sunny	80	90	Strong	No
D3	Overcast	83	86	Weak	Yes
D4	Rain	70	96	Weak	Yes
D5	Rain	68	80	Weak	Yes
D6	Rain	65	70	Strong	No
D7	Overcast	64	65	Strong	Yes
D8	Sunny	72	95	Weak	No
D9	Sunny	69	70	Weak	Yes
D10	Rain	75	80	Weak	Yes
D11	Sunny	75	70	Strong	Yes
D12	Overcast	72	90	Strong	Yes
D13	Overcast	81	75	Weak	Yes
D14	Rain	71	91	Strong	No

For Continuous Attributes

- The single threshold with the highest gain for a set of data can be found by examining the following choices.

```
for (each continuous feature A){
   Sort the examples according to their value for A;
   for (each ordered pair, X_i, X_i+1, in the sorted list)
      if (the category of X_i and X_i+1 are different)
          find the midpoint of X_i and X_i+1 denoted as c_i  to
          define threshold A < c_i
}
```

Temp	64	65	68	69	70	71	72	72	75	75	80	81	83	85
Class	+	-	+	+	+	-	-	+	+	+	-	+	+	-

Thresholds for Temp 64.5, 66.5, 70.5, 72, 77.5 80.5, 84

Attributes with Many Values

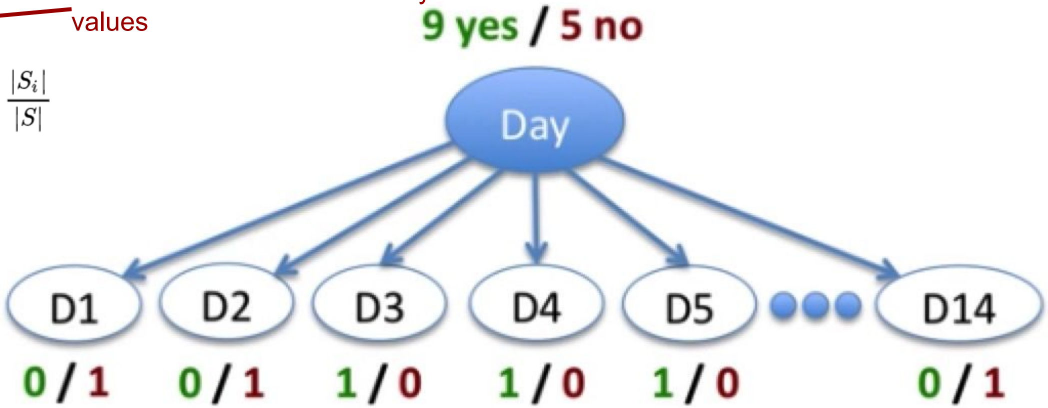
- Information Gain will bias toward the attribute with many values
- For example, using day as the attribute, the data will be perfectly splitted into subsets of size 1.
- However, it won't work for new data.
- Use GainRatio instead of information gain as criteria:

$$GainRatio(S, A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$$

Penalizes attributes with many values

$$SplitInformation(S, A) = -\sum_{i \in Values(A)} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|}$$

A candidate attribute
i possible values of A
S set of examples {X}
S_i subset where X_A = i



Other Criterions

- Gini Impurity (corresponding to entropy)
  - Suppose we
    - Randomly pick a datapoint in our dataset, then
    - **Randomly classify it according to the class distribution in the dataset.**
    - The probability we classify the data point incorrectly is Gini Impurity

Decision Tree: Regression

Decision Tree Regression

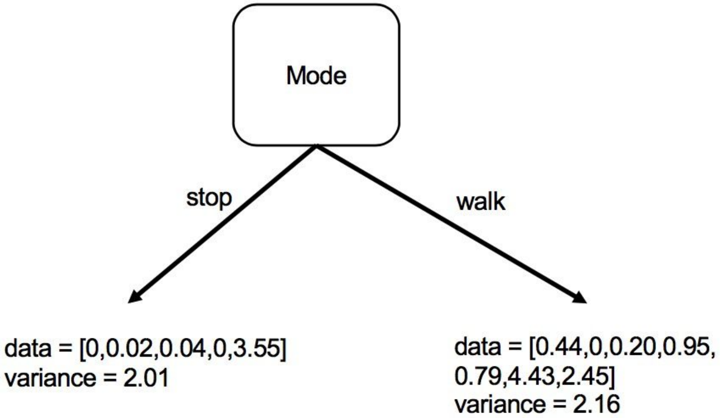
- Everything learned from the decision tree classification is the same except:
  - Use variance reduction as splitting criterion
  - Use aggregate statistic as prediction output

Target

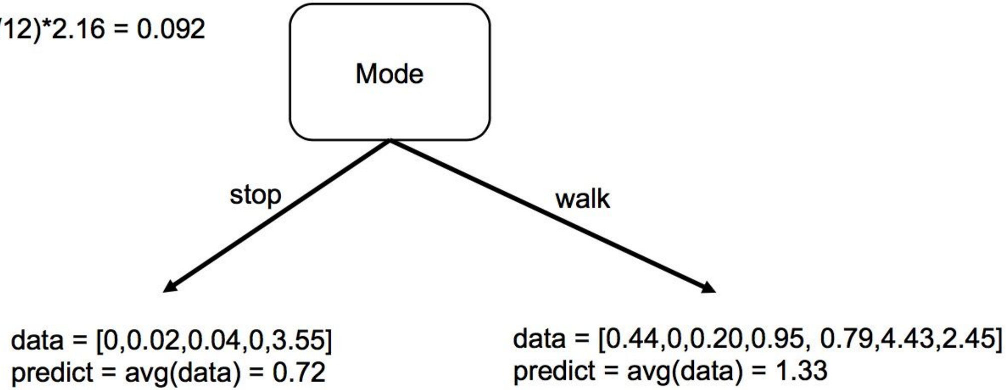
Total distance	Average accuracy	Average speed	Average acceleration	Mode
0	10	0	0	stop
0	10	0.02	0	stop
0	14	0.04	0	stop
0	10	0	0	stop
94.721062	60	3.559097	-0.213634	stop
13.798621	10	0.446895	-0.068557	walk
0	10	0	0	walk
5.02672	10	0.201069	0	walk
29.51251	10	0.953268	0.046126	walk
18.448198	18	0.798536	-0.040226	walk
93.197034	77.5	4.434993	-0.378918	walk
63.604663	37	2.450164	0.035747	walk

Example

data = [0,0.02,0.04,0,3.55,0.44,0,0.20,
0.95,0.79,4.43,2.45]



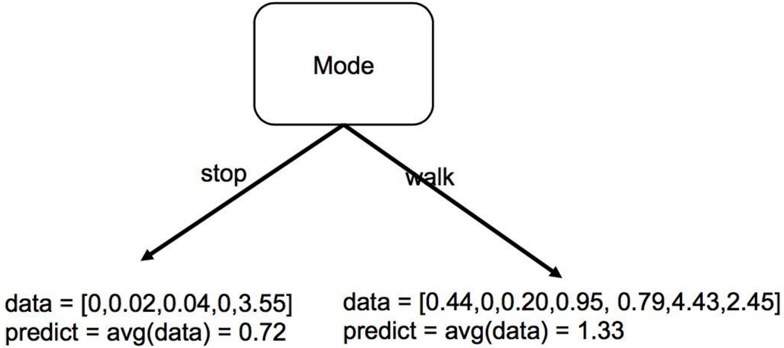
variance reduction: 2.19 – (5/12)*2.01 – (7/12)*2.16 = 0.092



Problem of Decision Tree Regression

[distance, accuracy, acceleration, mode] -> speed

Use average value as the stat used



Cannot predict values outside the historically observed range

Total distance	Average accuracy	Average speed	Average acceleration	Mode
0	10	0	0	stop
0	10	0.02	0	stop
0	14	0.04	0	stop
0	10	0	0	stop
94.721062	60	3.559097	-0.213634	stop
13.798621	10	0.446895	-0.068557	walk
0	10	0	0	walk
5.02672	10	0.201069	0	walk
29.51251	10	0.953268	0.046126	walk
18.448198	18	0.798536	-0.040226	walk
93.197034	77.5	4.434993	-0.378918	walk
63.604663	37	2.450164	0.035747	walk

Decision Tree Learning

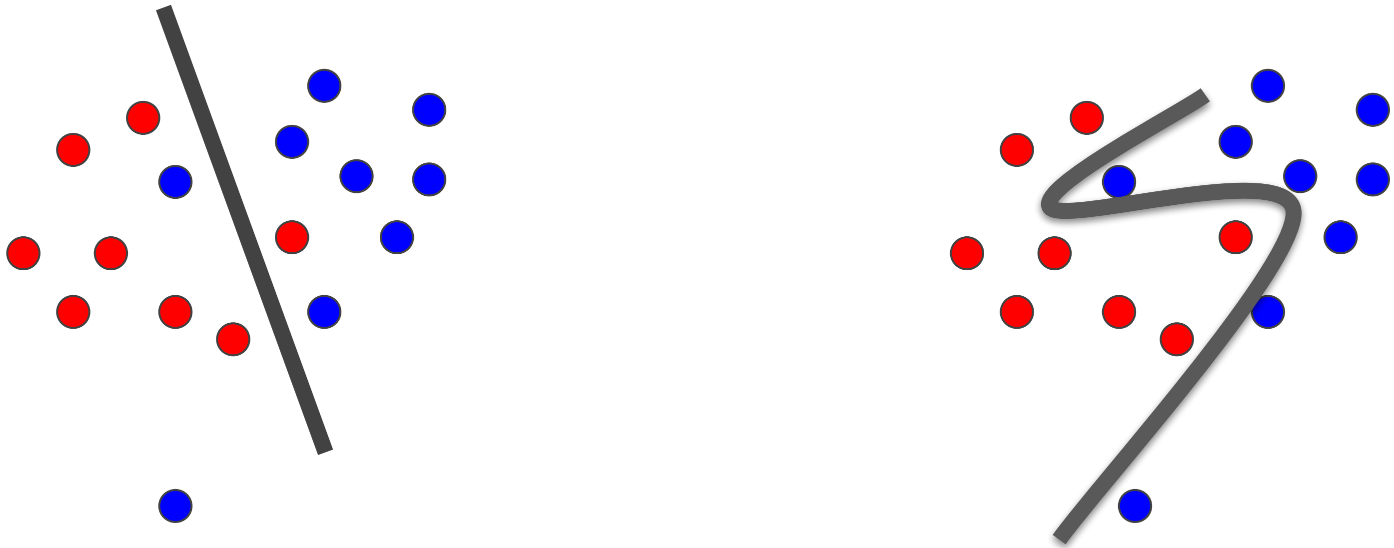
- Pros
  - Easy to understand: decision tree output is very easy to understand
  - Data exploration: feature selection
  - Less data cleaning required: not influenced by scale and missing values to a fair degree
  - Data type is not a constraint: handle both numerical and categorical variables
  - Non parametric method: no assumptions about the space distribution and the classifier structure
  - In nature, can handle multiclass directly
- Cons
  - Overfitting: overfitting is one of the most practical difficulty for decision tree models
  - Not ideal for continuous variables: while working with continuous numerical variables, decision tree lost information when it categorizes variables in different categories.

Decision Tree: Overfitting

Generalization

- In ML, a model is used to fit the data
- Once trained, the model is applied upon new data
- Generalization is the prediction capability of the model on live/new data

Which model is better?



SPAM VS **Not SPAM**

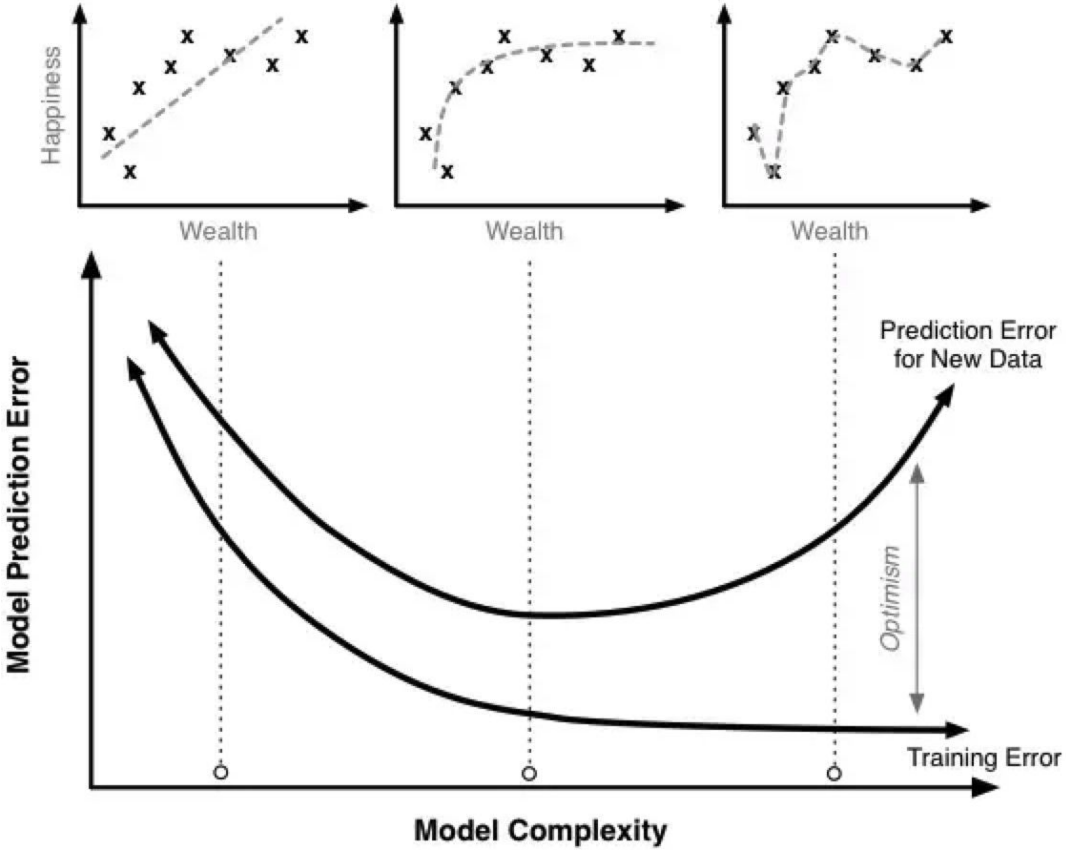
Model Complexity

- Complex model easily overfits the training data

- Then, the trained model is unable to generalize on testing data

- overfitting vs underitting
    - overfitting: small training error but large testing error
    - underfitting: large training and testing errors

Model Complexity



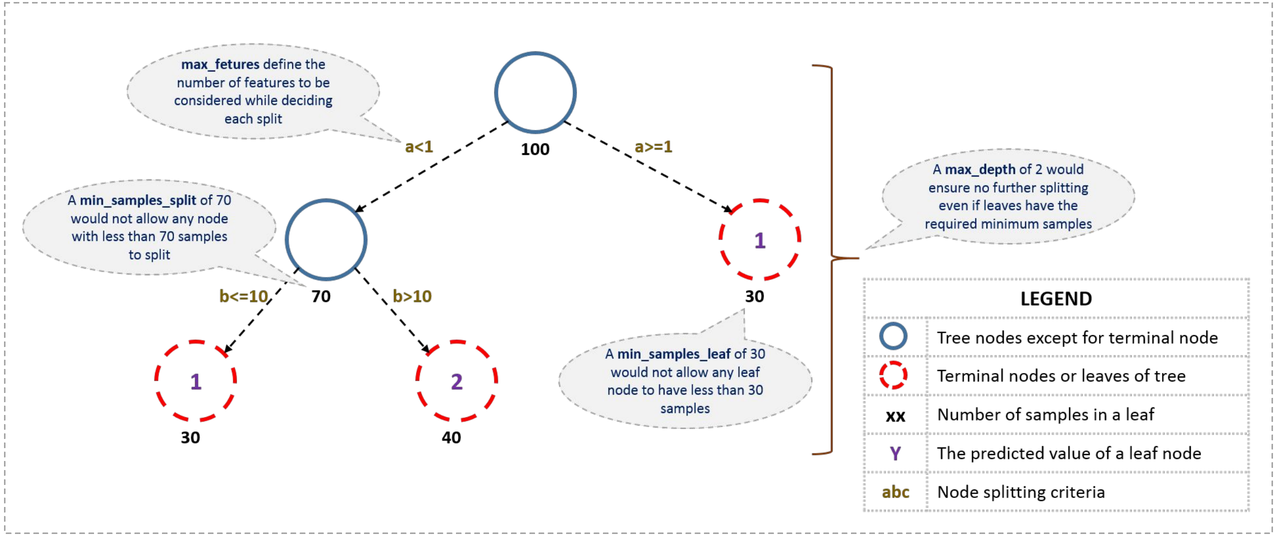
source: stackoverflow

Overfitting for Decision Tree

- Is it a good idea to grow the full tree all the time?
  - Suppose we add one noisy training sample: [sunny, hot, normal, true, no]
  - What effect on earlier tree?
- There may exist multiple trees that perform exactly the same, then which one should we select?
  - General Principle: **prefer the simplest hypothesis that fits the data**

How to Avoid Overfitting I

- Stop growing the tree given stopping criterions
  - Minimum samples for a node split
  - Minimum samples for a terminal node (leaf)
  - Maximum depth of tree (vertical depth)
  - ….



source:https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

How to Avoid Overfitting II

- Grow a “full” tree, and then to perform post-pruning
  - Split data into training and validation set
  - Build a full tree that classify training data
  - Do until further pruning is harmful, greedily remove the one that most improves validation set accuracy.