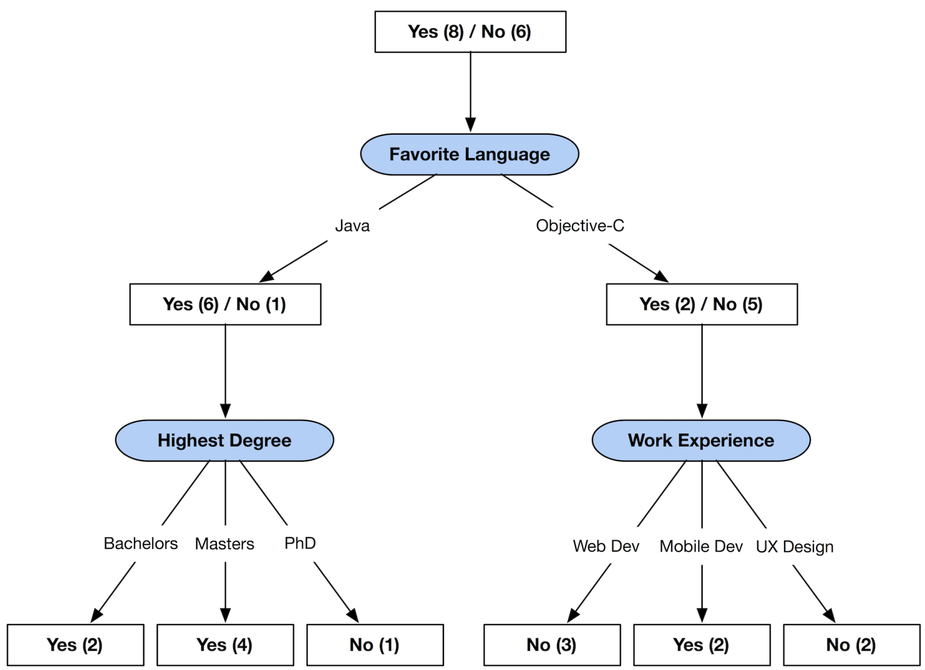
Artificial Intelligence

Machine Learning

Tree classifiers



Outline

1. Tree classifiers: Definition & History
2. A toy example
3. Splitting criteria in C4.5
4. Numerical features
5. Pruning strategies
6. Alternative method: CART
7. Practical considerations
8. Tree classifiers: Pros & cons

Tree Classifiers

- Popular classification methods.
- Easy to understand, simple algorithmic approach.
- No assumption about linearity.
- **History:**
  - CART (Classification And Regression Trees): Friedman 1977.
  - ID3 and C4.5 family: Quilan 1979-1983.
  - Refinements in mid 1990's (e.g., pruning, numerical features etc.).
- **Applications:**
  - Botany (e.g., New Flora of the British Isles Stace 1991).
  - Medical research (e.g., Pima Indian diabetes diagnosis, early diagnosis of acute myocardial infarction).
  - Computational biology (e.g., interaction between genes)

Tree Classifiers

- The terminology **Tree** is graphic.
- However, a decision tree is grown from the root downward. The idea is to send the examples down the tree, using the concept of information entropy.

Tree Classifiers

- The terminology **Tree** is graphic.
- However, a decision tree is grown from the root downward. The idea is to send the examples down the tree, using the concept of information entropy.
- **General Steps to build a tree:**

  1. Start with the root node that has all the examples.
  2. Greedy selection of the next best feature to build the branches. The splitting criteria is *node purity*.
  3. Class majority will be assigned to the leaves.

Classification

Given: Training data:

$$(x_1, y_1), \ldots, (x_n, y_n)$$

Where $x_i \in \mathbb{R}^d$ and y_i is discrete (categorical/qualitative), $y_i \in \mathbb{Y}$.

Example $\mathbb{Y} = \{-1, +1\}, \mathbb{Y} = \{0, 1\}$.

Task: Learn a classification function:

$$f : \mathbb{R}^d \longrightarrow \mathbb{Y}$$

Classification

Given: Training data:

$$(x_1, y_1), \ldots, (x_n, y_n)$$

Where $x_i \in \mathbb{R}^d$ and y_i is discrete (categorical/qualitative), $y_i \in \mathbb{Y}$.

Example $\mathbb{Y} = \{-1, +1\}, \mathbb{Y} = \{0, 1\}$.

Task: Learn a classification function:

$$f : \mathbb{R}^d \longrightarrow \mathbb{Y}$$

In the case of Tree Classifiers:

1. No need for $x_i \in \mathbb{R}^d$, so no need to turn categorical features into numerical features.
2. The model is a tree.

Toy example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

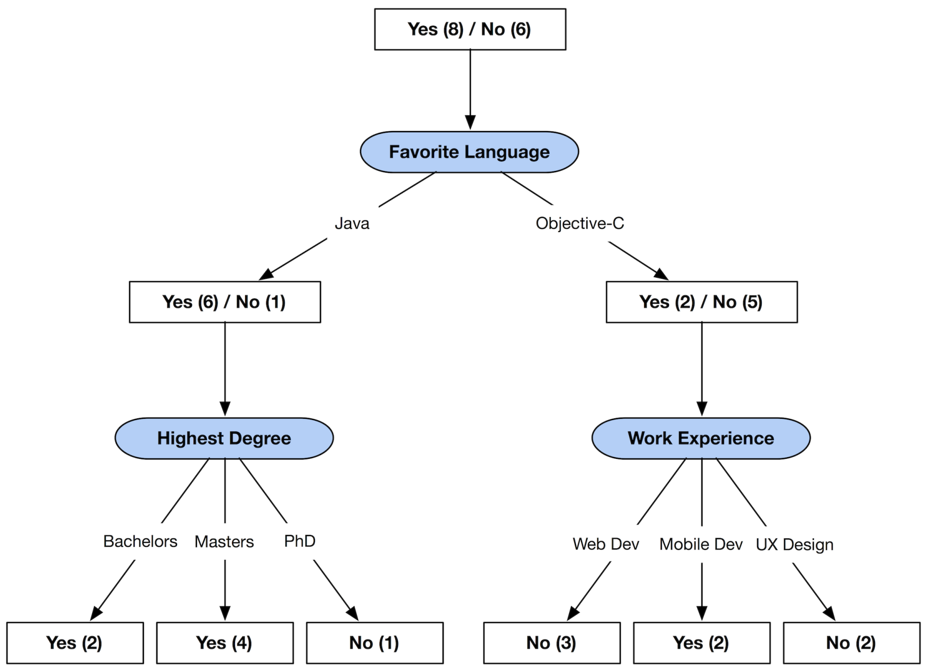
Toy example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

Toy example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no



Splitting criteria in C4.5

1. The central choice is selecting the next attribute to split on.
2. We want some criteria that measures the homogeneity or impurity of examples in the nodes:

Splitting criteria in C4.5

1. The central choice is selecting the next attribute to split on.
2. We want some criteria that measures the homogeneity or impurity of examples in the nodes:
   (a) Quantify the mix of classes at each node.
   (b) Maximum if equal number of examples from each class.
   (c) Minimum if the node is pure.

Splitting criteria in C4.5

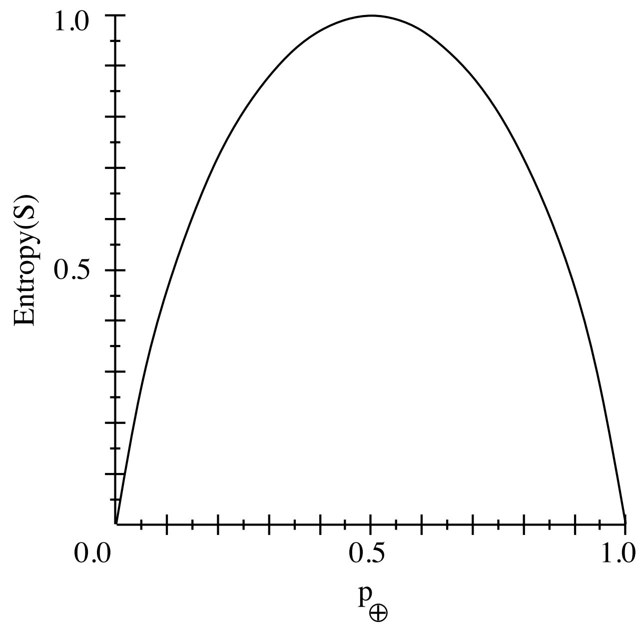
1. The central choice is selecting the next attribute to split on.
2. We want some criteria that measures the homogeneity or impurity of examples in the nodes:
   (a) Quantify the mix of classes at each node.
   (b) Maximum if equal number of examples from each class.
   (c) Minimum if the node is pure.
3. A perfect measure commonly used in *Information Theory*:

$$\boxed{\text{Entropy(S)} = \text{-}\ p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}}$$

$p_{\oplus}$ is the proportion of positive examples.
$p_{\ominus}$ is the proportion of negative examples.

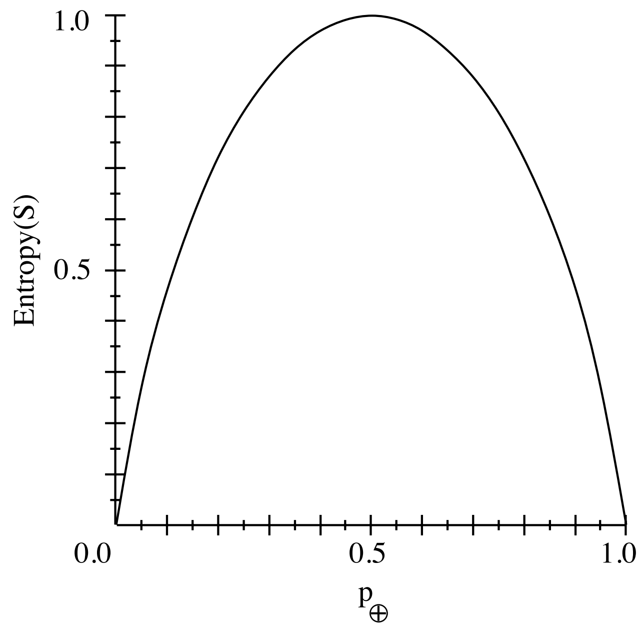
Splitting criteria in C4.5

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



Splitting criteria in C4.5

$$\boxed{\text{Entropy(S)} = -\ p_{\oplus}\ \log_2 p_{\oplus}\ -\ p_{\ominus}\ \log_2 p_{\ominus}}$$



In general, for c classes:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Splitting Criteria in C4.5

- Now each node has some entropy that measures the homogeneity in the node.
- How to decide which attribute is best to split on based on entropy?

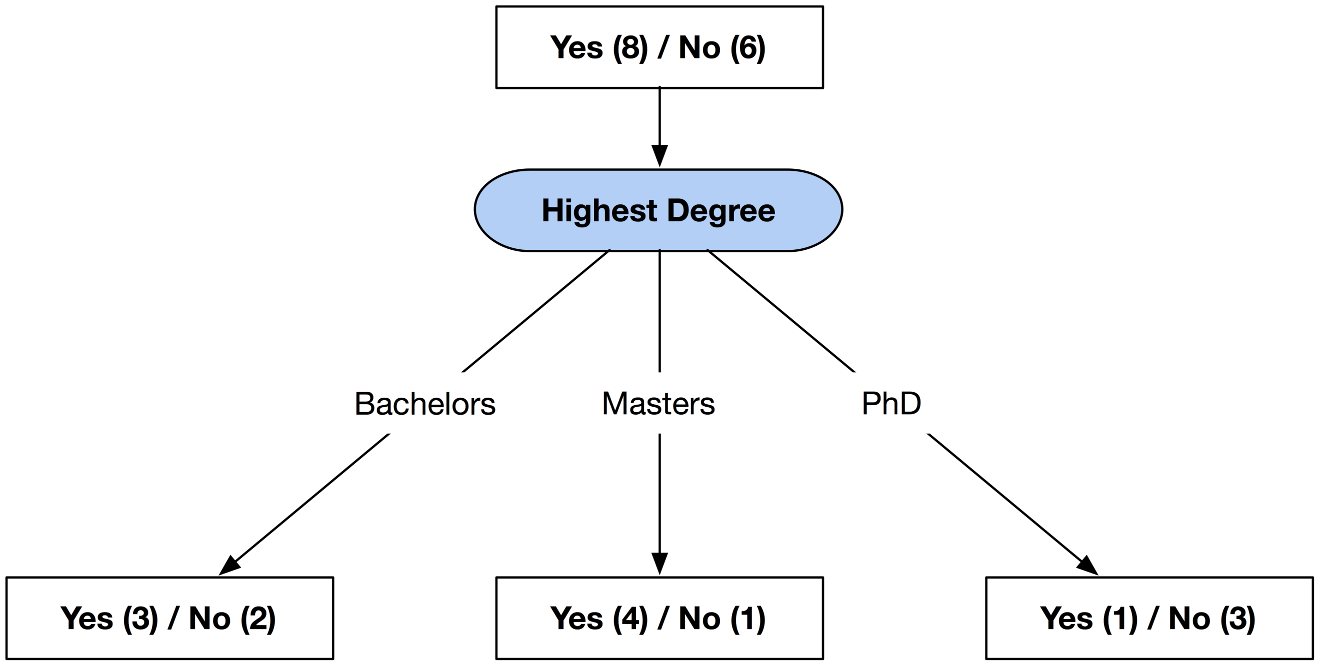
Splitting Criteria in C4.5

- Now each node has some entropy that measures the homogeneity in the node.
- How to decide which attribute is best to split on based on entropy?
- We use **Information Gain** that measures the expected reduction in entropy caused by partitioning the examples according to the attributes:

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

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985

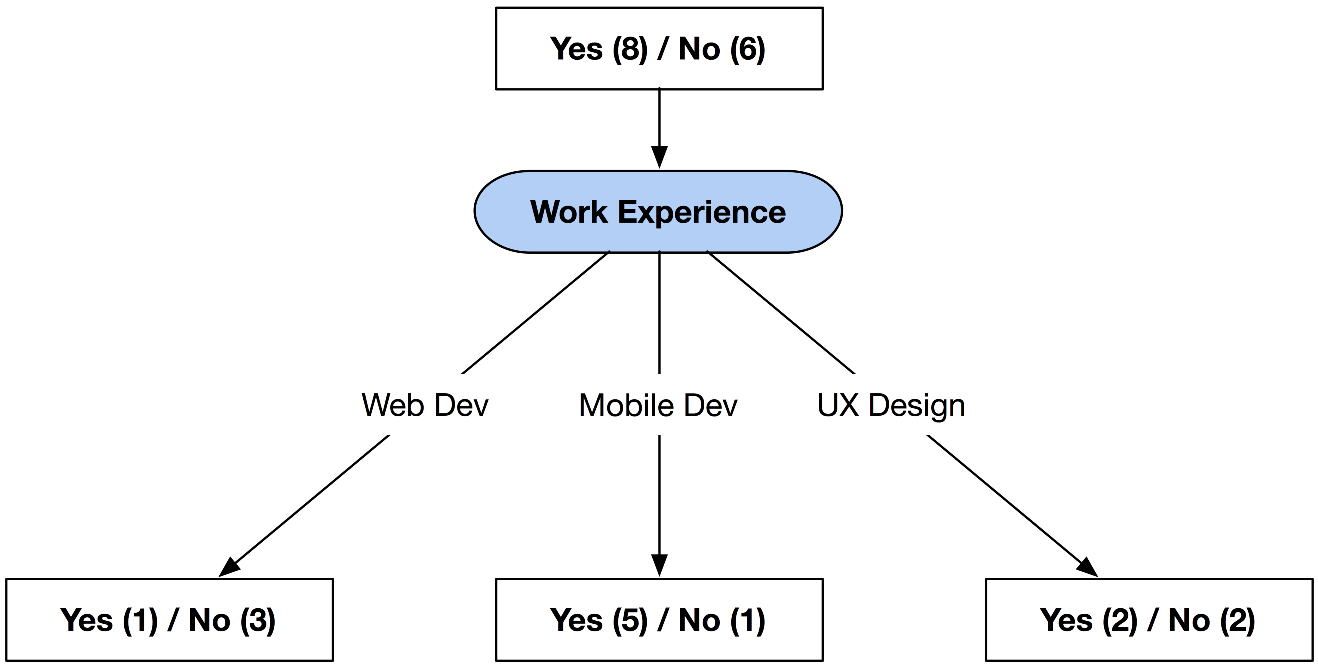


Entropy(3, 2) = 0.971 Entropy(4, 1) = 0.722 Entropy(1, 3) = 0.811

Gain(S, Highest Degree) = 0.985 – (5/14) x 0.971 – (5/14) x 0.722 – (4/14) x 0.811 = 0.149

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(1, 3) = 0.811

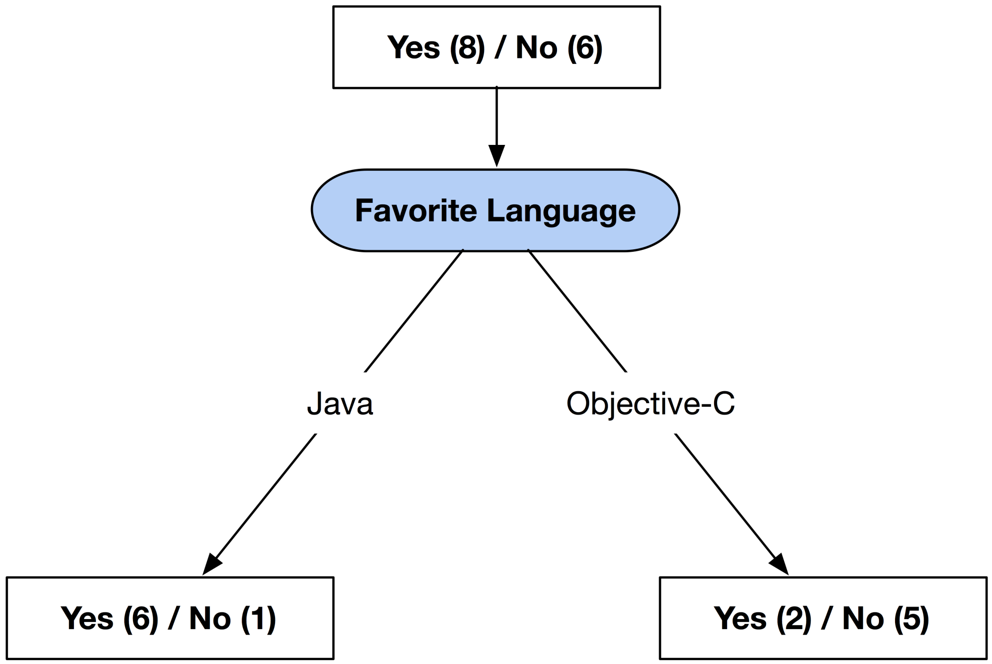
Entropy(5, 1) = 0.650

Entropy(2, 2) = 1.000

Gain(S, Work Experience) = 0.985 – (4/14) x 0.811 – (6/14) x 0.650 – (4/14) x 1.000 = 0.189

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



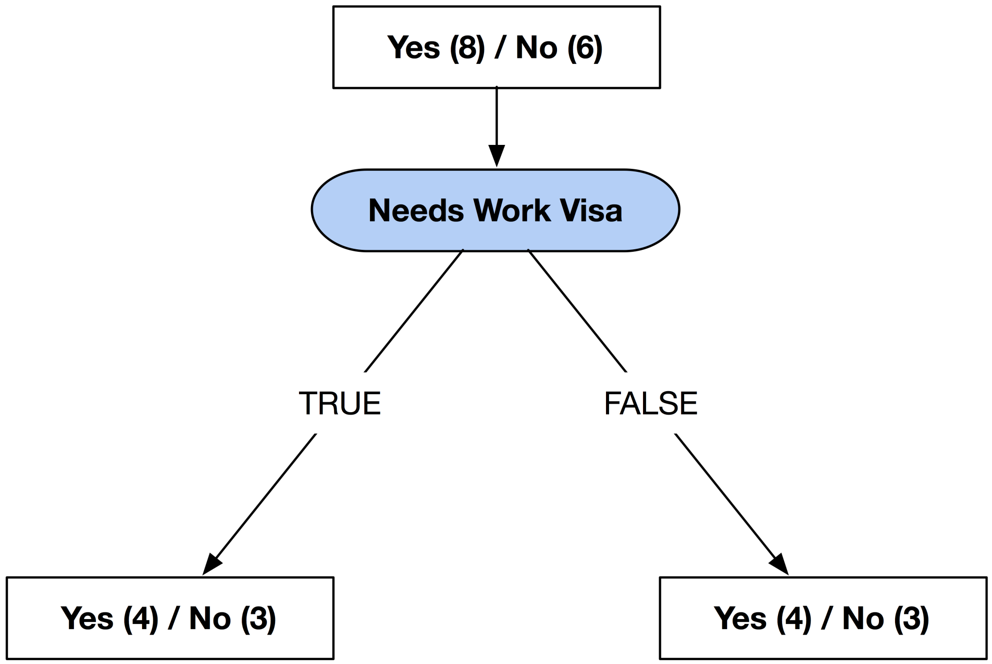
Entropy(6, 1) = 0.592

Entropy(2, 5) = 0.863

Gain(S, Favorite Language) = 0.985 – (7/14) x 0.592 – (7/14) x 0.863 = 0.258

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



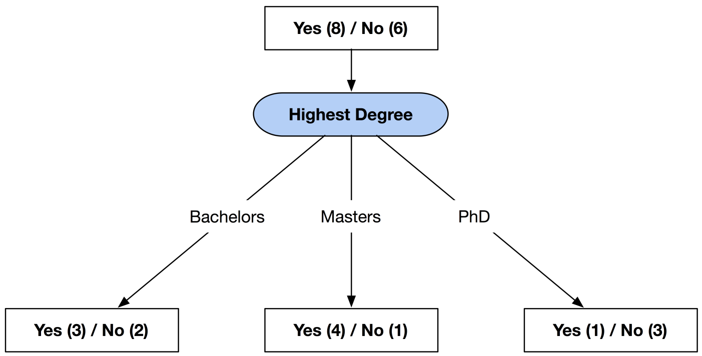
Entropy(4, 3) = 0.985

Entropy(4, 3) = 0.985

Gain(S, Needs Work Visa) = 0.985 – (7/14) x 0.985 – (7/14) x 0.985 = 0.000

Back to the example

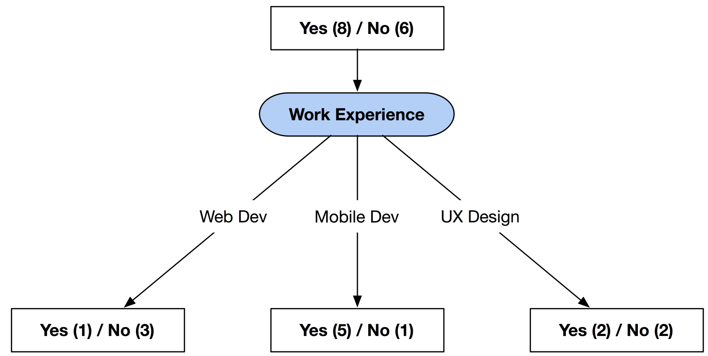
Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(3, 2) = 0.971 Entropy(4, 1) = 0.722 Entropy(1, 3) = 0.811

Gain(S, Highest Degree) = 0.985 – (5/14) x 0.971 – (5/14) x 0.722 – (4/14) x 0.811 = 0.149

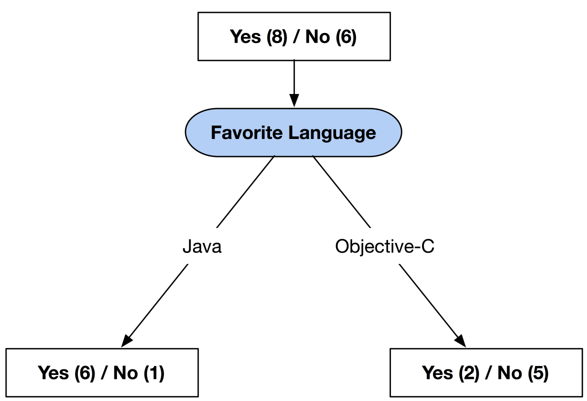
Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(1, 3) = 0.811 Entropy(5, 1) = 0.650 Entropy(2, 2) = 1.000

Gain(S, Work Experience) = 0.985 – (4/14) x 0.811 – (6/14) x 0.650 – (4/14) x 1.000 = 0.189

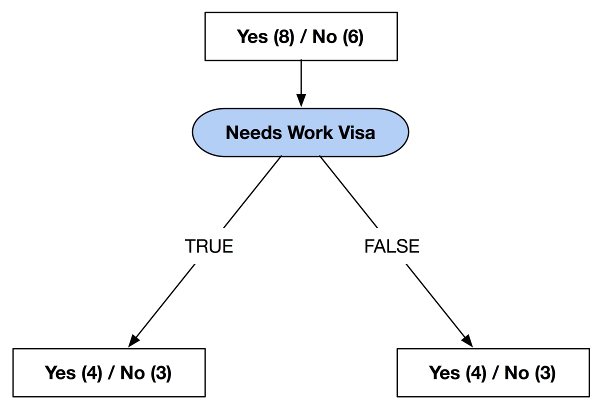
Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(6, 1) = 0.592 Entropy(2, 5) = 0.863

Gain(S, Favorite Language) = 0.985 – (7/14) x 0.592 – (7/14) x 0.863 = 0.258

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(4, 3) = 0.985 Entropy(4, 3) = 0.985

Gain(S, Needs Work Visa) = 0.985 – (7/14) x 0.985 – (7/14) x 0.985 = 0.000

Back to the example

Feature	Information Gain
Highest Degree	0.149
Work Experience	0.189
Favorite Language	**0.258**
Needs Work Visa	0.000

At the first split starting from the root, we choose the attribute that has the max gain.

Then, we re-start the same process at each of the children nodes (if node not pure).

Numerical features

Papers Published	Years of Work	Grade Point Average	Needs Work Visa	Hire
0 paper(s)	5 year(s)	3.20	TRUE	yes
5 paper(s)	1 year(s)	3.64	FALSE	yes
4 paper(s)	6 year(s)	2.92	TRUE	yes
10 paper(s)	4 year(s)	4.00	TRUE	yes
12 paper(s)	3 year(s)	3.21	TRUE	no
0 paper(s)	8 year(s)	3.37	TRUE	no
0 paper(s)	5 year(s)	4.00	FALSE	yes
8 paper(s)	3 year(s)	2.59	FALSE	no
0 paper(s)	7 year(s)	3.70	FALSE	yes
4 paper(s)	7 year(s)	3.78	TRUE	no
2 paper(s)	9 year(s)	4.00	FALSE	yes
9 paper(s)	4 year(s)	4.00	FALSE	no
7 paper(s)	4 year(s)	2.71	TRUE	yes
0 paper(s)	2 year(s)	3.03	FALSE	no

Overfitting the data

Pruning strategies

To get suitable tree sizes and avoid overfitting:

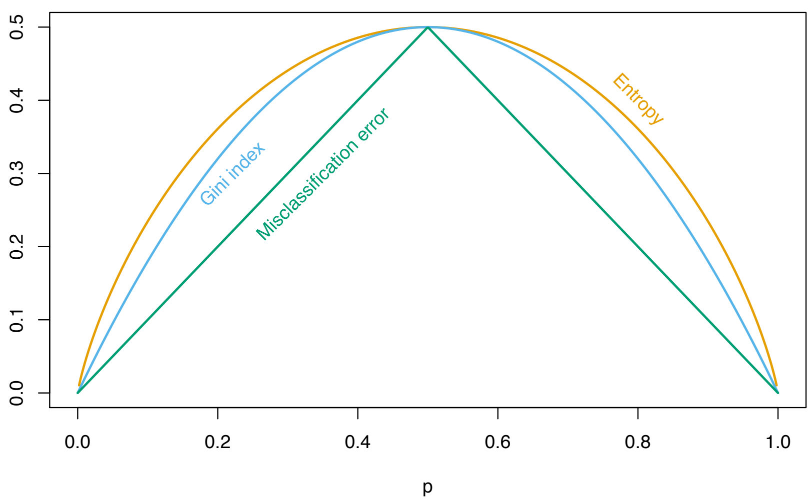
- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training examples. (difficult to know when to stop!).
- Grow a complex tree then to prune it back (Best strategy found).

  1. Use a validation set / Cross validation to evaluate the utility of post-pruning (remove a subtree if the performance of the new tree is no worse than the original tree).
  2. Rule post pruning.

CART

- Adopt same greedy, top-down algorithm.
- Binary splits instead of multiway splits.
- Uses Gini Index instead of information entropy.

$$Gini = 1 - p_{\oplus}^2 - P_{\ominus}^2$$



Practical considerations

1. Consider performing dimensionality reduction beforehand to keep the most discriminative features.

2. Use ensemble methods. E.g., Random Forest, have a great performance.*

3. Balance your dataset before training to prevent the tree from creating a tree biased toward the classes that are dominant.
   - Under-sampling: reduce the majority class
   - Over-sampling: Synthetic data generation for the minority class (e.g., SMOTE, and ADASYN).

**An Empirical Comparison of Supervised Learning Algorithms Rich by Caruana and Alexandru Niculescu-Mizil. ICML 2006.*

Tree classifiers: Pros & Cons

+ Intuitive, interpretable (but...).

+ Can be turned into rules.

+ Well-suited for categorical data.

+ Simple to build.

+ No need to scale the data.

- Unstable (change in an example may lead to a different tree).

- Univariate (split one attribute at a time, does not combine features).

- A choice at some node depends on the previous choices.

- Need to balance the data.

Credit

- The elements of statistical learning. Data mining, inference, and prediction. 10th Edition 2009. T. Hastie, R. Tibshirani, J. Friedman.
- Machine Learning 1997. Tom Mitchell.