Decision Tree Model

Decision Tree

- In principle, there are exponentially many decision trees that can be constructed from a given set of attributes.
- Finding the optimal tree (in terms of performance) is computationally infeasible because of the exponential size of the search space.
- Efficient algorithms have been developed to induce a reasonably accurate although suboptimal decision tree in a reasonable amount of time.
- These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decision about which attribute to use for partitioning the data.
- One such algorithm is Hunt’s algorithm, which is the basis of many decision tree induction algorithms, including ID3, C4.5, and CART.

Hunt's Algorithm

- In this algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets.
- Let D_t be the set of training records that are associated with node t and y={y_1, y_2, …, y_c} be the class labels. The following is a recursive definition of Hunt's algorithm:
  - Step 1: If all the records in D_t belong to the same class y_t, then t is a leaf node labelled as y_t.
  - Step 2: If D_t contains records that belong to more than one class, an **attribute test condition** is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes.
  - The algorithm is then recursively applied to each child node.

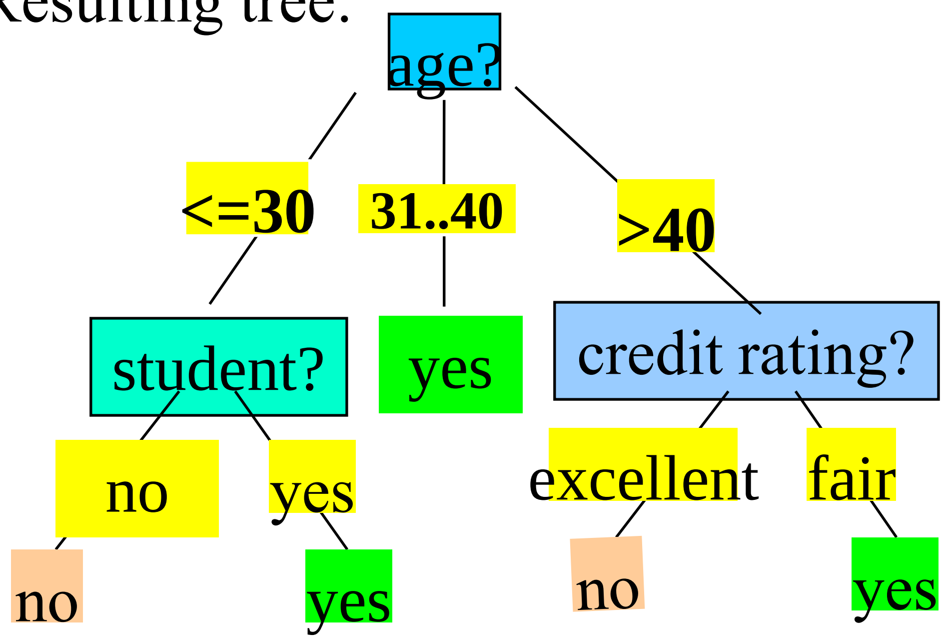
Decision Tree Induction: ID3

- Training data set: Buys_computer
- Decision tree model is constructed using the Quinlan ID3 algorithm.
- ID3 refers to Iterative Dichotomizer 3, and is developed by Ross Quinlan
- It iteratively dichotomizes (divides) the data using a top down greedy approach.
- The algorithm optimizes locally at each iteration.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree Induction: ID3

- Root is identified first, then the parents of next label, and so on of the decision tree are identified using entropy theory based optimization function
- Resulting tree:



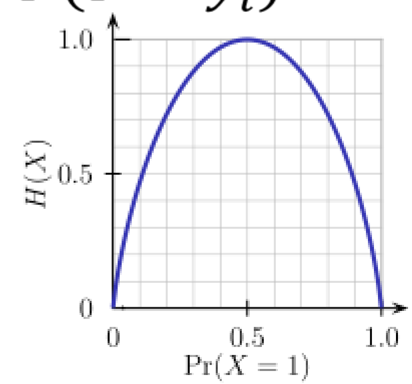
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Algorithm of ID3

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable *Y* taking *m* distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty

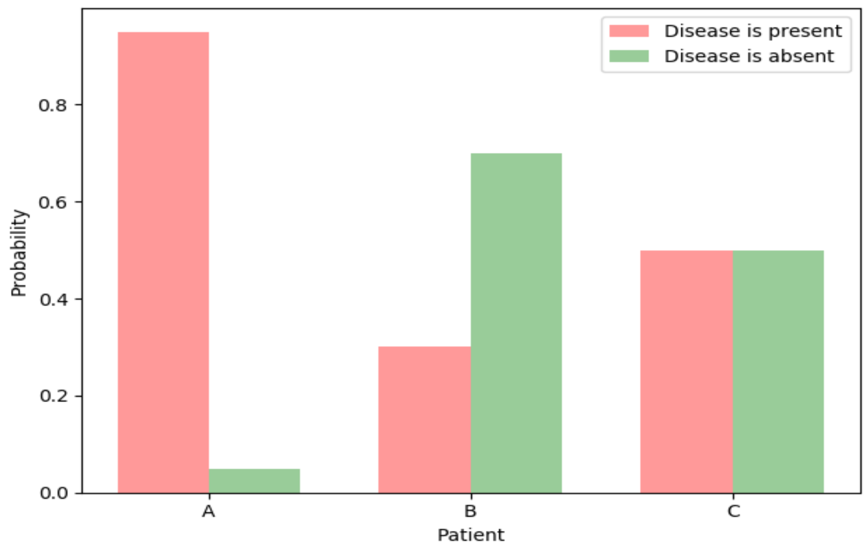


- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$

Y/(X=x1)	Y/(X=x2)	....	Y/(X=x)	Y/(X=x_n)

Entropy and uncertainty

- Suppose three patients have completed a medical test which, yields one of two possible results: the disease is either present or absent.
- Let Patient A has 95% chance that he has the disease. For Patient B and C it is 30% and 50%, respectively.
- So, if A, B, and C are in waiting room of a doctor's office, then the uncertainty of the waiting room is:



Entropy and uncertainty

- All other things being equal, which of the three patients is confronted with the greatest degree of uncertainty?
- I think the answer is clear: patient C.
- Compare this with patient A. Patient A is experiencing little uncertainty with regard to his medical prospects.
- Intuitively speaking, uncertainty of patient B falls in between that of A and C.
- **Measuring uncertainty:** Entropy is a measure of uncertainty.
- Entropy allows us to make precise statements and perform computations with regard to one of life's most pressing issues: not knowing how things will turn out.
- By the term entropy, I will refer to **Shannon entropy,** which is used most frequently in natural language processing and machine learning.

Entropy and uncertainty

- The **Shannon entropy** formula for an event *X* with *n* possible outcomes and probabilities $p_1, \ldots, p_n$:

$$H(X) = H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- For previous patients example,

H(A)=-0.95log(0.95)-0.05log(0.05)=0.08

H(B)= -0.7log(0.7)-0.3log(0.3)=0.27

H(C)= -0.5log(0.5)-0.5log(0.5)=0.3 (maximum)

Entropy and uncertainty

- The **Shannon entropy** formula for an event *X* with *n* possible outcomes and probabilities $p_1, \ldots, p_n$:

$$H(X) = H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

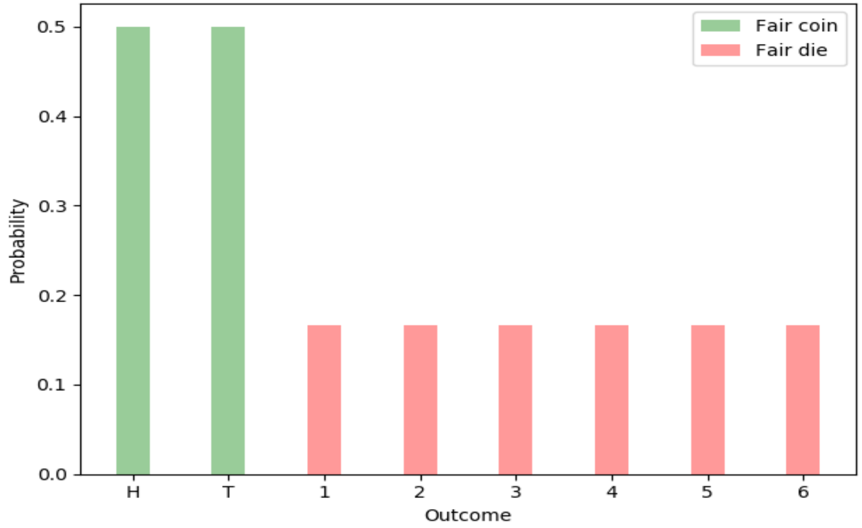
- **Basic properties of uncertainty:**

Prop. 1: Uniform distributions have maximum uncertainty

If your goal is to minimize uncertainty, stay away from **uniform probability distributions**.

Prop -1 of uncertainty

- A probability distribution is a function that assigns a probability to every possible outcome such that the probabilities add up to 1.
- A distribution is uniform when all of the outcomes have the same probability.
- For example, fair coins (50% heads, 50% tails) and fair dice (1/6 probability for each of the six faces) follow uniform distributions.

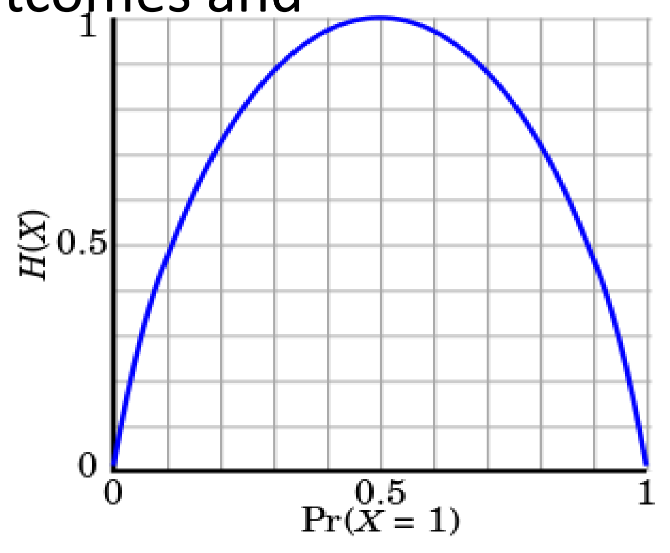


Prop -1 of uncertainty

- A good measure of uncertainty achieves its highest values for uniform distributions.
- Entropy satisfies the criterion. Given *n* possible outcomes, maximum entropy is maximized by equiprobable outcomes:

$$p_1 = ... = p_n = \frac{1}{n}$$

- Here is the plot of the Entropy function as applied to Bernoulli trials (events with two possible outcomes and probabilities *p* and *1-p*):
- In the case of Bernoulli trials, entropy reaches its maximum value for p=0.5



Prop -2 of uncertainty

- **Prop-2: Uncertainty is additive for independent events**
- Let *A* and *B* be independent events. In other words, knowing the outcome of event *A* does not tell us anything about the outcome of event *B*.
- The uncertainty associated with both events should be the sum of the individual uncertainties:

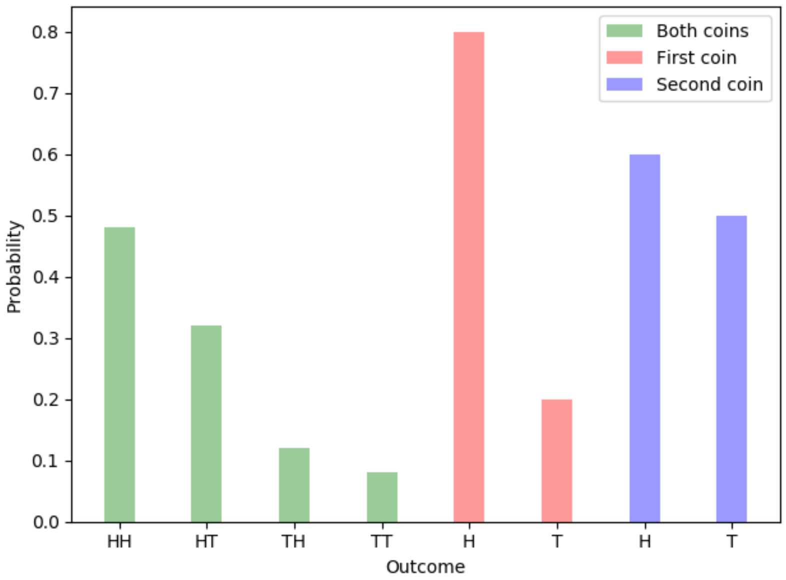
$$H(X, Y) = H(X) + H(Y)$$

- Let's use the example of flipping two coins to make this more concrete. We can either flip both coins simultaneously or first flip one coin and then flip the other one.
- In other words, we can either report the outcome of the two coin flips at once or separately. The uncertainty is the same in either case.

Prop -2 of uncertainty

- Ex: Let the first coin lands heads (*H*) up with an 80% probability and tails (*T*) up with a probability of 20%.
- The probabilities for the other coin are 60% and 40%.
- If we flip both coins simultaneously, there are four possible outcomes: *HH*, *HT*, *TH* and *TT*. The corresponding probabilities are given by *[0.48, 0.32, 0.12, 0.08]*.

• The joint entropy (green) for the two independent events is equal to the sum of the individual events (red and blue).



Prop -2 of uncertainty

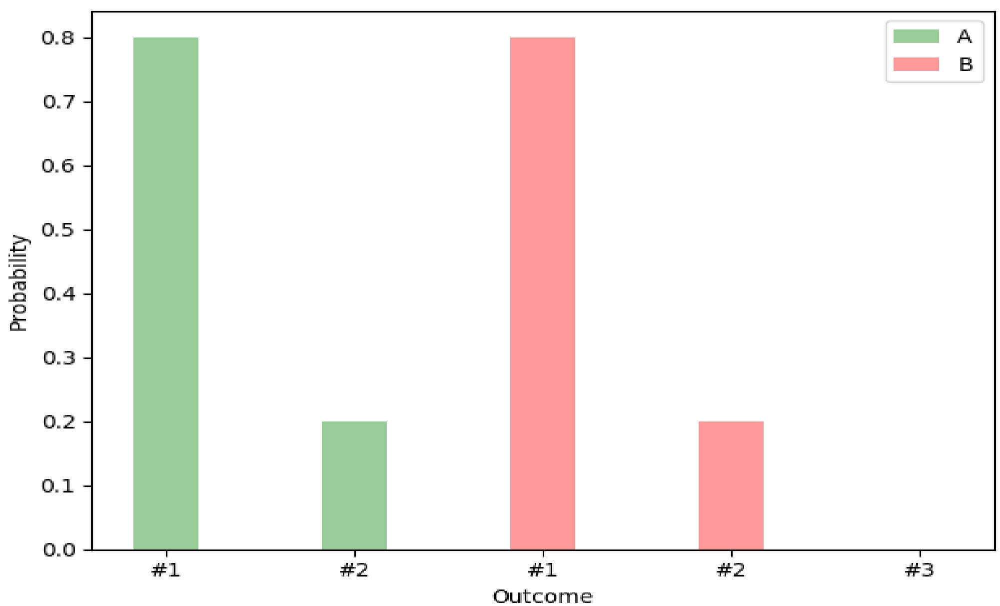
- Plugging the numbers into the entropy formula, we see that: Just as promised,

$$H(0.48, 0.32, 0.12, 0.08) = H(0.8, 0.2) + H(0.6, 0.4)$$

Prop -3 of uncertainty

Prop-3: Adding an outcome with zero probability has no effect

- Suppose (a) you win whenever outcome #1 occurs and (b) you can choose between two probability distributions, *A* and *B*.
- Distribution *A* has two outcomes: say, 80% and 20%. Distribution *B* has three outcomes with probabilities 80%, 20% and 0%.



Prop -3 of uncertainty

- Given the options *A* and *B*, which one would you choose?
- Ans: The inclusion of the third outcome neither increases nor decreases the uncertainty associated with the game. *A* or *B*, who cares. It doesn't matter.
- The entropy formula agrees with this assessment:

$$H(p_1, ..., p_n) = H(p_1, ..., p_n, 0)$$

- In words, adding an outcome with zero probability has no effect on the measurement of uncertainty.

Prop -4 of uncertainty

Prop-4: The measure of uncertainty is continuous in all its arguments

- The last of the basic properties is continuity.
- Famously, the intuitive explanation of a continuous function is that arbitrarily small changes in the output (uncertainty, in our case) should be achievable through sufficiently small changes in the input (probabilities).
- Logarithm functions are continuous at every point for which they are defined. So are sums and products of a finite number of functions that are continuous on a subset.
- It follows that the entropy function is continuous in its probability arguments.

Entropy

- **The Uniqueness Theorem:** Khinchin (1957) showed that the only family of functions satisfying the four basic properties described above is of the following form:

$$H(p_1, ..., p_n) = -\lambda \sum_{i=1}^{n} p_i \log p_i,$$

- where λ is a positive constant. Khinchin referred to this as the **Uniqueness Theorem**.
- Setting λ = 1 and using the binary logarithm gives us the Shannon entropy.
- To reiterate, entropy is used because it has desirable properties and is the natural choice among the family functions that satisfy all items on the basic properties.

Entropy

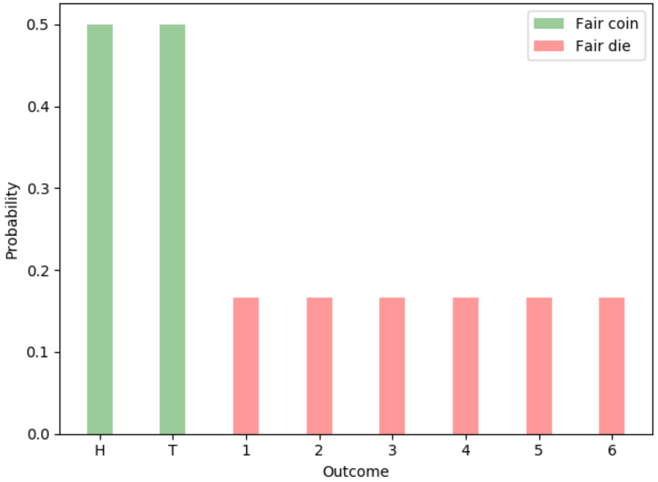
Other properties of Entropy:

- Entropy has many other properties used in Khinchin's Uniqueness Theorem. Some of them:

Prop-5: Uniform distributions with more outcomes have more uncertainty

- Suppose you have the choice between a fair coin and a fair die:

Fair coin or fair die ?
Let, A for coin and B for die.
H(A)=- 0.5ln(0.5) - 0.5ln(0.5) = 0.69
H(B)=[-(1/6)ln(1/6)] x 6=1.8



Entropy

- And let's say you win if the coin lands heads up or the die lands on face 1.
- Which of the two options would you choose?

  (i) if you are a profit maximizer and

  (ii) if you prefer with more variety and uncertainty.
- As the number of equiprobable outcomes increases, so should our measure of uncertainty.
- And this is exactly what Entropy does: H(1/6, 1/6, 1/6, 1/6, 1/6, 1/6) > H(0.5, 0.5).
- And, in general, if we let *L(k)* be the entropy of a uniform distribution with *k* possible outcomes,

  $L(m) > L(n)$ for $m > n$

Entropy

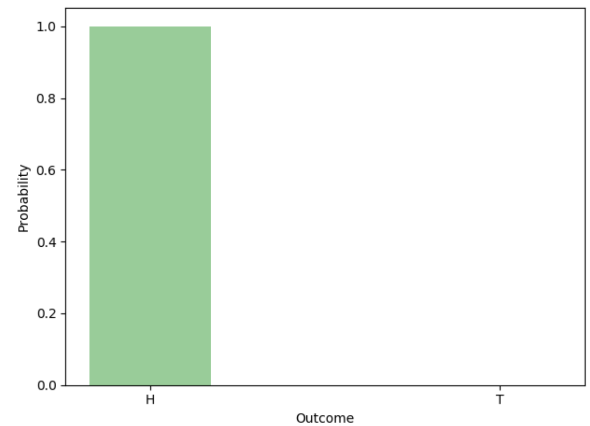
Prop-6: Events have non-negative uncertainty

- Do you know what negative uncertainty is? Neither do I.
- A user-friendly measure of uncertainty should always return a non-negative quantity, no matter what the input is.
- This is yet another criterion that is satisfied by entropy. Let's take another look at the formula: $$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$
- Probabilities are, by definition,

  in the range between 0 and 1 and, therefore, non-negative.
- The logarithm of a probability is non-positive. Multiplying the logarithm of a probability with a probability doesn't change the sign. The sum of non-positive products is non-positive. And finally, the negative of a non-positive value is non-negative.
- Entropy is, thus, non-negative for every possible input.

Entropy

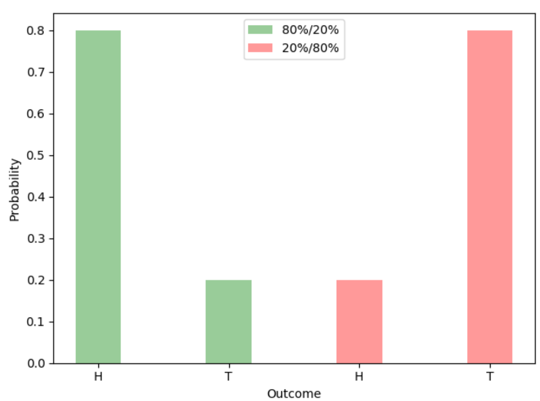
- **Prop-7: Events with a certain outcome have zero uncertainty**
- Suppose you are in possession of a magical coin. No matter how you flip always lands head up.
  - Suppose that outcome *i* certain to occur. It follows that *pi*, the probability of outcome *i*, is equal to 1. *H(X),* thus, simplifies to:



$$H(0, ..., 1, ..., 0) = -log_2 1 = 0$$

Entropy

- **Prop-8: Flipping the arguments has no effect**
- This is another obviously desirable property. Consider two cases. In the first case, the probability of heads and tails are 80% and 20%. In the second case, the probabilities are reversed: heads 20%, tails 80%.



Entropy

- Both coin flips are equally uncertain and have the same entropy: *H(0.8, 0.2) = H(0.2, 0.8)*.
- In more general terms, for the case of two outcomes, we have:

$$H(p_1, p_2) = H(p_2, p_1)$$

- This fact applies to any number of outcomes. We can position the arguments (i.e., the probabilities of a distribution) in any order we like. The result of the entropy function is always the same.

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Entropy (a measure of unpredictability or impurity) of the class or target variable in the dataset = Expected information needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Entropy of the target variable after split the dataset D based on a feature A = Weighted average of the entropies of the subsets formed by the split of D using A = Information needed (after using A to split D into v partitions) to classify a tuple in D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- The information gain is the difference between the entropy before the split and the entropy after the split.

$$Gain(A) = Info(D) - Info_A(D)$$

- Information gain is the reduction in entropy after a dataset is split on a feature.
- A higher information gain indicates that the feature provides a better separation of the target variable, making it a preferred choice for splitting the dataset in decision tree algorithms.

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  – GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$

  – gain_ratio(income) = 0.029/1.557 = 0.019

  Similarly, Gain_ratio(student) = 0.151/0.69=0.22

- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index

- The Gini coefficient measures the inequality among the values of a frequency distribution, such as levels of income. In machine learning, it is utilized as an impurity measure in decision tree algorithms for classification tasks.
- The Gini index is the most commonly used measure of inequality. It was developed by Italian statistician Corrado Gini and is named after him.
- A Gini coefficient of 0 reflects perfect equality, where all income or wealth values are the same, while a Gini coefficient of 1 (or 100%) reflects maximal inequality among values, a situation where a single individual has all the income while all others have none.

Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, *gini*(D) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index *gini*(D) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the largest reduction in impurity is chosen to split the node.

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"