Module-3

Divide and Conquer – Classification Using Decision Trees and Rules

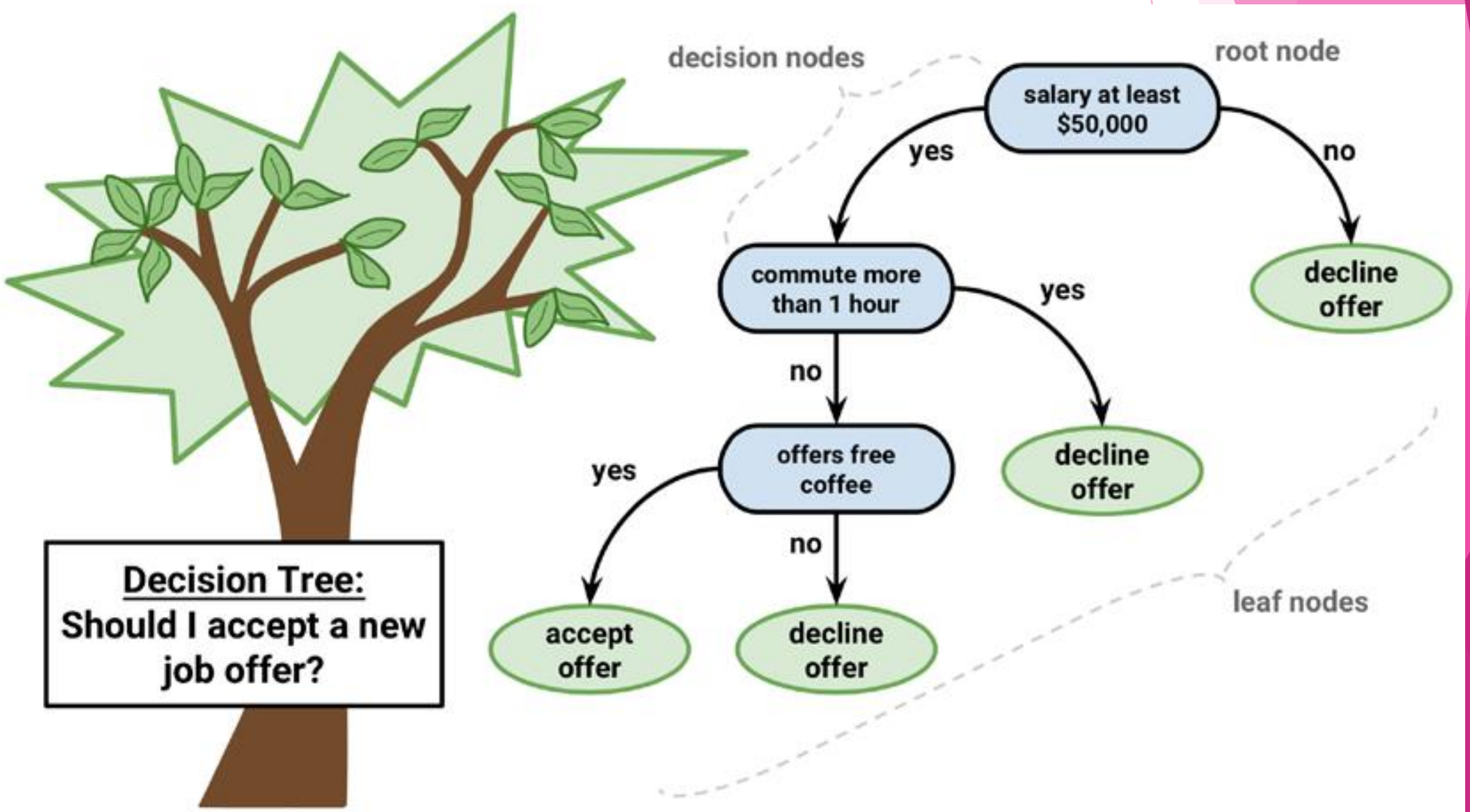
decision trees and rule learners

- two machine learning methods that make complex decisions from sets of simple choices
  - present their knowledge in the form of logical structures
  - particularly useful for business strategy and process improvement
- Objectives
  - How trees and rules "greedily" partition data into interesting segments
  - The most common decision tree and classification rule learners, including the C5.0, 1R, and RIPPER algorithms
  - How to use these algorithms to perform real-world classification tasks, such as identifying risky bank loans and poisonous mushrooms

Understanding decision trees

- **tree structure** to model the relationships among the features and the potential outcomes
  - branching decisions, which channel examples into a final predicted class value

E.g.,
- Root node
- Decision nodes – split the data across branches
- Leaf nodes (terminal nodes – the action to be taken or expected results



benefit of decision tree algorithms

- flowchart-like tree structure
  - Not necessarily exclusively for the learner's internal use
  - in a human-readable format
  - provides insight into how and why the model works or doesn't work
  - classification mechanism can be transparent
- some potential uses
  - Credit scoring models in which the criteria that causes an applicant to be rejected need to be clearly documented and free from bias
  - Marketing studies of customer behavior such as satisfaction or churn, which will be shared with management or advertising agencies
  - Diagnosis of medical conditions based on laboratory measurements, symptoms, or the rate of disease progression

Pros and cons

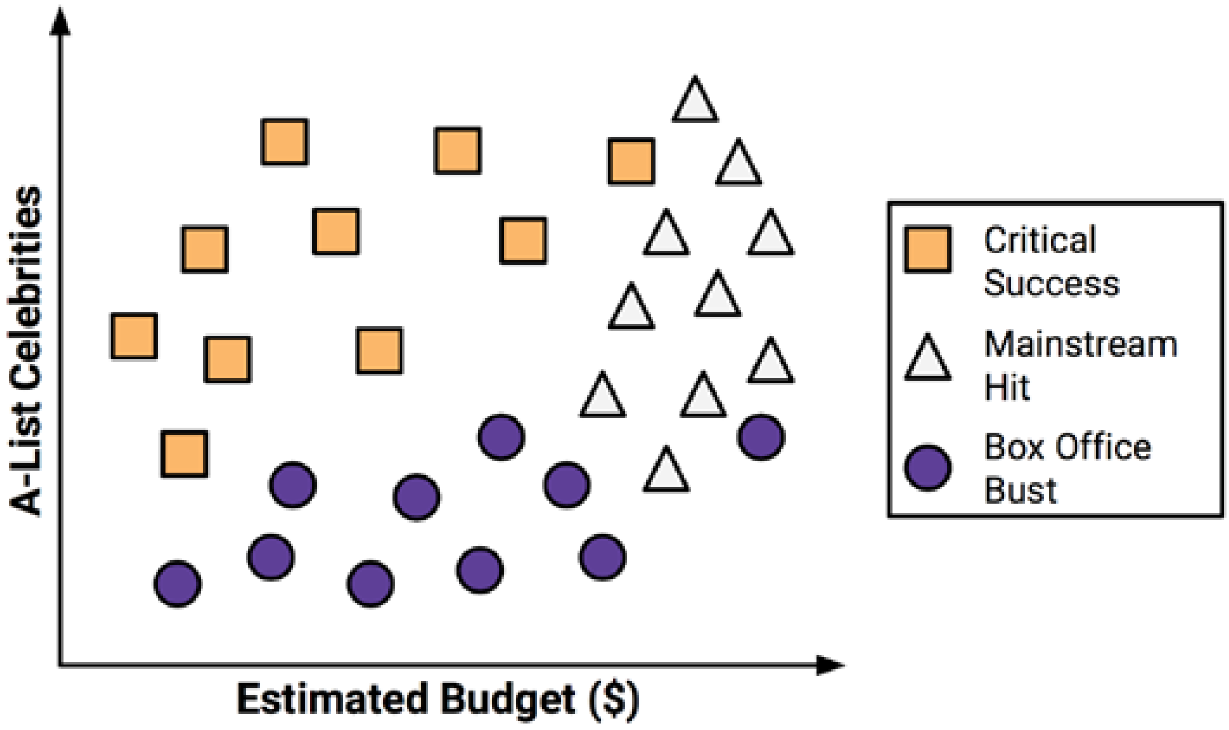
- Pros
  - decision trees are perhaps the single most widely used machine learning technique
  - can be applied to model almost any type of data
- Cons
  - trees may not be an ideal fit for a task where the data has a large number of nominal features with many levels or it has a large number of numeric features.
    - result in a very large number of decisions and an overly complex tree.
    - tendency of decision trees to overfit data

Divide and conquer

- **recursive partitioning**
  - splits the data into subsets, which are then split repeatedly into even smaller subsets, …
  - stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met
- Steps
  - the root node represents the entire dataset
  - choose a feature to split upon; ideally, it chooses the feature most predictive of the target class.
  - The examples are then partitioned into groups according to the distinct values of this feature
  - stop at a node in a case that:
    - All (or nearly all) of the examples at the node have the same class
    - There are no remaining features to distinguish among the examples
    - The tree has grown to a predefined size limit

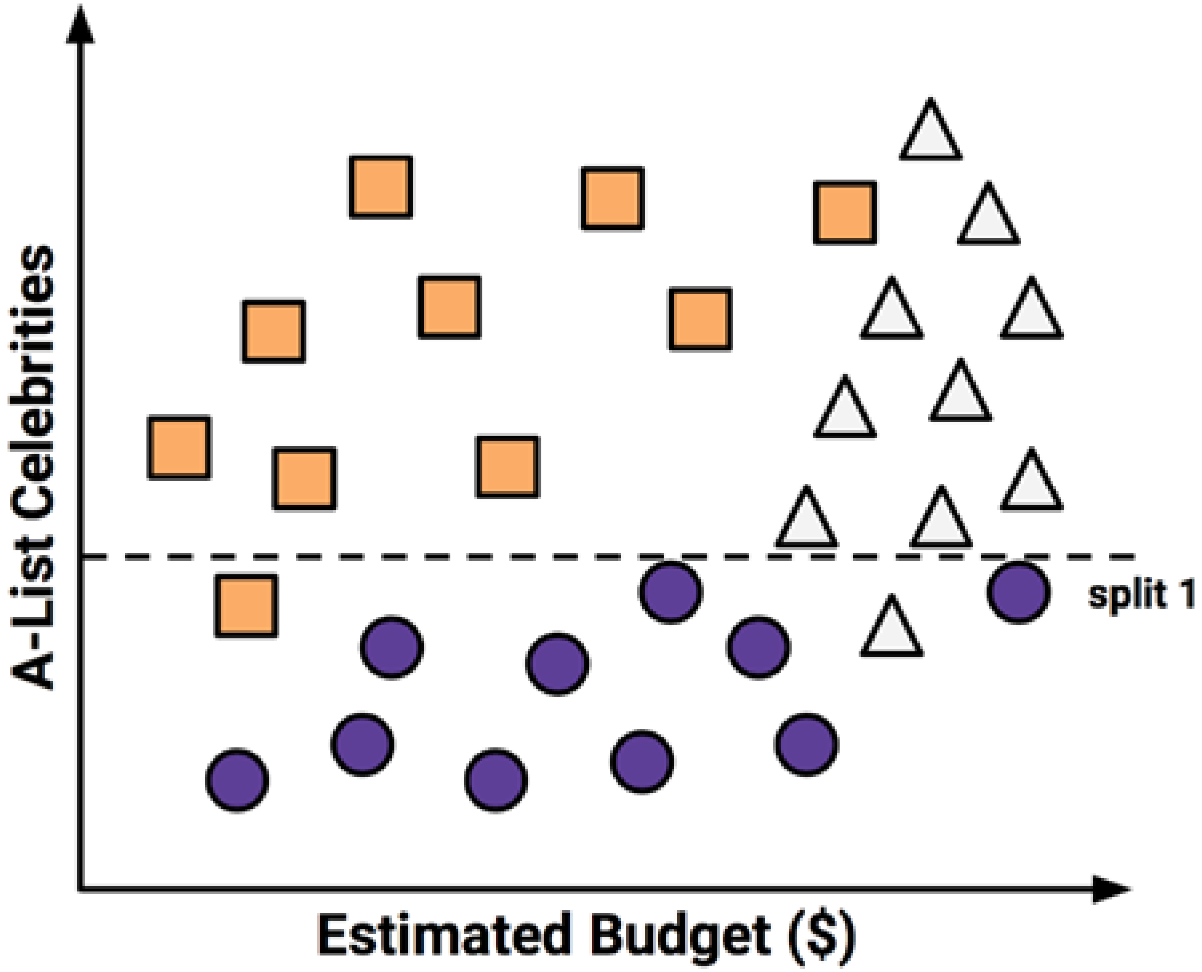
Example - Hollywood studio

- predict whether a potential movie would fall into one of three categories: **Critical Success**, **Mainstream Hit**, or **Box Office Bust**
- the factors leading to the success and failure of the company's 30 most recent releases



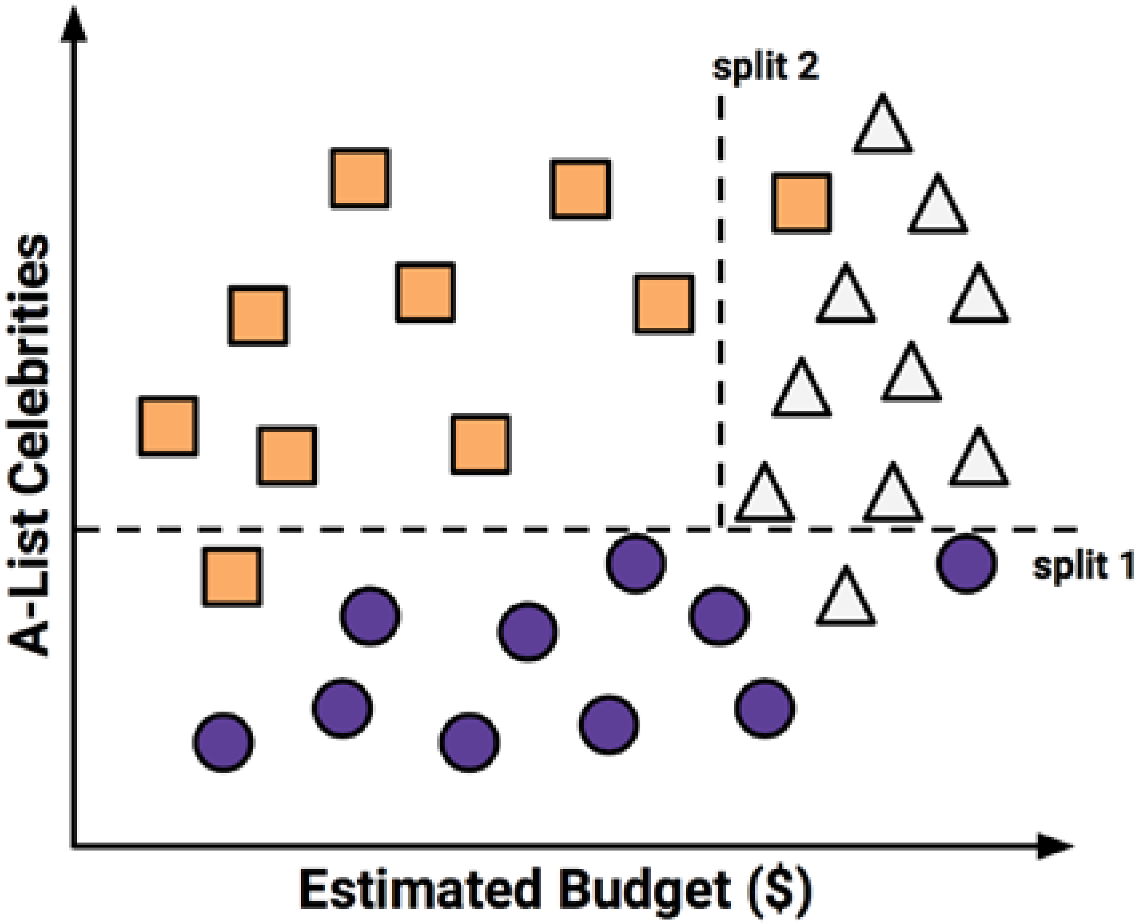
build a decision tree

- split the feature indicating the number of celebrities



build a decision tree cont'd

- another split between movies with and without a high budget

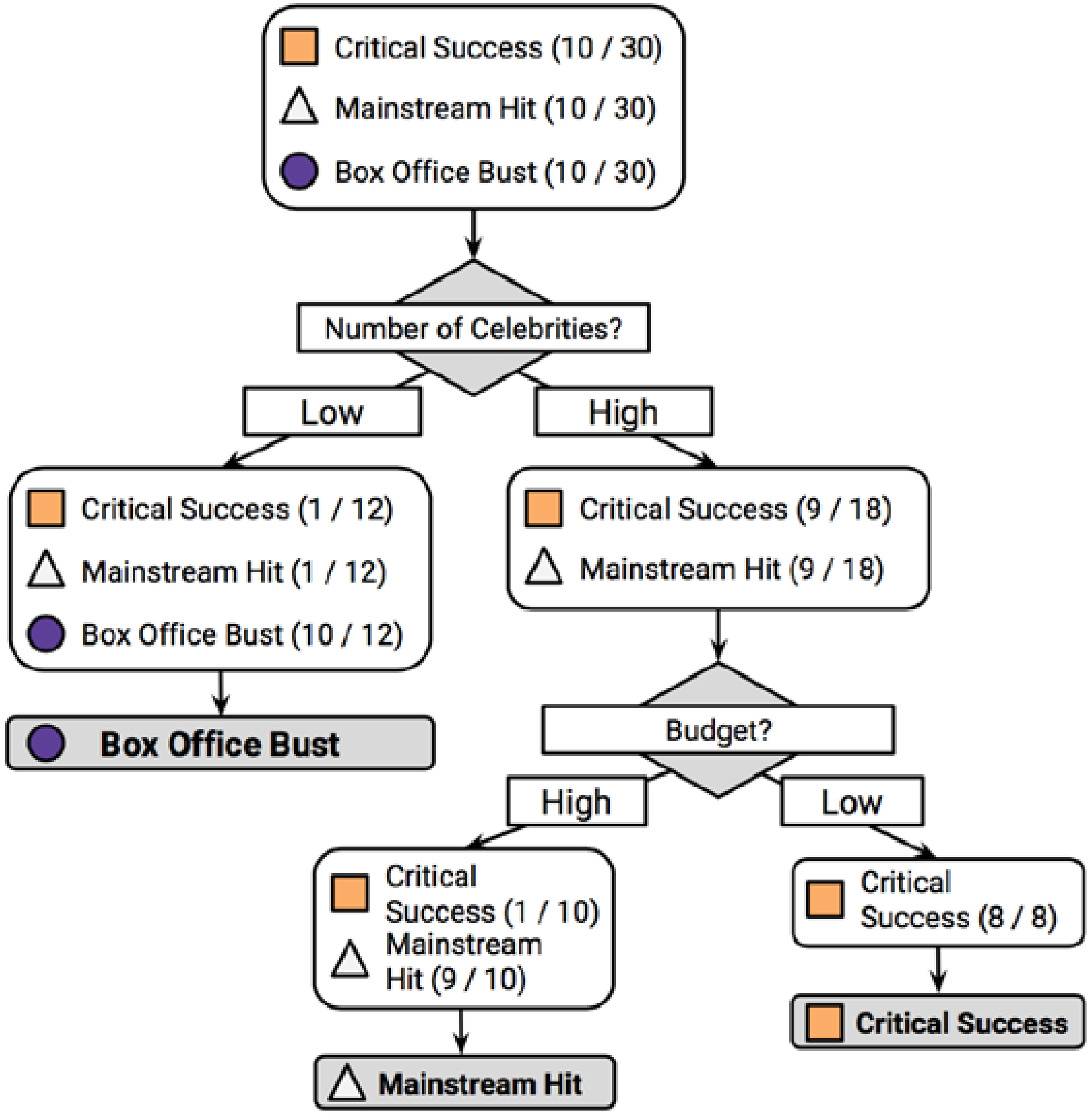


build a decision tree cont’d

- partitioned the data into three groups.
    - top-left corner: entirely of critically acclaimed films.
        - high number of celebrities and a relatively low budget.
    - top-right corner: box office hits with high budgets and a large number of celebrities.
    - final group: little star power but budgets ranging from small to large
- could continue to divide and conquer the data by splitting it based on the increasingly specific ranges of budget and celebrity count
- not advisable to overfit a decision tree

axis-parallel splits

- The fact that each split considers one feature at a time prevents the decision tree from forming more complex decision boundaries
  - a diagonal line could be created by a decision that asks, "is the number of celebrities is greater than the estimated budget?" If so, then "it will be a critical success."



The C5.0 decision tree algorithm

- **C5.0 algorithm**
  - Developed by computer scientist J. Ross Quinlan

    Iterative Dichotomiser 3 (**ID3**) -> **C4.5 -> C5.0**
  - http://www.rulequest.com/
  - single-threaded version publically available
  - industry standard to produce decision trees
- **J48 -** Java-based open source alternative to C4.5

Pros and cons

Strengths	Weaknesses
• An all-purpose classifier that does well on most problems • Highly automatic learning process, which can handle numeric or nominal features, as well as missing data • Excludes unimportant features • Can be used on both small and large datasets • Results in a model that can be interpreted without a mathematical background (for relatively small trees) • More efficient than other complex models	• Decision tree models are often biased toward splits on features having a large number of levels • It is easy to overfit or underfit the model • Can have trouble modeling some relationships due to reliance on axis-parallel splits • Small changes in the training data can result in large changes to decision logic • Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

Choosing the best split

- first challenge - identify which feature to split upon
  - split the data such that the resulting partitions contained examples primarily of a single class
- **purity -** degree to which a subset of examples contains only a single class
  - Any subset composed of only a single class is called **pure**
- **Entropy -** concept borrowed from information theory that quantifies the randomness, or disorder
  - measurements of purity
  - Sets with high entropy are very diverse

entropy

- If there are only two possible classes, entropy values can range from 0 to 1.
- For *n* classes, entropy ranges from 0 to *log2(n)*.
- the minimum value indicates that the sample is completely homogenous
- the maximum value indicates that the data are as diverse as possible

mathematical notion

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

- S - given segment of data
- *c* - the number of class levels
- p_i - the proportion of values falling into class level *i*.
- E.g., partition of data with two classes: red (60 percent) and white (40 percent)

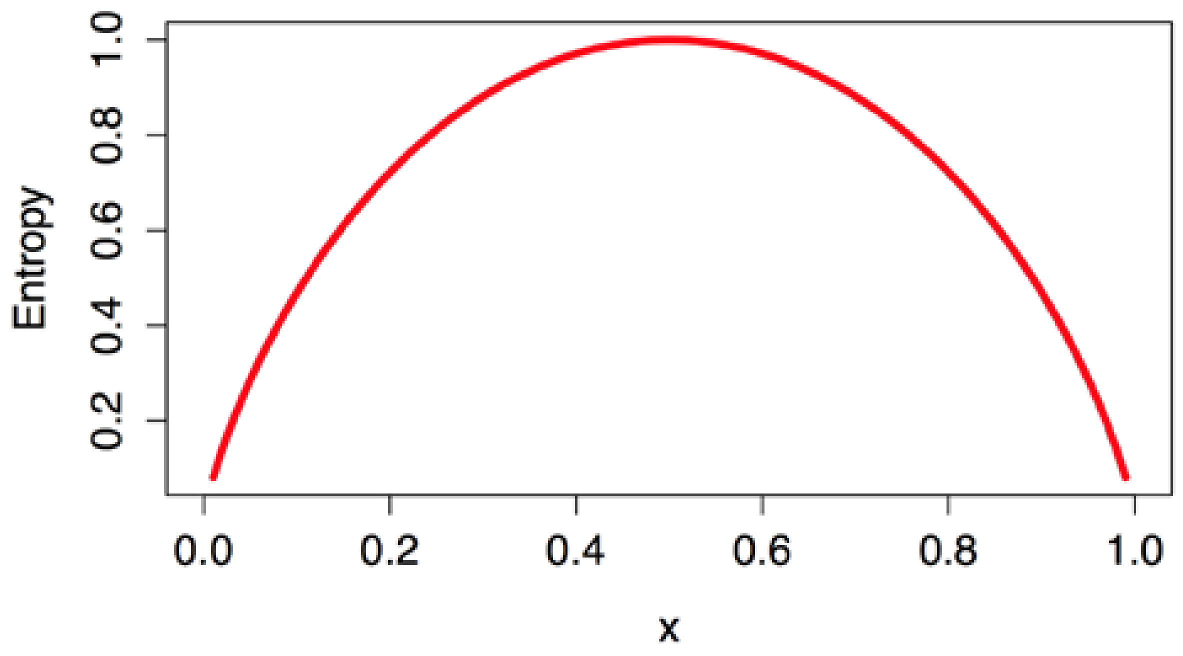
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506

entropy for all the possible two-class arrangements

- Using the curve() function, plot the entropy for all the possible values of *x*:

```
> curve(-x * log2(x) - (1 - x) * log2(1 - x), col = "red", xlab = "x", ylab = "Entropy", lwd = 4)
```

- 50-50 split results in maximum entropy.
- As one class increasingly dominates the other, the entropy reduces to zero



determine the optimal feature to split upon

- **information gain -** the change in homogeneity that would result from a split on each possible feature
- information gain for a feature *F* - the difference between the entropy in the segment before the split *(S_1)* and the partitions resulting from the split *(S_2)*:

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

- function to ca py across all of the partitions
  - weighing each partition's entropy by the proportion of records falling into the partition
  - the total entropy resulting from a split is the sum of the entropy of each of the *n* partitions weighted by the proportion of examples falling in the partition (w_i).

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

information gain

- The higher the information gain, the better a feature is at creating homogeneous groups after a split on this feature.
- If the information gain is zero, there is no reduction in entropy for splitting on this feature.
- the maximum information gain is equal to the entropy prior to the split, which means that the split results in completely homogeneous groups

splitting on numeric features

- Test various splits that divide the values into groups greater than or less than a numeric threshold
- reduces the numeric feature into a two-level categorical feature
- The numeric cut point yielding the largest information gain is chosen for the split

Pruning the decision tree

- A decision tree can continue to grow indefinitely
- if the tree grows overly large, many of the decisions it makes will be overly specific and the model will be overfitted to the training data
- **pruning** a decision tree - reducing its size such that it generalizes better to unseen data
- One solution - **early stopping** or **pre-pruning** the decision tree: stop the tree from growing once it reaches a certain number of decisions or when the decision nodes contain only a small number of examples
  - there is no way to know whether the tree will miss subtle, but important patterns that it would have learned had it grown to a larger size

post-pruning

- growing a tree that is intentionally too large and pruning leaf nodes to reduce the size of the tree to a more appropriate level
- often a more effective approach than pre-pruning
  - it is quite difficult to determine the optimal depth of a decision tree without growing it first.
  - Pruning the tree later on allows the algorithm to be certain that all the important data structures were discovered
- **subtree raising** and **subtree replacement**
  - first grows a large tree that overfits the training data.
  - Later, the nodes and branches that have little effect on the classification are removed.
  - In some cases, entire branches are moved further up the tree or replaced by simpler decisions