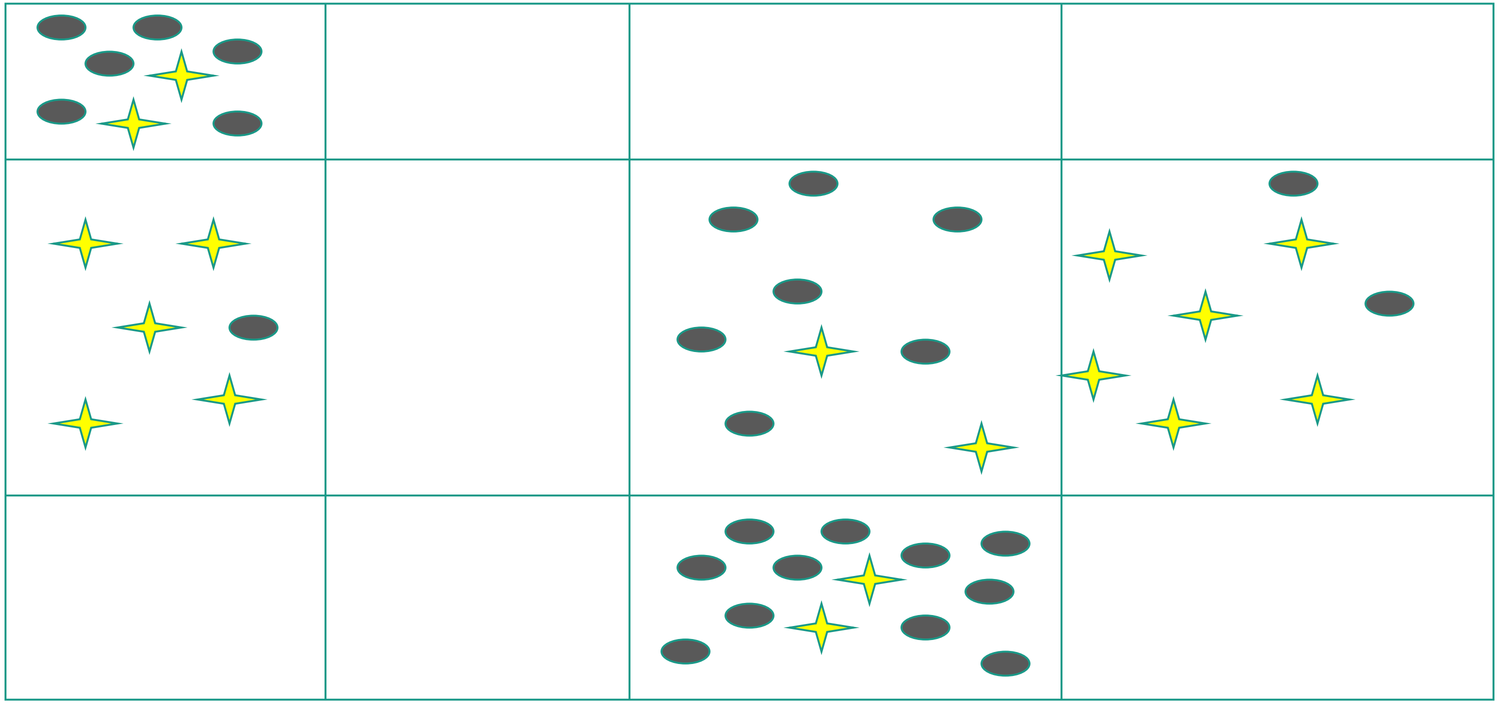
Decision Trees

A programming task



Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Support Vector Machines
- K Nearest Neighbor (KNN)
- Naïve Bayes and Bayesian Belief Networks

- **Decision Tree**: Every hiring manager has a set of criteria such as education level, number of years of experience, interview performance. A decision tree is analogous to a hiring manager interviewing candidates based on his or her own criteria.
- **Bagging**: Now imagine instead of a single interviewer, now there is an interview panel where each interviewer has a vote. Bagging or bootstrap aggregating involves combining inputs from all interviewers for the final decision through a democratic voting process.
- **Random Forest**: It is a bagging-based algorithm with a key difference wherein only a subset of features is selected at random. In other words, every interviewer will only test the interviewee on certain randomly selected qualifications (e.g. a technical interview for testing programming skills and a behavioral interview for evaluating non-technical skills).

- **Boosting**: This is an alternative approach where each interviewer alters the evaluation criteria based on feedback from the previous interviewer. This ‘boosts’ the efficiency of the interview process by deploying a more dynamic evaluation process.
- **Gradient Boosting**: A special case of boosting where errors are minimized by gradient descent algorithm e.g. the strategy consulting firms leverage by using case interviews to weed out less qualified candidates.
- **XGBoost**: Think of XGBoost as gradient boosting on ‘steroids’ (‘Extreme Gradient Boosting) It is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time.

Decision trees

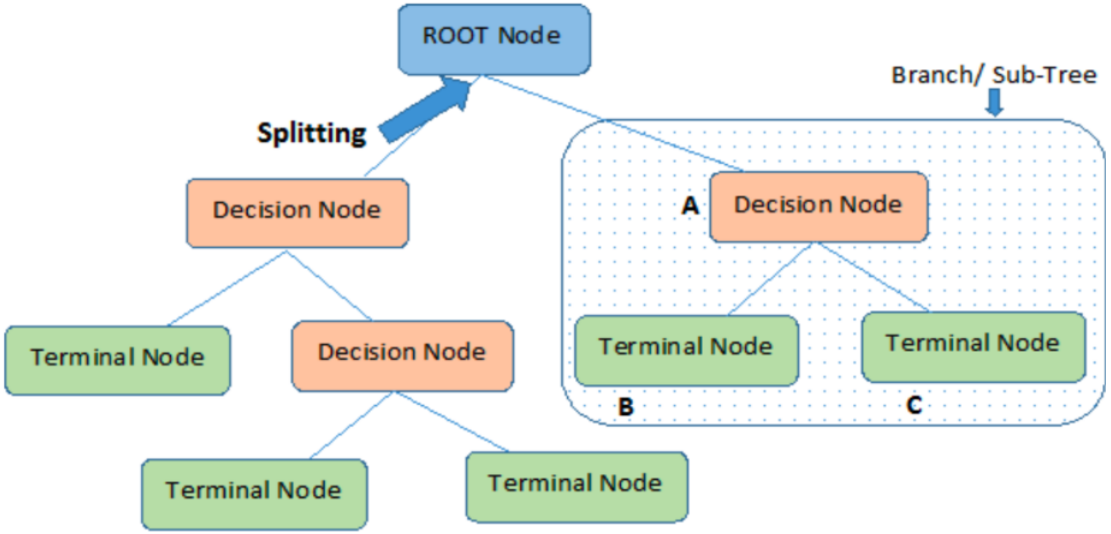
- **Decision trees**: one of most popular learning methods commonly used for data exploration

- A **decision tree** classifies data items by posing a series of questions about the features associated with the items.

- One type of decision tree is called C&RT (classification and regression tree)
    - greedy, top-down binary, recursive partitioning that divides feature space into sets of disjoint rectangular regions.

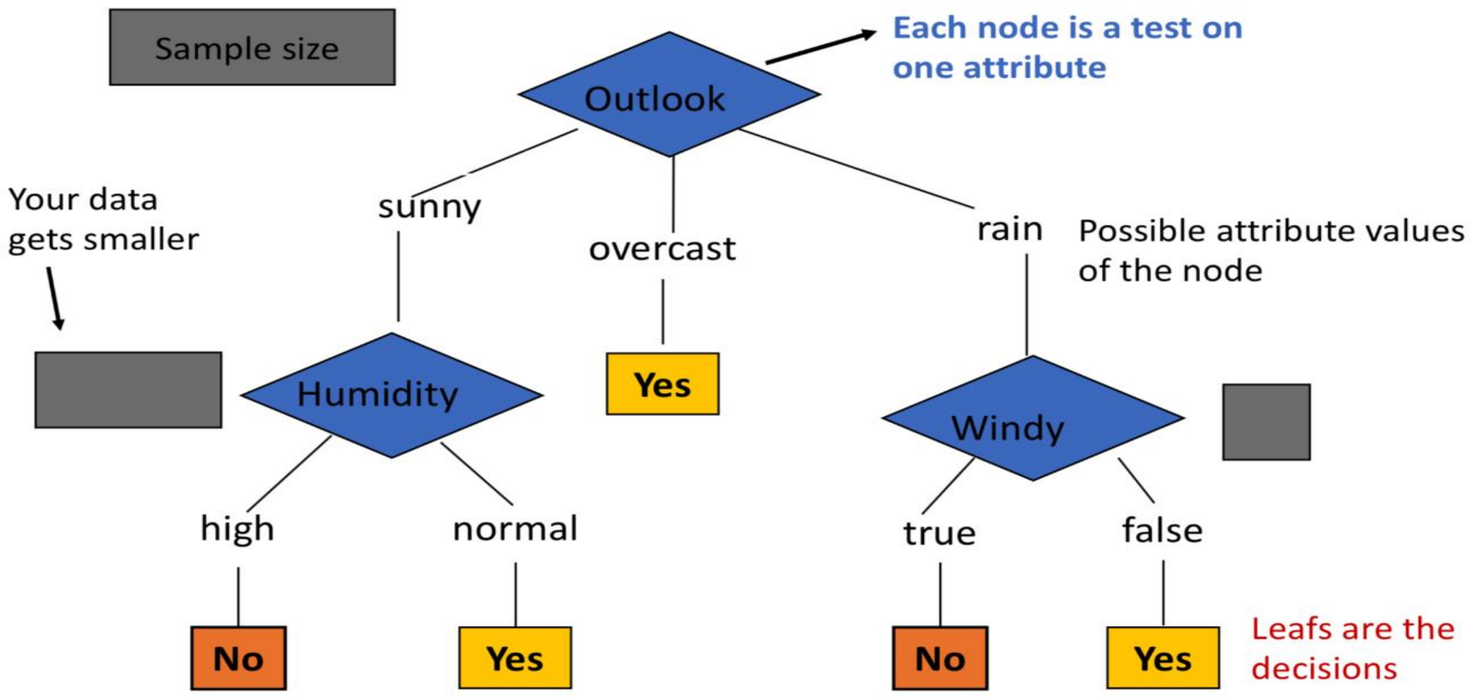
Terminology used in decision trees

- Root node
- Splitting
- Decision node
- Leaf / terminal node
- Pruning:
    - When we remove sub-nodes of a decision node, this process is called pruning.
- Branch / subtree
- Parent and child node

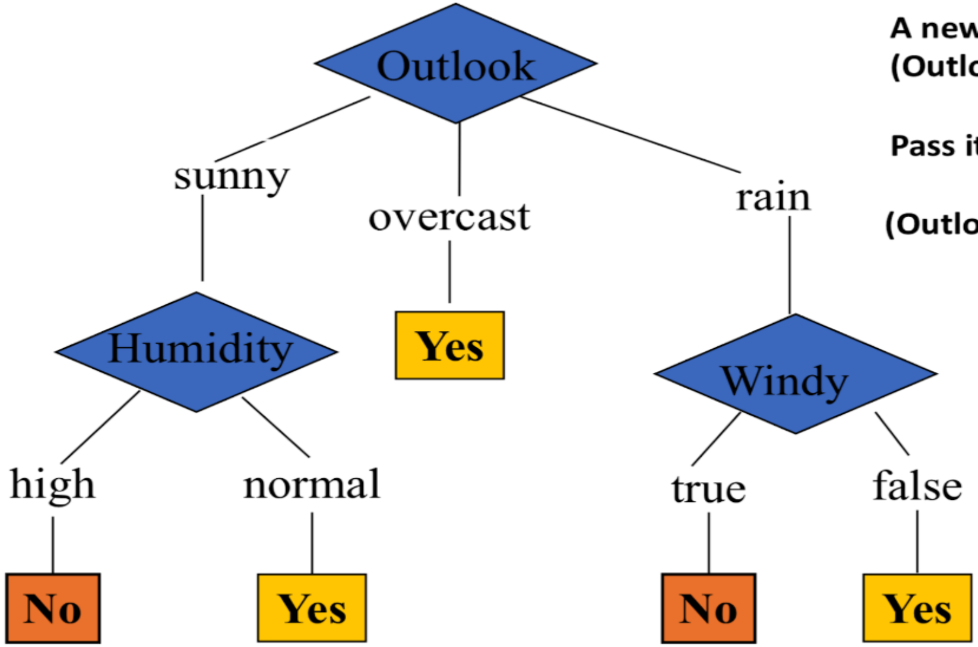


Note:- A is parent node of B and C.

To play Tennis or not



To play tennis or not



A new test example:
(Outlook == rain) and (not Windy == false)

Pass it on the tree -> Decision is yes.

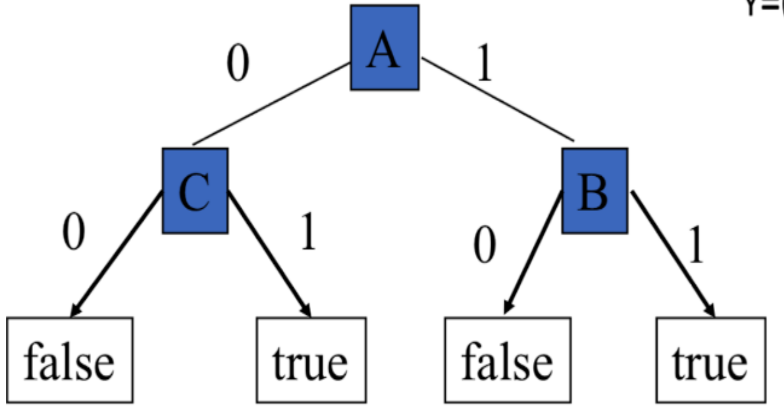
(Outlook == sunny) and (Humidity = high) -> no

Decision trees

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

```
(Outlook ==overcast)
 OR
((Outlook==rain) and (not Windy==false))
 OR
((Outlook==sunny) and (Humidity=normal))
 => yes play tennis
```

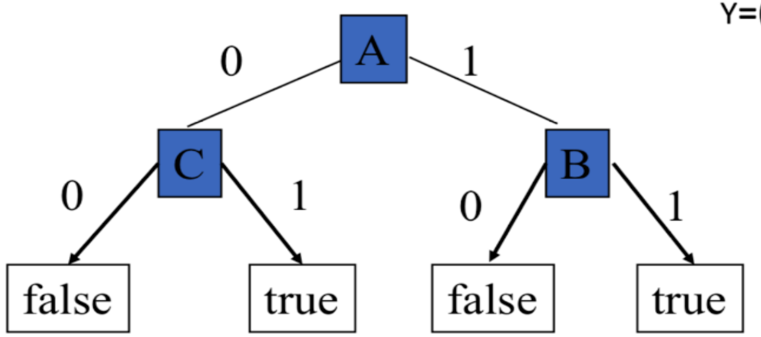
Representation

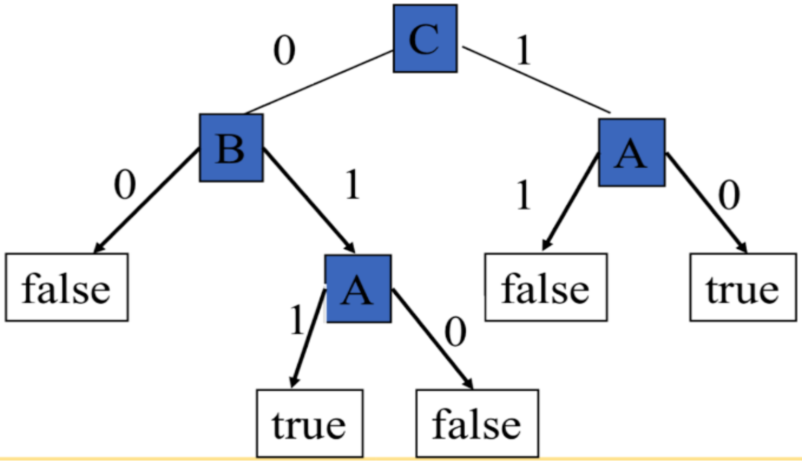


Y=((A and B) or ((not A) and C))

Same concept different representation

Y=((A and B) or ((not A) and C))



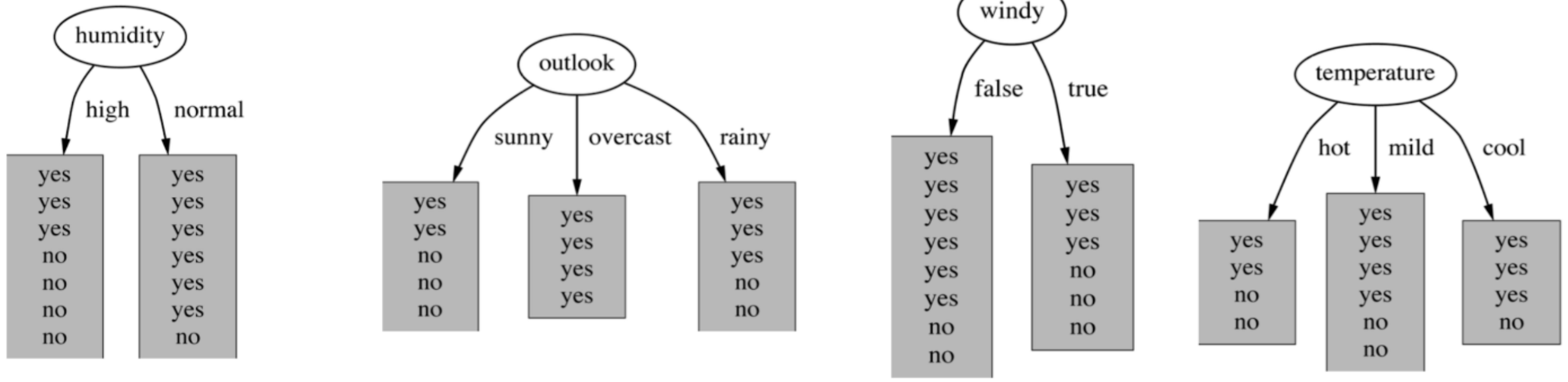


How to choose test?

- Which attribute should be used as the test?

- Intuitively, you would prefer the one that *separates* the training examples as much as possible.

Information Gain!



Information

- Imagine:

1. Someone is about to tell you your own name
2. You are about to observe the outcome of a dice roll
3. You are about to observe the outcome of a coin flip
4. You are about to observe the outcome of a biased coin flip

- Each situation have a different *amount of uncertainty* as to what outcome you will observe.

Entropy

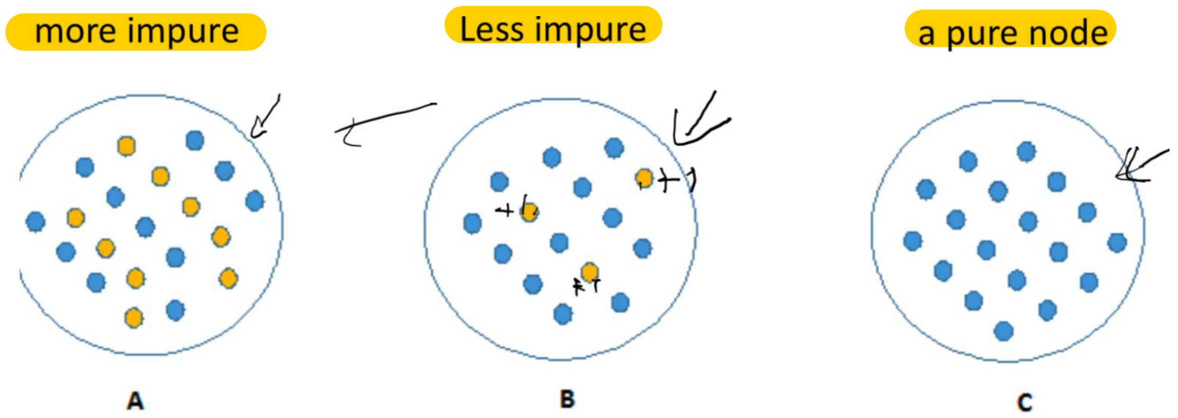
- A measure of homogeneity of the set of examples.
- Given a set S of positive and negative examples of some target concept (a 2-class problem), the entropy of set S relative to this binary classification is

E(S) = - p(P)log2 p(P) – p(N)log2 p(N)

$$H(X) = -\sum p(x_i)\log_2 p(x_i)$$

Entropy and purity

- Entropy measures the purity



Example

- Suppose S has 24 examples, 14 positive and 10 negatives [14+, 10-]. Then the entropy of S relative to this classification is

  E(S)=-(14/24) log2(14/24) - (10/24) log2 (10/24)

Suppose S has 24 examples, 24 positive and 0 negatives

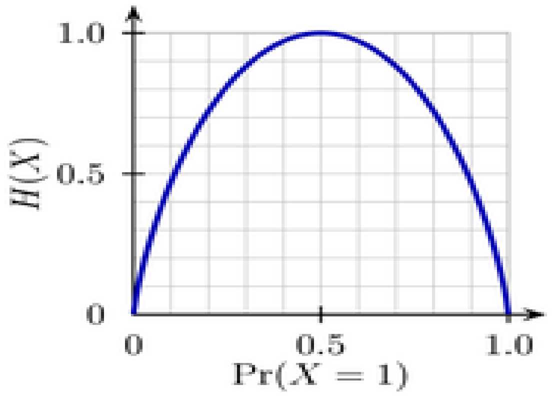
Then what is the entropy?

Suppose S has 24 examples, 12 positive and 12 negatives

Then what is the entropy?

Some Intuitions

- The entropy is 0 if the outcome is ``certain''.
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).



Entropy of a 2-class problem with regard to the portion of one of the two groups

Information and Entropy

- Equation for calculating the range of Entropy:

  $0 \leq \text{Entropy} \leq \log(n)$, where n is number of outcomes
- Entropy 0 (minimum entropy) occurs when one of the probabilities is 1 and rest are 0's
- Entropy log(n) (maximum entropy) occurs when all the probabilities have equal values of 1/n.

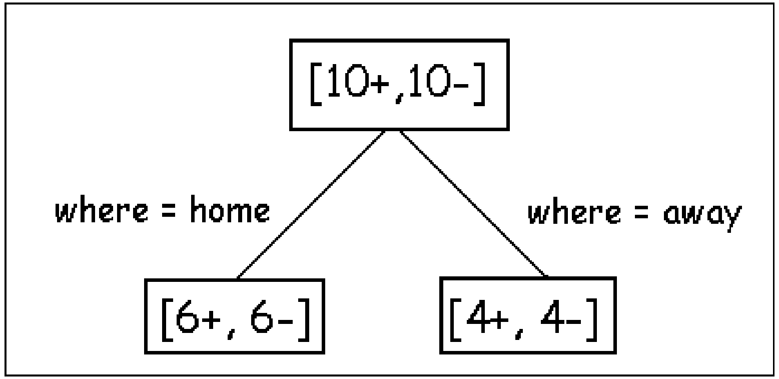
Information Gain

- Information gain measures the expected reduction in entropy, or uncertainty.

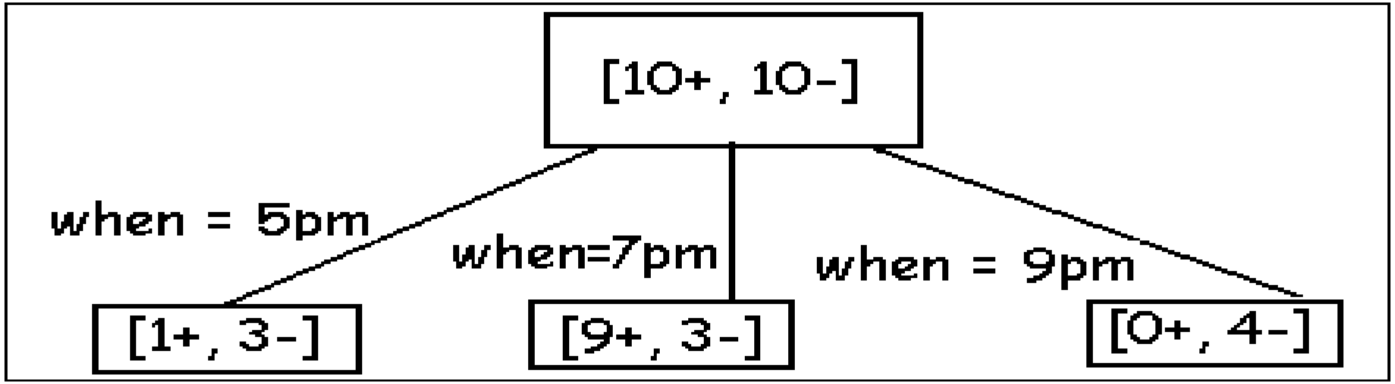
$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

  - IG = (information before split) – (information after split)
  - the first term in the equation for *Gain* is just the entropy of the original collection *S*
  - the second term is the expected value of the entropy after S is partitioned using attribute A

Examples



- Before partitioning, the entropy is
    - *H(10/20, 10/20) = - 10/20 log(10/20) - 10/20 log(10/20) = 1*
- Using the ``where'' attribute, divide into 2 subsets
    - Entropy of the first set   H(home) = - 6/12 log(6/12) - 6/12 log(6/12) = 1
    - Entropy of the second set H(away) = - 4/8 log(6/8) - 4/8 log(4/8) = 1
- Expected entropy after partitioning
    - 12/20 * H(home) + 8/20 * H(away) = 1



- Using the ``when'' attribute, divide into 3 subsets
  - Entropy of the first set H(5pm) = - 1/4 log(1/4) - 3/4 log(3/4);
  - Entropy of the second set H(7pm) = - 9/12 log(9/12) - 3/12 log(3/12);
  - Entropy of the second set H(9pm) = - 0/4 log(0/4) - 4/4 log(4/4) = 0
- Expected entropy after partitioning
  - 4/20 * H(1/4, 3/4) + 12/20 * H(9/12, 3/12) + 4/20 * H(0/4, 4/4) = 0.65
- Information gain 1-0.65 = 0.35

Decision

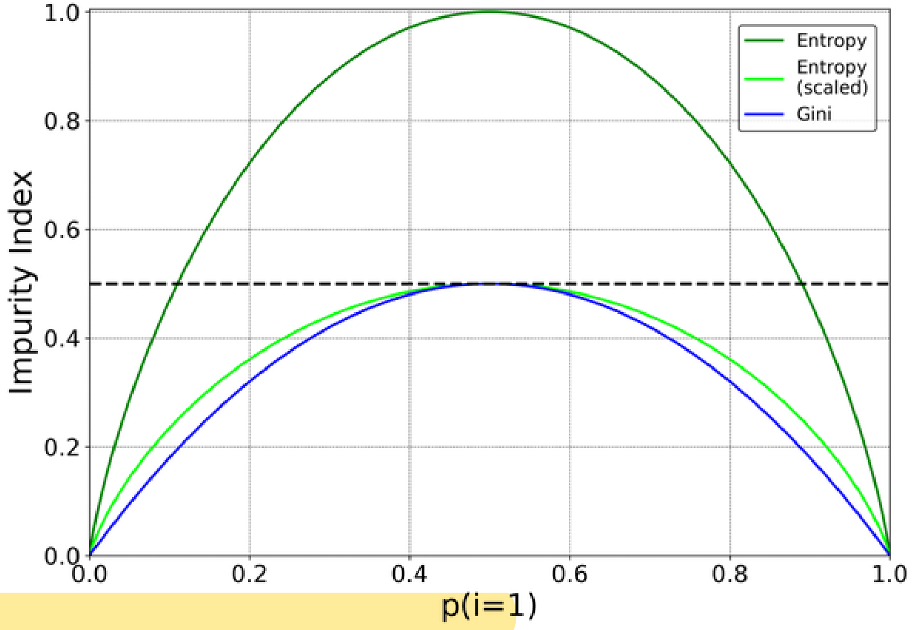
- Knowing the ``when'' attribute values provides larger information gain than ``where''.
- Therefore the ``when'' attribute should be chosen for testing prior to the ``where'' attribute.
- Similarly, we can compute the information gain for other attributes.
- At each node, choose the attribute with **the largest information gain.**

- Stopping rule
  - Every attribute has already been included along this path through the tree, or
  - The training examples **associated with this leaf node all have the same target attribute value** (i.e., their entropy is zero).

Decision Tree (ID3)

- ***Step 1***: Calculate entropy of the target.
- ***Step 2*:** The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.
- ***Step 3***: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.
- ***Step 4a*:** A branch with entropy of 0 is a leaf node\
- ***Step 4b*:** A branch with entropy more than 0 needs further splitting
- ***Step 5*:** The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

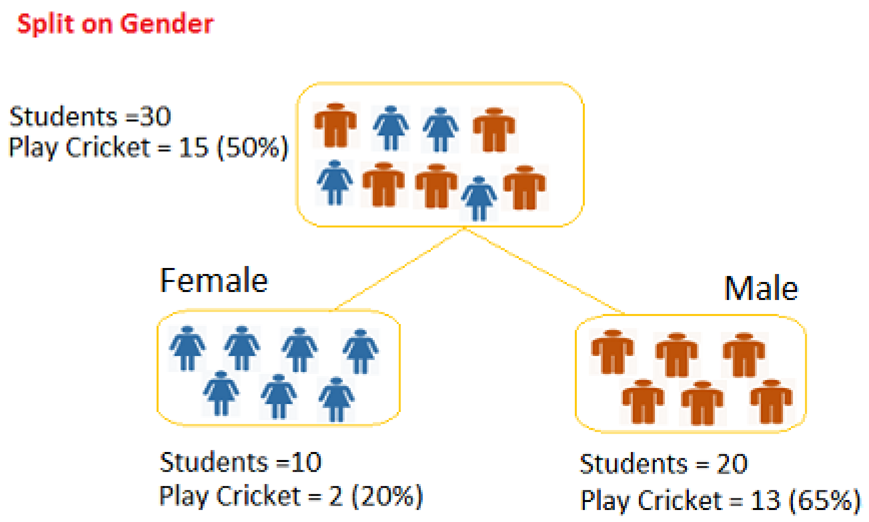
Purity Measures

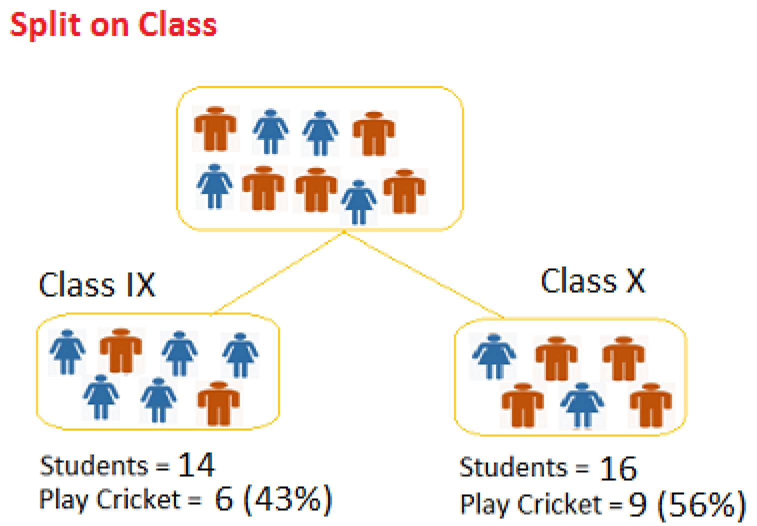


- Purity Measures:
  - Information Gain
  - Gini Index(Used in classification problem)
  - Regression error
- **Steps to Calculate Gini for a split**
- Calculate Gini for sub-nodes, using f
  probability for success and failure 1 – ($\mathbf{p^2+q^2}$).
- Calculate Gini for split using weighted Gini score of each node of that split
- **Computationally efficient**

Gini-index

- Split on Gender:
- Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68
- Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55
- Weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = 0.59
- Similar for Split on Class:
- Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51
- Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51
- Weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = 0.51





Gini Index

- It works with categorical target variable “Success” or “Failure”.
- It performs only Binary splits
- Higher the value of Gini higher the homogeneity.
- CART (Classification and Regression Tree) uses Gini method to create binary splits.
- **CART**, or *Classification And Regression Trees* is often used as a generic acronym for the term Decision Tree
- The Algorithm used in the decision trees are ID3 , C4.5, CART, C5.0, CHAID, QUEST, CRUISE, etc.

https://stackoverflow.com/questions/9979461/different-decision-tree-algorithms-with-comparison-of-complexity-or-performance

Overfitting

- You can perfectly fit to any training data using a decision tree

- Two approaches to avoid overfitting

1. **Stop growing the tree** when further splitting the data does not yield an improvement

2. **Grow a full tree, then prune the tree,** by eliminating nodes
Collapsing internal nodes into leaves or removing nodes

Decision tree summary

- Non-linear classifier
- Easy to use
- Easy to interpret
- Fast!
- Can handle both numerical and categorical data
- Prone to overfitting but can be avoided.
- Heuristic: mostly little theoretical explanations

An Aggregation story

- Your T friends g_1, ...g_T predicts whether stock will go up as $g_t(\mathbf{x})$
- You can
  - Select the most trust-worthy friend from their usual performance
    - validation!
  - Mix the predictions from all your friends uniformly
    - let them vote!
  - Mix the predictions from all your friends non-uniformly
    - let them vote, but give some ore ballots
  - Combine the predictions conditionally
    - If [t satisfies some condition] give some ballots to friend t

Aggregation models: mix or combine hypotheses (for better performance)

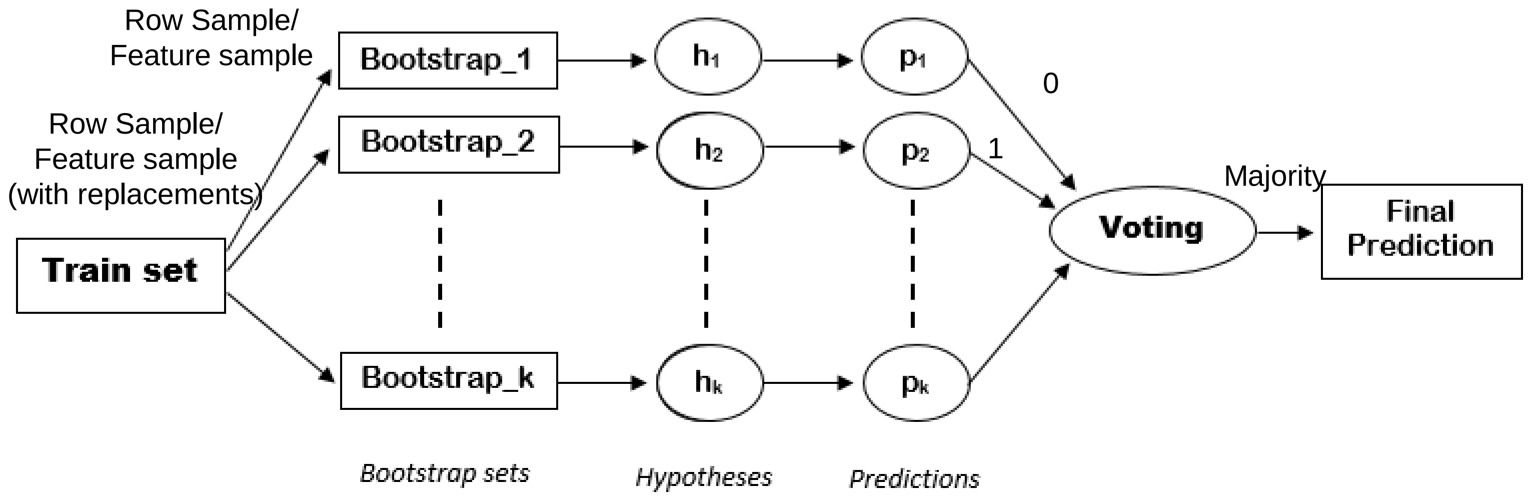
Ensemble Techniques

- Bagging = (Bootstrap Aggregating): N new training data sets by random sampling with replacement; **parallel**

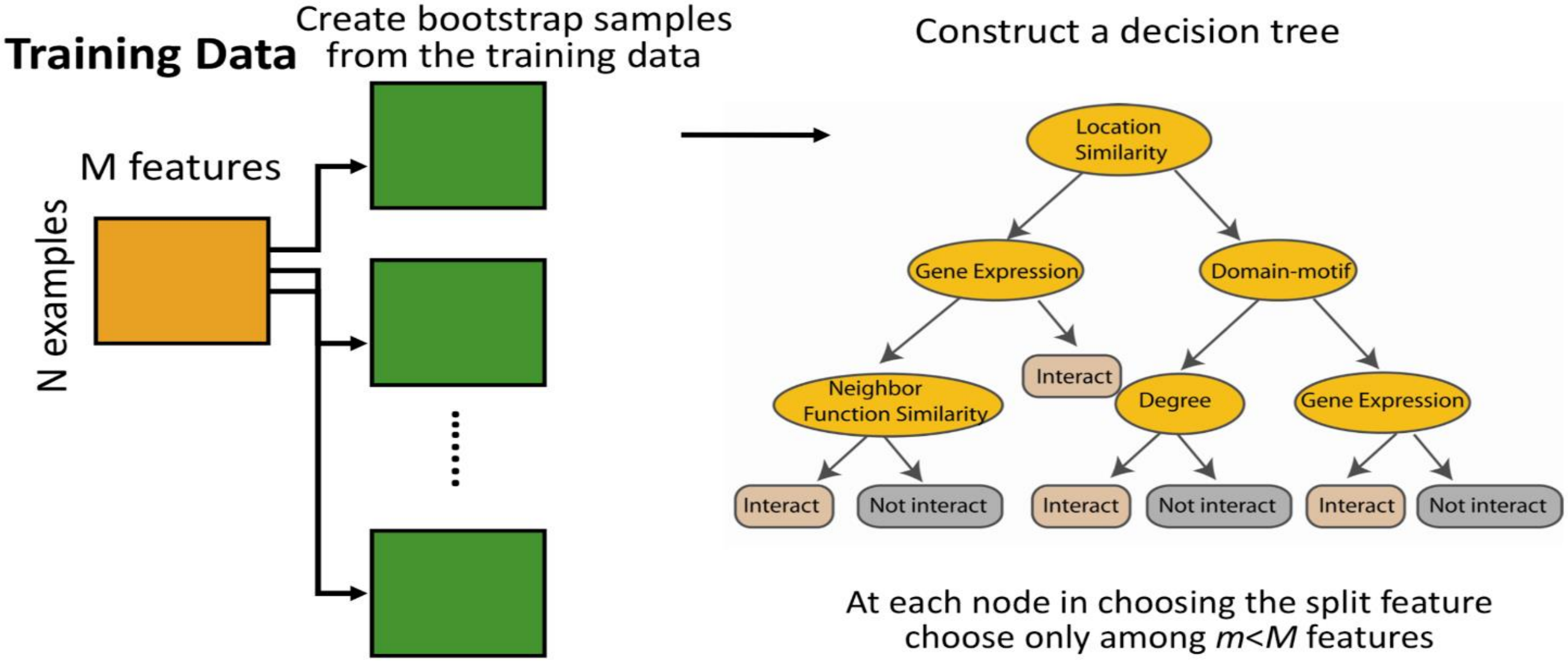
- Boosting: random sampling with replacement over weighted data; **serial**, need to keep track of learner’s errors. AdaBoost, XGBoost, GradientBoost, etc

Bootstrap

Randomly draw datasets with replacement from the training data



Random forest Classifier


Training Data
Create bootstrap samples from the training data
M features
N examples
Construct a decision tree
Location Similarity
Gene Expression
Domain-motif
Neighbor Function Similarity
Interact
Degree
Gene Expression
Interact
Not interact
Interact
Not interact
Interact
Not interact
At each node in choosing the split feature choose only among *m*<*M* features

Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Introduce two sources of randomness: “Bagging” and “Random input vectors”
    - Bagging method: each tree is grown using a bootstrap sample of training data
    - Random vector method: At each node, best split is chosen from a random sample of m attributes instead of all attributes

Random forest: second randomized through predictor subset

- Bagging alone utilizes the same full set of features (predictors) to determine each split
- However random forest applies another randomness: namely by selecting a random **subset** of the features for each split
- Number of predictors to try at each split is known as $m_{try.}$
  - typically $m = \sqrt{k}$ (classification) or $m = k/3$ (regression) works quite well. RF is not overly sensitive to m_{try}

Random forest: Practical Considerations

- Splits are chosen according to a purity measure:
  - e.g. squared error (regression), Gini index or deviance (classification)

- How to select N_{trees}?
  - Build trees until the error no longer decreases

- How to select m_{try} ?
  - Try the recommended defaults, half of them and twice them, and pick the best

Random forest algorithm

- Let N_{trees} be the number of trees to build
- for each of N_{trees} iterations
  1. Select a new bootstrap sample from training set
  2. Grow an un-pruned tree on this bootstrap.
  3. At each internal node, randomly select m_{try} features and determine the best split using only these features.
  4. Do not perform cost complexity pruning. Save tree as is, along side those built thus far.
- Output overall prediction as the average response (regression) or majority vote (classification) from all individually trained trees