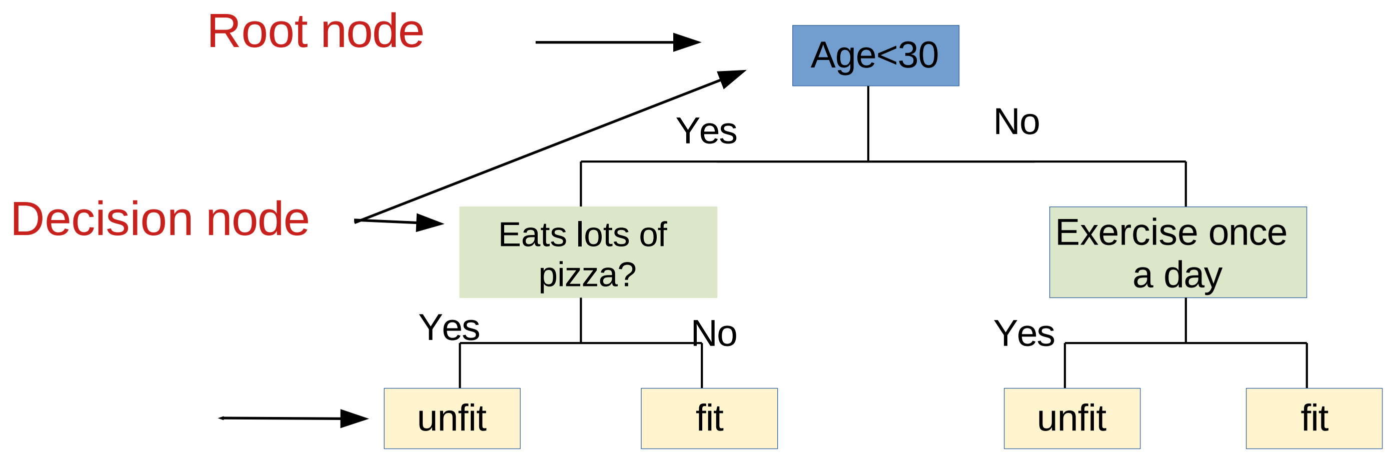
Decision Tree and Random Forest

Dr Gopal Jamnal

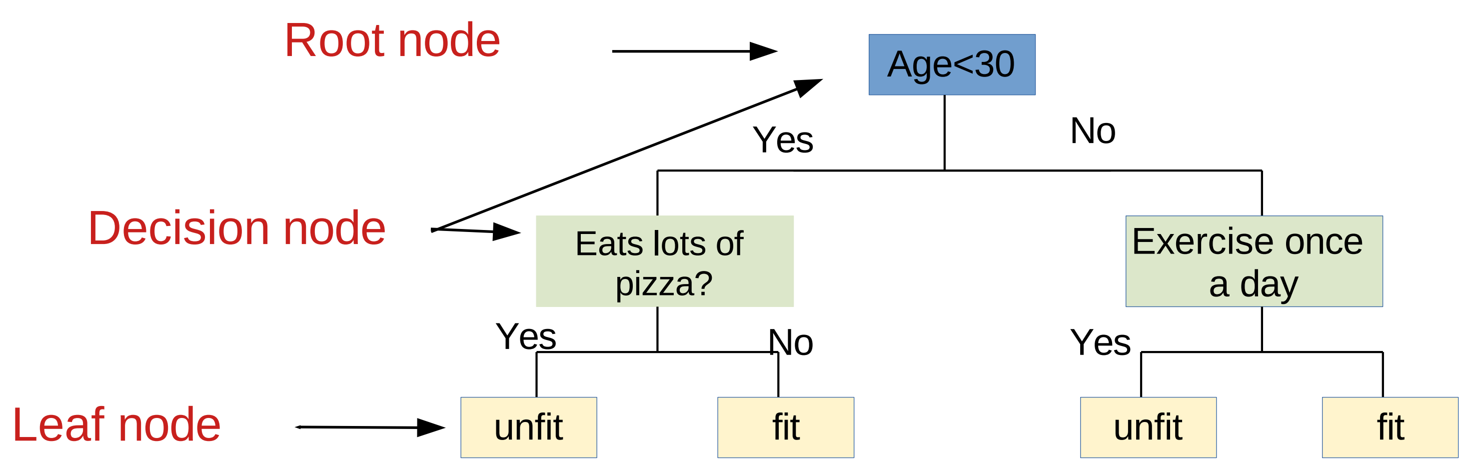
Decision Trees

- **A decision tree** is a **non-parametric supervised learning** algorithm.
- Decision trees are a type of model used for both **classification and regression problems.**
- Decision tree builds models in the **form of a tree structure**.
- The model comprises a series of **logical decisions** with **decision nodes.**
- **Decision nodes** indicate a decision to be made on a variable.



Decision Trees

- The tree is terminated by **leaf nodes** that denote the result of following a combination of decision.
- Data to be classified begin at the **root node** is passed it is passed through the various decisions in the tree according to the values of its features.
- The **decision/prediction** is determined by the **leaf nodes**.
- Decision trees are built using a heuristic called **recursive partitioning** (also known as **divide and conquer**).



Decision Tree Algorithm

Top down construction of Decision Tree

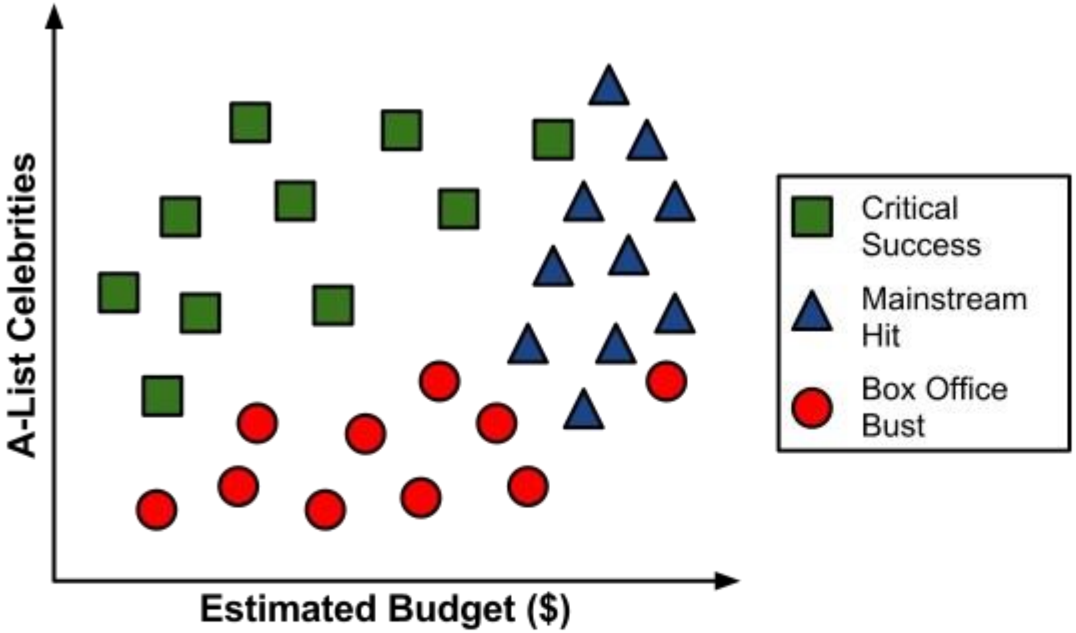
1. Start with **root node** that includes all data
2. **Select best** node to split
3. Check the stopping criteria: if **yes** stop otherwise go back to **step 2**
4. Tree **pruning** - reduces the size of decision trees

Stopping Criteria

- homogeneous values within a node
- length/depth of a tree

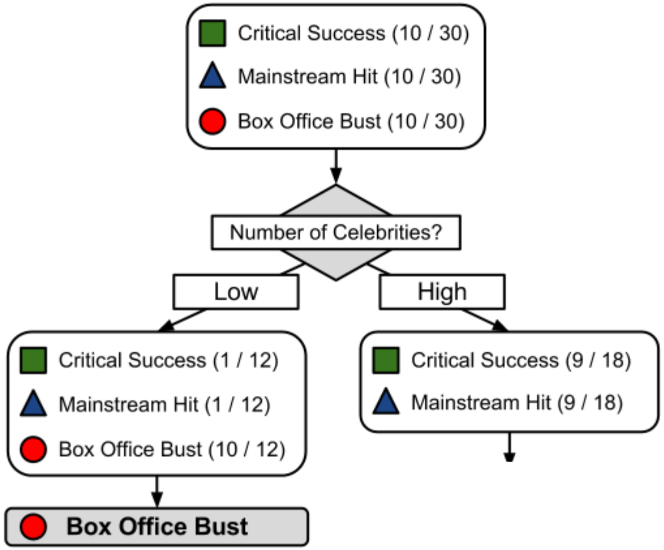
Divide and Conquer for Decision Trees

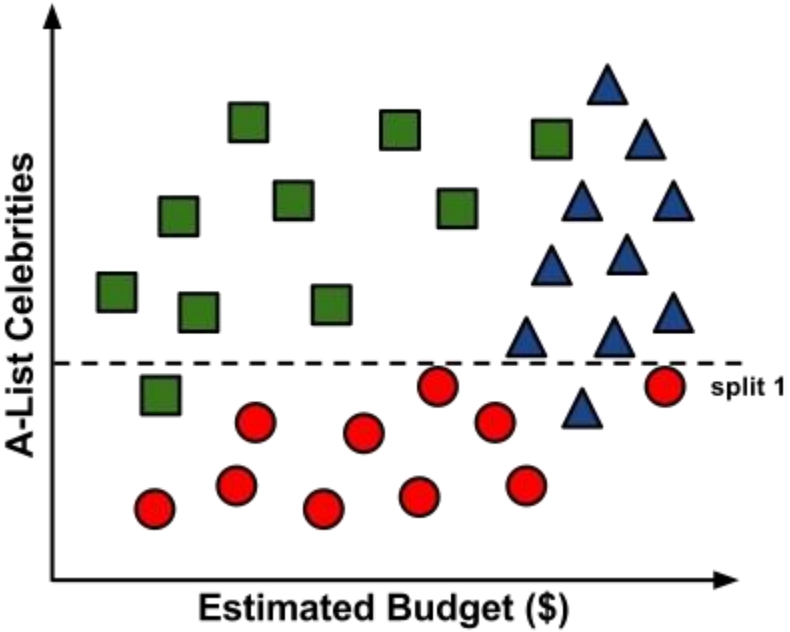
Develop a decision tree algorithm to predict whether a potential movie would fall into one of three categories: mainstream hit, critic's choice, or box office bust.



Divide and Conquer for Decision Trees

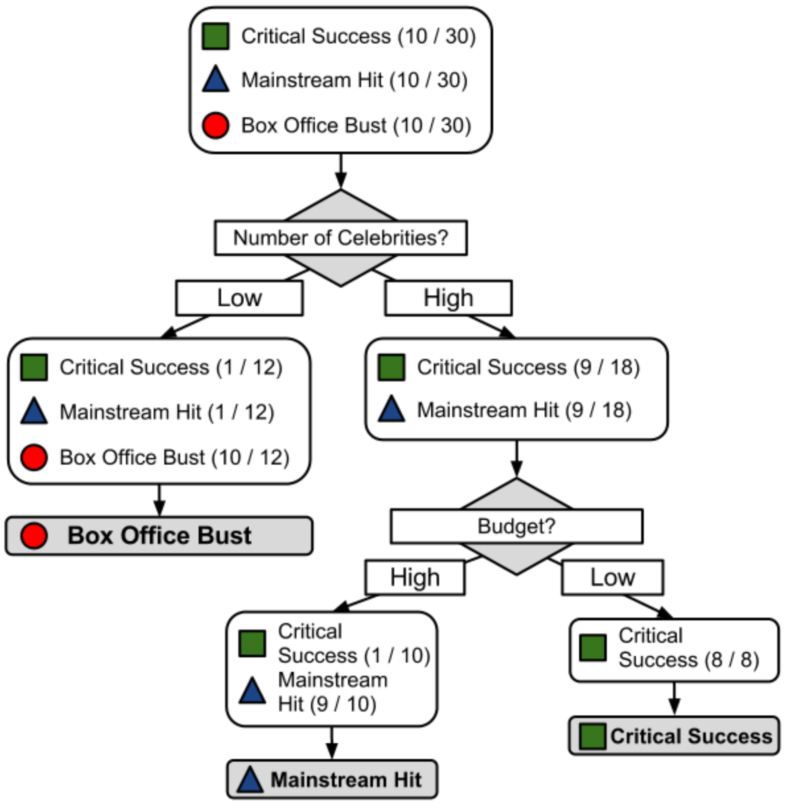
Step 1: Split the feature indicating the **number of celebrities,** partitioning the movies into groups with and without a low number of A-list stars:

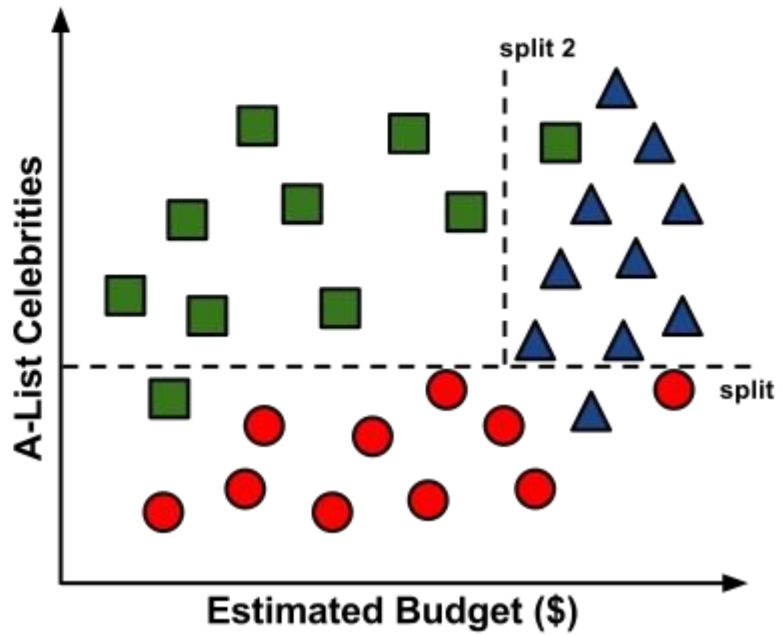




Divide and Conquer for Decision Trees

Step 2: Among the group of movies with a larger number of celebrities, split between movies with and without a **high budget:**





How to chose variable for a split?

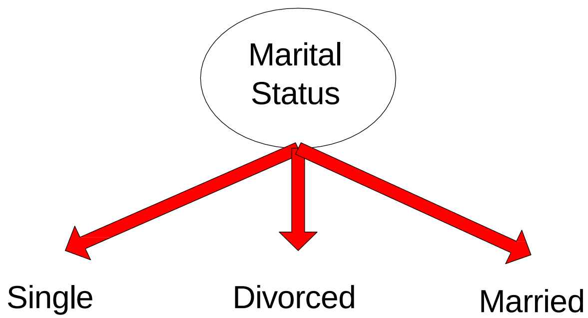
Design Issues of Decision Tree Development

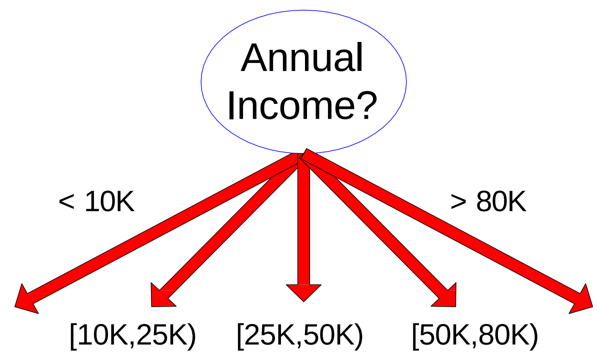
- How to chose the best split?
  - Find the test condition (depending the data types)?
  - Measure goodness of the test (the best slit)?

- When do we stop?
  - Stop splitting if all records belong to the same class or have identical variable values
  - Early determination

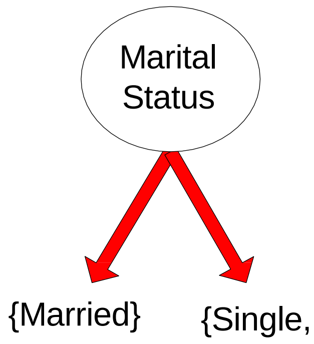
Choosing the Best Split

- **Multi**-way split - Use as many partitions as distinct values

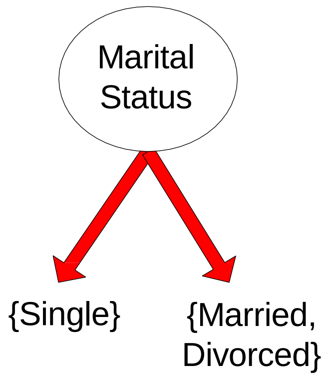


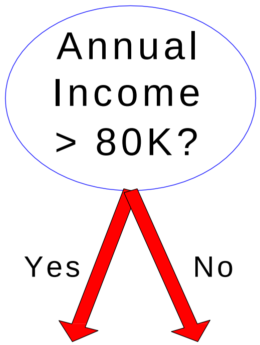


- **Binary** split - Divides values into two subsets, Preserve order property among variable values (for ordinal data)



OR





Information Entropy

How much variance the data has.

- A dataset of only blues would have very **low** (in fact, zero) entropy. A set of **only one class** is extremely predictable. This would have **low** entropy.

- A dataset of mixed blues, greens, and reds would have relatively **high** entropy. A set of many **mixed classes** is unpredictable. This would have **high** entropy.

$$E = -\sum_{i}^{C} p_i \log_2 p_i$$

Where P_i is the probability of randomly picking an element of class i (i.e. the proportion of the dataset made up of class i)

Choosing the Best Split

2 popular Criteria/methods

1. **Information Gain :** calculated for a split by subtracting the weighted entropies of each branch from the original entropy. the best split is chosen by **maximizing Information Gain**

$$Information\ Gain = Entropy_{parent} - Entropy_{children}$$

1. **Gini Index** : measures the probability for a random instance being misclassified when chosen randomly. The **lower the Gini Index, the better** the lower the likelihood of misclassification

$$Gini = 1 - \sum_{i=1}^{n}(p_i)^2$$

Information Gain

- Information Gain, measures the reduction in entropy , **Compute** impurity measure (P) **before splitting**, Compute impurity measure (M) **after splitting**

Compute impurity measure of each child node

M is the weighted impurity of child nodes

InfoGain (f) = P – M

Choose the feature test condition that produces the highest **information gain**

- Calculated for a split by **subtracting the weighted entropies of each branch from the original entropy**. the best split is chosen by **maximizing Information Gain**

$$Information\,Gain = Entropy_{\,parent} - Entropy_{\,children}$$

Gini Index

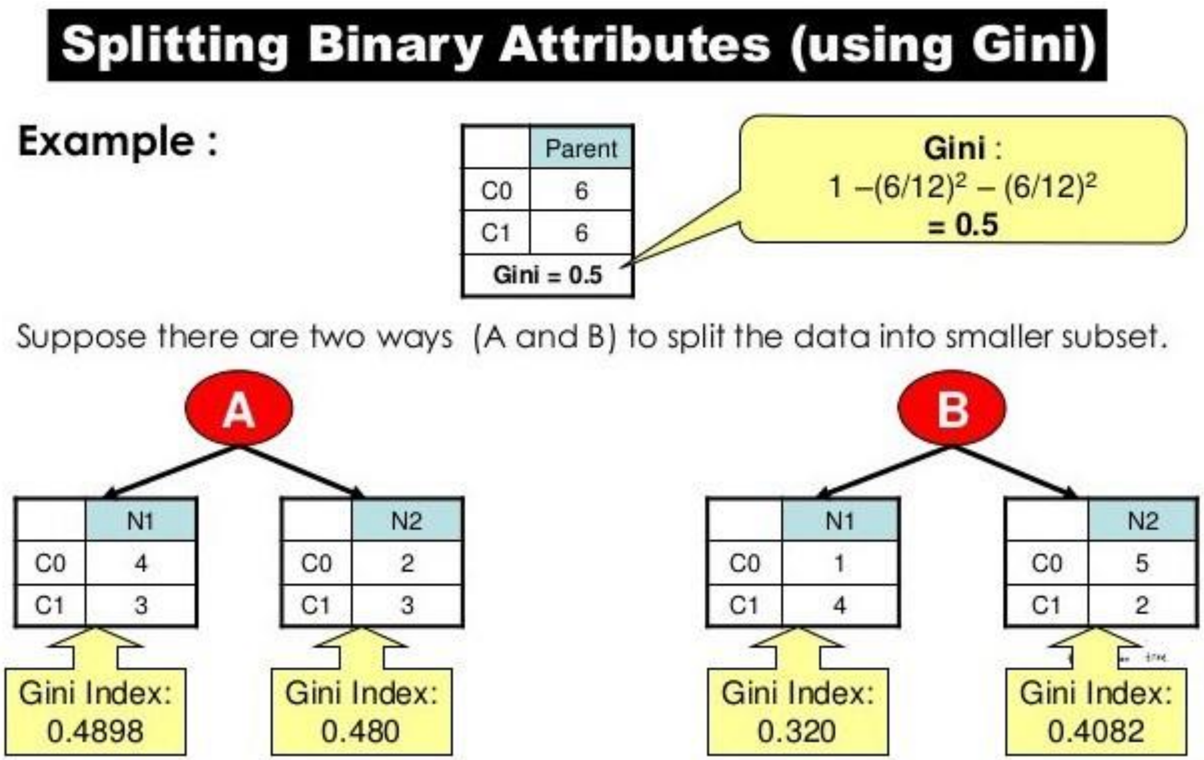
- The Gini Index or Impurity, measures the **probability for a random instance being misclassified when chosen randomly**.
- Gini Index measures the impurity of the data, with **a lower Gini Index indicating better** subsets.
- Split node is calculated by **summing the squared probabilities** of each class being chosen **times** the **probability of a misclassification** for that class.
- **Slightly faster** to compute since it doesn't involve calculating **logarithms**.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

where,

$\mathbf{P_i}$ is the probability of an object being classified to a particular class.
$\boldsymbol{n}$ is the number of classes in target variable

Best Split Using GINI



Gini(N1)= 1-(4/7)²-(3/7)²= 0.4898

$Gini_{split\ A}$=7/12 * 0.489 +5/12 *0.48 =0.485

$Gini_{split\ B}$=5/12 * 0.320 +7/12 *0.408 =0.371

InfoGain=0.5-0.485=0.015

InfoGain=0.5-0.371= 0.129

Best Split Using GINI

Splitting Binary Attributes (using Gini)

Example :

	Parent
C0	6
C1	6
Gini = 0.5	

Gini : $1 - (6/12)^2 - (6/12)^2$ **= 0.5**

Suppose there are two ways (A and B) to split the data into smaller subset.

$Gini(N1) = 1-(4/7)^2-(3/7)^2 = 0.4898$

A

	N1
C0	4
C1	3

Gini Index: 0.4898

	N2
C0	2
C1	3

Gini Index: 0.480

$Gini_{split\ A} = 7/12 * 0.489 + 5/12 * 0.48 = 0.485$

InfoGain=0.5-0.485=0.015

B

	N1
C0	1
C1	4

Gini Index: 0.320

	N2
C0	5
C1	2

Gini Index: 0.4082

$Gini_{split\ B} = 5/12 * 0.320 + 7/12 * 0.408 = 0.371$

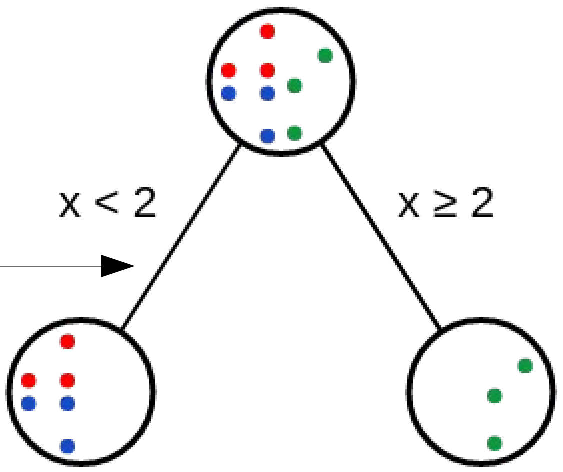
InfoGain=0.5-0.371= 0.129

$Gini_{splitB} < Gini_{slipt\ A}$

B has lower impurity

Best Split Value

Split	Left Branch	Right Branch	Gini Gain
$x = 0.4$	🔴	🔵🔵🔵🔴🔴🟢🟢🟢	0.083
$x = 0.8$	🔵🔴	🔵🔵🔴🔴🟢🟢🟢	0.048
$x = 1.1$	🔵🔵🔴🔴	🔵🔴🟢🟢🟢	0.133
$x = 1.3$	🔵🔵🔴🔴🔴	🔵🟢🟢🟢	0.233
$x = 2$	🔵🔵🔵🔴🔴🔴	🟢🟢🟢	0.333
$x = 2.4$	🔵🔵🔵🔴🔴🔴🟢	🟢🟢	0.191
$x = 2.8$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083
$y = 0.8$	🔵	🔵🔵🔴🔴🔴🟢🟢🟢	0.083
$y = 1.2$	🔵🔵🟢	🔵🔴🔴🔴🟢🟢	0.111
$y = 1.8$	🔵🔵🔵🟢	🔴🔴🔴🟢🟢	0.233
$y = 2.1$	🔵🔵🔵🟢🟢	🔴🔴🔴🟢	0.233
$y = 2.4$	🔵🔵🔵🔴🟢🟢	🔴🔴🟢	0.111
$y = 2.7$	🔵🔵🔵🔴🔴🟢🟢	🔴🟢	0.048
$y = 2.9$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083

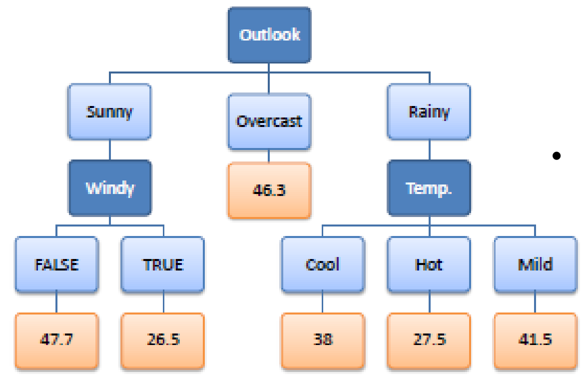


Prediction

Regression

Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Predictors: Outlook, Temp., Humidity, Windy. Target: Hours Played.



Prediction:

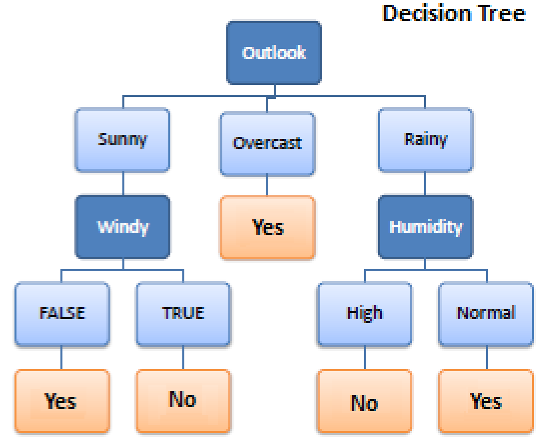
- Value of target variable for a given record - if leaf node with single item
- **Mean of target** value for all records in leaf node

Classification

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Predictors: Outlook, Temp., Humidity, Windy. Target: Play Golf.

Decision Tree

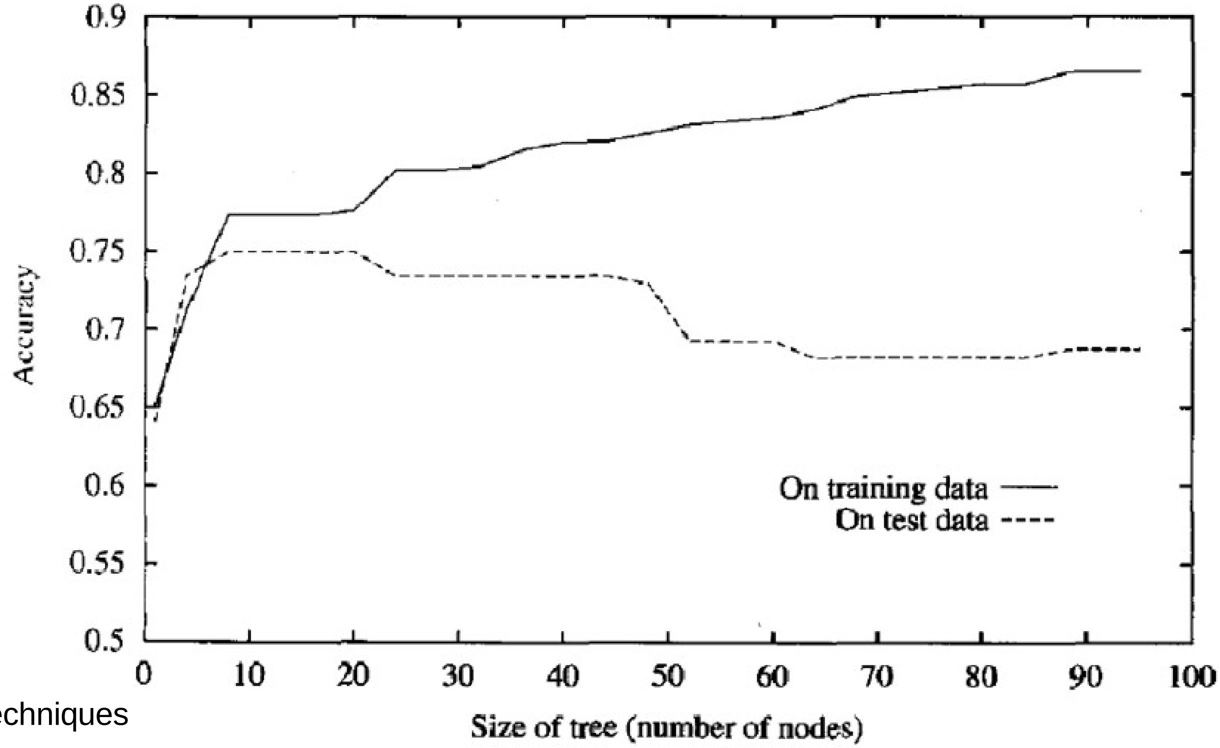


Prediction:

- Value of target variable for a given record - if leaf node with single item
- **Majority voting** for target value among all records in leaf node

Pruning the Decision Tree

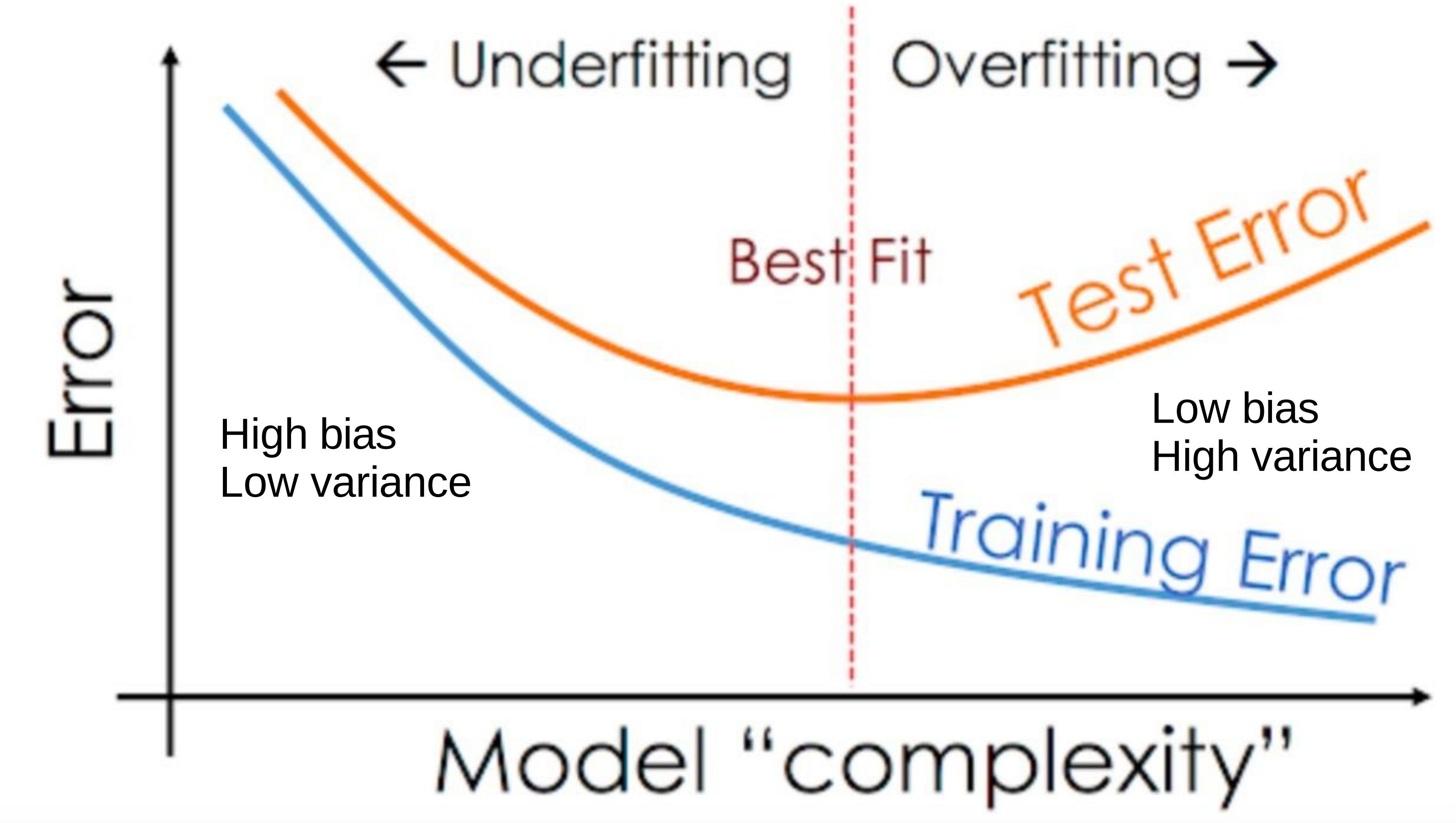
- A decision tree can continue to grow indefinitely.
- If the tree grows large, many of the decisions it makes will be overly specific and the model will have been **overfitted** to the training data.



Pruning the Decision Tree

- A decision tree can continue to grow indefinitely.
- If the tree grows large, many of the decisions it makes will be overly specific and the model will have been **overfitted** to the training data.
- The **process of pruning** a decision tree involves reducing its size such that it generalizes better to unseen data:
  - **Pre-pruning –** (early stopping) - stop to grow tree when it reaches a certain number of decisions or if the decision nodes contain only a small number of examples.
  - **Post-pruning** – (backward pruning) grow the tree to the largest extend then using pruning criteria based on the error rates or information gain at the nodes to reduce the size of the tree to a more appropriate level.

Underfitting vs Overfitting



Strength of Decision Trees

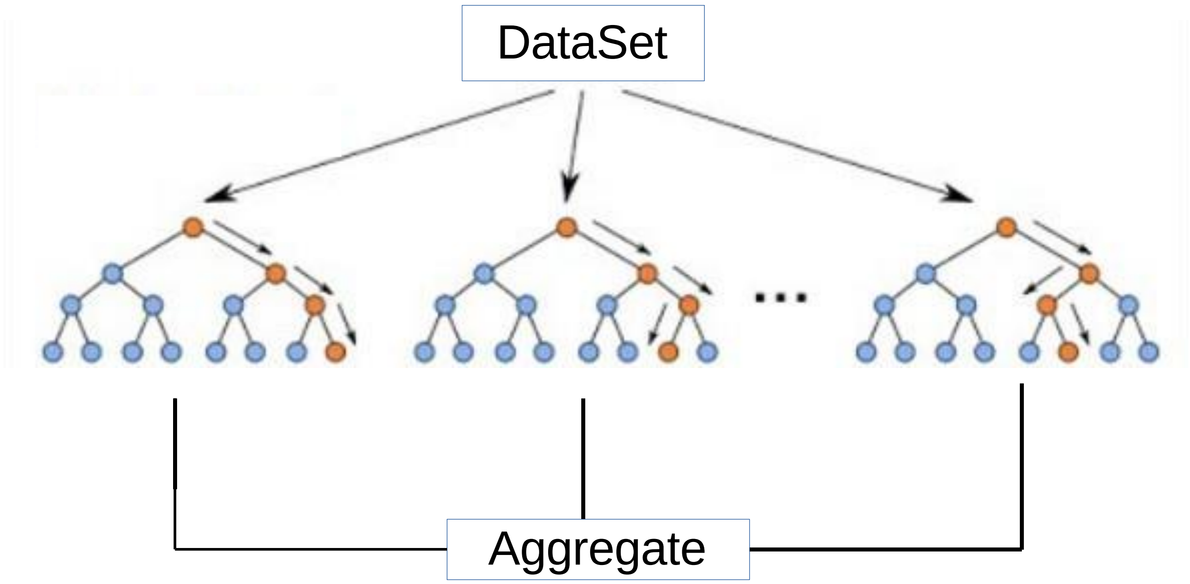
- Inexpensive to construct.
- Extremely fast at classifying unknown records.
- Easy to interpret for small-sized trees.
- Accuracy comparable to other classification techniques for many simple data sets.
- Excludes unimportant features.

Weaknesses of Decision Trees

- 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 training data can result in large changes to decision logic.
- Large trees can be difficult to interpret and the decisions they make may seem counter-intuitive

Random Forest Model

Random forests or random decision forests are an **ensemble learning** method for **classification, regression** and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

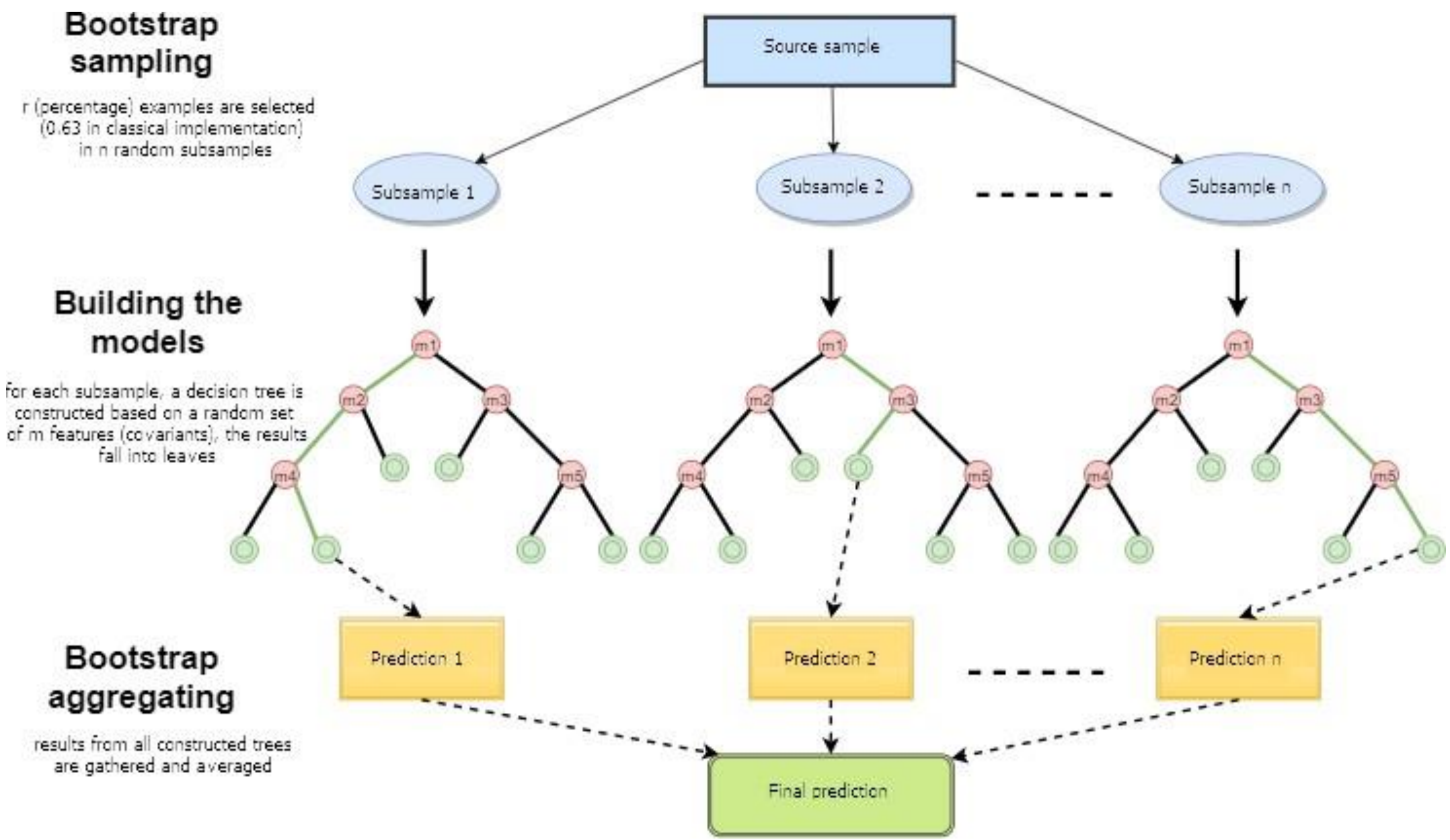


Random Forest (RF) Algorithm

A random forest (RF) model introduced by Breiman (2001) is a collection of tree predictors. Each tree is grown according to the following procedure:

- **the bootstrap phase:** select randomly a subset of the training dataset and testing **out-of-bag (OOB)** set used to estimate the RF's goodness-of-fit.
- **the growing phase:** grow the tree by splitting the local training set at each node according to the value of one variable from a randomly selected subset of variables (the best split) using classification and regression tree (CART)
- each tree is grown to the **largest extent possible**. There is no pruning.

Random Forest Aggregation



Random Forest Aggregation

Classification

- Single tree prediction:
  - A class of the item in a leaf node (if single element)
  - Majority voting in a leaf node
- Average tree prediction among the forest.
  - Class: Majority voting
  - Prob: a ratio of trees voting for a given class

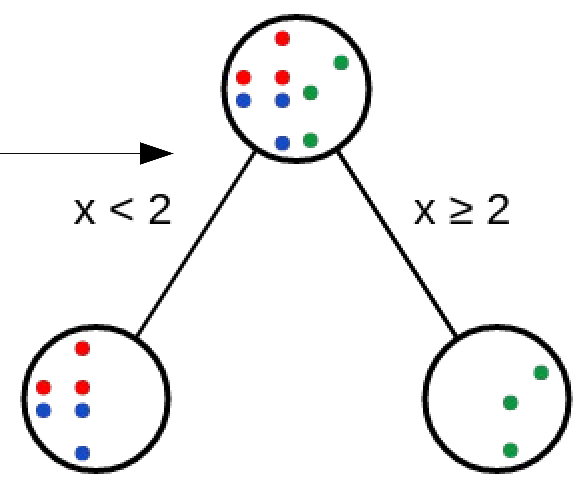
Regression

- Single tree prediction:
  - A value of the item in a leaf node (if single element)
  - Average of values in a leaf node
- Average tree prediction among the forest.

Random Forest Best Split

- Generate a split based on a value of a given variable
- Repeat for all variables selected for best split
- Select the variable and value with highest Gini Gain

Split	Left Branch	Right Branch	Gini Gain
$x = 0.4$	🔴	🔵🔵🔵🔴🔴🟢🟢🟢	0.083
$x = 0.8$	🔵🔴	🔵🔵🔴🔴🟢🟢🟢	0.048
$x = 1.1$	🔵🔵🔴🔴	🔵🔴🟢🟢🟢	0.133
$x = 1.3$	🔵🔵🔴🔴🔴	🔵🟢🟢🟢	0.233
$x = 2$	**🔵🔵🔵🔴🔴🔴**	**🟢🟢🟢**	**0.333**
$x = 2.4$	🔵🔵🔵🔴🔴🔴🟢	🟢🟢	0.191
$x = 2.8$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083
$y = 0.8$	🔵	🔵🔵🔴🔴🔴🟢🟢🟢	0.083
$y = 1.2$	🔵🔵🟢	🔵🔴🔴🔴🟢🟢	0.111
$y = 1.8$	🔵🔵🔵🟢	🔴🔴🔴🟢🟢	0.233
$y = 2.1$	🔵🔵🔵🟢🟢	🔴🔴🔴🟢	0.233
$y = 2.4$	🔵🔵🔵🔴🟢🟢	🔴🔴🟢	0.111
$y = 2.7$	🔵🔵🔵🔴🔴🟢🟢	🔴🟢	0.048
$y = 2.9$	🔵🔵🔵🔴🔴🔴🟢🟢	🟢	0.083



Random Forest

- Strengths:
  - **Robust to outliers.**
  - Works well with **non-linear data.**
  - Lower risk of **overfitting.**
  - Runs efficiently on a **large dataset**.
  - Better **accuracy** than other classification algorithms.
- Weaknesses:
  - Random forests are found to be biased while dealing with **categorical variables.**
  - **Slow** Training.
  - Not suitable for linear methods with a lot of **sparse features**

References

- B.Lantz “Machine Learning with R – chapter 5
- P.Tan “Introduction to Data Mining – chapter 3 and 4.10
- N.Ye. Data Mining – chapter 4
- Random Forest – Leo Breiman
  https://www.stat.berkeley.edu/~breiman/RandomForests/